C

Cache Manager

► Buffer Manager

Cache Performance

► Performance Analysis of Transaction Processing Systems

Cache-Aware Query Processing

► Cache-Conscious Query Processing

Cache-Conscious Query Processing

Kenneth A. Ross
Columbia University, New York, NY, USA

Synonyms

Cache-aware query processing; Cache-sensitive query processing

Definition

Query processing algorithms are designed to efficiently exploit the available cache units in the memory hierarchy. Cache-conscious algorithms typically employ knowledge of architectural parameters such as cache size and latency. This knowledge can be used to ensure that the algorithms have good temporal and/or spatial locality on the target platform.

Historical Background

Between 1980 and 2005, processing speeds improved by roughly four orders of magnitude, while memory speeds improved by less than a single order of magnitude. As a result, it is common (at the time of writing) for data accesses to RAM to require several hundred CPU cycles to resolve. Many database workloads have shifted from being I/O bound to being memory/CPU-bound as the amount of memory per machine has been increasing. For such workloads, improving the locality of data-intensive operations can have a direct impact on the system's overall performance.

Foundations

A cache is a hardware unit that speeds up access to data. Several cache units may be present at various levels of the memory hierarchy, depending on the processor architecture. For example, a processor may have a small but fast Level-1 (L1) cache for data, and another L1 cache for instructions. The same processor may have a larger but slower L2 cache storing both data and instructions. Some processors may even have an L3 cache. On multicore processors, the lower level caches may be shared among groups of cores.

Some initial analysis would typically be performed to determine the performance characteristics of a workload. For example, Ailamaki et al. [1] used hardware performance counters to demonstrate that several commercial systems were, at that time, suffering many L2 data cache misses and L1 instruction cache misses. Based on such observations, one can determine that the L2 data cache and L1 instruction cache are targets for performance tuning.

If the operating system does not provide direct access to system parameters such as the cache size, a database system can run a calibration test to estimate the relevant parameters [13].

To get good cache performance, algorithm designers typically utilize one or more of the following general approaches:

- Improve spatial locality, so that data items that are often accessed together are in the same cache lines.
- Improve temporal locality for data, so that after an initial cache miss, subsequent data item accesses occur while the item is still cache-resident.

- Improve temporal locality for instructions, so that code that needs to be applied to many data items is executed many times while the instructions reside in the cache.
- Hide the latency. Latency can be hidden in several ways. If the data access pattern is predictable, prefetching data elements into the cache can overlap the memory latency with other work. On architectures that support multiple simultaneous outstanding memory requests, cache miss latencies can be overlapped with one another.
- Sample the data to predict the cache behavior, and choose an algorithm accordingly.

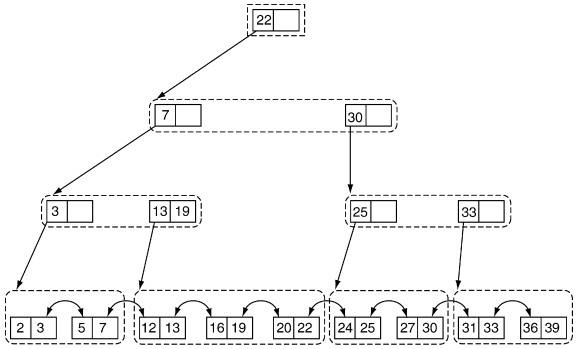
Examples of each of these approaches are given below.

Spatial Locality

In many query-processing contexts, only a few columns for a table are needed. In such cases, it pays to organize the table column-wise, so that column values from consecutive records are contiguous. Cache lines then contain many useful data elements. A row-wise organization would require more cache-line accesses, since each cache line would contain some data from unneeded columns. Examples of systems with column-wise storage are Sybase IQ [12], MonetDB [3], and C-Store [19]. The PAX storage model [6] allows for column-wise storage within each disk page. The main advantage of such an approach is that existing page-oriented database systems can improve cache behavior with limited changes to the whole system.

Chilimbi et al. [7] advocate placing multiple levels of a binary tree within a cache line, to reduce the number of cache misses per traversal. Chilimbi et al. also use cache coloring to place frequently accessed items in certain ranges of physical memory. The idea is to reduce the number of conflict misses by making sure that the low order bits of the addresses of certain items cannot be the same.

To reduce the number of cache lines needed to search for an item in an index, Rao and Ross proposed CSS-trees [16] and CSB+-trees [17]. CSS-trees eliminate pointers; a node contains only keys. Nodes are of fixed size, typically one cache line, aligned to the cache line boundaries. Nodes are stored in an array in such a way that the children of a node can be determined using simple arithmetic operations, making pointers unnecessary. CSB+-trees extend this idea, allowing just one pointer per node and requiring that all sibling nodes be contiguous. CSB+-trees have better update performance than CSS-trees, while retaining almost all of the cache-efficiency. The diagram below shows a CSB+-Tree of Order 1. Note that each node has only one child pointer and that each node's children are allocated contiguously.



Several other ways to compress B+-tree nodes for cache performance, such as key compression and key-prefix truncation, are discussed in [11].

Temporal Locality

Blocking is a general technique for ensuring temporal locality. Data is processed in cache-sized units, so that all data within the block stays cache-resident. The blocks are then combined in a later phase. Alpha-Sort [14] is an example of such a method: cache-sized units of input data are read and quick-sorted into runs. These runs are merged in a later phase. Padmanabhan et al. [15] modified a commercial database system to pass data between certain operators in blocks, and demonstrated improved cache behavior.

Buffering is a related strategy to improve temporal locality. Zhou and Ross [20] propose buffering to speed up bulk B-tree index lookups. By sending probes only one level at a time through the tree, in batches, one can amortize the cost of reading an index node over many probes. The savings in data cache misses usually outweigh the extra cost of reading and writing to intermediate buffers. Zhou and Ross also examine the code size of database operators, and propose to buffer data to ensure that the footprint of the active code is smaller than the size of the L1 instruction cache [21]. Again, the savings in instruction cache misses usually outweigh the cost of buffering.

Partitioning the data into cache-sized units for later processing is the dual of blocking. Examples include the partitioned hash join [18] and radix-join [3].

When multiple processors, or multiple threads within a single processor access a shared cache, cache interference can result. Even if each individual thread is cache-conscious, the total cache resources may be

insufficient for all threads, and cache thrashing may result. To counter this interference, one could design cooperative threads that work together on a common task using common cache-conscious data structures. Multithreaded join operators [10,22] and aggregation operators [8] have been proposed.

Many divide-and-conquer style algorithms generate temporal locality at recursively smaller granularities. Such algorithms have been termed *cache oblivious* because they can achieve locality at multiple levels of the memory hierarchy without explicit knowledge of the cache parameters [9].

Prefetching

Prefetching involves reading data into the cache ahead of when it is to be used. When access patterns can be predicted in advance, and when memory bandwidth is not saturated, prefetching can effectively hide the memory latency. Some hardware platforms automatically recognize certain access patterns, such as regular fixed-stride access to memory. The hardware then automatically prefetches ahead in the access sequence. For access patterns that are not so easily recognized, or for hardware platforms that do not support hardware prefetching, one can explicitly prefetch memory locations using software.

Chen et al. [6] show how to prefetch parts of a B+-tree node or CSB+-tree node to get a bigger effective node size. For example, if the memory system can support n outstanding memory requests, then a node consisting of n cache lines could be retrieved in only slightly more time than a single cache line. Since wider nodes result in shallower trees, the optimal node size might be several cache lines wide.

Chen et al. [4] use prefetching to speed up hash joins. The internal steps for processing records are divided into stages. A memory access is typically required between stages. Stages for multiple records are scheduled so that while data for a forthcoming operation is being prefetched, useful work is being performed on other records.

Zhou et al. [22] define the notion of a work-ahead set, a data structure that describes a memory location and a computation stage for some data-intensive operation. One thread of a two-threaded system is devoted purely to prefetching the data into the cache, while the other thread does the algorithmic work.

Sampling

Inspector joins sample the data during an initial partitioning phase [5]. This information is used to accelerate a cache-optimized join algorithm for processing the partitions. Cieslewicz et al. [8] sample a stream of tuples for aggregation to estimate (among other things) the locality of reference of group-by values. Based on that information, an appropriate aggregation algorithm is chosen for the remainder of the stream.

Key Applications

Data intensive operators such as sorts, joins, aggregates, and index lookups, form the "assembly language" into which complex queries are compiled. By making these operators as efficient as possible on modern hardware, all database system users can effectively exploit the available resources.

Future Directions

Future processors are likely to scale by placing many cores on a chip, with only a modest increase in clock frequency. As a result, the amount of cache memory per processor may actually decrease over time, making cache optimization even more critical. For chips with shared caches, interference between cores will be a significant performance hazard. While locality is good for cache behavior, it can be bad for concurrency due to hot-spots of contention [8]. Cache performance will need to be considered together with parallelism to find appropriate performance trade-offs.

Cross-references

- ▶ Architecture-Conscious Database System
- ▶ Cache-Conscious Transaction Processing

Recommended Reading

1. Ailamaki A., Dewitt D.J., Hill M.D., and Wood D.A. DBMSs on a modern processor: where does time go? In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 266–277.
2. Ailamaki A., DeWitt D.J., Hill M.D., and Skounakis M. Weaving relations for cache performance. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 169–180.
3. Boncz P.A., Manegold S., and Kersten M.L. Database architecture optimized for the new bottleneck: memory access. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 54–65.
4. Chen S., Ailamaki A., Gibbons P.B., and Mowry T.C. Improving hash join performance through prefetching. In Proc. 20th Int. Conf. on Data Engineering, 2004, pp. 116–127.
5. Chen S. et al. Inspector joins. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 817–828.
6. Chen S., Gibbons P.B., and Mowry T.C. Improving index performance through prefetching. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 235–246.
7. Chilimbi T.M., Hill M.D., and Larus J.R. Cache-conscious structure layout. In Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation, 1999, pp. 1–12.
8. Cieslewicz J. and Ross K.A. Adaptive aggregation on chip multiprocessors. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 339–350.

9. Frigo M., Leiserson C.E., Prokop H., and Ramachandran S. Cache-oblivious algorithms. In Proc. 40th Annual Symp. on Foundations of Computer Science, 1999, pp. 285–298.
10. Garcia P. and Korth H. Database hash-join algorithms on multithreaded computer architectures. In Proc. 3rd Conf. on Computing Frontiers, 2006, pp. 241–251.
11. Graefe G. and Larson P. B-tree indexes and CPU caches. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 349–358.
12. MacNicol R. and French B. Sybase IQ multiplex – designed for analytics. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 1227–1230.
13. Manegold S., Boncz P.A., and Kersten M.L. What happens during a join? dissecting CPU and memory optimization Effects. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 339–350.
14. Nyberg C., Barclay T., Cvetanovic Z., Gray J., and Lomet D.B. AlphaSort: a cache-sensitive parallel external sort. VLDB J., 4 (4):603–627, 1995.
15. Padmanabhan S., Malkemus T., Agarwal R., and Jhingran A. Block oriented processing of relational database operations in modern computer architectures. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 567–574.
16. Rao J. and Ross K.A. Cache conscious indexing for decision-support in main memory. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 78–89.
17. Rao J. and Ross K.A. Making B+ trees cache conscious in main memory. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 475–486.
18. Shatdal A., Kant C., and Naughton J.F. Cache conscious algorithms for relational query processing. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 510–521.
19. Stonebraker M., Abadi D.J., Batkin A., Chen X., Cherniack M., Ferreira M., Lau E., Lin A., Madden S., O'Neil E.J., O'Neil P.E., Rasin A., Tran N., and Zdonik S.B. C-Store: a column-oriented DBMS. In Proc. 31th Int. Conf. on Very Large Data Bases, 2005, pp. 553–654.
20. Zhou J. and Ross K.A. Buffering accesses to memory-resident index structures. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 405–416.
21. Zhou J. and Ross K.A. Buffering database operations for enhanced instruction cache performance. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 191–202.
22. Zhou J., Cieslewicz J., Ross K., and Shah M. Improving database performance on simultaneous multithreading processors. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 49–60.

Cache-Sensitive Query Processing

► Cache-Conscious Query Processing

Calculus Expression

► Comprehensions

Calendar

Christian S. Jensen[1], Richard Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Definition

A *calendar* provides a human interpretation of time. As such, calendars ascribe meaning to temporal values such that the particular meaning or interpretation provided is relevant to its users. In particular, calendars determine the mapping between human-meaningful time values and an underlying time line.

Key Points

Calendars are most often cyclic, allowing human-meaningful time values to be expressed succinctly. For example, dates in the common Gregorian calendar may be expressed in the form <*month*, *day*, *year*> where the month and day fields cycle as time passes.

The concept of calendar defined here subsumes commonly used calendars such as the Gregorian calendar, the Hebrew calendar, and the Lunar calendar, though the given definition is much more general. This usage is consistent with the conventional English meaning of the word.

Dershowitz and Reingold's book presents complete algorithms for fourteen prominent calendars: the present civil calendar (Gregorian), the recent ISO commercial calendar, the old civil calendar (Julian), the Coptic an Ethiopic calendars, the Islamic (Muslim) calendar, the modern Persian (solar) calendar, the Bahá'í calendar, the Hebrew (Jewish) calendar, the Mayan calendars, the French Revolutionary calendar, the Chinese calendar, and both the old (mean) and new (true) Hindu (Indian) calendars. One could also envision more specific calendars, such as an academic calendar particular to a school, or a fiscal calendar particular to a company.

Cross-references

► Calendric System
► SQL
► Temporal Database

Recommended Reading

1. Bettini C., Dyreson C.E., Evans W.S., Snodgrass R.T., and Wang X.S. A Glossary of Time Granularity Concepts. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS, vol. 1399, Springer, Berlin, pp. 406–413, 1998.

2. Dershowitz N. and Reingold E.M. Calendrical Calculations, Cambridge, 1977.
3. Jensen C.S. and Dyreson C.E. (eds.). Böhlen M., Clifford J., Elmasri R., Gadia S.K., Grandi F., Hayes P., Jajodia S., Käfer W., Kline N., Lorentzos N., Mitsopoulos Y., Montanari A., Nonen D., Peressi E., Pernici B., Roddick J.F., Sarda N.L., Scalas M.R., Segev A., Snodgrass R.T., Soo M.D., Tansel A., Tiberio R. and Wiederhold G. A Consensus Glossary of Temporal Database Concepts – February 1998 Version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS, vol. 1399, Springer, Berlin, 1998, pp. 367–405.
4. Urgun B., Dyreson C.E., Snodgrass R.T., Miller J.K., Kline N., Soo M.D., and Jensen C.S. Integrating Multiple Calendars using τZaman. Software Pract. Exper., 37(3):267–308, 2007.

Calendric System

Curtis E. Dyreson[1], Christian S. Jensen[2], Richard Snodgrass[3]
[1]Utah State University, Logan, UT, USA
[2]Aalborg University, Aalborg, Denmark
[3]University of Arizona, Tucson, AZ, USA

Definition

A calendric system is a collection of calendars. The calendars in a calendric system are defined over contiguous and non-overlapping intervals of an underlying time-line. Calendric systems define the human interpretation of time for a particular locale as different calendars may be employed during different intervals.

Key Points

A calendric system is the abstraction of time available at the conceptual and logical (query language) levels. As an example, a Russian calendric system could be constructed by considering the sequence of six different calendars used in that region of the world. In prehistoric epochs, the Geologic calendar and Carbon-14 dating (another form of calendar) are used to measure time. Later, during the Roman empire, the lunar calendar developed by the Roman republic was used. Pope Julius, in the first century B.C., introduced a solar calendar, the Julian calendar. This calendar was in use until the 1917 Bolshevik revolution when the Gregorian calendar, first introduced by Pope Gregory XIII in 1572, was adopted. In 1929, the Soviets introduced a continuous schedule work week based on 4 days of work followed by 1 day of rest, in an attempt to break tradition with the 7-day week. This new calendar, the Communist calendar, had the failing that only eighty percent of the work force was active on any day, and it was abandoned after only 2 years in favor of the Gregorian calendar, which is still in use today.

The term "calendric system" has been used to describe the calculation of events within a single calendar. However, the given definition generalizes that usage to multiple calendars in a very natural way.

Cross-references

► Calendar
► Temporal Database
► Time Interval

Recommended Reading

1. Jensen C.S. and Dyreson C.E. (eds.). Böhlen M., Clifford J., Elmasri R., Gadia S.K., Grandi F., Hayes P., Jajodia S., Käfer W., Kline N., Lorentzos N., Mitsopoulos Y., Montanari A., Nonen D., Peressi E., Pernici B., Roddick J.F., Sarda N.L., Scalas M.R., Segev A., Snodgrass R.T., Soo M.D., Tansel A., Tiberio R. and Wiederhold G., A Consensus Glossary of Temporal Database Concepts – February 1998 Version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS, vol. 1399, Springer, Berlin, 1998, pp. 367–405.

Camera Break Detection

► Video Shot Detection

Capsule

► Snippet

Cardinal Direction Relationships

Spiros Skiadopoulos
University of Peloponnese, Tripoli, Greece

Synonyms

Orientation relationships; Directional relationships

Definition

Cardinal direction relationships are qualitative spatial relations that describe how an object is placed relative to other objects utilizing a co-ordinate system. This knowledge is expressed using symbolic (qualitative) and not numerical (quantitative) methods. For instance, *north* and *southeast* are cardinal direction relationships. Such relationships are used to describe and

constrain the relative positions of objects and can be used to pose queries such as "Find all objects *a*, *b* and *c* such that *a* is *north of b* and *b* is *southeast of c*".

Historical Background

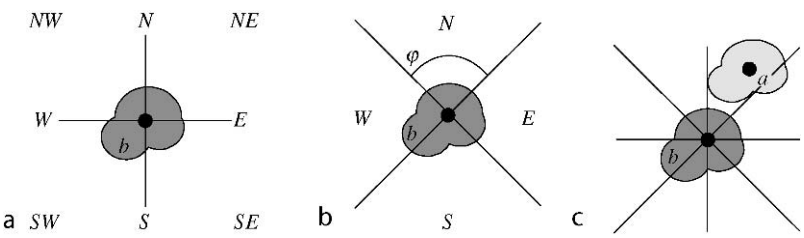
Qualitative spatial relationships (QSR) approach common sense knowledge and reasoning about space using symbolic rather than numerical methods [5]. QSR has found applications in many diverse scientific areas such as geographic information systems, artificial intelligence, databases, and multimedia. Most researchers in QSR have concentrated on the three main aspects of space, namely topology, distance and direction. The uttermost aim in these lines of research is to define new and more expressive categories of spatial operators, as well as to build efficient algorithms for the automatic processing of queries involving these operators.

Foundations

Several models capturing cardinal direction relationships have been proposed in the literature. Typically, a cardinal direction relationship is a binary relation that describes how a *primary object a* is placed relative to a *reference object b* utilizing a co-ordinate system (e.g., object *a* is *north of* object *b*). Early models for cardinal direction relationships approximate an extended spatial object by a representative point [3,6], e.g., objects in Fig. 1 are approximated by their centroid. Typically, such models partition the space around the reference object *b* into a number of mutually exclusive areas. For instance, the *projection* model partitions the space using lines parallel to the axes (Fig. 1a) while the *cone* model partitions the space using lines with an origin angle ϕ (Fig. 1b). Depending on the adopted model, the relation between two objects may change. For instance, consider Fig. 1c. According to the projection model, *a* is northeast of *b* while according to the cone model, *a* is north of *b*. Point based approximations may be crude [4], thus, later models more finely approximate an object using a representative area (most commonly the minimum bounding box (The minimum bounding box of an object *a* is the smallest rectangle, aligned with the axis, that encloses *a*.)) and express directions on these approximations [7,9]. Unfortunately, even with finer approximations, models that approximate both the primary and the reference object may give misleading directional relations when objects are overlapping, intertwined, or horseshoe-shaped [4].

Recently, more precise models for cardinal direction relationships have been proposed. Such models define directions on the exact shape of the primary object and only approximate the reference object (using its minimum bounding box). The *projection-based directional relations* ($\mathcal{PDR}$) model is the first model of this category [4,11,12]. The $\mathcal{PDR}$ model partitions the plane around the reference object into nine areas similarly to the projection model (Fig. 2a). These areas correspond to the minimum bounding box (B) and the eight cardinal directions. Intuitively, the cardinal direction relationship is characterized by the names of the reference areas occupied by the primary object. For instance, in Fig. 2b, object *a* is partly *NE* and partly *E* of object *b*. This is denoted by *a NE*:*E b*. Similarly in Fig. 2c, *a B*:*S*:*SW*:*W*:*NW*:*N*:*E*:*SE b* holds. In total, the $\mathcal{PDR}$ model identifies 511 ($= 2^9 - 1$) relationships.

Clearly, the $\mathcal{PDR}$ model offers a more precise and expressive model than previous approaches that approximate objects using points or rectangles [4]. The $\mathcal{PDR}$ model adopts a projection-based partition using lines parallel to the axes (Fig. 2a). Typically, most people find it more natural to organize the surrounding space using lines with an origin angle similarly to the cone model (Fig. 3a). This partition of space is adopted by the *cone-based directional relations* ($\mathcal{CDR}$) model. Similarly to the $\mathcal{PDR}$ model, the $\mathcal{CDR}$ model uses the exact shape of the primary object and only approximates the reference object using its minimum bounding box

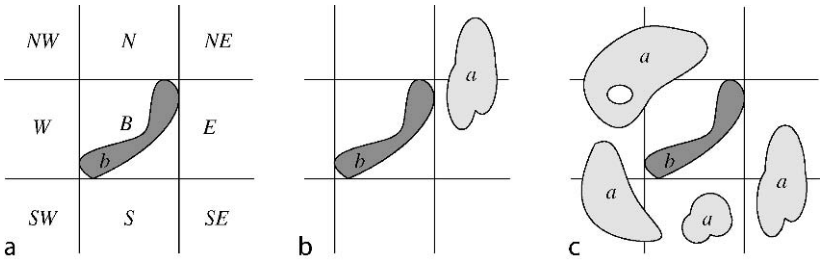


Cardinal Direction Relationships. Figure 1. Projection-based and cone-based point models.

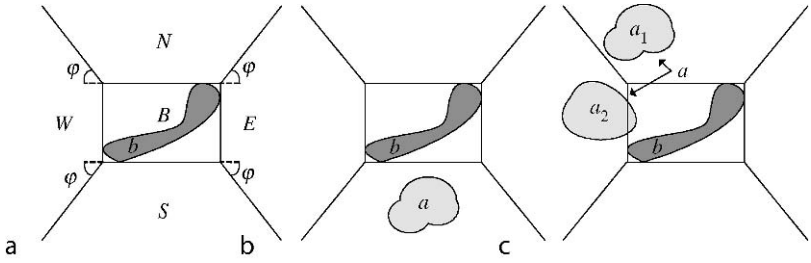
[14]. But, for the $\mathcal{CDR}$ model the space around the reference object is partitioned into five areas (Fig. 3a). The cardinal direction relationship is formed by the areas that the primary object falls in. For instance, in Fig. 3b, *a* is south of *b*. This is denoted by *a S b*. Similarly in Fig. 3c, *a B:W:N b* holds. In total, the $\mathcal{CDR}$ model identifies 31 ($= 2^5 - 1$) relationships.

In another line of research, cardinal direction relationships are modeled as ternary relationships [2]. Given three objects *a*, *b* and *c*, the ternary model expresses the direction relation of the primary object *a* with respect to a reference frame constructed by objects *b* and *c*. Specifically, the convex-hull, the internal and the external tangents of objects *b* and *c* divide the space into five areas as in Fig. 4a. These areas correspond to the following directions: right-side (*RS*), before (*BF*), left-side (*LS*), after (*AF*) and between (*BT*). Similarly to $\mathcal{PDR}$ and $\mathcal{CDR}$, the name of the areas that *a* falls into, determines the relation. For instance, in Fig. 4b, *a* is before and to the left-side of *b* and *c*. This is denoted by *LS:BF*(*a*,*b*,*c*). Notice that, if the order of the reference objects changes, the relationship also changes. For instance, in Fig. 4b, *RS:AF*(*a*,*c*,*b*) also holds.

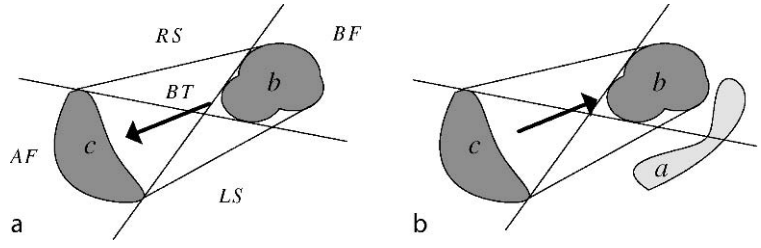
For all the above models of cardinal direction relationships, research has focused on four interesting operators: (i) efficiently determining the relationships that hold between a set of objects, (ii) calculating the inverse of a relationship, (iii) computing the composition of two relationships and (iv) checking the consistency of a set of relationships. These operators are used as mechanisms that compute and infer cardinal direction relations. Such mechanisms are important as they are in the heart of any system that retrieves collections of objects similarly related to each other using spatial relations. Table 1 summarizes current research on the aforementioned problems.



Cardinal Direction Relationships. Figure 2. Extending the projection model.



Cardinal Direction Relationships. Figure 3. Extending the cone model.



Cardinal Direction Relationships. Figure 4. Ternary cardinal direction relationships.

Cardinal Direction Relationships. Table 1. Operations for cardinal direction relationships

Model	Computation	Inverse	Composition	Consistency
Point approximations	[10]	[6]	[6]	[6]
Rectangle approximations	[9]	[9]	[9]	[9]
$\mathcal{PDR}$	[13]	[1]	[11]	[8, 12]
$\mathcal{CDR}$	Open problem	[14]	[14]	Open problem
Ternary	[2]	[2]	Open problem	Open problem

Key Applications

Cardinal direction relationships intuitively describe the relative position of objects and can be used to constrain and query spatial configurations. This information is very useful in several applications like geographic information systems, spatial databases, spatial arrangement and planning, etc.

Future Directions

There are several open and important problems concerning cardinal direction relations. For the models discussed in the previous section, as presented in Table 1, there are four operators that have not been studied (two for the $\mathcal{CDR}$ and two for the ternary model). Another open issue is the integration of cardinal direction relationships with existing spatial query answering algorithms and data indexing structures (like the R-tree). Finally, with respect to the modeling aspect, even the most expressive cardinal direction relationships models define directions by approximating the reference objects. Currently, there is not a simple and easy to use model that defines cardinal direction relationships on the exact shape of the involved objects.

Cross-references

- ▶ Geographic Information System
- ▶ Spatial Operations and Map Operations
- ▶ Topological Relationships

Recommended Reading

1. Cicerone S. and Di Felice P. Cardinal directions between spatial objects: the pairwise-consistency problem. Inf. Sci., 164 (1–4):165–188, 2004.
2. Clementini E. and Billen R. Modeling and computing ternary projective relations between regions. IEEE Trans. Knowl. Data Eng., 18(6):799–814, 2006.
3. Freksa C. Using orientation information for qualitative spatial reasoning. In Proceedings of COSIT'92, LNCS, vol. 639, 1992, pp. 162–178.
4. Goyal R. Similarity Assessment for Cardinal Directions Between Extended Spatial Objects. PhD Thesis, Department of Spatial Information Science and Engineering, University of Maine, April 2000.
5. Hernández D. Qualitative Representation of Spatial Knowledge, LNCS, vol. 804. Springer, Berlin, 1994.
6. Ligozat G. Reasoning about cardinal directions. J. Visual Lang. Comput., 9:23–44, 1998.
7. Mukerjee A. and Joe G. A qualitative model for space. In Proc. 7th National Conf. on AI, 1990, pp. 721–727.
8. Navarrete I., Morales A., and Sciavicco G. Consistency checking of basic cardinal constraints over connected regions. In Proc. 20th Int. Joint Conf. on AI, 2007, pp. 495–500.
9. Papadias D. Relation-based representation of spatial knowledge. PhD Thesis, Department of Electrical and Computer Engineering, National Technical University of Athens, 1994.
10. Peuquet D.J. and Ci-Xiang Z. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. Pattern Recognit, 20(1):65–74, 1987.
11. Skiadopoulos S. and Koubarakis M. Composing cardinal direction relations. Artif. Intell., 152(2):143–171, 2004.
12. Skiadopoulos S. and Koubarakis M. On the consistency of cardinal directions constraints. Artif. Intell., 163(1):91–135, 2005.
13. Skiadopoulos S., Giannoukos C., Sarkas N., Vassiliadis P., Sellis T., and Koubarakis M. Computing and managing cardinal direction relations. IEEE Trans. Knowl. Data Eng., 17 (12):1610–1623, 2005.
14. Skiadopoulos S., Sarkas N., Sellis T., and Koubarakis M. A family of directional relation models for extended objects. IEEE Trans. Knowl. Data Eng., 19(8):1116–1130, 2007.

Cartesian Product

Cristina Sirangelo
University of Edinburgh, Edinburgh, UK

Synonyms

Cross product

Definition

Given two relation instances R_1, over set of attributes U_1, and R_2, over set of attributes U_2 – with U_1 and U_2 disjoint – the cartesian product $R_1 \times R_2$ returns a new

relation, over set of attributes $U_1 \cup U_2$, consisting of tuples $\{t|t(U_1) \in R_1$ and $t(U_2) \in R_2\}$. Here $t(U)$ denotes the restriction of the tuple t to attributes in the set U.

Key Points

The cartesian product is an operator of the relational algebra which extends to relations the usual notion of cartesian product of sets.

Since the sets of attributes of the input relations are disjoint, in $R_1 \times R_2$ each tuple of R_1 is combined with each tuple of R_2; moreover the arity of the output relation is the sum of the arities of R_1 and R_2.

As an example, consider a relation *Students* over attributes (*student-number, student-name*), containing tuples {(1001, *Black*), (1002, *White*)}, and a relation *Courses* over attributes (*course-number, course-name*), containing tuples {(*EH*1, *Databases*), (*GH*5, *Logic*)}. Then *Students* × *Courses* is a relation over attributes (*student-number, student-name, course-number, course-name*) containing tuples (1001, *Black*, *EH*1, *Databases*), (1001, *Black*, *GH*5, *Logic*), (1002, *White*, *EH*1, *Databases*), (1002, *White*, *GH*5, *Logic*).

The cartesian product can also be viewed as a special case of natural join, arising when the set of attributes of the operands are disjoint. However, relations over non-disjoint sets of attributes can also be combined by the cartesian product, provided that the renaming operator is used to rename common attributes in one of the two relations.

In the presence of attribute names, the cartesian product is commutative. In the case that relation schemas do not come with attribute names, but are specified by a relation name and arity, the cartesian product $R_1 \times R_2$ returns the concatenation t_1t_2 of all pairs of tuples such that $t_1 \in R_1$ and $t_2 \in R_2$. Moreover the output schema is specified by the sum of the arities of the input schemas. In this case the cartesian product is a non-commutative operator.

Cross-references

- ▶ Join
- ▶ Relation
- ▶ Relational Algebra
- ▶ Renaming

Cartography

- ▶ Visual Interfaces for Geographic Data

CAS

- ▶ Storage Security

CAS Query

- ▶ Content-and-Structure Query

Case Handling

- ▶ Business Process Management

Case Management

- ▶ Workflow Management

Case Report Forms

- ▶ Clinical Data Acquisition, Storage and Management

Cataloging

- ▶ Cataloging in Digital Libraries

Cataloging in Digital Libraries

Mary Lynette Larsgaard
University of California-Santa Barbara, Santa Barbara, CA, USA

Synonyms

Cataloging; Classification

Definition

Cataloging is using standard rules to create a mainly text surrogate that describes an object sufficiently in detail so that the object is uniquely differentiated from all other objects. Without looking at the object, a user may know enough about the object to know if it suits

the user's needs. It is generally considered to include bibliographic description, and the application of subjects, both as words and as classification.

Historical Background

Devising and using methods of arranging and describing information – respectively termed, within the standard library world, classification and cataloging – have been primary concerns of libraries ever since libraries began, in the ancient world of the Greeks and the Romans. A collection of information without classification and cataloging is not a library. The whole point of classification and cataloging is to make access quick and easy for users; it was discovered very early that putting like objects together (classification) and creating text or relatively speaking much smaller surrogates to describe an information object (cataloging) made finding information much quicker for the user.

Experiments in using digital records in libraries started in approximately the late 1960s. But it was only in the mid-1970s, with the success of what is called "shared cataloging" – many libraries using a catalog record contributed as "original cataloging" by the first library to catalog the item – that using digital systems for cataloging came into its own. This sharing of bibliographic records in online form began with OCLC, initially as a consortium of college libraries in Ohio (starting in 1967), but growing rapidly to become the most successful such library utility, currently with about 60,000 participating libraries in 112 countries and territories (http://www.oclc.org). The development of integrated library systems (ILS) or library management systems (LMS) – software, or a combination of software and hardware, that permits a library to perform multiple functions, such as acquisitions/ordering, cataloging and classification, circulation, preservation, and reference, using digital files in large databases with many tables – has continued to the present in the "Library of Congress Authorities" online authority system, http://authorities.loc.gov).

The inception and speedy growth of the World Wide Web (Web) since the mid-1990s has spread this interest in arrangement and description of, and access to, information objects to non-library communities, and within the library world to how specifically to arrange and describe digital objects made available over the Web in digital libraries. There have been many standards for the description of information objects. They are most often intended either to apply at least in theory to all materials (e.g., Anglo-American Cataloging Rules, hereafter referred to as AACR; Dublin Core, which began in 1995, http://www.dublincore.org/documents/dces/) or to apply to the description of a specific body of information (e.g., for digital geospatial data, "Content Standard for Digital Geospatial Metadata"; 2nd edition, 1998, http://www.fgdc.gov/standards/projects/FGDC-standards-projects/metadata/base-metadata/v2_0698.pdf and ISO Standard 19115, "Geographic Information, metadata; Information géographique, métadonnées," 2003).

Foundations

Classification

Classification and cataloging complement each other. Classification is a form of subject cataloging, which is where the major overlap between the two occurs. Classification tends to be hierarchical, breaking a given world of information or knowledge into broad divisions (e.g., Law) and then breaking that into smaller divisions (e.g., education for law; law of various countries; etc.). Classification is most often placing like items about like subjects (e.g., works by Shakespeare) and like genres or formats (e.g., maps) together. It also provides a physical location within a library for each object, be it digital or hardcopy. The most prominent systems used are the Dewey Decimal Classification (often used by public libraries and smaller libraries generally), and the Library of Congress Classification (most often used by university and other research libraries in the United States). There are many more systems, such as the Bliss Classification and the Colon Classification, and special and research libraries (such as the New York Public Library) devise and maintain their own systems. Devising a classification system is easy, but maintaining it is very difficult and time-consuming. Often a library maintaining a classification system unique to itself will find that it is far less work to convert to a rigorously maintained system that is used by many libraries than it is to maintain a unique system not used by any other collections.

Classification of digital objects at first glance seems unnecessary. Why not just assign an arbitrary number (e.g., a unique identifier that database software assigns to each separate catalog record) and be done with it? Libraries of digital objects have found that, for several reasons, it is very practical to assign classification to digital objects just as one would to hard-copy objects. The main one is that very often one needs to move around, or to perform the same operation (e.g., create

distribution forms) on large numbers of digital objects in groups of items that are, e.g., the same file type. For example, the Alexandria Digital Library (ADL), a collection of digital geospatial data, with a catalog that includes catalog records both for digital and hard-copy geospatial data (http://webclient.alexandria.ucsb.edu), always gives a digital geospatial data object (e.g., a scan of a paper map; a born-digital object) the same classification number as the hardcopy geospatial data equivalent (e.g., the paper map that has been scanned). Classification of digital objects allows one file to be accessible from multiple classification numbers, an action that few libraries of hardcopy items have ever been able to afford. For example, a digital map of California and Nevada may have two classification numbers (either pointing toward one file, or storing the same file two separate places), one for California and one for Nevada. Few libraries of hardcopy items have financial resources available to buy multiple copies of an item and store a copy at each applicable classification number.

Cataloging

A catalog record (whether in hardcopy or in digital form) provides information on the thematic and physical nature of an item (whether hard-copy or digital) being cataloged. Libraries first used hard-copy catalogs, generally book-format catalogs, then cards, and then beginning in the late 1960s the use of databases as catalogs. This started with in 1969 the Library of Congress' MARC, MAchine-Readable Catalog format for the transmission of catalog-card information in digital form (http://www.loc.gov/marc/). While MARC may be used as the machine format for bibliographic records formulated using any set of cataloging rules, it is most often used for records based on the cataloging rules of the Anglo-American library community, AACR (Anglo-American Cataloging Rules) in its various editions, first issued in 1967. AACR itself is based on the International Standard Bibliographic Descriptions (ISBDs) as far as content of fields, and field order, are concerned.

The purpose of ISBDs is to standardize the form and content – including in what order information appears – for the bibliographic description of an object. ISBDs specifically do not include rules for fields for subject cataloging, added entries for additional authors, or classification. The idea of ISBDs arose at a conference of the International Meeting of Cataloging Experts held in 1969, with the first ISBD (for monographic publications) being issued in 1971. ISBDs were issued for numerous categories of publications (e.g., computer files; cartographic materials; serials; etc.; for full list, see http://www.ifla.org/VI/3/nd1/isbd-list.htm). In 2002, work began on a single consolidated ISBD, to supersede all existing ISBDs. The consolidated edition (available online at http://www.ifla.org/VII/s13/pubs/cat-isbd.htm) was published in 2007 (International standard bibliographic description (ISBD), 2007).

The same organization that issues ISBDS – the International Federation of Library Associations (IFLA) – has been a prime agent in the move toward an international cataloging standard. In the early 1990s, IFLA's Study Group on the Functional Requirements for Bibliographic Records (FRBR) began work on its report to recommend basic functionalities of a catalog and bibliographic-record data requirements. The Group's final report was issued in 1998 (Functional Requirements for Bibliographic Records (FRBR)).

While FRBR put forward several ideas, the one that most engaged the interest and discussion of the library community was the importance of incorporating into the bibliographic description the concept of the relationship between the work, the expression, the manifestation, and the item (called Group 1 Entities). A work is a distinct intellectual or artistic creation but a concept rather than an actual physical object. An expression is the intellectual or artistic realization of a work but still not generally an actual physical object. The manifestation is all copies of a published object (e.g., all copies of a printed map; all copies of a DVD of a specific piece of music), and the item is one copy of a manifestation. For example, all digital versions of one given map (one could be raster and the other could be vector; there could be more than one level of resolution of raster and of vector images), and all hard-copy versions of the same map (e.g., paper; microfiche; microfilm) are two expressions' the basic map itself is the work and the 1973 edition of all copies of the paper map is a specific manifestation, with each one of those printed maps being an item.

Key Applications

Key applications of classification and cataloging:

Classification: Arrangement of digital objects in digital libraries. See previous section
Cataloging: Creation of metadata for digital objects in digital libraries

While full-text searching is extremely useful and a major step forward in the history of information retrieval, it does have the following main problems: it may give the user very large numbers of hits; it is not as efficient as searching well-constructed text surrogates; and it does not work for what are primarily non-text materials (e.g., music; maps; etc.). The reason for creating metadata records for digital objects is the same as that for performing standard cataloging – constructing a surrogate for the item so that users may quickly and efficiently find resources that suit the users' needs. Metadata is constructed by non-library entities (e.g., federal government agencies) and by libraries. Metadata records include but are not limited to the information contained in what the standard library cataloging world terms "bibliographic records."

Metadata records tend to be considerably longer than catalog records, because they contain far more and generally more detailed technical information than a catalog record would. The latter is much more likely to simply include a URL that points to an online version of that technical information, quite possibly to a metadata record. The reasons for this are that metadata records tend to be constructed for very focused, often technically skilled audiences, and a geographic digital dataset (such as a geographic information system, more commonly known as a GIS) is often quite large and complicated, with many layers of information, and therefore is by no means as easily browsed – in order to determine its suitability for use – as is a hard-copy map. For example, the catalog record for Digital Orthophoto Quarter Quadrangles (DOQQs; mosaics of rectified aerial photographs) is relatively brief – about one standard printed page – when compared with the metadata record for DOQQs, which is seven pages. For example, see http://fisher.lib.virginia.edu/collections/gis/doq/helps/doqq_meta.html (Fig. 1).

A library generating metadata records has two major options: load the records into the library's ILS (integrated library system) online catalog; or create what is in effect another ILS, or at the very least an online catalog for the metadata records. The first technique generally requires that the records be in MARC format, since the alternative is that the online-catalog software must be capable of searching over multiple catalog-record databases in different formats.

For the second technique, the following is required: software (UNIX; a database manager; a user interface; and middleware to connect inquiries on the user interface with the data and return results); hardware; computer technical staff/programmers (for a digital library of any size, a minimum of three computer programmers to deal with adding new data and metadata and maintaining and improving the system, including the interface, plus one more programmer to deal with the operating system, disk storage, and manipulation and maintenance of the directories of digital material).

An example of this is the ADL (Alexandria Digital Library) Catalog, http://webclient.alexandria.ucsb.edu. The ADL webpage (http://www.alexandria.ucsb.edu) provides an outline as to what kind and how much work is required to start up, develop, and maintain such a library; software is free for download, and general instructions are given as to what work should be done in order to get the software working.

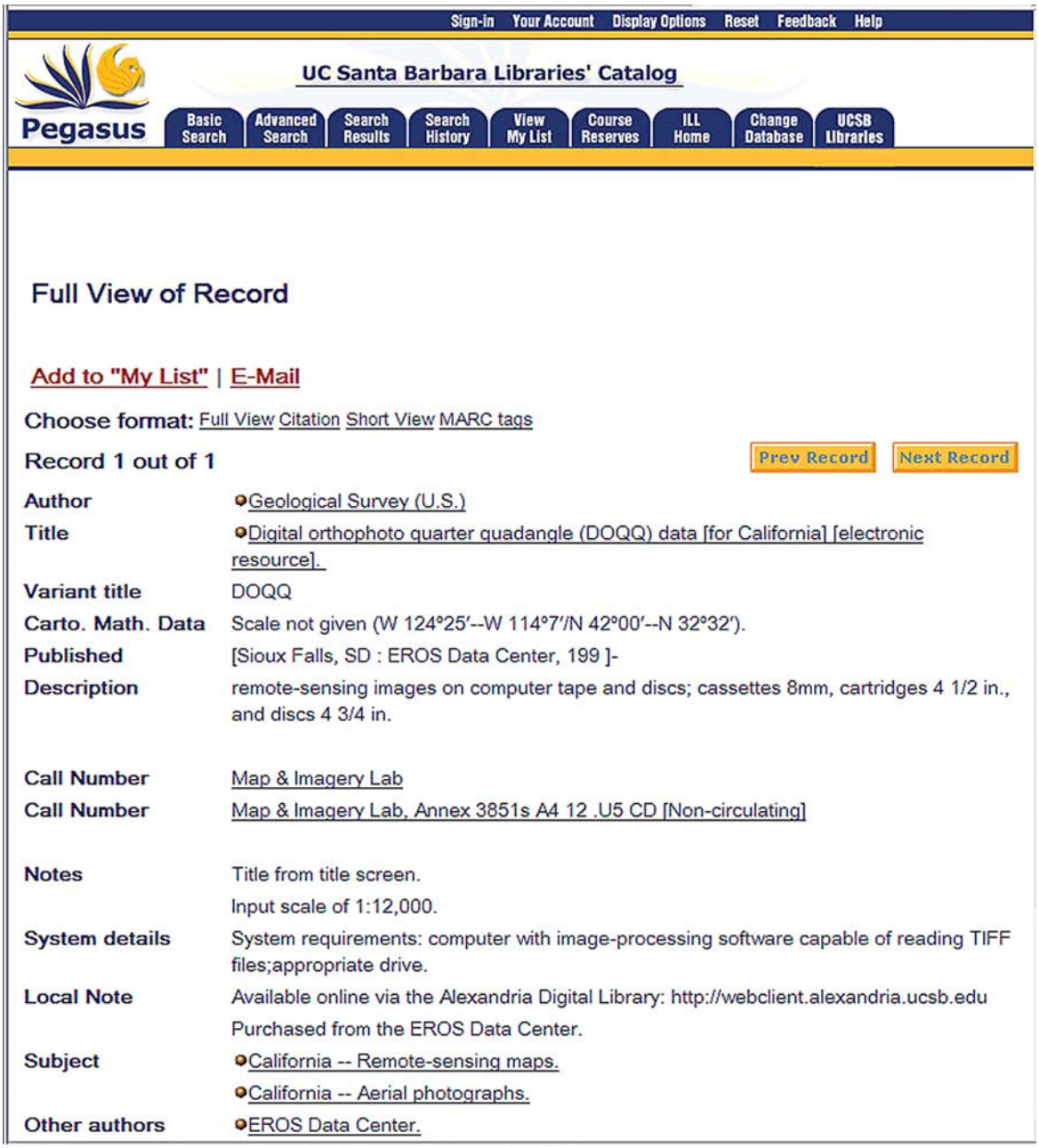
As previously indicated, there are numerous metadata standards. The following are major standards that digital libraries creating metadata records will probably need to deal with, at least in the United States: Dublin Core; XML; METS; and MODS.

Dublin Core

Dublin Core (DC) (http://dublincore.org/) is extremely heavily used by libraries cataloging digital content. Its adaptability to any form or type of digital data and its brevity (15 fields, what libraries term minimal-level cataloging) with no fields required and all fields repeatable, makes it very flexible. While DC may be used either as "qualified" (each of the 15 elements may be qualified in some way to make the information clear, e.g., for Coverage, one might state that the geographic area is given in decimal degrees), the experience in libraries over the nearly 15 years since DC was made available for use is that it is strongly advised only qualified DC be used. This is because unqualified DC results in metadata records that are so unstructured as to be nearly useless [12]. For digital libraries needing keep at least one foot solidly in the traditional library world, there is a DC-to-MARC2 crosswalk at http://www.loc.gov/marc/dccross.html, and also one the other direction.

XML

XML has achieved primacy as the format of choice for metadata for digital libraries and is of considerable importance to the standard library world, as evidenced by the Library of Congress having MARC21 in XML available over the Web at http://www.loc.gov/standards/marcxml/Sandburg/sandburg.xml. It was announced in April 2005 that the ISO



Cataloging in Digital Libraries. Figure 1. Record from a library online catalog.

committee for Technical Interoperability – Information and Documentation was to vote on a proposal for a New Work Item concerning an XML schema to wrap MARC records. ISO 2709 had been used for many years and has worked well, but the library community needed a standard exclusively for MARC records (of which there are over 100 million worldw ide) in XML. The standard is to be published as ISO 25577 with the short name of MarcXchange; the temporary Webpage for the standard is http://www.bs.dk/marcxchange/.

METS and MODS

METS (Metadata Encoding and Transmission Standard) is a standard for encoding descriptive, administrative,

and structural metadata of objects in a digital library (http://www.loc.gov/standards/mets/).

The "Metadata Object Description Schema" (MODS) is intended both to carry metadata from existing MARC21 records, and to be used to create new catalog records. It has a subset of MARC fields, and – unlike MARC21. It uses language-based tags rather than numeric tags. It occasionally regroups elements from the MARC21 bibliographic format (http://www.loc.gov/standards/mods/). METS and MODS are both expressed using XML, and are maintained by the Library of Congress's Network Development and MARC Standards Office.

Conclusion

Organizations creating metadata records and arranging digital files are best advised to follow well-maintained national or international standards. In no case should organizations just starting out on this work create their own standards. Instead, use of sturdy standards – some of which have an extensions feature, to enable customization of the records to the library users' needs – is recommended.

Creating metadata records is relatively easy compared with the difficult and expensive work of setting up what is in effect an ILS online catalog. Libraries need to consider this very carefully. If the library cannot sustain the programming effort required to develop and then to maintain and add to the catalog, then the library should not begin the project. If anything, digital libraries and their catalogs are, at the moment, at least as expensive and time-consuming to develop and maintain as are hard-copy libraries.

Cross-references

- ▶ Annotation
- ▶ Audio Metadata
- ▶ Biomedical Data/Content Acquisitions
- ▶ Biomedical Metadata Management and Resource Discovery
- ▶ Browsing in Digital Libraries
- ▶ Classification by Association Rule Analysis
- ▶ Classification on Streams
- ▶ Clinical Data/Content Acquisition
- ▶ Cross-Modal Information Retrieval
- ▶ Curation
- ▶ Data Warehouse Metadata
- ▶ Digital Libraries
- ▶ Discovery
- ▶ Dublin Core (DC)
- ▶ Field Based Information Retrieval Models
- ▶ Geographic Information Retrieval
- ▶ Image Metadata
- ▶ Indexing Historical Spatio-Temporal Data
- ▶ Information Retrieval
- ▶ Information Retrieval Model
- ▶ ISO/IEC 11179
- ▶ Knowledge Discovery Metamodel
- ▶ Metadata
- ▶ Metadata Interchange Specification(MDIS)
- ▶ Metadata Registry
- ▶ Metadata Repository
- ▶ Metasearch Engines
- ▶ METS
- ▶ Multimedia Metadata
- ▶ Ontologies
- ▶ Ontology
- ▶ Schema Mapping
- ▶ Searching Digital Libraries
- ▶ Text Indexing and Retrieval
- ▶ XML
- ▶ XML Metadata Interchange

Recommended Reading

1. Anglo-American Cataloging Rules, American Library Association, Chicago, 2005.
2. Borgman C.L. From Gutenberg to the Global Information Infrastructure: Access to Information in the Networked World. MIT Press, Cambridge, MA, 2000.
3. Chan L.M. Cataloging and Classification: An Introduction. Scarecrow Press, Blue Ridge Summit, PA, 2007.
4. IFLA Study Group. Functional Requirements for Bibliographic Records (FRBR). K.G. Saur, Munchen, 1998, (UBCIM publications, new series; vol. 19). Available online at: http://www.ifla.org/VII/s13/frbr/frbr.htm.
5. IFLA Study Group. International Standard Bibliographic Description (ISBD), (Preliminary consolidated edn.). K.G. Saur, München, 2007 (IFLA series on bibliographic control; vol. 31).
6. Kochtanek T.R. Library Information Systems, From Library Automation to Distributed Information Access Solutions. Libraries Unlimited, Westport, CT, 2002.
7. Libraries. Encyclopedia Britannica, Micropedia 7:333–334; Macropedia 22:947–963. Encyclopedia Britannica, Chicago, 2002. Available online at: http://search.eb.com/.
8. Library of Congress. 1969? MARC21 Concise Bibliographic, Library of Congress, Washington, DC. Available online at http://www.loc.gov/marc/.
9. Linton J. Beyond Schemas, Planning Your XML Model. O'Reilly, Sebastopol, CA, 2007.
10. Reitz J.M. Dictionary for Library and Information Science. Libraries Unlimited, Westport, CT, 2004.

11. Svenonius E. The Intellectual Foundation of Information Organization. MIT Press, Cambridge, MA, 2000.
12. Tennant R. Bitter Harvest: Problems and Suggested Solutions for OAI-PMH Data and Service Providers. California Digital Library, Oakland, CA, 2004. Available online at:http://www.cdlib.org/inside/projects/harvesting/bitter_harvest.html.

CDA

▶ Clinical Document Architecture

CDA R1

▶ Clinical Document Architecture

CDA R2

▶ Clinical Document Architecture

CDP

▶ Continuous Data Protection

CDs

▶ Storage Devices

CDS

▶ Clinical Decision Support

Cell Complex

▶ Simplicial Complex

Certain (and Possible) Answers

Gösta Grahne
Concordia University, Montreal, QC, Canada

Synonyms

True answer (Maybe answer); Validity (Satisfiability)

Definition

Let T be a finite theory expressed in a language L, and ϕ an L-sentence. Then T finitely entails ϕ, in notation $T \models \phi$, if all finite models of T also are models of ϕ. A theory T is said to be *complete in the finite* if for each L-sentence ϕ either $T \models \phi$ or $T \models \neg\phi$. In particular, if T is incomplete (not complete in the finite), then there is an L-sentence ϕ, such that $T \not\models \phi$ and $T \not\models \neg\phi$. It follows from classical logic that a first order theory is complete in the finite if and only if all its finite models are isomorphic. Consider now a theory

$$T_1 = \begin{cases} R(a,b) \wedge R(a,c), \\ \forall x,y : R(x,y) \rightarrow (x,y) = (a,b) \vee (a,c), \\ a \neq b, a \neq c, b \neq c. \end{cases}$$

where a, b, and c are constants. This theory is complete, and clearly for instance $T \models R(a, b)$, $T \models R(a, c)$, and $T \not\models R(d, c)$, for all constants d different from a and b. Consider then the theory

$$T_2 = \begin{cases} R(a,b) \vee R(a,c), \\ \forall x,y : R(x,y) \rightarrow (x,y) = (a,b) \vee (a,c), \\ a \neq b, a \neq c, b \neq c. \end{cases}$$

This theory is incomplete, since for instance $T_2 \not\models R(a, b)$, and $T_2 \not\models \neg R(a, b)$. If "finitely entails" is equated with "certainly holds," it is possible to say that $R(a, b)$ and $R(a, c)$ *certainly* hold in T_1. Dually, it is possible to say that $R(a, b)$ *possibly* holds in T_2, since $T_2 \not\models \neg R(a, b)$, and similarly that $R(a, c)$ possibly holds in T_2.

Key Points

An *incomplete database* is similar to a logical theory: it is defined using a finite specification, usually a *table* T (relation with nulls and conditions) of some sort, and a function *Rep* that associates a set of complete (ordinary, finite) databases $Rep(T)$ with T. Then each instance $I \in Rep(T)$ represents one isomorphism class (isomorphism up to renaming of the constants) of the finite models of the table T regarded as a logical theory. Depending on the interpretation of facts missing from T, either the *closed world assumption* is made [9], which postulates or axiomatizes (as in the middle "row" in T_1 and T_2) that any facts not deducible from T are *false*, or the *open world assumption* (omit the middle rows), in which there are certain and possible facts, but no false ones. There is actually a spectrum

of closed world assumptions, ranging up to semantics best axiomatized in third order logic [4].

Having settled on a representation T, and an interpretation *Rep*, the *certain answer* to a query Q on an incomplete database T, is now defined as $\bigcap_{I \in Rep(T)} Q(I)$, sometimes also denoted $\bigcap\ Q(Rep(T))$. In database parlance the certain answer consists of those facts that are true in *every* possible database instance I that T represents. Likewise, the *possible answer* to a query Q on an incomplete database T, consists of those facts which are true in *some* possible database, i.e., $\bigcup_{I \in Rep(T)} Q(I)$. Needless to say, the possible answer $\bigcup\ Q(Rep(T))$ is interesting only under a closed world assumption, since otherwise every fact is possible.

These definitions are clear and crisp, but unfortunately it doesn't mean that they always have tractable computational properties. Consider the membership problem for the set

$$\textsc{Cert}(Q) = \{(t, T) : t \in \bigcap Q(Rep(T))\}.$$

If T actually is a complete instance I, it is well known that $\textsc{Cert}(Q)$ has polynomial time complexity, for any first order (relational algebra) or datalog query Q. Likewise, the set

$$\textsc{Poss}(Q) = \{(t, T) : t \in \bigcup Q(Rep(T)).$$

has PTIME complexity for first order and datalog queries Q, and tables T that actually are complete databases.

A table T with *unmarked nulls* is a classical instance containing existentially quantified variables (nulls), such that each existential quantifier has only one variable in its scope. This means that each occurrence of a null can be independently substituted by a constant for obtaining one possible database in $Rep(T)$.

If only simple incomplete databases with unmarked nulls are allowed, only existential first order queries Q need to be admitted, or alternatively algebraic expressions with operators from $\{\pi, \sigma, \cup, \bowtie\}$, in order for $\textsc{Cert}(Q)$ to become coNP-complete, and $\textsc{Poss}(Q)$ to become NP-complete [2]. The use of inequalities $\neq$ or disjunctions $\vee$ in Q is essential. If the use of inequalities and disjunctions is denied, $\textsc{Cert}(Q)$ and $\textsc{Poss}(Q)$ remain in PTIME. If one admits arbitrary first order or full relational queries Q, along with an open world assumption, $\textsc{Cert}(Q)$ and $\textsc{Poss}(Q)$ become undecidable. This follows from validity and satisfiability of a variant of first order logic known to be undecidable [3]. (Note that under the open world assumption $\textsc{Poss}(Q)$ equals all possible databases, assuming $\textsc{Poss}(Q) \neq \emptyset$. The problem then becomes to decide whether $\textsc{Poss}(Q)$ is non-empty or not.)

On the other hand, if the representation mechanism allowed for T is more powerful that the simple incomplete databases above, $\textsc{Cert}(Q)$ and $\textsc{Poss}(Q)$ again become coNP and NP complete, respectively, already with Q being the identity query. For this, the *conditional tables* of [6] are needed. As observed in [5], conditional tables can be obtained as a closure of simple incomplete databases by requiring that the exact result $\{q(I) : I \in Rep(T)\}$ of any relational algebra query on a any table T is representable by conditional table. In other words, for each T conditional table and Q relational algebra query, there exists a conditional table U, such that $Rep(U) = Q(Rep(T))$.

Another way of representing incomplete databases, is to consider an *information integration* scenario, where the basic facts are stored in *views* of a virtual *global schema*. For instance, in the integration scenario

$$T_3 = \begin{cases} V(a), \\ \forall x, y : R(x, y) \rightarrow V(x), \\ \forall x : V(x) \rightarrow x = a, \\ \forall x, y : R(x, y) \rightarrow x = a \end{cases}$$

gives (closed world) $Rep(T_3) = \{I : V^I = \pi_1(R^I) = \{(a)\}\}$. ($R^I$ means the value (interpretation) of predicate symbol R in instance/model I. The meaning of V^I is similar.) Open world (omit in T_3 the third and fourth rows) $Rep(T_3)$ would be defined as $\{I : V^I \supseteq \{(a)\}, \pi_1(R^I) \supseteq \{(a)\}\}$. If T is allowed to use conjunctive queries (such as the second row of T_3) to express the views V in terms of the global relations R, then $\textsc{Cert}(Q)$ is in PTIME for existential first order and datalog queries under the open world assumption, and coNP complete under the closed world assumption [1]. The latter is due to the negation implicit in the closed world assumption. If one allows inequalities $\neq$ in the query, $\textsc{Cert}(Q)$ is coNP complete also under the open world assumption. Undecidability of $\textsc{Cert}(Q)$ is achieved by allowing negation in the query or the view definitions, or, under the open world assumption by allowing view definitions in (recursive) datalog.

Finally, one can see the *data exchange* problem [7,8] as a variation of the integration problem. The data exchange problem consists of importing the data from a source database R_s to a target database R_t,

using data dependencies (implicational sentences) to express the translation. For example, a (closed world) exchange scenario could be

$$T_4 = \{R_s(a), \forall x : [R_s(x) \leftrightarrow \exists y : R_t(x, y)].$$

The base facts are in a source database R_s, and the user query is expressed against the target database R_t. As T_4 obviously is incomplete, the certain answer of Q on T_4 is defined as $\cap Q(Rep(T_4))$, and the possible answer as $\cup Q(Rep(T_4))$. It is perhaps no big surprise that essentially the same complexity landscape for CERT(Q) and POSS(Q) as in the previous table- and integration-scenarios emerges: the boundaries between undecidability, intractability, and polynomial time depend on similar restrictions on the use of negation, of inequalities or unions in the exchange mappings, and on the open or closed world assumptions.

Cross-references

- ▶ Conditional Tables
- ▶ Incomplete Information
- ▶ Naive Tables
- ▶ Null Values

Recommended Reading

1. Abiteboul S. and Duschka O.M. Complexity of answering queries using materialized views. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 254–263.
2. Abiteboul S., Kanellakis P.C., and Grahne G. On the representation and querying of sets of possible worlds. Theor. Comput. Sci., 78(1):158–187, 1991.
3. Di Paola R.A. The recursive unsolvability of the decision problem for the class of definite formulas. J. ACM, 16(2):324–327, 1969.
4. Eiter T., Gottlob G., Gurevich Y. Curb your theory! a circumspective approach for inclusive interpretation of disjunctive information. In Proc. 13th Int. Joint Conf. on AI, 1993, pp. 634–639.
5. Green T.J. and Tannen V. Models for incomplete and probabilistic information. In Proc. EDBT 2006 workshops LNCS Vol. 4251, 2006.
6. Imielinski T. and Lipski W. Incomplete information in relational databases. J. ACM, 31(4):761–791, 1984.
7. Kolaitis P.G. Schema mappings, data exchange, and metadata management. In Proc. 24th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2005, pp. 61–75.
8. Libkin L. Data exchange and incomplete information. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 60–69.
9. Reiter R. On closed world data bases. In Logic and Data Bases, 1977, pp. 55–76.

Chandra and Harel Complete Query Languages

▶ Complete query languages

Change Detection and Explanation on Streams

▶ Change Detection on Streams

Change Detection on Streams

DANIEL KIFER
Yahoo! Research, Santa Clara, CA, USA

Synonyms

Change detection and explanation on streams

Definition

A *data stream* is a (potentially infinite) sequence of data items x_1,x_2,.... As opposed to traditional data analysis, it is not assumed that the data items are generated independently from the same probability distribution. Thus *change detection* is an important part of data stream mining. It consists of two tasks: determining when there is a change in the characteristics of the data stream (preferably as quickly as possible) and explaining what is the nature of the change.

The nature of the data stream model means that it may be infeasible to store all of the data or to make several passes over it. For this reason, change detection algorithms should satisfy the following desiderata: the memory requirements should be constant or increase logarithmically, and the algorithm should require only one pass over the data.

Historical Background

There has been a lot of work on detecting change in time series data *after* all of the data has been collected. Change point analysis [5] is a statistical field devoted to detecting the point in time where the distribution of the data has changed. The description of the change is often concerned with how the parameters of the distributions (such as the mean) have changed. Scan statistics [11] can be used to detect and describe changes (not just along the time dimension) by

identifying regions where the probability mass has changed the most. For examples of offline analysis of change in terms of data mining models see [6] for itemset mining, [10] for itemset mining and decision trees, and [14] for burst detection. Offline algorithms for describing change include [1] for hierarchical numerical data and [7,8] for semi-structured data.

The offline methods are useful for data analysis, but as data acquisition becomes easier and easier, the data stream model becomes more and more relevant. The assumptions behind this model are that there is so much data that it cannot all be stored on disk (let alone kept in memory) and that the data arrives at such a rate that expensive online computations are impractical. Often results are expected in real-time – for example, notification of change should occur as soon as possible. The data stream model is discussed in detail in [2].

In the data stream model, one can predefine a certain set of stochastic processes. One of these processes is initially active and the goal is to determine when a different process in that set becomes active [3]. The description of the change is then the identity of the new process. Alternative approaches [9,13] avoid making distributional assumptions by using ideas from nonparametric statistics. In these approaches the main idea is to divide the domain of the data into (possibly overlapping) regions, estimate the probability of the regions, and then determine whether the changes in probability in any of the regions are statistically significant. Alternate approaches for handling change include testing if the data are exchangeable (i.e., any permutation of the data is equally likely) [15] and developing data mining algorithms (such as decision tree construction) that adapt to change [12].

Foundations

Let $x_1,x_2,...$, be a potentially infinite sequence of data points. To detect changes in the data stream, one first has to determine a plausible framework that describes how the data can be generated. In the simplest case, data are generated from one of two probability distributions S_1 and S_2 (for example a Gaussian with mean 0 and variance 1, and a Gaussian with mean 10 and variance 1). Initially, the data points are generated independently from S_1 and after some time the data are generated independently from S_2. The celebrated CUSUM algorithm by Page [4] can be used to detect that a change from S_1 to S_2 occurred by comparing the likelihoods that parts of the data were generated by S_1 or S_2. Suppose S_1 has density f_1 and S_2 has density f_2, and let $\delta > 0$ be a threshold.

A user of the change-detection system is interested in the first time k where. $\sum_{i=k}^{\text{now}} \log(f_2(x_i)/f_1(x_i)) > \delta$. When this happens, the system signals that a change has occurred and can return k as a plausible estimate of the change point. This test can be done in an online fashion by defining $T_0 = 0$ and $T_k = \max(T_{k-1} + \log(f_2(x_k)/f_1(x_k)),0)$ and signaling a change if $T_{\text{now}} > \delta$. Typically S_1 is chosen based on an initial sample of the data stream and S_2 is then chosen to represent the smallest change whose detection is desired. For example, suppose that S_1 is chosen to be a Gaussian with mean 0 and variance 1, and suppose that for the current application it is desirable to detect any change in mean greater than 10. Then a natural choice for S_2 is a Gaussian with mean 10 and variance 1.

This framework has been generalized by Bansal and Papantoni-Kazakos [3] to the case where S_1 and S_2 are stochastic processes that need not generate each point independently. Additional generalizations of the CUSUM algorithm, including the case of multiple data generating distributions, are discussed in [4].

The framework of the CUSUM algorithm is an example of a *parametric* framework: the set of possible data-generating distributions has been prespecified and elements in that set can be identified using a small number of parameters. Parametric approaches are powerful in cases where they can accurately model the data. In cases where the data is not well modeled by a parametric framework, performance may deteriorate in terms of more false change reports and/or fewer detection of changes.

Kifer, Ben-David, and Gehrke [13] showed how to avoid problems with parametric approaches by using a nonparametric framework. In this framework the data points $x_1,x_2,...$, are k-dimensional vectors of real numbers. Point x_1 is generated by some (arbitrary) probability distribution F_1, x_2 is generated by F_2 (independently of x_1), x_3 is generated by probability distribution F_3 (independently of x_1 and x_2), etc. A change is defined as a change in the data-generating distribution; if the first change occurs at time n_1 then $F_1 = F_2 = ... = F_{n_1-1} \neq F_{n_1}$; if the second change occurs at time n_2 then $F_{n_1} = F_{n_1+1} = ... = F_{n_2-1} \neq F_{n_2}$, etc. This framework for detecting change consists of three parts: a collection of regions of interests, a method for estimating probabilities, and a statistical test.

Regions of Interest

A collection of regions of interest serves two purposes: to restrict attention to changes that are considered meaningful, and to provide a means for describing the change.

Ideally a change is said to have occurred whenever the data-generating distribution changes. However, for many practical applications not all changes are meaningful. For example, consider the case where F_1 is a probability distribution that assigns probability 1 to the set of real numbers between 0 and 1 whose seventeenth significant digit is odd and furthermore suppose that F_1 is uniform over this set. From time 1 up to $n-1$ the data are generated independently from the distribution F_1. At time n a change occurs and from that point the data are generated independently from the distribution F_n defined as follows: F_n assigns probability 1 to the set of real numbers between 0 and 1 whose seventeenth significant digit is even and is uniform over this set. Letting f_1 be the probability density function for F_1 and f_n be the probability density function for F_n it can be seen that f_1 and f_n are very different according to some common similarity measures. Indeed, the L^1 distance between f_1 and f_n (i.e., $\int_0^1 |f_1(x) - f_n(x)|\, dx$), is as large as possible. However, in many applications it is of no practical consequence whether the true distribution is F_1 or F_n. This is because one may be interested only in questions such as "what is the probability that the next point is larger than 0.75" or "what is the probability that the next point is within the safety range of 0.14–0.95." For these types of questions one would only be interested in the probabilities of various intervals, so instead of receiving notification of arbitrary types of change, one would be happy to know only when some interval has become more or less probable. In this case, the intervals are said to be the *regions of interest*. In general, if the domain of each data element is $\mathcal{D}$ then the set of regions of interest is a collection of subsets of $\mathcal{D}$. Note that regions of interest can be overlapping, as in the case of intervals, or they can form a partition of the domain. Dasu et al. [9] also proposed to partition the domain based on an initial sample of the data.

Once the regions of interest have been specified, the goal is to report a change whenever the system detects that the probability of a region has changed. The region (or regions) with the largest change in probability are then given as the description of the change.

Estimating Probabilities

Since the true distributions F_i are unknown, it is necessary to estimate them. A *window of size m* is a set of m consecutive data points. The initial distribution is estimated using a window that contains the first m data points and the most recent distribution is estimated using a window containing the most recent m data points (in practice, several change detection algorithms can be run in parallel, each using a different value of m). In each window W_i, the probability of a particular region of interest R can be estimated by $\frac{|W_i \cap R|}{|W_i|}$, the fraction of points in the window that occur in the region. Alternatively, if the regions of interest form a partition of the domain into k regions, then a Bayesian-style correction $\frac{|W_i \cap R| + \alpha}{|W_i| + \alpha k}$ can also be used [9].

Statistical Testing

In this setting, a *statistic f* is a function that assigns a number to a pair of windows (W_1, W_2). The larger this number is, the more likely it is that the points in one window were generated from one distribution and the points in the other window were generated from a different distribution. A *statistical test* is a statistic f and a real number τ that serves as a threshold; when $f(W_1, W_2) \geq \tau$ then one can conclude that the points in W_1 were generated from a different distribution than the points in W_2.

For each $i \geq 1$ let W_i be the window that contains the points $x_i, \ldots, x_{i+m-1}$, so that W_1 is the set of the first m data points. To use a statistical test f with threshold τ, one computes the values $f(W_1, W_m)$, $f(W_1, W_{m+2})$, $f(W_1, W_{m+3})$,..., and signals a change the first time i such that $f(W_1, W_i) \geq \tau$. At this time, the current window W_i is considered to be a set of m points generated from the new distribution. The distribution may change again in the future, so one proceeds by computing the values of $f(W_i, W_{i+m})$, $f(W_i, W_{i+m+1})$, $f(W_{i+m+2})$, etc., until another change is detected.

Note that in order for it to be useful, a statistic should be easy to compute since a new value must be computed every time a new data point arrives. The value of the threshold τ should also be carefully chosen to avoid incorrect detections of change. A *false positive* is said to have occurred if the algorithm reports a change when the underlying distribution has not changed. Since a stream is a potentially unbounded source of data, false positives will occur and so instead of bounding the probability of a false positive, the goal is to choose a value of τ that bounds the *expected rate* of

false positives. Several statistics for detecting change and a method for choosing τ are presented next.

Let $\mathcal{A}$ be the collection of regions of interest. Let W and W' be two windows and let P and P' be the corresponding probability estimates: for any $A \in \mathcal{A}$, $P(A)$ is the estimated probability of region A based on window W and $P'(A)$ is the estimated probability of A based on window W'. The following statistics can be used in the change detection framework [13]:

$$d_{\mathcal{A}}(W, W') = \sup_{A \in \mathcal{A}} |P(A) - P'(A)|$$

$$\phi_{\mathcal{A}}(W, W') = \sup_{A \in \mathcal{A}} \frac{|P(A) - P'(A)|}{\sqrt{\min\left\{\frac{P(A)+P'(A)}{2}, 1 - \frac{P(A)+P'(A)}{2}\right\}}}$$

$$\Xi_{\mathcal{A}}(W, W') = \sup_{A \in \mathcal{A}} \frac{|P(A) - P'(A)|}{\sqrt{\frac{P(A)+P'(A)}{2}\left(1 - \frac{P(A)+P'(A)}{2}\right)}}$$

Note that when $\mathcal{A}$ is the set of intervals of the form $(-\infty, b)$ then $d_{\mathcal{A}}$ is also known as the Kolmogorov-Smirnov statistic. For any one of these statistics, the region $A \in \mathcal{A}$ which maximizes the statistic is the region where the change in observed probability is the most statistically significant; this region (or the ℓ most significant regions, depending on user preferences) and its change in probability is therefore the description of the change.

To use these statistics, one must determine the value of the threshold τ and the corresponding expected rate of false positives. To do this one can take advantage of the fact that for one-dimensional data, the worst-case behavior of the $d_{\mathcal{A}}$, $\phi_{\mathcal{A}}$, and $\Xi_{\mathcal{A}}$ statistics occur when the data are generated by continuous distributions and that the statistics behave in the same way for all continuous distributions [13]. This means that one can perform an offline computationally-intensive simulation to determine τ and then use this value for *any* one-dimensional stream afterwards.

To perform the simulation, a user must specify a test statistic f and four parameters: a window size m, a real number p between 0 and $\frac{1}{2}$, a large integer $q >= 2m$ (e.g., 1 million), and the number of repetitions B. For each repetition i, generate q points independently from any continuous distribution (e.g., from a Gaussian distribution with mean 0 and variance 1). Compute the value $\tau_i \equiv \max_{m \le j \le q-m+1} f(W_1, W_j)$ (this represents the largest value of the statistic f that would have encountered if this were the real data). After B repetitions, choose a value for the threshold τ such that τ is greater than $(1 - p)B$ of the values τ_1,...,τ_B. This value of τ guarantees that the probability of a false positive in the first q points is approximately p.

To compute the expected rate of false positives corresponding to τ, one first notes that false reports of change should occur in pairs for the following reason. Once a false positive has occurred, one has a window with points that are considered anomalous (since they caused a change to be reported); as new data points arrive, one compares the m most recent points (which are still generated from the original distribution) with this anomalous window using the chosen test statistic and therefore a second report of change should soon occur. Thus one can upper bound the expected number H of false positives in the first q points using the following probability distribution: $P(H = 2) = p$, $P(H = 4) = p^2$, etc., and $P(H = 0) = \frac{1-2p}{1-p}$. The expected value is $\frac{2p}{(1-p)^2}$ and one can use this as an upper bound on the number of false positives in the first q points. One can approximate the expected number of errors in the next q points also by $\frac{2p}{(1-p)^2}$ so that the expected rate of false positives is approximated by $\frac{2p}{q(1-p)^2}$.

When the regions of interest form a partition of the domain into k regions, other statistics, such as the KL-distance can be used [9]:

$$KL_{\mathcal{A}}(W, W') = \sum_{A \in \mathcal{A}} P(A) \log \frac{P(A)}{P'(A)}$$

where the probabilities $P(A)$ and $P'(A)$ are estimated using the Bayesian correction (i.e., $P(A) = \frac{|W \cap A| + \alpha}{|W| + \alpha k}$). Dasu et al. propose using the KL-distance with the following scheme (which uses a user-defined parameter γ): initially collect m data points for the window W_1 and use these points to create the regions of interest which partition the (possibly high-dimensional) domain; then compute $KL_{\mathcal{A}}(W_1, W_m)$, $KL_{\mathcal{A}}(W_1, W_{m+1})$, $KL_{\mathcal{A}}(W_1, W_{m+2})$, etc., and report a change whenever $m\gamma$ consecutive values of the statistic exceed a threshold τ. The value of τ depends on the points in W_1 and must be recomputed every time a change is detected. As before, τ is estimated via simulation.

To determine the value of τ, choose a parameter p $(0 < p < 1/2)$ and number of repetitions B. For each repetition i, use the probability distribution P estimated from W_1 to generate two windows V_1 and V_2 of m points each. Define τ_i to be $KL_{\mathcal{A}}(V_1, V_2)$. After B

repetitions, choose τ so that it is greater than $(1-p)B$ of the $\tau_1,\ldots,\tau_B$.

Key Applications

Data mining, network monitoring.

Future Directions

Key open problems include efficiently detecting change in high-dimensional spaces (see also [9]) and detecting change in streams where data points are not generated independently.

Cross-references

- ▶ Stream Data Management
- ▶ Stream Mining

Recommended Reading

1. Agarwal D., Barman D., Gunopulos D., Korn F., Srivastava D., and Young N. Efficient and effective explanation of change in hierarchical summaries. In Proc. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2007, pp. 6–15.
2. Babcock B., Babu S., Datar M., Motwani R., and Wisdom J. Models and issues in data stream systems. In Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2002, pp. 1–16.
3. Bansal R.K. and Papantoni-Kazakos P. An algorithm for detecting a change in a stochastic process. IEEE Trans. Inf. Theor., 32(2):227–235, 1986.
4. Basseville M. and Nikiforov I.V. Detection of Abrupt Changes: Theory and Application. Prentice-Hall, Englewood Cliffs, NJ, 1993.
5. Carlstein E., Müller H.-G., and Siegmund D. (eds.) Change-point problems. Institute of Mathematical Statistics, Hayward, CA, USA, 1994.
6. Chahrabarti S., Sarawagi S., and Dom B. Mining surprising patterns using temporal description length. In Proc. 24th Int. Conf. on Very Large Data Bases, 1998, pp. 606–617.
7. Chawathe S.S., Abiteboul S., and Widom J. Representing and querying changes in semi-structured data. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 4–13.
8. Chawathe S.S. and Garcia-Molina H. Meaningful change detection in structured data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 26–37.
9. Dasu T., Krishnan S., Venkatasubramanian S., and Yi K. An information-theoretic approach to detecting changes in multi-dimensional data streams. In Proc. 38th Symp. on the Interface of Statistics, Computing Science, and Applications, 2006.
10. Ganti V., Gehrke J., and Ramakrishnan R. Mining data streams under block evolution. SIGKDD Explorations, 3(2):1–10, 2002.
11. Glaz J. and Balakrishnan N. (eds.) Scan Statistics and Applications. Birkhäuser, Boston, USA, 1999.
12. Hulten G., Spencer L., and Domingos P. Mining time-changing data streams. In Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2001, pp. 97–106.
13. Kifer D., Ben-David S., and Gehrke J. Detecting change in data streams. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 180–191.
14. Kleinberg J.M. Bursty and hierarchical structure in streams. In Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2002, pp. 91–101.
15. Vovk V., Nouretdinov I., and Gammerman A. Testing exchangeability on-line. In Proc. 20th Int. Conf. on Machine Learning, 2003, pp. 768–775.

Channel-Based Publish/Subscribe

Hans-Arno Jacobsen
University of Toronto, Toronto, ON, Canada

Synonyms

Event channel; Event service

Definition

Channel-based publish/subscribe is a communication abstraction that supports data dissemination among many sources and many sinks. It is an instance of the more general publish/subscribe concept. The communication channel mediates between publishing data sources and subscribing data sinks and decouples their interaction.

Key Points

Publishing data sources submit messages to the channel and subscribing data sinks listen to the channel. All messages published to the channel are received by all subscribers listening on the channel. The channel broadcasts a publication message to all listening subscribers.

The channel decouples the interaction among publishing data sources and subscribing data sinks. The same decoupling characteristics as discussed under the general publish/subscribe concept apply here as well. Realizations of this model found in practice vary in the exact decoupling offered. To properly qualify as publish/subscribe, at least the anonymous communication style must exist. That is publishing clients must not be aware of who the subscribing clients are and how many subscribing clients exist, and vice versa. Thus, channel-based publish/subscribe enables the decoupled interaction of n sources with m sinks for $n, m \geq 1$.

Channel-based publish/subscribe systems often allow the application developer to create multiple logical channels, where each channel can be configured to offer different qualities-of-service to an application. Furthermore, a channel can be dedicated to the dissemination of messages pertaining to a specific subject or type. The channel-based publish/subscribe model does not support message filtering, except through the use of various channels to partition the publication space. It is the clients' responsibility to select the right channel for the dissemination of messages, which are sent to all listeners on the channel. This enables a limited form of filtering by constraining messages to be disseminated on one channel to a given message type. Finally, channel-based publish/subscribe is often coupled with client-side filtering, where messages are still broadcast throughout the channel, but filtered upon arrival at the data sink before passing to the application. More fine-grained filtering functionalities are provided by the other publish/subscribe models, such as the topic-based model and the content-based model.

In channel-based publish/subscribe, the publication data model is defined by the type of message the channel-based communication abstraction supports. This is often closely tied to the programming language or the library that implements the model.

Similarly, the subscription language model is defined by the programming language or library that allows the application developer to select channels to listen to, unless special provisions for subscriber-side filtering are offered. If supported, subscriber-side filtering can be arbitrarily complex, even selecting messages based on their content, as offered by the content-based publish/subscribe model.

Matching in the sense of evaluating a publication message against a set of subscriptions, as is common in the other publish/subscribe instantiations, does not occur in channel-based publish/subscribe.

Channel-based publish/subscribe systems are often coupled with different client interaction styles. These are the *push-style* and the *pull-style*. In the push-style, data sources initiate the transfer of messages to the channel, which delivers the messages to all listening data sinks. In the pull-style, data sinks initiate the message transfer by requesting messages from the channel, which requests any available messages from all connected data sources. Both interaction styles can also be combined. That is on one channel some clients can connect to the channel through the push-style, while others connect via the pull-style.

Channel-based publish/subscribe systems are distinguished by the qualities-of-service the channel offers to its clients, such as various degrees of reliability, persistence, real-time constraints, and message delivery guarantees. Channel-based publish/subscribe relates to topic-based publish/subscribe in that publishing a message to a channel is similar to associating a message with a topic, which could be the name or identity of the channel. However, in topic-based publish/subscribe this association is reflected in the message itself, while in channel-based publish/subscribe the association is indirect, reflected by selecting a channel, not part of the message. Also, topics can go far beyond channel identities, as discussed under the topic-based publish/subscribe concept. Examples that follow the channel-based publish/subscribe model are the CORBA Event Service [2], IP multicast [?], Usenet newsgroups [?], mailing lists, and group communication [?]. Elements of channel-based publish/subscribe can also be found in the Java Messaging Service [1], the OMG Data Dissemination Service [3], and other messaging middleware. However, these approaches are not directly following the channel-based model as described above; rather these approaches are enriched with elements of message queuing, topic-based publish/subscribe, and content-based publish/subscribe.

There are many applications of channel-based publish/subscribe. Examples include change notification, update propagation, information dissemination, newsgroups, email lists, and system management. Channel-based publish/subscribe serves well, if one or more entities have to communicate date to an anonymous group of receivers that may change over time, without the need of filtering messages within the channel.

In the literature the term channel-based publish/subscribe is not used uniformly. Abstractions that exhibit the above described functionality are also often referred to as event services, event channels, and simply channels. Messages disseminated to listeners are also often referred to as events. Publishing data sources are often referred to as publishers, producers or suppliers, and subscribing data sinks are often referred to as subscribers, consumers or listeners.

Cross-references

▶ Content-Based Publish/Subscribe
▶ Publish/Subscribe
▶ Topic-Based Publish/Subscribe

Recommended Reading

1. Hapner M., Burridge R., and Sharma R. Java Message Service. Sun Microsystems, version 1.0.2 edition, Nov 9, 1999.
2. OMG. Event Service Specification, version 1.2, formal/04–10–02 edition, October 2004.
3. OMG. Data Distribution Service for Real-time Systems, version 1.2, formal/07–01–01 edition, January 2007.

Chart

Hans Hinterberger
ETH Zurich, Zurich, Switzerland

Synonyms

Chart; Map; Diagram; Information graphic; Graph

Definition

A chart is an instrument to consolidate and display information.

The term is applied to virtually any graphic that displays information, be it a map used for navigation, a plan for military operations, a musical arrangement, barometric pressure, genealogical data, and even lists of tunes that are most popular at a given time.

Definitions of specialized charts typically include the graphical method on which the chart is based (e.g., bar chart) and or its application area (e.g., CPM chart) but it does not specify design principles. Tufte [2] introduces the notion of "chartjunk" and defines it to be that part of the chart which functions only as decoration, all of which is considered to redundant data ink.

Sometimes charts are reduced to refer to maps and diagrams, excluding graphs and tables.

Key Points

Because charts are used for different purposes there exist almost as many different types of charts as there are applications. Some maps, however, are more general in their use and therefore assigned to one of the following four categories: Graphs, Maps, Diagrams, and Tables. Each of these categories is broken down further into subcategories. In the category Diagrams, one therefore finds pie charts as well as flow charts or organization charts. A detailed categorization can be found in [1].

Charts are not only used to visualize data. They serve as a useful tool for many tasks where information is the main ingredient such as planning, presentation, analysis, monitoring.

Cross-references

▶ Data Visualization
▶ Diagram
▶ Graph
▶ Map
▶ Table
▶ Thematic Map

Recommended Reading

1. Harris R.L. Information Graphics: A Comprehensive Illustrated Reference, Oxford University Press, New York/ Oxford, 1999.
2. Tufte E.R. The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT, 1983.

Chase

Alin Deutsch[1], Alan Nash[2]
[1]University of California-San Diego, La Jolla, CA, USA
[2]Aleph One LLC, La Jolla, CA, USA

Definition

The chase is a procedure that takes as input a set Σ of constraints and an instance I. The chase does not always terminate, but if it does it produces as output an instance U with the following properties:

1. $U \models \Sigma$; that is, U satisfies Σ.
2. $I \rightarrow U$; that is, there is a homomorphism from I to U.
3. For every instance J (finite or infinite), if $J \models \Sigma$ and $I \rightarrow J$, then $U \rightarrow J$.

In [7], an instance that satisfies (1) and (2) above is called a *model of* Σ *and* I and an instance that satisfies (3) above is called *strongly universal*.

In summary, the chase is a procedure which – whenever it terminates – yields a strongly-universal model.

Comments

1. The set Σ of constraints is usually a set of tuple-generating dependencies (tgds) and equality-generating dependencies (egds) [5], or, equivalently, embedded dependencies [5,10]. However, the chase has been extended to wider classes of constraints and to universality under functions other than homomorphisms [6,7,9]. In this case, the chase often produces a strongly-universal model *set* (see below), instead of a single model.

2. It was noted in [7] that in database applications, *weak universality* (condition 3 above restricted to finite instances) would suffice. Nevertheless, the chase gives strong universality.

Historical Background

The term "chase" was coined in [14], where it was used to test the logical implication of dependencies (i.e., whether all databases satisfying a set Σ of dependencies must also satisfy a given dependency σ). The implication problem was one of the key concerns of dependency theory, with applications to automatic schema design. The chase was defined in [14] for the classes of functional, join and multivalued dependencies. Related chase formulations for various kinds of dependencies were introduced in [15,17]. The work [5] unified the treatment of the implication problem for various dependency classes by introducing and defining the chase for tuple-generating and equality-generating dependencies (sufficiently expressive to capture all prior dependencies).

Ancestors of the chase (introduced as unnamed algorithms) appear in [2–4]. [4] introduces tableaux, a pattern-based representation for relational queries, and shows how to check the equivalence of tableau queries in the presence of functional dependencies, with applications to query optimization. To this end, the tableaux are modified using an algorithm that coincides with the chase with functional dependencies. The same algorithm is used in [3] for minimization of tableaux under functional dependencies. This algorithm is extended in [2] to include also multivalued dependencies, for the purpose of checking whether the join of several relations is lossless (i.e., the original relations can be retrieved as projections of the join result).

The chase was extended to include disjunction and inequality in [9], and to arbitrary $\forall\exists$-sentences in [6]. Independently, [13] extended the chase to a particular case of disjunctive dependencies incorporating disjunctions of equalities between variables and constants (see also [12]). There are also extensions of the chase to deal with more complex data models beyond relational. The chase (and the language of embedded dependencies) is extended in [16] to work over complex values and dictionaries. For an excellent survey of the history of the chase prior to 1995, consult [1].

Foundations

A *tuple-generating dependency (tgd)* is a constraint σ of the form

$$\forall \bar{x}, \bar{y}\ (\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \beta(\bar{x}, \bar{z}))$$

where α and β are conjunctions of relational atoms. Furthermore, every variable in $\bar{x}$ appears in both α and β. The $\forall \bar{x}, \bar{y}$ prefix of universal quantifiers is usually omitted. If $\bar{z}$ is empty, then σ is *full*.

An *equality-generating dependency (egd)* is a constraint ϕ of the form

$$\forall x_1, x_2, \bar{y}\ (\alpha(x_1, x_2, \bar{y}) \rightarrow x_1 = x_2)$$

where α is a conjunction of relational atoms.

The chase is used on instances whose active domain consists of constants and *labeled nulls*. A *homomorphism* from A to B is denoted $A \rightarrow B$. It is a mapping h on the constants and nulls in A that (i) preserves constants (i.e., $h(c) = c$ for every constant c) and preserves relationships (i.e., for every tuple $R(x_1,...,x_n) \in A$, that is $R(h(x_1),...,h(x_n)) \in B$). Two instances A and B are *homomorphically equivalent* if $A \rightarrow B$ and $B \rightarrow A$.

The chase is a natural procedure for building strong universal models. Indeed, it turns out that checking for strong universality is undecidable as shown in [7]). In contrast, checking whether an instance is a model can be done efficiently. Therefore, it is natural to define any procedure for constructing strong universal models by steps which always preserve strong universality while attempting to obtain a model and then to check whether a model was indeed obtained. This is precisely what the chase does.

A tgd $\sigma \in \Sigma$ *fails* (or *applies*) on instance A and tuple $\bar{a}$ if there is tuple $\bar{b}$ in A such that the premise α of σ satisfies $A \models \alpha(\bar{a}, \bar{b})$, yet there is no tuple $\bar{c}$ in A such that the conclusion β of σ satisfies $A \models \beta(\bar{a}, \bar{c})$. Assume that the instance A' is obtained by adding to A the tuples in $\beta(\bar{a}, \bar{n})$ where $\bar{n}$ is a tuple of new nulls. Then A' is the result of *firing* σ on A, $\bar{a}$. Notice that $A \subseteq A'$ and that σ does not fail on A', $\bar{a}$. It is easy to verify that if A is strongly universal for Σ and I, then so is A' (towards this, it is essential that all the nulls in $\bar{n}$ be new and distinct).

An egd $\sigma \in \Sigma$ *fails* (or *applies*) on instance A and values a_1, a_2 if there is tuple $\bar{b}$ in A such that the premise α of σ satisfies $A \models \alpha(a_1, a_2, \bar{b})$, yet $a_1 \neq a_2$. If a_2 is a null a_2 is replaced everywhere in A with a_1

to obtain A', then say that A' is the result of *firing* σ on A, a_1, a_2. Notice that $A \rightarrow A'$ and that σ does not fail on A', a_1, a_1. It is easy to verify that if A is strongly universal for Σ and I, then so is A'. If a_2 is a constant, but a_1 is null, then it is possible to replace a_1 everywhere in A with a_2 instead. However, if both a_1 and a_2 are constants, then it is not possible to satisfy σ and preserve strong universality and the chase *fails*.

The standard chase procedure proceeds as follows.

1. Set $A_0 = I$.
2. Repeat the following:
   a. If A_n is a model of Σ and I, stop and return A_n.
   b. Otherwise, there must be either
      i. a tgd σ and $\bar{a}$ such that σ fails on A, $\bar{a}$, or
      ii. an egd σ' and a_1, a_2 such that σ' fails on A, a_1, a_2.

   Obtain A_{n+1} by picking one such σ and $\bar{a}$ and firing σ on A_n, $\bar{a}$, or by picking one such σ' and a_1, a_2 and firing σ' on A, a_1, a_2. (This is one *chase step* of the standard chase.)

Notice that, at every chase step, there may be a choice of σ and $\bar{a}$, respectively σ' and a_1, a_2. How these choices are picked is often left unspecified and in that case the standard chase is non-deterministic. The chase *terminates* if A_n is a model of Σ and I for some n.

The chase of instance I with tgds Σ produces a sequence of instances $I = A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$ such that every A_i is strongly universal for Σ and I. The chase with tgds and egds produces a sequence $I = A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow \ldots$ such that every A_i is strongly universal for Σ and I. In the presence of egds, it is no longer the case that $A_i \subseteq A_j$ for $i \leq j$ and there is the additional complication that a chase step may fail. The chase for tgds and egds is described in more detail in [1].

Example 1 Consider the schema consisting of two relations:

1. employee Emp(ss#, name, dept#), with social security, name, and dept. number, and
2. department Dept(dept#, name, location, mgr#), with dept. number, name, location, and its manager's social security number.

Assume that Σ consists of the constraints

σ_1: dept# is a foreign key in Emp,
σ_2: mgr# is a foreign key in Dept, and
σ_3: every manager manages his own department.

(This omits the constraints that say that ss# is a key for Emp and that dept# is a key for Dept to keep the example simple.) These constraints can be written as follows (where σ_1 and σ_2 are tgds and σ_3 is an egd):

σ_1: Dept(d, e, ℓ, m) $\rightarrow \exists n, d'$Emp($m$, n, d'),
σ_2: Emp(s, n, d) $\rightarrow \exists e, \ell, m$Dept($d$, e, ℓ, m), and
σ_3: Dept(d, e, ℓ, m), Emp(m, n, d') $\rightarrow d = d'$.

Consider the initial instance

$$I_0 = \text{Dept}(1, \text{"HR"}, \text{"somewhere"}, 333 - 33 - 3333)$$

containing a single tuple. Then in the first step of the chase, σ_1 fires, giving

$$I_1 = \{\text{Dept}(1,\}\text{HR}\},\}\text{somewhere}\}, 33), \text{Emp}(33, \alpha, \beta)\}$$

where α and β are labeled nulls. In the second step, both σ_2 and σ_3 apply. If σ_3 fires, then β is set to 1 and yields

$$I_2 = \{\text{Dept}(1, \text{"HR"}, \text{"somewhere"}, 33), \text{Emp}(33, \alpha, 1)\}.$$

Since I_2 satisfies Σ, the chase terminates. However, if instead at the second step σ_2 fires, then it gives

$$I'_2 = \{\text{Dept}(1, \text{"HR"}, \text{"somewhere"}, 33), \text{Emp}(33, \alpha, \beta),\ \text{Dept}(\beta, \gamma, \delta, \epsilon\}$$

where γ, δ, and ε are new nulls. In this case, it is possible to continue firing σ_1, σ_2, and σ_3 in such a way as to obtain a chase that does not terminate, perpetually introducing new nulls.

If the standard chase (or any other chase listed below) terminates, it yields a strongly-universal model of Σ and I and it is straightforward to verify that all such models are homomorphically equivalent. Therefore the result of the standard chase is unique up to homomorphic equivalence. However, the choice of what constraint to fire and on what tuple may affect whether the chase terminates or not.

There are several variations of the chase, which shall be called here the *standard* chase, the *parallel* chase, and the *core* chase. The standard chase was described above. In the *parallel chase*, at every chase step σ is fired on A_n, $\bar{a}$ for all pairs $(\sigma, \bar{a})$ such that σ fails on A, $\bar{a}$.

One writes I^{Σ} for the result of the chase on Σ and I, if the chase terminates. In that case, one says that I^{Σ} is defined. In general, it holds that if $A \rightarrow B$, then $A^{\Sigma} \rightarrow B^{\Sigma}$, whenever the latter are defined.

It was shown in [7] that the standard chase is incomplete, in the following sense: it may be that Σ and I have a strongly-universal model, yet the standard chase does not terminate. The parallel chase is also incomplete in this sense. In contrast, the *core chase* introduced in [7] is complete: if a strongly universal-model exists, the core chase terminates and yields such a model. A chase step of the core chase consists of one chase step of the parallel chase, followed by computing the core of the resulting instance.

Any of the above mentioned variations of the chase can be applied to sets of constraints which consist of

1. tgds only
2. tgds and egds
3. tgds and egds with disjunctions
4. tgds and egds with disjunctions and negation which are equivalent to general $\forall\exists$ sentences

The chase with tgds and egds has been described above. The chase has been extended to handle disjunction and negation. In this case, it gives not a single model, but a set S of models which is strongly universal, in the sense that for any model J (finite or infinite) of Σ and I, there is a model $A \in S$ such that $A \rightarrow J$. Such a set arises from a single initial model by branching due to disjunction. For example, consider the set Σ with the single disjunctive tgd

$$\sigma : R(x) \rightarrow S(x) \vee T(x)$$

and the instance I containing the single fact $R(1)$. Clearly every model of Σ and I, must contain either $S(1)$ or $T(1)$. It is easy to verify that the set $S = \{I_1, I_2\}$ where $I_1 = \{R(1), S(1)\}$ and $I_2 = \{R(1), T(1)\}$ is strongly universal for I and Σ, but no proper subset of S is. The disjunctive chase with Σ on I consists of a single step, which produces not a single model, but the set S of models. Intuitively, whenever a disjunctive tgd fires on a set W of models, it produces, for every instance $A \in W$, one instance for every disjunct in its conclusion. For details, to see how negation is handled, and to see how universality for functions other than homomorphisms is achieved, see [6,7].

It was shown in [7] that it is undecidable whether the standard, parallel, or core chase with a set of tgds terminates. A widely-applicable, efficiently-checkable condition on a set Σ of tgds, which is sufficient to guarantee that the chase with Σ on any instance I terminates, was introduced in [9,11]. A set of tgds satisfying this condition is called *weakly acyclic* in [11] and is said to have *stratified witnesses* in [9]. A more widely-applicable condition, also sufficient for chase termination, was introduced in [7], where a set of tgds satisfying this condition is called *stratified*.

Key Applications

The chase has been used in many applications, including

- Checking containment of queries under constraints (which in turn is used in such query rewriting tasks as minimization, rewriting using views, and semantic optimization)
- Rewriting queries using views
- Checking implication of constraints
- Computing solutions to data exchange problems
- Computing certain answers in data integration settings

To check whether a query P is contained in a query Q under constraints Σ, written $P \sqsubseteq_\Sigma Q$, it is sufficient to (1) treat P as if it was an instance in which the free variables are constants and the bound variables are nulls (this is known as the "frozen instance" or "canonical database" [1] corresponding to P) (2) chase it with Σ, and if this chase terminates to yield P^Σ (3) check whether the result of this chase is contained in Q, written $P^\Sigma \sqsubseteq Q$. In symbols, if the chase described above terminates, then

$$P \sqsubseteq_\Sigma Q \text{ iff } P^\Sigma \sqsubseteq Q.$$

That is, the chase reduces the problem of query containment under constraints to one of query containment without constraints.

To check whether a set Σ of tgds implies a tgd σ of the form

$$\forall \bar{x}, \bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\beta(\bar{x}, \bar{z}))$$

which is logically equivalent to

$$\forall \bar{x}\,(\underbrace{\exists \bar{y}\alpha(\bar{x}, \bar{y})}_{Q_\alpha(\bar{x})} \rightarrow \underbrace{\exists \bar{z}\beta(\bar{x}, \bar{z})}_{Q_\beta(\bar{x})}),$$

it suffices to check whether the query Q_α in the premise of σ is contained under the constraints Σ in the query Q_β in the conclusion of σ. That is, if $Q_\alpha{}^\Sigma$ is defined, then

$$\Sigma \models \sigma \text{ iff } Q_\alpha \sqsubseteq_\Sigma Q_\beta \text{ iff } Q_\alpha^\Sigma \sqsubseteq Q_\beta.$$

The chase was also employed to find equivalent rewritings of conjunctive queries using conjunctive query views, in the presence of constraints. Given a set $\mathcal{V}$ of conjunctive query views and a conjunctive query Q, one can construct, using the chase, a query R expressed in terms of $\mathcal{V}$, such that Q has some equivalent rewriting using $\mathcal{V}$ if and only if R is itself such a rewriting. Moreover, every minimal rewriting of Q is guaranteed to be a sub-query of R. The algorithm for constructing R and exploring all its sub-queries is called the *Chase&Backchase (CB)* [8], and it is sound and complete for finding all minimal rewritings under a set Σ of embedded dependencies, provided the chase with Σ terminates [9]. The CB algorithm constructs R by simply (i) constructing a set $\Sigma_{\mathcal{V}}$ of tgds extracted from the view definitions, and (ii) chasing Q with $\Sigma \cup \Sigma_{\mathcal{V}}$ and restricting the resulting query to only the atoms using views in $\mathcal{V}$.

In [11] is was shown that the certain answers to a union Q of conjunctive queries on a ground instance I under a set Σ of source-to-target tgds and target tgds and egds can be obtained by computing $Q(U)$ – where U is a *universal solution* for I under Σ – then discarding any tuples with nulls. Universal solutions, which are the preferred solutions to materialize in data exchange, are closely related to strongly-universal models [7] and it was shown in [11] that they can be obtained using the chase.

Cross-references

- ▶ Data Exchange
- ▶ Data Integration
- ▶ Database Dependencies
- ▶ Equality-Generating Dependencies
- ▶ Query Containment
- ▶ Query Optimization
- ▶ Query Rewriting
- ▶ Query Rewriting Using Views
- ▶ Tuple-Generating Dependencies

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of Databases. Addison Wesley, Reading, MA, 1995.
2. Aho A.V., Beeri C., and Ullman J.D. The theory of joins in relational databases. ACM Trans. Database Syst., 4(3):297–314, 1979.
3. Aho A.V., Sagiv Y., and Ullman J.D. Efficient optimization of a class of relational expressions. ACM Trans. Database Syst. 4(4):435–454, 1979.
4. Aho A.V., Sagiv Y., and Ullman J.D. Equivalence of relational expressions. SIAM J. Comput., 8(2):218–246, 1979.
5. Beeri C. and Vardi M.Y. A proof procedure for data dependencies. J. ACM, 31(4):718–741, 1984.
6. Deutsch A., Ludaescher B., and Nash A. Rewriting queries using views with access patterns under integrity constraints. In Proc. 10th Int. Conf. on Database Theory, 2005, pp. 352–367.
7. Deutsch A., Nash A., and Remmel J. The chase revisited. In Proc. 27th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2008, pp. 149–158.
8. Deutsch A., Popa L., and Tannen V. Physical Data Independence, Constraints, and Optimization with Universal Plans. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 459–470.
9. Deutsch A. and Tannen V. XML queries and constraints, containment and reformulation. Theor. Comput. Sci. 336(1): 57–87, 2005, preliminary version in ICDT 2003.
10. Fagin R. Horn clauses and database dependencies. J. ACM, 29(4):952–985, 1982.
11. Fagin R., Kolaitis P.G., Miller R.J., and Popa L. Data Exchange: Semantics and Query Answering. Theor. Comput. Sci., 336 (1):89–124, 2005, preliminary version in PODS 2005.
12. Fuxman A., Kolaitis P.G., Miller R.J., and Tan W.C. Peer Data Exchange. ACM Trans. Database Syst., 31(4):1454–1498, 2006, preliminary version in PODS 2005.
13. Grahne G. and Mendelzon A.O. Tableau Techniques for Querying Information Sources through Global Schemas. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 332–347.
14. Maier D., Mendelzon A.O., and Sagiv Y. Testing implications of data dependencies. ACM Trans. Database Syst., 4(4):455–469, 1979.
15. Maier D., Sagiv Y., and Yannakakis M. On the complexity of testing implication of functional and join dependencies. J. ACM, 28(4):680–695, 1981.
16. Popa L. and Tannen V. An Equational Chase for Path-Conjunctive Queries, Constraints, and Views. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 39–57.
17. Vardi M. Inferring multivalued dependencies from functional and join dependencies. Acta Informatica, 19:305–324, 1983.

Checkpoint

- ▶ Logging and Recovery

Checksum and Cyclic Redundancy Check Mechanism

Kenichi Wada
Hitachi, Ltd, Tokyo, Japan

Synonyms

Cyclic Redundancy Check (CRC)

Definition

Checksum and CRC are schemes for detecting the errors of data which occur during transmission or storage. The data computed and appended to original data in order to detect errors are also referred as checksum and CRC.

A checksum consists of a fixed number of bits computed as a function of the data to be protected, and is appended to the data. To detect errors, the function is recomputed, and the result is compared to that appended to the data. Simple implementation of checksum is to divide the data into same length bits chunk and to make exclusive-or of all chunks. Cyclic redundancy check mechanism exploits mathematical properties of cyclic codes. Specifically, CRC uses polynomial devisor circuits with a given generator polynomial so as to obtain the remainder polynomial. The remainder is similarly appended to the original data for transmission and storage, and then utilized for error detection. CRC can be used as a kind of checksum.

Key Points

CRC is usually expressed by the use of binary polynomials due to mathematical convenience. When original data $M(x)$ is given, basic CRC mechanism calculates redundancy data $R(x)$ by using a pre-defined generator polynomial $G(x)$. That is, supposing the degree of $G(x)$ is m, a polynomial $M(x) * x^m$ is divided by $G(x)$ and the remainder is used for R(x) such that a concatenated polynomial $T(x) = M(x) * x^m + R(x)$ is divisible by $G(x)$. The obtained $T(x)$ is used for transmission or storage. For error detection, CRC mechanism similarly checks the divisibility of $T(x)$ by $G(x)$. These encoding and detection processes can be implemented by using multi-level shift register circuits.

Given below is an example of CRC calculation. Assume that a generator polynomial and original data are given as follows.

$$\mathrm{G}(x) = x^3 + x + 1 \quad (\textit{binary expression}: 1011)$$

$$M(x) = x^4 + 1 \quad (10001)$$

In this case, a remainder polynomial $R(x)$ can be obtained by dividing $M(x) * x^3$ by $G(x)$.

$$R(x) = x \quad (010)$$

Therefore, the resulting data $T(x)$ can be obtained as follows.

$$T(x) = x^7 + x^3 + x \quad (10001010)$$

Theoretically, CRC is capable of detecting m-bit long or shorter bust errors. This property is suitable for communication infrastructure and storage infrastructure, which often introduce burst errors rather than random errors.

Cross-references

► Disk

Recommended Reading

1. Houghton A. Error Coding for Engineers. Kluwer Academic Publishers, Dordrecht, 2001.
2. Sweeney P. Error Control Coding from Theory to Practice. Wiley, NY, 2002.

Choreography

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

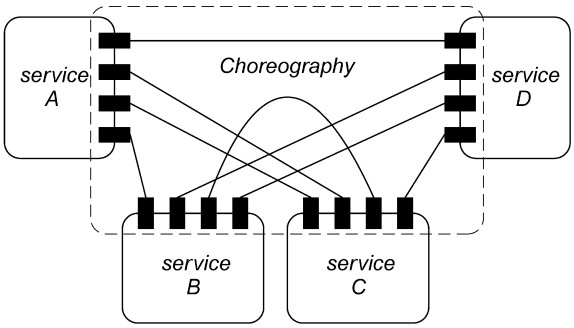
Definition

In a service oriented architecture (SOA) services are interacting by exchanging messages, i.e., by combining services more complex services are created. Choreography is concerned with the composition of such services seen from a global viewpoint focusing on the common and complementary observable behavior. Choreography is particulary relevant in a setting where there is not a single coordinator.

Key Points

The terms orchestration and choreography describe two aspects of integrating services to create business processes [1,3]. The two terms overlap somewhat and the distinction is subject to discussion. Orchestration and choreography can be seen as different "perspectives." Choreography is concerned with the exchange of messages between those services. Orchestration is concerned with the interactions of a single service with its environment.

Figure 1 illustrates the notion of choreography. The dashed area shows the focal point of choreography,



Choreography. Figure 1. Choreography.

i.e., the aim is to establish a "contract" containing a "global" definition of the constraints under which messages are exchanged. Unlike orchestration, the viewpoint is not limited to a single service. The Web Services Choreography Description Language (WS-CDL, cf. [2]) and the Web Service Choreography Interface (WSCI) are two languages aiming at choreography. Since the focus is on agreement rather than enactment, choreography is quite different from traditional workflow languages. The goal is not to control and enact but to coordinate autonomous parties. Some characterize choreography as "Dancers dance following a global scenario without a single point of control" to emphasize this distinction.

Cross-references

- ▶ BPEL
- ▶ Business Process Management
- ▶ Orchestration
- ▶ Web Services
- ▶ Workflow Management

Recommended Reading

1. Dumas M., van der Aalst W.M.P., and ter Hofstede A.H.M. Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley, New York, 2005.
2. Kavantzas N., Burdett D., Ritzinger G., Fletcher T., and Lafon Y. Web Services Choreography Description Language Version 1.0 (W3C Candidate Recommendation). http://www.w3.org/TR/2005/CR-ws-cdl-10-20051109/, 2005.
3. Weske M. Business Process Management: Concepts, Languages, Architectures. Springer, Berlin, 2007.

Chronicle Recognition

- ▶ Event Detection

Chronon

Curtis Dyreson
Utah State University, Logan, UT, USA

Synonyms

Instant; Moment; Time quantum; Time unit

Definition

A chronon is the smallest, discrete, non-decomposable unit of time in a temporal data model. In a one-dimensional model, a chronon is a *time interval* or *period*, while in an *n*-dimensional model it is a non-decomposable region in *n*-dimensional time. Important special types of chronons include valid-time, transaction-time, and bitemporal chronons.

Key Points

Data models often represent a time line by a sequence of non-decomposable, consecutive time periods of identical duration. These periods are termed chronons. A data model will typically leave the size of each particular chronon unspecified. The size (e.g., one microsecond) will be fixed later by an individual application or by a database management system, within the restrictions posed by the implementation of the data model. The number of chronons is finite in a bounded model (i.e., a model with a minimum and maximum chronon), or countably infinite otherwise. Consecutive chronons may be grouped into larger intervals or segments, termed *granules*; a chronon is a granule at the lowest possible granularity.

Cross-references

- ▶ Temporal Granularity
- ▶ Time Domain
- ▶ Time Instant
- ▶ Time Interval

Recommended Reading

1. Dyreson C.E. and Snodgrass R.T. The base-line clock. In The TSQLZ temporal query language, Kluwer, pp. 73–92, 1987.
2. Dyreson C.E. and Snodgrass R.T. Timestamp Semantics and Representation. Inf. Syst., 18(3):143–166, 1993.

CIFS

- ▶ Storage Protocols

Cipher

► Data Encryption

Citation

Prasenjit Mitra
The Pennsylvania State University, University Park, PA, USA

Synonyms

Reference; Bibliography

Definition

A citation is a reference from one article to another article. A citation is a record that consists of the names of the authors, the title of the referred article, the time and place of publication, as well as various other fields. The fields in the citation should collectively specify unambiguously where the full text of the referred article could be obtained. Typically, all citations are presented at the end of the referring article. However, articles in certain domains list the citations as footnotes in the pages where the reference occurs. Citations can range from references to be to single articles or to entire books.

Key Points

Often, authors have to refer to knowledge that is derived from another work. For example, when quoting text from another article or book, the author must specify from which article or book the quotation is obtained. Authors need to refer to other works in order to point out preliminary information on the shoulders of which the current treatise stands, to refer to related work and contrast the current work with previous works, etc. A citation is used for primarily two purposes: (i) to provide a reference to an article or book such that the reader can retrieve the article or book easily and read the article to gain additional knowledge, and (ii) to provide credit (or discredit for "negative" citations) to the authors of the works that are being cited.

There are various widely used formats for citations. Citation formats vary by discipline; typically a discipline adheres to one (or a few) "style-guide" that indicates what fields should be mentioned in a citation and how the fields should be formatted and presented. Recently, with the proliferation of electronic documents published over the World-Wide-Web, citations to Uniform Resource Locators (URLs) of websites are increasingly common. Unlike printed articles and books, websites are dynamic and can change frequently. Therefore, in order to specify precisely which version of the webpage was being referred, apart from the publication date, authors usually provide the date on which the website was accessed.

Citations analysis has been performed to identify the impact of published articles. Because different authors use different formats, automatic analysis of citations requires *citation matching*. Citation matching helps identify which different citations formatted differently refer to the same article or book. The term *bibliometrics* is used to refer to metrics designed based on citation analysis. Citation indexing for academic journals was popularized by Eugene Garfield [1,2]. A citation index contains the information about which document cites which. The term *co-citation* refers to the frequency with which two documents are cited together [3]. Today, Google Scholar (http://scholar.google.com) provides a readily-available collection of indexed citations on the web.

Cross-references

► Digital Library

Recommended Reading

1. Garfield E. Citation Indexing: Its Theory and Application in Science, Technology, and Humanities. Wiley, New York, NY, USA, 1979.
2. Garfield E. Citation analysis as a tool in journal evaluation: journals can be ranked by frequency and impact of citations for science policy studies. Science, 178(4060):471–479, 1972.
3. Small H. Co-citation in the scientific literature: a new measure of the relationship between two documents. J. Am. Soc. Inf. Sci., Wiley Periodicals, 24(4):265–269, 1973.

CLARA (Clustering LARge Applications)

► K-Means and K-Medoids

CLARANS (Clustering Large Applications Based Upon Randomized Search)

► K-Means and K-Medoids

Classification

Ian H. Witten
University of Waikato, Hamilton, New Zealand

Synonyms

Classification learning; Supervised learning; Learning with a teacher, Concept learning; Statistical decision techniques

Definition

In *Classification learning*, an algorithm is presented with a set of classified examples or "instances" from which it is expected to infer a way of classifying unseen instances into one of several "classes". Instances have a set of features or "attributes" whose values define that particular instance. Numeric prediction, or "regression," is a variant of classification learning in which the class attribute is numeric rather than categorical. Classification learning is sometimes called *supervised* because the method operates under supervision by being provided with the actual outcome for each of the training instances. This contrasts with clustering where the classes are not given, and with association learning which seeks any association – not just one that predicts the class.

Historical Background

Classification learning grew out of two strands of work that began in the 1950s and were actively pursued throughout the 1960s: statistical decision techniques and the Perceptron model of neural networks. In 1955, statisticians Bush and Mosteller published a seminal book *Stochastic Models for Learning* which modeled in mathematical terms the psychologist B. F. Skinner's experimental analyses of animal behavior using reinforcement learning [2]. The "perceptron" was a one-level linear classification scheme developed by Rosenblatt around 1957 and published in his book *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* [10]. In a response published in 1969, Minsky and Papert argued that perceptrons were simplistic in terms of their representational capability and had been greatly over-hyped as potentially universal learning machines [6]. This scathing response by widely-respected artificial intelligence pioneers dampened research in neural nets and machine learning in general. Meanwhile, in 1957 others were investigating the application of Bayesian decision schemes to pattern recognition; the general conclusion was that full Bayesian models were prohibitively expensive. In 1960 Maron investigated in the context of information retrieval what has since become known as the "naïve Bayes" approach, which assumes independence between attributes notwithstanding overwhelming evidence to the contrary [5]. Other early machine learning work was buried in cybernetics, the study of feedback and derived concepts such as communication and control in living and artificial organisms. Throughout the 1960s classification learning applied to pattern recognition was the central thread of the embryo field of machine learning, as underlined by the subtitle of Nilsson's 1965 landmark book *Learning Machines – Foundations of Trainable Pattern-Classifying Systems* [7].

Symbolic learning techniques began to recover from the doldrums in the late 1970s, with influential and almost simultaneous publications by Breiman et al. on *classification and regression trees* (the CART system) [1] and Quinlan on *decision tree induction* (the ID3 and later C4.5 systems) [8,9]. Whereas Breiman was a statistician, Quinlan was an experimental computer scientist who first used decision trees not to generalize but to condense large collections of chess end-games. Their work proceeded independently, and the similarities remained unnoticed until years later. CART (by default) produces multivariate trees whose tests can involve more than one attribute: these are more accurate and smaller than the univariate trees produced by Quinlan's systems, but take longer to generate.

The first workshop devoted to machine learning was held in 1980 at Carnegie-Mellon University. Further workshops followed in 1983 and 1985. These invitation-only events became an open conference in 1988. Meanwhile the journal *Machine Learning* was established in 1986. By the 1990s the subject had become the poster child of artificial intelligence – a successful, burgeoning, practical technology that eschewed the classical topics of general knowledge representation, logical deduction, theorem proving, search techniques, computational linguistics, expert systems and philosophical foundations that still characterize the field today. Classification learning, which forms the core of machine learning, outgrew its behaviorist and neurological roots and moved into the practical realm of database systems.

Early work focused on the *process* of learning – learning curves, the possibility of sustained learning, and the like – rather than the *results* of learning. However,

with the new emphasis on applications, objective techniques of empirical testing began to supplant the scenario-based style of evaluation that characterized the early days. A major breakthrough came during the 1980s, when researchers finally realized that evaluating a learning system on its training data gave misleading results, and instead put the subject on a secure statistical footing.

Foundations

One of the most instructive lessons learned since the renaissance of classification in the 1980s is that simple schemes often work very well. Today, practitioners strongly recommend the adoption of a "simplicity-first" methodology when analyzing practical datasets. There are many different kinds of simple structure that datasets can exhibit. One dataset might have a single attribute that does all the work, the others being irrelevant or redundant. Alternatively, the attributes might contribute independently and equally to the final outcome. Underlying a third dataset might be a simple contingent structure involving just a few attributes. In a fourth, a few independent rules may govern the assignment of instances to classes. In a fifth, classifications appropriate to particular regions of instance space might depend on the distance between the instances themselves. A sixth might exhibit dependence among numeric attributes, determined by a sum of attribute values with appropriately chosen weights. This sum might represent the final output for numeric prediction, or be compared to a fixed threshold in a binary decision setting. Each of these examples leads to a different style of method suited to discovering that kind of structure.

Rules Based on a Single Attribute

Even when instances have several attributes, the classification decision may rest on the value of just one of them. Such a structure constitutes a set of rules that all test the same attribute (or, equivalently, a one-level decision tree). It can be found by evaluating the success, in terms of the total number of errors on the training data, of testing each attribute in turn, predicting the most prevalent class for each value of that attribute. If an attribute has many possible values – and particularly if it has numeric values – this may "overfit" the training data by generating a rule that has almost as many branches as there are instances. Minor modifications to the scheme overcome this problem.

A startling discovery published in 1993 was that "very simple classification rules perform well on most commonly used datasets" [3]. In an empirical investigation of the accuracy of rules that classify instances on the basis of a single attribute, on most standard datasets the resulting rule was found to be as accurate as the structures induced by the majority of machine learning systems – which are far more complicated. The moral? – always compare new methods with simple baseline schemes.

Statistical Modeling (see entry Bayesian Classification)

Another simple technique is to use all attributes and allow them to make contributions to the decision that are *equally important* and *independent* of one another, given the class. Although grossly unrealistic – what makes real-life datasets interesting is that the attributes are certainly not equally important or independent – it leads to a statistically-based scheme that works surprisingly well in practice. Employed in information retrieval as early as 1960 [5], the idea was rediscovered, dubbed "naïve Bayes," and introduced into machine learning 30 years later [4]. Despite the disparaging moniker it works well on many actual datasets. Over-reliance on the independence of attributes can be countered by applying attribute selection techniques.

Divide and Conquer Technique (see entry Decision Tree Classification)

The process of constructing a decision tree can be expressed recursively. First, select an attribute to use at the root, and make a branch for each possible value. This splits the instance set into subsets, one for every value of the attribute. Now repeat the process recursively for each branch, using only those instances that actually reach the branch. If all instances at a node have the same classification, stop developing that part of the tree. This method of "top-down induction of decision trees" was explored and popularized by Quinlan [8,9]. The nub of the problem is to select an appropriate attribute at each stage. Of many heuristics that have been investigated, the dominant one is to measure the expected amount of information gained by knowing that attribute's actual value. Having generated the tree, it is selectively pruned back from the leaves to avoid over-fitting. A series of improvements include ways of dealing with numeric attributes, missing values, and noisy data; and generating rules from trees.

Covering Algorithms (see entry Rule-Based Classification)

Classification rules can be produced by taking each class in turn and seeking a rule that covers all its instances, at the same time excluding instances not in the class. This bottom-up approach is called *covering* because at each stage a rule is identified that "covers" some of the instances. Although trees can always be converted into an equivalent rule set, and vice versa, the perspicuity of the representation often differs. Rules can be symmetric whereas trees must select one attribute to split on first, which can produce trees that are much larger than an equivalent set of rules. In the multiclass case a decision tree split takes account of all classes and maximizes the information gained, whereas many rule generation methods concentrate on one class at a time, disregarding what happens to the others.

Instance-Based Learning (see entry Nearest Neighbor Classification)

Another approach is to store training instances verbatim and, given an unknown test instance, use a distance function to determine the closest training instance and predict its class for the test instance. Suitable distance functions are the Euclidean or Manhattan (city-block) metric; attributes should be normalized to lie between 0 and 1 to compensate for scaling effects. For nominal attributes that assume symbolic rather than numeric values, the distance between two values is 1 if they are not the same and 0 otherwise. In the *k*-nearest neighbor strategy, some fixed number of nearest neighbors – say five – are located and used together to determine the class of the test instance by majority vote. Another way of proofing the database against noise is to selectively and judiciously choose the exemplars that are added. Nearest-neighbor classification was notoriously slow until advanced data structures like *k*D-trees were applied in the early 1990s.

Linear Models (see entry Linear Regression)

When the outcome and all attributes are numeric, linear regression can be used. This expresses the class as a linear combination of the attributes, with weights that are calculated from the training data. Linear regression has been popular in statistical applications for decades. If the data exhibits a nonlinear dependency, the best-fitting straight line will be found, where "best" is interpreted in the least-mean-squared-difference sense. Although this line may fit poorly, linear models can serve as building blocks for more complex learning schemes.

Linear Classification (see entry Neural Networks, Support Vector Machine)

The idea of linear classification is to find a hyperplane in instance space that separates two classes. (In the multiclass case, a binary decision can be learned for each pair of classes.) If the linear sum exceeds zero the first class is predicted; otherwise the second is predicted. If the data is linearly separable – that is, it can be separated perfectly using a hyperplane – the perceptron learning rule espoused by Rosenblatt is guaranteed to find a separating hyperplane [10]. This rule adjusts the weight vector whenever the prediction for a particular instance is erroneous: if the first class is predicted the instance (expressed as a vector) is added to the weight vector (making it more likely that the result will be positive next time around); otherwise the instance is subtracted.

There have been many powerful extensions of this basic idea. Support vector machines use linear decisions to implement nonlinear class boundaries by transforming the input using a nonlinear mapping. Multilayer perceptrons connect many linear models in a hierarchical arrangement that can represent nonlinear decision boundaries, and use a technique called "back-propagation" to distribute the effect of errors through this hierarchy during training.

Missing Values

Most datasets encountered in practice contain missing values. Sometimes different kinds are distinguished (e.g., unknown vs. unrecorded vs. irrelevant values). They may occur for a variety of reasons. There may be some significance in the fact that a certain instance has an attribute value missing – perhaps a decision was taken not to perform some test – and that might convey information about the instance other than the mere absence of the value. If this is the case, *not tested* should be recorded as another possible value for this attribute. Only someone familiar with the data can make an informed judgment as to whether a particular value being missing has some significance or should simply be coded as an ordinary missing value. For example, researchers analyzing medical databases have noticed that cases may, in some circumstances, be diagnosable strictly from the tests that a doctor decides to make,

regardless of the outcome of the tests. Then a record of which values are "missing" is all that is needed for a complete diagnosis – the actual measurements can be ignored entirely!

Meta-Learning

Decisions can often be improved by combining the output of several different models. Over the past decade or so the techniques of *bagging*, *boosting*, and *stacking* have been developed that learn an ensemble of models and deploy them together. Their performance is often astonishingly good. Researchers have struggled to understand why, and during that struggle new methods have emerged that are sometimes even better. For example, whereas human committees rarely benefit from noisy distractions, shaking up bagging by adding random variants of classifiers can improve performance. Boosting – perhaps the most powerful of the three methods – is related to the established statistical technique of additive models, and this realization has led to improved procedures.

Combined models share the disadvantage of being rather hard to analyze: they can comprise dozens or even hundreds of individual learners and it is not easy to understand in intuitive terms what factors are contributing to the improved decisions. In the last few years methods have been developed that combine the performance benefits of committees with comprehensible models. Some produce standard decision tree models; others introduce new variants of trees that provide optional paths.

Evaluation

For classification problems, performance is naturally measured in terms of the *error rate*. The classifier predicts the class of each test instance: if it is correct, that is counted as a success; if not, it is an error. The error rate is the proportion of errors made over a whole set of instances, and reflects the overall performance of the classifier. Performance on the training set is definitely *not* a good indicator of expected performance on an independent test set. A classifier is *overfitted* to a dataset if its structure reflects that particular set to an excessive degree. For example, the classifier might be generated by rote learning without any generalization whatsoever. An overfitted classifier usually exhibits performance on the training set which is excellent but far from representative of performance on other datasets from the same source.

In practice, one must predict performance bounds based on experiments with whatever data is available. Labeled data is required for both training and testing, and is often hard to obtain. A single data set can be partitioned for training and testing in various different ways. In a popular statistical technique called *cross-validation* the experimenter first decides on a fixed number of "folds," or partitions of the data – say three. The data is split into three approximately equal portions, and each in turn is used for testing while the remainder serves for training. The procedure is repeated three times so that in the end every instance has been used exactly once for testing. This is called *threefold cross-validation*. "Stratification" is the idea of ensuring that all classes are represented in all folds in approximately the right proportions. *Stratified tenfold cross-validation* has become a common standard for estimating the error rate of a classification learning scheme. Alternatives include *leave-one-out* cross-validation, which is effectively n-fold cross-validation where n is the size of the data set; and the *bootstrap*, which takes a carefully-judged number of random samples from the data with replacement and uses these for training, combining the error rate on the training data (an optimistic estimate) with that on the test data (a pessimistic estimate, since the classifier has only been trained on a subset of the full data) to get an overall estimate.

Key Applications

Classification learning is one of the flagship triumphs of research in artificial intelligence. It has been used for problems that range from selecting promising embryos to implant in a human womb during in vitro fertilization to the selection of which cows in a herd to sell off to an abattoir. Fielded applications are legion. They include decisions involving judgment, such as whether a credit company should make a loan to a particular person; screening images, such as the detection of oil slicks from satellite images; load forecasting, such as combining historical load information with current weather conditions and other events to predict hourly demand for electricity; diagnosis, such as fault finding and preventative maintenance of electromechanical devices; marketing and sales, such as detecting customers who are likely to switch to a competitor.

URL to Code

The Weka machine learning workbench is a popular tool for experimental investigation and comparison

of classification learning techniques, as well as other machine learning methods. It is described in [11] and available for download from http://www.cs.waikato.ac.nz/ml/weka.

Cross-references

- ▶ Abstraction
- ▶ Association Rules
- ▶ Bagging
- ▶ Bayesian Classification
- ▶ Boosting
- ▶ Bootstrap
- ▶ Cataloging in Digital Libraries
- ▶ Classification by Association Rule Analysis
- ▶ Clustering Overview and Applications
- ▶ Cross-Validation
- ▶ Data Mining
- ▶ Decision Rule Mining in Rough Set Theory
- ▶ Decision Tree Classification
- ▶ Fuzzy Set Approach
- ▶ Genetic Algorithms
- ▶ Linear Regression
- ▶ Log-Linear Regression
- ▶ Nearest Neighbor Classification
- ▶ Neural Networks
- ▶ Receiver Operating Characteristic
- ▶ Rule-Based classification
- ▶ Support Vector Machine

Recommended Reading

1. Breiman L., Friedman J.H., Olshen R.A., and Stone C.J. Classification and Regression Trees. Wadsworth, Pacific Grove, CA, 1984.
2. Bush R.R. and Mosteller F. Stochastic Models for Learning. Wiley, New York, 1955.
3. Holte R.C. Very simple classification rules perform well on most commonly used datasets. Mach. Learn., 11:63–91, 1993.
4. Kononebko I. ID3, sequential Bayes, naïve Bayes and Bayesian neural networks. In Proc. 4th European Working Session on Learning, 1989, pp. 91–98.
5. Maron M.E. and Kuhns J.L. On relevance, probabilistic indexing and information retrieval. J. ACM, 7(3):216–244, 1960.
6. Minsky M.L. and Papert S. Perceptrons. Cambridge, MIT Press, 1969.
7. Nilsson N.J. Learning Machines. McGraw-Hill, New York, 1965.
8. Quinlan J.R. Induction of decision trees. Mach. Learn., 1(1):81–106, 1986.
9. Quinlan J.R. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco, CA, 1993.
10. Rosenblatt F. Principles of Neurodynamics. Spartan, Washington, DC, 1961.
11. Witten I.H. and Frank E. Data Mining: Practical Machine Learning Tools and Techniques (2nd edn.). Morgan Kaufmann, San Francisco, CA, 2003.

C

Classification by Association Rule Analysis

Bing Liu
University of Illinois at Chicago, Chicago, IL, USA

Synonyms

Associative classification

Definition

Given a training dataset D, build a classifier (or a classification model) from D using an association rule mining algorithm. The model can be used to classify future or test cases.

Historical Background

In the previous section, it was shown that a list of rules can be induced or mined from the data for classification. A decision tree may also be converted to a set of rules. It is thus only natural to expect that association rules [1] be used for classification as well. Yes, indeed! Since the first classification system (called CBA) that used association rules was reported in [10], many techniques and systems have been proposed by researchers [2–4,6–8,13,15,16]. CBA is based on class association rules (CAR), which are a special type of association rules with only a class label on the right-hand-side of each rule. Thus, syntactically or semantically there is no difference between a rule generated by a class association rule miner and a rule generated by a rule induction system (or a decision tree system for that matter). However, class association rule mining inherits the completeness property of association rule mining [1]. That is, all rules that satisfy the user-specified minimum support and minimum conference are generated. Other classification algorithms only generate a small subset of rules existing in data for classification [9,10].

Most existing classification systems based on association rules (also called *associative classifiers*) employ CARs directly for classification, although their ways of using CARs can be quite different [3,7,8,10,15,16].

To deal with unbalanced class distributions, the multiple minimum class supports approach is proposed in [9,11], which gives each class a different minimum support based on its relative frequency in the data. In [2,4,6,13], the authors also proposed to use rules as features or attributes to augment the original data or even to replace the original data. That is, in these techniques, CARs are not directly used for classification, but are used only to expand or to replace the original data. Any classification technique can be used subsequently to build the final classifier based on the expanded data, e.g., naïve Bayesian and SVM. Since the number of class association rules can be huge, closed rule sets have been proposed for classification in [3]. This approach helps solve the problem that in many data sets the complete sets of CARs cannot be generated due to combinatorial explosion. The closed rule set is a smaller, lossless and concise representation of all rules. Thus, long rules (rules with many conditions) may be used in classification, which otherwise may not be generated but can be crucial for accurate classification. Finally, normal association rules may be used for prediction or classification as well.

This section thus introduces the following three approaches to using association rules for classification:

1. Using class association rules for classification
2. Using class association rules as features or attributes
3. Using normal association rules for classification

The first two approaches can be applied to tabular data or transactional data. The last approach is usually employed for transactional data only. Transactional data sets are difficult to handle by traditional classification techniques, but are very natural for association rules. Below, the three approaches are described in turn. Note that various sequential rules can be used for classification in similar ways as well if sequential data sets are involved [6].

Foundations

Classification Using Class Association Rules

As mentioned above, a class association rule (CAR) is an association rule with only a class label on the right-hand side of the rule. Any association rule mining algorithm can be adapted for mining CARs. For example, the Apriori algorithm [1] for association rule mining was adapted to mine CARs in [10].

There is basically no difference between rules generated from a decision tree (or a rule induction system) and CARs if only categorical (or discrete) attributes (more on this later) are considered. The differences are in the mining processes and the final rule sets. CAR mining finds all rules in data that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf) constraints. A decision tree or a rule induction system finds only a subset of the rules (expressed as a tree or a list of rules) for classification. In many cases, rules that are not in the decision tree (or the rule list) may be able to perform the classification more accurately. Empirical comparisons reported by several researchers have shown that classification using CARs can perform more accurately on many data sets than decision trees and rule induction systems [7,8,10,15,16].

The complete set of rules from CAR mining is also beneficial from the rule usage point of view. In many applications, the user wants to act on some interesting rules. For example, in an application for finding causes of product problems in a manufacturing company, more rules are preferred to fewer rules because with more rules, the user is more likely to find rules that indicate causes of problems. Such rules may not be found by a decision tree or a rule induction system. A deployed data mining system based on CARs is reported in [12] for finding actionable knowledge from manufacturing and engineering data sets.

One should, however, also bear in mind of the following differences between CAR mining and decision tree construction (or rule induction):

1. Decision tree learning and rule induction do not use the minsup or minconf constraint. Thus, some rules that they find can have very low supports, which, of course, are likely to be pruned because the chance that they overfit the training data is high. Although a low minsup for CAR mining can be used, it may cause combinatorial explosion. In practice, in addition to minsup and minconf, a limit on the total number of rules to be generated may be used to further control the CAR generation process. When the number of generated rules reaches the limit, the algorithm stops. However, with this limit, long rules (with many conditions) may not be generated. Recall that the Apriori algorithm works in a level-wise fashion, i.e., short

rules are generated before long rules. In some applications, this may not be an issue as short rules are often preferred and are sufficient for classification or for action. Long rules normally have very low supports and tend to overfit the data. However, in some other applications, long rules can be useful.

2. CAR mining does not use continuous (numeric) attributes, while decision trees deal with continuous attributes naturally. Rule induction can use continuous attributes as well. There is still no satisfactory method to deal with such attributes directly in association rule mining. Fortunately, many attribute discretization algorithms exist that can automatically discretize the value range of a continuous attribute into suitable intervals [e.g., [5]], which are then considered as discrete values.

Mining Class Association Rules for Classification There are many techniques that use CARs to build classifiers. Before describing them, it is useful to first discuss some issues related to CAR mining for classification.

Rule pruning: CAR rules are highly redundant, and many of them are not statistically significant (which can cause overfitting). Rule pruning is thus needed. The idea of pruning CARs is basically the same as tree pruning in decision tree building or rule pruning in rule induction. Thus, it will not be discussed further (see [8,10] for some of the pruning methods).

Multiple minimum class supports: A single minsup may be inadequate for mining CARs because many practical classification data sets have uneven class distributions, i.e., some classes cover a large proportion of the data, while others cover only a very small proportion (which are called *rare* or *infrequent classes*).

For example, there is a data set with two classes, *Y* and *N*. 99% of the data belong to the *Y* class, and only 1% of the data belong to the *N* class. If the minsup is set to 1.5%, no rule for class *N* will be found. To solve the problem, the minsup needs to be lowered. Suppose the minsup is set to 0.2%. Then, a huge number of overfitting rules for class *Y* may be found because minsup = 0.2% is too low for class *Y*.

Multiple minimum class supports can be applied to deal with the problem. A different *minimum class support* $minsup_i$ for each class c_i can be assigned, i.e., all the rules of class c_i must satisfy $minsup_i$. Alternatively, one single total minsup can be provided, denoted by *t_minsup*, which is then distributed to each class according to the class distribution:

$$minsup_i = t\_minsup \times sup(c_i)$$

where $sup(c_i)$ is the support of class c_i in the training data. The formula gives frequent classes higher minsups and infrequent classes lower minsups. There is also a general algorithm for mining normal association rules using multiple minimum supports in [9,11].

Parameter selection: The parameters used in CAR mining are the minimum supports and the minimum confidences. Note that a different minimum confidence may also be used for each class. However, minimum confidences do not affect the classification much because classifiers tend to use high confidence rules. One minimum confidence is sufficient as long as it is not set too high. To determine the best $minsup_i$ for each class c_i, a range of values can be tried to build classifiers and then use a validation set to select the final value. Cross-validation may be used as well.

Classifier Building After all CAR rules are found, a classifier is built using the rules. There are many existing approaches, which can be grouped into three categories.

Use the strongest rule: This is perhaps the simplest strategy. It simply uses CARs directly for classification. For each test instance, it finds the strongest rule that covers the instance. A rule *covers* an instance if the instance satisfies the conditions of the rule. The class of the strongest rule is then assigned as the class of the test instance. The strength of a rule can be measured in various ways, e.g., based on confidence, χ^2 test, or a combination of both support and confidence values.

Select a subset of the rules to build a classifier: The representative method of this category is the one used in the CBA system [10]. The method is similar to the sequential covering method, but applied to class association rules with additional enhancements as discussed above.

Let the set of all discovered CARs be *S*. Let the training data set be *D*. The basic idea is to select a subset L ($\subseteq S$) of high confidence rules to cover *D*. The set of selected rules, including a default class, is then used as the classifier. The selection of rules is based on a total order defined on the rules in *S*.

Definition: *Given two rules,* r_i *and* r_j, $r_i \succ r_j$*(called* r_i *precedes* r_j*, or* r_i *has a higher precedence than* r_j*) if*

1. *The confidence of r_i is greater than that of r_j, or*
2. *Their confidences are the same, but the support of r_i is greater than that of r_j, or*
3. *Both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j.*

A CBA classifier L is of the form:

$L = < r_1, r_2,...,r_k, default\text{-}class>$

where $r_i \in S$, $r_a \succ r_b$ if $b > a$. In classifying a test case, the first rule that satisfies the case classifies it. If no rule applies to the case, it takes the default class (*default-class*). A simplified version of the algorithm for building such a classifier is given in Fig. 1. The classifier is the *RuleList*.

This algorithm can be easily implemented by making one pass through the training data for every rule. However, this is extremely inefficient for large data sets. An efficient algorithm that makes at most two passes over the data is given in [10].

Combine multiple rules: Like the first approach, this approach does not have an additional step to build a classifier. At the classification time, for each test instance, the system first finds the subset of rules that covers the instance. If all the rules in the subset have the same class, the class is assigned to the test instance. If the rules have different classes, the system divides the rules into groups according to their classes, i.e., all rules of the same class are in the same group. The system then compares the aggregated effects of the rule groups and finds the strongest group. The class label of the strongest group is assigned to the test instance. To measure the strength of a rule group, there again can be many possible techniques. For example, the CMAR system uses a weighted χ^2 measure [8].

Class Association Rules as Features

In the above two approaches, rules are directly used for classification. In this approach, rules are used as features to augment the original data or simply form a new data set, which is then fed to a traditional classification algorithm, e.g., decision trees or the naïve Bayesian algorithm.

To use CARs as features, only the conditional part of each rule is needed, and it is often treated as a Boolean feature/attribute. If a data instance in the original data contains the conditional part, the value of the feature/attribute is set to 1, and 0 otherwise. Several applications of this method have been reported [2,4,6,13]. The reason that this approach is helpful is that CARs capture multi-attribute or multi-item correlations with class labels. Many classification algorithms do not find such correlations (e.g., the naïve Bayesian method), but they can be quite useful.

Classification Using Normal Association Rules

Not only can class association rules be used for classification, but also normal association rules. For example, association rules are commonly used in e-commerce Web sites for product recommendations, which work as follows: When a customer purchases some products, the system recommends him/her some other related products based on what he/she has already purchased.

Recommendation is essentially a prediction problem. It predicts what a customer is likely to buy. Association rules are naturally applicable to such applications. The classification process is as follows:

1. The system first uses previous purchase transactions (the same as market basket transactions) to mine association rules. In this case, there are no fixed classes. Any item can appear on the left-hand side or the right-hand side of a rule. For recommendation purposes, usually only one item appears on the right-hand side of a rule.
2. At the prediction (e.g., recommendation) time, given a transaction (e.g., a set of items already

```
Algorithm CBA(S, D)
1  S = sort(S);                    // sorting is done according to the precedence>
2  RuleList = ∅;                   // the rule list classifier
3  for each rule r ∈ S in sequence do
4     if D ≠ ∅ AND r classifies at least one example in D correctly then
5        delete from D all training examples covered by r;
6        add r at the end of RuleList
7     endif
8  endfor
9  add the majority class as the default class at the end of RuleList
```

Classification by Association Rule Analysis. Figure 1. A simple classifier building algorithm.

purchased by a customer), all the rules that cover the transaction are selected. The strongest rule is chosen and the item on the right-hand side of the rule (i.e., the consequent) is the predicted item and is recommended to the user. If multiple rules are very strong, multiple items can be recommended.

This method is basically the same as the "use the strongest rule" method described earlier. Again, the rule strength can be measured in various ways, e.g., confidence, χ^2 test, or a combination of both support and confidence. Clearly, the other two classification methods discussed earlier can be applied here as well.

The key advantage of using association rules for recommendation is that they can predict any item since any item can be the class item on the right-hand side. Traditional classification algorithms only work with a single fixed class attribute, and are not easily applicable to recommendations.

Finally, it should be noted that multiple minimum supports in rule mining [11] can be of significant help. Otherwise, *rare items* will never be recommended, which is called the *coverage* problem [14]. It is shown in [14] that using multiple minimum supports can dramatically increase the coverage.

Key Applications

The applications of associative classifiers are very wide. Three main scenarios are briefly described below.

1. Since classification using class association rules is a supervised learning technique, it can be (and has been) used as a classification algorithm just like any other classification algorithm from machine learning, e.g., decision trees, naïve Bayesian classifiers, SVM, and rule induction. In many cases, an associative classifier performs better than these classic machine learning techniques.
2. Apart from classification, individual class association rules themselves are very useful in practice due to the completeness property. In many practical applications (especially diagnostic data mining applications), the user wants to find interesting rules that are actionable. As discussed earlier, traditional classification algorithms (e.g., rule induction or any other technique) are not suitable for such applications because they only find a small subset of rules that exist in data. Many interesting or actionable rules are not discovered. A deployed data mining system, called Opportunity Map, for Motorola Corporation was based on class association rules [12]. When this entry was written, the system had been in use in Motorola for more than 2 years and further improvements were still being made. Although the system was originally designed for finding rules that indicate causes of phone call failures, it had been used in a variety of other applications in Motorola.
3. Using normal association rules for classification or prediction is also very common, especially for the transaction type of data. For such kind of data, as described above, traditional classification techniques are not easily applicable because they can only predict some fixed class items (or labels).

Cross-references

- ▶ Association Rule Mining on Streams
- ▶ Decision Trees
- ▶ Rule Induction

Recommended Reading

1. Agrawal R. and Srikant R. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 487–499.
2. Antonie M.L. and Zaiane O. Text document categorization by term association. In Proc. 2002 IEEE Int. Conf. on Data Mining, 2002, pp. 19–26.
3. Baralis E. and Chiusano S. Essential classification rule sets. ACM Trans. Database Syst, 29(4):635–674, 2004.
4. Cheng H., Yan X., Han J., and Hsu C.-W. Discriminative frequent pattern analysis for effective classification. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 706–715.
5. Dougherty J., Kohavi R., and Sahami M. Supervised and unsupervised discretization of continuous features. In Proc. 12th Int. Conf. on Machine Learning, 1995, pp. 194–202.
6. Jindal N. and Liu B. Identifying comparative sentences in text documents. In Proc. 32nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2006, pp. 244–251.
7. Li J., Dong G., and Ramamohanarao K. Making use of the most expressive jumping emerging patterns for classification. In Advances in Knowledge Discovery and Data Mining, 4th Pacific-Asia Conf., 2000, pp. 220–232.
8. Li W., Han J., and Pei J. CMAR: Accurate and efficient classification based on multiple class-association rules. In Proc. 2001 IEEE Int. Conf. on Data Mining, 2001, pp. 369–376.
9. Liu B. Web data mining: exploring hyperlinks, contents and usage data. Springer, Berlin, 2007.
10. Liu B., Hsu W., and Ma Y. Integrating classification and association rule mining. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, 1998, pp. 80–86.
11. Liu B., Hsu W., and Ma Y. Mining association rules with multiple minimum supports. In Proc. 5th ACM SIGKDD

Int. Conf. on Knowledge Discovery and Data Mining, 1999, pp. 337–341.
12. Liu B., Zhao K., Benkler J., and Xiao W. Rule interestingness analysis using OLAP operations. In Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2006, pp. 297–306.
13. Meretakis D. and Wüthrich B. Extending naïve bayes classifiers using long itemsets. In Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 1999, pp. 165–174.
14. Mobasher B., Dai H., Luo T., and Nakagawa N. Effective personalization based on association rule discovery from web usage data. In Proc. 3rd ACM Workshop on Web Information and Data Management, 2001, pp. 9–15.
15. Wang K., Zhou S., and He Y. Growing decision trees on support-less association rules. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 265–269.
16. Yin X. and Han J. CPAR: classification based on predictive association rules. In Proc. SIAM International Conference on Data Mining, 2003.

Classification Learning

► Classification

Classification in Streams

Charu C. Aggarwal
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

Synonyms

Learning in streams; Knowledge discovery in streams

Definition

The classification problem is a well defined problem in the data mining domain, in which a training data set is supplied, which contains several feature attributes, and a special attribute known as the class attribute. The class attribute is specified in the training data, which is used to model the relationship between the feature attributes and the class attribute. This model is used in order to predict the unknown class label value for the test instance.

A data stream is defined as a large volume of continuously incoming data. The classification problem has traditionally been defined on a static training or test data set, but in the stream scenario, either the training or test data may be in the form of a stream.

Historical Background

The problem of classification has been studied so widely in the classification literature, that a single source for the problem cannot be identified. Most likely, the problem was frequently encountered in practical commercial scenarios as a statistical problem, long before the field of machine learning was defined. With advances in hardware technology, data streams became more common, and most data mining problems such as clustering and association rule mining were applied to the data stream domain. Domingos and Hulten [2] were the first to model the problem in the context of data streams.

Foundations

There are numerous techniques available for classification in the classical literature [3]. However, most of these techniques cannot be used directly for the stream scenario. This is because the stream scenario creates a number of special constraints which are as follows:

- The data stream typically contains a large volume of continuously incoming data. Therefore the techniques for training or testing need to be very efficient. Furthermore, a data point may be examined only once over the course of the entire computation. This imposes hard constraints on the nature of the algorithms which may be used for stream classification. This constraint is generally true of almost all data mining algorithms.
- Often the patterns in the underlying data may evolve continuously over time. As a result, the model may soon become stale for data mining purposes. It is therefore important to keep the models current even when the patterns in the underlying data may change. This issue is known as concept drift.
- Many stream classification methods have considerable memory requirements in order to improve computational efficiency. The stream case is particularly resource constrained, since the memory may sometimes be limited, while the computational efficiency requirements continue to be very high.
- In many cases, the rate of incoming data cannot be controlled easily. Therefore, the classification process needs to be nimble enough in order to provide effective tradeoffs between accuracy and efficiency.

Most of the known classification methods can be made to work in the data stream scenario with a few modifications. These modifications are generally designed to deal with either the one-pass constraint, or the

stream evolution scenario. The different types of classifiers which can be modified for the data stream scenario are as follows:

- Nearest Neighbor Classifiers: In these techniques, the class label of the nearest neighbor to the target record is used in order to perform the classification. Since the nearest neighbor cannot be defined easily over the entire stream, a stream sample is used in order to perform the classification. This stream sample can be dynamically maintained with the one-pass constraint with the use of a technique called *reservoir sampling*. In order to deal with issues of stream evolution, one can used *biased* reservoir sampling. In biased sampling, a time decay function is used in order to maintain a sample which is biased towards more recent data points.
- Decision Tree Classifiers: In this techniques, decision trees need to be built in one pass of the stream. A method known as Very Fast Decision Trees (VFDT) was proposed in [2] which uses probabilistic split methods in order to create decision trees with predictable accuracy. In particular, the Hoeffding inequality is used in order to ensure that the generated tree produces the same tree as a conventional learner. Several other techniques were proposed by the same authors subsequently, which deal with the time-changing aspect of the data streams.
- Cluster-based Classifiers: An on-demand stream classification model was proposed which uses clustering techniques in order to build the optimal model for a classifier on demand. In this technique, a micro-clustering technique is used in order to compress the underlying data into clusters. The data belonging to different classes are compressed into different clusters. For a given test example, the class of the closest cluster is used in order to predict the class label. One key aspect of this classifier is that it assumes that both the training and the test data are in the form of a stream. The technique calculates the optimal horizon for using the cluster statistics.
- Ensemble Classifiers: In this case, a combination of different models is used in order to deal with the issue of concept drift. This is because different kinds of models work better with different kinds of data patterns. Therefore, an optimal model is picked depending upon the current data pattern. The idea is that different classifiers are more effective for different kinds of data patterns. Therefore, by making an optimal choice of the classifier from the ensemble, it is possible to improve the classification accuracy significantly.
- Bayes Classifiers: The naive Bayes classifier computes the Bayes a-posteriori probabilities of a test instance belonging to a particular class using the inter-attribute independence assumption. The key in adapting such classifiers is to be able to effectively maintain the statistics used to compute conditional probabilities in one pass. In the case of an evolving data stream, the statistics need to be maintained over particular user-specific horizons.

A number of other methods for stream classification also exist which cannot be discussed within the scope of this entry. A detailed survey on classification methods may be found in [3].

Key Applications

Stream classification finds application to numerous data domains such as network intrusion detection, target marketing and credit card fraud detection. In many of these cases, the incoming data clearly has very large volume. For example, typical intrusion scenarios have a large volume of incoming data. Similarly, in the case of target-marketing, super-store transactions may have very large volumes of incoming data.

Many of the traditional classification applications are still used in the batch mode, since the stream technology is still in its infancy, and it is sometimes simpler to collect a sample of the data set and run a batch process on it. Most of the traditional problems for the classification domain will eventually be transformed to the data stream scenario. This is because more and more data domains are being converted to the stream scenario with advances in hardware technology.

Cross-references

► Association Rule Mining on Streams
► Clustering on Streams
► Data Stream

Recommended Reading

1. Aggarwal C.C. (ed.). Data Streams: Models and Algorithms. Springer, Berlin Heidelberg, New York, 2007.
2. Domingos P. and Hulten G. Mining high speed data streams. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
3. James M. Classification Algorithms. Wiley, New York, 1985.

Classification Tree

▶ Decision Tree Classification

Classification Trees

▶ Decision Trees

Classifier Combination

▶ Ensemble

Client-Server DBMS

M. Tamer Özsu
University of Waterloo, Waterloo, ON, Canada

Definition

Client-server DBMS (database management system) refers to an architectural paradigm that separates database functionality between client machines and servers.

Historical Background

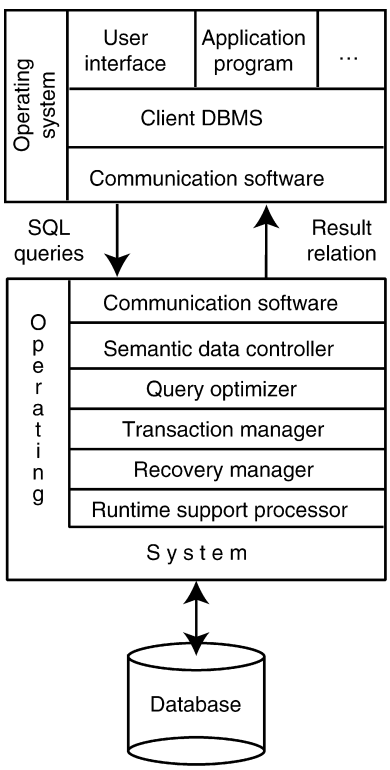
The original idea, which is to offload the database management functions to a special server, dates back to the early 1970s [1]. At the time, the computer on which the database system was run was called the *database machine*, *database computer*, or *backend computer*, while the computer that ran the applications was called the *host computer*. More recent terms for these are the *database server* and *application server*, respectively.

The client-server architecture, as it appears today, has become a popular architecture around the beginning of 1990s [2]. Prior to that, the distribution of database functionality assumed that there was no functional difference between the client machines and servers (i.e., an earlier form of today's peer-to-peer architecture). Client-server architectures are believed to be easier to manage than peer-to-peer systems, which has increased their popularity.

Foundations

Client-server DBMS architecture involves a number of database client machines accessing one or more database server machines. The general idea is very simple and elegant: distinguish the functionality that needs to be provided and divide these functions into two classes: server functions and client functions. This provides a *two-level architecture* that makes it easier to manage the complexity of modern DBMSs and the complexity of distribution.

In client-server DBMSs, the database management functionality is shared between the clients and the server(s) (Fig. 1). The server is responsible for the bulk of the data management tasks as it handles the storage, query optimization, and transaction management (locking and recovery). The client, in addition

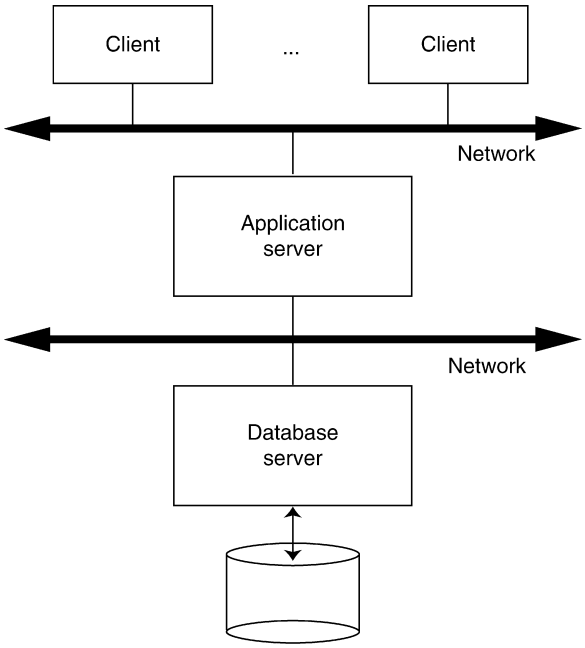


Client-Server DBMS. Figure 1. Client-server reference architecture.

to the application and the user interface, has a *DBMS client* module that is responsible for managing the data that are cached to the client, and (sometimes) managing the transaction locks that may have been cached as well. It is also possible to place consistency checking of user queries at the client side, but this is not common since it requires the replication of the system catalog at the client machines. The communication between the clients and the server(s) is at the level of SQL statements: the clients pass SQL queries to the server without trying to understand or optimize them; the server executes these queries and returns the result relation to the client. The communication between clients and servers are typically over a computer network.

In the model discussed above, there is only one server which is accessed by multiple clients. This is referred to as *multiple client-single server* architecture [3]. There are a number of advantages of this model. As indicated above, they are simple; the simplicity is primarily due to the fact that data management responsibility is delegated to one server. Therefore, from a data management perspective, this architecture is similar to centralized databases although there are some (important) differences from centralized systems in the way transactions are executed and caches are managed. A second advantage is that they provide predictable performance. This is due to the movement of non-database functions to the clients, allowing the server to focus entirely on data management. This, however, is also the cause of the major disadvantage of client-server systems. Since the data management functionality is centralized at one server, the server becomes a bottleneck and these systems cannot scale very well.

The disadvantage of the simple client-server systems are partially alleviated by a more sophisticated architecture where there are multiple servers in the system (the so-called *multiple client-multiple server* approach). In this case, two alternative management strategies are possible: either each client manages its own connection to the appropriate server or each client knows of only its "home server", which then communicates with other servers as required. The former approach simplifies server code, but loads the client machines with additional responsibilities, leading to what has been called "heavy client" systems. The latter approach, on the other hand, concentrates the data management functionality at the servers. Thus, the transparency of data access is provided at the server interface, leading to "light clients."



Client-Server DBMS. Figure 2. Database server approach.

The integration of workstations in a distributed environment enables an extension of the client-server architecture and provides for a more efficient function distribution. Application programs run on workstations, called *application servers*, while database functions are handled by dedicated computers, called *database servers*. The *clients* run the user interface. This leads to the present trend in three-tier distributed system architecture, where sites are organized as specialized servers rather than as general-purpose computers (Fig. 2).

The application server approach (indeed, a n-tier distributed approach) can be extended by the introduction of multiple database servers and multiple application servers, as can be done in client-server architectures. In this case, it is common for each application server to be dedicated to one or a few applications, while database servers operate in the multiple server fashion discussed above.

Key Applications

Many of the current database applications employ either a two-layer client-server architecture of the three-layer application-server approach.

Cross-references

► DBMS Architecture

Recommended Reading

1. Canaday R.H., Harrisson R.D., Ivie E.L., Rydery J.L., and Wehr L.A. A back-end computer for data base management. Commun. ACM, 17(10):575–582, 1974.
2. Orfali R., Harkey D., and Edwards J. Essential Client/Server Survival Guide, Wiley, New York, 1994.
3. Özsu M.T. and Valduriez P. Principles of Distributed Database Systems, 2nd edn., Prentice-Hall, Englewood Cliffs, NJ, 1999.

Clinical Classifications

► Clinical Ontologies

Clinical Content Database

► Clinical Knowledge Repository

Clinical Content Registry

► Clinical Knowledge Repository

Clinical Content Repository

► Clinical Knowledge Repository

Clinical Data Acquisition, Storage and Management

Chimezie Ogbuji
Cleveland Clinic Foundation, Cleveland, OH, USA

Synonyms

Electronic data capture; Case report forms; Clinical data management systems

Definition

The management of clinical data for supporting patient care and for supporting retrospective clinical research requires a means to acquire the clinical data and a repository that stores the data and provides the functions necessary for managing them over their lifetime. Typically, patient data are collected "at the point of care" (i.e., onsite where health care is being provided) and entered into a patient record system [2]. Data entry is typically the first line of precaution for maintaining a certain amount of quality on the data collected. Subsequently, a representative from an externally sponsoring organization or authorized personnel from within the health care institution then extracts a select set of medical record data into a Clinical Data Management System (*CDMS*). The entries in such systems are often referred to as secondary patient records since they are derived from a primary patient record and are used by personnel who are not involved in direct patient care [2]. These systems are also often referred to as *patient registries*.

The study committee of the Institute of Medicine (IOM) defined [2] a computer-based patient record (*CPR*) as

> ► an electronic patient record that resides in a system specifically designed to support users by providing accessibility to complete and accurate data, alerts, reminders, clinical decision support systems, links to medical knowledge, and other aids.

Such systems are also often referred to as Electronic Health Records (*EHRs*). The committee also defined a *primary* patient record as one used by health care professionals while providing patient care services to review patient data or document their own observations, actions, or instructions [2]. Finally, the committee emphasized the distinction between clinical data and the systems that capture and process this data by defining a patient record *system* as

> ► the set of components that form the mechanism by which patient records are created, used, stored, and retrieved.

A CDMS is the repository for the management of the data used for clinical studies or trials. One of the core services provided by a CDMS is to facilitate the identification (and correction) of errors due to human entry as well as errors that existed in the source from which

the data was gathered. Data completeness implies that CDMS will accommodate an expected range and complexity for the data in the system [2]. In addition, the CDMS can employ the use and enforcement of one or more vocabulary standards.

Finally, a CDMS will also provide services for querying the data as well as generate reports from the data. The generated reports and results of such queries are typically transmitted to a centralized authority or normalized for use by statistical analysis tools.

Historical Background

Virtually every person in the United States who has received health care in the United States since 1918 has a patient record [2]. Most of these records consist of structured paper forms with sections that consist solely of narrative text. However, conventional patient records can also appear in other forms such as scanned media, microfilm, optical disk, etc.

They are created and used most frequently in health care provider settings. However, their use also extends to other facilities such as correctional institutions, the armed forces, occupational health programs, and universities [2].

The process of recording patient care information has primarily consisted of entry into a paper patient record. For the purpose of a clinical trial or study, data are manually transcribed into a paper Case Report Form (*CRF*). CRFs are typically in the form of a questionnaire formulated to collect information specific to a particular clinical trial. The ICH Guidelines for Good Clinical Practice define [4] the CRF as:

> ► A printed, optical, or electronic document designed to record all of the protocol required information to be reported to the sponsor on each trial subject.

CRFs are then collected by a representative from the sponsoring organization and manually entered into a CDMS. This secondary transcription is often called *double data entry*. In some cases, Optical Character Recognition (*OCR*) is used to semi-automate the transcription from a CRF into a CDMS [1].

Traditionally, clinical data management systems consist of infrastructure built on top of relational database management systems. Depending on the nature of the requirements for the creation of analysis data sets for biostatisticians, accuracy of the data, and speed of data entry, a wide spectrum of database management or spreadsheet systems are used as the underlying medium of storage for the CDMS.

Good database design and proper application of relational model theory for normalizing the data is typically used to ensure data accuracy. Traditional relational query languages such as SQL are used to identify and extract relevant variables for subsequent analysis or reporting purposes.

Foundations

Electronic Data Capture

There is a slow, but steady move by pharmaceutical companies towards the adoption of an electronic means of capturing patient record information directly from the source and at the point of care into an electronic system that submits the data relevant to the trial to the sponsor or to other consumers of electronic patient record data. This new shift of emphasis from paper to a direct electronic system is often referred to (in the health care industry) as Electronic Data Capture (*EDC*) [3].

Infrastructure and Standards for Data Exchange

Once patient record data are collected and stored in an electronic information system, the increasing need to transfer the machine-readable data to external systems emphasizes the importance of standardized formats for communication between these disparate systems [2]. Efforts to standardize a common format for communication between health care systems and other external consumers of health care data have settled on the adoption of Extensible Markup Language (*XML*) as the primary data format for the transmission of Health Level 7 (*HL7*) messages.

HL7 is an organization with a mission to develop standards that improve the delivery of care, optimize the management of workflow, reduce ambiguity in clinical terminology and facilitate efficient transfer of knowledge between the major stakeholders.

Document Models and Management Systems

As its name implies, XML is a markup language for describing structured data (or *documents*) in a manner

that is both human- and machine-readable. It can be extended to support users who wish to define their own vocabularies. It is meant to be highly reusable across different information systems for a variety of purposes. It is recommended by the World Wide Web Consortium (*W3C*) and is a free and open standard.

XML is at the core of an entire suite of technologies produced by the W3C that includes languages for querying XML documents as well as describing their structure for the purpose of validating their content. This suite of technologies is meant to serve as infrastructure for a contemporary set of information systems each known more broadly as a Document Management System (*DMS*).

Common Components of Information Systems

Like most information systems, document management systems are comprised of a particular data model (XML in this case), one or more query languages, and a formal processing model for systems that wish to support queries written in language against the underlying data. Document management systems (and information systems in general) typically also offer security services that ensure limited access to the data. This is particularly important for clinical data management systems.

With respect to the kind of services they provide for the systems that are built on top of them (such as clinical data management systems), document management systems are very much like relational database management systems. However, whereas relational database management systems have an underlying relational model that is tabular and rigid, document management systems have a data model that is hierarchical with data elements that can be extended to support new terminology over the life of the data.

Text-Oriented Information Systems

Most modern computer-based patient record systems mainly adopt information systems with hierarchical, relational, or text-oriented data models. Text-based information systems typically store their content primarily as narrative text and often employ natural language processing for extracting structured data for transcription into a clinical data management system. Querying such systems usually involves keyword-based searches that use text indexes that are used to associate words with the sections of narrative in which they can be found.

Key Applications

Electronic data capture and clinical data management systems constitute the majority of the infrastructure necessary in the overall process of clinical research from the point of interaction with primary patient records all the way to the analysis of the curated clinical research data. The sections below describe the major areas where their application makes a significant difference.

Electronic Data Collection Options

There are a variety of ways in which data can be acquired electronically for transcription into a clinical data management system. The most desired means is one where the data are directly retrieved electronically from an existing source such as the primary patient record. This method is often referred to as *single entry* [1]. It requires that the primary patient record adopt or align with a set of consistent format standards such that they can facilitate the support of primary care as well as reuse for the purpose of (unanticipated) clinical research. Unfortunately the lack of adoption of computer-based patient records remains a primary impediment to this more direct means of acquiring clinical data [2,3].

Alternatively, clinical data can be transcribed from a primary patient record into a secondary patient record using some form of an electronic user interface on a particular device. Typically, such user interfaces are either web browser-based (i.e., they are built on top of an existing web browser such as Internet Explorer or Firefox) or they are written as independent applications that are launched separately. The latter approach is often referred to as a *thick-client system* [6].

Patient Registries

The set of functions associated with a secondary computer-based patient record system is often adopted from the underlying information system. Modern document and relational database management systems are equipped with a wide spectrum of capabilities each of which is directly relevant to the needs of users of these systems. This includes: content organization, archival, creation of documents, security, query services, disaster recovery, and support for web-based user interfaces.

Clinical Workflow Management

Equally important to the clinical data is the management of the pattern of activity, responsibilities, and resources associated with a particular clinical study or trial. These patterns are often referred to as *workflow*. Orchestrating the overall process can also have a significant impact on the success of a clinical study. Clinical data management systems sometimes have off-the-shelf capabilities for managing workflow. These usually support some amount of automation of the workflow process. Document management systems that include capabilities for building customized application are well-suited for supporting workflows that are either specific to a particular study protocol or capable of supporting multiple (or arbitrary) protocols.

Quality Management, Report Generation, and Analysis

Finally, document management and relational database management systems include capabilities for monitoring error in the data collected. This is often supported through the application of a set of common constraints that are relevant to the research protocol. Typically, these systems have an automated mechanism for indicating when the underlying data does not adhere to the constraints specified.

In addition, document management and relational database management systems include services for generating reports and extracting variables for statistical analysis.

Future Directions

Modern information management systems are adopting standards for representation formats that push the envelope of machine-readability. In particular, the W3C has recently been developing a suite of technologies that build on the standards associated with the World Wide Web and introduce a formal model for capturing knowledge in a manner that emphasizes the meaning of terms rather than their structure. Such approaches to modeling information are often referred to as knowledge representation or conceptual models. This particular collection of standards is commonly referred to as *semantic web technologies* [5].

Semantic web technologies are built on a graph-based data model known as the Resource Description Framework (*RDF*) as well as a language for describing conceptual models for RDF data known as Ontology Web Language (*OWL*). RDF leverages a highly-distributable addressing and naming mechanism known as Uniform Resource Identifiers (*URI*s) that is the foundation of the current web infrastructure.

Semantic web technologies also include a formal mechanism for rendering or transforming XML document dialects into RDF known as Gleaning Resources Descriptions from Dialects of Languages (*GRDDL*). Finally a common query language has been defined for accessing data expressed in RDF known as *SPARQL*.

The Institute of Medicine has indicated [2] that the flexibility of computer-based patient records is primarily due to their adoption of a data dictionary that can be expanded to accommodate new elements. In addition, the IOM has identified [2] the following as crucial to the evolution of content and standard formats in computer-based patient record systems:

- The content of CPRs must be defined and contain a uniform core set of data elements.
- Data elements must be named consistently via the enforcement of some form of vocabulary control.
- Format standards for data exchange must be developed and used.

In addition, the IOM's study committee identified the ability for CPRs to be linked with other clinical records as a critical attribute of a comprehensive computer-based patient record. The combination of these observations is a strong indication that in the future, clinical data management systems will be built on information management systems that adopt semantic web technologies in order to better meet the growing needs of the management of clinical research data.

Finally, a new generation of technologies for building declarative web applications will lower the technological barrier associated with the kind of user interfaces necessary for the adoption of electronic data capture methods at health care institutions. In particular, an XML-based technology known as *XForms* is well positioned to have a significant impact on the front end of the clinical data pipeline (data collection).

XForm applications are web form-based, independent of the device on which they are deployed and use XML as the data model of the underlying content. This approach has strong correspondence with the current direction of clinical data exchange standards with the adoption of XML as the format for communication between health care systems.

In the near future, lightweight devices (such as Tablet PCs) will connect to remote, distributed computer-based patient record systems over a secure web-based

network protocol. Electronic data capture will be implemented by XForm applications that run in a browser and compose XML documents that represent sections of a computer-based patient record. These documents will adhere to a standard document format for the exchange of medical records such as the HL7 Clinical Document Architecture (*CDA*). The HL7 CDA is an XML-based document markup standard that specifies the structure and semantics of clinical documents for the purpose of exchange.

These documents will be securely transmitted directly into a primary computer-based patient record which employs XML as its core data model and uses GRDDL to also store an RDF representation of the document that conforms to a formal, standard ontology (expressed in OWL) that describes the meaning of the terms. This ontology provides a certain degree of logical consistency that facilitates ad hoc analysis through the use of logical inference.

Patients that meet the criteria for a particular research protocol will be identified by a SPARQL query that is dispatched against the patient record system, which uses terminology easily understood by the investigators themselves (rather than an intermediary database administrator). These patient records will then be transmitted directly into a clinical data management system (or patient registry) that will include the facilities for managing the workflow associated with the relevant research protocol. These facilities will be implemented as web applications built on the same underlying information management systems as those used by the primary computer-based patient records.

Cross-references

- ▶ Clinical Content Management
- ▶ Clinical Data Quality and Validation
- ▶ Data warehousing and Quality Management for Clinical Practice
- ▶ Electronic Health Record
- ▶ Life Cycles and Provenance
- ▶ Versioning

Recommended Reading

1. Anisfeld M.H. and Prokscha S. Practical Guide to Clinical Data Management. CRC Press, Boca Raton, FL, 1999.
2. Committee on Improving the Patient Record, Institute of Medicine. The computer-based patient record: an essential technology for health care (revised edition). National Academies Press, 1997.
3. Lori A. and Nesbitt. Clinical Research: What It Is and How It Works. Jones and Bartlett Publishers, Sudbury, MA, 2003.
4. Rondel R.K., Varley S.A., and Webb C.F. Clinical Data Management. Wiley, Chichester, 2000.
5. Ruttenberg A., Clark T., Bug W., Samwald M., Bodenreider O., Chen H., Doherty D., Forsberg K., Gao Y., Kashyap V., Kinoshita J., Luciano J., Marshall M.S., Ogbuji C., Rees J., Stephens S., Wong G.T., Elizabeth Wu, Davide Zaccagnini, Tonya Hongsermeier, Neumann E., Herman I., and Cheung K.-H. Advancing translational research with the Semantic Web. BMC Bioinformatics, 8(Suppl. 3), 2007.
6. Wilson D., Pace M.D., Elizabeth W., and Staton, M. S. T. C. Electronic Data Collection Options for Practice-Based Research Networks. Ann. Fam. Med., 3:S2–S4, 2005.

Clinical Data and Information Models

Chintan Patel, Chunhua Weng
Columbia University, New York, NY, USA

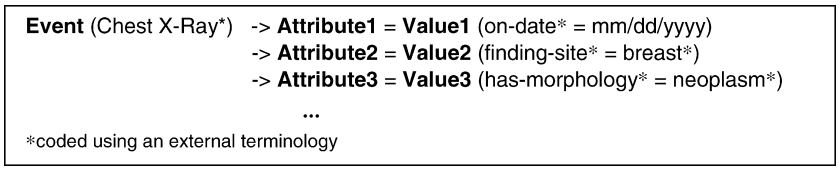
Definition

A formal representation of the clinical data using entities, types, relationships and attributes. The abstraction of clinical data into an information model enables reusability and extensibility of the database to satisfy different application needs and accommodate changes in the underlying data.

Key Points

The clinical domain is a data rich environment with multitude of different data entities ranging from several thousands of laboratory tests, procedures or medications that change often with new ones getting added almost every day. Furthermore these data are generated from different information systems or devices (often from different vendors) in the hospital. Integrating such wide variety of data streams into a common information model is a challenging task.

Most healthcare databases use generic information models [3,4] such as event-component models with an Entity-Attribute-Value [5] (EAV) schema to represent the data (see Fig. 1). The advantage of using a generic information model is to accommodate the data heterogeneity and extensibility. Generally, an external terminology or vocabulary is used in conjunction with a generic information model to represent the clinical domain (laboratory tests, medications and so on) and the healthcare activities, for example, LOINC is a



Clinical Data and Information Models. Figure 1. The event component information model.

standard vocabulary for representing laboratory data or SNOMED CT for healthcare activities.

Various information models have been proposed towards standardizing the representation of clinical data. The goal of standardizing the information model is to facilitate exchange, sharing and reuse of clinical data by different systems locally as well as nationally. Following are some current standardized models:

HL7 Reference Information Model: The HL7 standards organization [2] has developed a Reference Information Model (RIM) to share consistent meaning of healthcare data beyond local context. The RIM specifies a set of abstract bases classes Entity, Role, Participation and Act, which contain specific classes/attributes such as Person, Organization, Patient, Provider, Intent, Observation and so on. This model is used to create concrete concepts by combining the RIM types, for example, *elevated blood pressure* would be represented in RIM as class = Observation with code = Finding of increased blood pressure (SNOMED#241842005), mood = Event, interpretation code = abnormal (HL7#A), target site = heart (LOINC#LP7289). Note that standardized terminology codes (SNOMED CT and LOINC) are used to represent specific findings and modifiers. An implementation of HL7 RIM based model over a relational database schema is described here [1].

openEHR Reference Model: The openEHR specification [6] (developed largely by the institutions in EU and Australia) provides information models for the electronic health record (EHR), demographics, data structures, integration and so on. The openEHR EHR model represents various facets of EHR such as clinician/patient interaction, audit-trailing, technology/data format independence and supporting secondary uses. The openEHR project uses the notion of archetypes that enable domain experts to formally model a domain concept (or an aggregation of concepts), corresponding constraints and other compositions, for example, an archetype on *blood pressure measurement* consists of systolic, diastolic measurements and units with other clinically relevant information such as history.

Recommended Reading

1. Eggebraaten T.J., Tenner J.W., and Dubbels J.C. A health-care data model based on the HL7 reference information model. IBM Syst. J., 46(1):5–18, 2007.
2. HL7 Reference Information Model. Available at: http://www.hl7.org/ (Accessed April 18, 2008).
3. Huff S., Rocha R., Bray B., Warner H., and Haug P. An event model of medical information representation. J. Am. Med. Inform. Assoc., 2(2):116–134, 1995.
4. Johnson S. Generic data modeling for clinical repositories. J. Am. Med. Inform Assoc., 3(5):328–367, 1996.
5. Nadkarni P., Marenco L., Chen R., Skoufos E., Shepherd G., and Miller P. Organization of heterogeneous scientific data using the EAV/CR representation. J. Am. Med. Inform. Assoc., 6(6):478–571, 1999.
6. openEHR Reference Information Model. Available at: http://www.openehr.org/ (Accessed April 18, 2008).

Clinical Data Management Systems

► Clinical Data Acquisition, Storage and Management

Clinical Data Quality and Validation

Chintan Patel, Chunhua Weng
Columbia University, New York, NY, USA

Definition

Clinical data quality is defined as the accuracy and completeness of the clinical data for the purposes of clinical care, health services and other secondary uses such as decision support and clinical research. The quality of clinical data can be achieved by the standardization, inspection and evaluation of the data generating processes and tools [2].

Key Points

The term data quality can potentially have different meanings or interpretation based on the domain or the application using the data [1]. Even within the context

of clinical databases, there exists a multitude of different data types (administrative data, procedure data, laboratory data and so on) that may be used for several different applications such as clinical report generation, billing or research. The major components of clinical data quality can be broadly characterized as follows:

Accuracy

Clinical data are often generated by automated systems (such as lab equipment) or manually entered by clinicians (notes). These data generating processes are prone to errors that result in incorrect data being stored in the database. The severity of errors can vary significantly, for example, a minor misspelling in patient history note versus a prescription error in drug dosage order can lead to drastically different outcomes in terms of patient care. The accuracy of clinical data is defined as the proportion of correct data (truly representing the actual patient condition or measurement) in the clinical database. The accuracy of clinical data depends on the enforcement of well-defined data entry standards and protocols.

Completeness

It is defined as the availability of data elements in a clinical database that are necessary to accomplish a given task, for example, a clinical trial recruitment application with detailed eligibility criteria would require information from the clinical notes in addition to coded problem list data. The completeness of a patient record is critical for a clinician to choose a most appropriate treatment plan for the patient. The availability of complete patient information is critical during an emergency condition. In the case of unavailability of data elements, some applications tend to substitute data sources, which can lead to sub-optimal results. Consider for example, a clinical decision support application reusing coarse ICD (International Classification of Disease) classification to generate decisions.

Reliability

The notion of "repeatability" – to determine whether the clinical data generation processes produce consistent data at different times or settings. Hospitals are a chaotic environment with multiple care providers taking care of a single patient. It becomes critical to develop data entry protocols to ensure consistent representation of patient information in the clinical database. Often to eliminate the variations across different users the data entry software systems such as the EMR (electronic medical record) contain various checks to ensure the correctness and completeness of the data elements [3]. The coding of clinical data using terminologies such as ICD has to be done in a consistent fashion to facilitate applications that require data integration or comparative analysis.

Maintaining quality in clinical databases is a continuous process requiring strong commitment from different stakeholders involved. The amount of electronic biomedical data generated is growing at an exponential rate. Developing high quality clinical databases can have significant implications for the applications reusing the data.

Cross-references

▶ Quality and Trust of Information Content and Credentialing

Recommended Reading

1. Arts D., De Keizer N., and Scheffer G. Defining and improving data quality in medical registries: a literature review, case study, and generic framework. J. Am. Med. Inform. Assoc., 9(6): 600–611, 2002.
2. Black N. High-quality clinical databases: breaking down barriers. Lancet, 353(9160):1205–1211, 2006.
3. Hogan W. and Wagner M. Accuracy of data in computer-based patient records. J. Am. Med. Inform. Assoc., 4(5): 342–397, 1997.

Clinical Decision Support

Adam Wright
Partners HealthCare, Boston, MA, USA

Synonyms

CDS; Decision support

Definition

Clinical Decision Support systems are computer systems which assist humans in making optimal clinical decisions. While clinical decision support systems are most often designed for clinicians, they can also be developed to assist patients or caregivers. Common examples of clinical decision support systems include drug-drug interaction checks, dose range checking for medication and preventive care reminders.

Historical Background

The first clinical decision support system was described in 1959 by Robert Ledley and Lee Lusted [6] in their paper "Reasoning foundations of medical diagnosis; symbolic logic, probability, and value theory aid our understanding of how physicians reason." Ledley and Lusted described an analog computer used to sort cards containing a diagnosis and a series of punches which represented symptoms. By selecting the cards which matched the symptoms present in a given case a clinician could develop a possible set of diagnosis.

In 1961, Homer Warner [15] described a clinical decision support system for diagnosing congenital heart defects. The system was developed around a contingency table that mapped clinical symptoms to forms of congenital heart disease. A physician would input the patient's symptoms and findings from the clinical exam and other studies into the system, which would then proceed to suggest the most probable diagnoses based on the contingency table.

In the 1970s, Edward Shortliffe developed the well-known MYCIN system for antibiotic therapy. MYCIN was an expert system with a large knowledge base of clinical rules [12]. Users of MYCIN would input known facts about their patient, and MYCIN would apply them to the rule base using backward chaining to yield a probable causative agent for infections as well as suggestions for antibiotic therapy.

While the systems described so far all focused on a specific area of medicine, the INTERNIST-I system, developed by Randy Miller, Harry Pople and Jack Myers [8] targeted the broad domain of diagnosis in internal medicine. The INTERNIST-I knowledge base consisted of a large set of mappings between symptoms and diagnoses. These links were scored along three axes: evoking strength (the likelihood that a patient has a diagnosis given a particular symptom), frequency (how often a symptom is present given a particular diagnosis) and import (how critical it is that a particular diagnosis be considered given that it is possible or probable based on a set of symptoms). Octo Barnett's DXplain system for diagnostic decision support was developed around the same time as INTERNIST-I.

The earliest decision support systems were standalone, but the second wave in clinical decision support, beginning in the 1970s, was the integration of decision support systems into broader clinical information systems. The first two examples of this integration were the Health Evaluation through Logical Processing (HELP) system at the University of Utah and LDS Hospital, and the Regenstrief Medical Records System (RMRS) developed at the Regenstrief Institute in Indianapolis. The HELP system, which was used for many facets of patient care, had support for the development of a variety of kinds of decision support, and was especially well known for its Bayesian reasoning modules. The RMRS was developed, from the ground up, with a large knowledge base of clinical care rules. Both HELP and RMRS are in active use today.

Most current commercially available clinical information systems have some support for clinical decision support, and efforts to standardize representation and enable the sharing of decision support content are ongoing.

Foundations

Development of clinical decision support systems entails a variety of issues. The first step in developing any clinical information system is to identify an important clinical target, and then consider interventions. The most critical database systems related issues are knowledge representation, storage and standards.

Issues of Knowledge Representation

Once a desired clinical decision support target has been identified and relevant medical knowledge has been collected, the knowledge must somehow be represented. Knowledge in clinical decision support systems has been represented in a variety of ways, the most common being if-then rules, expert systems, probabilistic and Bayesian systems and reference content.

Perhaps the simplest form of knowledge is if-then rules. Much of clinical decision support content can be represented this way (for example "if the acetaminophen dose is 10 g per day, alert the user that this is too high" or "if the patient is over 50 years of age and has not had a sigmoidoscopy, recommend one"). These rules are frequently designed to be chained together, although generally in a fixed and predetermined pattern.

More complex than simple if-then rules are expert systems. These systems are composed of large knowledge bases which contain many intermediate states and assertions. Like if-then rules, these rules are composed of an antecedent, a consequent and an implication. However, expert systems are generally designed to elicit emergent behavior from extensive chaining including, in many cases, goal-directed backward chaining.

Probabilistic and Bayesian systems share much in common with if-then rules. However, instead of modeling knowledge and clinical states as deterministic values, they use probabilities. By combining these probabilities with knowledge provided by the user, these systems can estimate the likelihood of various diagnostic possibilities, or the relative utility of different therapeutic modalities. It is important to note that many expert systems employ probabilistic or Bayesian reasoning.

A simpler form of knowledge representation is reference knowledge designed to be read by a human. This form of decision support provides information to the user but expects him or her to formulate a plan of action on his or her own. In many cases knowledge, such as clinical guidelines, can be equivalently modeled as rules or as reference content. Reference content is simpler to construct, but it sometimes can not be as proactive as rule-based content.

Storage of Clinical Knowledge in Database Systems

A key challenge for developers of database systems for clinical decision support is selecting the optimal strategy for storing clinical knowledge in a database. This selection has many tradeoffs among performance, space, maintainability and human readability.

Rule based decision support content is often stored as compiled or interpreted code and, when properly integrated into clinical information support systems, can be very efficient. However, many systems instead choose to store rules in some intermediate form, often indexed according to their trigger (a clinical event, such as a new prescription, which causes decision support rules to fire). A chained hash table with these triggers as its keys and decision support rules to invoke as values can be a particularly efficient representation.

In cases of particularly high transaction volume, where performance is important and the number of rules to evaluate is large, more sophisticated storage and processing mechanisms can be used. One of the most effective in terms of performance (although not necessarily in terms of space) is Charles Fogarty's Rete algorithm. The Rete algorithm is an efficient network-based method for pattern matching in rule-based systems.

Because it is not rule based, reference knowledge requires a different set of storage and retrieval strategies, based largely on the principles of information retrieval. In general, these strategies employ one or some combination of full-text search and metadata queries.

Standards for Sharing Clinical Decision Support Content between Database Systems

In addition to the aforementioned issues of internal representation of clinical knowledge, there are also issues relating to the sharing of clinical decision support content between systems. Several standards for sharing such content have been proposed, beginning with Arden Syntax, a standard for event-driven rule-based decision support content. Other standards, such as Guideline Interchange Format (GLIF) and the related expression language GELLO exist to represent more complex forms of clinical knowledge. While construction of standards for representing clinical knowledge may seem straightforward, issues relating to terminology and a reference model for patient information have proven formidable.

An alternate approach to strict structured knowledge representation formalisms for sharing clinical decision support content is the use of services. Several recent efforts, including SEBASTIAN and SANDS have defined a set of interfaces and, in the case of SANDS, patient data models to help overcome prior difficulties in sharing decision support content.

Key Applications

Applications of clinical decision support can be categorized along a variety of axes, including intervention type (alert, reminder, reference information, etc.), clinical purpose (diagnosis, therapy, prevention), disease target (diabetes, hypertension, cancer, etc.) and user (physician, nurse, patient, etc.).

Several clinical decision support systems have been described in the historical background section. Additional significant systems include:

- Morris Collen's system for "Automated Multiphasic Screening And Diagnosis."
- Howard Bleich's system for diagnosis and treatment of acid-base disorders.
- A system for the diagnosis and management of abdominal complaints developed by F.T. de Dombal.
- The ATTENDING system developed by Perry Miller and designed to critique and suggest improvements to anesthesia plans.
- A system for ventilator management by Dean Sittig.

- A blood product ordering critiquing system by Reed Gardner.
- An antibiotic advising system by Scott Evans.

Experimental Results

There is a long experimental tradition in the field of clinical decision support, and many systems have shown strong results, even for the earliest systems. Warner's system for congenital heart defects was compared favorably to experienced cardiologists, MYCIN proposed clinically appropriate antibiotic therapy 75% of the time (and got better as more rules were added) and INTERNIST performed about as well as average doctors at diagnosis.

Just as significant is the effect that such systems have on physician practice. In a landmark paper, Clem McDonald described the results of an experimental trial performed within the RMRS system. In the trial, half of the physician users of RMRS received patient care suggestions based on the knowledge base of rules, while half did not. Physicians who received the suggestions carried them out 51% of the time, while physicians who did not receive suggestions performed the actions that would have been suggested only 22% of the time. When the reminder system was turned off, physician performance returned almost immediately to baseline.

There have been several significant systematic reviews of clinical decision support systems. A 2005 review by Amit Garg [2] found that decision support systems were associated with improved provider performance in 64% of the controlled trials reviewed. Another systematic review by Ken Kawamoto found that decision support systems improved performance in 68% of trials, and that systems designed to the highest criteria improved performance in 94% of trials.

Cross-references

► Clinical Data and Information Models
► Clinical Prediction Rule

Recommended Reading

1. Bates D.W., Kuperman G.J., and Wang S., et al. Ten commandments for effective clinical decision support: making the practice of evidence-based medicine a reality. J. Am. Med. Inform. Assoc., 10(6):523–530, 2003.
2. Garg A.X., Adhikari N.K., and McDonald H., et al. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. Jama, 293(10):1223–1238, 2005.
3. Kawamoto K., Houlihan C.A., Balas E.A., and Lobach D.F. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. BMJ, 330(7494):765, 2005.
4. Kawamoto K. and Lobach D.F. Design, implementation, use, and preliminary evaluation of SEBASTIAN, a standards-based web service for clinical decision support. In Proc. AMIA Symposium, 2005, pp. 380–384.
5. Kuperman G.J., Gardner R.M., and Pryor T.A. HELP: A Dynamic Hospital Information System. Springer, New York, 1991.
6. Ledley R.S. and Lusted L.B. Reasoning foundations of medical diagnosis; symbolic logic, probability, and value theory aid our understanding of how physicians reason. Science, 130(3366):9–21, 1959.
7. McDonald C.J. Protocol-based computer reminders, the quality of care and the non-perfectability of man. N. Engl. J. Med., 295(24):1351–1355, 1976.
8. Miller R.A., Pople H.E. Myers J.D. Internist-1, an experimental computer-based diagnostic consultant for general internal medicine. N. Engl. J. Med., 307(8):468–476, 1982.
9. Osheroff J.A., Pifer E.A., Sittig D.F., Jenders R.A., and Teich J.M. Improving Outcomes with Clinical Decision Support: an Implementers' Guide. HIMSS, Chicago, 2005.
10. Osheroff J.A., Teich J.M., Middleton B., Steen E.B., Wright A., and Detmer D.E. A roadmap for national action on clinical decision support. J. Am. Med. Inform. Assoc., 14(2):141–145, 2007.
11. Sittig D.F., Wright A., and Osheroff J.A., et al. Grand challenges in clinical decision support. J. Biomed. Inform., 41(2): 387–392, 2007.
12. Shortliffe E.H., Davis R., Axline S.G., Buchanan B.G., Green C.C., and Cohen SN. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the MYCIN system. Comput. Biomed. Res., 8(4):303–320, 1975.
13. Wright A., Goldberg H., Hongsermeier T., and Middleton B. A description and functional taxonomy of rule-based decision support content at a large integrated delivery network. J. Am. Med. Inform. Assoc., 14(4):489–496, 2007.
14. Wright A., Sittig D.F., SANDS: A service-oriented architecture for clinical decision support in a National Health Information Network. J. Biomed. Inform. (2008), doi:10.1016/j.jbi.2008.03.001.
15. Warner H.R., Toronto A.F., Veasey L.G., and Stephenson R. A mathematical approach to medical diagnosis. Application to congenital heart disease. Jama, 177:177–183, 1961.

Clinical Document Architecture

Amnon Shabo (Shvo)
IBM Research Lab-Haifa, Haifa, Israel

Synonyms

CDA; CDA R1; CDA R2

Definition

The Clinical Document Architecture (CDA) is a document markup standard that specifies the structure and semantics of clinical documents for the purpose of exchange and share of patient data. The standard is developed by Health Level Seven (HL7) – a Standards Development Organization [2] focused on the area of healthcare. At the time of writing this entry, two releases of CDA were approved: CDA R1 was approved in 2000 and CDA R2 in 2005. Both releases are part of the HL7 new generation of standards (V3), all derived from a core reference information model (RIM) that assures semantic consistency across the various standards such as laboratory, medications, care provision and so forth. The RIM is based on common data types and vocabularies, and together these components constitute the HL7 V3 Foundation that is an inherent part of the CDA standard specification.

Key Points

Clinical documents such as discharge summaries, operative notes and referral letters are ubiquitous in healthcare and currently exist mostly in paper. The computerized clinical document is similar in purpose to its paper counterpart and the clinician's narratives are a key component of both versions. Narratives are compositions based on the natural language of the writer, while computerized structuring of a document is limited to some computer language. The design of the CDA standard strives to bridge the gap between these "languages" especially when it comes to the mixture of structured and unstructured data intertwined to describe the same phenomena, while addressing two important goals: human readability and machine-processability. The drive to structure medical narratives is also challenging the thin line between art and craftsmanship in the medical practice [3].

The basic structure of a CDA document consists of a header and a body. The header represents an extensive set of metadata about the document such as time stamps, the type of document, encounter details and of course the identification of the patient and those who participated in the documented encounter or service. While the header is a structured part of the document and is similar in the two releases of CDA, the body consists of clinical data organized in sections and only in CDA R2 it enables the formal representation of structured data along with narratives [1]. Data is structured in clinical statements based on entries such as observations, medication administrations, or adverse events where several entries are associated into a compound clinical statement. Nevertheless, only the narrative parts of the CDA body are mandatory, which makes CDA easy to adopt if structured data is not yet available. It is even possible to simply wrap a non-XML document with the CDA header or create a document with a structured header and sections containing only narrative content. The purpose of this design is to encourage widespread adoption, while providing an information infrastructure to incrementally move toward structured documents, serving the goal of semantic interoperability between disparate health information systems.

Beside text, CDA can also accommodate images, sounds, and other multimedia content. It can be transferred within a message and can be understood independently, outside the relaying message and its sending and receiving systems. CDA documents are encoded in Extensible Markup Language (XML), and they derive their machine processable meaning from the RIM, coupled with specific vocabularies.

A CDA document is a collection of information that is intended to be legally authenticated and has to be maintained by an organization entrusted with its care (stewardship). Inherent in the HL7 CDA standard are mechanisms for dealing with the authentication and versioning of documents so that it can be used in medical records enterprise repositories as well as in cross-institutional sharing of personal health information to facilitate continuity of care.

Cross-references

► Electronic Health Record

Recommended Reading

1. Dolin R.H., Alschuler L, Boyer S, Beebe C, Behlen FM, Biron PV, Shabo A. HL7 Clinical Document Architecture, Release 2. J. Am. Med. Inform. Assoc., 13(1):30–39, 2006.
2. Health Level Seven (HL7) – http://www.hl7.org.
3. Shabo A. Synopsis of the Patient Records Section: Structuring the Medical Narrative in Patient Records – A Further Step towards a Multi-Accessible EHR. The IMIA 2004 Yearbook of Medical Informatics: Towards Clinical Bioinformatics, 2004.

Clinical Event

Dan Russler
Oracle Health Sciences, Redwood Shores, CA, USA

Definition

Vernacular Definition

1. In event planning circles, a "clinical event" is an event, e.g., meeting or party, attended by clinicians as opposed to administrative or financial personnel.

Technical Definitions

1. A state transition, normally a "create" or "update" state transition, targeting a record in an electronic medical record system or one of the systems associated with an electronic medical record system.
2. A report generated within a clinical trial that is subsequently evaluated for the presence of an adverse event by a clinical trial Clinical Event Committee.

Words often confused by use of the term "Clinical Event" include: Clinical Event (multiple definitions); Adverse Event; Clinical Act; Patient Event, Information Event.

The primary technical definition of "clinical event" includes the kind of "events" that are monitored by a "clinical event monitor"[1–3] used in synchronous or asynchronous decision support functions. Examples of these events include clinical orders, electronic medical record entries, admission, transfer and discharge notifications, lab results, and patient safety reports. These events trigger state transitions in an electronic medical record system or related system.

Typically, once the clinical event monitoring system, such as an HL7 Arden Syntax-based system, discovers a state transition, in the electronic medical record system, a decision support rule is applied to the clinical event and related data in order to determine whether a notification of a person or another system is required.

Key Points

"Event" or "Action" analysis traces its roots to the work of Aristotle on propositions. Propositions usually follow the form of Subject-Predicate and, upon analysis, may be found to be "true" or "false." The classic example of a proposition is "Socrates is a man." "Socrates" is the "Subject" and "is a man" is the "Predicate." An analogous proposition in healthcare is "Peter has a potassium level of 5.5 mg/dl." Clinical Events are propositions in healthcare that may be evaluated themselves by clinicians as "true" or "false" or may be applied in rules that evaluate to true or false.

For example, the creation of a record asserting that "Peter has a potassium level of 5.5 mg/dl" might trigger a clinical event monitoring system to implement the rule: "If potassium level record created, then evaluate if ("record value" >5.0); if "true," then notify Dr. X."

"Event-driven programming" as opposed to "procedural programming" utilizes the same kinds of predicate logic in evaluating state transitions or triggers to state transitions in a modern computer-programming environment. Consequently, Clinical Events drive programming logic in many modern systems.

The HL7 Reference Information Model (RIM) describes clinical events; the term "Act" in the RIM identifies objects that are instantiated in XML communications between systems or in records within the electronic healthcare systems themselves. These "Acts" correspond to "clinical events" used for monitoring systems in healthcare. However, in the RIM, "Event" is defined narrowly as an instance of an Act that has been completed or is in the process of being completed. Clinical event monitoring systems may also evaluate HL7 "Orders or Requests" or other kinds of "Act" instances as events of interest (www.hl7.org).

Cross-references

- ▶ Clinical Observation
- ▶ Clinical Order
- ▶ Interface Engines in Healthcare
- ▶ Event Driven Architecture
- ▶ HL7 Reference Information Model
- ▶ Predicate Logic
- ▶ Propositions

Recommended Reading

1. Glaser J., et al. Impact of information events on medical care. HIMSS, 1996.
2. Hripisak G., et al. Design of a clinical event monitor. Comp. Biomed. Res., 29:194–221, 1996.
3. McDonald C. Action-oriented Decisions in Ambulatory Medicine. Yearbook Medical Publishers, Chicago, IL, 1981.

Clinical Genetics

► Implications of Genomics for Clinical Informatics

Clinical Genomics

► Implications of Genomics for Clinical Informatics

Clinical Judgment

► Clinical Observation

Clinical Knowledge Base

► Clinical Knowledge Repository

Clinical Knowledge Directory

► Clinical Knowledge Repository

Clinical Knowledge Management Repository

► Clinical Knowledge Repository

Clinical Knowledge Repository

Roberto A. Rocha
Partners Healthcare System, Inc., Boston, MA, USA

Synonyms

Clinical knowledge base; Clinical content repository; Clinical content database; Clinical knowledge management repository; Clinical content registry; Clinical knowledge directory

Definition

A clinical knowledge repository (CKR) is a multipurpose storehouse for clinical knowledge assets. "Clinical knowledge asset" is a generic term that describes any type of human or machine-readable electronic content used for computerized clinical decision support. A CKR is normally implemented as an enterprise resource that centralizes a large quantity and wide variety of clinical knowledge assets. A CKR provides integrated support to all asset lifecycle phases such as authoring, review, activation, revision, and eventual inactivation. A CKR routinely provides services to search, retrieve, transform, merge, upload, and download clinical knowledge assets. From a content curation perspective, a CKR has to ensure proper asset provenance, integrity, and versioning, along with effective access and utilization constraints compatible with collaborative development and deployment activities. A CKR can be considered a specialized content management system, designed specifically to support clinical information systems. Within the context of clinical decision support systems, a CKR can be considered a special kind of knowledge base – one specially designed to manage multiple types of human and machine-readable clinical knowledge assets.

Key Points

In recent years, multiple initiatives have attempted to better organize, filter, and apply the ever-growing biomedical knowledge. Among these initiatives, one of the most promising is the utilization of computerized clinical decision support systems. Computerized clinical decision support can be defined as computer systems that provide the correct amount of relevant knowledge at the appropriate time and context, contributing to improved clinical care and outcomes. A wide variety of knowledge-driven tools and methods have resulted in multiple modalities of clinical decision support, including information selection and retrieval, information aggregation and presentation, data entry assistance, event monitors, care workflow assistance, and descriptive or predictive modeling. A CKR provides an integrated storage platform that enables the creation and maintenance of multiple types of knowledge assets. A CKR ensures that different modalities of decision support can be combined to properly support the activities of clinical workers. Core requirements guiding the implementation of a CKR include clinical knowledge asset provenance (metadata), versioning, and integrity. Other essential requirements include the proper representation of access and utilization constraints, taking into account

the collaborative nature of asset development processes and deployment environments. Another fundamental requirement is to aptly represent multiple types of knowledge assets, where each type might require specialized storage and handling. The CKR core requirements are generally similar to those specified for other types of repositories used for storage and management of machine-readable assets.

Historical Background

Biomedical knowledge has always been in constant expansion, but unprecedented growth is being observed during the last decade. Over 30% of the 16.8 million citations accumulated by MEDLINE until December of 2007 were created in the last 10 years, with an average of over 525,000 new citations per year [5]. The number of articles published each year is commonly used as an indicator of how much new knowledge the scientific community is creating. However, from a clinical perspective, particularly for those involved with direct patient care, the vast amount of new knowledge represents an ever-growing gap between what is known and what is routinely practiced. Multiple initiatives in recent years have attempted to better organize, filter, and apply the knowledge being generated. Among these various initiatives, one of the most promising is the utilization of computerized clinical decision support systems [6]. In fact, some authors avow that clinical care currently mandates a degree of individualization that is inconceivable without computerized decision support [1].

Computerized clinical decision support can be defined as computer systems that provide the correct amount of relevant knowledge at the appropriate time and context, ultimately contributing to improved clinical care and outcomes [3]. Computerized clinical decision support has been an active area of informatics research and development for the last three decades [2]. A wide variety of knowledge-driven tools and methods have resulted in multiple modalities of clinical decision support, including information selection and retrieval (e.g., infobuttons, crawlers), information aggregation and presentation (e.g., summaries, reports, dashboards), data entry assistance (e.g., forcing functions, calculations, evidence-based templates for ordering and documentation), event monitors (e.g., alerts, reminders, alarms), care workflow assistance (e.g., protocols, care pathways, practice guidelines), and descriptive or predictive modeling (e.g., diagnosis, prognosis, treatment planning, treatment outcomes). Each modality requires specific types of knowledge assets, ranging from production rules to mathematical formulas, and from automated workflows to machine learning models. A CKR provides an integrated storage platform that enables the creation and maintenance of multiple types of assets using knowledge management best practices [4].

The systematic application of knowledge management processes and best practices to the biomedical domain is a relatively recent endeavor [2]. Consequently, a CKR should be seen as a new and evolving concept that is only now being recognized as a fundamental component for the acquisition, storage, and maintenance of clinical knowledge assets. Most clinical decision support systems currently in use still rely on traditional knowledge bases that handle a single type of knowledge asset and do not provide direct support for a complete lifecycle management process. Another relatively recent principle is the recognition that different modalities of decision support have to be combined and subsequently integrated with information systems to properly support the activities of clinical workers. The premise of integrating multiple modalities of clinical decision support reinforces the need for knowledge management processes supported by a CKR.

Foundations

Core requirements guiding the implementation of a CKR include clinical knowledge asset provenance (metadata), versioning, and integrity. Requirements associated with proper access and utilization constraints are also essential, particularly considering the collaborative nature of most asset development processes and deployment environments. Another fundamental requirement is to aptly represent multiple types of knowledge assets, where each type might require specialized storage and handling. The CKR core requirements are generally similar to those specified for other types of repositories used for storage and management of machine-readable assets (e.g., "ebXML Registry" (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev = regrep)).

Requirements associated with asset provenance can be implemented using a rich set of metadata properties that describe the origin, purpose, evolution, and status of each clinical knowledge asset. The metadata properties should reflect the information that needs to be captured during each phase of the knowledge asset lifecycle process, taking into account multiple iterative

authoring and review cycles, followed by a possibly long period of clinical use that might require multiple periodic revisions (updates). Despite the diversity of asset types, each with a potentially distinct lifecycle process, a portion of the metadata properties should be consistently implemented, enabling basic searching and retrieval services across asset types. Ideally, the shared metadata should be based on metadata standards (e.g., "Dublin Core Metadata Element Set" (http://dublincore.org/documents/dces/)). The adoption of standard metadata properties also simplifies the integration of external collections of clinical knowledge assets in a CKR. In addition to a shared set of properties, a CKR should also accommodate extended sets of properties specific for each clinical knowledge asset type and its respective lifecycle process. Discrete namespaces are commonly used to represent type-specific extended metadata properties.

Asset version and status, along with detailed change tracking, are vital requirements for a CKR. Different versioning strategies can be used, but as a general rule there should be only one clinically active version of any given knowledge asset. This general rule is easily observed if the type and purpose of the clinical knowledge asset remains the same throughout its lifecycle. However, a competing goal is created with the very desirable evolution of human-readable assets to become machine-readable. Such evolution invariably requires the creation of new knowledge assets of different types and potentially narrower purposes. In order to support this "natural" evolution, a CKR should implement the concept of asset generations, while preserving the change history that links one generation to the next. Also within a clinical setting, it is not uncommon to have to ensure that knowledge assets comply with, or directly implement, different norms and regulations. As a result, the change history of a clinical knowledge asset should identify the standardization and compliance aspects considered, enabling subsequent auditing and/or eventual certification.

Ensuring the integrity of clinical knowledge assets is yet another vital requirement for a CKR. Proper integrity guarantees that each asset is unique within a specific type and purpose, and that all its required properties are accurately defined. Integrity requirements also take into account the definition and preservation of dependencies between clinical knowledge assets. These dependencies can be manifested as simple hyperlinks, or as integral content defined as another independent asset. Creating clinical knowledge assets from separate components or modules (i.e., modularity) is a very desirable feature in a CKR – one that ultimately contributes to the overall maintainability of the various asset collections. However, modularity introduces important integrity challenges, particularly when a new knowledge asset is being activated for clinical use. Activation for clinical use requires a close examination of all separate components, sometimes triggering unplanned revisions of components already in routine use. Another important integrity requirement is the ability to validate the structure and the content of a clinical knowledge asset against predefined templates (schemas) and dictionaries (ontologies). Asset content validation is essential for optimal integration with clinical information systems. Ideally, within a given healthcare organization all clinical information systems and the CKR should utilize the same standardized ontologies.

Contextual characteristics of the care delivery process establish the requirements associated with proper access, utilization, and presentation of the clinical knowledge assets. The care delivery context is a multidimensional constraint that includes characteristics of the patient (e.g., gender, age group, language, clinical condition), the clinical worker (e.g., discipline, specialty, role), the clinical setting (e.g., inpatient, outpatient, ICU, Emergency Department), and the information system being used (e.g., order entry, documentation, monitoring), among others. The care delivery context normally applies to the entire clinical knowledge asset, directly influencing search, retrieval, and presentation services. The care delivery context can also be used to constrain specific portions of a knowledge asset, including links to other embedded assets, making them accessible only if the constraints are satisfied. An important integrity challenge created by the systematic use of the care delivery context is the need for reconciling conflicts caused by incompatible asset constraints, particularly when different teams maintain the assets being combined. In this scenario, competing requirements are frequently present, namely the intention to maximize modularity and reusability versus the need to maximize clinical specificity and ease or use.

The accurate selection, retrieval, and presentation of unstructured assets is generally perceived as a simple but very useful modality of clinical decision support, particularly if the information presented to the clinical

worker is concise and appropriate to the care being delivered. However, the appropriateness of the information is largely defined by the constraints imposed by the aforementioned care delivery context. Moreover, the extent of indexing ("retrievability") of most collections of unstructured clinical knowledge assets is not sufficient to fully recognize detailed care delivery context expressions. Ultimately, the care delivery context provides an extensible mechanism for defining the appropriateness of a given clinical knowledge asset in response to a wide variety of CKR service requests.

The requirements just described are totally or partially implemented as part of general-purpose (enterprise) content management systems. However, content management systems have been traditionally constructed for managing primarily human-readable electronic content. Human-readable content, more properly characterized as unstructured knowledge assets, include narrative text, diagrams, and multimedia objects. When combined, these unstructured assets likely represent the largest portion of the inventory of clinical knowledge assets of any healthcare institution. As a result, in recent years different healthcare organizations have deployed CKRs using enterprise content management systems, despite their inability to manage machine-readable content.

Key Applications

Computerized Clinical Decision Support, Clinical Knowledge Engineering, Clinical Information Systems.

Cross-references

- ▶ Biomedical Data/Content Acquisition, Curation
- ▶ Clinical Data Acquisition, Storage and Management
- ▶ Clinical Decision Support
- ▶ Dublin Core
- ▶ Evidence Based Medicine
- ▶ Executable Knowledge
- ▶ Metadata
- ▶ Reference Knowledge

Recommended Reading

1. Bates D.W. and Gawande A.A. Improving safety with information technology. N. Engl. J. Med. 348(25):2526–2534, 2003.
2. Greenes R.A (ed.). Clinical Decision Support: The road ahead. Academic Press, Boston, 2007, pp. 544.
3. Osheroff J.A., Teich J.M., Middleton B., Steen E.B., Wright A., and Detmer D.E. A roadmap for national action on clinical decision support. J. Am. Med. Inform. Assoc., 14(2):141–145, 2007.
4. Rocha R.A., Bradshaw R.L., Hulse N.C., and Rocha B.H.S.C. The clinical knowledge management infrastructure of Intermountain Healthcare. In: Clinical Decision Support: The road ahead, RA.Greenes (ed.). Academic Press, Boston, 2007, pp. 469–502.
5. Statistical Reports on MEDLINE®/PubMed® Baseline Data, National Library of Medicine, Department of Health and Human Services [Online]. Available at: http://www.nlm.nih.gov/bsd/licensee/baselinestats.html. Accessed 8 Feb 2008.
6. Wyatt J.C. Decision support systems. J. R. Soc. Med., 93(12): 629–633, 2000.

Clinical Nomenclatures

▶ Clinical Ontologies

Clinical Observation

Dan Russler
Oracle Health Sciences, Redwood Shores, CA, USA

Synonyms

Clinical result; Clinical judgment; Clinical test; Finding of observation

Definition

1. The act of measuring, questioning, evaluating, or otherwise observing a patient or a specimen from a patient in healthcare; the act of making a clinical judgment.
2. The result, answer, judgment, or knowledge gained from the act of observing a patient or a specimen from a patient in healthcare.

These two definitions of "observation" have caused confusion in clinical communications, especially when applying the term to the rigor of standardized terminologies. When developing a list of observations, the terminologists have differed on whether the list of terms should refer to the "act of observing" or the "result of the observation."

Logical Observation Identifiers Names and Codes (LOINC) (www.loinc.org) focus on observation as the "act of observing." Systematized Nomenclature of Medicine (SNOMED) (www.ihtsdo.org) asserts that "General finding of observation of patient" is a synonym for "General observation of patient." Of note is the analysis in HL7 that identifies many shared

attributes between descriptions of the act of observing and the result obtained. As a consequence, in HL7 Reference Information Model (RIM), both the act of observing and the result of the observation are contained in the same Observation Class (www.hl7.org).

Key Points

The topic of clinical observation has been central to the study of medicine since medicine began. Early physicians focused on the use of all five senses in order to make judgments about the current condition of the patient, i.e., diagnosis, or to make judgments about the future of patients, i.e., prognosis. Physical exam included sight, touch, listening, and smell. Physicians diagnosed diabetes by tasting the urine for sweetness.

As more tests on bodily fluids and tissues were discovered and used, the opportunity for better diagnosis and prognosis increased. Philosophy of science through the centuries often included the study of clinical observation in addition to the study of other observations in nature.

During the last century, the study of rigorous testing techniques that improve the reproducibility and interpretation of results has included the development of extensive nomenclatures for naming the acts of observation and observation results, e.g., LOINC and SNOMED. These terminologies were developed in part to support the safe application of expert system rules to information recorded in the electronic health care record.

The development of the HL7 Reference Information Model (RIM) was based on analysis of the "act of observing" and the "result of the act of observing" [1]. Today, new Entity attributes proposed for the HL7 RIM are evaluated for inclusion based partly on whether the information is best communicated in a new attribute for an HL7 Entity or best communicated in an HL7 Observation Act.

Improved standardization of clinical observation techniques, both in the practice of bedside care and the recording of clinical observations in electronic healthcare systems is thought to be essential to the continuing improvement of healthcare and patient safety.

Cross-references

- ▶ Clinical Event
- ▶ Clinical Order
- ▶ Interface Engines in Healthcare

Recommended Reading

1. Russler D., et al. Influences of the unified service action model on the HL7 reference information model. In JAMIA Symposium Supplement, Proceedings SCAMC, 1999, pp. 930–934.

Clinical Ontologies

Yves A. Lussier, James L. Chen
University of Chicago, Chicago, IL, USA

Synonyms

Clinical terminologies; Clinical nomenclatures; Clinical classifications

Definition

An ontology is a formal representation of a set of heterogeneous concepts. However, in the life sciences, the term clinical ontology has also been more broadly defined as also comprising all forms of classified terminologies, including classifications and nomenclatures. Clinical ontologies provide not only a controlled vocabulary but also relationships among concepts allowing computer reasoning such that different parties, like physicians and insurers, can efficiently answer complex queries.

Historical Background

As the life sciences integrates increasingly sophisticated systems of patient management, different means of data representation have had to keep pace to support user systems. Simultaneously, the explosion of genetic information from breakthroughs from the Human Genome Project and gene chip technology have further expedited the need for robust, scalable platforms for handling heterogeneous data. Multiple solutions have been developed by the scientific community to answer these challenges at all different levels of biology.

This growing field of "systems medicine" starts humbly at the question – how can one best capture and represent complex data in a means that can be understood globally without ambiguity? In other words, does the data captured have the same semantic validity after retrieval as it did prior? These knowledgebases are in of themselves organic. They need to be able to expand, shrink, and rearrange themselves based on user or system needs. This entry will touch upon existing clinical ontologies used in a variety of applications.

Foundations

The complexity of biological data cannot be understated. Issues generally fall into challenges with (i) definition, (ii) context, (iii) composition, and (iv) scale. One cannot even take for granted that the term "genome" is well-understood. Mahner found five different characterizations for the term "genome" [8]. Ontologies then provide a means of providing representational consistency through their structure and equally important provide the ability to connect these terms together in a semantically informative and computationally elegant manner [9]. This has led to their ubiquity in the life sciences. Formal ontologies are designated using frames or description logics [5]. However, few life science knowledgebases are represented completely in this manner due to difficulties with achieving consensus on definitions regarding the terms and the effort required to give context to the terms. Thus, this article defines well-organized nomenclatures and terminologies as clinical ontologies – regardless if their terms adhere to strict formalism.

Looking at elevations in gene expression, it matters what organism and under what experimental conditions the experiment was conducted. Clinical context changes the meaning of terms. The term "cortex" can either indicate a part of the kidney or that of the brain. Generalized or "essential hypertension" can be what is known colloquially as "high blood pressure" or localized to the lungs as "pulmonary hypertension." One can have pulmonary hypertension but not essential hypertension. This leads to the next representational challenge – that of composition. Should hypertension be represented implicitly as "essential hypertension" and as "pulmonary hypertension"? Or should it be stored explicitly as "hypertension" with a location attribute? These representational decisions are driven by the queries that may be asked. The difficulty arises in anticipating the queries and in post-processing of the query to split the terminological components of the overall concept. Finally, the knowledge model needs to be able to scale upward. The same decision logic that was relevant when the knowledgebase contained 100 concepts needs to still be relevant at 1,000,000 concepts.

Properties of Clinical Ontologies

Ontologies vary widely in their degree of formalism and design. With this comes differing computability. In 1998, Cimino proposed desirable properties for purposes of clinical computation [3,4]. Table 1 summarizes the overall properties of the commonly used clinical ontologies.

1. Concept-oriented: a single concept is the preferred unit
2. Formal semantic definition: well-defined terms
3. Nonredundancy: each concept needs to be unique
4. Nonambiguity: different concepts should not overlap or be conflated
5. Relationships: the structure of connections between concepts differentiate ontologies:

- Monohierarchy (tree): each concept only has one parent concept
- Polyhierarchy: each concept may multiply inherit from multiple parents
- Directed Acycle Graph (DAG): there are no cycles in the graph – in other words, children concepts may not point to parent terms

Key Applications

This section reviews different, well-used life science ontologies used to annotate datasets. First, this discussion summarizes a select number of archetypal clinical

Clinical Ontologies. Table 1. Properties of clinical ontologies

Ontology	Architecture					Relationship
	Concept oriented	Formal semantic definition	Concept permanence	Nonredundancy	Uniqueness	
ICD-9	+		±	+	+	M
LOINC	±		+	+		P
CPT	±		±			M
SNOMED	+	+	+	+	+	DAG
UMLS	+		+	+	+	CG

M = Monohierarchy/tree, P = Polyhierarchy, DAG = Directed Acyclic Graph, CG = cyclic graph

Clinical Ontologies. Table 2. Coverage of classification, nomenclatures and ontologies

Ontology	Content						Number of concepts (order of magnitude)
	Diseases	Anatomy	Morphology	Labs	Procedures	Drugs	
ICD-9	X						10^4
LOINC				X			10^5
CPT					X		10^4
SNOMED	X	X	X	X	X	X	10^5
UMLS	X	X	X	X	X	X	10^6

ontologies that comprise one or several types of clinical entities such as diseases, clinical findings, procedures, laboratory measurements, and medications. Table 2 below summarizes the content coverage of each of archetypal health ontologies.

Prototypical Clinical Ontologies

a. The Systematized Nomenclature of Medicine (SNOMED CT) SNOMED CT is the most extensive set of publically available collection of clinical concepts. It is organized as a directed acyclic graph (DAG) and contains class/subclass relationships and partonomy relationships. It is maintained by the College of American Pathologists and is available in the United States through a license from the National Library of Medicine in perpetuity. SNOMED CT is one of the designated data standards for use in U.S. Federal Government systems for the electronic exchange of clinical health information. SNOMED CT is now owned by the International Healthcare Terminology Standards Development Organization [6].

b. International Statistical Classification of Diseases (ICD-9, ICD-10, ICD-CM) ICD-9 and ICD 10 are detailed ontologies of disease and symptomatology used ubiquitiously for reimbursement systems (i.e., Medicare/Medicaid) and automated decision support in medicine. ICD-10 is used worldwide for morbidity and mortality statistics. Owned by the World Health Organization (WHO), licenses are available generally free for research. ICD-9 CM is a subtype of ICD-9 with clinical modifiers for billing purposes [11].

c. Medical Subject Headings (MeSH) MeSH grew out of an effort by the NLM for indexing life science journal articles and books. {Nelson S.J., 2001 #6}. The extensive controlled vocabulary MeSH serves as the backbone of the MEDLINE/PubMed article database. MeSH can be browsed and downloaded free of charge on the Internet [10].

d. International Classification of Primary Care (ICPC-2, ICPC-2-E) ICPC is a primary care encounter classification system [12]. It has a biaxial structure of 17 clinical systems and 7 types of data. It allows for the classification of the patient's reason for encounter (RFE), the problems/diagnosis managed, primary care interventions, and the ordering of the data. of the primary care session in an episode of care structure. ICPC-2-E refers to a revised electronic version.

e. Diagnostic and Statistical Manual of Mental Disorders (DSM-IV, DSM-V) The DSM is edited and published by the American Psychiatric Association provides categories of and diagnosis criteria for mental disorders [2]. It is used extensively by clinicians, policy makers and insurers. The original version of the DSM was published in 1962. DSM-V is due for publication in May 2012. The diagnosis codes are developed to be compatible with ICD-9.

f. Logical Observation Identifiers Names and Codes (LOINC) LOINC is a database protocol aimed at standardizing laboratory and clinical codes. The Regenstrief Institute, Inc, maintains the LOINC database and supporting documentation. LOINC is endorsed by the American Clinical Laboratory Association and College of American Pathologist and is one of the accepted standards by the US Federal Government for information exchange [7].

g. Current Procedural Terminology (CPT) The CPT code set is owned and maintained by the American Medical Association through the CPT Editorial Panel [1]. The CPT code set is used extensively to

communicate medical and diagnostic services that were rendered among physicians and payers. The current version is the CPT 2008.

Cross-references

- ▶ Anchor text
- ▶ Annotation
- ▶ Archiving Experimental Data
- ▶ Biomedical Data/Content Acquisition, Curation
- ▶ Classification
- ▶ Clinical Data Acquisition, Storage and Management
- ▶ Clinical Data and Information Models
- ▶ Clinical Decision Support
- ▶ Data Integration Architectures and Methodology for the Life Sciences
- ▶ Data Types in Scientific Data Management
- ▶ Data Warehousing for Clinical Research
- ▶ Digital Curation
- ▶ Electronic Health Record
- ▶ Fully-Automatic Web Data Extraction
- ▶ Information Integration Techniques for Scientific Data
- ▶ Integration of Rules and Ontologies
- ▶ Logical Models of Information Retrieval
- ▶ Ontologies
- ▶ Ontologies and Life Science Data Management
- ▶ Ontology
- ▶ Ontology Elicitation
- ▶ Ontology Engineering
- ▶ Ontology Visual Querying
- ▶ OWL: Web Ontology Language
- ▶ Query Processing Techniques for Ontological Information
- ▶ Semantic Data Integration for Life Science Entities
- ▶ Semantic Web
- ▶ Storage Management
- ▶ Taxonomy: Biomedical Health Informatics
- ▶ Web Information Extraction

Recommended reading

1. American Medical Association [cited; Available at: http://www.cptnetwork.com].
2. American Psychiatric Association [cited; Available at: http://www.psych.org/MainMenu/Research/DSMIV.aspx].
3. Cimino J.J. Desiderata for controlled medical vocabularies in the twenty-first century. Methods Inf. Med., 37(4–5):394–403, 1998.
4. Cimino J.J. In defense of the Desiderata. [comment]. J. Biomed. Inform., 39(3):299–306, 2006.
5. Gruber T.R. Toward principles for the design of ontologies used for knowledge sharing. Int. J. Hum. Comput. Stud., 43(4–5): 907–928, 1995.
6. I.H.T.S.D. [cited; Available from: http://www.ihtsdo.org/our-standards/snomed-ct].
7. Khan A.N. et al. Standardizing laboratory data by mapping to LOINC. J Am Med Inform Assoc, 13(3):353–355, 2006.
8. Mahner M. and Kary M. What exactly are genomes, genotypes and phenotypes? And what about phenomes? J. Theor. Biol., 186(1):55–63, 1997.
9. Musen M.A. et al. PROTEGE-II: computer support for development of intelligent systems from libraries of components. Medinfo, 8 (Pt 1):766–770, 1995.
10. Nelson S.J., Johnston D., and Humphreys. B.L Relationships in medicical subject headings. In Relationships in the Organization of Knowledge, A.B. Carol, G. Rebecca (eds.). Kluwer, Dordecht, 2001, pp. 171–184.
11. World Health Organization [cited; Available at: http://www.who.int/classifications/icd/en/].
12. World Organization of National Colleges, Academies, and Academic Associations of General Practitioners/Family Physicians, ICPC. International Classification of Primary Care. Oxford University Press, Oxford, 1987.

Clinical Order

Dan Russler
Oracle Health Sciences, Redwood Shores, CA, USA

Synonyms

Order item; Service order; Service request; Service item; Procedure order; Procedure request

Definition

The act of requesting that a service be performed for a patient.

Clinical orders in healthcare share many characteristics with purchase orders in other industries. Both clinical orders and purchase orders establish a customer-provider relationship between the person placing the request for a service to be provided and the person or organization filling the request. In both cases, the clinical order and purchase order are followed by either a promise or intent to fill the request, a decline to fill the request, or a counter-proposal to provide an alternate service. In both scenarios, an authorization step such as an insurance company authorization or a credit company authorization may be required. Therefore, the dynamic flow of communications between a placer and filler in a clinical order

management system and a purchase order management system are very similar.

Both clinical order and purchase order management systems maintain a catalog of items that may be requested. These items in both kinds of systems may represent physical items from supply or services from a service provider. Each of these items in both kinds of systems is associated with an internally unique identifier, a text description, and often a code. Dates, status codes, delivery locations, and other attributes of a clinical order and purchase order are also similar. Therefore, in addition to similarities in the dynamic flow of order communications, the structure of the content in clinical orders and purchase orders is similar.

Logical Observation Identifiers Names and Codes (LOINC) (www.loinc.org) describe many of the requested services in healthcare, especially in laboratory systems. Other procedural terminologies exist for healthcare, either independently in terminologies like LOINC or included in more comprehensive terminologies such as Systematized Nomenclature of Medicine (SNOMED) (www.ihtsdo.org).

Key Points

Clinical orders exist in the context of a larger clinical management, process. The order management business process of an organization, that includes defining a catalog of services to be provided and then allowing people to select from the catalog of services, is common in many industries. However, the decision support opportunities for helping providers select the optimum set of services for a patient are often more complex in healthcare than occurs in other industries. The outcomes of this selection process are studied in clinical research, clinical trials on medications and devices, and in organizational quality improvement initiatives. Finally, the outcomes of the service selection process are used to improve the clinical decision support processes utilized by providers selecting services for patients. This business process in healthcare as well as in many other industries describes a circular feedback loop defined by the offering of services, the selection of services, the delivery of services, the outcome of services, and finally, the modification of service selection opportunities and decision support.

In the HL7 Reference Information Model (RIM), "ACT" classes sub-typed with the moodCode attribute support the healthcare improvement process (www.hl7.org). These objects with process "moods" support the sequence of objects created during the execution of a process defined in Business Process Execution Language (BPEL) in a service oriented architecture that begins with an "order", evolves into an "appointment", which then is completed as an "event". The reason the term "mood" is used is that the values of the moodCode attribute are analogous to the models of verbs in many languages, e.g., the "Definition mood" used to define service catalogs corresponds to the "infinitive" verbal mood, i.e., a possible action; the "Request or Order mood" corresponds to the "imperative" verbal mood; the "Event mood" corresponds to the "indicative" verbal mood; and the "Goal mood," which describes the desired outcome of the selected service, corresponds to the "subjunctive" verbal mood.

Cross-references

- ▶ Clinical Event
- ▶ Clinical Observation
- ▶ Interface Engines in Healthcare

Clinical Research Chart

- ▶ Data Warehousing for Clinical Research

Clinical Result

- ▶ Clinical Observation

Clinical Terminologies

- ▶ Clinical Ontologies

Clinical Test

- ▶ Clinical Observation

Clock

- ▶ Physical Clock
- ▶ Time-Line Clock

Closed Itemset Mining and Non-redundant Association Rule Mining

Mohammed J. Zaki
Rensselaer Polytechnic Institute, Troy, NY, USA

Synonyms

Frequent concepts; Rule bases

Definition

Let I be a set of binary-valued attributes, called *items*. A set $X \subseteq I$ is called an *itemset*. A transaction database D is a multiset of itemsets, where each itemset, called a transaction, has a unique identifier, called a tid. The *support* of an itemset X in a dataset D, denoted $sup(X)$, is the fraction of transactions in D where X appears as a subset. X is said to be a *frequent* itemset in D if $sup(X) \geq minsup$, where *minsup* is a user defined minimum support threshold. An (frequent) itemset is called *closed* if it has no (frequent) superset having the same support.

An *association rule* is an expression $A \Rightarrow B$, where A and B are itemsets, and $A \cap B = \emptyset$. The *support* of the rule is the joint probability of a transaction containing both A and B, given as $sup(A \Rightarrow B) = P(A \wedge B) = sup(A \cup B)$. The *confidence* of a rule is the conditional probability that a transaction contains B, given that it contains A, given as: $conf(A \Rightarrow B) = P(B|A) = \frac{P(A \wedge B)}{P(A)} = \frac{sup(A \cup B)}{sup(A)}$. A rule is frequent if the itemset $A \cup B$ is frequent. A rule is confident if $conf \geq minconf$, where *minconf* is a user-specified minimum threshold. The aim of non-redundant association rule mining is to generate a *rule basis*, a small, non-redundant set of rules, from which all other association rules can be derived.

Historical Background

The notion of closed itemsets has its origins in the elegant mathematical framework of Formal Concept Analysis (FCA) [3], where they are called *concepts*. The task of mining frequent closed itemsets was independently proposed in [7,11]. Approaches for non-redundant association rule mining were also independently proposed in [1,9]. These approaches rely heavily on the seminal work on rule bases in [5,6]. Efficient algorithms for mining frequent closed itemsets include CHARM [10], CLOSET [8] and several new approaches described in the Frequent Itemset Mining Implementations workshops [4].

Foundations

Let $I = \{i_1, i_2, \ldots, i_m\}$ be the set of items, and let $T = \{t_1, t_2, \ldots, t_n\}$ be the set of tids, the transaction identifiers. Just as a subset of items is called an itemset, a subset of tids is called a tidset. Let $\mathbf{t} : 2^I \rightarrow 2^T$ be a function, defined as follows:

$$\mathbf{t}(X) = \{t \in T \mid X \subseteq \mathbf{i}(t)\}$$

That is, $\mathbf{t}(X)$ is the set of transactions that contain *all* the items in the itemset X. Let $\mathbf{i} : 2^T \rightarrow 2^I$ be a function, defined as follows:

$$\mathbf{i}(Y) = \{i \in I \mid \forall t \in Y,\ t \text{ contains x}\}$$

That is, $\mathbf{i}(T)$ is the set of items that are contained in *all* the tids in the tidset Y. Formally, an itemset X is closed if $\mathbf{i} \circ \mathbf{t}(X) = X$, i.e., if X is a fixed-point of the closure operator $\mathbf{c} = \mathbf{i} \circ \mathbf{t}$. From the properties of the closure operator, one can derive that X is the maximal itemset that is contained in all the transactions $\mathbf{t}(X)$, which gives the simple definition of a closed itemset, namely, a closed itemset is one that has no superset that has the same support.

Based on the discussion above, three main families of itemsets can be distinguished. Let $\mathcal{F}$ denote the set of all frequent itemsets, given as

$$\mathcal{F} = \{X \mid X \subseteq I \text{ and } sup(\mathrm{X}) \geq minsup\}$$

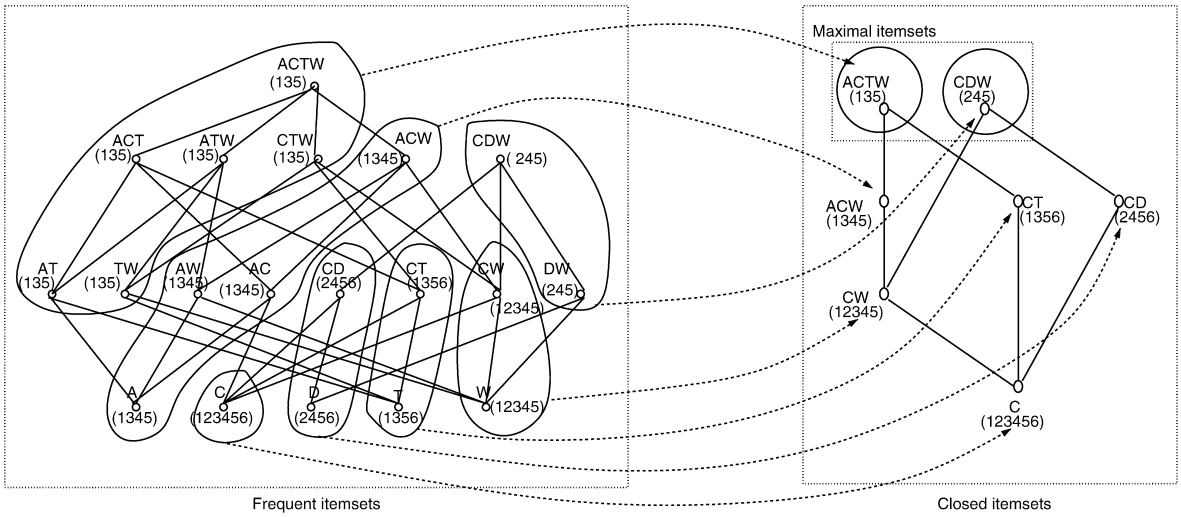
Let $\mathcal{C}$ denote the set of all closed frequent itemsets, given as

$$\mathcal{C} = \{X | X \in \mathcal{F} \text{ and } \nexists \mathrm{Y} \supset \mathrm{X} \text{ with } sup(\mathrm{X}) = sup(\mathrm{Y})\}$$

Finally, let $\mathcal{M}$ denote the set of all *maximal* frequent itemsets, given as

$$\mathcal{M} = \{X | X \in \mathcal{F} \text{ and } \nexists \mathrm{Y} \supset \mathrm{X}, \text{ such that } \mathrm{Y} \in \mathcal{F}\}$$

The following relationship holds between these sets: $\mathcal{M} \subseteq \mathcal{C} \subseteq \mathcal{F}$, which is illustrated in Fig. 1, based on the example dataset shown in Table 1 and using minimum support $minsup = 3$. The *equivalence classes* of itemsets that have the same tidsets have been shown clearly; the largest itemset in each equivalence class is a closed itemset. The figure also shows that the maximal itemsets are a subset of the closed itemsets.



Closed Itemset Mining and Non-redundant Association Rule Mining. Figure 1. Frequent, closed frequent and maximal frequent itemsets.

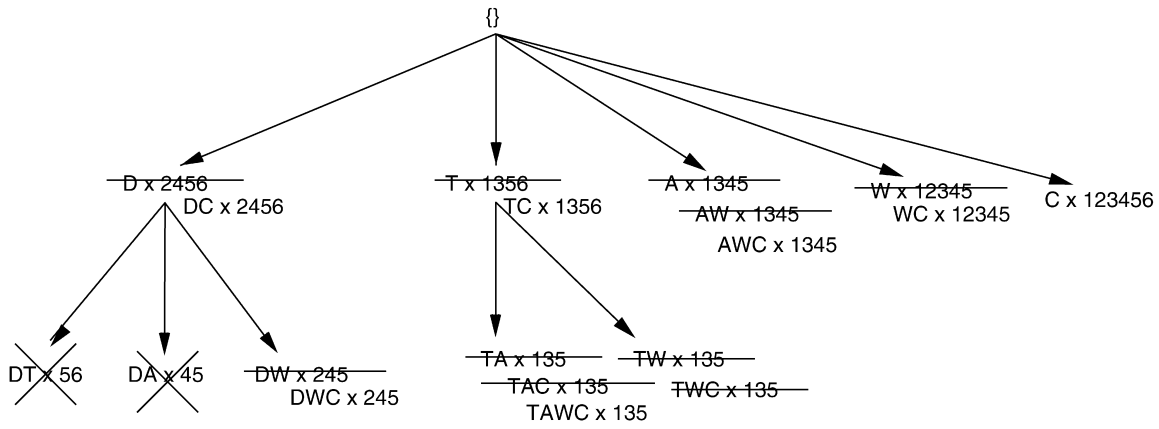
Closed Itemset Mining and Non-redundant Association Rule Mining. Table 1. Example transaction dataset

	i(*t*)
1	ACTW
2	CDW
3	ACTW
4	ACDW
5	ACDTW
6	CDT

Mining Closed Frequent Itemsets

CHARM [8] is an efficient algorithm for mining closed itemsets. Define two itemsets X,Y of length k as belonging to the same *prefix equivalence class*, $[P]$, if they share the $k-1$ length prefix P, i.e., $X = Px$ and $Y = Py$, where $x,y \in I$. More formally, $[P] = \{Px_i \mid x_i \in I\}$, is the class of all itemsets sharing P as a common prefix. In CHARM there is no distinct candidate generation and support counting phase. Rather, counting is simultaneous with candidate generation. For a given prefix class, one performs intersections of the tidsets ofall pairs of itemsets in the class, and checks if the resulting tidsets have cardinality at least *minsup*. Each resulting frequent itemset generates a new class which will be expanded in the next step. That is, for a given class of itemsets with prefix P, $[P] = \{Px_1, Px_2, \ldots, Px_n\}$, one performs the intersection of Px_i with all Px_j with $j > i$ to obtain a new class $[Px_i] = [P']$ with elements $P'x_j$ provided the itemset Px_ix_j is frequent. The computation progresses recursively until no more frequent itemsets are produced. The initial invocation is with the class of frequent single items (the class $[\emptyset]$). All tidset intersections for pairs of class elements are computed. However in addition to checking for frequency, CHARM eliminates branches that cannot lead to closed sets, and grows closed itemsets using subset relationships among tidsets. There are four cases: if $\mathbf{t}(X_i) \subset \mathbf{t}(X_j)$ or if $\mathbf{t}(X_i) = \mathbf{t}(X_j)$, then replace every occurrence of X_i with $X_i \cup X_j$, since whenever X_i occurs X_j also occurs, which implies that $\mathbf{c}(X_i) \subseteq \mathbf{c}(X_i \cup X_j)$. If $\mathbf{t}(X_i) \supset \mathbf{t}(X_j)$ then replace X_j for the same reason. Finally, further recursion is required if $\mathbf{t}(X_i) \neq \mathbf{t}(X_j)$. These four properties allow CHARM to efficiently prune the search tree (for additional details see [10]).

Figure 2 shows how CHARM works on the example database shown in Table 1. First, CHARM sorts the items in increasing order of support, and initializes the root class as $[\emptyset] = \{D \times 2456, T \times 1356, A \times 1345, W \times 12345, C \times 123456\}$. The notation $D \times 2456$ stands for the itemset D and its tidset $\mathbf{t}(D) = \{2,4,5,6\}$. CHARM first processes the node $D \times 2456$; it will be combined with the sibling elements. DT and DA are not frequent and are thus pruned. Looking at W, since $\mathbf{t}(D) \neq \mathbf{t}(W)$, W is inserted in the new equivalence class $[D]$. For C, since $\mathbf{t}(D) \subset \mathbf{t}(C)$, all occurrences of D are replaced with DC, which means that $[D]$ is also changed to $[DC]$, and the element DW to DWC. A recursive call with class $[DC]$ is then made and since



Closed Itemset Mining and Non-redundant Association Rule Mining. Figure 2. CHARM: mining closed frequent itemsets.

there is only a single itemset DWC, it is added to the set of closed itemsets $\mathcal{C}$. When the call returns to D (i.e., DC) all elements in the class have been processed, so DC itself is added to $\mathcal{C}$.

When processing T, $\mathbf{t}(T) \neq \mathbf{t}(A)$, and thus CHARM inserts A in the new class $[T]$. Next it finds that $\mathbf{t}(T) \neq \mathbf{t}(W)$ and updates $[T] = \{A, W\}$. When it finds $\mathbf{t}(T) \subset \mathbf{t}(C)$ it updates all occurrences of T with TC. The class $[T]$ becomes $[TC] = \{A, W\}$. CHARM then makes a recursive call to process $[TC]$. When combining TAC with TWC it finds $\mathbf{t}(TAC) = \mathbf{t}(TWC)$, and thus replaces TAC with $TACW$, deleting TWC at the same time. Since $TACW$ cannot be extended further, it is inserted in $\mathcal{C}$. Finally, when it is done processing the branch TC, it too is added to $\mathcal{C}$. Since $\mathbf{t}(A) \subset \mathbf{t}(W) \subset \mathbf{t}(C)$ no new recursion is made; the final set of closed itemsets $\mathcal{C}$ consists of the uncrossed itemsets shown in Fig. 2.

Non-redundant Association Rules

Given the set of closed frequent itemsets $\mathcal{C}$, one can generate all non-redundant association rules. There are two main classes of rules: (i) those that have 100% confidence, and (ii) those that have less than 100% confidence [9]. Let X_1 and X_2 be closed frequent itemsets. The 100% confidence rules are equivalent to those directed from X_1 to X_2, where $X_2 \subseteq X_1$, i.e., from a superset to a subset (not necessarily proper subset). For example, the rule $C \Rightarrow W$ is equivalent to the rule between the closed itemsets $\mathbf{c}(W) \Rightarrow \mathbf{c}(C) \equiv CW \Rightarrow C$. Its support is $sup(CW) = 5/6$, and its confidence is $\frac{sup(CW)}{sup(W)} = 5/5 = 1$, i.e., 100%. The less than 100% confidence rules are equivalent to those from X_1 to X_2 where $X_1 \subset X_2$, i.e., from a subset to a proper superset. For example, the rule $W \Rightarrow T$ is equivalent to the rule $\mathbf{c}(W) \Rightarrow \mathbf{c}(W \cup T) \equiv CW \Rightarrow ACTW$. Its support is $sup(TW) = 3/6 = 0.5$, and its confidence is $\frac{sup(TW)}{sup(W)} = 3/5 = 0.6$ or 60%. More details on how to generate these non-redundant rules appears in [9].

Key Applications

Closed itemsets provide a loss-less representation of the set of all frequent itemsets; they allow one to determine not only the frequent sets but also their exact support. At the same time they can be orders of magnitude fewer. Likewise, the non-redundant rules provide a much smaller, and manageable, set of rules, from which all other rules can be derived. There are numerous applications of these methods, such as market basket analysis, web usage mining, gene expression pattern mining, and so on.

Future Directions

Closed itemset mining has inspired a lot of subsequent research in mining compressed representations or summaries of the set of frequent patterns; see [2] for a survey of these approaches. Mining compressed pattern bases remains an active area of study.

Experimental Results

A number of algorithms have been proposed to mine frequent closed itemsets, and to extract non-redundant rule bases. The Frequent Itemset Mining Implementations (FIMI) Repository contains links to many of the latest implementations for mining closed itemsets. A report on the comparison of these methods

also appears in [4]. Other implementations can be obtained from individual author's websites.

Data Sets

The FIMI repository has a number of real and synthetic datasets used in various studies on closed itemset mining.

Url to Code

The main FIMI website is at http://fimi.cs.helsinki.fi/, which is also mirrored at: http://www.cs.rpi.edu/~zaki/FIMI/

Cross-references

- ► Association Rule Mining on Streams
- ► Data Mining

Recommended Reading

1. Bastide Y., Pasquier N., Taouil R., Stumme G., and Lakhal L. Mining minimal non-redundant association rules using frequent closed itemsets. In Proc. 1st Int. Conf. Computational Logic, 2000, pp. 972–986.
2. Calders T., Rigotti C., and Boulicaut J.-F. A Survey on Condensed Representation for Frequent Sets. In Constraint-based Mining and Inductive Databases, LNCS, Vol. 3848, J-F. Boulicaut, L. De Raedt, and H. Mannila (eds.). Springer, 2005, pp. 64–80.
3. Ganter B. and Wille R. Formal Concept Analysis: Mathematical Foundations. Springer, Berlin Heidelberg New York, 1999.
4. Goethals B. and Zaki M.J. Advances in frequent itemset mining implementations: report on FIMI'03. SIGKDD Explor., 6(1): 109–117, June 2003.
5. Guigues J.L. and Duquenne V. Familles minimales d'implications informatives resultant d'un tableau de donnees binaires. Math. Sci. hum., 24(95):5–18, 1986.
6. Luxenburger M. Implications partielles dans un contexte. Math. Inf. Sci. hum., 29(113):35–55, 1991.
7. Pasquier N., Bastide Y., Taouil R., and Lakhal L. Discovering frequent closed itemsets for association rules. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 398–416.
8. Pei J., Han J., and Mao R. Closet: An efficient algorithm for mining frequent closed itemsets. In Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2000, pp. 21–30.
9. Zaki M.J. Generating non-redundant association rules. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 34–43.
10. Zaki M.J. and Hsiao C.-J. CHARM: An efficient algorithm for closed itemset mining. In Proc. SIAM International Conference on Data Mining, 2002, pp. 457–473.
11. Zaki M.J. and Ogihara M. Theoretical foundations of association rules. In Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 1998.

Closest Pairs

► Closest-Pair Query

Closest-Pair Query

Antonio Corral[1], Michael Vassilakopoulos[2]
[1]University of Almeria, Almeria, Spain
[2]University of Central Greece, Lamia, Greece

Synonyms

Closest pairs; k-Closest pair query; k-Distance join; Incremental k-distance join; k-Closest pair join

Definition

Given two sets P and Q of objects, a closest pair (CP) query discovers the pair of objects (p, q) with a distance that is the smallest among all object pairs in the Cartesian product P×Q. Similarly, a k closest pair query (k-CPQ) retrieves k pairs of objects from P and Q with the minimum distances among all the object pairs. In spatial databases, the distance is usually defined according to the Euclidean metric, and the set of objects P and Q are disk-resident. Query algorithms aim at minimizing the processing cost and the number of I/O operations, by using several optimization techniques for pruning the search space.

Historical Background

The closest pair query, has been widely studied in computational geometry. More recently, this problem has been approached in the context of spatial databases [4,8,12,14]. In spatial databases, existing algorithms assume that P and Q are indexed by a spatial access method (usually an R-tree [1]) and utilize some pruning bounds and heuristics to restrict the search space.

[8] was the first to address this issue, and proposed the following distance-based algorithms: incremental distance join, k distance join and k distance semijoin between two R-tree indices. The incremental processing reports one-by-one the desired elements of the result in ascending order of distance (k is unknown in advance and the user can stop when he/she is satisfied by the result). The algorithms follow the Best-First (BF) traversal policy, which keeps a heap with the entries of the nodes visited so far (it maintains a priority queue which contains pairs of index entries

and objects, and pop out the closest pair and process it). BF is near-optimal for CP queries; i.e., it only visits the pairs of nodes necessary for obtaining the result with a high probability. In [12] several modifications to the algorithms of [8] had been proposed in order to improve performance. Mainly, a method was proposed for selecting the sweep axis and direction for the plane sweep technique in bidirectional node expansion which minimizes the computational overhead of [8].

Later, an improved version of BF and several algorithms that follow Depth-First (DF) traversal ordering from the non-incremental point of view (which assumes that k is known in advance and reports the k elements of the result all together at the end of the algorithm) was proposed in [4]. In general, a DF algorithm visit the roots of the two R-trees and recursively follows the pair of entries $< E_P, E_Q >$, $E_P \in R_P$ and $E_Q \in R_Q$, whose MINMINDIST is the minimum distance among all pairs. At the opposite of BF, DF is sub-optimal, i.e., it accesses more nodes than necessary. The main disadvantage of BF with respect to DF is that it may suffer from buffer thrashing if the available memory is not enough for the heap (it is space-consuming), when a great quantity of elements of the result is required. In this case, part of the heap must be migrated to disk, incurring frequent I/O accesses. The implementation of DF is by recursion, which is available in most of the programming languages, and linear-space consuming with respect to the height of the R-trees. Moreover, BF is not favored by page replacement policies (e.g., LRU), as it does not exhibit locality between I/O accesses.

Another interesting contribution to the CP query was proposed by [14], in which a new structure called the b-Rdnn tree was presented, along with a better solution to the k-CP query when there is high overlap between the two datasets. The main idea is to find k objects from each dataset which are the closest to the other dataset.

There are a lot of papers related to k-CP query, like buffer query [3], iceberg distance join query [13], multi-way distance join query [6], k-nearest neighbor join [2], closest pair query with spatial constraints [11], etc. For example, a buffer query [3] involves two spatial datasets and a distance threshold ρ; the answer to this query is a set of pairs of spatial objects, one from each input dataset, that are within distance ρ of each other.

Foundations

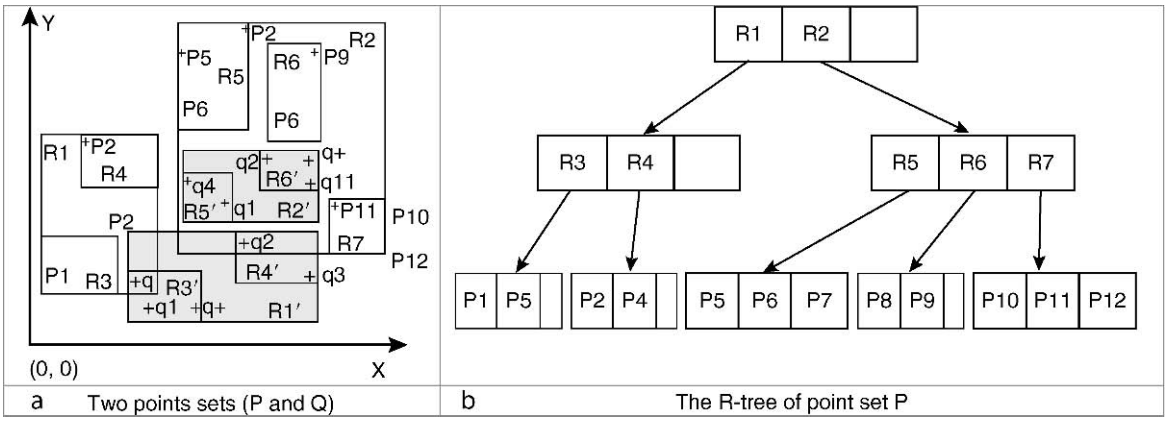
In spatial databases, existing algorithms assume that sets of spatial objects are indexed by a spatial access method (usually an R-tree [1]) and utilize some pruning bounds to restrict the search space. An R-tree is a hierarchical, height balanced multidimensional data structure, designed to be used in secondary storage based on B-trees. The R-trees are considered as excellent choices for indexing various kinds of spatial data (points, rectangles, line-segments, polygons, etc.). They are used for the dynamic organization of a set of spatial objects approximated by their Minimum Bounding Rectangles (MBRs). These MBRs are characterized by *min* and *max* points of rectangles with faces parallel to the coordinate axis. Using the MBR instead of the exact geometrical representation of the object, its representational complexity is reduced to two points where the most important features of the spatial object (position and extension) are maintained. The R-trees belong to the category of data-driven access methods, since their structure adapts itself to the MBRs distribution in the space (i.e., the partitioning adapts to the object distribution in the embedding space). Figure 1a shows two points sets P and Q (and the node extents), where the closest pair is (p_8, q_8), and Fig. 1b is the R-tree for the point set $P = \{p_1, p_2,...,p_{12}\}$ with a capacity of three entries per node (branching factor or fan-out).

Assuming that the spatial datasets are indexed on any spatial tree-like structure belonging to the R-tree family, then the main objective while answering these types of spatial queries is to reduce the search space. In [5], three MBR-based distance functions to be used in algorithms for CP queries were formally defined, as an extension of the work presented in [4]. These metrics are MINMINDIST, MINMAXDIST and MAXMAXDIST. MINMINDIST (M_1, M_2) between two MBRs is the minimum possible distance between any point in the first MBR and any point in the second MBR. Maxmaxdist between two MBRs (M_1, M_2) is the maximum possible distance between any point in the first MBR and any point in the second MBR. Finally, MINMAXDIST between two MBRs (M_1, M_2) is the minimum of the maximum distance values of all the pairs of orthogonal faces to each dimension. Formally, they are defined as follows:

Given two MBRs $M_1 = (a, b)$ and $M_2 = (c, d)$, in the d-dimensional Euclidean space,

$M_1 = (a, b)$, where $a = (a_1, a_2,...,a_d)$ and $b = (b_1, b_2,...,b_d)$ such that $a_i \leq b_i$ $1 \leq i \leq d$

$M_2 = (c, d)$, where $c = (c_1, c_2,...,c_d)$ and $d = (d_1, d_2,...,d_d)$ such that $a_i \leq b_i$ $1 \leq i \leq d$



Closest-Pair Query. Figure 1. Example of an R-tree and a point CP query.

the MBR-based distance functions are defined as follows:

$$MINMINDIST(M_1, M_2) = \sqrt{\sum_{i=1}^{d} \begin{cases} (c_i - b_i)^2, & c_i > b_i \\ (a_i - d_i)^2, & a_i > d_i \\ 0, & otherwise \end{cases}}$$

$$MAXMAXDIST(M_1, M_2) = \sqrt{\sum_{i=1}^{d} \begin{cases} (d_i - a_i)^2, & c_i > b_i \\ (b_i - c_i)^2, & a_i > d_i \\ \max\{(d_i - a_i)^2, (b_i - c_i)^2\}, & otherwise \end{cases}}.$$

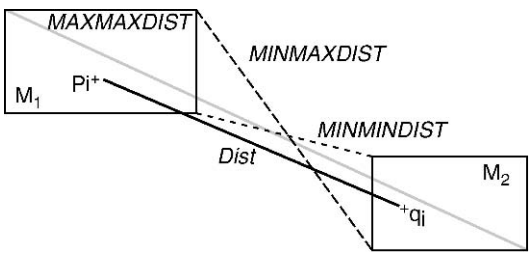
$$MINMAXDIST(M_1, M_2) = \sqrt{\min_{1 \le j \le d} \left\{ x_j^2 + \sum_{i=1, i \ne j}^{d} y_i^2 \right\}}$$

where

$$x_j = \min\{|a_j - c_j|, |a_j - d_j|, |b_j - c_j|, |b_j - d_j|\} \text{and}$$
$$y_i = \max\{|a_i - d_i|, |b_i - c_i|\}$$

To illustrate the distance functions MINMINDIST, MINMAXDIST and MAXMAXDIST which are the basis of query algorithms for CPQ, in Fig. 2, two MBRs and their MBR-based distance functions and their relation with the distance (dist) between two points (p_i, q_j) are depicted in 2-dimensional Euclidean space.

According to [5], MINMINDIST(M_1,M_2) is monotonically non-decreasing with the R-tree heights. MINMINDIST(M_1,M_2) and MAXMAXDIST(M_1, M_2) serve respectively as lower and upper bounding functions of the Euclidean distance from the k closest pairs of spatial objects within the MBRs M_1, and M_2. In the same sense, MINMAXDIST(M_1, M_2) serves as an upper bounding function of the Euclidean distance from the closest pair of spatial objects enclosed by the MBRs M_1 and M_2. As long as the distance functions are consistent, the branch-bound algorithms based on them will work correctly [5].



Closest-Pair Query. Figure 2. MBR-based distance functions in 2-dimensional Euclidean space.

Moreover, the general pruning mechanism for k-CP queries over R-tree nodes using branch-and-bound algorithms is the following: *if MINMINDIST (M_1, M_2) > z, then the pair of MBRs (M_1, M_2) will be discarded*, where z is the distance value of the k-th closest pair that has been found so far (during the processing of the algorithm), or the distance value of the k-th largest MAXMAXDIST found so far (z is also called as the pruning distance).

Branch-and-bound algorithms can be designed following DF or BF traversal ordering (Breadth-First traversal order (level-by-level) can also be implemented, but the processing of each level must follow a BF order) to report k closest pairs in non-incremental way (for incremental processing the ordering of traversal must be BF [8]).

As an example, Fig. 3 shows the BF k-CPQ algorithm for two R-trees, for the non-incremental processing version. This algorithm needs to keep a minimum binary heap (H) with the references to pairs of internal nodes (characterized by their MBRs) accessed so far from the two different R-trees and their minimum distance (<MINMINDIST, $Addr_{MPi}$, $Addr_{MQj}$>). It visits the pair of MBRs (nodes) with the minimum MINMINDIST in H, until it becomes empty or the MINMINDIST value of the pair of MBRs located in the root of H is larger than the distance value of the k-th closest pair that has been found so far (z). To keep track of z, an additional data structure that stores the k closest pairs discovered during the processing of the algorithm is needed. This data structure is organized as a maximum binary heap (k-heap) and holds pairs of objects according to their minimum distance (the pair with the largest distance resides in the root). In the implementation of k-CPQ algorithm, the following cases must be considered: (i) initially the k-heap is empty (z is initialized to ∞), (ii) the pairs of objects reached at the leaf level are inserted in the k-heap until it gets full (z keeps the value of ∞), (iii) if the distance of a new pair of objects discovered at the leaf level is smaller than the distance of the pair residing in the k-heap root, then the root is extracted and the new pair is inserted in the k-heap, updating this data structure and z (distance of the pair of objects residing in the k-heap root).

Several optimizations had been proposed in order to improve performance, mainly with respect to the CPU cost. For instance, a method for selecting the sweep axis and direction for the plane sweep technique has been proposed [12]. But the most important optimization is the use of the plane-sweep technique for k-CPQ [5,12], which is a common technique for computing intersections. The basic idea is to move a sweep-line perpendicular to one of the dimensions, so-called the sweeping dimension, from left to right. This technique is applied for restricting all possible combinations of pairs of MBRs from two R-tree nodes from R_P

```
Create and initialize H, k-heap, z = ∞;
for each pair of MBRs <MPi, MQj> in <rootRP, rootRQ> do
      H.insert(MINMINDIST(MPi, MQj), AddrMPi, AddrMQj);
enddo
While (not H.isEmpty()) and (H.MinimumDistance() ≤ z) do
      minimum = H.deleteMinimum();
      nodeRP = RP.readNode(minimum.AddrMP);
      nodeRQ = RQ.readNode(minimum.AddrMQ);
      if (nodeRP and nodeRQ are internal nodes) then
          for each pair of MBRs <MPi, MQj> in < nodeRP, nodeRQ> do
              if (MINMINDIST(MPi, MQj) ≤ z) then
                  H.insert(MINMINDIST(MPi, MQj), AddrMPi, AddrMQj);
              endif
          enddo
      else
          for each pair of points <pi, qj> in <nodeRP, nodeRQ> do
              distance = dist(pi, qj);
              if (k-heap.isFull()) then
                  if (distance ≤ z) then
                      k-heap.delete Maximum();
                      k-heap.insert(distance, pi, qj);
                      z = k-heap.get Maximum();
                  endif
              else
                  k-heap.insert(distance, pi, qj);
              endif
          enddo
      endif
endod
Destroy H;
```

Closest-Pair Query. Figure 3. Best-First k-CPQ Algorithm using R–trees.

and R_Q. If this technique is not used, then a set with all possible combinations of pairs of MBRs from two R-tree nodes must be created. In general, the technique consists in sorting the MBRs of the two current R-tree nodes, based on the coordinates of one of the lower left corners of the MBRs in increasing order. Each MBR encountered during a plane sweep is selected as a pivot, and it is paired up with the non-processed MBRs in the other R-tree node from left to right. The pairs of MBRs with MINMINDIST on the sweeping dimension that are less than or equal to z (pruning distance) are selected for processing. After all possible pairs of MBRs that contain the pivot have been found, the pivot is updated with the MBR of the next smallest value of a lower left corner of MBRs on the sweeping dimension, and the process is repeated. In summary, the application of this technique can be viewed as a *sliding window* on the sweeping dimension with a width of z starting in the lower end of the pivot MBR, where all possible pairs of MBRs that can be formed using the MBR of the pivot and the other MBRs from the remainder entries of the other R-tree node that fall into the current sliding window are chosen. For example, in Fig. 4, a set of MBRs from two R-tree nodes ($\{M_{P1}, M_{P2}, M_{P3}, M_{P4}, M_{P5}, M_{P6}\}$ and $\{M_{Q1}, M_{Q2}, M_{Q3}, M_{Q4}, M_{Q5}, M_{Q6}, M_{Q7}\}$) is shown. Without plane-sweep, 6*7 = 42 pairs of MBRs must be generated. If the plane-sweep technique is applied over the X axis (sweeping dimension) and taking into account the distance value of z (pruning distance), this number of possible pairs will reduced considerably (the number of selected pairs of MBRs using the plane sweep technique is only 29).

Key Applications

Geographical Information Systems

Closest pair is a common distance-based query in the spatial database context, and it has only recently received special attention. Efficient algorithms are important for dealing with the large amount of spatial data in several GIS applications. For example, k-CPQ can discover the K closest pairs of cities and cultural landmarks providing an increase order based on its distances.

Data Analysis

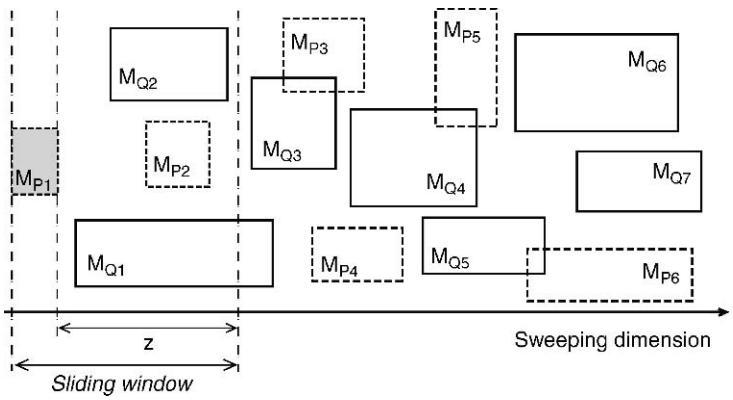
Closest pair queries have been considered as a core module of clustering. For example, a proposed clustering algorithm [10] owes its efficiency to the use of closest pair query, as opposed to previous quadratic-cost approaches.

Decision Making

A number of decision support tasks can be modeled as closest pairs query. For instance, find the top k factory-house pairs ordered by the closeness to one another. This gives us a measure of the effect of individual factory on individual household, and can give workers a priority to which factory to address first.

Future Directions

k-closest pair query is a useful type of query in many practical applications involving spatial data, and the traditional technique to handle this spatial query generally assumes that the objects are static. Objects represented as a function of time have been studied in other domains, as in spatial semijoin [9]. For this reason,



Closest-Pair Query. Figure 4. Using plane-sweep technique over MBRs from two R-tree nodes.

closest pair query in spatio-temporal databases could be an interesting line of research.

Another interesting problem to study is the monitoring of k-closest pairs over moving objects. It aims at maintaining closest pairs results while the underlying objects change the positions [15]. For example, return k pairs of taxi stands and taxies that have the smallest distances.

Other interesting topics to consider (from the static point of view) are to study k-CPQ between different spatial data structures (Linear Region Quadtrees for raster and R-trees for vector data), and to investigate k-CPQ in non-Euclidean spaces (e.g., road networks).

Experimental Results

In general, for every presented method, there is an accompanying experimental evaluation in the corresponding reference. [4,5,8] compare BF and DF traversal order for conventional k-CPQ (from the incremental and non-incremental point of view). In [7], a cost model for k-CPQ using R-trees was proposed, evaluating their accuracy. Moreover, experimental results on k-closest pair queries to support the fact that b-Rdnn tree is a better alternative with respect to the R*-trees, when there is high overlap between the two datasets, were presented in [14].

Data Sets

A large collection of real datasets, commonly used for experiments, can be found at: http://www.rtreeportal.org/

URL to Code

R-tree portal (see above) contains the code for most common spatial access methods (mainly R-tree and variations), as well as data generators and several useful links for researchers and practitioners in spatial databases.

The sources in C + + of k-CPQ are in: http://www.ual.es/~acorral/DescripcionTesis.htm

Cross-references

- ▶ Multi-Step Query Processing
- ▶ Nearest Neighbor Query
- ▶ R-Tree (and family)
- ▶ Spatial Indexing Techniques
- ▶ Spatial Join

Recommended Reading

1. Beckmann N., Kriegel H.P., Schneider R., and Seeger B. The R*-tree: an efficient and robust access method for points and rectangles. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1990, pp. 322–331.
2. Böhm C. and Krebs F. The k-nearest neighbour join: Turbo charging the KDD process. Knowl. Inform. Syst., 6(6):728–749, 2004.
3. Chan E.P.F. Buffer queries. IEEE Trans. Knowl. Data Eng., 15(4):895–910, 2003.
4. Corral A., Manolopoulos Y., Theodoridis Y., and Vassilakopoulos M. Closest pair queries in spatial databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 189–200.
5. Corral A., Manolopoulos Y., Theodoridis Y., and Vassilakopoulos M. Algorithms for processing K-closest-pair queries in spatial databases. Data Knowl. Eng., 49(1):67–104, 2004.
6. Corral A., Manolopoulos Y., Theodoridis Y., and Vassilakopoulos M. Multi-way distance join queries in spatial databases. GeoInformatica, 8(4):373–402, 2004.
7. Corral A., Manolopoulos Y., Theodoridis Y., and Vassilakopoulos M. Cost models for distance joins queries using R-trees. Data Knowl. Eng., 57(1):1–36, 2006.
8. Hjaltason G.R. and Samet H. Incremental distance join algorithms for spatial databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 237–248.
9. Iwerks G.S., Samet H., and Smith K. Maintenance of spatial semijoin queries on moving points. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 828–839.
10. Nanopoulos A., Theodoridis Y., and Manolopoulos Y. C^2P: clustering based on closest pairs. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 331–340.
11. Papadopoulos A.N., Nanopoulos A., and Manolopoulos Y. Processing distance join queries with constraints. Comput. J., 49(3):281–296, 2006.
12. Shin H., Moon B., and Lee S. Adaptive multi-stage distance join processing. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 343–354.
13. Shou Y., Mamoulis N., Cao H., Papadias D., and Cheung D.W. Evaluation of iceberg distance joins. In Proc. 8th Int. Symp. Advances in Spatial and Temporal Databases, 2003, pp. 270–288.
14. Yang C. and Lin K. An index structure for improving closest pairs and related join queries in spatial databases. In Proc. Int. Conf. on Database Eng. and Applications, 2002, pp. 140–149.
15. Zhu M., Lee D.L., and Zhang J. k-closest pair query monitoring over moving objects. In Proc. 3rd Int. Conf. on Mobile Data Management, 2002, pp. 14–14.

Cloud Computing

- ▶ Replication in Multi-Tier Architectures
- ▶ Storage Grid

Cluster and Distance Measure

Dimitrios Gunopulos[1,2]
[1]Computer Science and Eng. Dept., Univ. of California Riverside, Riverside, CA 92521, USA
[2]Dept. of Informatics and Telecommunications, University of Athens, Athens, Greece

Synonyms

Unsupervised learning; Segmentation

Definition

Clustering

Clustering is the assignment of objects to groups of similar objects (clusters). The objects are typically described as vectors of features (also called attributes). So if one has n attributes, object $\mathbf{x}$ is described as a vector $(x_1,..,x_n)$. Attributes can be numerical (scalar) or categorical. The assignment can be hard, where each object belongs to one cluster, or fuzzy, where an object can belong to several clusters with a probability. The clusters can be overlapping, though typically they are disjoint. Fundamental in the clustering process is the use of a distance measure.

Distance Measure

In the clustering setting, a distance (or equivalently a similarity) measure is a function that quantifies the similarity between two objects.

Key Points

The choice of a distance measure depends on the nature of the data, and the expected outcome of the clustering process. The most important consideration is the type of the features of the objects. One first focuses on distance measures when the features are all numerical. This includes features with continuous values (real numbers) or discrete values (integers). In this case, typical choices include:

1. The L_p norm. It is defined as $D(\mathbf{x},\mathbf{y}) = \left(\sum_{1\leq i\leq n} (X_1 - Y_1)^p\right)^{1/p}$. Typically p is 2 (the intuitive and therefore widely used Euclidean distance), or 1 (the Manhattan or city block distance), or infinity (the Maximum distance).
2. The Mahalanobis distance. It is defined as $D(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})\sum^{-1}(\mathbf{x}-\mathbf{y})^T$ which generalizes the Euclidean and allows the assignment of different weights to different features.
3. The angle between two vectors, computed using the inner product of two vectors $\mathbf{x}\cdot\mathbf{y}$.
4. The Hamming distance, which measures the number of disagreements between two binary vectors.

In different settings different distance measures can be used. The edit, or Levenshtein, distance, is an extension of the Hamming distance, and is typically used for measuring the distance between two strings of characters. The edit distance is defined as the minimum number of insertions, deletions or substitutions that it takes to transform one sting to another.

When two time series are compared, the Dynamic Time Warping distance measure is often used to quantify their distance. The length of the Longest Common Subsequence (LCSS) of two time series is also frequently used to provide a similarity measure between the time series. LCSS is a similarity measure because the longest common subsequence becomes longer when two time series are more similar. To create a distance measure, LCSS is typically normalized by dividing by the length of the longest of the two sequences, and then subtracting the ratio from one.

Finally, when sets of objects are compared, the Jaccard coefficient is typically used to compute their distance. The Jaccard coefficient of sets A and B is defined as $J(A,B) = |A \cap B|/|A \cup B|$, that is, the fraction of the common elements over the union of the two sets.

The majority of the distance measures used in practice, and indeed most of the ones described above are metrics. Formally, a distance measure D is a metric if it obeys the following properties:

For objects A, B, (i) D(A,B)\geq 0, (ii) D(A,B) = 0 if and only if A = B, and (iii) D(A,B) = D(B,A), and (iv) for any objects A,B,C, D(A,B) + D(B,C)\geq D(A,C) (triangle inequality).

Most distance measures can be trivially shown to observe the first three properties, but do not necessarily observe the triangle inequality. For example, the constrained Dynamic Time Warping distance, a typically used measure to compute the similarity between time series which does not allow arbitrary stretching of a time series, is not a metric because it does not satisfy the triangle inequality. Experimental results have shown that the constrained Dynamic Time Warping distance performs at least as good as the unconstrained one and it is also faster to compute, thus justifying its use although it is not a metric. Note however that, if it is so required, any distance measure can be converted into a metric by

taking the shortest path between objects A and B in the complete graph where each object is a node and each edge is weighted by the distance between the two nodes.

Cross-references

- ▶ Clustering Overview and Applications
- ▶ Data Mining

Recommended Reading

1. Everitt B.S., Landau S., Leese M. Cluster Analysis. Wiley, 2001.
2. Jain A.K., Murty M.N., and Flyn P.J. Data Clustering: A Review. ACM Comput Surv, 31(3):1999.
3. Theodoridis S. and Koutroubas K. Pattern recognition. Academic Press, 1999.

Cluster Database Replication

- ▶ Replica Control

Cluster Databases

- ▶ Process Structure of a DBMS

Cluster Replication

- ▶ Replication for Scalability
- ▶ Replication in Multi-Tier Architectures

Cluster Stability

- ▶ Clustering Validity

Cluster Validation

- ▶ Clustering Validity

Clustering

- ▶ Deduplication in Data Cleaning
- ▶ Physical Database Design for Relational Databases

Clustering for Post Hoc Information Retrieval

Dietmar Wolfram
University of Wisconsin-Milwaukee, Milwaukee, WI, USA

Synonyms

Document clustering

Definition

Clustering is a technique that allows similar objects to be grouped together based on common attributes. It has been used in information retrieval for different retrieval process tasks and objects of interest (e.g., documents, authors, index terms). Attributes used for clustering may include assigned terms within documents and their co-occurrences, the documents themselves if the focus is on index terms, or linkages (e.g., hypertext links of Web documents, citations or co-citations within documents, documents accessed). Clustering in IR facilitates browsing and assessment of retrieved documents for relevance and may reveal unexpected relationships among the clustered objects.

Historical Background

A fundamental challenge of information retrieval (IR) that continues today is how to best match user queries with documents in a queried collection. Many mathematical models have been developed over the years to facilitate the matching process. The details of the matching process are usually hidden from the user, who is only presented with an outcome. Once a set of candidate documents has been identified, they are presented to the user for perusal. Traditional approaches have relied on ordered linear lists of documents based on calculated relevance or another sequencing criterion (e.g., date, alphabetical by title or author). The resulting linear list addresses the assessed relationship of documents to queries, but not the relationships of the documents themselves. Clustering techniques can reduce this limitation by creating groups of documents (or other objects of interest) to facilitate more efficient retrieval or perusal and evaluation of retrieved sets.

The application of clustering techniques to IR extends back to some of the earliest experimental IR systems including Gerard Salton's SMART system, which relied on document cluster identification within

a vector space as a means of quickly identifying sets of relevant documents. The rationale for applying clustering was formalized as the "cluster hypothesis," proposed by Jardine and van Rijsbergen [6]. This hypothesis proposes that documents that are relevant to a query are more similar to each other than to documents that are not relevant to the query. The manifestation of this relationship can be represented in different ways by grouping like documents or, more recently, visualizing the relationships and resulting proximities in a multidimensional space.

Early applications of clustering emphasized its use to more efficiently identify groups of related, relevant documents and to improve search techniques. The computational burden associated with real-time cluster identification during searches on increasingly larger data corpora and the resulting lackluster performance improvements have caused clustering to lose favor as a primary mechanism for retrieval. However, clustering methods continue to be studied and used today (see, for example, [7]). Much of the recent research into clustering for information retrieval has focused on other areas that support the retrieval process. For instance, clustering has been used to assist in query expansion, where additional terms for retrieval may be identified. Clustering of similar terms can be used to construct thesauri, which can be used to index documents [3].

Recent research on clustering has highlighted its benefits for post hoc retrieval tasks, in particular for the presentation of search results to better model user and usage behavior. The focus of applications presented here is on these post hoc IR tasks, dealing with effective representation of groups of objects once identified to support exploratory browsing and to provide a greater understanding of users and system usage for future IR system development.

Foundations

Methods used to identify clusters are based on cluster analysis, a multivariate exploratory statistical technique. Cluster analysis relies on similarities or differences in object attributes and their values. The granularity of the analysis and the validity of the resulting groups are dependent on the range of attributes and values associated with objects of interest. For IR applications, clusters are based on common occurrences and weights of assigned terms for documents, the use of query terms, or linkages between objects of interest represented as hypertext linkages or citations/co-citations.

Clustering techniques can be divided into hierarchical and non-hierarchical approaches. Non-hierarchical clustering methods require that a priori assumptions be made about the nature and number of clusters, but can be useful if specific cluster parameters are sought. Hierarchical clustering, which is more commonly used, begins with many small groups of objects that serve as initial clusters. Existing groups are clustered into larger groups until only one cluster remains. Visually, the structure and relationship of clusters may be represented as a dendrogram, with different cluster agglomerations at different levels on the dendrogram representing the strength of relationship between clusters. Other visualization techniques may be applied and are covered elsewhere. In hierarchical methods, the shorter the agglomerative distance, the closer the relationship and the more similar the clusters are. As an exploratory technique, there is no universally accepted algorithm to conduct the analysis, but the general steps for conducting the analysis are similar. First, a similarity measure is applied to the object attributes, which serves as the basis for pairwise comparisons. Standard similarity or distance measures applied in IR research such as the Euclidean distance, cosine measure, Jaccard coefficient, and Dice coefficient can be used. Next, a method for cluster determination is selected. Common methods include: single complete linkage, average linkage, nearest neighbor, furthest neighbor, centroid clustering (representing the average characteristics of objects within a cluster), and Ward's method. Each method uses a different algorithm to assess cluster membership and may be found to be more appropriate in given circumstances. Outcomes can vary significantly depending on the method used. This flexibility underscores one of the challenges for effectively implementing cluster analysis. With no one correct or accepted way to conduct the analysis, outcomes are open to interpretation, but may be viewed as equally valid. For example, single linkage clustering, which links pairs of objects that most closely resemble one another, is comparatively simple to implement and has been widely used, but can result in lengthy linear chains of clusters. Parameters may need to be specified that dictate the minimum size of clusters to avoid situations where there are large orders of difference in cluster membership. Another challenge inherent in clustering is that different clustering algorithms can produce similar

numbers of clusters, but if some clusters contain few members, this does little to disambiguate the members within large clusters. The number of clusters that partition the object set can be variable in hierarchical clustering. More clusters result in fewer objects per cluster with greater inter-object similarity, but with potentially more groups to assess. It is possible to test for an optimal number of clusters using various measures that calculate how differing numbers of clusters affect cluster cohesiveness.

Clustering may be implemented in dynamic environments by referencing routines based on specific clustering algorithms developed by researchers or through specialty clustering packages. Details on clustering algorithms for information retrieval can be found in Rasmussen [8]. Standard statistical and mathematical software packages such as SAS and SPSS also support a range of clustering algorithms. Special algorithms may need to be applied to very large datasets to reduce computational overhead, which can be substantial for some algorithms.

Key Applications

In addition to early applications of clustering for improving retrieval efficiency, clustering techniques in IR have included retrieval results presentation, and modeling of IR user and usage characteristics based on transactions logs. Although largely a topic of research interest, some applications have found their way into commercial systems. Clustering of search results has been applied by several Web-based search services since the late 1990s, some of which are no longer available. Most notable of the current generation is Clusty (clusty.com), which organizes retrieval results from several search services around topical themes.

The application of clustering to support interactive browsing has been an active area of investigation in recent years. Among the earliest demonstrations for this purpose was the Scatter/Gather method outlined by Cutting et al. [4], in which the authors demonstrated how clustering of retrieved items can facilitate browsing for vaguely defined information needs. This approach was developed to serve as a complement to more focused techniques for retrieval assessment. In application, the method presents users with a set of clusters that serves as the starting point for browsing. The user selects the clusters of greatest interest. The contents of those clusters are then gathered into a single cluster, which now serves as the corpus for a new round of clustering, into which the new smaller corpus of items is scattered. The process continues until the user's information need is met or the user abandons the search. To support real time clustering of datasets, the authors developed an efficient clustering algorithm, called buckshot, plus a more accurate algorithm, called fractionation, to permit more detailed clustering in offline environments where a timely response is less critical. Another algorithm, called cluster digest, was used to encapsulate the topicality of a given cluster based on the highest weighted terms within the cluster. Hearst and Pedersen [5] evaluated the efficacy of Scatter/Gather on the top-ranked retrieval outcomes of a large dataset, and tested the validity of the cluster hypothesis. The authors compared the number of known relevant items to those appearing in the generated clusters. A user study was also conducted, which demonstrated that participants were able to effectively navigate and interact with the system incorporating Scatter/Gather.

Increasingly, IR systems provide access to heterogeneous collections of documents. The question arises whether the cluster hypothesis, and the benefits of capitalizing on its attributes, extends to the distributed IR environment, where additional challenges include the merger of different representations of documents and identification of multiple occurrences of documents across the federated datasets. Crestani and Wu [2] conducted an experimental study to determine whether the cluster hypothesis holds in a distributed environment. They simulated a distributed environment by using different combinations of retrieval environments and document representation heterogeneity, with the most sophisticated implementation representing three different IR environments with three different collections. Results of the different collections and systems were clustered and compared. The authors concluded that the cluster hypothesis largely holds true in distributed environments, but fails when brief surrogates of full text documents are used.

With the growing availability of large IR system transaction logs, clustering methods have been used to identify user and usage patterns. By better understanding patterns in usage behavior, IR systems may be able to identify types of behaviors and accommodate those behaviors through context-sensitive assistance or through integration of system features that accommodate identified behaviors. Chen and Cooper [1] relied on a rich dataset of user sessions collected from the University of California MELVYL online public access

catalog system. Based on 47 variables associated with each user session (e.g., session length in seconds, average number of items retrieved, average number of search modifications), their analysis identified six clusters representing different types of user behaviors during search sessions. These included help-intensive searching, knowledgeable usage, and known-item searching. Similarly, Wen et al. [9] focused on clustering of user queries in an online encyclopedia environment to determine whether queries could be effectively clustered to direct users to appropriate frequently asked questions topics. IR environments that cater to a broad range of users are well-known for short query submissions by users, which make clustering applications based solely on query term co-occurrence unreliable. In addition to the query content, the authors based their analysis on common retrieved documents viewed by users. By combining query content with common document selections, a link was established between queries that might not share search terms. The authors demonstrated how the application of their clustering method, which was reportedly adopted by the encyclopedia studied, could effectively guide users to appropriate frequently asked questions.

The previous examples represent only a sample of clustering applications in an IR context. Additional recent research developments and applications using clustering may be found in Wu et al. [10].

Cross-references

- ▶ Data Mining
- ▶ Text Mining
- ▶ Visualization for Information Retrieval

Recommended Reading

1. Chen H.M. and Cooper M.D. Using clustering techniques to detect usage patterns in a web-based information system. J. Am. Soc. Inf. Sci. Technol., 52(11):888–904, 2001.
2. Crestani F. and Wu S. Testing the cluster hypothesis in distributed information retrieval. Inf. Process. Manage., 42(5):1137–1150, 2006.
3. Crouch C.J. A cluster-based approach to thesaurus construction. In Proc. 11th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1988, pp. 309–320.
4. Cutting D.R., Karger D.R., Pedersen J.O., and Tukey J.W. Scatter/Gather: a cluster-based approach to browsing large document collections. In Proc. 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1992, pp. 318–329.
5. Hearst M.A. and Pedersen J.O. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, pp. 76–84.
6. Jardine N. and van Rijsbergen C. The use of hierarchic clustering in information retrieval. Inf. Storage Retr., 7(5):217–240, 1971.
7. Liu X. and Croft W.B. Cluster-based retrieval using language models. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 186–193.
8. Rasmussen E. Clustering algorithms. In Information Retrieval Data Structures & Algorithms, W.B. Frakes, R. Baeza-Yates (eds.). Prentice Hall, Englewood Cliffs, NJ, 1992, pp. 419–442.
9. Wen J.R., Nie J.Y., and Zhang H.J. Query clustering using user logs. ACM Trans. Inf. Syst., 20(1):59–81, 2002.
10. Wu W., Xiong H., and Shekhar S. (eds.) Clustering and Information Retrieval. Kluwer, Norwell, MA, 2004.

Clustering Index

▶ Primary Index

Clustering on Streams

Suresh Venkatasubramanian
University of Utah, Salt Lake City, UT, USA

Definition

An instance of a clustering problem (see clustering) consists of a collection of points in a distance space, a measure of the *cost* of a clustering, and a measure of the *size* of a clustering. The goal is to compute a partitioning of the points into clusters such that the cost of this clustering is minimized, while the size is kept under some predefined threshold. Less commonly, a threshold for the cost is specified, while the goal is to minimize the size of the clustering.

A data stream (see data streams) is a sequence of data presented to an algorithm one item at a time. A stream algorithm, upon reading an item, must perform some action based on this item and the contents of its working space, which is sublinear in the size of the data sequence. After this action is performed (which might include copying the item to its working space), the item is discarded.

Clustering on streams refers to the problem of clustering a data set presented as a data stream.

Historical Background

Clustering (see clustering) algorithms typically require access to the entire data to produce an effective clustering. This is a problem for large data sets, where

random access to the data, or repeated access to the entire data set, is a costly operation. For example, the well-known k-means heuristic is an iterative procedure that in each iteration must read the entire data set twice. One set of approaches to performing clustering on large data involves sampling: a small subset of data is extracted from the input and clustered, and then this clustering is extrapolated to the entire data set.

The *data stream* paradigm [14] came about in two ways: first, as a way to model access to large streaming sources (network traffic, satellite imagery) that by virtue of their sheer volume, cannot be archived for off-line processing and need to be aggregated, summarized and then discarded in real time. Second, the streaming paradigm has shown itself to be the most effective way of accessing large databases: Google's Map Reduce [9] computational framework is one example of the efficacy of stream processing.

Designing clustering algorithms for stream data requires different algorithmic ideas than those useful for traditional clustering algorithms. The online computational paradigm [4] is a potential solution: in this paradigm, an algorithm is presented with items one by one, and using only information learned up to the current time, must make a prediction or estimate on the new item being presented. Although the online computing paradigm captures the sequential aspect of stream processing, it does not capture the additional constraint that only a small portion of the history may be stored. In fact, an online algorithm is permitted to use the entirety of the history of the stream, and is usually not limited computationally in any way. Thus, new ideas are needed to perform clustering in a stream setting.

Foundations

Preliminaries

Let X be a domain and d be a distance function defined between pairs of elements in X. Typically, it is assumed that d is a metric (i.e., it satisfies the triangle inequality $d(x, y) + d(y, z) \geq d(x, z) \forall x,y,z \in X$). One of the more common measures of the cost of a cluster is the so-called *median cost*: the cost of a cluster $C \subseteq X$ is the function

$$\text{cost}(C) = \sum_{x \in C} d(x, c^*)$$

where $c^* \in X$, the *cluster center*, is the point that minimizes cost(C). The *k-median problem* is to find a collection of k disjoint clusters, the sum of whose costs is minimized.

An equally important cost function is the *mean cost*: the cost of a cluster $C \subseteq X$ is the function

$$\text{cost}(C) = \sum_{x \in C} d^2(x, c^*)$$

where c^* is defined as before. The *k-means problem* is to find a collection of clusters whose total *mean cost* is minimized. It is useful to note that the median cost is more robust to outliers in the data; however, the mean cost function, especially for points in Euclidean spaces, yields a very simple definition for c^*: it is merely the centroid of the set of points in the cluster. Other measures that are often considered are the k-center cost, where the goal is to minimize the maximum radius of a cluster, and the diameter cost, where the goal is to minimize the maximum diameter of a cluster (note that the diameter measure does not require one to define a cluster center).

A data stream problem consists of a sequence of items $x_1, x_2,...,x_n$, and a function $f(x_1,...,x_n)$ that one wishes to compute. The limitation here is that the algorithm is only permitted to store a *sublinear* number of items in memory, because n is typically too large for all the items to fit in memory. Further, even random access to the data is prohibitive, and so the algorithm is limited to accessing the data in sequential order.

Since most standard clustering problems (including the ones described above) are NP-hard in general, one cannot expect solutions that minimize the cost of a clustering. However, one can often show that an algorithm comes close to being optimal: formally, one can show that the cost achieved by an algorithm is within some multiplicative factor c of the optimal solution. Such an algorithm is said to be a c-approximation algorithm. Many of the methods presented here will provide such guarantees on the quality of their output. As usual, one should keep in mind that these guarantees are *worst-case*, and thus apply to any possible input the algorithm may encounter. In practice, these algorithms will often perform far better than promised.

General Principles

Stream clustering is a relatively new topic within the larger area of stream algorithms and data analysis. However, there are some general techniques that have proven their usefulness both theoretically as well as practically, and are good starting points for the design

and analysis of stream clustering methods. This section reviews these ideas, as well as pointing to examples of how they have been used in various settings.

Incremental Clustering The simplest way to think about a clustering algorithm on stream data is to imagine the stream data arriving in chunks of elements. Prior to the arrival of the current chunk, the clustering algorithm has computed a set of clusters for all the data seen so far. Upon encountering the new chunk, the algorithm must update the clusters, possibly expanding some and contracting others, merging some clusters and splitting others. It then requests the next chunk, discarding the current one. Thus, a core component of any stream clustering algorithm is a routine to incrementally update a clustering when new data arrives. Such an approach was developed by Charikar et al. [6] for maintaining clusterings of data in a metric space using a diameter cost function. Although their scheme was phrased in terms of incremental clusterings, rather than stream clusterings, their approach generalizes well to streams. They show that their scheme yields a provable approximation to the optimal diameter of a k-clustering.

Representations One of the problems with clustering data streams is choosing a representation for a cluster. At the very least, any stream clustering algorithm stores the location of a cluster center, and possibly the number of items currently associated with this cluster. This representation can be viewed as a *weighted point*, and can be treated as a single point in further iterations of the clustering process. However, this representation loses information about the geometric size and distribution of a cluster. Thus, another standard representation of a cluster consists of the center and the number of points augmented with the sum of squared distances from the points in the cluster to the center. This last term informally measures the variation of points within a cluster, and when viewed in the context of density estimation via Gaussians, is in fact the sample variance of the cluster.

Clusters reduced in this way can be treated as weighted points (or weighted balls), and clustering algorithms should be able to handle such generalized points. One notable example of the use of such a representation is the one-pass clustering algorithm of Bradley et al. [5], which was simplified and improved by Farnstrom et al. [11]. Built around the well known k-means algorithm (that iteratively seeks to minimize the k-means measure described above), this technique proceeds as follows.

Algorithm 1: Clustering with representations

Initialize cluster centers randomly
While chunk of data remains to be read **do**
  Read a chunk of data (as much as will fit in memory), and cluster it using the k-means algorithm.
  For each cluster, divide the points contained within it into the core (points that are very close to the center under various measures), and the periphery.
  Replace the set of points in the core by a summary as described above. Discard all remaining points.
  Use the current cluster list as the set of centers for the next chunk.

It is important that representations be *linear*. Specifically, given two chunks of data c,c', and their representations r,r', it should be the case that the representation of $c \cup c'$ be formed from a linear combination of r and r'. This relates to the idea of *sketching* in stream algorithms, and is important because it allows the clustering algorithm to work in the (reduced) space of representations, rather than in the original space of data. Representations like the one described above are linear, and this is a crucial factor in the effectiveness of these algorithms.

Hierarchical Clustering Viewing a cluster as a weighted point in a new clustering problem quickly leads to the idea of *hierarchical clustering*: by thinking of a point as a single-element cluster, and connecting a cluster and its elements in a parent-child relationship, a clustering algorithm can represent multiple levels of merges as a tree of clusters, with the root node being a single cluster containing all the data, and each leaf being a single item. Such a tree is called a Hierarchical Agglomerative Clustering (HAC), since it can be viewed bottom-up as a series of agglomerations. Building such a hierarchy yields more general information about the relationship between clusters, and the ability to make better judgments about how to merge clusters.

The well-known clustering algorithm BIRCH [15] makes use of a hierarchy of cluster representations to cluster a large database in a few passes. In a first pass, a tree called the CF-tree is constructed, where each internal node represents a cluster of clusters, and each leaf represents a cluster of items. This tree is controlled by two parameters: B, the branching factor, and T, a diameter threshold that limits the size of leaf clusters.

In further passes, more analysis is performed on the CF-tree to compress clusters further. The tree is built much in the way a B+-tree is built: new items are inserted in the deepest cluster possible, and if the threshold constraint is violated, the cluster is split, and updates are propagated up the tree.

BIRCH is one of the best-known large-data clustering algorithms, and is generally viewed as a benchmark to compare other clustering algorithms against. However, BIRCH does not provide formal guarantees on the quality of the clusterings thus produced. The first algorithm that computes a hierarchical clustering on a stream while providing formal performance guarantees is a method for solving the k-median problem developed by Guha et al. [12,13]. This algorithm is best described by first presenting it in a non-streaming context:

Algorithm 2: Small space

Divide the input into l disjoint parts.
Cluster each part into $O(k)$ clusters. Assign each point to its nearest cluster center.
cluster the $O(lk)$ cluster centers, where each center is weighted by the number of points assigned to it.

Note that the total space required by this algorithm is $O(\ell k + n/\ell)$. The value of this algorithm is that it propagates good clusterings: specifically, if the intermediate clusterings are computed by algorithms that yield constant-factor approximations to the best clustering (under the k-median cost measure), then the final output will also be a (larger) constant factor approximation to the best clustering. Also note that the final clustering step may itself be replaced by a recursive call to the algorithm, yielding a hierarchical scheme.

Converting this to a stream algorithm is not too difficult. Consider each chunk of data as one of the disjoint parts the input is broken into. Suppose each part is of size m, and there exists a clustering procedure that can cluster these points into $2k$ centers with reasonable accuracy. The algorithm reads enough data to obtain m centers ($m^2/2k$ points). These m "points" can be viewed as the input to a second level streaming process, which performs the same operations. In general, the ith-level stream process takes $m^2/2k$ points from the $(i-1)$th-level stream process and clusters them into m points, which are appended to the stream for the next level.

The guarantees provided by the method rely on having accurate clustering algorithms for the intermediate steps. However, the general paradigm itself is useful as a heuristic: the authors show that using the k-means algorithm as the intermediate clustering step yields reasonable clustering results in practice, even though the method comes with no formal guarantees.

On Relaxing the Number of Clusters If one wishes to obtain guarantees on the quality of a clustering, using at least k clusters is critical; it is easy to design examples where the cost of a $(k-1)$-clustering is much larger than the cost of a k-clustering. One interesting aspect of the above scheme is how it uses weaker clustering algorithms (that output $O(k)$ rather than k clusters) as intermediate steps on the way to computing a k-clustering. In fact, this idea has been shown to be useful in a formal sense: subsequent work by Charikar et al. [7] showed that if one were to use an extremely weak clustering algorithm (in fact, one that produces $O(k \log n)$ clusters), then this output can be fed into a clustering algorithm that produces k clusters, while maintaining overall quality bounds that are better than those described above. This idea is useful especially if one has a fast algorithm that produces a larger number of clusters, and a more expensive algorithm that produces k clusters: the expensive algorithm can be run on the (small) output of the fast algorithm to produce the desired answer.

Clustering Evolving Data

Stream data is often temporal. Typical data analysis questions are therefore often limited to ranges of time ("in the last three days," "over the past week," "for the period between Jan 1 and Feb 1," and so on). All of the above methods for clustering streams assume that the goal is to cluster the entire data stream, and the only constraint is the space needed to store the data. Although they are almost always incremental, in that the stream can be stopped at any time and the resulting clustering will be accurate *for all data seen upto that point*, they cannot correctly output clusterings on *windows* of data, or allow the influence of past data to gradually wane over time. Even with non-temporal data, it may be important to allow the data analysis to operate on a subset of the data to capture the notion of *concept drift* [10], a term that is used to describe a scenario when natural data characteristics change as the stream evolves.

Sliding Windows A popular model of stream analysis is the *sliding window* model, which introduces a new parameter W. The goal of the stream analysis is to produce summary statistics (a clustering, variance estimates or other statistics), on the *most recent W items only*, while using space that is sublinear in W. This model can be thought of as represented by a *sliding window* of length W with one end (the sliding end) anchored to the current element being read. The challenge of dealing with sliding windows is the problem of deletion. Although not as general as a fully dynamic data model where arbitrary elements can be inserted and deleted, the sliding window model introduces with the problem of updating a cluster representation under deletions, and requires new ideas.

One such idea is the *exponential histogram*, first introduced by Datar et al. [8] to estimate certain statistical properties of sliding windows on streams, and used by Babcock et al. [3] to compute an approximate k-median clustering in the sliding window model. The idea here is to maintain a set of buckets that together partition all data in the current window. For each bucket, relevant summary statistics are maintained. Intuitively, the smaller the number of items assigned to a bucket, the more accurate the summary statistics (in the limit, the trivial histogram has one bucket for each of the W items in the window). The larger this number, the fewer the number of buckets needed. Balancing these two conflicting requirements yields a scheme where each bucket stores the items between two timestamps, and the bucket sizes increase exponentially as they store items further in the past. It requires more detailed analysis to demonstrate that such a scheme will provide accurate answers to queries over windows, but the use of such exponentially increasing bucket sizes allows the algorithm to use a few buckets, while still maintaining a reasonable approximation to the desired estimate.

Hierarchies of Windows The sliding window model introduces an extra parameter W whose value must be justified by external considerations. One way of getting around this problem is to maintain statistics for multiple values of W (typically an exponentially increasing family). Another approach, used by Aggarwal et al. [1] is to maintain *snapshots* (summary representations of the clusterings) at time steps at different levels of resolution. For example, a simple two level snapshot scheme might store the cluster representations computed after times t, $t+1$,...$t+W$, as well as t, $t+2$, $t+4$,...$t+2W$ (eliminating duplicate summaries as necessary). Using the linear structure of representations will allow the algorithm to extract summaries for time intervals: they show that such a scheme uses space efficiently while still being able to detect evolution in data streams at different scales.

Decaying Data For scenarios where such a justification might be elusive, another model of evolving data is the *decay model*, in which one can think of a data item's influence waning (typically exponentially) with time. In other words, the value of the ith item, instead of being fixed at x_i, is a function of time $x_i(t) = x_i(0)\exp(-c(t-i))$. This reduces the problem to the standard setting of computing statistics over the entire stream, while using the decay function to decide which items to remove from the limited local storage when computing statistics. The use of exponentially decaying data is quite common in temporal data analysis: one specific example of its application in the clustering of data streams is the work on HPStream by Aggarwal et al. [2].

Key Applications

Systems that manage large data sets and perform data analysis will require stream clustering methods. Many modern *data cleaning systems* require such tools, as well as large scientific databases. Another application of stream clustering is for network traffic analysis: such algorithms might be situated at routers, operating on packet streams.

Experimental Results

Most of the papers cited above are accompanied by experimental evaluations and comparisons to prior work. BIRCH, as mentioned before, is a common benchmarking tool.

Cross-references

▶ Data Clustering
▶ Information Retrieval
▶ Visualization

Recommended Reading

1. Aggarwal C.C., Han J., Wang J., and Yu P.S. A framework for clustering evolving data streams. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 81–92.
2. Aggarwal C.C., Han J., Wang J., and Yu P.S. A framework for projected clustering of high dimensional data streams. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 852–863.

3. Babcock B., Datar M., Motwani R., and O'Callaghan L. Maintaining variance and k-medians over data stream windows. In Proc. 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2003, pp. 234–243.
4. Borodin A. and El-Yaniv R. Online computation and competitive analysis. Cambridge University Press, New York, NY, USA, 1998.
5. Bradley P.S., Fayyad U.M., and Reina C. Scaling Clustering Algorithms to Large Databases. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, 1998, pp. 9–15.
6. Charikar M., Chekuri C., Feder T., and Motwani R. Incremental Clustering and Dynamic Information Retrieval. SIAM J. Comput., 33(6):1417–1440, 2004.
7. Charikar M., O'Callaghan L., and Panigrahy R. Better streaming algorithms for clustering problems. In Proc. 35th Annual ACM Symp. on Theory of Computing, 2003, pp. 30–39.
8. Datar M., Gionis A., Indyk P., and Motwani R. Maintaining stream statistics over sliding windows: (extended abstract). In Proc. 13th Annual ACM -SIAM Symp. on Discrete Algorithms, 2002, pp. 635–644.
9. Dean J. and Ghemaway S. MapReduce: simplified data processing on large clusters. In Proc. 6th USENIX Symp. on Operating System Design and Implementation, 2004, pp. 137–150.
10. Domingos P. and Hulten G. Mining high-speed data streams. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
11. Farnstrom F., Lewis J., and Elkan C. Scalability for clustering algorithms revisited. SIGKDD Explor., 2(1):51–57, 2000.
12. Guha S., Meyerson A., Mishra N., Motwani R., and O'Callaghan L. Clustering Data Streams: Theory and practice. IEEE Trans. Knowl. Data Eng., 15(3):515–528, 2003.
13. Guha S., Mishra N., Motwani R., and O'Callaghan L. Clustering data streams. In Proc. 41st Annual Symp. on Foundations of Computer Science, 2000, p. 359.
14. Muthukrishnan S. Data streams: algorithms and applications. Found. Trend Theor. Comput. Sci., 1(2), 2005.
15. Zhang T., Ramakrishnan R., and Livny M. BIRCH: A New Data Clustering Algorithm and Its Applications. Data Min. Knowl. Discov., 1(2):141–182, 1997.

Clustering Overview and Applications

Dimitrios Gunopulos[1,2]
[1]Computer Science and Eng. Dept., Univ. of California Riverside, Riverside, CA 92521, USA
[2]Dept. of Informatics and Telecommunications, University of Athens, Athens, Greece

Synonyms

Unsupervised learning

Definition

Clustering is the assignment of objects to groups of similar objects (clusters). The objects are typically described as vectors of features (also called attributes). Attributes can be numerical (scalar) or categorical. The assignment can be hard, where each object belongs to one cluster, or fuzzy, where an object can belong to several clusters with a probability. The clusters can be overlapping, though typically they are disjoint. A distance measure is a function that quantifies the similarity of two objects.

Historical Background

Clustering is one of the most useful tasks in data analysis. The goal of clustering is to discover groups of similar objects and to identify interesting patterns in the data. Typically, the clustering problem is about partitioning a given data set into groups (clusters) such that the data points in a cluster are more similar to each other than points in different clusters [4,8]. For example, consider a retail database where each record contains items purchased at the same time by a customer. A clustering procedure could group the customers in such a way that customers with similar buying patterns are in the same cluster. Thus, the main concern in the clustering process is to reveal the organization of patterns into "sensible" groups, which allow one to discover similarities and differences, as well as to derive useful conclusions about them. This idea is applicable to many fields, such as life sciences, medical sciences and engineering. Clustering may be found under different names in different contexts, such as unsupervised learning (in pattern recognition), numerical taxonomy (in biology, ecology), typology (in social sciences) and partition (in graph theory) [13].

The clustering problem comes up in so many domains due to the prevalence of large datasets for which labels are not available. In one or two dimensions, humans can perform clustering very effectively visually, however in higher dimensions automated procedures are necessary. The lack of training examples makes it very difficult to evaluate the results of the clustering process. In fact, the clustering process may result in different partitioning of a data set, depending on the specific algorithm, criterion, or choice of parameters used for clustering.

Foundations

The Clustering Process

In the clustering process, there are no predefined classes and no examples that would show what kind of

desirable relations should be valid among the data. That is the main difference from the task of classification: Classification is the procedure of assigning an object to a predefined set of categories [FSSU96]. Clustering produces initial categories in which values of a data set are classified during the classification process. For this reason, clustering is described as "unsupervised learning"; in contrast to classification, which is considered as "supervised learning." Typically, the clustering process will include at least the following steps:

1. Feature selection: Typically, the objects or observations to be clustered are described using a set of features. The goal is to appropriately select the features on which clustering is to be performed so as to encode as much information as possible concerning the task of interest. Thus, a pre-processing step may be necessary before using the data.
2. Choice of the clustering algorithm. In this step the user chooses the algorithm that is more appropriate for the data at hand, and therefore is more likely to result to a good clustering scheme. In addition, a similarity (or distance) measure and a clustering criterion are selected in tandem
   - The distance measure is a function that quantifies how "similar" two objects are. In most of the cases, one has to ensure that all selected features contribute equally to the computation of the proximity measure and there are no features that dominate others.
   - The clustering criterion is typically a cost function that the clustering algorithm has to optimize. The choice of clustering criterion has to take into account the type of clusters that are expected to occur.
3. Validation and interpretation of the results. The correctness of the results of the clustering algorithm is verified using appropriate criteria and techniques. Since clustering algorithms define clusters that are not known a priori, irrespective of the clustering methods, the final partition of the data typically requires some kind of evaluation. In many cases, the experts in the application area have to integrate the clustering results with other experimental evidence and analysis in order to draw the right conclusion.

After the third phase the user may elect to use the clustering results obtained, or may start the process from the beginning, perhaps using different clustering algorithms or parameters.

Clustering Algorithms Taxonomy

With clustering being a useful tool in diverse research communities, a multitude of clustering methods has been proposed in the literature. Occasionally similar techniques have been proposed and used in different communities. Clustering algorithms can be classified according to:

1. The type of data input to the algorithm (for example, objects described with numerical features or categorical features) and the choice of similarity function between two objects.
2. The clustering criterion optimized by the algorithm.
3. The theory and fundamental concepts on which clustering analysis techniques are based (e.g., fuzzy theory, statistics).

A broad classification of clustering algorithms is the following [8,14]:

1. *Partitional clustering algorithms*: here the algorithm attempts to directly decompose the data set into a set of (typically) disjoint clusters. More specifically, the algorithm attempts to determine an integer number of partitions that optimize a certain criterion function.
2. *Hierarchical clustering algorithms*: here the algorithm proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. The result of the algorithm is a tree of clusters, called dendrogram, which shows how the clusters are related. By cutting the dendrogram at a desired level, a clustering of the data items into disjoint groups is obtained.
3. *Density-based clustering*: The key idea of this type of clustering is to group neighbouring objects of a data set into clusters based on density conditions. This includes grid-based algorithms that quantise the space into a finite number of cells and then do operations in the quantised space.

For each of above categories there is a wealth of subtypes and different algorithms for finding the clusters. Thus, according to the type of variables allowed in the data set additional categorizations include [14]: (i) Statistical algorithms, which are based on statistical analysis concepts and use similarity measures to partition objects and they are limited to numeric data.

(ii) Conceptual algorithms that are used to cluster categorical data. (iii) Fuzzy clustering algorithms, which use fuzzy techniques to cluster data and allow objects to be classified into more than one clusters. Such algorithms lead to clustering schemes that are compatible with everyday life experience as they handle the uncertainty of real data. (iv) Crisp clustering techniques, that consider non-overlapping partitions so that a data point either belongs to a class or not. Most of the clustering algorithms result in crisp clusters, and thus can be categorized in crisp clustering. (v) Kohonen net clustering, which is based on the concepts of neural networks.

In the remaining discussion, partitional clustering algorithms will be described in more detail; other techniques will be dealt with separately.

Partitional Algorithms

In general terms, the clustering algorithms are based on a criterion for assessing the quality of a given partitioning. More specifically, they take as input some parameters (e.g., number of clusters, density of clusters) and attempt to define the best partitioning of a data set for the given parameters. Thus, they define a partitioning of a data set based on certain assumptions and not necessarily the "best" one that fits the data set.

In this category, K-Means is a commonly used algorithm [10]. The aim of K-Means clustering is the optimisation of an objective function that is described by the equation:

$$E = \sum_{i=1}^{c} \sum_{x \in C_i} d(x, m_i)$$

In the above equation, m_i is the center of cluster C_i, while $d(x, m_i)$ is the Euclidean distance between a point x and m_i. Thus, the criterion function E attempts to minimize the distance of every point from the center of the cluster to which the point belongs.

It should be noted that optimizing E is a combinatorial problem that is NP-Complete and thus any practical algorithm to optimize it cannot guarantee optimality. The K-means algorithm is the first practical and effective heuristic that was suggested to optimize this criterion, and owes its popularity to its good performance in practice. The K-means algorithm begins by initialising a set of c cluster centers. Then, it assigns each object of the dataset to the cluster whose center is the nearest, and re-computes the centers. The process continues until the centers of the clusters stop changing.

Another algorithm of this category is PAM (Partitioning Around Medoids). The objective of PAM is to determine a representative object (medoid) for each cluster, that is, to find the most centrally located objects within the clusters. The algorithm begins by selecting an object as medoid for each of c clusters. Then, each of the non-selected objects is grouped with the medoid to which it is the most similar. PAM swaps medoids with other non-selected objects until all objects qualify as medoid. It is clear that PAM is an expensive algorithm with respect to finding the medoids, as it compares an object with the entire dataset [12].

CLARA (Clustering Large Applications), is an implementation of PAM in a subset of the dataset. It draws multiple samples of the dataset, applies PAM on samples, and then outputs the best clustering out of these samples [12]. CLARANS (Clustering Large Applications based on Randomized Search), combines the sampling techniques with PAM. The clustering process can be presented as searching a graph where every node is a potential solution, that is, a set of k medoids. The clustering obtained after replacing a medoid is called the neighbour of the current clustering. CLARANS selects a node and compares it to a user-defined number of their neighbours searching for a local minimum. If a better neighbor is found (i.e., having lower-square error), CLARANS moves to the neighbour's node and the process starts again; otherwise the current clustering is a local optimum. If the local optimum is found, CLARANS starts with a new randomly selected node in search for a new local optimum.

The algorithms described above result in crisp clusters, meaning that a data point either belongs to a cluster or not. The clusters are non-overlapping and this kind of partitioning is further called crisp clustering. The issue of uncertainty support in the clustering task leads to the introduction of algorithms that use fuzzy logic concepts in their procedure. A common fuzzy clustering algorithm is the Fuzzy C-Means (FCM), an extension of classical C-Means algorithm for fuzzy applications [2]. FCM attempts to find the most characteristic point in each cluster, which can be considered as the "center" of the cluster and, then, the grade of membership for each object in the clusters.

Another approach proposed in the literature to solve the problems of crisp clustering is based on probabilistic models. The basis of this type of clustering algorithms is the EM algorithm, which provides a quite general approach to learning in presence of

unobservable variables [11]. A common algorithm is the probabilistic variant of K-Means, which is based on the mixture of Gaussian distributions. This approach of K-Means uses probability density rather than distance to associate records with clusters. More specifically, it regards the centers of clusters as means of Gaussian distributions. Then, it estimates the probability that a data point is generated by the jth Gaussian (i.e., belongs to jth cluster). This approach is based on Gaussian model to extract clusters and assigns the data points to clusters assuming that they are generated by normal distribution. Also, this approach is implemented only in the case of algorithms based on the EM (Expectation Maximization) algorithm.

Another type of clustering algorithms combine graph partitioning and hierarchical clustering algorithms characteristics. Such algorithms include CHAMELEON [9], which measures the similarity among clusters based on a dynamic model contrary to the clustering algorithms discussed above. Moreover in the clustering process both the inter-connectivity and closeness between two clusters are taken into account to decide how to merge the clusters. The merge process based on the dynamic model facilitates the discovery of natural and homogeneous clusters. Also it is applicable to all types of data as long as a similarity function is specified. Finally, BIRCH [ZRL99] uses a data structure called CF-Tree for partitioning the incoming data points in an incremental and dynamic way, thus providing an effective way to cluster very large datasets.

Partitional algorithms are applicable mainly to numerical data sets. However, there are some variants of K-Means such as K-prototypes, and K-mode [7] that are based on the K-Means algorithm, but they aim at clustering categorical data. K-mode discovers clusters while it adopts new concepts in order to handle categorical data. Thus, the cluster centers are replaced with "modes," a new dissimilarity measure used to deal with categorical objects.

The K-means algorithm and related techniques tend to produce spherical clusters due to the use of a symmetric objective function. They require the user to set only one parameter, the desirable number of clusters K. However, since the objective function gets smaller monotonically as K increases, it is not clear how to define what is the best number of clusters for a given dataset. Although several approaches have been proposed to address this shortcoming [14], this is one of the main disadvantages of partitional algorithms. Another characteristic of the partitional algorithms is that they are unable to handle noise and outliers and they are not suitable to discover clusters with non-convex shapes. Another characteristic of K-means is that the algorithm does not display a monotone behavior with respect to K. For example, if a dataset is clustered into M and 2M clusters, it is intuitive to expect that the smaller clusters in the second clustering will be subsets of the larger clusters in the first; however this is typically not the case.

Key Applications

Cluster analysis is very useful task in exploratory data analysis and a major tool in a very wide spectrum of applications in many fields of business and science. Clustering applications include:

1. *Data reduction.* Cluster analysis can contribute to the compression of the information included in the data. In several cases, the amount of the available data is very large and its processing becomes very demanding. Clustering can be used to partition the data set into a number of "interesting" clusters. Then, instead of processing the data set as an entity, the representatives of the defined clusters are adopted in the process. Thus, data compression is achieved.
2. *Hypothesis generation.* Cluster analysis is used here in order to infer some hypotheses concerning the data. For instance, one may find in a retail database that there are two significant groups of customers based on their age and the time of purchases. Then, one may infer some hypotheses for the data, that it, "young people go shopping in the evening," "old people go shopping in the morning."
3. *Hypothesis testing.* In this case, the cluster analysis is used for the verification of the validity of a specific hypothesis. For example, consider the following hypothesis: "Young people go shopping in the evening." One way to verify whether this is true is to apply cluster analysis to a representative set of stores. Suppose that each store is represented by its customer's details (age, job, etc.) and the time of transactions. If, after applying cluster analysis, a cluster that corresponds to "young people buy in the evening" is formed, then the hypothesis is supported by cluster analysis.
4. *Prediction based on groups.* Cluster analysis is applied to the data set and the resulting clusters are characterized by the features of the patterns that

belong to these clusters. Then, unknown patterns can be classified into specified clusters based on their similarity to the clusters' features. In such cases, useful knowledge related to this data can be extracted. Assume, for example, that the cluster analysis is applied to a data set concerning patients infected by the same disease. The result is a number of clusters of patients, according to their reaction to specific drugs. Then, for a new patient, one identifies the cluster in which he/she can be classified and based on this decision his/her medication can be made.

5. *Business Applications and Market Research.* In business, clustering may help marketers discover significant groups in their customers' database and characterize them based on purchasing patterns.
6. *Biology and Bioinformatics.* In biology, it can be used to define taxonomies, categorize genes with similar functionality and gain insights into structures inherent in populations.
7. *Spatial data analysis.* Due to the huge amounts of spatial data that may be obtained from satellite images, medical equipment, Geographical Information Systems (GIS), image database exploration etc., it is expensive and difficult for the users to examine spatial data in detail. Clustering may help to automate the process of analysing and understanding spatial data. It is used to identify and extract interesting characteristics and patterns that may exist in large spatial databases.
8. *Web mining.* Clustering is used to discover significant groups of documents on the Web huge collection of semi-structured documents. This classification of Web documents assists in information discovery. Another application of clustering is discovering groups in social networks.

In addition, clustering can be used as a pre-processing step for other algorithms, such as classification, which would then operate on the detected clusters.

Cross-references

- ▶ Cluster and Distance Measure
- ▶ Clustering for Post Hoc Information Retrieval
- ▶ Clustering on Streams
- ▶ Clustering Validity
- ▶ Clustering with Constraints
- ▶ Data Mining
- ▶ Data Reduction
- ▶ Density-Based Clustering
- ▶ Dimension Reduction Techniques for Clustering
- ▶ Document Clustering
- ▶ Feature Selection for Clustering
- ▶ Hierarchial Clustering
- ▶ Semi-Supervised Learning
- ▶ Spectral Clustering
- ▶ Subspace Clustering Techniques
- ▶ Text Clustering
- ▶ Visual Clustering
- ▶ Visualizing Clustering Results

Recommended Reading

1. Agrawal R., Gehrke J., Gunopulos D., and Raghavan P. Automatic subspace clustering of high dimensional data for data mining applications. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 94–105.
2. Bezdeck J.C., Ehrlich R., and Full W. FCM: Fuzzy C-Means algorithm. Comput. Geosci., 10(2–3):191–203, 1984.
3. Ester M., Kriegel H.-Peter., Sander J., and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
4. Everitt B.S., Landau S., and Leese M. Cluster Analysis. Hodder Arnold, London, UK, 2001.
5. Fayyad U.M., Piatesky-Shapiro G., Smuth P., and Uthurusamy R. Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, CA, 1996.
6. Han J. and Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Fransisco, CA, 2001.
7. Huang Z. A fast clustering algorithm to cluster very large categorical data sets in data mining. In Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 1997.
8. Jain A.K., Murty M.N., and Flyn P.J. Data clustering: a review. ACM Comput. Surv., 31(3):264–323, 1999.
9. Karypis G., Han E.-H., and Kumar V. CHAMELEON: a hierarchical clustering algorithm using dynamic modeling. IEEE Computer., 32(8):68–75, 1999.
10. MacQueen J.B. Some methods for classification and analysis of multivariate observations. In Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability, vol. 1, 1967, pp. 281–297.
11. Mitchell T. Machine Learning. McGraw-Hill, New York, 1997.
12. Ng R. and Han J. Efficient and effective clustering methods for spatial data mining. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 144–155.
13. Theodoridis S. and Koutroubas K. Pattern Recognition. Academic Press, New York, 1999.
14. Vazirgiannis M., Halkidi M., and Gunopulos D. Uncertainty Handling and Quality Assessment in Data Mining. Springer, New York, 2003.
15. Wang W., Yang J., and Muntz R. STING: A statistical information grid approach to spatial data mining. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 186–195.
16. Zhang T., Ramakrishnman R., and Linvy M. BIRCH: an efficient method for very large databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 103–114.

Clustering Validity

Michalis Vazirgiannis
Athens University of Economics & Business, Athens, Greece

Synonyms

Cluster validation; Cluster stability; Quality assessment; Stability-based validation of clustering

Definition

A problem one faces in clustering is to decide the optimal partitioning of the data into clusters. In this context visualization of the data set is a crucial verification of the clustering results. In the case of large multidimensional data sets (e.g., more than three dimensions) effective visualization of the data set is cumbersome. Moreover the perception of clusters using available visualization tools is a difficult task for humans that are not accustomed to higher dimensional spaces. The procedure of evaluating the results of a clustering algorithm is known under the term *cluster validity*. Cluster validity consists of a set of techniques for finding a set of clusters that best fits natural partitions (of given datasets) without any a priori class information. The outcome of the clustering process is validated by a cluster validity index.

Historical Background

Clustering is a major task in the data mining process for discovering groups and identifying interesting distributions and patterns in the underlying data. In the literature a wide variety of algorithms for different applications and sizes of data sets. The application of an algorithm to a data set, assuming that the data set offers a clustering tendency, aims at discovering its inherent partitions. However, the clustering process is an unsupervised process, since there are no predefined classes or examples. Then, the various clustering algorithms are based on some assumptions in order to define a partitioning of a data set. As a consequence, they may behave in a different way depending on: i. the features of the data set (geometry and density distribution of clusters) and ii. the input parameter values.

One of the most important issues in cluster analysis is the evaluation of clustering results to find the partitioning that best fits the underlying data. This is the main subject of cluster validity. If clustering algorithm parameters are assigned an improper value, the clustering method results in a partitioning scheme that is not optimal for the specific data set leading to wrong decisions. The problems of deciding the number of clusters better fitting a data set as well as the evaluation of the clustering results has been subject of several research efforts. The procedure of evaluating the results of a clustering algorithm is known under the term cluster validity. In general terms, there are three approaches to investigate cluster validity. The first is based on *external criteria*. This implies that the results of a clustering algorithm are evaluated based on a prespecified structure, which is imposed on a data set and reflects one's intuition about the clustering structure of the data set. The second approach is based on *internal criteria*. The results of a clustering algorithm may be evaluated in terms of quantities that involve the vectors of the data set themselves (e.g., proximity matrix). The third approach of clustering validity is based on *relative criteria*. Here the basic idea is the evaluation of a clustering structure by comparing it to other clustering schemes, resulting by the same algorithm but with different parameter values. There are two criteria proposed for clustering evaluation and selection of an optimal clustering scheme: (i) Compactness, the members of each cluster should be as close to each other as possible. A common measure of compactness is the variance, which should be minimized. (ii) Separation, the clusters themselves should be widely spaced.

Foundations

This section discusses methods suitable for the quantitative evaluation of the clustering results, known as cluster validity methods. However, these methods give an indication of the quality of the resulting partitioning and thus they can only be considered as a tool at the disposal of the experts in order to evaluate the clustering results. The cluster validity approaches based on external and internal criteria rely on statistical hypothesis testing. In the following section, an introduction to the fundamental concepts of hypothesis testing in cluster validity is presented.

In cluster validity the basic idea is to test whether the points of a data set are randomly structured or not. This analysis is based on the *Null Hypothesis,* denoted as *Ho*, expressed as a statement of random structure of a data set X. To test this hypothesis, statistical tests are used, which lead to a computationally complex procedure.

Monte Carlo techniques are used as a solution to this problem.

External Criteria

Based on external criteria, one can work in two different ways. First, one can evaluate the resulting clustering structure **C**, by comparing it to an independent partition of the data ***P*** built according to one's intuition about the clustering structure of the data set. Second, one can compare the proximity matrix P to the partition ***P***.

Comparison of C with Partition P (Non-hierarchical Clustering) Let $\boldsymbol{C} = \{C_1...C_m\}$ be a clustering structure of a data set X and $\boldsymbol{P} = \{P_1...P_s\}$ be a defined partition of the data. Refer to a pair of points $(\mathbf{x_v}, \mathbf{x_u})$ from the data set using the following terms:

- ***SS***: if both points belong to the same cluster of the clustering structure **C** and to the same group of partition ***P***.
- ***SD***: if points belong to the same cluster of **C** and to different groups of ***P***.
- ***DS***: if points belong to different clusters of **C** and to the same group of ***P***.
- ***DD***: if both points belong to different clusters of **C** and to different groups of ***P***.

Assuming now that **a**, **b**, **c** and **d** are the number of SS, SD, DS and DD pairs respectively, then $a+b+c+d=\mathbf{M}$ which is the maximum number of all pairs in the data set (meaning, $M = N(N-1)/2$ where N is the total number of points in the data set).

Now define the following indices to measure the degree of similarity between ***C*** and ***P***:

1. *Rand Statistic*: $R = (a + d)/M$
2. *Jaccard Coefficient*: $J = a/(a + b + c)$
   The above two indices range between 0 and 1, and are maximized when m=s. Another known index is the:
3. *Folkes and Mallows index*:

$$FM = a/\sqrt{m_1 m_2} = \sqrt{\frac{a}{a+b}\cdot\frac{a}{a+c}} \qquad [1]$$

   where $m_1 = (a + b)$, $m_2 = (a + c)$.
   For the previous three indices it has been proven that the higher the values of these indices are the more similar ***C*** and ***P*** are. Other indices are:

4. *Huberts Γ statistic*:

$$\Gamma = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} X(i,j)Y(i,j) \qquad [2]$$

   High values of this index indicate a strong similarity between the matrices X and Y.
5. *Normalized Γ statistic*:

$$\hat{\Gamma} = \frac{\left[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(X(i,j)-\mu_X)(Y(i,j)-\mu_Y)\right]}{\sigma_X\sigma_Y} \qquad [3]$$

   where X(i, j) and Y(i, j) are the (i, j) element of the matrices X, Y respectively that one wants to compare. Also μ_x, μ_y, σ_x, σ_y are the respective means and variances of X, Y matrices. This index takes values between –1 and 1.

All these statistics have right-tailed probability density functions, under the random hypothesis. In order to use these indices in statistical tests, one must know their respective probability density function under the Null Hypothesis, *Ho*, which is the hypothesis of random structure of the data set. Thus, if one accepts the Null Hypothesis, the data are randomly distributed. However, the computation of the probability density function of these indices is computationally expensive. A solution to this problem is to use Monte Carlo techniques.

After having plotted the approximation of the probability density function of the defined statistic index, its value, denoted by q, is compared to the $q(C_i)$ values, further referred to as q_i. The indices R, J, FM, Γdefined previously are used as the q index mentioned in the above procedure.

Internal Criteria

Using this approach of cluster validity the goal is to evaluate the clustering result of an algorithm using only quantities and features inherited from the data set. There are two cases in which one applies internal criteria of cluster validity depending on the clustering structure: (i) hierarchy of clustering schemes, and (ii) single clustering scheme.

Validating Hierarchy of Clustering Schemes A matrix called cophenetic matrix, P_c, can represent the

hierarchy diagram that is produced by a hierarchical algorithm. The element $P_c(i, j)$ of cophenetic matrix represents the proximity level at which the two vectors x_i and x_j are found in the same cluster for the first time. A statistical index can be defined to measure the degree of similarity between P_c and P (proximity matrix) matrices. This index is called *Cophenetic Correlation Coefficient* and defined as:

$$\text{CPCC} = \frac{(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}c_{ij} - \mu_P\mu_C}{\sqrt{\left[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2 - \mu_P^2\right]\left[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}^2 - \mu_C^2\right]}}, \quad [4]$$

where $M = N\times(N-1)/2$ and N is the number of points in a data set. Also, μ_p and μ_c are the means of matrices P and P_c respectively, and are defined in the (Eq. 5):

$$\mu_P = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j), \qquad \mu_C = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P_c(i,j) \quad [5]$$

Moreover, d_{ij}, c_{ij} are the (i, j) elements of P and P_c matrices respectively. The CPCC values range in [–1, 1]. A value of the index close to 1 is an indication of a significant similarity between the two matrices.

Validating a Single Clustering Scheme The goal here is to find the degree of match between a given clustering scheme C, consisting of n_c clusters, and the proximity matrix P. The defined index for this approach is Hubert's Γstatistic (or normalized Γstatistic). An additional matrix for the computation of the index is used, that is

$$Y(i,j) = \begin{cases} 1, & \text{if } x_i \text{ and } x_j \text{ belong to different clusters} \\ 0, & \text{otherwise } \textit{array}. \text{ where } i, j = 1, ¼, N. \end{cases}$$

The application of Monte Carlo techniques is also a means to test the random hypothesis in a given data set.

Relative Criteria

The major drawback of techniques based on internal or external criteria is their high computational complexity. A different validation approach is discussed in this section. The fundamental idea of the relative criteria is to choose the best clustering scheme of a set of defined schemes according to a pre-specified criterion. More specifically, the problem can be stated as follows:

Let P_{alg} be the set of parameters associated with a specific clustering algorithm (e.g., the number of clusters n_c). Among the clustering schemes C_i, $i = 1,...,n_c$, is defined by a specific algorithm. For different values of the parameters in P_{alg}, choose the one that best fits the data set.

Then, consider the following cases of the problem:

1. *P_{alg} does not contain the number of clusters, n_c, as a parameter.* In this case, the choice of the optimal parameter values are described as follows: The algorithm runs for a wide range of its parameters' values and the largest range for which n_c remains constant is selected (usually $n_c << N$ (number of tuples)). Then the values that correspond to the middle of this range are chosen as appropriate values of the P_{alg} parameters. Also, this procedure identifies the number of clusters that underlie the data set.
2. *P_{alg} contains n_c as a parameter.* The procedure of identifying the best clustering scheme is based on a validity index. Selecting a suitable performance index, q, one proceeds with the following steps:
   - clustering runs for all values of n_c between n_{cmin} and n_{cmax} defined a priori by the user.
   - For each of n_c values, the algorithm runs r times, using different sets of values for the other parameters of the algorithm (e.g., different initial conditions).
   - The best values of the index q obtained by each n_c are plotted as the function of n_c.

Based on this plot, the best clustering schemes are identified. There are two approaches for defining the best clustering depending on the behavior of q with respect to n_c. Thus, if the validity index does not exhibit an increasing or decreasing trend as n_c increases, one seeks the max (min) of the plot. On the other hand, for indices that increase (decrease) as the number of clusters increase, one searches for the values of n_c at which a significant local change in value of the index occurs. This change appears as a "knee" in the plot and it is an indication of the number of clusters underlying the data set. The absence of a knee is an indication that the data set possesses no

clustering structure. Below, some representative relative validity indices are presented.

The Modified Hubert Γ Statistic

The definition of the modified Hubert Γstatistic is given by the equation

$$\Gamma = (1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} P(i,j)\cdot Q(i,j) \qquad [6]$$

where N is the number of objects in a data set, M = N (N−1)/2, P is the proximity matrix of the data set and Q is an N × N matrix whose (i, j) element is equal to the distance between the representative points (v_{ci}, v_{cj}) of the clusters where the objects x_i and x_j belong.

Similarly, one can define the normalized Hubert Γstatistic, given by equation

$$\hat{\Gamma} = \frac{\left[(1/M)\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(P(i,j)-\mu_P)(Q(i,j)-\mu_Q)\right]}{\sigma_P\sigma_Q}. \qquad [7]$$

where μ_P, μ_Q, σ_P, σ_Q are the respective means and variances of P, Q matrices.

If the $d(v_{ci}, v_{cj})$ is close to $d(x_i, x_j)$ for i, j =1, 2,...,N, P and Q will be in close agreement and the valuesof Γ and $\hat{\Gamma}$(normalized Γ) will be high. Conversely, a high value of Γ ($\hat{\Gamma}$) indicates the existence of compact clusters. Thus, in the plot of normalized Γ versus n_c, one seeks a significant knee that corresponds to a significant increase of normalized Γ. The number of clusters at which the knee occurs is an indication of the number of clusters that occurs in the data. Note that for $n_c = 1$ and $n_c = N$, the index is not defined.

Dunn Family of Indices

A cluster validity index for crisp clustering proposed by Dunn (1974), aims at the identification of "compact and well separated clusters". The index is defined in the following equation for a specific number of clusters

$$D_{n_c} = \min_{i=1,..,n_c}\left\{\min_{j=i+1,...,n_c}\left(\frac{d(c_i,c_j)}{\max_{k=1,...,n_c}(\mathrm{diam}(c_k))}\right)\right\} \qquad [8]$$

where $d(c_i, c_j)$ is the dissimilarity function between two clusters c_i and c_j defined as $d(c_i, c_j) = \min_{x\in C_i, y\in C_j} d(x, y)$, and diam(c) is the diameter of a cluster, which may be considered as a measure of clusters' dispersion. The diameter of a cluster *C* can be defined as follows:

$$\mathrm{diam}(C) = \max_{x,y\in C}\{d(x,y)\} \qquad [9]$$

If the data set contains compact and well-separated clusters, the distance between the clusters is expected large and the diameter of the clusters is expected small. Based on the Dunn's index definition, one concludes that large values of the index indicate the presence of compact and well-separated clusters.

The problems of the Dunn index are: (i) its considerable time complexity, and (ii) its sensitivity to the presence of noise in data sets, since these are likely to increase the values of the diameter.

RMSSDT, SPR, RS, CD

This family of validity indices is applicable in the cases that hierarchical algorithms are used to cluster the data sets. Hereafter the discussion refers to the definitions of four validity indices, which have to be used simultaneously to determine the number of clusters existing in the data set. These four indices are applied to each step of a *hierarchical* clustering algorithm and they are known as:

- *Root-mean-square standard deviation (RMSSTD) of the new cluster,*
- *Semi-partial R-squared (SPR),*
- *R-squared (RS),*
- *Distance between two clusters (CD).*

Getting into a more detailed description of them, one can say that:

RMSSTD of a new clustering scheme defined at a level of a clustering hierarchy is the square root of the variance of all the variables (attributes used in the clustering process). This index measures the homogeneity of the formed clusters at each step of the hierarchical algorithm. Since the objective of cluster analysis is to form homogeneous groups the RMSSTD of a cluster should be as small as possible. Where the values of RMSSTD are higher than the ones of the previous step, one has an indication that the new clustering scheme is worse.

In the following definitions, the term SS is used, which means *Sum of Squares* and refers to the equation:

$$SS = \sum_{i=1}^{n}(X_i - \overline{X})^2 \qquad [10]$$

Along with this, additional terms will be used, such as:

1. SS_w referring to the sum of squares within group,
2. SS_b referring to the sum of squares between groups,
3. SS_t referring to the total sum of squares, of the whole data set.

In the case cluster join to form a new one, *SPR*- for the new cluster – is defined as the difference between SS_w of the new cluster and the sum of the SS_w's values of clusters joined to obtain the new cluster (*loss of homogeneity*), divided by the SS_t for the whole data set. This index measures the loss of homogeneity after merging the two clusters of a single algorithm step. If the index value is zero then the new cluster is obtained by merging two perfectly homogeneous clusters. If its value is high then the new cluster is obtained by merging two heterogeneous clusters.

RS of the new cluster is the ratio of SS_b over SS_t. SS_b is a measure of difference between groups. Since $SS_t = SS_b + SS_w$, the greater the SS_b the smaller the SS_w and vice versa. As a result, the greater the differences between groups, the more homogenous each group is and vice versa. Thus, RS may be considered as a measure of dissimilarity between clusters. Furthermore, it measures the degree of homogeneity between groups. The values of RS range between 0 and 1. Where the value of RS is zero, there is an indication that no difference exists among groups. On the other hand, when RS equals 1 there is an indication of significant difference among groups.

Key Applications

There is a certain cross disciplinary interest for clustering validity method and indices. A prominent area where cluster validity measures apply is the area of biological data [2,6]. Patterns hidden in gene expression data offer a tremendous opportunity for an enhanced understanding of functional genomics. However, the large number of genes and the complexity of biological networks greatly increase the challenges of comprehending and interpreting the resulting mass of data, which often consists of millions of measurements. The data mining process aims to reveal natural structures and identify interesting patterns in the underlying data. Clustering techniques constitute a first essential step toward addressing this challenge. Moreover recent research effort papers in the area of image segmentation [13,3].

The area is fertile as the clustering issue is a fundamental problem and the application domains are still widening. Challenging relevant research directions [9] follow:

- Is there a principled way to measure the quality of a clustering on particular data set?
- Can every clustering task be expressed as an optimization of some explicit, readily computable, objective cost function?
- Can stability be considered a first principle for meaningful clustering?
- How should the similarity between different clusterings be measured?
- Can one distinguish clusterable data from structureless data?
- What are the tools that should be imported from other relevant areas of research?

Cross-references

- ▶ Cluster and Distance Measure
- ▶ Clustering on Streams
- ▶ Clustering with Constraints
- ▶ Density-Based Clustering
- ▶ Document Clustering
- ▶ Feature Selection for Clustering
- ▶ Hierarchial Clustering
- ▶ Semi-Supervised Learning
- ▶ Spectral Clustering
- ▶ Subspace Clustering Techniques
- ▶ Text Clustering
- ▶ Visual Clustering
- ▶ Visualizing Clustering Results

Recommended Reading

1. Bezdek J.C. and Pal N.R. Some new indexes of cluster validity, IEEE Trans., Systems, Man, and Cybernetics, Part B. 28 (3):301–315, 1998.
2. Datta S. and Datta S. Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics, 19(4):459–466, 2003.
3. El-Melegy M.T., Zanaty E.A., Abd-Elhafiez W.M., and Farag A.A. On cluster validity indexes in fuzzy and hard clustering algorithms for image segmentation. In Proc. Int. Conf. Image Processing, 2007, pp. 5–8.
4. Halkidi M., Batistakis Y., and Vazirgiannis M. On clustering validation techniques. J. Intell. Inf. Syst., 17(2–3):107–145, 2001.
5. Halkidi M., Gunopulos D., Vazirgiannis M., Kumar N., and Domeniconi C. A clustering framework based on subjective and objective validity criteria. ACM Trans. Knowl. Discov. Data, 1(4), 2008.

6. Jiang D., Tang C., and Zhang A. Cluster Analysis for Gene Expression Data: A Survey. IEEE Trans. Knowl. Data Eng., 16(11):1370–1386, 2004.
7. Kim M. and Ramakrishna R.S. New indices for cluster validity assessment. Pattern Recogn. Lett., 26(15):2353–2363, 2005.
8. Maulik U. and Bandyopadhyay S. Performance evaluation of some clustering algorithms and validity indices. IEEE Trans. Pattern Anal. Mach. Intell., 24(12):1650–1654, 2002.
9. NIPS 2005 workshop on theoretical foundations of clustering, Saturday, December 10th, 2005. Available at: (http://www.kyb.tuebingen.mpg.de/bs/people/ule/clustering_workshop_nips05/clustering_workshop_nips05.htm_).
10. Pal N.R. and Bezdek J.C. On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Systems., 3(3):370–379, 1995.
11. Rand W.M. Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc., 66(336):846–850, 1971.
12. Wang J.-S. and Chiang J.-C. A cluster validity measure with a hybrid parameter search method for the support vector clustering algorithm. Pattern Recognit., 41(2):506–520, 2008.
13. Zhang J. and Modestino J.W. A model-fitting approach to cluster validation with application to stochastic model-based image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 12(10):1009–1017, 1990.

Clustering with Constraints

Ian Davidson
University of California-Davis, Davis, CA, USA

Synonyms

Semi-supervised clustering

Definition

The area of clustering with constraints makes use of hints or advice in the form of constraints to aid or bias the clustering process. The most prevalent form of advice are conjunctions of pair-wise instance level constraints of the form must-link (ML) and cannot-link (CL) which state that pairs of instances should be in the same or different clusters respectively. Given a set of points P to cluster and a set of constraints C, the aim of clustering with constraints is to use the constraints to improve the clustering results. Constraints have so far being used in two main ways: (i) Writing algorithms that use a standard distance metric but attempt to satisfy all or as many constraints as possible and (ii) Using the constraints to learn a distance function that is then used in the clustering algorithm.

Historical Background

The idea of using constraints to guide clustering was first introduced by Wagstaff and Cardie in their seminal paper ICML 2000 [13] with a modified COBWEB-style algorithm that attempts to satisfy all constraints. Later [14] they introduced constraints to the *k*-means algorithms. Their algorithms (as most algorithms now do) look at satisfying a conjunction of must-link and cannot-link constraints. Independently, Cohn, Caruana and McCallum [3,4] introduced constraints as a user feedback mechanism to guide the clustering algorithm to a more useful result.

In 2002 Xing and collaborators [15] (NIPS 2002) and Klein and collaborators (ICML 2002) [12] explored making use of constraints by learning a distance function for non-hierarchical clustering and a distance matrix for hierarchical clustering respectively.

Basu and collaborators more recently have looked at key issues such as which are the most informative sets of constraints [2] and seeding algorithms using constraints [1]. Gondek has explored using constraints to find orthogonal/alternative clusterings of data [3,11]. Davidson and Ravi explored the intractability issues of clustering under constraints for non-hierarchical clustering [6], hierarchical clustering [5] and non-hierarchical clustering with feedback [9].

Foundations

Clustering has many successful applications in a variety of domains where the objective function of the clustering algorithm finds a novel and useful clustering. However, in some application domains the typical objective functions may lead to well-known or non-actionable clusterings of the data. This could be overcome by an ad hoc approach such as manipulating the data. The introduction of constraints into clustering allows a principled approach to incorporate user preferences or domain expertise into the clustering process so as to guide the algorithm to a desirable solution or away from an undesirable solution. The typical semi-supervised learning situations involves having a label associated with a subset of the available instances. However in many domains, knowledge of the relevant categories is incomplete and it is easier to obtain pairwise constraints either automatically or from domain experts.

Types of Constraints. Must-link and cannot-link constraints are typically used since they can be easily generated from small amounts of labeled data (generate a must-link between two instances if the labels

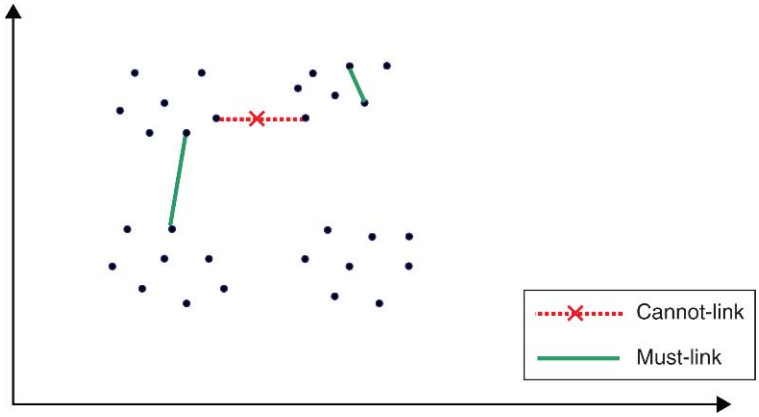
agree, cannot-link if they disagree) or from domain experts. They can be used to represent geometric properties [6,14] by noting that for instance, making the maximum cluster diameter be α is equivalent to enforcing a conjunction of cannot-link constraints between all points whose distance is greater than α. Similarly, clusters can be separated by distance at at least δ by enforcing a conjunction of must-link constraints between all points whose distance is less than δ. Both types of instance-level constraints have interesting properties that can be used to effectively generate many additional constraints. Must-link constraints are transitive: $ML(x,y)$, $ML(y,z) \rightarrow ML(x,z)$ and cannot link constraints have an entailment property: $ML(a,b)$, $ML(x,y)$, $CL(a,x) \rightarrow CL(a,y)$, $CL(b,x)$, $CL(b,y)$.

How Constraints Are Used. Constraints have typically been used in clustering algorithms in two ways. Constraints can be used to modify the cluster assignment stage of the cluster algorithm [4,14], to enforce satisfaction of the constraints or as many as possible [2,6]. These approaches typically use a standard distance or likelihood function. Alternatively, the distance function of the clustering algorithm can also be trained either before or after the clustering actually occurs using the constraints [12,15]. The former are called constraint-based approaches and the later distance based approaches.

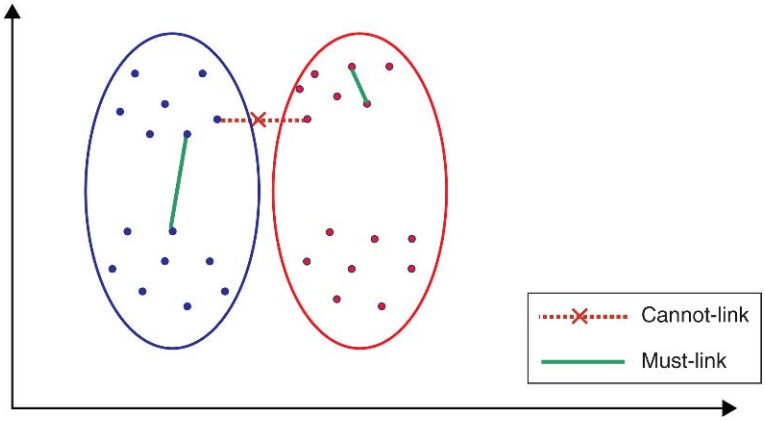
Constraint-Based Methods. In constraint-based approaches, the clustering algorithm itself (typically the assignment step) is modified so that the available constraints are used to bias the search for an appropriate clustering of the data. Fig. 2 shows how though two clusterings exist (a horizontal and vertical clustering) just three constraints can rule out the former.

Constraint-based clustering is typically achieved using one of the following approaches:

1. Enforcing constraints to be satisfied during the cluster assignment in the clustering algorithm [5,13].



Clustering with Constraints. Figure 1. Input instances and constraints.



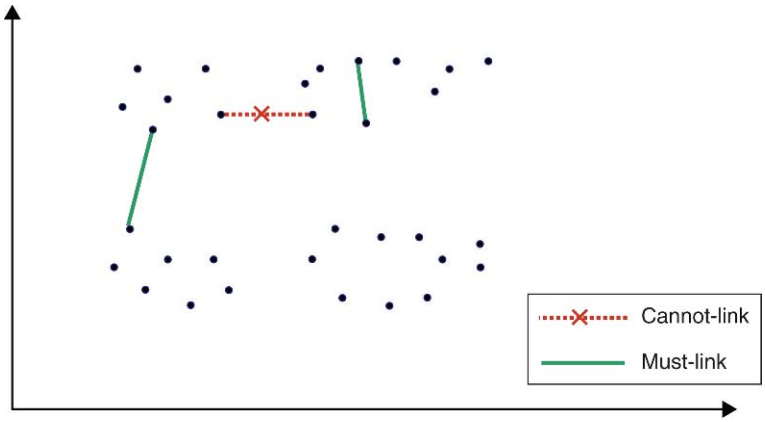
Clustering with Constraints. Figure 2. A clustering that satisfies all constraints.

2. Modifying the clustering objective function so that it includes a term for satisfying specified constraints. Penalties for violating constraints have been explored in the maximum likelihood framework [2] and distance framework [6].
3. Initializing clusters and inferring clustering constraints based on neighborhoods derived from labeled examples [1].

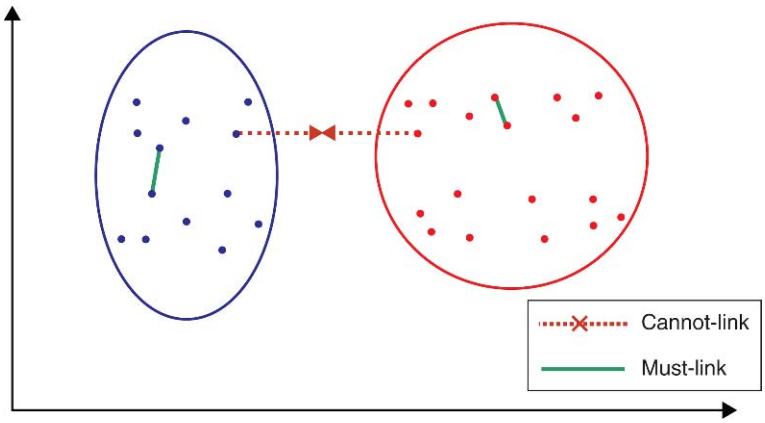
Each of the above approaches provides a simple method of modifying existing partitional and agglomerative style hierarchical algorithms to incorporate constraints. For more recent advances in algorithm design such as the use of variational techniques for constrained clustering see [3].

Distance-Based Methods. In distance-based approaches, an existing clustering algorithm that uses a distance measure is employed. However, rather than use the Euclidean distance metric, the distance measure is first trained to "satisfy" the given constraints. The approach of Xing and collaborators [15] casts the problem of learning a distance *metric* from the constraints so that the points (and surrounding points) that are part of the must-link (cannot-link) constraints are close together (far apart). They consider two formulations: firstly learning a generalized Mahanabolis distance metric which essentially stretches or compresses each axis as appropriate. Figure 4 gives an example where the constraints can be satisfied by stretching the *x*-axis and compressing the *y*-axis and then applying a clustering algorithm to the new data space. The second formulation allows a more complex transformation on the space of points.

Klein and collaborators [12] explore learning a distance *matrix* from constraints for agglomerative clustering. Only points that are directly involved in the constraints are brought closer together or far apart



Clustering with Constraints. Figure 3. Input instances and constraints.



Clustering with Constraints. Figure 4. A learnt distance space respective of the constraints.

using a multi-step approach of making must-linked points have a distance of 0 and cannot-linked points having the greatest distance.

There have been some algorithms that try to both enforce constraints and learn distance functions from constraints [2].

Key Applications

Key application areas include images, video, biology, text, web pages, audio (speaker identification) [3] and GPS trace information [14].

URL to Code

http://www.constrained-clustering.org

Cross-references

- ▶ Clustering
- ▶ Semi-Supervised Learning

Recommended Reading

1. Basu S., Banerjee A., and Mooney R. Semi-supervised clustering by seeding. In Proc. 19th Int. Conf. on Machine Learning, 2002, pp. 27–34.
2. Basu S., Banerjee A., and Mooney R.J. Active semi-supervision for pairwise constrained clustering. In Proc. SIAM International Conference on Data Mining, 2004.
3. Basu S., Davidson I., and Wagstaff K. (eds.). Constrained Clustering: Advances in Algorithms, Theory and Applications. Chapman & Hall, CRC Press, 2008.
4. Cohn D., Caruana R., and McCallum A. Semi-Supervised Clustering with User Feedback. Technical Report 2003–1892. Cornell University, 2003.
5. Davidson I. and Ravi S.S. Agglomerative hierarchical clustering with constraints: theoretical and empirical results. In Principles of Data Mining and Knowledge Discovery, 9th European Conf., 2005, pp. 59–70.
6. Davidson I. and Ravi S.S. Clustering with constraints: feasibility issues and the *k*-means algorithm. In Proc. SIAM International Conference on Data Mining, 2005.
7. Davidson I. and Ravi S.S. Identifying and generating easy sets of constraints for clustering. In Proc. 15th National Conf. on AI, 2006.
8. Davidson I., Ester M., and Ravi S.S. Efficient incremental clustering with constraints. In Proc. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2007, pp. 204–249.
9. Davidson I. and Ravi S.S. Intractability and clustering with constraints. In Proc. 24th Int. Conf. on Machine Learning, 2007, pp. 201–208.
10. Davidson I. and Ravi S.S. The complexity of non-hierarchical clustering with instance and cluster level constraints. Data Mining Know. Discov., 14(1):25–61, 2007.
11. Gondek D. and Hofmann T. Non-redundant data clustering. In Proc. 2004 IEEE Int. Conf. on Data Mining, 2004, pp. 75–82.
12. Klein D., Kamvar S.D., and Manning C.D. From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In Proc. 19th Int. Conf. on Machine Learning, 2002, pp. 307–314.
13. Wagstaff K. and Cardie C. Clustering with instance-level constraints. In Proc. 17th Int. Conf. on Machine Learning, 2000, pp. 1103–1110.
14. Wagstaff K., Cardie C., Rogers S., and Schroedl S. Constrained K-means clustering with background knowledge. In Proc. 18th Int. Conf. on Machine Learning, 2001, pp. 577–584.
15. Xing E., Ng A., Jordan M., and Russell S. Distance metric learning, with application to clustering with side-information. Adv. Neural Inf. Process. Syst. 15, 2002.

CM Sketch

▶ Count-Min Sketch

CMA

▶ Computational Media Aesthetics

CO Query, Content-Only Query

▶ Content-Only Query

CO+S Query

▶ Content-and-Structure Query

Co-clustering

▶ Subspace Clustering Techniques

CODASYL Data Model

▶ Network Data Model

Collaborative Software

▶ Social Applications

Co-locations

► Spatial Data Mining

Colored Nets

► Petri Nets

Column Segmentation

Sunita Sarawagi
IIT Bombay, Mumbai, India

Synonyms

Text segmentation; Record extraction; Information extraction

Definition

The term column segmentation refers to the segmentation of an unstructured text string into segments such that each segment is a column of a structured record.

As an example, consider a text string S=``18100 New Hampshire Ave. Silver Spring, MD 20861'' representing an unstructured form of an Address record. Let the columns of this record be House number, Street name, City name, State, Zip and Country. In column segmentation, the goal is to segment S and assign a column label to each segment so as to get an output of the form:

```
House Number   :  18100
Street Name    :  New Hampshire Ave.
City           :  Silver Spring
State          :  MD
Zip            :  20861
Country        :  --
```

Historical Background

The column segmentation problem is a special case of a more general problem of Information Extraction (IE) that refers to the extraction of structure from unstructured text. Column segmentation is typically performed on short text strings where most of the tokens belong to one of a fixed set of columns. In the more general IE problem, the unstructured text could be an arbitrary paragraph or an HTML document where the structured entities of interest form a small part of the entire string.

There is a long history of work on information extraction [5]. Most of the early work in the area was in the context of natural language processing, for example for extracting named entities like people names, organization names, and location names from news articles. The early systems were based on hand-coded set of rules and relied heavily on dictionaries of known records. Later systems were based on statistical methods like maximum entropy taggers [9], Hidden Markov Models [11] and Conditional Random Fields (CRFs) [7].

In the database research community, interest in column segmentation arose in the late nineties as a step in the process of cleaning text data for data warehousing. Many commercial tools were developed purely for the purposes of cleaning names and addresses. These were based on hand-coded, rule-based, database driven methods that work only for the region that they are developed for and do not extend to other domains. Much manual work has to be done to rewrite these rules when shifting the domain from one locality to another. This led to the adoption of statistical techniques [1,3] which proved to be more robust to noisy inputs.

Foundations

A formal definition of column segmentation follows. Let $Y = \{y_1,...,y_m\}$ denote the set of column types of the structured record. Given any unstructured text string $\mathbf{x}$, column segmentation finds segments of $\mathbf{x}$ and labels each with one of the columns in Y. The input $\mathbf{x}$ is typically treated as a sequence of tokens obtained by splitting $\mathbf{x}$ along a set of delimiters. Let $x_1,...,x_n$ denote such a sequence of tokens. A segmentation of $\mathbf{x}$ is a sequence of segments $s_1...s_p$. Each segment s_j consists of a *start position* t_j, an *end position* u_j, and a *label* $y_j \in Y \cup \{\text{“Other”}\}$. The special label “Other” is used to label tokens not belonging to any of the columns. The segments are assumed to be contiguous, that is, segment s_{j+1} begins right after segment s_j ends. Also, the last segment ends at n and the first segment starts at 1.

As a second example consider a citation String T=P.P.Wangikar, T.P. Graycar, D.A. Estell, D.S. Clark, J.S. Dordick (1993) Protein and Solvent Engineering of Subtilising BPN' in Nearly Anhydrous Organic Media J.Amer. Chem. Soc. 115, 12231-12237. and a set of

columns: Author names, title, year, publication venue, volume, number. A segmentation of this string is:

```
Title     :   Protein and Solvent Engineering
              of Subtilising BPN
Authors   :   P.P.Wangikar, T.P.Graycar,
              D.A. Estell, D.S. Clark, J.S. Dordick
Year      :   1993
Venue     :   Nearly Anhydrous Organic Media
              J.Amer. Chem. Soc.
Volume    :   115
Number    :   --
Pages     :   12231-12237
```

In this example, the tokens "in", "(" and ")" of the input have been assigned label "Other".

Challenges

The problem of column segmentation is challenging because of the presence of various kinds of noise in the unstructured string.

- The same column might be represented in many different forms, for example "Street" might be abbreviated as "St." or "st".
- The order in which columns appear might be different in different strings: for example, in some citations authors could be before title, and after title in others.
- Columns might be missing: some addresses might contain a country name, others may not.
- Strings from different sources might be formatted differently: for example, some citations might use a comma to separate fields whereas others might have no regular delimiter between fields.

Main Techniques

A column segmentation technique needs to combine information from multiple different sources of evidence to be able to correctly recognize segmentations in noisy strings. One source is the characteristic words in each elements, for example the word "street" appears in road-names. A second source is the limited partial ordering between its element. Often the first element is a house number, then a possible building name and so on and the last few elements are zipcode and state-name. A third source is the typical number of words in each element. For example, state names usually have one or two words whereas road names are longer. Even within a field, some words are more likely to appear in the beginning of the field rather than towards its end. Often, there is a pre-existing database of known values of columns. Match of a substring of the text to an existing database column, can be a valuable clue for segmentation. The format of the entry, presence of certain regular expression, capitalization, and punctuation patterns can be useful when word-level matches are absent. A good column segmentation technique would combine evidence from all of these clues in performing the final segmentation.

The three main types of column segmentation techniques are:

Rule-Based Systems

A rule-based technique, as the name suggests, encodes one or more of the above clues as rules. These are applied in a specified order and when more than two rules conflict, another set of rule resolution mechanisms are used to decide which one wins.

Here are some examples of rules that can be used to extract columns from citation records:

Punctuation CapsWord{2–10} Dot → Title
CapsWord Comma Initial Dot Initial Dot → Author name
Initial Dot CapsWord Comma → Author name
AllCaps Words{1–2} Journal → Journal

For example, the first rule marks as title any substring of two to ten capitalized words appearing between a punctuation and a full-stop. The second rule marks as an author name any substring consisting of a capitalized word followed by comma and two initials. This would identify strings of the form "Gandhi, M. K." as author names. Whereas the third rule would mark strings like "V. Ganti," as author names. The fourth rule would mark string like "ACM computing Journal" as journal names.

Such rules could be either hand-coded or learnt from example datasets [2,6]. Existing rule-based techniques are able to concentrate only on a subset of the above mentioned clues to limit the complexity of the learnt rules. They provide high precision segmentation in uniform settings where the amount of noise is limited. When the input becomes noisy, rule-based systems tend to lose on recall.

Hidden Markov Models

Hidden Markov Models (HMMs) provide an intuitive statistical method for combining many of the above clues in a unified model. A HMM is a probabilistic finite state automata where the states represent the fields to be extracted, directed edges between edges are attached with probability values indicating

probability of transitioning from one state to another, and states are attached with a distribution over the words that can be generated from the state. A segmentation of a string S is achieved by finding the sequence of states for which the product of the probability of generating the words in S and following the transitions in state sequences is maximized. Such a path can be found efficiently using a dynamic programming algorithm. The parameters controlling the transition and word distributions of states are learnt using examples of correctly segmented strings.

An example, of a Hidden Markov Model trained to recognize Indian addresses appears in Fig. 1. The number of states is 10 and the edge labels depict the state transition probabilities. For example, the probability of an address beginning with House Number is 0.92 and that of seeing a City after Road is 0.22. The dictionary and the emission probabilities are not shown for compactness.

For more details on the use of HMMs in column segmentation see [1,3,11].

Conditional Models

A limitation of HMMs is that the distribution that controls the generation of words within a state is generative, and can therefore capture only a limited set of properties of the words it can generate. For example, it is complicated to account for various orthographic properties of words, like its capitalization pattern, or the delimiter following that word. These limitations are removed by recently proposed formalisms like Conditional Random Fields (CRFs) that capture the conditional distribution of column sequence *given* the sequence of words in a string S. This enables the incorporation of any arbitrary set of clues derived from a word and the words in its neighborhood. Also it becomes easy to incorporate clues derived from the degree of match of a proposed column with pre-existing values in the database.

A CRF models the conditional probability distribution over segmentations $\mathbf{s}$ for a given input sequence $\mathbf{x}$ as follows:

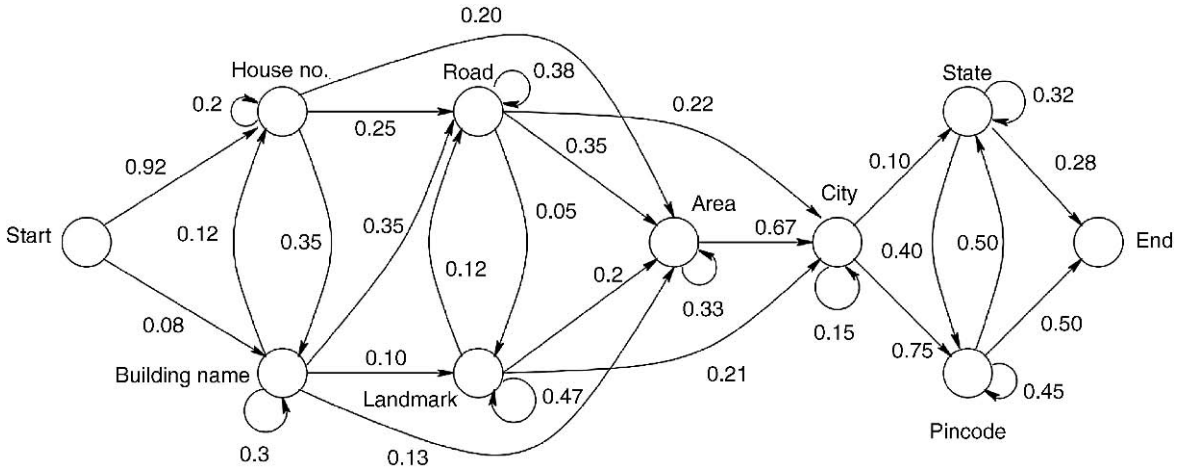
$$\Pr(\mathbf{s}|\mathbf{x},\mathbf{W}) = \frac{1}{Z(\mathrm{x})}\exp\left(\mathbf{W}.\sum_{\mathrm{j}}\mathbf{f}(j,\mathbf{x},\mathbf{s})\right) \qquad (1)$$

where $\mathbf{f}(j,\mathbf{x},\mathbf{s})$ is a vector of local feature functions $f_1 \ldots f_N$ of $\mathbf{s}$ at the jth segment and $\mathbf{W} = (W_1, W_2, \ldots, W_N)$ is a weight vector that encodes the importance of each feature function in $\mathbf{f}$. $Z(\mathbf{x}) = \sum_{\mathbf{s}'} \exp(\mathbf{W} \cdot \sum_{\mathrm{j}} \mathbf{f}(j, \mathbf{x}, \mathbf{s}'))$ is a normalization factor. The label of a segment depends on the label of the previous segment and the properties of the tokens comprising this segment and the neighboring tokens. Thus a feature for segment $s_j = (t_j, u_j, y_j)$ is a function of the form $f(y_j, y_{j-1}, \mathbf{x}, t_j, u_j)$ that returns a numeric value. Example of such features are:

$$f_8(y_i, y_{i-1}, \mathbf{x}, 3, 5) = [\![x_3 x_4 x_5 \text{ appears in a journal list}]\!].[\![y_i = \text{journal}]\!]$$

$$f_{12}(y_i, y_{i-1}, \mathbf{x}, 19, 19) = [\![x_{19} \text{ is an integer}]\!].[\![y_i = \text{year}]\!].[\![y_{i-1} = \text{month}]\!]$$

The weight vector $\mathbf{W}$ is learnt during training via a variety of methods, such as likelihood maximization [7]. During *segmentation*, the goal is to find a



Column Segmentation. Figure 1. An example of trained HMM for segmenting addresses.

$\mathbf{s} = s_1 \ldots s_p$ for the input sequence $\mathbf{x} = x_1 \ldots x_n$ such that $\Pr(\mathbf{s}|\mathbf{x}, \mathbf{W})$ (as defined by (1) is maximized.

$$\arg\max_{\mathbf{s}} \Pr(\mathbf{s}|\mathbf{x}, \mathbf{W}) = \arg\max_{\mathbf{s}} \mathbf{W}. \sum_j \mathbf{f}(y_i, y_{j-1}, \mathbf{x}, t_j, u_j)$$

The right hand side can be efficiently computed using dynamic programming. Let L be an upper bound on segment length. Let $\mathbf{s}_{i:y}$ denote set of all partial segmentation starting from 1 (the first index of the sequence) to i, such that the last segment has the label y and ending position i. Let $V(i, y)$ denote the largest value of $\mathbf{W} \cdot \sum_j \mathbf{f}(j, \mathbf{x}, \mathbf{s}')$ for any $\mathbf{s}' \in \mathbf{s}_{i:y}$. The following recursive calculation finds the best segmentation:

$$V(i,y) = \begin{cases} \max_{y', i'=i-L \ldots i-1} V(i', y') & \text{if } i > 0 \\ \quad + \mathbf{W} \cdot \mathbf{f}(y, y', \mathbf{x}, i'+1, i) & \\ 0 & \text{if } i = 0 \\ -\infty & \text{if } i < 0 \end{cases}$$

The best segmentation then corresponds to the path traced by $\max_y V(|\mathbf{x}|, y)$.

More details on CRFs can be found in [7] and the extension of CRFs for segmentation can be found in [10]. [8] reports an empirical evaluation of CRFs with HMMs for segmenting paper citations. [4] shows how to perform efficient segmentation using CRFs in the presence of a large pre-existing database of known values.

Key Applications

Column segmentation has many applications, including,

Cleaning of text fields during warehouse construction: In operational datasets, text fields like addresses are often recorded as single strings. When warehousing such datasets for decision support, it is often useful to identify structured elements of the address. This not only allows for richer structured queries, it also serves as a useful pre-processing step for duplicate elimination.

Creation of citation databases: A key step in the creation of citation databases like Citeseer and Google Scholar, is to resolve for each citation, which paper it refers to in the database. Citations as extracted from papers are unstructured text strings. These have to be segmented into component author names, titles, years, and publication venue before they can be correctly resolved to a paper entry in the database.

Extraction of product information from product descriptions: Comparison shopping websites often need to parse structured fields representing various attributes of product from unstructured HTML sources.

URL to Code

Java packages for column segmentation using conditional random fields are available via Source Forge at http://crf.sf.net and as part of the Mallet package at http://mallet.cs.umass.edu

Cross-references

▶ Data Cleaning

Recommended Reading

1. Agichtein E. and Ganti V. Mining reference tables for automatic text segmentation. In Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2004, pp. 20–29.
2. Aldelberg B. Nodose: a tool for semi-automatically extracting structured and semi-structured data from text documents. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 283–294.
3. Borkar V.R., Deshmukh K., and Sarawagi S. Automatic text segmentation for extracting structured records. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 175–186.
4. Chandel A., Nagesh P.C., and Sarawagi S. Efficient batch top-k search for dictionary-based entity recognition. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
5. Cunningham H. Information Extraction, Automatic. Encyclopedia of Language and Linguistics, 2nd edn., 2005.
6. Kushmerick N., Weld D.S., and Doorenbos R. Wrapper induction for information extraction. In Proc. 15th Int. Joint Conf. on AI., 1997, pp. 729–737.
7. Lafferty J., McCallum A., and Pereira F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th Int. Conf. on Machine Learning, 2001, pp. 282–289.
8. Peng F. and McCallum A. Accurate information extraction from research papers using conditional random fields. In HLT-NAACL. 2004, pp. 329–336.
9. Ratnaparkhi A. Learning to parse natural language with maximum entropy models. Mach. Learn., 34, 1999.
10. Sarawagi S. and Cohen W.W. Semi-markov conditional random fields for information extraction. In Advances in Neural Inf. Proc. Syst. 17, 2004.
11. Seymore K., McCallum A., and Rosenfeld R. Learning Hidden Markov Model structure for information extraction. In Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction. 1999, pp. 37–42.

Committee-based Learning

▶ Ensemble

Common Object Request Broker Architecture

► CORBA

Common Subexpression Elimination

► Multi-Query Optimization

Common Warehouse Metadata Interchange (CWMI)

► Common Warehouse Metamodel (CWM™)

Common Warehouse Metamodel

Liam Peyton
University of Ottawa, Ottawa, ON, Canada

Synonyms

Common Warehouse Metadata Interchange (CWMI); CWM

Definition

The Common Warehouse Metamodel (CWM™) is an adopted specification from the OMG (Object Management Group) standards body. It defines standard interfaces that can be used to enable easy interchange of data warehouse and business intelligence metadata between data warehouse tools, data warehouse platforms and data warehouse metadata repositories in distributed heterogeneous environments. It supports relational, non-relational, multi-dimensional, and most other objects found in a data warehousing environment. It leverages three other standards from OMG:

- UML – Unified Modeling Language
- MOF – Meta Object Facility
- XMI – XML Metadata Interchange

The Object Management Group has been an international, open membership, not-for-profit computer industry consortium since 1989 with over 700 member organizations.

Historical Background

An initial Request For Proposal (RFP) for a common warehouse metadata interchange (CWMI) was issued by the OMG (Object Management Group) in 1998. A joint submission was received by the OMG in 1999 from Dimension EDI, Genesis Development Corporation, Hyperion Solutions, International Business Machines, NCR, Oracle, UBS AG, and Unisys.

At the time, there was a competing initiative from the Meta Data Coalition (MDC) which was supported by Microsoft and others. In 2000, however the two initiatives merged when the MDC joined OMG [5]. In 2001, version 1.0 of the specification was adopted with the name: Common Warehouse Metamodel (CWM™). The currently adopted version is 1.1 [1] (March, 2003).

The purpose of the Common Warehouse Metamodel specification was to make it possible for large organizations to have a metadata repository with a single metamodel. In practice this was not possible to achieve, since every data management and analysis tool requires different metadata and different metadata models [3]. Instead, the CWM specification defines interfaces that facilitate the interchange of data warehouse metadata between tools. In particular, the OMG Meta-Object Facility (MOF™) bridges the gap between dissimilar meta-models by providing a common basis for meta-models. If two different meta-models are both MOF-conformant, then models based on them can reside in the same repository.

However, compliance with the CWM specification does not guarantee tools from different vendors will integrate well, even when they are "CWM-compliant." The OMG addressed some of these issues by releasing patterns and best practices to correct these problems in a supplementary specification, the Common Warehouse Metamodel (CWM™)) Metadata Interchange Patterns (MIP) Specification. Version 1.0 [2] was released in March 2004.

Foundations

The Common Warehouse Metamodel enables organizations and tool vendors to define and represent their metadata, metadata models and the processes which manipulate them in a common format so that the information can be streamed between tools and accessed programmatically [4].

The basic architecture and key technologies supporting the Common Warehouse Metamodel are shown in Fig. 1, on the next page. Metadata in a variety of

formats, and from a variety of sources (Tools, Repositories, Databases, Files, etc.) is defined and represented in UML notation, based on the objects and classes that are defined in the Common Warehouse Metamodel. That representation is persisted in an XML notation that can be streamed to other tools, repositories, databases or files based on the XMI protocol. Finally, MOF is used to provide a broker facility that supports the ability to define and manipulate metamodels programmatically using fine grained CORBA interfaces. Using this architecture, organizations can create a single common repository which stores all the CWM-modeled descriptions of metadata and metamodels.

An example of a CWM description of a table from a relational database is shown below, along with the metadata description of the type of one of its columns (type="22");

```
<CWMRDB:Table xmi.id="_15" name="MyTableName">
 <CWM:Classifier.feature>
  <CWMRDB:Column xmi.id="_16" name="myPrimaryKeyID" precision="4" type="_17"/>
  <CWMRDB:Column xmi.id="_18" name="myForeignKey1ID" precision="4" type="_17"/>
  <CWMRDB:Column xmi.id="_19" name="myForeignKey2ID" precision="4" type="_17"/>
  <CWMRDB:Column xmi.id="_20" name=" myForeignKey3ID" precision="4" type="_17"/>
  <CWMRDB:Column xmi.id="_21" name="description" length="200" type="_22" />
 </CWM:Classifier.feature>
 <CWM:Namespace.ownedElement>
  <CWMRDB:ForeignKey xmi.id="_23" name="unnamed_23" namespace="_15" feature="_19" uniqueKey="_24"/>
 </CWM:Namespace.ownedElement>
</CWMRDB:Table>
<CWMRDB:SQLSimpleType
  xmi.id="_22"
  name="VARCHAR2"
  visibility="public"characterMaximumLength="200"
  characterOctetLength="1" type
  Number="12"/>
```

The CWM specifications consists of a collection of metamodels (defined in UML) that capture all the elements of metadata, metamodels, and their processing that can be expressed when exchanging information between tools. These are what is identified in Fig. 1 as the Common Warehouse Metamodel. It is organized into five layers of abstraction.

Object Model

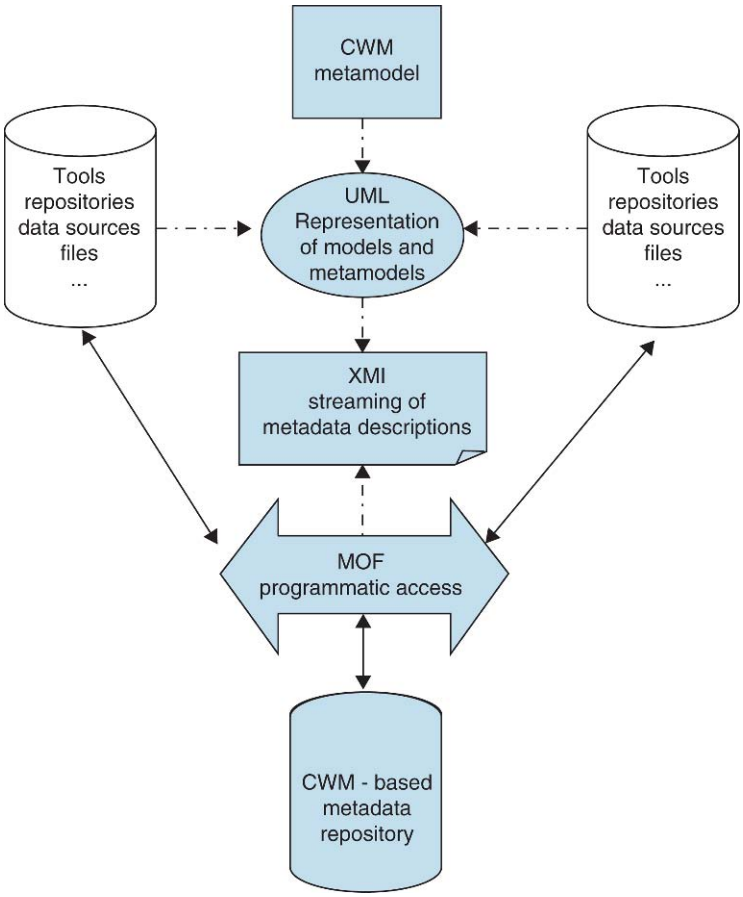
The Object Model layer is the base layer of the Common Warehouse Metamodel. The metamodels in the Object Model layer define the subset of UML that is used for creating and describing the CWM. They are the building blocks used by all the metamodels in the upper layers. This enables CWM to leverage UML's concepts without requiring implementations to support all full of UML's capabilities.

- Core
  The Core metamodel contains basic classes and associations used by all other CWM metamodels like Namespace, Constraint, Attribute, ModeledElement etc.
- Behavioral
  The Behavioral metamodel describe the behavior of CWM types and how that behavior is invoked with classes like Event, Parameter, CallAction etc.
- Relationships
  The Relationships metamodel describes two types of relationships between object within a CWM information store: generalizations (for parent-child relationships) and associations (for links between objects).
- Instance
  The Instance metamodel contains classes to support the inclusion of data instances with the metadata.

Foundation

The metamodels in the Foundation layer contain general model elements that represent concepts and structures shared by other CWM packages. Metamodels in this layer are not necessarily complete, but serve as a common basis that can be shared with other metamodels.

- Data Types
  The DataTypes metamodel supports definition of metamodel constructs that modelers can use to create the specific data types they need with classes like Enumeration, Union, EnumerationLiteral, UnionMember, etc.
- Expression
  The Expressions metamodel supports the definition of expression trees.
- Keys and Indexes
  This metamodel defines the basic concepts of Index, IndexedFeature, UniqueKey, and KeyRelationship.



Common Warehouse Metamodel. Figure 1. Common warehouse metamodel architecture.

- Type Mapping
  This metamodel is used to support the mapping of types between different tools or data sources.
- Business Information
  The Business Information metamodel supports business-oriented information about model elements with classes like ResponsibleParty, Contact, ResourceLocater etc.
- Software Deployment
  The Software Deployment metamodel contains classes like SoftwareSystem, Component, Site to record how the software in a data warehouse is used.

Resource

The metamodels in the resource layer define the type of data sources and formats that are supported.

- Relational
  The Relational metamodel describes relational data this is accessed through an interface like ODBC, JDBC or the native interface of a relational database.
- Record
  The Record metamodel describes the concept of a record and its structure that can be applied to data records stored in files and databases, or to programming language structured data types.
- Multi-Dimensional
  The Multi-Dimensional metamodel describes a generic representation of a multidimensional database using classes like Schema, Dimension, Member etc.
- XML
  The XML metamodel describes the metadata of XML data with classes like ElementType, Attribute etc.

Analysis

The metamodels in the analysis layer define the types of interaction with metadata that are supported.

- Transformation
  The Transformation metamodel contains classes to describe common transformation metadata used

in Extract, Transform, Load (ETL) tools and processes.

- OLAP
  The OLAP metamodel contains classes to describe common analysis metadata using in OLAP processing with classes like MemberSelection and CubeDeployment.
- Data Mining
  The Data Mining metamodel provide the necessary abstractions to model generic representations of both data mining tasks and models (i.e., mathematical models produced or generated by the execution of data mining algorithms).
- Information Visualization
  The Information Visualization metamodel contains classes like Rendering, RenderedObject, to describe metadata associated with the display of data.
- Business Nomenclature
  The Business Nomenclature metamodel supports the definition of terms used in capturing business requirements with classes like Nomenclature, BusinesDomain, Taxonomy, Glossary, Term, Concept, etc.

Management

The metamodels in the management layer define two aspects of warehouse management.

- Warehouse Process
  The Warehouse Process metamodel supports the documentation of process flows used to execute transformations. A process flow can associate a transformation with a set of events, which will be used to trigger the execution of the transformation.
- Warehouse Operation
  The Warehouse Operation metamodel contains classes for the day-to-day operation and maintenance of the warehouse including scheduled activities, measurements, and change requests.

Key Applications

Vendors of data warehouse tools, conform to the CWM specification to ensure that the metadata in their tools is open and accessible to any CWM compliant tool.

Large organizations leverage the specification in order to be able to manage and maintain there data warehouses in a common metadata repository. By using the CWM specification IT administrators and system integrators can extract and link metadata from different vendors tools.

Oracle, IBM, SA, Informatica, Meta Integration Technology Incorporated are among several industry leaders who have data warehouse tools that are CWM compliant to facilitate interoperability.

Cross-references

- ▶ Data Warehouse Metadata
- ▶ Metadata
- ▶ Metadata Interchange Specification
- ▶ Metadata Registry, ISO/IEC 11179
- ▶ Meta Object Facility
- ▶ Metamodel
- ▶ Unified Modelling Language
- ▶ XML Metadata Interchange

Recommended Reading

1. Common Warehouse Model (CWM) Specification, Version 1.1, Object Management Group. Needham, MA, March 2, 2003. http://www.omg.org/technology/documents/formal/cwm.htm.
2. CWM Metadata Interchange Patterns Specification, Version 1.0, Object Management Group. Needham, MA, March 25, 2004.
3. Grossman R.L., Hornick M.F., and Meyer G. Data mining standards initiatives. Commun. ACM., 45(8):59–61, 2002.
4. Poole J., Chang D., Tolbert D., and Mellor D. Common Warehouse Metamodel: An Introduction to the Standard for Data Warehouse Integration. Wiley, 2002.
5. Vaduva A. and Dittrich K.R. Metadata Management for Data Warehousing: Between Vision and Reality. In Proc. Int. Conf. on Database Eng. and Applications, 2001, p. 0129.

Communication Boundary of a DBMS

- ▶ DBMS Interface

Compact Suffix Tries

- ▶ Suffix Trees

Comparative Analysis

- ▶ Comparative Visualization

Comparative Visualization

Hans Hinterberger
ETH Zurich, Zurich, Switzerland

Synonyms

Comparative analysis

Definition

Comparative visualization refers to

1. Methods that support the process of understanding in what way different datasets are similar or different.
2. Methods that allow comparing different characteristics of a given dataset.
3. Methods that allow a comparison of different types of (linked) data graphics.

Key Points

Comparisons of datasets may occur in different ways. Data value to data value: entries of different datasets are compared to one another based on their values; derived quantity to derived quantity: these could be statistical moments of data fields or topological characteristics; methodology to methodology: comparisons of methodologies involve quantifying differences in experiment or simulation parameters; and, if the data are visualized, image to image: such comparisons quantify the differences in the visualizations produced by a given graphical method.

Comparative visualization methods have been developed as an enabling technology for computational and experimental scientists whose ability to collect and generate data far outpaces their ability to analyze and understand such data. The Visualization and Analysis Center for Enabling Technologies (http://www.vacet.org), for example, provides (publicly available) comparative data visualization software [1] for the scientists at the various research labs associated with the US Department of Energy.

Graphical displays that readily allow simultaneous comparisons of several characteristics of multivariate datasets – the parallel coordinate display for example – are sometimes referred to as "comparative graphs."

There is evidence that visual explorations into a dataset's structure are particularly effective when the data can be compared by simultaneously observing different visualizations of the same data. Today's data visualization packages routinely include several different graphic methods to allow such comparisons. To be truly effective, the different graphics should be operationally linked. See [2] and [3] for two examples among others.

Cross-references

- ▶ Exploratory Data Analysis
- ▶ Parallel Coordinates

Recommended Reading

1. Bavoil L., Callahan S.P., Crossno P.J., Freire J., Scheidegger C.E., Silva C.T., and Vo H.T. VisTrails: enabling interactive multiple-view visualizations. In Proc. IEEE Visualization, 2005.
2. Schmid C. and Hinterberger H. Comparative multivariate visualization across conceptually different graphic displays. In Proc. 7th Int. Working Conf. on Scientific and Statistical Database Management, 1994.
3. Siirtola H. Combining parallel coordinates with the reorderable matrix. In Proc. Int. Conf. on Coordinated & Multiple Views in Exploratory Visualization, 2003.

Compensating Transactions

Greg Speegle
Baylor University, Waco, TX, USA

Definition

Given a transaction T, and its compensating transaction C, then for any set of transactions H executing concurrently with T, the database state D resulting from executing THC is equivalent to the database state D' resulting from executing H alone. Typically, equivalent means both D and D' satisfy all database consistency constraints, but D and D' do not have to be identical.

A compensating transaction is defined in terms of its corresponding failed transaction, and once started, must be completed. This may involve re-executing the compensating transaction multiple times. The result of compensation is application dependent.

Key Points

A *compensating transaction* is a set of database operations that perform a logical undo of a failed transaction. The goal of the compensating transaction is to restore any database consistency constraints violated

by the failed transaction without adversely affecting other concurrent transactions (e.g., *cascading aborts*). However, it does not require the database to be in the exact same state as if the transaction had never executed as with traditional *ACID* properties. A compensating transaction also removes the externalized affects of a failed transaction [2].

Compensating transactions can best be understood by comparing them to traditional atomicity requirements. Under traditional atomicity, either all effects of a transaction are present in the database, or none of them are. Thus, if a transaction T_1 updates a data item and transactions T_2 reads that update, in order to remove all of the effects of T_1, T_2 must also be removed. With compensating transactions, the abort of T_2 is not be required.

Consider an example application of a company manufacturing widgets. The transaction for buying widgets consists of two subtransactions, one to order the widgets and another to pay for them. Since this business is very efficient, as soon as the widgets are ordered, another transaction starts executing the desired widgets. It is possible to compensate for the ordered widgets by simply removing the order from the system. The extra widgets would be produced, but they will be consumed by later orders. Under traditional atomicity requirements, the production transaction would have to be aborted if the buying transaction failed after the order was placed (for example, if the customer could not pay for the widgets).

Compensating transactions are used in long duration transactions called *Sagas* [1], and other applications that require *semantic atomicity*. Unfortunately, compensation is not universally possible – the common example of an externalized event that cannot be undone is the launching of a missile – or may be very complex. Thus, compensating transactions are used when the benefits of avoiding cascading aborts and early externalization of results outweigh the difficulty in determining the compensation.

Cross-references

- ▶ ACID Properties
- ▶ Sagas
- ▶ Semantic Atomicity

Recommended Reading

1. Garcia-Molina H. and Salem K. SAGAS. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 249–259.
2. Korth H.F., Levy E., and Silberschatz A. A formal approach of recovery by compensating transactions. In Proc. 16th Int. Conf. on Very Large Data Bases, 1990, pp. 95–106.

Computationally Complete Relational Query Languages

Victor Vianu[1], Dirk Van Gucht[2]
[1]University of California-San Diego, La Jolla, CA, USA
[2]Indiana University, Bloomington, IN, USA

Synonyms

Complete query languages; Chandra and Harel complete query languages

Definition

A *relational query language* (or query language) is a set of expressions (or programs). The semantics of a query language defines for each of these expressions a corresponding query which is a generic, computable function from finite relation instances to finite relation instances over fixed schemas. A query language is *computationally complete* (or complete) if it defines all queries.

The genericity condition is a consistency criterion requiring that a query commute with isomorphisms of the database domain. Thus, when applied to isomorphic input relation instances, a query returns isomorphic output relation instances. The concept of genericity is based on the well-accepted idea that the result of a query should be independent of the representation of data in a database, and should treat the elements of the database as uninterpreted objects [4]. The computability condition requires that the query can be effectively computed, in other words it must be *implementable* by a program of a Turing-complete programming language under some suitable encoding of relation instances into objects of that language.

Historical Background

The search for an appropriate notion of "complete" query language began soon after the introduction of the relational model by Codd, with its accompanying query languages relational algebra (RA) and relational calculus (RC) [9]. Initially, RA was proposed as a yardstick for query expressiveness. A language was called by Codd "relationally complete" if it was able

to simulate RA [10]. Bancilhon and Paredaens independently proposed the notion of BP-completeness of a language, using an instance-based approach: a language is BP-complete if for every pair of input and output instances satisfying a consistency criterion (The criterion requires that every automorphism of the input be also an automorphism of the output) there exists a query in the language mapping the input instance to the output instance [5,13]. The notion of genericity was first articulated in the database context by Aho and Ullman [4], although its roots can already be found in the consistency criterion used in the definition of BP-completeness, and an idea similar to genericity underlies Tarski's concept of "logical notion," introduced in a series of lectures in the mid 1960s [15]. The modern notion of computationally complete query language is due to Chandra and Harel, who also defined the first such language, QL [7].

Foundations

Codd introduced the relational model and its query languages, relational algebra (RA) and relational calculus (RC). These query languages are equivalent, i.e., they define the same set of queries. For example, assuming that **R** and **S** are relation schemas both of arity 2, then the RA-expression $\pi_{1,4}(\sigma_{2=3}(\mathbf{R} \times \mathbf{S})) \cup (\mathbf{R} - \mathbf{S})$, and the RC-expression $\{(x, y) \mid \exists z : (\mathbf{R}(x, z) \wedge \mathbf{S}(z, y)) \vee (\mathbf{R}(x, y) \wedge \neg \mathbf{S}(x, y))\}$ define the same computable query Q which maps each relation instance R over **R** and each relation instance S over **S** to their join unioned with their set difference. Notice that Q is also generic: consider, for example, the input relation instances

$$R_1 = \boxed{\begin{array}{cc} a & b \\ a & c \end{array}} \quad S_1 = \boxed{\begin{array}{cc} a & b \\ b & d \end{array}}$$

and the isomorphic input relation instances

$$R_2 = \boxed{\begin{array}{cc} e & f \\ e & g \end{array}} \quad S_2 = \boxed{\begin{array}{cc} e & f \\ f & h \end{array}}$$

then $Q(R_1, S_1)$ and $Q(R_2, S_2)$ are the isomorphic output relation instances $\boxed{\begin{array}{cc} a & c \end{array}}$ and $\boxed{\begin{array}{cc} e & g \end{array}}$, respectively. (As a caveat to genericity, consider the query C defined by the RA-expression $\sigma_{1=\mathbf{a}}(\mathbf{R})$, where **a** is some *constant* interpreted as a. Then, $C(R_1) = R_1$, but $C(R_2) = \emptyset$. The difficulty is that, though R_1 and R_2 are isomorphic, they are not isomorphic by an isomorphism that *fixes* a. If however, the value of e in R_2 is replaced by a, then $C(R_2{}^{e \leftarrow a}) = R_2{}^{e \leftarrow a}$. This suggests that when constants are involved, genericity should be modified to isomorphisms that fix these constants. In the literature, this is referred to as **C**-genericity.)

The development of the relational model led to the introduction of the query language SQL which has, at its logical core, a sub-language pure-SQL, that is equivalent with RA and RC. For example, in pure-SQL, the query Q can be defined by the expression (Here A, B, C, and D are attribute names referring to the first and second columns of **R**, and the first and second columns of **S**, respectively.)

```
(SELECT R.A, S.D AS B FROM R, S WHERE R.B = S.C)
  UNION
  (
  (SELECT R.A, R.B FROM R)
  EXCEPT
  (SELECT S.C AS A, S.D AS B FROM S)
  );
```

A natural question is now "Are RA, RC, and pure-SQL complete query languages?" The answer is no. To this end, consider the following three queries, which are easily seen to be generic and computable:

1. TC maps each binary relation instance R to its transitive closure R^*.
2. EVEN maps each unary relation instance R to $\{()\}$ (true) if $|R|$ is even, and to $\emptyset$ (false), otherwise.
3. EVEN$^<$ maps each pair of a unary relation instance R and a binary relation instance O to EVEN(R) if O defines an ordering on $dom(R)$, and is undefined otherwise. (Here, $dom(R)$ denotes the set of values that occur in the tuples of R.)

It turns out that none of the above queries is expressible in RA (or RC, or pure-SQL). Consider (1). Fagin showed in [11] that the TC query cannot be defined by any RC expression (the result was later re-proven for RA by Aho and Ullman [4]). Intuitively, the difficulty in computing TC is the following. For each $i \geq 0$, there exists an RA-expression $\mathbf{E}^i$ that defines the pairs of elements in R at distance i in the directed graph represented by R. However, there does not exist a *single* RA-expression that defines the union of all these pairs, as needed for computing TC. A solution to this problem is to augment RA with an *iteration* construct. This led Chandra and Harel to define the language While (initially introduced as RQ in [8] and LE in [6]). The language uses, in addition to database relations, typed relational variables (of fixed arity)

initialized to $\emptyset$, to which RA expressions can be assigned. Iteration is provided by a construct "while change to **R** do ⟨program⟩ od" whose semantics is to iterate ⟨*program*⟩ as long as the value of the relational variable **R** changes. For example, the following While program defines TC:

TC := **R**;
while change to **TC**
do **TC** := **TC** $\cup$ $\pi_{1,4}(\sigma_{3=4}(\mathbf{TC} \times \mathbf{R}))$ od.

Here **TC** is a binary relation variable (initialized to $\emptyset$). The program first assigns R to **TC**, then loops as long as **TC** changes. Upon termination, this value is R^*. One might hope that While is a complete query language. However, this is not the case. Indeed, even though it is easy to write a While program that defines the EVEN$^<$ query, Chandra showed that no such program can define the simple linear-time computable EVEN query [6]. Thus, While is not computationally complete. Intuitively, While programs do not have the ability to compute with natural numbers, unless such computations can be simulated by utilizing an ordering on the elements of its input. With such orderings available, While can define precisely the PSPACE-computable queries [17]. The PSPACE upper bound (that holds with or without order) is a consequence of the fact that a program's finite set of variables are of fixed arity and can only hold relation instances built from the elements of its inputs.

To overcome these problems, it appears natural to embed RA into a language that can perform arbitrarily powerful computations. This is in the spirit of "embedded SQL" languages, in which a computationally complete programming language such as C or Java accesses the database using SQL queries. A language called LC (for *Looping+Counters*), abstracting the "embedded SQL" paradigm, was introduced by Chandra [6] (with a variant called WhileN later defined in [1]). The language LC extends While by allowing integer variables (initialized to zero) that can be incremented or decremented. Iteration for computation on integers is provided by an additional while loop of the form "while i $>$ 0 do ⟨program⟩ od" which causes ⟨program⟩ to iterate as long as the value of the integer variable i is positive. For example, consider the following program in an LC-like syntax: (The "if-else" statement is a macro that can be easily written using just the "while-change" construct.)

TC := **R**;
$n := 0$;
while change to **TC**
do
$n := n + 1$;
TC := **TC** $\cup$ $\pi_{1,4}(\sigma_{2=3}(\mathbf{TC} \times \mathbf{R}))$
od;
if $n \leq 1$ return $\{()\}$ else return $\emptyset$

At the end of the computation, n contains the number of times the body of the *while* loop was executed. Thus, if R is non-empty, the final value of n is the diameter of the directed graph represented by R (here the diameter means the maximum finite distance between two nodes). The program returns $\{()\}$ (true) if n $\leq$ 1 and $\emptyset$ (false) otherwise. Note that, since LC is computationally complete on the integers, the condition "n $\leq$ 1" could be replaced by *any* computable property of n. Thus, LC can test any computable property of the diameter of R. Clearly, LC can define strictly more queries than While, since all computable functions on natural numbers can be defined and used. This leads to the next question: "Is LC a complete query language?" Again, the answer is no. Indeed, Chandra showed that the EVEN query can still not be defined in this language. This time, the difficulty stems from the fact that, even though the values of the relation and natural number variables can depend on each other, LC programs lack the ability to *explicitly coerce* (encode) these values into each other. However, when input relation instances are accompanied by an ordering of the domain, such coercions can be simulated, and Abiteboul and Vianu showed that then LC *is* complete [1].

To obtain a complete language without order, several solutions are possible. A brute force approach to the coercion problem is to augment LC with an encoding function *enc* mapping relations to integers, and a decoding function *dec* returning query answers from their encodings and the original input database. Since LC is computationally complete on integers, it can compute the integer encoding of the answer from that of the input. Although this theoretically produces a complete language, manipulating integer encodings of databases is not a satisfying solution, so further discussion of this approach is omitted. Instead, two more appealing alternatives are described, that both go back to While as a starting point and extend it in different ways. Recall that While is limited to PSPACE computations, as a

consequence of two facts: (i) only relations of fixed arity are used, and (ii) the relations can be populated by tuples using only elements occurring in the input. The first approach, proposed by Chandra and Harel, breaks the PSPACE space barrier by relaxing (i) it allows untyped relational variables, whose arity can grow arbitrarily. The other approach, introduced by Abiteboul and Vianu, relaxes (ii) it keeps typed relational variables but allows the introduction of new domain values in the course of the computation. These languages are described next.

The complete language proposed by Chandra and Harel was called QL [6]. Up to minor syntactic differences, QL is very similar to While, only with untyped relation variables. Consider the following QL program (For simplicity, the syntax used here differs slightly from the original QL syntax.) which is strikingly similar to the LC program shown above:

$\mathbf{TC} := \mathbf{R}$;
$\mathbf{ONE} := \pi_1(\mathbf{R}) \cup \pi_2(\mathbf{R})$;
$\mathbf{N} := \{()\}$;
while change to **TC**
  do
    $\mathbf{N} := \mathbf{N} \times \mathbf{ONE}$;
    $\mathbf{TC} := \mathbf{TC} \cup \pi_{1,4}(\sigma_{2=3}(\mathbf{TC} \times \mathbf{R}))$
  od;
if $(\mathbf{N} = \{()\})$ or $(\mathbf{N} = \mathbf{ONE})$ then return $\{()\}$ else return $\emptyset$.

In this program, **R** is a binary relation input variable, and **TC**, **ONE**, and **N** are relation variables. Note that the arities of **TC** and **ONE** remain fixed throughout the execution of the program, while the arity of **N** changes. Integers can be easily simulated using the arity of relations. Thus, starting with relation instance R, the variable **ONE** is initialized to dom(R), which plays the role of the natural number 1. The variable **N** plays the role of a natural number variable n. The statement $\mathbf{N} := \{()\}$ corresponds the statement n = 0, and the statement $\mathbf{N} := \mathbf{N} \times \mathbf{ONE}$ serves to increment n by 1; notice that the $\times$ operator plays the role of the addition operator + over natural numbers, and the decrement operator can be simulated by projection. Similarly to the earlier LC program, the final arity of **N** is the diameter of the directed graph represented by **R**. The final "if" statement compares **N** to $\{()\}$ or **ONE**, and the program returns $\{()\}$ (true) if the diameter of **R** is at most 1, and $\emptyset$ (false) otherwise. Observe therefore that this QL program defines the same query as its corresponding LC program.

The above example illustrates how arithmetic on natural numbers can be simulated in QL. So far, this allows simulating LC. Recall that LC is not complete, but becomes so if an ordering of the domain is provided. QL is however complete even if an ordering is not provided, because it can construct its own orderings! Indeed, such orderings of the domain can simply be constructed in QL by building one relation whose arity equals the size of the input domain. In such a relation, any tuple that does not contain repeated elements provides a successor relation on the domain, which in turns induces an ordering. The completeness of QL now follows from the completeness of LC on ordered domains. Thus, QL can express all computable queries. But is everything it expresses a query? It is easy to see that all mappings defined by QL programs are computable and generic. The difficulty is to guarantee that a QL program always produces answers of the desired arity. In fact, this property is undecidable for QL programs. Fortunately, there is an effective syntactic restriction guaranteeing that QL programs are "well behaved," i.e., always produce answers of fixed arity. Moreover, all computable queries can be expressed by QL programs satisfying the syntactic restriction.

The language WhileNew, introduced by Abiteboul and Vianu in [3], extends While by allowing the creation of new values throughout the computation. This is achieved by an instruction of the form $\mathbf{S} := \text{new}(\mathbf{R})$, where **R** and **S** are relational variables and $\text{arity}(\mathbf{S}) = \textit{arity}(\mathbf{R}) + 1$. The semantics is the following. Given a relation instance R over **R**, the relation instance S over **S** is obtained by extending each tuple of R by one distinct new value not occurring in the input, the current state, or in the program. For example, if R is the relation instance in Fig. 1 then S is of the form shown in the same figure. The values α,β,γ are distinct new values. Note that the new construct is, strictly speaking, nondeterministic. Indeed, the new values are arbitrary, so several outcomes are possible depending on the choice of values. However, the different outcomes differ *only* in the choice of new values.

The ability to successively introduce new values throughout the computation easily allows simulating integers and arithmetic, yielding the power of LC. Moreover, orderings of the input domain can also be constructed and marked by distinct new values. Since LC is complete on ordered domains, this shows that

WhileNew can express all computable queries. As in the case of QL, one must ask whether *all* mappings expressed by WhileNew are in fact queries according to the definition. The difficulty arises from the presence of new values. Indeed, if new values may appear in the outputs of a WhileNew program, the mapping it defines is non-deterministic. Moreover, it is undecidable whether a WhileNew program never contains new values in its output. The solution to this problem is similar to the one for QL: one can impose a syntactic restriction on WhileNew programs guaranteeing that no new value appears in their answers. All generic computable queries can be expressed by WhileNew programs satisfying the syntactic restriction.

As an aside, suppose the definition of query is extended by allowing new values in query answers. This arises naturally in some contexts such as object-oriented databases, where outputs to queries may contain new objects with their own fresh identifiers. One might hope that WhileNew remains complete for this extension. Surprisingly, it was shown by Abiteboul and Kanellakis that the answer is negative [2]. Indeed, WhileNew cannot express the query containing the input/output pair shown in Fig. 2, where $\psi_0,...,\psi_3$ are new values. As shown by Abiteboul and Kanellakis, completeness with new values can be achieved by adding to WhileNew a construct called *duplicate elimination*. This is however a rather complex construct, that encapsulates a test for isomorphism of relations. The search for a language using more natural primitives and complete for queries with new values in the answer remains open.

The relational model, though very simple, is not always the most natural model for databases in certain application domains. In the late 1980s and early 1990s database researchers considered object-oriented databases as an alternative to the relational model, and a significant amount of theory was developed around the model and its query languages, including the completeness of object-oriented query languages (see [1,16]). Finally, consistency notions other than genericity can be considered for specialized application domains. This was done, for example, in the context of spatial databases [12,14].

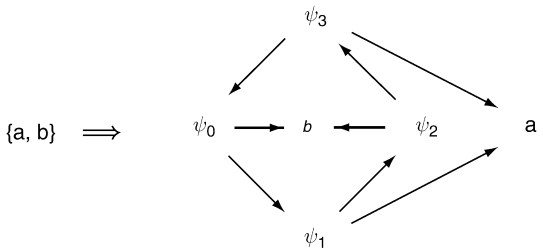
R =

a	b
a	c
c	a

S =

a	b	α
a	c	β
c	a	γ

Computationally Complete Relational Query Languages. Figure 1. An application of new. Here, S = new(R).



Computationally Complete Relational Query Languages. Figure 2. A query with new values not expressible in WhileNew.

Key Applications

The theoretical query languages discussed here are closely related to various practical languages. Thus, RA and RC correspond to pure-SQL. The language While corresponds to wrapping programming constructs such as loops around SQL, as done in PL/SQL (Oracle); assignment statements can be implemented using SQL insert and delete operations.

The language LC (or WhileN) can again be simulated in PL/SQL augmented with natural number variables (with no coercion allowed). Another approach is to embed SQL in a programming language such as C or Java. In such languages, relational variables must be statically defined and so have fixed arity. One significant feature of the embedded SQL languages that sets them apart from LC is that they allow accessing tuples in relations one at a time, using looping over cursors. In particular, this allows coercing the entire database into a native data structure, and yields computational completeness. However, there is a catch: programs using cursors are generally non-deterministic, in the sense that running the same program on the same database content may yield different results. Unfortunately, it is undecidable whether a given embedded SQL program is deterministic, and no natural syntactic restriction is known that ensures determinism while preserving completeness. Thus, completeness is achieved at the cost of losing the guarantee of determinism.

The computationally complete language QL can be simulated in Dynamic SQL. In this language one can dynamically create relation variables whose schemas depend on the data in the database. This can be used to support untyped relational variables. The language WhileNew allowing the introduction of new domain values is akin to object-oriented languages that allow the creation of new object identifiers.

Future Directions

The database area is undergoing tremendous expansion and diversification under the impetus of the Web and a host of specialized applications. Consequently, new structures and objects have to be modeled and manipulated. For example, in biological and scientific applications, sequences and matrices occur prominently; in XML databases, text and tree-structured documents are the main objects. This has led to new database models and query languages. Their formal foundations are fast developing, but are not yet as mature as for the relational data model. Notions of computationally complete languages for the new models are still emerging, and are likely to build upon the theory developed for relational databases.

Cross-references

- ▶ BP-completeness
- ▶ Constraint Query Languages
- ▶ Data Models with Nested Collections and Classes
- ▶ Ehrenfeucht-Fraïssé Games
- ▶ Expressive Power of Query Languages
- ▶ Object Data Models
- ▶ Query Language
- ▶ Relational Calculus
- ▶ Relational Model
- ▶ Semantic Web Query Languages
- ▶ Semi-Structured Query Languages
- ▶ SQL
- ▶ XML
- ▶ XPath/XQuery

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of Databases. Addison-Wesley, Reading, MA, 1995.
2. Abiteboul S. and Kanellakis P.C. Object identity as a query language primitive. J. ACM, 45(5):798–842, 1998.
3. Abiteboul S. and Vianu V. Procedural languages for database queries and updates. J. Comput. Syst. Sci., 41(2):181–229, 1990.
4. Aho A.V. and Ullman J.D. Universality of data retrieval languages. In Proc. 6th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages, 1979, pp. 110–120.
5. Bancilhon F. On the completeness of query languages for relational data bases. In Proc. 7th Symp. on the Mathematical Foundations of Computer Science, 1978, pp. 112–123.
6. Chandra A. Programming primitives for database languages. In Proc. 8th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages, 1981, pp. 50–62.
7. Chandra A. and Harel D. Computable queries for relational data bases. J. Comput. Syst. Sci., 21(2):156–178, 1980.
8. Chandra A. and Harel D. Structure and complexity of relational queries. J. Comput. Syst. Sci., 25:99–128, 1982.
9. Codd E. A relational model for large shared databanks. Commun. ACM, 13(6):377–387, 1970.
10. Codd E. Relational completeness of data base sublanguages. In Data Base Systems, R. Rustin (ed.). Prentice-Hall, Englewood, Cliffs, NJ, 1972, pp. 65–98.
11. Fagin R. Monadic generalized spectra. Zeitschrift für Math. Logik Grundlagen d. Math, 2189–96, 1975.
12. Gyssens M., Van den Bussche J., and Van Gucht D. Complete geometric query languages. J. Comput. Syst. Sci., 58(3):483–511, 1999.
13. Paredaens J. On the expressive power of the relational algebra. Inf. Process. Lett., 7(2):107–111, 1978.
14. Paredaens J. Spatial databases, a new frontier. In Proc. 5th Int. Conf. on Database Theory. 1995, pp. 14–32.
15. Tarski A. What are logical notions? History Phil. Logic, 7:154, 1986. J. Corcoran (ed.).
16. Van den Bussche J., Van Gucht D., Andries M., and Gyssens M. On the completeness of object-creating database transformation languages. J. ACM, 44(2):272–319, 1997.
17. Vardi M.Y. The complexity of relational query languages. In Proc. 14th Annual ACM Symp. on Theory of Computing, 1982, pp. 137–146.

Complex Event

Opher Etzion
IBM Research Lab in Haifa, Haifa, Israel

Synonyms

Composite event; Derived event

Definition

A complex event is an event derived from a collection of events by either aggregation or derivation function [3].

Key Points

A complex event [2], [1] is a derived event; it can be derived by various means:

1. Explicit concatenation of a collection of events,
   - Example: Create an event that contains all the events that are related to the 2008 USA presidential elections.

2. Derivation of an aggregated value from a collection of events from the same type.
   - Example: Create an event that contains the average, maximal and minimal value of a certain stock during a single trade day.
3. Derivation [4] of an event as a function of other events that is a result of event pattern detection.
   - Example: Whenever a sequence of three complain-events from the same customer occurs within a single week, create an event "angry customer" with the customer-id.
     Note that this event may or may not contain the raw complain events.

Cross-references

► Complex Event Processing
► Event Pattern Detection

Recommended Reading

1. Ericsson A.M., Pettersson P., Berndtsson M., Seiriö M. Seamless formal verification of complex event processing applications. In Proc. Inaugural Int. Conf. Distributed Event-Based Systems, 2007, pp. 50–61.
2. Luckham D. The Power of Events. Addison-Wesley, 2002.
3. Luckham D., and Schulte R. (eds.) - EPTS Event Processing Glossary version 1.1. http://complexevents.com/?p=409.
4. Zimmer D., and Unland R. On the Semantics of Complex Events in Active Database Management Systems. In Proc. 15th Int. Conf. on Data Engineering, 1999, pp. 392–399.

Complex Event Processing

Opher Etzion
IBM Research Lab in Haifa, Haifa, Israel

Synonyms

Event processing; Event stream processing

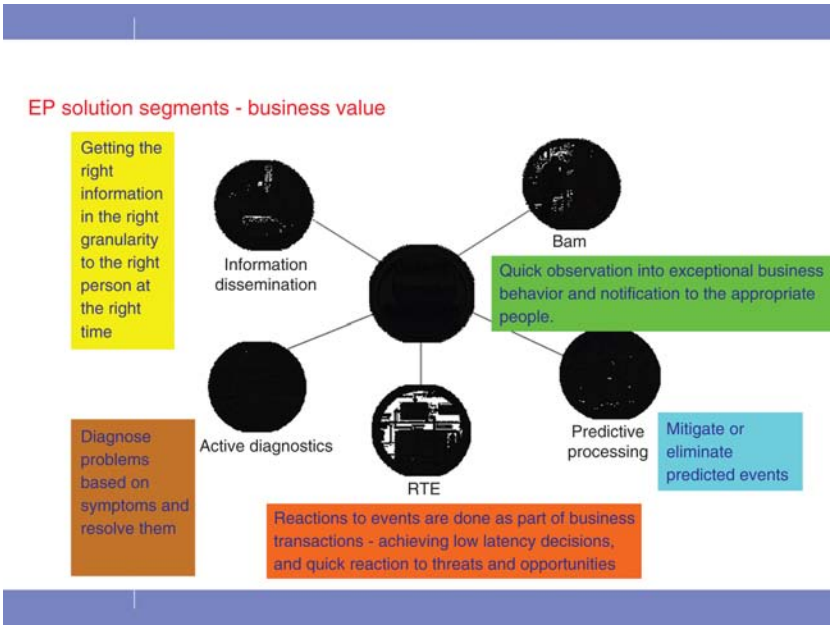
Definition

Complex event processing deals with various types of processing complex events.

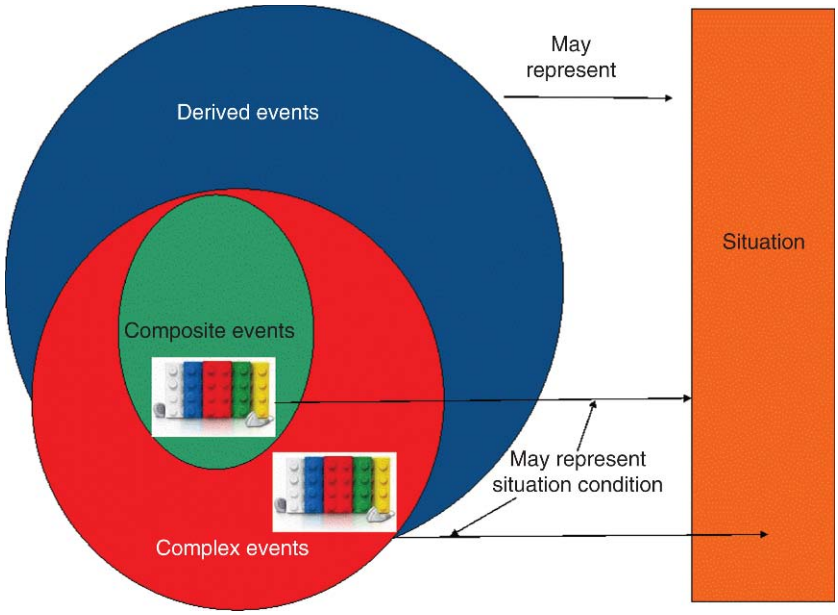
Key Points

Figure 1 shows that the different applications of the CEP technology are not monolithic, and can be classified into five different solution segments, which differ in their motivation, from the user's perspective, they are:

- *RTE (Real-Time Enterprise)*: The processing should affect business processes while they are still running. For example, stop an instance of a workflow that deals with trading a certain stock, if the trade request has been withdrawn.
- *Active Diagnostics*: Finding the root-cause of a problem based on events that are symptoms.
- *Information Dissemination*: A personalized subscription that enables subscriptions in lower granularity, where the subscription does not match the published



Complex Event Processing. Figure 1. Various CEP solution semgents.



Complex Event Processing. Figure 2. Relationships among major complex event processing terms.

event, but a combination of event. For example, notify me when IBM stock has gone up 2% within 1 hour.

- *BAM (Business Activity Management)*: Monitor for exceptional behavior, by defining KPI (Key Performance indicators) and other exceptional behavioral constraints. For example, the delivery has not been shipped by the deadline.
- *Prediction*: Mitigate or eliminate future predicted events.

Figure 2 shows the relations among the different terms around complex event processing. Complex event may be a derived event, but the overlapping among them is partial, the complex event processing is materialized by detecting patterns which may correspond to situations (cases that require action). The exact definitions of terms can be found in the EPTS glossary.

Cross-references

► Complex Event
► Event Pattern Detection

Recommended Reading

1. Etzion O. Event processing, architecture and patterns, Tutorial. In Proc. 2nd Int. Conf. Distributed Event-Based Systems, 2008.
2. Event processing glossary. Available at: http://www.epts.com
3. Luckham D. The Power of Events. Addison-Wesley, Reading, MA, 2002.
4. Sharon G. and Etzion O. Event processing networks – model and implementation. IBM Syst. J., 47(2):321–334, 2008.
5. Zimmer D. and Unland R. On the semantics of complex events in active database management systems. In Proc. 15th Int. Conf. on Data Engineering, 1999, pp. 392–399.

Complex Event Processing (CEP)

► Event and Pattern Detection over Streams
► Stream Processing

Compliance

► Storage Security

Component Abstraction

► Abstraction

Composed Services and WS-BPEL

Francisco Curbera
IBM Research, T.J. Watson Research Center, Hawthorne, NY, USA.

Synonyms

Service orchestration; Service choreography; WS-BPEL; Web services business process execution language

Definition

Service oriented architectures (SOAs) are models of distributed software components where business or scientific functions are delivered by a network of distributed services. Services can be classified into atomic services and composed services, based on how they are created and run. Atomic services are those that do not depend on other services for their operation, and are typically built on technologies native to a specific platform, such as COBOL or Enterprise Java Beans. Composed services are created by composing the function provided by one or more external services into a new service. There is no restriction implied as to the programming model used to create composed services; platform specific programming models such as Enterprise Java Beans or C# have been extended to support the creation of composed services. In addition to that approach, service centric programming models have been defined to support the development of composed services. Foremost among those is the Web Services Execution Language for Web Services (WS-BPEL or BPEL), a service composition language based on the workflow programming model.

Historical Background

The service oriented architecture model [1] evolved early on to include aspects of a component oriented architecture, based on the similarity between the reuse of services and of traditional software components. Component oriented software development has been a constant reference point in the development of SOA. Service composition also draws from the experience of workflow programming and business process re-engineering [5] as it developed throughout the 1990s. The workflow programming model relies on a two level programming paradigm in which applications are combined to accomplish a business goal by means of a graph oriented programming approach. The term "two level programming" derives from the differentiation between the programming task whose goal is to create the individual applications and the creation of the workflow control and data graphs whose goal is the use of those applications to achieve a particular business goal. Because of the close alignment between two level programming and SOA service reuse through composition, workflow programming became a second reference point in the early development of the SOA model, leading to process oriented service composition models and eventually the WS-BPEL language [10].

Foundations

Services and Components

The component oriented software development model (COSD) assumes that software applications and systems can be more efficiently developed and managed when created through the aggregation (composition) of a set of software building blocks (components) independently produced by third parties. Szyperski [8] restricts components to binary code and associated resources, to differentiate components from other software abstractions. More importantly, components are units of deployment (can be independently deployed) and composition (can be independently integrated in composed applications). Components are endowed with well defined interfaces through which they interact with other components.

Service oriented architectures build on this same paradigm. Services in SOA are nothing but the interfaces of SOA components. SOAs add an important new perspective to the usual COSD approach. Components in SOAs are not only units of independent deployment, but also units of independent ownership and management by third parties. The implication is that when creating a composite application two related perspectives are possible. In a traditional COSD approach ("component composition"), a single party deploys and maintains control and management rights over all components. The subjects of the composition task are individual software (binary) components. At runtime they are managed as parts of the composed application to which they belong. By contrast, in a service oriented approach, the subjects of the composition are the services exposed by software components, which are in principle deployed and managed independently of the composite application itself. The term "service composition" is used to refer to a new SOA application created by composing services. The services exposed by a service composition are called "composite services."

The traditional COSD approach is followed by the Service Component Architecture (SCA, see [2]), which defines a straightforward model for the deployment and composition of service oriented components. In SCA, components are "wired" to each other by connecting their interfaces, to create new SCA applications or "composites." SCA composites are component compositions, and they can provide services by exposing one or more service interfaces. Composites are

deployed by deploying and configuring their constituent component's binary implementations. SCA composites are potentially components themselves, thus supporting a recursive composition model.

Service Composition Models

Focusing on both the development time and runtime natures of a service composition, it is possible to distinguish two types of service composition strategies.

"Localized" compositions specify the composite's application internal operation and its dependencies on a set of external services, including in particular the expected behavior of those services. Localized compositions are concerned with applications executing in a single logical location of control. The composition relies on a single logical node where the application logic is executed and from which the interactions with the composed services are controlled. The services used by the composite are naturally executed in separate nodes, but their operation is not the concern of the composition except for its externally visible behavior. The term "service orchestration" is sometimes used to refer to localized service compositions. WS-BPEL is the model for this type of service composition.

"Distributed" service compositions specify the behavior of a set of independent SOA applications interacting through service interfaces as part of a distributed composite application. In its purest form, a distributed composition doesn't specify the internal behavior of any of the participating applications, but only the interactions between the participating services. The execution of the composite is assumed to be distributed among a set of independent nodes, which coordinate their operation according to the composition's specification. The Web Services Choreography Description Language [7] is the prime example of this type of composition. It is also important to mention in this category of service composition the use of SCA as a service wiring specification, in particular in the context of Enterprise Service Bus runtimes; this usage of SCA is discussed later in this entry, under "Key Applications."

In spite of the difference in approach, all service composition models share a set of common characteristics. Both types of service composition require the specification of the expected behavior from the composed services. Expected behavior refers here to the message exchange sequence between the services, which is frequently referred to as a "service conversation." A service conversation is the realization of a business protocol between two parties, over a SOA infrastructure, and is the basis for business level interoperability between services.

Together with a particular service conversation, the relationship between two services is characterized by a particular interaction pattern. Several of these patterns are possible. In a hierarchical organization, the lifecycle of one service is subordinated to another's, such that the later is responsible for the creation and termination of runtime instances of the former. This is the relationship between a process and a sub-process. In a peer-to-peer interaction, both services maintain independent lifecycles and interact as peers. Again, both relationships are possible in both localized and distributed compositions.

It is clear that maintaining the appropriate level of distributed coordination between participating services at runtime is a non-trivial problem. Hierarchical lifecycle dependencies as well as distributed transactional behavior must rely on coordination middleware to ensure agreement on the outcome and synchronization of the interaction. These "coordinated behaviors" of distributed systems are supported in the Web services specification stack by the Web Services Coordination specification, which defines a framework for the creation of distributed protocols [11].

Finally, "recursive composition" is a common characteristic of all service composition models, and the basis for the creation of composite services through composition. In a recursive model, the service composition becomes a service itself, available for further invocation and composition. WS-BPEL, SCA and WS-CDL all allow the creation of new services through composition. A survey of different methodologies for development of service compositions can be found in [4].

Workflow Oriented Composition in WS-BPEL

WS-BPEL is the model and industry standard for localized service composition. WS-BPEL is also the prototype for workflow (or process) oriented composition, a form of service composition that follows the workflow programming model. The WS-BPEL specification adapts the workflow two level programming model to a service centric environment.

The WS-BPEL language consists of two main parts, one dealing with the representation of service interaction requirements and a second one dealing

with the specification of the control and data logic of the workflow.

Service Interaction in Processes A WS-BPEL composition is called a "process model." Service interaction requirements are defined in terms of "abstract" Web services interfaces, that is, XML interface definitions containing no references to service access details such as interaction protocol or endpoint address. Abstract interfaces are defined using the Web Services Description Language (WSDL, see [11]); abstract interfaces are called portTypes in WSDL 1.1. The interaction between the process model and each of the composed services requires in the general case two such abstract interfaces, one used by the process to call the service and one for the service to call back on the process. This pair of interfaces characterizes the interaction and is called a "partner link" in WS-BPEL. The actual ways in which these interfaces are exercised is defined by the business logic of the process definition, which is described later in this section.

WS-BPEL service composition takes place at the interface type level ("service types composition") instead at the instance level ("service instance composition"): abstract service interfaces are referenced by the process, instead of actually deployed services. The goal of this approach to composition is to expand the reusability of WS-BPEL processes, since it allows the same process composition to be used with different services and using different access protocols, as long as the correct portTypes are supported. One particular consequence of this approach is the absence of quality of service specifications in WS-BPEL processes.

A second aspect of the interaction between services is the identification of dynamic correlation data fields for process instance identification and message routing. A WS-BPEL process specifies a model for the execution of individual "process instances." A new process instance is started any time a "start" message (as defined by the process model business logic) is received by the process engine. At any point in time, many instances of the each process model are executing the same process engine. In traditional workflow infrastructures, each instance is identified by a unique identifier, which is carried by all messages sent to the process instance. By carrying this identifier, messages can be routed to the correct instance.

WS-BPEL takes a different approach to the routing problem. The correlation between messages and process instances is done using business information fields, grouped in data sets that the WS-BPEL specification calls "correlation sets." A correlation set is a group of message fields that collectively identify the executing process instance. Correlation fields are identified using XPath expressions to point to individual data fields within the messages received by the process, allowing content based message routing to instances. Correlation information is also specified in outgoing messages, when the values of these fields need to be communicated to an external service. The main benefit of the WS-BPEL correlation approach is to replace the use of platform specific artifacts by business meaningful information for the purpose of interoperable message routing and transaction identification. Correlation values are a key element of any message oriented business protocol.

Specification of Business Logic in Process Compositions Business logic (the control and data graphs of a workflow) is specified in WS-BPEL by means of a set of "atomic" and "structured activities." Atomic activities represent individual steps in the computation of the process graph, and they stand for external service interaction steps (calling or being called by a service), or data manipulation primitives (assignment of data fields). Atomic activities are combined according to a particular sequence of execution using structured activities. WS-BPEL provides structured activities for sequential, parallel as well as conditional and iterative execution. These activities allow the creation of "structured" process graphs, those in which the programming style is similar to that of structured programming languages (albeit with intrinsic parallel capabilities).

WS-BPEL also supports a graph oriented process modeling approach, where atomic activities are combined as nodes of an explicit graph. Edges of the graph are called "control links," and represent explicit transfer of control between a source and a target activity (as opposed to the implicit dependency defined by a sequence activity for example). The execution of the control link graph follows the "dead path elimination" operational semantics where those graph branches not followed in the execution of a process instance (typically because of conditional statements are not satisfied) are transitively eliminated to ensure that every activity in the graph is eventually executed or marked as part of an eliminated "dead path." Dead path elimination and the rule that prevents cyclic control graphs ensure the termination of every valid WS-BPEL

graph's execution. The graph and structured styles are however not strictly separated and can be combined in a rich but at times challenging authoring style. Dead path elimination semantics are embedded in the structured execution model through exception handling, see [3].

Data flow in WS-BPEL is not explicitly modeled, but implied by the use of a set of process variables as inputs and outputs of atomic activities. Data variables in WS-BPEL contain XML data, which is typed according to the XML Schema language.

WS-BPEL contains error handling and recovery primitives to support business interaction in loosely coupled environment. Error handling is supported through the introduction of "fault handlers" which are charged with recovering from errors generated in the course of process execution: faults generated in the course of a service call, errors explicitly raised by the process when certain business conditions are detected, and system generated faults raised when the underlying execution runtime cannot comply with the requirements of process logic (such as errors accessing message data and other error conditions).

Fault handlers are associated with sections of the process called "scopes," the complete process being the outermost scope. Faults originated within a scope are handled by the fault handlers attached to the scope. In the course of recovering from a fault, it may be determined that a particular action already competed must be "undone." A "compensation handler" may be associated with that action (atomic activity) or with a collection of actions (a scope) to indicate the steps required to "undo" that activity or scope, and would then be executed by the fault handler. A compensation handler defines a set of business level actions that provide a logical reversal (backward execution path) of the action in question. Compensation recovery represents an alternative to automatic rollback, and is necessary in service oriented scenarios where loosely coupled services cannot participate in transactions with atomic semantics and automatic rollback recovery. Business transaction in SOA environments require looser transactional semantics (see [7] for details) where recovery is typically specified at the application level.

Key Applications

Current practice of service composition is closely tied to two types of SOA runtimes available in the industry: SOA-enabled business process management (BPM) platforms and enterprise service bus (ESB) infrastructures.

WS-BPEL service composition on BPM platforms represents is by far the most extensive application of the service composition model in enterprise computing. Its success is due to two factors: the increased focus of enterprises on end-to-end business automation, and the fact that service composition builds on the well known business process integration paradigm.

There are important differences between platforms for process oriented service composition and traditional BPM, appearing in two main areas: the reach of the process integration capabilities and the approach to process management. Service oriented process composition adds a new perspective typically absent from traditional BPM platforms, namely, a uniform model for representing internal and external business function based on the service paradigm. The result is the ability to seamlessly incorporate cross departmental and cross organizational services to capture and automate end-to-end business requirements. End-to-end business automation is the main driver of SOA adoption by businesses today.

The management capabilities of the BPM platform are also significantly affected by the service oriented model. Full management is now limited to the process itself and any services deployed locally within the BPM platform. Unlike in traditional BPM, the ability to manage other services is limited (or missing altogether) because they are typically run and managed by different parties (other departmental organizations or different enterprises). BPM platforms supporting service composition must rely on service management standards (such as the Web Services Distributed Management specification [7]) and service level agreements (such as the Web Services Agreement specification, [7]) to provide visibility and limited control over the execution of remote services.

The enterprise service bus architecture (ESB, see [7]) is quickly gaining widespread adoption because it enables simplified service access and reuse across the enterprise. Services are plugged to the ESB to make them available for access by other enterprise applications. ESBs are usually built as service extensions to traditional messaging backbones ("messaging clouds"). Services and applications are connected across the ESB by creating "wires" that declaratively create a logical communication channel between the two. The SCA component and wiring model is used in this context both to drive deployment of SOA components and also to wire existing services, thus exposing its ability to

function as both a component and a service composition model.

Scientific computing middleware software has, independently of its commercial counterpart, identified the need for an architectural model in which computing resources are consumed following the service model. The Open Grid Services Architecture (see [6,7]) shows how the Grid application model is supported by SOA. In this context, scientific workflows have been characterized as compositions of scientific services and specialized languages have been developed to enable the composition of complex scientific computing experiments as process oriented service compositions [9]. The Grid Process Execution Language (GPEL) in particular adapts WS-BPEL to deal with scientific and Grid computing requirements such as processing extremely large data sets, allocating dynamic resources from a Grid infrastructure, and supporting the integration with legacy scientific code among several others (see Chapter 15 in [9]).

Cross-references

► Workflow Management and Workflow Management System

Recommended Reading

1. Burbeck S. The Tao of e-business services. Available at http://www.ibm.com/developerworks/webservices/library/ws-tao/, October 2000.
2. Curbera F., Ferguson D., Nally M., and Stockton M. Toward a Programming Model for Service-Oriented Computing. In Proc. 3rd Int. Conf. Service-Oriented Computing. 2005, pp. 33–47.
3. Curbera F., Khalaf R., Leymann F., and Weerawarana S. Exception Handling in the BPEL4WS Language, In Proc. Int. Conf. Business Process Management, 2003, pp. 276–290.
4. Dustdar S. and Schreiner W. A survey on web services composition. Int. J. Web Grid Serv.,1(1):1–30, 2005.
5. Leymann F. and Roller D. Production Workflow. Prentice Hall, Englewood Cliffs, NJ, 1999.
6. Open Grid Services Architecture, Version 1.5. Available at http://www.ggf.org/documents/GFD.80.pdf, July 2006.
7. Papazoglou M. Web Services: Principles and Technology. Prentice Hall, Englewood Cliffs, NJ, 2007.
8. Szyperski C. Component Software. Addison Wesley, Reading, MA, 2002.
9. Taylor I.J., Deelman E., Gannon D.B., Shields M., (eds.). Workflows for e-Science. Scientific Workflows for Grids. Springer, Berlin, 2007.
10. Web Services Business Process Execution Language Version 2.0. Available at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, April 2007.
11. Weerawarana S., Curbera F., Leymann F., Storey T., and Ferguson D. Web Services Platform Architecture. Prentice Hall, Englewood Cliffs, NJ, 2005.

Composite Event

AnnMarie Ericsson, Mikael Berndtsson, Jonas Mellin
University of Skövde, Skövde, Sweden

Definition

A composite event is a set of events matching an event specification.

Key Points

Pioneering work on composite events was done in the HiPAC project [3] and the ideas were extended and refined in most proposals for active object-oriented databases during the early 1990s.

A composite event is composed according to an event specification (in an event algebra), where the composition is performed using a set of event operators such as disjunction, conjunction and sequence. More advanced event operators have been suggested in literature, e.g., [2,4,5].

The *initiator* of a composite event is the event initiating the composite event occurrence and the *terminator* is the event terminating the composite event occurrence.

Events contributing to composite events may carry parameters (e.g., temporal) in which the event is said to occur. Events contributing to composite events are also referred to as constituent events.

Composite events need to be composed according to some event context that define which event that can participate in the detection of a composite event. The event context is an interpretation of the streams of contributing events. The seminal work by Chakravarthy et al. [1,2], defines four different event contexts (or consumption policies): recent, chronicle, continuous, and cumulative.

In the recent event context, only the most recent constituent events will be used to form composite events. The recent event context is, for example, useful if calculations must be performed on combinations of the last measured values of temperature and pressure in a tank [1,2].

In the chronicle event context, events are consumed in chronicle order. The earliest unused initiator/terminator pair are used to form the composite event. The chronicle event context is, for example, useful if sensors are placed along a conveyor-belt monitoring objects traveling along the belt and combinations of sensor events triggered by the same object is needed. In that case events must be combined in occurrence order since the first event from the first sensor and the first event from the second sensor are likely triggered by the same object [1,2].

In the continuous event context, each initiator starts the detection of a new composite event and a terminator may terminate one or more composite event occurrences. The difference between continuous and chronicle event contexts is that in the continuous event context, one terminator can detect more than one occurrence of the composite event.

In the cumulative event context, all events contributing to a composite event are accumulated until the composite event is detected. When the composite event is detected, all contributing events are consumed [1,2].

Cross-references

- ▶ Active Database (aDB)
- ▶ Active Database Execution Model
- ▶ Active Database Knowledge Model
- ▶ Active Database (Management) System (aDBS/aDBMS)
- ▶ ECA Rules
- ▶ Event
- ▶ Event Detection
- ▶ Event Specification

Recommended Reading

1. Chakravarthy S., Krishnaprasad V., Anwar E., and Kim S.K. Composite events for active databases: semantics contexts and detection. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 606–617.
2. Chakravarthy S. and Mishra D. Snoop: an expressive event specification language for active databases. Data Knowl. Eng., 14(1):1–26, 1994.
3. Dayal U., Blaustein B., Buchmann A., and Chakravarthyand S. et al. HiPAC: A Research Project in Active, Time-Constrained Database Management. Tech. Rep. CCA-88-02, Xerox Advanced Information Technology, Cambridge, 1988.
4. Gatziu S. Events in an Active Object-Oriented Database System. Ph.D. thesis, University of Zurich, Switzerland, 1994.
5. Gehani N., Jagadish H.V., and Smueli O. Event Specification in an Active Object-Oriented Database. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1992, pp. 81–90.

Composite Event Query

▶ Event Specification

Composite Web Applications

▶ Web Mashups

Composition

W. M. P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Synonyms

Service composition; Process composition

Definition

In computer science, *composition* is the act or mechanism to combine simple components to build more complicated ones. Composition exists at different levels. For example, one can think of the usual composition of functions in mathematics, i.e., the result of the composed function is passed to the composing one via a parameter. If one has to functions f and g, these can be combined into a new function $h = f.g$, i.e., $h(x) = f(g(x))$. Another level of abstraction is the level of activities. Here all kinds of process modeling languages can be used to compose activities into processes (e.g., Petri nets, BPMN, etc.). Typical composition operators are sequential composition, parallel composition, etc. Process composition is related to business process management, workflow management, etc. Yet another level of abstraction is provided by services, i.e., more complex services can be composed from simpler ones even when they do not reside in the same organization. Service composition is sometimes also referred to as orchestration and a typical language used for this purpose is BPEL.

Key Points

The composition of more complex components from simpler components has been common practice

in computer science right from the start. It is clear that composition is needed to allow for "divide and conquer" strategies and reuse. One of the most complex issues is the composition of processes. There are basically two types of composition approaches: graphs-based languages and process algebras. Examples of graph-based languages are Petri nets, state charts, BPMN, EPCs, etc. In these languages activities and subprocesses are connected to impose some ordering relations. For example two transitions in a Petri net can be connected by a place such that the first one triggers the second one [3]. Process algebras enforce a more structured way of modeling processes. Typical operations are sequential composition ($x.y$, i.e., x is followed by y), alternative composition ($x + y$, i.e., there is a choice between x and y), and parallel composition ($x||y$, i.e., x and y are executed in parallel) [1,2]. Languages like BPEL provide a mixture of both styles, e.g., operators such as `sequence`, `switch`, `while` and `pick` correspond to the typical process-algebraic operators while the `flow` construct defines in essence an acyclic graph.

The principle of compositionality states that the meaning of a composite is determined by the meanings of its constituent parts and the rules used to combine them. For example, if a process is composed of parts that have certain properties, then these properties should be preserved by the composition and should not depend on lower-level interactions. Such properties can be obtained by simplifying the language used or restricting the compositions allowed.

Cross-references

- ▶ Abstraction
- ▶ BPEL
- ▶ BPMN
- ▶ Business Process Management
- ▶ Orchestration
- ▶ Petri Nets
- ▶ Web Services
- ▶ Workflow Management
- ▶ Workflow Patterns

Recommended Reading

1. Baeten J.C.M. and Weijland W.P. Process Algebra. Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, Cambridge, 1990.
2. Milner R. Communicating and Mobile Systems: The Pi-Calculus. Cambridge University Press, Cambridge, UK, 1999.
3. van der Aalst W.M.P. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, G. Rozenberg (eds.). Lectures on Concurrency and Petri Nets. LNCS, vol. 3098. Springer, Berlin, 2004, pp. 1–65.

Comprehensions

Peter M.D. Gray
University of Aberdeen, Aberdeen, UK

Synonyms

Calculus expression; List comprehension; Set abstraction; ZF-expression

Definition

The comprehension comes from ideas of mathematical set theory. It originated as a way of defining sets of values so as to avoid the famous paradoxes of early set theory, by starting from other well-defined sets and using some carefully chosen constructors and filters. The values in the sets could be tuples of basic values, which suits the *relational model*, or they could be object identifiers, which fits with *ODMG object data models* [2], or they could be tagged variant records which fit well with *semi-structured data*. They could even be sets, lists or bags defined by other comprehensions.

The abstract structure of a comprehension precisely describes almost all the computations done in functional query languages, despite their very different surface syntax. Better still, it allows many optimizations to be expressed as well defined mathematical transformations.

Key Points

Consider an example, using SQL syntax, to find the set of surnames of persons whose forename is "Jim":

```
SELECT surname FROM person WHERE forename = "Jim"
```

Using a list comprehension this can be written as:

```
[surname(p) | p <- person; f <- forename(p); f = "Jim"]
```

This denotes the list of values of the expression to the left of the vertical bar. This expression usually

includes variables such as p which are instantiated by generators to the right of the bar. It can be transliterated as:

The set of values of the surname of p *such that* p *is in the set* person *and* f *is in the set* of forenames of p *and* f is equal to "Jim". Here *forename(p)* could alternatively be written *p.forename* or *(forename p)*.

Thus, the vertical bar can be read as *such that* and the semicolons as conjunctions (*and*). The arrows act as *generators*, supplying alternative possible values, subject to restrictions by predicate terms to the right, acting as *filters*. Thus p is generated from the set of persons but is only chosen where the forename of p satisfies the test of equalling "Jim".

In the above syntax the arrow operator is overloaded , so that if a function such as forename delivers a single value instead of a set then the arrow just assigns that single value to the variable on its left. Strictly, one should make a singleton set containing this value, and then extract it:

```
[surname(p) | p <- person; f <- [forename(p)]; f = "Jim"]
```

This wasteful operation would be compiled away to give this equivalent form:

```
[surname(p) | p <- person; forename(p) = "Jim"]
```

The term "list comprehension" is commonly used, but one should really distinguish between lists, sets and bags [1]. Thus comprehensions are usually represented internally as lists, but often the order is ignored, as in sets, and sometimes it is necessary to keep duplicates and form a bag, especially when totaling up the contents! Particular classes of operator used in comprehensions give rise to *monad comprehensions* and *monoid comprehensions* with valuable mathematical properties.

Cross-references

▶ OQL

Recommended Reading

1. Buneman P., Libkin L., Suciu D., Tannen V., and Wong L. Comprehension syntax. ACM SIGMOD Rec., 23(1):87–96, 1994.
2. Fegaras L. and Maier D. Towards an effective calculus for Object Query Languages. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 47–58.

Compressed and Searchable Data Format

▶ Indexing Compressed Text

Compressed Full-Text Indexing

▶ Indexing Compressed Text

Compressed Suffix Array

▶ Indexing Compressed Text

Compressed Suffix Tree

▶ Indexing Compressed Text

Compressing XML

▶ Managing Compressed Structured Text

Compression of Mobile Location Data

Goce Trajcevski[1], Ouri Wolfson[2], Peter Scheuermann[1]
[1]Northwestern University, Evanston, IL, USA
[2]University of Illinois at Chicago, Chicago, IL, USA

Synonyms

Spatio-temporal data reduction

Definition

In moving objects databases (MOD) [8], the data pertaining to the whereabouts-in-time of a given mobile object is commonly represented as a sequence of *(location, time)* points, ordered by the temporal dimension. Depending on the application's settings, such points may be obtained by different means, e.g., an on-board GPS-based system, RFID sensors, road-network sensors, base stations in a cellular architecture,

etc. The main motivation for compressing the location data of the moving objects is twofold: (i) Reducing the storage requirements: for example, maintaining the information about the daily routes of a few million vehicles, even if the GPS samples are taken once every 30 s, can still generate Terra-Bytes (TB) of data. In addition, with the increase in the number of cellular phones and personal digital assistants that are location aware, the volume of the data corresponding all the mobile entities in a given region will even further increase. However, if a given point, say, (x, y, t) can be eliminated from the representation of the particular trajectory without prohibitively sacrificing the accuracy of its representation, then the space required for that point's storage can be saved; (ii) If a particular point along a given trajectory can be eliminated as soon as it is "generated" (i.e., when the *location* value is obtained by the on-board GPS at a given *time*), yet another type of savings can be achieved – that object need not transmit the *(location, time)* value to a given server, thus reducing the bandwidth consumption. This entry explains the basic problems involved in compressing spatio-temporal data corresponding to trajectories of mobile objects, and outlines the foundations of the approaches that have addressed some of those problems.

Historical Background

The field of *data compression* originated in the works of Shannon, Fano, and Huffman in the 1940s [11], and its main goal is to represent information in as compact form as possible. Some popular forms of data compression have, historically, been around even earlier, for instance, the Morse code has been used in telegraphy since the mid-nineteenth century. Based on the observation that some letters occur more frequently than others, the code assigns shorter sequences of (combinations of) "·" and "−" to such letters. Thus, for example, "*e*" → " · ", "*a*" → " ·−". On the other hand, the letters which occur less frequently, are assigned longer sequences like, for example, "*q*" → "−−·−". In this setting, the frequency of the occurrence of single letters provided statistical structure that was exploited to *reduce the average time* to transmit a particular message since, in practice, the duration of the symbol " − " is (approximately) three times longer than the duration of the "·" symbol. A natural extension is to use frequency of the *words* over a given alphabet, in order to further compress the encoding of a given text, which is used in the Grad-2 Braille coding. When the probability model of the source is known, the popular approach for encoding a collection of *letters* of a given *alphabet* is the Huffman coding [11]. Contrary to the ASCII/EBDCIC which are *fixed-length* codes, in the sense that every symbol is assigned same number of bits, Huffman code is a *variable-length* one, which assigns shorter codewords to symbols occurring less frequently, in an optimal manner with respect to the entropy of the source. When dealing with texts, some statistical correlations can be detected in terms of the occurrences of *words*. Taking this into consideration the, so called, *dictionary techniques* for data compression have been obtained, an example of which is the UNIX `compress` command. In computer science, the need for compression techniques was mainly motivated by the reduction of the size of the data for storage and transmission purposes.

Different kind of data may exhibit different kinds of structures that can be exploited for compression, provided a proper model is developed. For example, given the sequence of numbers {9,11,11,11,14,13, 15,17,16,17,20,21}, let x_n denote its nth element. If one transmits the binary representation of each x_i ($i \in \{1,2,...,12\}$), 5 bits-per-sample are needed, for a total of 60 bits transmitted. However, if one provides a model represented by the equation $\overline{x}_n = n + 8$, then the difference-sequence (i.e., the residual) of the initial sequence, represented as $e_n = x_n - \overline{x}_n$ becomes: {0, 1, 0, −1, 1, −1, 0, 1, −1, −1, 1, 1}. This sequence consists of only three different numbers {− 1, 0, 1} Using the mapping " − 1" → "00"; "0" → " − 1"; "1" → "10", each e_i can be encoded with only 2 bits. Hence, the sequence can be transmitted with a total of 24 bits, achieving 60% compression ratio and, consequently, savings in the transmission. Such intrinsic properties of the underlying domain have been heavily exploited in the areas of speech compression, image compression, etc. [11].

There are several classification of compression techniques. One example, as mentioned above, is *fixed* vs. *variable* length, however, one may also need to distinguish between *static* (the codewords are fixed, say, before the transmission) and *dynamic/adaptive*. The classification that is most relevant for this article is *lossless* vs. *lossy* compression. With lossless compression, the original data can be *exactly* recovered from the compressed one, which it is not the case for the lossy compression.

There are several different measures regarding the quality of a given compression method: (i) the complexity of the algorithms; (ii) the memory footprint required; (iii) the amount of compression; (iv) the quality of the data (in lossy compression). The main goal of the methods for compressing *spatio-temporal* data is to strike a good balance between the complexity of the algorithm and the error-bound on the compressed data with respect to the original one.

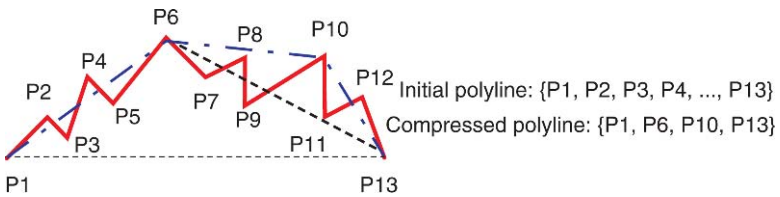
There are two research fields that have addressed problems similar in spirit to the ones of compressing mobile location data:

1. *Cartography*. The goal of the *map generalization* in cartography is to reduce the size/complexity of a given map for the purpose of simplified representation of the details appropriate to a given scale [16].
2. *Computational geometry* (CG). In particular, the problem of *polyline* (which is, a sequence of nodes specifying a chain of line segments) simplification [3], that can be described as follows. Given a polyline PL_1 with vertices $\{v_1,v_2,...,v_n\}$ in a respective k-dimensional Euclidean space, and a tolerance ε, construct another polyline PL_1' with vertices $\{v_1',v_2',...,v_m'\}$ in the same space, such that $m \leq n$ and for every point $P \in PL_1$ its distance from PL_1' is smaller than a given threshold: $dist(P, PL_1') \leq \varepsilon$. In case $\{v_1',v_2',...,v_m'\} \subseteq \{v_1, v_2,...,v_n\}$, PL_1' is a *strong* simplification of PL_1; otherwise PL_1' is called a *weak* simplification. There are two distinct facets of the minimal line simplification problem: (i) Given PL and ε, minimize the number of points m in PL' (known as min-# problem) [5], and (ii) Given PL and the "budget" m of the vertices in PL', minimize the error ε (known as min-ε problem).

A popular heuristic for polyline simplification in the context of map generalization was proposed by Douglas and Peucker in [6]. Essentially, it recursively approximates a given polyline in a "divide and conquer" manner, where the farthest vertex, according to the distance used, is selected as the "divide" point. Given a *begin_vertex* p_i and an *end_vertex* p_j, if the greatest distance from some vertex p_k to the straight line segment $\overline{p_ip_j}$ is greater than the tolerance ε, then the trajectory is broken into two parts at p_k and the procedure is recursively called on each of the sub-polylines $\{p_i,...,p_k\}$ and $\{p_k,...,p_j\}$; Otherwise, the vertices between p_i and p_j are removed from trajectory and this segment is simplified as a straight line $\overline{p_ip_j}$. An illustration of the DP heuristic is given in Fig. 1. Although the original version of the algorithm, as presented in [6], has a running time $O(n^2)$, an $O(n\log n)$ algorithm was presented in [9]. However, none of these algorithms can ensure an optimality, in terms of the size of the compression (alternatively, in terms of a minimal ε-error for a fixed reduction factor). An optimal algorithm was presented in [5], with a complexity of $O(n^2)$, subsequently extended for 3D and higher dimensions in [3].

Foundations

Assuming that the objects are moving in a 2D space with respect to a given coordinate system, a *trajectory*, which is often used in the MOD literature [8,13,15] to describe the motion of the moving objects, is defined as a function $F_t : T \rightarrow \mathcal{R}^2$ which maps a given (temporal) interval $[t_b, t_e]$ into a one-dimensional subset of $\mathcal{R}^2$. It is represented as a sequence of 3D points (2D geography + time) (x_1,y_1,t_1), (x_2,y_2,t_2),...,(x_n,y_n,t_n), where $t_b = t_1$ and $t_e = t_n$ and $t_1 \leq t_2 \leq ... \leq t_n$. Each point (x_i,y_i,t_i) in the sequence represents the 2D location (x_i,y_i) of the object, at the time t_i. For every $t \in (t_i,t_{i+1})$, the *location* of the object is obtained by a *linear interpolation* between (x_i,y_i) and (x_{i+1},y_{i+1}) with the ratio $(t - t_i)/(t_{i+1} - t_i)$, which is, in between two points the object is assumed to move along a straight line-segment and with a constant speed. The 2D



Compression of Mobile Location Data. Figure 1. Douglas–Peucker heuristic.

projection of the trajectory is a polygonal chain with vertices (x_1, y_1), (x_2, y_2)...(x_n, y_n), called a *route*.

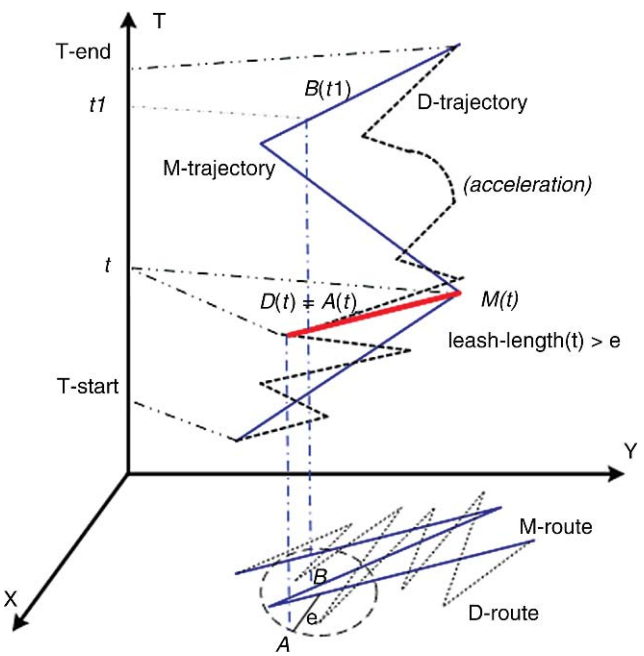
Observe that a trajectory may represent both the *past* and the *future* motion, i.e., the *motion plan* of a given object (c.f. [8]). Typically, for future trajectories, the user provides the starting location, starting time and the destination (plus, possibly, a set of to-be-visited) points, and the MOD server uses these, along with the distribution of the speed-patterns on the road segments as inputs to a dynamic extension of the Dijkstra's algorithm [13], to generate the shortest travel-time trajectory.

One may be tempted to straightforwardly apply the existing results on polyline simplification from the CG literature (e.g., the DP [6,9] or the optimal algorithm [3,5]), in order to compress a given trajectory. However, as pointed out in [4], the semantics of the spatial + temporal domain combined, raises two major concerns:

1. What is the *function* used to measure the *distance between points* along trajectories?
2. How does the choice of that function affect the *error* that the compressed trajectory introduces in the answers of the popular spatio-temporal queries? In the sequel, each of these questions is addressed in a greater detail.

Distance Function

A popular distance function between two curves, often used in CG applications is the, so called, Hausdorff distance [1]. Essentially, two curves C_1 and C_2, their Hausdorff distance simply looks for the smallest ε such that C_1 is completely contained in the ε-neighborhood of C_2 (i.e., C_1 is completely contained in the Minkowski Sum of C_2 and a disk with radius ε) and vice versa. Although it is arguably a very natural distance measure between curves and/or compact sets, the Hausdorff distance is too "static", in the sense that it neither considers any direction nor any dynamics of the motion along the curves. A classical example of the inadequacy of the Hausdorff distance, often used in the CG literature [1,2] is the "*man walking the dog*". Figure 2 illustrates the corresponding routes of the man *(M-route)* and the dog *(D-route)*, as well as their trajectories *M-trajectory* and *D-trajectory*. Observe that, ignoring the temporal aspect of their motions, the D-route and the M-route are within Hausdorff distance of *e*, as exemplified by the points *A* and *B* in the *XY* plane. However, their actual *temporally aware* distance corresponds to the minimal length of the leash that the man needs to hold. The 3D part of Fig. 2 illustrates the discrepancy between the distances among the points along the *routes*, and their corresponding counterparts along *trajectories*: when the dog is at the point



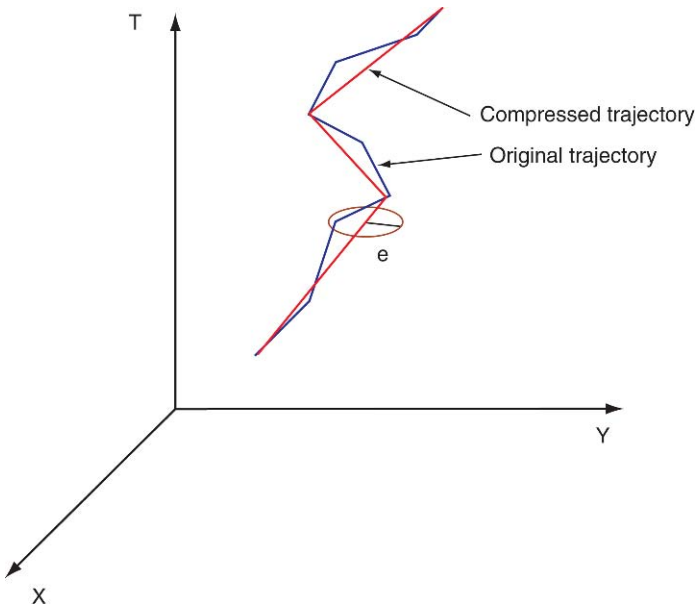
Compression of Mobile Location Data. Figure 2. Hausdorff vs. Fréchet distance.

A, which is at time t, the man is actually at $M(t)$, and their distance is much greater then e (the man is at the geo-location B at the time $t_1 > t$). The Fréchet distance [2] is more general than the Hausdorff one, in the sense that it allows for a variety of possible motion-patterns along the given route-segments. As an illustration, observe that on the portion of the *D-trajectory*, the dog may be moving non-uniformly (i.e., accelerating) along a route segment.

The discussion above illustrates two extreme points along the spectrum of distance functions for moving objects. Although the Fréchet distance is the most general one, regarding the possible dynamics of motions, it is unnecessarily complex for the common trajectory model in MOD settings. The inadequacy of the L_2 norm as a distance function for spatio-temporal trajectories was pointed out in [4] where, in order to properly capture the semantics of the problem domain, alternative distance functions were introduced. Given a spatio-temporal point $p_m = (x_m, y_m, t_m)$ and a trajectory segment $\overline{p_i, p_j}$ between the vertices $p_i = (x_i, y_i, t_i)$ and $p_j = (x_j, y_j, t_j)$, [4] proposed the E_u and E_t distance functions between the p_m and $\overline{p_i, p_j}$, which are explained next

1. E_u – The three dimensional time_uniform distance, which is defined when $t_m \in [t_i, t_j]$, as follows: $E_u(p_m, \overline{p_i p_j}) = \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2}$ where $p_c = (x_c, y_c, t_c)$ is the unique point on $\overline{p_i p_j}$ which has the same *time* value as p_m (i.e., $t_c = t_m$). An illustration of using the E_u distance function for reducing the size of a given trajectory is presented in Fig. 3. Intuitivelly, the distance is measured at equal *horizontal* planes, for the respective values of the temporal dimension. One can "visually" think of the relationship between the original trajectory and the compressed trajectory as follows: the original trajectory is contained inside the sheared cylinder obtained by sweeping (the center of) a horizontal disk with radius ε along the compressed trajectory.
2. E_t – The time distance is defined as: $E_t(p_m, \overline{p_i p_j}) = |t_m - t_c|$, where t_c is the time of the point on the XY projection $\overline{p'_i p'_j}$ of $\overline{p_i p_j}$, which is closest (in terms of the 2D Euclidean distance) to the XY projection p'_m of p_m. Intuitively, to find the time distance between p_m and $\overline{p_i p_j}$, one needs to:
   1. Project each of them on the XY plane;
   2. Find the point $p'_c \in \overline{p'_i, p'_j}$ that is closest to p'_m;
   3. Find the difference between the corresponding times of p_c and p_m.

An important observation regarding the *computation* of the compressed version of a given original trajectory as an input, is that both the DP [6] and the



Compression of Mobile Location Data. Figure 3. E_u distance function for trajectory compression.

optimal algorithm [5] can be used, provided they are appropriately modified to reflect the distance function used. Experimental results in [4] demonstrated that the DP heuristics yields a compression factor that is very comparable to the one obtained by the optimal algorithm, however, its execution is much faster.

Spatio-Temporal Queries and Trajectory Compression

The most popular categories of spatio-temporal queries, whose efficient processing has been investigated by many MOD researchers [8] are:

1. *where_at*(T,t) – returns the expected location at time t.
2. *when_at*(T,x,y) – returns the time t at which a moving object on trajectory T is expected to be at location (x,y).
3. *intersect*(T,P,t_1,t_2) – is *true* if the trajectory T intersects the polygon P between the times t_1 and t_2. This is an instance of the, so called, spatio-temporal range query).
4. *nearest_neighbor*(T,O,t) – The operator is defined for an arbitrary set of trajectories O, and it returns a trajectory T' of O. The object moving according to T', at time t, is closest than any other object of O to the object moving according to T.
5. *join*(O, Θ) – O is a set of trajectories and the operator returns the pairs (T_1,T_2) such that their distance, according to the distance function used, is less than a given threshold Θ.

An important practical consideration for compressing trajectory data is how the (im)precision generated by the compression, affects the answers of the spatio-temporal queries. As it turns out, the distance function used in the compression process plays an important role and, towards this, the concept of *soundness* [4] of a distance function with respect to a particular query was introduced in [4]. A pair *(distance_function, query)* is called *sound* if the error of the *query*-answer, when processed over the compressed trajectory is bounded. In case the error is *unbounded*, which is, although the compression itself guarantees a distance-error of ε between the points on the compressed trajectory with respect to the original one, the error of the answer to the query can grow arbitrarily large, the pair is called *unsound*. Table 1 below (adapted from [4]) summarizes the soundness properties of three distance functions with respect to five categories of spatio-temporal queries.

Compression of Mobile Location Data. Table 1. Distance soundness and error-bound on spatio-temporal query answers

	Where_at	*When_at*	*Intersect*	*Nearest neighbor*
E_2 (L_2over routes)	Unsound	Unsound	Unsound	Unsound
E_u	Sound (ε)	Unsound	Sound (ε)	Sound (2ε)
E_t	Unsound	Sound (ε)	Unsound	Unsound

As one can see, there is no single distance function that is sound for all the possible spatio-temporal queries.

The compression techniques for spatio-temporal data presented thus far, implicitly assumed that the trajectories are available in their entirety, i.e., they are past-motion trajectories. However, in practice, it is often the case that the *(location, time)* data is generated on-board mobile units, and is transmitted to the MOD server in real time [7,17]. *Dead-reckoning* is a policy which essentially represents an agreement between a given moving object and the MOD server regarding the updates transmitted by that particular object. The main idea is that the *communication* between them can be reduced (consequently, network bandwidth can be spared) at the expense of the *imprecision* of the data in the MOD representing the object's motion. In order to avoid an unbounded error of the object's location data, the agreement specifies a threshold δ that is a parameter of the policy, which can be explained as follows:

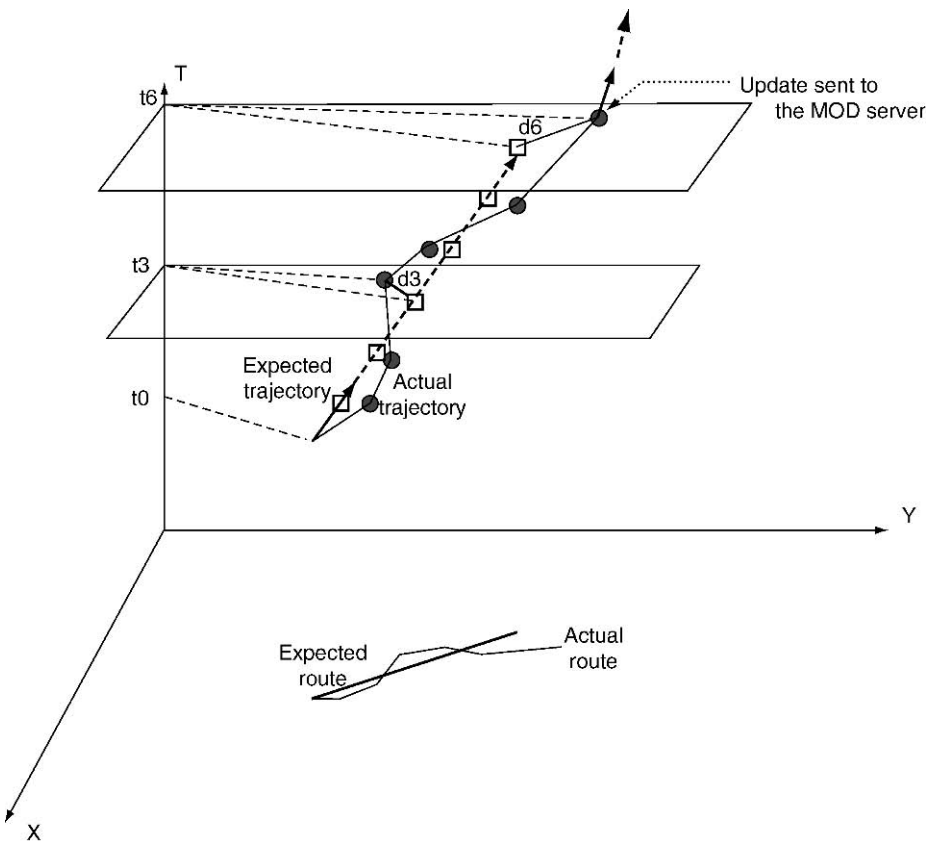
1. The object sends its *location* and the *expected velocity* to the MOD server and, as far as the MOD server is concerned, the future trajectory of that object is an infinite ray originating at the update point and obtained by extrapolation, using the velocity vector.
2. The information that the MOD server has is the *expected trajectory* of the moving object. However, each moving object is aware of its *actual location*, by periodically sampling it, e.g., using an on-board GPS.
3. For as long as its actual location at a given time t_i does not deviate by more than δ from the location that the MOD estimates at t_i using the information previously transmitted, the object does not

transmit any new updates. When the actual distance deviates by more then δ from its location on the expected trajectory, the object will send another *(location, time, velocity)* update.

The policy described above is commonly known as a *distance-based* dead reckoning, and an illustration is given in Fig. 4. At time t_0 the object sent its location and the predicted velocity (arrowed line) to the MOD server. The dashed line extending the vector indicate the expected trajectory of the moving object and the squares along it indicate the object's positions at six time instances, as estimated by the MOD, while the shaded circles indicate the actual positions of the object. Typically, the actual trajectory is obtained by connecting the GPS points with straight line-segments, assuming that in-between two updates, the object was moving with a constant speed. As illustrated, at t_6 the distance between the actual position and the MOD-estimated one exceeds the threshold agreed upon ($d_6 > \delta$) and the object sends a new update, at which point the MOD changes the expected trajectory, based on that update. Thus, at t_6, the MOD server actually performs two tasks:

1. Corrects its own "knowledge" about the recent past and approximates the actual trajectory between t_0 and t_6 with a straight line-segment, which defines the *actual simplification* of the near-past trajectory;
2. generates another infinite ray corresponding to the future-expected trajectory, starting at the last update-point, and using the newly received velocity vector for extrapolation.

Various trade-offs between the update costs and the (impacts on the) imprecision of the MOD data for several different variants of dead reckoning are investigated in [17]. The dead-reckoning, in a sense, achieves in real-time both of the goals of compression: – reduces the communication, and enables the MOD server to store only a subset of the actual trajectory. Assuming that a dead-reckoning policy with threshold δ was used in real-time, clearly, the MOD has obtained a compressed past-trajectory, say Tr_m^c, of a given mobile



Compression of Mobile Location Data. Figure 4. Distance-based dead-reckoning policy.

object o_m. If o_m was to transmit every single GPS-based update, i.e., no dead-reckoning applied, the MOD would have an uncompressed trajectory Tr_m available. The results in [14] have established that Tr_m^c is a strong simplification of Tr_m, with an error-bound 2δ.

Key Applications

The compression of moving objects trajectories data is of interest in several scientific and application domains.

Wireless Sensor Networks (WSN)

Wireless sensor networks consist of a large number of *sensors* – devices that are capable of measuring various phenomena; performing elementary calculations; and communicating with each other, organizing themselves in an ad hoc network [19]. A particularly critical aspect of the WSN is the efficient management of the energy-reserves, given that the communication between two nodes drains a lot more battery-power than the operations of sensing and (local) computing. Consequently, in many tracking applications that can tolerate delays and imprecision in the *(location, time)* data, performing local compression of the trajectory data, before it is sent to a particular sink, can yield substantial increase in the networks' lifetime. Different policies for such compressions are presented in [18].

Location-Based Services (LBS)

A variety of applications in LBS depend on the data for mobile objects with different mobility properties (e.g., pedestrians, private vehicles, taxis, public transportation, etc.). Typically, LBS are concerned with a context-aware delivery of the data which matches the preferences of users based on their locations [12]. In order to provide faster response time, and more relevant information, the LBS should be able to predict, based on the motion patterns, what kind of data will be relevant/requested in a near future by given users. This, in turn, implies some accumulated knowledge about the mobility patterns of the users in the (near) past. However, keeping such data in its entirety can impose prohibitively high storage requirements.

Geographic Information Systems (GIS)

Recently, a plethora of services and devices has emerged for providing path planning and navigation for the mobile users: from MapQuest and Google-maps, through Garmin and iPaq Travel Companion. Each of these services relies on some traffic-based information in order generate the optimal (in distance or travel-time) path for their users. However, as the traffic conditions fluctuate, the future-portions of the routes may need to be recalculated. In order to better estimate the impact of the traffic fluctuations, some knowledge from the past is needed which, ultimately means storing some past information about trajectories. However, as observed in the literature [4], storing the uncompressed trajectory data corresponding to daily driving activities of few millions of users, could require TBs of data.

Spatio-Temporal Data Mining

Clustering is a process of grouping a set of (physical or abstract) objects into classes of similar objects, and its purpose is to facilitate faster data analysis in a given domain of interest. With the recent advances in miniaturization of computing devices and communications technologies, the sheer volume makes it very costly to apply clustering to the original trajectories' data. Compressing such data, especially if one can guarantee a bounded error for the queries of interest, can significantly improve the processing time for many algorithms for trajectories clustering [10].

Future Directions

Any problem-domain that depends on storing large volumes of trajectories' data, in one way and level or another, needs some sort of data compression in order to reduce the storage requirements and to speed up processing of spatio-temporal queries of interest. Clearly, a desirable property of the compression techniques is to ensure a bound on the errors of the answers to the queries.

There are several directions of interest for the future research on mobile data compression. In applications like GIS and LBS, it is a paramount to add some context-awareness to the compression techniques. For example, combining the mobile location data compression with the particular tourists attractions and the season/time, could provide a speed-up in algorithms which are used for generating real-time advertisements, while ensuring that the error (in terms of users that received particular ad) is bounded. An interesting aspect that has been presented in [4] is the, so-called, *aging* of the trajectories: a trajectory that is 1-week old could have higher impact on the traffic-impact analysis, than a trajectory that was recorded

5weeks ago. Consequently, one may reduce the older trajectory with a higher error-bound, thus further reducing the storage requirements. Automatizing this process in a manner that reflects the specifics of a given problem-domain (e.g., context-aware information delivery) is an open question. Despite the large body of works on OLAP and warehousing of traditional data, very little has been done on spatio-temporal OLAP. It is likely that the process of mobile data compression will play an important role in these directions.

Cross-references

- ▶ Data Compression
- ▶ Data Mining
- ▶ Moving Objects Databases

Recommended Reading

1. Alt H. and Guibas L. Discrete geometric shapes: matching, interpolation, and approximation. In Handbook of Computational Geometry. Elsevier, Amsterdam, 1999.
2. Alt A., Knauer C., and Wenk C. Comparison of distance measures for planar curves. Algorithmica, 38(1):45–58, 2004.
3. Barequet G., Chen D.Z., Deascu O., Goodrich M.T., and Snoeyink J. Efficiently approximating polygonal path in three and higher dimensions. Algorithmica, 33(2):150–167, 2002.
4. Cao H., Wolfson O., and Trajcevski G. Spatio-temporal data reduction with deterministic error bounds. VLDB J., 15(3):211–228, 2006.
5. Chan W. and Chin F. Approximation of polygonal curves with minimum number of line segments or minimal error. Int. J. Computat. Geometry Appl., 6(1):59–77, 1996.
6. Douglas D. and Peucker T. Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. Can. Cartographer, 10(2):112–122, 1973.
7. Gedik B. and Liu L. Mobieyes: a distributed location monitoring service using moving location queries. IEEE Trans. Mobile Comput., 5(10):1384–1402, 2006.
8. Güting R.H. and Schneider M. Moving objects databases. Morgan Kaufmann, Los Altos, CA, 2005.
9. Hershberger J. and Snoeyink J. Speeding up the douglas-peucker line-simplification algorithm. In Proc. 5th Int. Symp. on Spatial Data Handling, 1992, pp. 134–143.
10. Jensen C.S., Lin D., and Ooi B.C. Continuous clustering of moving objects. IEEE Trans. Knowl. Data Eng., 19(9):1161–1174, 2007.
11. Sayood K. Introduction to Data Compression. Morgan Kaufmann, Los Altos, CA, 1996.
12. Schiller J. and Voisard A. Location-Based Services. Morgan Kaufmann, Los Altos, CA, 2004.
13. Trajcevski G., Wolfson O., Hinrichs K., and Chamberlain K. Managing uncertainty in moving objects databases. ACM Trans. Database Syst., 29(3):463–507, 2004.
14. Trajcevski G., Cao H., Wolfson H., Scheuermann P., and Vaccaro D. On-line data reduction and the quality of history in moving objects databases. In Proc. 5th ACM Int. Workshop on Data Eng. for Wireless and Mobile Access, 2006, pp. 19–26.
15. Vlachos M., Hadjielefteriou M., Gunopulos D., and Keogh E. Indexing multidimensional time-series. VLDB J., 15(1):1–20, 2006.
16. Weibel R. Generalization of spatial data: Principles and selected algorithms. In Algorithmic Foundations of Geographic Information Systems. Van Kreveld M. Nievergelt J., Roos T., and Widmayer P. (eds.). LNCS Tutorial Springer, Berlin, 1998.
17. Wolfson O., Sistla A.P., Chamberlain S., and Yesha Y. Updating and querying databases that track mobile units. Distrib. Parallel Databases, 7(3):257–387, 1999.
18. Xu Y. and Lee W.-C. Compressing moving object trajectory in wireless sensor networks. Int. J. Distrib. Sensor Netw. 3(2): 151–174, 2007.
19. Zhao F. and Guibas L. Wireless sensor networks: an information processing approach. Morgan Kaufmann, Los Altos, CA, 2004.

Computational Media Aesthetics

Chitra Dorai
IBM T. J. Watson Research Center, Hawthorne, NY, USA

Synonyms

CMA; Media semantics; Production-based approach to media analysis

Definition

Computational media aesthetics is defined as the algorithmic study of a variety of image and aural elements in media founded on their patterns of use in film grammar, and the computational analysis of the principles that have emerged underlying their manipulation, individually or jointly, in the creative art of clarifying, intensifying, and interpreting some event for the audience [3]. It is a computational framework to establish semantic relationships between the various elements of sight, sound, and motion in the depicted content of a video and to enable deriving reliable, high-level concept-oriented content annotations as opposed to verbose low-level features computed today in video processing for search and retrieval, and nonlinear browsing of video. This media production knowledge-guided semantic analysis has led to a shift away from a focus on low level features that cannot answer high level queries for all types of users, to applying the principled approach of computational media aesthetics to analyzing and interpreting diverse video domains such as movies, instructional media, broadcast video, etc.

Historical Background

With the explosive growth of media available on the Web, especially on hugely popular video sharing websites such as YouTube, managing the digital media collections effectively and leveraging the media content in the archives in new and profitable ways continues to be a challenge to enterprises, big and small. Multimedia content management refers to everything from ingesting, archival and storage of media to indexing, annotation and tagging of content for easy access, search and retrieval, and browsing of images, video and audio. One of the fundamental research problems in multimedia content management is the semantic gap – that renders all automatic content annotation systems of today brittle and ineffective – between the shallowness of features in their descriptive power that can be currently computed automatically and the richness of meaning and interpretation that users desire search algorithms to associate with their queries for easy searching and browsing of media. Smeulders et al. [8] describe that while "the user seeks semantic similarity, the database can only provide similarity on data processing." This semantic gap is a crucial obstacle that content management systems have to overcome in order to provide reliable media descriptions to drive search, retrieval, and browsing services that can gain widespread user acceptance and adoption. There is a lack of framework to establish semantic relationships between the various elements in the content since current features are frame/shot-representational and far too simple in their expressive power.

Addressing the semantic gap problem in video processing will enable innovative media management, annotation, delivery and navigational services for enrichment of online shopping, help desk services, and anytime-anywhere training over wireless devices. Creating technologies to annotate content with deep semantics results in an ability to establish semantic relationships between the form and the function in the media, thus for the first time enabling user access to stored media not only in predicted manner but also in unforeseeable ways of navigating and accessing media elements. Semantics-based media annotations will break the traditional linear manner of accessing and browsing media, and support vignette-oriented viewing of audio and video as intended by the content creators. This can lead to new offerings of customized media management utilities for various market segments such as education and training video archives, advertisement houses, news networks, broadcasting studios, etc.

Foundations

Computational Media Aesthetics advocates an approach that markedly departs from existing methods for deriving video content descriptions by analyzing audio and visual features (for a survey of representative work, see [8]). It proposes that to go beyond describing just what is seen in a video, the visual and emotional impact of how the content is depicted needs to be understood. Both media compositional and aesthetic principles need to guide media analysis for richer, more expressive descriptions of the content depicted and seen.

What are the methodologies for analyzing and interpreting media? Structuralism, in film studies for example, proposes film segmentation followed by an analysis of the parts or sections. Structural elements or portions of a video, when separated from cultural and social connotations can be treated as plain data and therefore, can be studied using statistical and computational tools. Another rich source is production knowledge or film grammar. Directors regularly use accepted rules and techniques to solve problems presented by the task of transforming a story from a written script to a captivating visual and aural narration [2]. These rules encompass a wide spectrum of cinematic aspects ranging from shot arrangements, editing patterns and the triangular camera placement principle to norms for camera motion and action scenes. Codes and conventions used in narrating a story with a certain organization of a series of images have become so standardized and pervasive over time that they appear natural to modern day film production and viewing. However, video production mores are found more in history of use, than in an abstract predefined set of regulations, are descriptive rather than prescriptive, and elucidate on ways in which basic visual and aural elements can be synthesized into larger structures and on the relationships that exist between the many cinematic techniques employed worldwide and their intended meaning to a movie audience.

Media aesthetics is both a process of examination of media elements such as lighting, picture composition, and sound by themselves, and a study of their role in manipulating the viewer's perceptual reactions, in communicating messages artistically, and in synthesizing effective media productions [10]. Inspired

by it, Dorai and Venkatesh defined Computational media aesthetics [3] as the algorithmic study of a variety of image and aural elements in media guided by the patterns of their use, and the computational analysis of the principles for manipulating these elements to facilitate high-level content annotations.

Computational media aesthetics provides a handle on interpreting and evaluating relative communication effectiveness of media elements in productions through knowledge of film codes that mediate perception of the content shown in the video. It exposes the semantic and semiotic information embedded in the media production by focusing not merely on the representation of perceived content in digital video, but on the semantic connections between the elements and the emotional, visual appeal of the content seen and remembered. It advocates a study of mappings between specific cinematic elements and narrative forms, and their intended visual and emotional import.

In multimedia processing, many research efforts have sought to model and describe specific events occurring in a particular video domain in detail for providing high-level descriptions; computational media aesthetics, on the other hand enables development of video analysis techniques founded upon production knowledge for film/video understanding, for the extraction of high-level semantics associated with the expressive elements and narrative forms synthesized from the cinematic elements, and for the detection of high-level mappings through the use of software models. It highlights the systematic use of film grammar, as motivation and also as foundation in the automated process of analyzing, characterizing, and structuring of produced videos for media search, segment location, and navigational functions.

Computational media aesthetics provides a framework to computationally determine elements of form and narrative structure in videos from the basic units of film grammar namely, the shot, the motion, the recording distances, and from the practices of combination that are commonly followed during the audiovisual narration of a story. At first, primitive computable aspects of cinematographic techniques are extracted. New expressive elements (higher order semantic entities) can then be defined and constructed from these primitive aspects. Both the definition and extraction of these semantic entities are based on film grammar, and these entities are formulated only if directors purposefully design them and manipulate them. The primitive features and the higher order semantic notions thus form the vocabulary of content description language for media.

Key Applications

In seeking to create tools for the automatic understanding of media, computational media aesthetics states the problem as one of faithfully reflecting the forces at play in media production, and interpreting the data with its maker's eye. Several studies have explored the workings of Computational Media Aesthetics when applied to extraction of meaning using many of the aesthetic elements introduced by Zettl [10]: Time, sound and color. Adams et al. [1] took an example of carrying one aspect of film grammar all the way from literature to computable entity, namely tempo and pace for higher level analysis of movies. Adams et al. [1] showed that although descriptive and sometimes fuzzy in scope, film grammar provides rich insights into the perception of subjective time as tempo and pace and its manipulation by the makers of film for drama. Further research [9,6,5,7,4] has applied this approach pervasively from extracting mood in music, emotion in movies, to adding musical accompaniment to videos and extracting semantic metadata for mobile images at the time of image capture.

Film is not the only domain with a grammar to leverage in analysis. News, sitcoms, educational video, etc., all have more or less complex grammars that may be used to capture their crafted structure and to derive semantic descriptions with automated techniques following the framework of computational media aesthetics.

Cross-references

- ▶ Media Semantics
- ▶ Multimedia Processing
- ▶ Video Analysis

Recommended Reading

1. Adams B., Dorai C., and Venkatesh S. Towards automatic extraction of expressive elements from motion pictures: tempo. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2000, pp. 641–645.
2. Arijon D. Grammar of the film language. Silman-James Press, Los Angeles, CA, 1976.
3. Dorai C. and Venkatesh S. Computational media aesthetics: finding meaning beautiful. IEEE Multimed., 8(4):10–12, 2001.

4. Marc Davis. Editing out video editing. IEEE Multimed., 10(2):2–12, 2003.
5. Mulhem P., Kankanhalli M.S., Ji Yi., and Hassan H. Pivot vector space approach for audio-video mixing. IEEE Multimed., 10 (2):28–40, 2003.
6. Salway A. and Graham M. Extracting information about emotions in films, In Proc. 9th Int. Conf. on Multimedia Modeling, 2003, pp. 299–302.
7. Sarvas R., Herrarte E., Wilhelm A., and Davis M. Metadata creation system for mobile images. In Proc. 2nd Int. Conf. Mobile Systems, Applications and Services, 2004, pp. 36–48.
8. Smeulders A., Worring M., Santini S., and Gupta A. Content based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell., 22(12):1349–1380, 2000.
9. Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Music information retrieval by detecting mood via computational media aesthetics. In Proc. IEEE/WIC Int. Conf. on Web Intelligence, 2003, pp. 235–241.
10. Zettl H. Sight, Sound, Motion: Applied Media Aesthetics Wadsworth Publishing, Belmont, CA, 1999.

Computational Ontology

▶ Ontology

Computer Human Interaction (CHI)

▶ Human-Computer Interaction

Computer-based Physician Order Entry

▶ Computerized Physician Order Entry

Computer-based Provider Order Entry

▶ Computerized Physician Order Entry

Computer-Interpretable Formalism

▶ Executable Knowledge

Computerized Order Entry (COE)

▶ Computerized Physician Order Entry

Computerized Physician Order Entry

Michael Weiner
Indiana University School of Medicine, Indianapolis, IN, USA

Synonyms

Computer-based physician order entry; Computer-based provider order entry; Computerized provider order entry; Computerized order entry (COE); Physician order entry

Definition

In daily medical practice, physicians routinely create plans of diagnosis and treatment for their patients. These plans typically contain specific, formal orders – directives – that are expected to be implemented by other medical professionals, such as nurses or personnel at laboratories or pharmacies. When such personnel are expected to implement part of a physician's diagnosis or treatment plan, corresponding orders must be created and documented in the patient's medical record. Physicians have traditionally used paper-based charting systems to record medical orders.

Computerized physician order entry (CPOE) is a process by which physicians directly enter medical orders into a computer. CPOE is typically done when the computer is being used to access an electronic health record (EHR), and the physician is creating a treatment plan for a specific patient in a clinical setting.

In many medical institutions, non-physicians such as nurses, dieticians, social workers, pharmacists, therapists, or advanced nurse practitioners can also enter certain types of orders, hence the broader, useful term "*computerized provider order entry.*"

Historical Background

CPOE was first implemented and described in the latter half of the twentieth century. In the US, early reports came from several institutions, including Harvard Medical School and Brigham and Women's Hospital, the US Veterans Health Administration [16,20], Vanderbilt University [8], University of Virginia [12], Indiana University, and Regenstrief Institute for Health Care [14].

In 1994, Sittig and Stead published "Computer-based physician order entry: the state of the art" [17], summarizing many of the early results. Many difficulties were reported regarding leadership, delays, cultural

resistance, high costs, technical support, workflow, and other operational difficulties for end users.

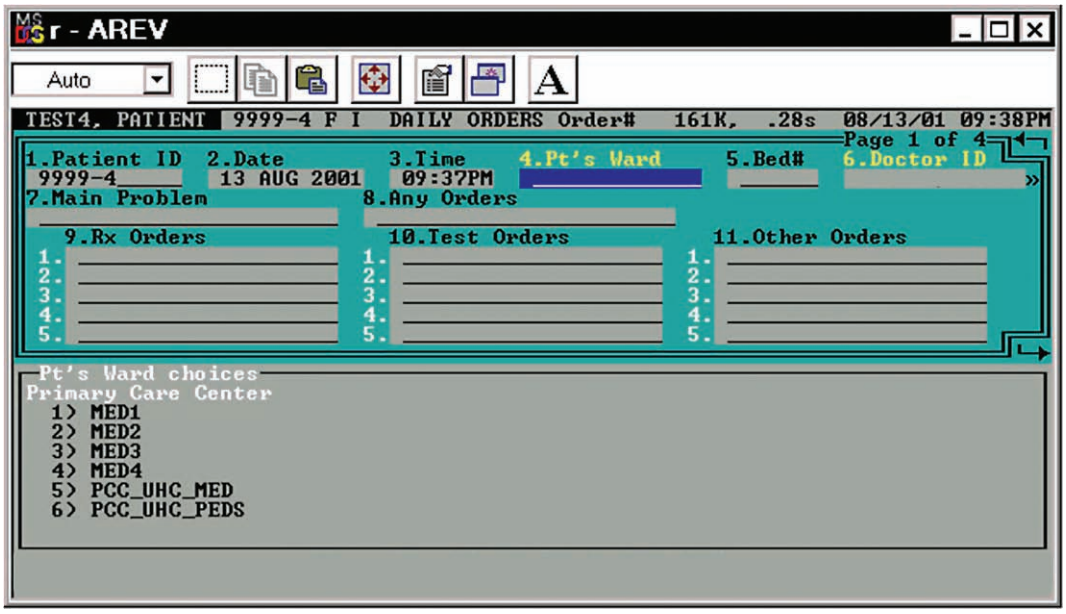
By the end of 2006, CPOE was on the rise, though adoption rates – often correlated with adoption of EHRs – varied widely throughout the world. Most modern EHR systems, whether developed by public, private, or academic institutions, would be expected to include at least some form of CPOE. In the US, only 10% of hospitals had complete availability of CPOE in 2002 [2]. In the United Kingdom, Australia, and New Zealand [21], the fraction is much higher, since use of EHRs exceeds 80% and is approaching 99% among general practitioners in ambulatory practice.

Foundations

CPOE can be used to order a variety of medical services. In some medical institutions with EHRs, CPOE can be used to order any type of medical service and may be required to generate any order. When not required, providers may have the opportunity to select between writing orders in a paper-based chart or an EHR, or clerks or other professionals may perform CPOE on a provider's behalf or direction.

CPOE is performed via a user interface of some kind, though this may occur on a desktop computer, terminal or thin client, personal digital assistant, other portable computer, or other form of computer. A user would typically authenticate himself or herself, identify a patient, and proceed to enter orders, often by navigating through a series of forms, each of which might facilitate a certain type of order, such as for radiology, laboratory, pharmacy, nursing, or referral (see Fig. 1). The layout or interface seen by the user is highly variable and may depend on the developer, personal preferences, underlying database structures, or medical or administrative processes generated in response to specific orders.

Many EHRs allow providers to generate orders and non-order documentation in a single computer session. Non-order documentation may include clinical details such as historical information, measurements, other observations, patient's preferences or directives, or narrative notes or reports. The workflow imposed by a CPOE system should be considered carefully in conjunction with the user's baseline workflow. Greatest success with implementation can often be found when the system does not disrupt the user's own pattern of work.



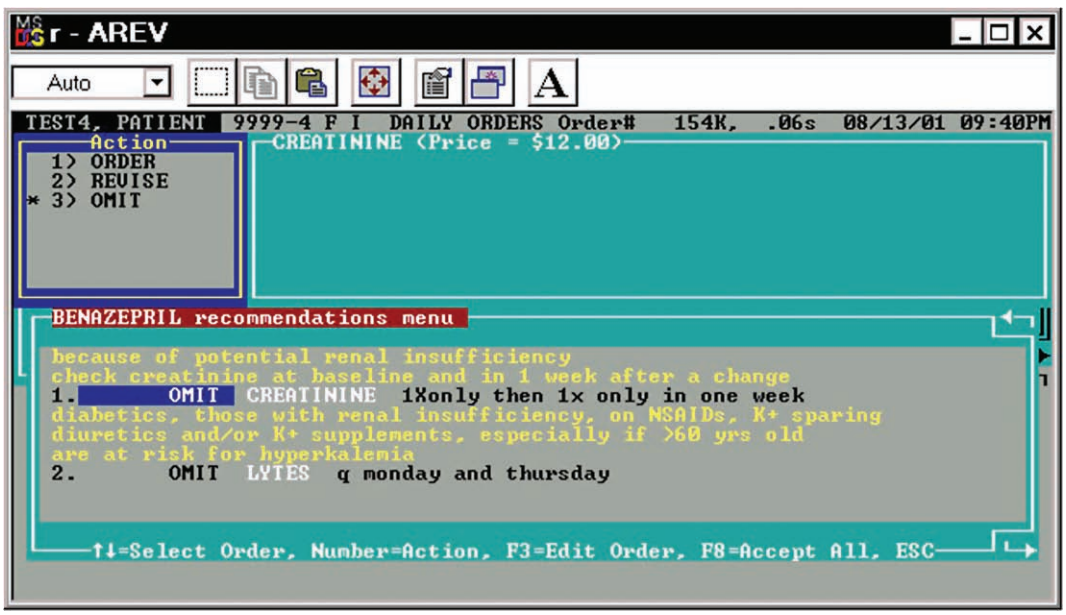
Computerized Physician Order Entry. Figure 1. A user interface for computerized physician order entry. A user would typically authenticate himself or herself, identify a patient, and proceed to enter orders, often by navigating through a series of forms, each of which might facilitate a certain type of order. This screenshot shows form fields that allow entry of orders for drugs, diagnostic tests, and other types of orders (e.g., nursing). The visual separation of types of orders into categories is done primarily for the user's convenience and organization; it may or may not reflect underlying data models.

Once a session is completed, the orders generated lead to action. This may occur via simple printing of the orders or through electronic delivery to remote locations, such as a radiology department, consultant's office, laboratory, or pharmacy. Some orders, such as for prescriptions for drugs or lifestyle changes, may be provided to patients for direct implementation or delivery elsewhere. A session's orders are then typically archived in the EHR. Consistent with traditional medical documentation, orders from CPOE are generated once and cannot be modified or deleted once finalized, though what is being ordered can often later be modified or discontinued with a subsequent order.

CPOE has been developed and implemented for a variety of reasons. Many advantages have been postulated, including rectification of substantial legibility problems with handwritten orders. CPOE systems have the potential to refer to all of a patient's medical history as well as all available medical knowledge, to improve the quality of medical care in real time, at the point of care. One of the most important potentials of CPOE is inclusion of *clinical decision support*, by which the computer can be programmed to suggest tailored orders *de novo* (e.g., for a vaccination recommended by clinical guidelines) or respond to specific orders, such as in the event of a possible drug reaction [11] or contraindication to a procedure (see Fig. 2). CPOE can reduce the rate of certain medication errors by more than half [4]. Removing the healthcare provider from electronic order entry, or moving CPOE outside the point of care, could negate these large potential benefits.

Some institutions are starting with, focus, or limit their computer-based development to electronic prescribing, or "e-prescribing." This is a form of CPOE. E-prescribing is targeted especially because prescribing is frequent, can be targeted by CPOE algorithms, and represents a most common form of medical error [13,18]. In the US, the Institute of Medicine has recommended e-prescribing of drugs, in conjunction with clinical decision support [9].

Medical orders symbolize but also allow and direct the operations behind medical care. By encoding and electronically documenting orders, CPOE improves capabilities to assess and improve quality of care and to conduct clinical research related to diagnosis and treatment. Medical practices, governments, and other authorities can gather and study data from CPOE systems to improve knowledge about how medical



Computerized Physician Order Entry. Figure 2. Example of clinical decision support. In this instance, the user has prescribed benazepril. The software responds, as shown, by prompting the user to decide about ordering blood tests, to monitor for possible side effects from the drug. To enable this capability, the application has been programmed with clinical rules that combine guidelines or medical knowledge with this patient's medical history, evaluation, or treatment.

care is formulated and delivered. CPOE can also facilitate billing processes that depend on orders, such as for certain procedures or drugs. Query languages or systems must be designed to accommodate data models used for CPOE.

Customization of CPOE systems may allow individual providers or groups of providers to create templates or order sets, which are groups of orders often used or often used together. This could save time, improve standardization, and improve care.

CPOE does have costs, risks, and unintended consequences [5]. A CPOE system must be developed thoughtfully and be maintained frequently and regularly, to ensure that it accommodates the latest tests, treatments, and guidelines, both locally and more broadly. If an institution does not stock a particular drug, the system might not allow that drug to be ordered or might at least alert the provider about the issue. Institutional changes and policies that can affect CPOE are frequent and so must lead to corresponding modifications to the CPOE system.

CPOE can increase the time required to generate a medical order [3,15]. Studies of this have reported mixed findings, with increases in some and decreases in others. Increased time for initial learning and ongoing use can cause dissatisfaction among providers and even complete failures of systems. Increasing time to generate or implement orders can have adverse clinical effects. Adverse effects might be expected especially in emergencies or acute care, when life-saving drugs may be needed rapidly. Errors in processing electronic orders could also be expected to lead to adverse effects in at least some cases. In 2005, Koppel et al. reported that one CPOE system design often facilitated medication-related errors, such as by providing inadequate views of medications and increasing inappropriate dosing and incompatible orders [10].

If not implemented effectively, increased use of computers in healthcare might distract providers, causing them to spend less time with patients or decrease patient's satisfaction [7,19]. This could have an adverse effect on patient-provider relationship or patient's health. Provider-to-provider communication might also suffer if appropriate internal communications systems are not used.

Clinical decision support, a key feature of CPOE systems, can backfire by presenting too many or inappropriate alerts. Effective solutions to "alert overload" are not yet well developed or widespread, though some solutions have been discussed [18]. Several recent studies of interventions in decision support have had negative results and require further investigation.

Financial costs of implementing CPOE can be high, especially initially. Developing cost-effectiveness analyses of EHRs and CPOE systems are thus complex, because benefits or harms can occur much later than implementation of a system and later than the time of initial care or clinical presentation.

Moving orders from paper to computers has created situations that require new handling. Computer programs, for example, must know the authority of the authenticated user and whether the user has permission to generate the requested orders. This need also exists with paper systems but is handled in those environments by people, rather than computers. In addition, the use of templates or order sets has not been heavily studied. Templates may in some cases decrease quality of care if they are adopted hastily or used in the wrong setting. In ultimately pooling or sharing data across institutions, it will be important to use standards and customary terminologies to represent orders.

Technical Issues

"Prescription" is another term for medical order, though the term is used conventionally to refer to providing instructions to patients. The structure of a traditional drug prescription provides a useful framework for understanding the primary components of medical orders. Drug orders have a superscription (including timestamp and patient's identifier), inscription (name and amount or strength of ingredient), subscription (formulation or method to prepare), and *signa* ("sig," or directions including route and frequency). Orders of any other type have analogous components, though some may have additional or somewhat different components. CPOE systems should handle components of orders with agility. Below are discussed a few key technical issues that present themselves in the design, study, and implementation of CPOE systems.

Data Models for CPOE. One must consider what data model would best support CPOE. For example, should orders be categorized and, if so, how? Many institutions have found that categories of orders, such as laboratory, consultative, pharmacy, nursing, and radiological, are clinically and informationally logical but may also be necessary from the standpoint of linking disparate data systems. A key goal in the design is the

ability to accommodate future expansion of order types and categories even before those types are developed or identified. This can prompt a somewhat "flat" model, in which the nature, type, or category of an order is a value of a database field, rather than a field or variable itself. Other important aspects of orders that may have implications for the data model are the indication for the order, urgency (i.e., when it should be implemented), and who is expected to implement it.

Standardization of text in an order can be helpful for both accuracy of implementation and research. For example, an order that can have multiple forms, such as "take two 40-mg tablets by mouth twice daily" and "take one 80-mg tablet by mouth every 12 hours" can complicate both clinical care and research. Allowing providers to add narratives or free text is essential for tailoring to patient's needs, but this demands effective handling in data storage and clinical decision support. A system that can standardize the order accurately without hindering the user's experience is desirable but challenging to create. Standardization of terminology used in orders should accommodate query systems. One difficulty is that each type of order – such as for a drug, diet, radiological procedure, or laboratory test – may have unique "domains" or components. For example, only drugs have a dose, yet the dose may need to be a discrete, searchable component of a query. Therefore, an ideal data model can handle all types of orders, as well as new types, but it can also identify and distinguish between various values of key components of orders, regardless of how widespread those components are across various types of orders.

Order sets can often translate directly into a group of individual orders, but users often desire the ability to customize order sets. This may mean maintaining a base of order sets but also the customizations that are unique to each provider or role. Order sets also need to be integrated with any available decision support systems, and many providers seek to share customized order sets with each other. There are thus aspects of order sets that pertain to the system itself and to particular patients, groups of providers, and individual providers. Associated with each order set is also generally information about conditions under which the order set applies, such as a diagnosis, age group, or other criteria found in a clinical guideline – hence the possible need for order sets to be linked to ontologies or terminology systems that in turn link to such guidelines.

Specific types of orders often require further processing or delivery to specific clinical departments. Therefore, the processing needed must be encoded into the system, though it might be a part of the main data engine more than a core part of the data model or record. In any case, if laboratory orders, for example, need to be delivered to the laboratory, then the data system must support this well enough so that all laboratory orders – and only laboratory orders – are processed in this way. Similarly, systems that notify particular professionals or departments should be modular enough that those systems can be updated readily as personnel or even departments change.

An audit trail is important for documentation, accreditation, and quality and safety of care. Included in the data system should be a method to indicate formally not just who created a record and when, but what happened to the record: where it was sent and who accessed it later. Whether this is part of CPOE or the larger records system may depend on the circumstances and design of a system. Many alert systems that stem from orders do not currently provide effective prioritization, so this is an area of important research.

Key Applications

CPOE continues to undergo development and will for the foreseeable future. Due to the difficulty and time required to generate electronic orders, various forms of data entry are being explored. These include transcription with or without voice recognition and input using portable devices, digital pens, or tablets. Due to capability for electronic communication of orders, development is also occurring in remote areas or environments with limited access to healthcare, such as for rural or homebound patients.

CPOE is undergoing significant development especially in the US, the United Kingdom and other parts of Europe, Asia, Australia, and New Zealand. It can be expected to grow throughout the world, even as developing countries create EHR systems.

Future Directions

The largest looming issues for CPOE are how to maximize efficiency of data entry and effectiveness of decision support in the complex environment. There are also unmet needs for CPOE to be linked to access to general medical knowledge. Health policy, attention to quality, patient safety, and reimbursement will likely

gain importance in driving uses of CPOE, especially in areas where its use is currently low. The precise roles, usefulness, impact, and specifications of incentives for healthcare providers to adopt health information technologies such as CPOE are not yet clear.

Cross-references

- ► Clinical Decision Support
- ► Data Acquisition
- ► Electronic Health Record

Recommended Reading

1. Agency for Healthcare Research and Quality. AHRQ National Resource Center for Health Information Technology. 2007. Available online at: http://healthit.ahrq.gov/ (accessed on August 29, 2007).
2. Ash J.S., Gorman P.N., Seshadri V., and Hersh W.R. Computerized physician order entry in U.S. hospitals: results of a 2002 survey. J. Am. Med. Inform. Assoc., 11(2):95–99, 2004.
3. Bates D.W., Boyle D.L., and Teich J.M. Impact of computerized physician order entry on physician time. In Proc. Annual Symp. on Computer Applications in Medical Care, 1994, p. 996.
4. Bates D.W., Leape L.L., Cullen D.J., et al. Effect of computerized physician order entry and a team intervention on prevention of serious medication errors. JAMA, 280(15):1311–1316, 1998.
5. Campbell E.M., Sittig D.F., Ash J.S., Guappone K.P., and Dykstra R.H. Types of unintended consequences related to computerized provider order entry. J. Am. Med. Inform. Assoc., 13(5): 547–556, 2006.
6. Certification Commission for Healthcare Information Technology. 2007. Available online at: http://www.cchit.org/ (accessed on August 29, 2007).
7. Frankel R., Altschuler A., George S., et al. Effects of exam-room computing on clinician-patient communication: a longitudinal qualitative study. J. Gen. Intern. Med., 20(8):677–682, 2005.
8. Geissbuhler A. and Miller R.A. A new approach to the implementation of direct care-provider order entry. In Proc. AMIA Annual Fall Symposium, 1996, pp. 689–693.
9. Institute of Medicine. Crossing the Quality Chasm: A New Health System for the 21st Century. The National Academies Press, Washington, DC, 2001.
10. Koppel R., Metlay J.P., Cohen A., et al. Role of computerized physician order entry systems in facilitating medication errors. JAMA, 293(10):1197–1203, 2005.
11. Kuperman G.J., Bobb A., Payne T.H., et al. Medication-related clinical decision support in computerized provider order entry systems: a review. J. Am. Med. Inform. Assoc., 14(1):29–40, 2007.
12. Massaro T.A. Introducing physician order entry at a major academic medical center: I. Impact on organizational culture and behavior. Acad. Med., 68(1):20–25, 1993.
13. Miller R.A., Gardner R.M., Johnson K.B., and Hripcsak G. Clinical decision support and electronic prescribing systems: a time for responsible thought and action. J. Am. Med. Inform. Assoc., 12(4):403–409, 2005.
14. Overhage J.M., Mamlin B., Warvel J., Warvel J., Tierney W., and McDonald C.J. A tool for provider interaction during patient care: G-CARE. In Proc. Annual Symp. on Computer Applications in Medical Care, 1995, pp. 178–182.
15. Overhage J.M., Perkins S., Tierney W.M., and McDonald C.J. Controlled trial of direct physician order entry: effects on physicians' time utilization in ambulatory primary care internal medicine practices. J. Am. Med. Inform. Assoc., 8(4):361–371, 2001.
16. Payne T.H. The transition to automated practitioner order entry in a teaching hospital: the VA Puget Sound experience. In Proc. AMIA Annual Symposium, 1999, pp. 589–593.
17. Sittig D.F. and Stead W.W. Computer-based physician order entry: the state of the art. J. Am. Med. Inform. Assoc., 1(2): 108–123, 1994.
18. Teich J.M., Osheroff J.A., Pifer E.A., Sittig D.F., and Jenders R.A. Clinical decision support in electronic prescribing: recommendations and an action plan: report of the joint clinical decision support workgroup. J. Am. Med. Inform. Assoc., 12(4):365–376, 2005.
19. Weiner M. and Biondich P. The influence of information technology on patient-physician relationships. J. Gen. Intern. Med., 21(Suppl 1):S35–S39, 2006.
20. Weir C, Lincoln M, Roscoe D, Turner C, and Moreshead G. Dimensions associated with successful implementation of a hospital based integrated order entry system. In Proc. Annual Symp. on Computer Applications in Medical Care, 1994, pp. 653–657.
21. Wells S., Ashton T., and Jackson R. Electronic clinical decision support. 2005. Updated October 2005. Available via Internet at: http://www.hpm.org/survey/nz/a6/2 (accessed on August 29, 2007).

Computerized Provider Order Entry

► Computerized Physician Order Entry

Concept Languages

► Description Logics

Conceptual Data Model

► Semantic Data Model

Conceptual Image Data Model

► Image Content Modeling

Conceptual Model

► Semantic Data Model

Conceptual Modeling

► Semantic Modeling for Geographic Information Systems

Conceptual Modeling for Geographic Information System

► Semantic Modeling for Geographic Information Systems

Conceptual Modeling for Spatio-Temporal Applications

► Semantic Modeling for Geographic Information Systems

Conceptual Schema Design

Alexander Borgida[1], John Mylopoulos[2]
[1]Rutgers University, New Brunswick, NJ, USA
[2]University of Trento, Trento, Italy

Definition

Conceptual schema design is the process of generating a description of the contents of a database in high-level terms that are natural and direct for users of the database. The process takes as input information requirements for the applications that will use the database, and produces a schema expressed in a conceptual modeling notation, such as the Extended Entity-Relationship (EER) Data Model or UML class diagrams. The challenges in designing a conceptual schema include: (i) turning informal information requirements into a *cognitive model* that describes unambiguously and completely the contents of the database-to-be; and (ii) using the constructs of a data modeling language appropriately to generate from the cognitive model a conceptual schema that reflects it as accurately as possible.

Historical Background

The history of conceptual schema design is intimately intertwined with that of conceptual data models (aka semantic data models). In fact, for many years researchers focused on the design of suitable *languages* for conceptual schemas, paying little attention to the design process itself. Jean-Raymond Abrial proposed the *binary semantic model* in 1974 [1], shortly followed by Peter Chen's *entity-relationship model* (ER for short) [4]. Both were intended as advances over logical data models proposed only a few years earlier, and both emphasized the need to model more naturally the contents of a database. The ER model and its extensions were relatively easy to map to logical schemas for relational databases, making EER [9] the first conceptual modeling notation to be used widely by practitioners. On the other hand, Abrial's semantic model was more akin to object-oriented data models that became popular more than a decade later.

The advent of object-oriented software analysis techniques in the late 1980s revived interest in object-oriented data modeling and led to a number of proposals. Some of these, notably OMT [7] adopted many ideas from the EER model. These ideas were consolidated into the *Unified Modeling Language* (UML), specifically UML class diagrams.

The process of designing conceptual schemas by using such modeling languages was not studied until the late 1970s, see for instance [8]. In the early 1980s, the DATAID project proposed a state-of-the-art design process for databases, including conceptual schema design [2].

Throughout this history, research on knowledge representation in Artificial Intelligence (AI) has advanced a set of concepts that overlaps with those of conceptual data models. Notably, semantic networks, first proposed in the 1960s, were founded on the notions of *concept*,

link and *isA* hierarchy (analogously to *entity*, *relationship* and *generalization* for the EER model). The formal treatment of these notations has led to modern modeling languages such as Description Logics, including OWL, for capturing the semantics of web data. In fact, Description Logics have been shown to be able to capture the precise semantics of conceptual modeling notations such as EER diagrams and UML class diagrams.

Another important recent development has been the rise of *ontological analysis*. An ontology is a specification of a conceptualization of a domain. As such, an ontology offers a set of concepts for modeling an application, and foundational ontologies strive to uncover appropriate cognitive primitives with which to describe and critique conceptual modeling notations [5]. Foundational ontologies have been used to analyze the appropriate use of EER constructs [10] and UML [6]. Based on this work, a two-phase perspective is adopted here by distinguishing between the design of a *cognitive model* based on cognitive primitives, and the design of a corresponding *conceptual schema* that is based on the constructs of a conceptual model such as the EER or UML class diagrams.

Foundations

Building a cognitive model. Information requirements for a database-to-be are generally expressed informally, based on multiple sources (e.g., applications/queries that need to be supported, existing paper and computerized systems). Information requirements describe some part of the world, hereafter the *application domain* (or *universe of discourse*). A *cognitive model* is a human conceptualization of this domain, described in terms of cognitive primitives that underlie human cognition. The following are some of the most important among these primitives.

An *object* is anything one may want to talk about, and often represents an *individual* ("my dog," "math422"). Usually, individual objects persist over many states of the application domain. Moreover, they have *qualities* (such as size, weight), and can be related to other individuals by *relations* (e.g., "friendOf," "part of," "between"). Individuals may be concrete (such as "Janet," or "that tree"), abstract (e.g., "the number 12," "cs422"), or even hypothetical (e.g., "Santa," "the king of the USA"). Individuals have a notion of *identity*, allowing them, for example, to be distinguished and counted. For some individuals such as "Janet," identity is an intrinsic notion that distinguishes her from all other objects. *Values* are special individuals whose identity is determined by their structure and properties (e.g., numbers, sets, lists and tuples). The number "7," for example, is the unique number that comes after "6," while "{a, b, c}" is the unique set with elements "a," "b" and "c."

Individuals can be grouped into *categories* (also called *concepts*, *types*), where each category (e.g., "Book") captures common properties of all its instances (e.g., "my DB textbook," "Ivanhoe"). Categories themselves are usefully structured into taxonomies according to their generality/specificity. For instance, "Book" is a specialization of "LibraryMaterial," along with "Journal" and "DVD." Moreover, "Book" has its own specializations, such as "Hardback," "Paperback."

Many categories (e.g., "Person") are *primitive*, in the sense that they don't have a definition. By implication, there is no algorithm to determine whether a given individual object is an instance of such a category – one must be told explicitly such facts. Other categories are *defined*, in the sense that instances can be recognized based on some rule. For example, an instance of "Teenager" can be recognized given an instance of "Person" and its age.

Relations relate two or more objects, for example "book45 is on loan to Lynn," or "Trento is between Bolzano and Verona" and can represent among others a dependence of some sort between these objects. Each relation is characterized by a positive integer n–its *arity*–representing the number of objects being related. In the example above, "onLoanTo" is binary, while "between" is ternary. Predicate logic notation is used to express specific relations between individuals, e.g., "between(Trento,Bolazano,Verona)." Like their individual counterparts, relations can be grouped into relation categories. In turn, relation categories can be organized into subcategory hierarchies: "brotherOf" and "sisterOf" are subcategories of "siblingOf," which in turn is a subcategory of "relativeOf." A binary relation category has a *domain* and a *range*. A cognitive model can specify arbitrary constraints between relations and categories, which describe the valid states ("semantics") of the application domain. Cardinality constraints, for instance, specify upper/lower bounds on how many instances of the range of a relation

can be associated to an instance of the domain, and vice versa.

A subtle complexity arises when one wants to describe information *about* a relation, for example when did it become or ceased to be true. In such cases the modeler can resort to *reification*. For example, the reification of "Trento is between Bolzano and Verona" consists of creating a new individual, "btw73" which is an instance of the category "Between" and is related to its three arguments via three functional relations: "refersTo (btw73,Trento)," "source(btw73,Bolzano)," "destination (btw73,Verona)." Note that this representation allows another instance of "Between," say "btw22," with functions to the same three individuals. To avoid this redundancy, one needs suitable constraints on "Between." One can now model other information about such reified relations, e.g., "believes(yannis,btw73)."

There are several categories of relations that deserve special consideration.

PartOf (with inverse *hasPart*) represents the part-whole relation that allows composite conceptualizations consisting of simpler parts. PartOf actually represents several distinct relations with different formal properties [5]. Most prominent among them are the relations *componentOf* (e.g., cover is a component of book) and *memberOf* (e.g., player member of a team). PartOf is frequently confused with other relations, such as *containment, connectedness, hasLocation*. A useful diagnostic test for this confusion is to check that if A is part of B, and B is damaged, then A is also considered damaged. Note how "love-note placed inside book" fails this test.

The relation between an object and a category is often called *instanceOf*, and the set of all instances of a category are called its *extension*. The *isA* (*subcategory*) relation represents a taxonomic ordering between categories, e.g., "isA(HardcoverBook,Book)." The isA relation is transitive and anti-symmetric and interacts with instanceOf in an important way: if "isA(A, B)" and "instanceOf(x,A)" then "instanceOf(x,B)." Any general statements one associates with a category, apply to all its specializations. For example, "every book has a title" will automatically apply to subcategories ("every HardcoverBook has a title"); this is called *inheritance* of constraints. In many cases, a group of subcategories are mutually *disjoint* (e.g., "Hardcover," "Paperback," are disjoint subcategories of the category "Book"). Sometimes, a set of subcategories *covers* their common parent category in the sense that every instance of the latter is also an instance of at least one subcategory (for example, "Male," "Female" cover "Person"). When a collection of subcategories partitions a parent category, it is often because some relation (e.g., "gender") takes on a single value from an enumerated set (e.g., {"M," "F"}).

When building an isA hierarchy, it is useful to start by building a backbone consisting of primitive, disjoint concepts that describe the basic categories of individuals in the application domain. Some categories can be distinguished from others by their *rigidity* property: an instance of the category remains an instance throughout its lifetime. For example, "Person" is such a category (in the sense of "once a person, always a person"), but "Student" is not. Once a backbone taxonomy has been constructed, other categories, such as role categories, can be added to the hierarchy as specializations of the categories. Welty and Guarino [11] present a principled approach to the construction of taxonomies. A useful rule for building meaningful isA hierarchies is to ensure that the children of any category are all at the same level of granularity. A leaf category in an isA hierarchy should be further refined if it has instances that can be usefully grouped into sub-categories, which participate in new relations or which are subject to new constraints.

Categories may also be seen as objects, and can then be grouped into meta-categories that capture common meta-properties of their instances. For example, the meta-category "LibraryMaterialType" has instances such as "Book," "Hardback" and "DVD," and may have an integer-valued meta-property such as "number-on-order."

In most worlds there are not just enduring objects but also occurrences of events, activities, processes, etc., such as borrowing, renewing or returning a book. These phenomena are called *events*. An event can be described from a number of perspectives: First, there are the *participants* in an event: the material borrowed, the patron who did the borrowing, the library from which the material was borrowed, the date when the material is required to be returned, etc. Often these participants are given names describing the *role* they play in the event: "borrower," "lender," "due date." Second, every event takes place in time, so its temporal aspects can be represented: starting and possibly ending time if it is of a long duration, cyclic nature, etc. Third, events may also have parts – the sub-events that need to take place (e.g., taking the

book to the counter, proffering the library card,...). There may also be special relations, such as causality or temporal precedence that hold among events.

In database modeling, one often ignores events because they are transient, while the database is supposed to capture persistence. However, events are in fact present in the background: relations between objects other than "partOf" are usually established or terminated by events. The "onLoan" relation, for example, is created by a "borrow" event and terminated by a "return" event. And values for many qualities (e.g., the size or weight of an object) are established through events representing acts of observation. As a result, relations often carry information about the events. Thus information about the "borrow" activity's participants is present in the arguments of the "onLoan" relation. And since "renew" shares the main participants of "borrow," its traces can also be attached to "onLoan," through an additional temporal argument, such as "lastRenewedOn."

Note that it is application requirements that determine the level of detail to be maintained in a cognitive model. For example, whether or not one records the time when the borrowing and renewal occurred, or whether it is sufficient to have a "dueDate" attribute on the "onLoan" relation. In addition, the details of the subparts of "borrow" will very likely be suppressed. On the other hand, semantic relations between events, such as the fact that a book renewal can only occur after the book has been borrowed, do need to be captured.

Because databases often have multiple sets of users, there may be several conflicting interpretations of terms and information needs. The important process of reconciling such conflicts, known in part as "view integration" is not addressed in this entry.

From a cognitive model to a conceptual schema. The above account has focused on the construction of a model that captures information requirements in terms of cognitive primitives, with constraints expressed in a possibly very rich language. Every effort was made to keep the modeling and methodology independent of a particular modeling language. Next, one must tackle the problem of producing a conceptual schema expressed in some particular formal notation, in this case the EER. Comparable discussions apply if the target was UML class diagrams, or even OWL ontologies.

The basic mapping is quite straightforward: *categories of individuals* in the conceptual model are mapped to ER *entity sets*; relation categories in the conceptual model are modeled directly as relationship sets, with the participating entities playing (potentially named) roles in the relationship set. Qualities and values related to individuals by binary relations, or appearing as arguments of relations become *attributes*. Cardinality constraints of the cognitive model are mapped directly to the conceptual schema.

One complex aspect of EER schema development is the definition of *keys* consisting of one or more attributes that uniquely distinguish each individual instance of an entity set. Moreover, the values of these attributes must be stable/unchanging over time. For example, "isbnNr" or "callNumber" would be a natural key attributes for "Book." Globally unique identifiers are actually relatively rare. Instead, entities are often identified with the help of intermediate relationships (and attributes). For example, "BookCopy," has attribute "copyNr," which surely does not identify a particular book copy; but if "BookCopy" is represented as a *weak entity*, related to "Book" via the "copyOf" identifying relationship then "BookCopy" will have a composite identifier. In other situations where one would need a large set of attributes to identify an entity, and especially if these attributes are not under the control of database administrators, the designer may introduce a *new* surrogate entity attribute specifically for identification purposes (e.g., "studentId" for the entity set "Student").

Sub-categories, with possible disjoint and coverage constraints, are represented in a direct manner in EER since it supports these notions. Significantly, in most version of EER key attribute(s) can only be specified for the top-most entity in a hierarchy, and must then be inherited by all sub-entities. Therefore one cannot have "Employee" with key "ssn," while sub-class "TeachingAssistant" has key "studentId." The reason for this is to avoid multiple ways of referring to what would otherwise be the same individual.

Since the EER model supports n-ary relationships, reified relationships are normally only required in case the model needs to make "meta" statements about relationships, e.g., recording that a particular loan was verified by a clerk. In variants of the EER that allow aggregate relationships, this is modeled by relating entity "Clerk" via relationship "hasVerified" to the aggregate representing the "lentTo" relationship. In impoverished variants of EER that do not support aggregates, this can be encoded using weak entities that reify the relationship.

The EER notation (in constrast to UML, say), does not provide support for distinguishing *partOf* relationships, nor for relationship hierarchies. However, the designer may encode these using reified relationships.

Conceptual Schema Design in Practice

There is a plethora of commercial tools for conceptual schema design based on the EER model or UML class diagrams. These support the drawing, documenting and layout of conceptual schemas, and even the automatic generation of standard logical (relational) schemas to support database design.

More advanced tools, such as icom (http://www.inf.unibz.it/~franconi/icom/), allow not just the drawing of conceptual schemas, but their translation to formal logic. Advantages of such tools include (i) precise, formal semantics for all the constructs of the conceptual model; (ii) the ability to check the conceptual schema for consistency–an issue which is particularly interesting in the case of *finite models*; (iii) the ability to augment a conceptual schema with constraints expressed in the underlying logic.

Key Applications

Conceptual schema design is the first–and for many the most important–step in the design of any database.

Future Directions

One of the major research challenges of the new century is data integration, and the semantics of data captured in conceptual models has been shown to form a useful foundation for merging multiple information sources. In this context, one can imagine using a conceptual schema as the mediating schema to access different database sources.

The advent of the Web has made databases a public resource that can be shared world-wide. In such a setting, where the user may know nothing about a database she is accessing, the issues that dominate are (i) encapsulating the semantics of data along with the data, so that the user can interpret the data; (ii) ensuring data quality, so that the user can determine whether the data are suitable for her purposes; (iii) ensuring compliance with privacy, security and governance regulations. In turn, these new requirements are going to redefine the scope of database design in general and conceptual schema design in particular.

Specifically, extended conceptual modeling languages and conceptual schema design techniques are envisioned where quality, privacy, security and governance policies can be expressed explicitly and can be accommodated during the design process to produce conceptual schemas that are well-suited to address such concerns. We also envision extensions to conceptual modeling languages that introduce primitive concepts for modeling dynamic, intentional and social aspects of an application domain. These aspects constitute an important component of the semantics of any database and a prerequisite for dealing with data quality and regulation compliance.

Cross-references

- ▶ Conceptual Data Model
- ▶ Description Logics
- ▶ Logical Schema Design
- ▶ Schema Integration
- ▶ Semantic Data Model

Recommended Reading

1. Abrial J-R. Data Semantics. In Data Management Systems, K. Koffeman North-Holland, Amsterdam, 1974.
2. Atzeni P., Ceri S., Paraboschi S., and Torlone R. Database Systems: Concepts, Languages & Architectures. McGraw Hill, New York, 1999.
3. Batini C., Ceri S., and Navathe S. Conceptual Database Design. Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA, 1991.
4. Chen P.P. The entity-relationship model – toward a unified view of data. ACM Trans. Database Syst., 1(1):9–36, 1976.
5. Gangemi A., Guarino A., Masolo C., Oltramari A., and Schneider L. Sweetening ontologies with DOLCE. In Proc. 12th Int. Conf. Knowledge Eng. and Knowledge Management: Ontologies and the Semantic Web, 2002, pp. 166–181.
6. Guizzardi G., Herre H., and Wagner G. Towards Ontological Foundations for UML Conceptual Models. In Proc. Confederated Int. Conf. DOA, CoopIS and ODBASE, 2002, pp. 1100–1117.
7. Rumbaugh J., Blaha M., Premerlani W., Eddy F., and Lorensen W. Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, NJ, 1991.
8. Sakai H. Entity-relationship approach to the conceptual schema design. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1980, pp. 1–8.
9. Teorey T., Yang D., and Fry J. A logical design methodology for relational databases using the extended entity-relationship model. ACM Comput. Surv., 18(2):197–222, 1986.
10. Wand Y., Storey V.C., and Weber R. An ontological analysis of the relationship construct in conceptual modeling. ACM Trans. Database Syst., 24(4):494–528, 1999.
11. Welty C. and Guarino N. Supporting ontological analysis of taxonomic relationships. Data Knowl. Eng., 39:51–74, 2001.

Conceptual Schemas

► Resource Description Framework (RDF) Schema (RDFS)

Concurrency Control

► Concurrency Control – Traditional Approaches
► Correctness Criteria Beyond Serializability
► Performance Analysis of Transaction Processing Systems

Concurrency Control – Traditional Approaches

Gottfried Vossen
University of Münster, Münster, Germany

Synonyms

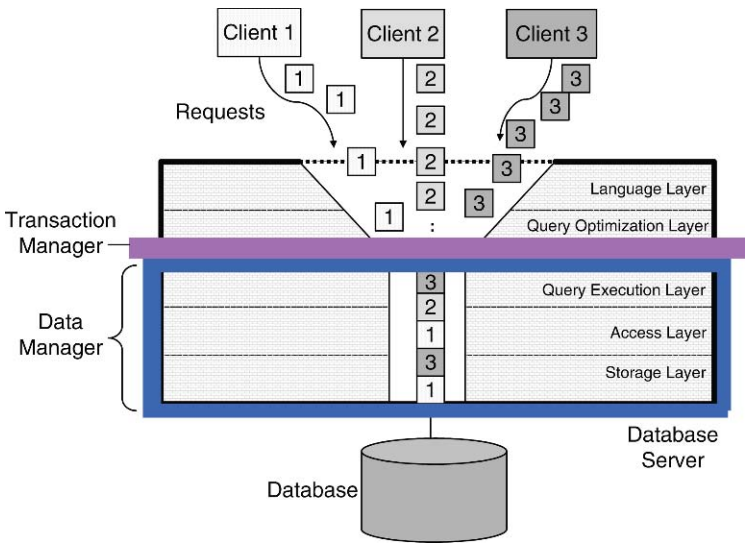
Concurrency control; Transaction execution; Scheduling; Page locking; Two-phase-locking

Definition

The core requirement on a transactional server is to provide the ACID properties for transactions, which requires that the server includes a *concurrency control* component as well as a *recovery component*. Concurrency control essentially guarantees the isolation properties of transactions, by giving each transaction the impression that it operates alone on the underlying database and, more technically, by providing serializable executions. To achieve serializability, a number of algorithms have been proposed. Traditional approaches focus on the read-write model of transactions and devise numerous ways for correctly scheduling read-write transactions. Most practical solutions employ a variant of the two-phase locking protocol.

Key Points

Concurrency control for transactions is done by the *transaction manager* of a database system and within that component by a *scheduler*. The scheduler implements an algorithm that takes operations from active transactions and places them in an interleaving or *schedule* that must obey the correctness criterion of serializability. As illustrated in Fig. 1, which indicates the "positioning" of a transaction manager within the multi-layer architecture of a database system, the scheduler receives steps from multiple transactions, one after the other, and tries to attach them to the schedule already output. This is possible if the new schedule is still serializable; otherwise, the scheduler can block or even reject a step (thereby causing the respective transaction to abort) [1,5,6]. The algorithm a scheduler follows must be such that serializability can be tested



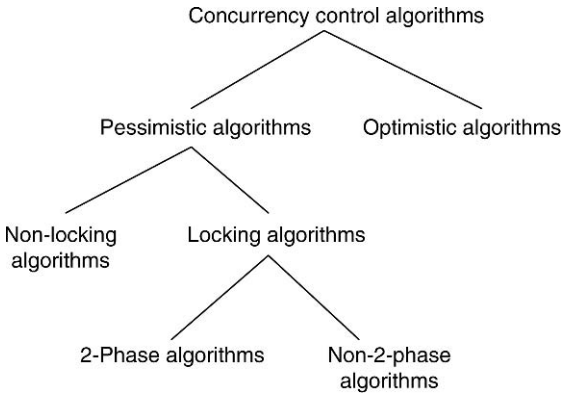
Concurrency Control – Traditional Approaches. Figure 1. Illustration of a transaction scheduler.

on the fly; moreover, it must be very efficient so that high throughput rates (typically measured in [committed] transactions per minute) can not only be achieved, but even be guaranteed.

Classification of Approaches

Scheduling algorithms for database transactions can be classified as "traditional" if they concentrate on scheduling read-write transactions; non-traditional schedulers takes semantic information into account which is not available at the syntactic layer of read and write page operations. Traditional schedulers generally fall into two categories: A scheduler is *optimistic* or aggressive if it mostly lets steps pass and rarely blocks; clearly, this bears the danger of "getting stuck" eventually when the serializability of the output can no longer be guaranteed. An optimistic scheduler is based on the assumption that conflicts between concurrent transactions are rare; it will only test from time to time whether the schedule produced so far is still serializable, and it takes appropriate measures if the schedule is not.

On the other hand, a scheduler is *pessimistic* or conservative if it mostly blocks (upon recognizing conflicts); in the extreme yet unlikely case that all transactions but one have been blocked, the output would become a serial schedule. This type of scheduler is based on the assumption that conflicts between transactions are frequent and therefore need to be constantly observed. Pessimistic schedulers can be *locking* or *non-locking* schedulers, where the idea of the former is to synchronize read or write access to shared data by using locks which can be set on and removed from data items on behalf of transactions. The intuitive meaning is that if a transaction holds a lock on a data object, the object is not available to transactions that execute concurrently. Non-locking schedulers replace locks, for example, by timestamps that are attached to transactions. Among locking schedulers, the most prominent protocol is based on a *two-phase* approach (and hence abbreviated two-phase locking or 2PL) in which, for each transaction, a first phase during which locks are obtained is strictly separated from a second phase where locks can only be released. Non-two-phase schedulers replace the two-phase property, for example, by an order in which transactions may access data objects. Figure 2 summarizes the major classes of concurrency control protocols. Beyond the approaches shown in Fig. 2, the concurrency control problem can even be broken into two subproblems,



Concurrency Control – Traditional Approaches.
Figure 2. Overview of concurrency control protocol classes.

which could then be solved individually by possibly distinct protocols: (i) read operations are synchronized against write operations or vice versa; (ii) write operations are synchronized against other write operations, but not against reads. If these synchronization tasks are distinguished, a scheduler can be thought of as consisting of two components, one for each of the respective synchronization tasks. Since the two components need proper integration, such a scheduler is called a hybrid scheduler. From an application point of view, most classes of protocols surveyed above, including the hybrid ones, have not achieved great relevance in practice. Indeed, 2PL is by far the most important concurrency control protocol, since it can be implemented with low overhead, it can be extended to abstraction levels beyond pure page operations, and it has always outperformed any competing approaches [2–4].

Cross-references

- ▶ B-Tree Locking
- ▶ Distributed Concurrency Control
- ▶ Locking Granularity and Locks Types
- ▶ Performance Analysis of Transaction Processing Systems
- ▶ Serializability
- ▶ Snapshot Isolation
- ▶ Two-Phase Locking

Recommended Reading

1. Bernstein P.A., Hadzilacos V., and Goodman N. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, MA, 1987.

2. Bernstein P.A. and Newcomer E. Principles of Transaction Processing for the Systems Professional. Morgan Kaufmann, San Francisco, CA, 1997.
3. Claybrook B. OLTP – Online Transaction Processing Systems. Wiley, New York, 1992.
4. Gray J. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA, 1993.
5. Papadimitriou C.H. The Theory of Database Concurrency Control. Computer Science, Rockville, MD, 1986.
6. Weikum G. and Vossen G. Transactional Information Systems – Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann, San Francisco, CA, 2002.

Concurrency Control and Recovery

► Transaction Management

Concurrency Control Manager

Andreas Reuter[1,2]
[1]EML Research GmbH Villa Bosch, Heidelberg, Germany
[2]Technical University Kaiserslautern, Kaiserslautern, Germany

Synonyms

Concurrency control manager; Lock manager; Synchronization component

Definition

The concurrency control manager (CCM) synchronizes the concurrent access of database transactions to shared objects in the database. It is responsible for maintaining the guarantees regarding the effects of concurrent access to the shared database, i.e., it will protect each transaction from anomalies that can result from the fact that other transactions are accessing the same data at the same time. Ideally, it will make sure that the result of transactions running in parallel is identical to the result of some serial execution of the same transactions. In real applications, however, some transactions may opt for lower levels of synchronization, thus trading protection from side effects of other transactions for performance. The CCM is responsible for orchestrating all access requests issued by the transactions such that each transaction receives the level of protection it has asked for. The CCM essentially implements one of a number of different synchronization protocols, each of which ensures the correct execution of parallel transactions, while making different assumptions regarding prevalent access patterns, frequency of conflicts among concurrent transactions, percentage of aborts, etc. The protocol that most CCMs are based on is using locks for protecting database objects against (inconsistent) parallel accesses; for that reason, the CCM is often referred to as the "lock manager." Some CCMs distinguish between multiple versions of a data object (e.g., current version, previous version) in order to increase the level of parallelism [2].

Key Points

The CCM monitors all access requests issued by higher-level components, be it to the primary data (tuples, records), or to access paths, directory data, etc. on behalf of the database transactions. This information (transaction **X** accesses object **O** in order to perform action **A** at time **T**) is employed in different ways, depending on the synchronization protocol used. In case of locking protocols, each access request has to be explicitly granted by the CCM. If no conflict will arise by performing action A on object O, the access request is granted. If, however, a conflict is detected (e.g., A is an update request, and some other transaction Y is already updating object O), the request is not granted, and the CCM will record the fact that X has to wait for the completion of transaction Y. In case of optimistic protocols, the requests are granted right away, but when X wants to commit, all its accesses are checked for conflicts with accesses performed by other transactions that are either still running or have committed while X was active. In those situations the time T of the access request is relevant. So the basic data structure maintained by the CCM is a table of accesses/access requests. For locking protocols, the CCM will also maintain a list of which transaction waits for the completion of which other transaction(s). That list is tested by the CCM for deadlocks. If a deadlock or, in case of optimistic protocols, a conflict is detected, the CCM decides which transaction will be aborted. It then informs the transaction manager, who will initiate the abort; rollback of the operations is performed by the recovery manager [1].

Cross-references

► Degrees of Consistency
► Dependency

- ▶ Isolation
- ▶ Locking
- ▶ Scheduling
- ▶ Synchronization

Recommended Reading

1. Gray J. and Reuter A. Transaction Processing – Concepts and Techniques. Morgan Kaufmann, San Mateo, 1993.
2. Weikum G. and Vossen G. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control. Morgan Kaufmann, San Mateo, 2001.

Condition Event Nets

- ▶ Petri Nets

Conditional Branching

- ▶ OR-Split

Conditional Routing

- ▶ OR-Split

Conditional Tables

Gösta Grahne
Concordia University, Montreal, QC, Canada

Synonyms

C-tables; Extended relations

Definition

A conditional table [4] generalizes relations in two ways. First, in the entries in the columns, variables, representing unknown values, are allowed in addition to the usual constants. The second generalization is that each tuple is associated with a condition, which is a Boolean combination of atoms of the form $x = y$, $x = a$, $a = b$, for x, y null values (variables), and a, b constants. A conditional table essentially represents an existentially quantified function free first order theory.

Formally, let *con* be a countably infinite set of constants, and *var* be a countably infinite set of variables, disjoint from *con*. Let U be a finite set of attributes, and $R \subseteq U$ a relational schema. A tuple in a c-table over R is a mapping from R, and a special attribute, denoted φ, to $con \cup var \cup \beta$, where β is the set of all Boolean combinations of equality atoms, as above. Every attribute in R maps to a variable or a constant, and φ maps to β. In a multirelational database schema, there are multi-tables, meaning in effect that variables can be shared between tables (just as constants are). An example of a 2-multitable is shown below. The conditions φ are all *true*, and it so happens that the two tables do not share any variables.

T_1	A	B	C	φ
	a	*b*	*x*	*true*
	e	*f*	*g*	*true*

T_2	B	C	D	φ
	y	*c*	*d*	*true*

Key Points

It is now possible to extend the complete set of regular relational operators $\{\pi,\sigma,\bowtie,\cup,-,\rho\}$ to work on c-tables. To distinguish the operators that apply to tables from the regular ones, the extended operators are accented by a dot. For instance, c-table join is denoted $\dot{\bowtie}$. The extended operators work as follows: projection $\dot{\pi}$ is the same as relational projection, except that the condition column φ can never be projected out. Selection $\dot{\sigma}_{A=a}(T)$ retains all tuples t in table T, and conjugates the condition $t(A) = a$ to $t(\varphi)$. A join $T \dot{\bowtie} T'$ is obtained by composing each tuple $t \in T$ with each tuple $t' \in T'$. The new tuple $t \cdot t'$ has condition $t(\varphi) \wedge t'(\varphi) \wedge \delta(t,t')$, where condition $\delta(t,t')$ states that the two tuples agree on the values of the join attributes. The example below serves as an illustration of this definition. The union $\dot{\cup}$ is the same as relational union, and so is renaming $\dot{\rho}$, except that the φ-column cannot be renamed. Finally, the set difference, say $T \dot{-} T'$ is obtained by retaining all tuples $t \in T$ and conjugating to them the condition stating that the tuple t differs from each tuple t' in T'. A tuple t differs from a tuple t', if it differs from t' in at least one column.

Let T_1 and T_2 be as in the figure above. The three c-tables in the figure below, illustrate the result of evaluating $\dot{\sigma}_{C=g}(T_1)$, $T_1 \dot{\bowtie} T_2$, and $\dot{\pi}_{BC}(T_2) \dot{-} \dot{\pi}_{BC}(T_1)$, respectively.

A	B	C	φ
a	*b*	*x*	$x = g$
e	*f*	*g*	$g = g$

A	B	C	D	φ
a	*y*	*x*	*d*	$b = y \wedge x = c$
e	*y*	*g*	*d*	$f = y \wedge g = c$

B	C	φ
y	*c*	$(y \neq b \vee c \neq x) \wedge (y \neq f \vee c \neq g)$

Note that the second tuple in the first c-table has a tautological condition. Likewise, since any two constants differ, the condition of the second tuple in the middle c-table is contradictory.

So far, nothing has been said about what the c-tables mean. In the *possible worlds* interpretation, an incomplete database is a usually infinite *set* of ordinary databases, one of which corresponds to the actual (unknown) database. Considering c-tables, they serve as finite representations of sets of possible databases. One (arbitrary) such database is obtained by instantiating the variables in the c-table to constants. Each occurrence of a particular variable is instantiated to the same constant. Formally, the instantiation is a valuation $\nu : con \cup var \rightarrow con$, that is identity on the constants. Valuations are extended to tuples and conditions tables in the obvious way, with the caveat that given a particular valuation ν, only those tuples t for which $\nu(t(\varphi)) \equiv true$, are retained in $\nu(T)$. Consider for instance the first table above. For those valuations ν, for which $\nu(x) = g$, there will be two tuples in $\nu(T)$, namely (a,b,g) and (e,f,g). For valuations ν', for which $\nu(x) \neq g$, there will only be the tuple (e,f,g) in $\nu'(T)$.

The remarkable property of c-tables is that for all c-tables T and relational expressions E, it holds that $\nu(\dot{E}(T)) = E(\nu(T))$ for all valuations ν. In other words, the extended algebra is a Codd sound and complete inference mechanism for c-tables. Furthermore, c-tables are closed under relational algebra, meaning that the result of applying any relational expression on any (schema-wise appropriate) c-table can be represented as another c-table. The extended algebra actually computes this representation, as was seen in the example above.

Needless to say, all of this comes with a price. Testing whether a c-table is satisfiable, that is, whether there exists at least one valuation ν, such that $\nu(T) \neq \emptyset$ is an NP-complete problem [2]. Furthermore, even if one starts with a simple c-table where all variables are distinct, and all conditions are *true*, applying even a monotone relational expression to such a c-table can result in quite a complex table, so here again [2] satisfiability of the resulting table is NP-complete [2]. To make matters even worse, testing containment of c-tables is Π_2^p-complete. A c-table T is contained in a c-table T', if every for every valuation ν of T, there exists a valuation ν' of T', such that $\nu(T) = \nu(T')$. Nonetheless, c-tables possess a natural robustness. For instance, it has been shown [1,3] that the set of of possible databases defined by a set of materialized views, can be represented as a c-table.

Cross-references

- ▶ Certain (and Possible) Answers
- ▶ Incomplete Information
- ▶ Maybe answer
- ▶ Naive tables

Recommended Reading

1. Abiteboul S., Duschka O.M. Complexity of Answering Queries Using Materialized Views. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 254–263.
2. Abiteboul S., Kanellakis P.C., Grahne G. On the Representation and Querying of Sets of Possible Worlds. Theor. Comput. Sci., 78(1):158–187, 1991.
3. Grahne G., Mendelzon A.O. Tableau Techniques for Querying Information Sources through Global Schemas. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 332–347.
4. Imielinski T., Lipski W. Jr. Incomplete Information in Relational Databases. J. ACM, 31(4):761–791, 1984.

Confidentiality Protection

- ▶ Statistical Disclosure Limitation For Data Access

Conflict Serializability

► Two-Phase Locking

Conjunctive Query

Val Tannen
University of Pennsylvania, Philadelphia, PA, USA

Synonyms

SPC query; Horn clause query

Definition

Conjunctive queries are first-order queries that both practically expressive and algorithmically relatively tractable. They were studied first in [2] and they have played an important role in database systems since then.

As a subset of the relational calculus, conjunctive queries are defined by formulae that make only use of atoms, conjunction, and existential quantification. As such they are closely related to Horn clauses and hence to logic programming. A single Datalog rule can be seen as a conjunctive query [1].

Optimization and reformulation for various purposes is quite feasible for conjunctive queries, as opposed to general relational calculus/algebra queries. The equivalence (and indeed the *containment*) of conjunctive queries is decidable, albeit NP-complete [1].

Key Points

This entry uses terminology defined in the entry Relational Calculus.

Conjunctive queries are first-order queries of a particular form: $\{\langle e_1,...,e_n\rangle \mid \exists x_1...x_m\psi\}$ where ψ is an (equality-free) conjunction of relational atoms, i.e., atoms of the form $R(d_1,...,d_k)$ (where $d_1,...,d_k$ are variables of constants). In addition, it is required that any variable among $e_1,...,e_n$ must also occur in one of the relational atoms of ψ. This last condition, called *range restriction* [3] is necessary for *domain independence*, e.g., consider $\{(x, y)|R(x)\}$, and, in fact, it is also sufficient.

The semantics of a conjunctive query in an instance $\mathcal{I}$, as a particular case of first-order queries, involves assignments μ defined on the variables among $e_1,...,e_n$ such that $\mathcal{I}, \mu \models \exists\, x_1...x_m\psi$. Note however that this is the same as extending μ to a *valuation* v defined in addition on $x_1...x_m$ and such that $\mathcal{I}, v \models \psi$. Since ψ is a conjunction of relational atoms, this amounts to v being a *homomorphism* from ψ seen as a relational instance (the *canonical instance* associated to the query) into $\mathcal{I}$. This simple observation has many useful consequences, including some that lead to the decidability of equivalence (containment). It also leads to an alternative way of looking at conjunctive queries, related to logic programming. Here is an example of a conjunctive query in both relational calculus form and in a Prolog-like, or "rule-based," formalism, also known as a *Datalog rule* [1,3]:

$$\{(x, c, x) \mid \exists y\ R(c, y) \wedge S(c, x, y)\} ans(x, c, x) :- R(c, y), S(c, x, y)$$

In the spirit of rule-based/logic programming, the output tuple of a conjunctive query is sometimes called the "head" of the query and the atom conjunction part the "body" of the query. So far, this discussion has considered only the class CQ of conjunctive without equality in the body. The class $CQ^=$ which allows equalities in the body defines essentially the same queries but there are a couple of technical complications. First, the range restriction condition must be strengthened since, for example, $\{(x, y)|\exists z\ R(x) \wedge y = z\}$ is domain dependent. Therefore, for $CQ^=$ it is required that any variable among $e_1,...,e_n$ must equal, as a consequence of the atomic equalities in ψ, some constant, or some variable that occurs in one of the relational atoms of ψ.

$CQ^=$ has the additional pleasant property (shared, in fact, with the full relational calculus) that query heads can be restricted to consist of just distinct variables.

Clearly $CQ \subseteq CQ^=$. The converse is "almost true." It is possible to get rid of equality atoms in a conjunctive query iff the query is *satisfiable* i.e., there exists some instance on which the query returns a nonempty answer. All the queries in CQ are satisfiable (take the canonical instance). Queries in $CQ^=$ are satisfiable iff the equalities in their body do not imply the equality of distinct constants. Thus, for conjunctive queries (of both kinds) satisfiability is decidable, as opposed to general first-order queries. Now, any satisfiable query in $CQ^=$ can be effectively translated into an equivalent query in CQ.

The conjunctive queries correspond to a specific fragment of the relational algebra, namely the fragment

that uses only the selection, projection, and cartesian product operations. This fragment is called the *SPC algebra*. There is an effective translation that takes every conjunctive query into an equivalent SPC algebra expression. There is also an effective translation that takes every SPC algebra expression into an equivalent conjunctive query.

Via the translation to the SPC algebra it can be seen that conjunctive queries correspond closely to certain SQL programs. For example, the $CQ^{=}$ query $ans(x, y) :- R(x, z), x = c, S(x, y, z)$ corresponds to

```
select r.1, s.2
from R r, S s
where r.1=c and s.1=r.1 and r.3=s.2
```

Such SQL programs, in which the "where" clause is a conjunction of equalities arise often in practice. So, although restricted, conjunctive queries are important.

Cross-references

- ▶ Datalog
- ▶ Relational Algebra
- ▶ Relational Calculus

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of Databases: The Logical Level. Addison Wesley, Reading, MA, 1994.
2. Chandra A.K. and Merlin P.M. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In Proc. 9th Annual ACM Symp. on Theory of Computing, 1977, pp. 77–90.
3. Ullman J.D. Principles of Database and Knowledge-Base Systems, Vol. I. Computer Science, Rockville, MD, 1988.

Connection

Sameh Elnikety
Microsoft Research, Cambridge, UK

Synonyms

Database socket

Definition

A connection is a mechanism that allows a client to issue SQL commands to a database server. In a typical usage, the client software opens a connection to the database server, and then sends SQL commands and receives responses from the server.

To open a connection, the client specifies the database server, database name, as well as the client's credentials. Opening the connection includes a handshake between the client software and the database server. The client sends its credentials, for example in the simplest form a user name and password. The server examines the credentials to authorize the connection. Further information may also be negotiated such as the specific protocol and data encoding.

Key Points

Handling and servicing connections is an important part of database servers because connections are the main source of concurrency.

Database servers limit the number of connections they can accept and may provide differentiated service to connections from high priority clients (e.g., from database administrators).

Connections are implemented using inter-process (e.g. pipes) or remote (e.g., TPC sockets) communication mechanisms. Database vendors and third-party providers supply libraries that client programs use to open connections to database servers. Several standards have emerged such as ODBC (Open Database Connectivity) [2], JDBC (Java Database Connectivity) [3], and ADO.NET (data access classes in Microsoft.NET platform) [1].

When client software uses a database system extensively, it employs a connection pool to reuse a group of open connections, allowing multiple concurrent SQL commands. Using a connection pool avoids closing and reopening connections, as well as opening too many connections that tie up resources at both ends of the connection.

Cross-references

- ▶ Session

Recommended Reading

1. Adya A., Blakeley J., Melnik S., and Muralidhar S. Anatomy of the ADO.NET entity framework. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 877–888.
2. Data Management: SQL Call Level Interface (CLI), Technical Standard C451–15/10/1993, The Open Group.
3. Sun Microsystems, Java Database Connectivity. Available at: http://java.sun.com/javase/technologies/database/

Connectionist Model

- ▶ Neural Networks

Consistency in Peer-to-Peer Systems

► Updates and Transactions in Peer-to-Peer Systems

Consistency Models For Replicated Data

Alan Fekete
University of Sydney, Sydney, NSW, Australia

Synonyms

Replica consistency; Memory consistency

Definition

When a distributed database system keeps several copies or replicas for a data item, at different sites, then the system may ensure that the copies are always consistent (that is, they have the same value), or the system may allow temporary discrepancy between the copies. Even if the copies are not the same, the algorithms that manage the data may be able to hide the discrepancies from clients. A consistency model defines the extent to which discrepancies can exist or be observed, between the copies. If the system offers a strong consistency model, then clients will not be aware of the fact that the system has replicated data, while a weak consistency model requires more careful programming of the clients, so they can cope with the discrepancies they observe.

Historical Background

Most replication research in the 1970s aimed to provide the illusion of an unreplicated database offering serializability. In the early 1980s, Bernstein and colleagues formalized this notion as a strong consistency model [2].

The late 1980s and early 1990s focused on systems that offered weak consistency in various definitions. Eventual consistency was introduced in the work of Demers et al. [3], while consistency models in which reads might see stale values were also explored by several groups [1,5].

Since 2000, a new strong consistency model, one-copy SI, was introduced [4], and it has attracted much attention.

Foundations

There are many different system architectures that can be used for a distributed database with replicated data. For example, clients may submit operations directly to the different databases, or instead requests may all go through a middleware layer; the local databases may communicate directly with one another, or only with the clients or middleware; the requests that arrive at a local database may be a read or write on the local replica of an item, or they may be a SQL statement (which might involve many reads and/or writes), or indeed a whole transaction may form a single request; a read may be performed on one replica and writes on all replicas (read-one-write-all), or else a complex quorum rule may determine where reads and writes are performed; and each site may perform the operations once only, or else there may be possibilities for operations to be done tentatively, then (after conflicting information is received) the system might be able to roll back some operations and then replay them in a different order. Sometimes several system designs offer clients the same functionality, so the choice would be based only on performance, or the validity of assumptions in the design (such as the ability to know in advance which transactions access which items). If the client cannot learn, by the values returned or the operations which are allowed, which system design is used, then one can say that the different designs offer the same consistency model. However, sometimes a difference in design does change the functionality of the whole system, in ways that clients can detect. If one abstracts away the details of the system design, and instead focuses on what the essential features are that distinguish between the properties, then one is describing the consistency model offered by the system. For example, some system designs allow clients to learn about the existence of several copies. Perhaps one client might read the same item several times and see different values in each read. These values may have been taken from different older transactions. This can't happen in a system where all the data is at one site, with one copy for each item. Thus this system provides a consistency model which reveals the existence of copies to the client.

A system provides a strong consistency model if it provides clients with the illusion that there is a single copy of each piece of data, hiding all evidence of the replication. There are in fact several variants among strong models, because there are several different isolation models used by different DBMS platforms, and because the formal definition of isolation doesn't always capture exactly the properties of an

implementation. For example, one-copy serializability (q.v.) was defined in [2] as a consistency model in which clients have the illusion of working with a single-site unreplicated database which uses a concurrency control algorithm that offer serializability (q.v.) as the isolation level. In contrast, one-copy SI [4] is a different strong consistency model, where clients see the same behaviors as in an unreplicated system where concurrency control is done by Snapshot Isolation (q.v.).

In contrast to strong consistency models which maintain an illusion of a single-site system, in weaker models the clients are able to see that the system has replicas. Different models are characterized by the ways in which the divergence between replicas is revealed. The best-known weak consistency model is eventual consistency (q.v.) which is suitable for replicated databases where an updating transaction can operate at any replica, and the changes are then propagated lazily to other replicas through an epidemic mechanism. Eventually, each replica learns about the updates, and this consistency model ensures that a reconciliation mechanism resolves conflicting information, so that when the system quiesces, the values in all the replicas of a logical item eventually converge to the same value. In a system providing eventual consistency, there is not much that can be said about the value seen by a read, before convergence has been reached.

A different weak consistency model is common in systems where there is a single master copy for each item, and all updates are done first at the master, before being propagated in order, to the replicas. In this model, writes happen in a well-defined order, and each read sees a value from some prefix of this order; however, a read can see a value that is stale, that is, it does not include the most recent updates to the item.

Key Applications

The commercial DBMS vendors all offer replication mechanisms with their products. The performance impact of strong consistency models is usually seen as high, and these are typically provided only within a single cluster. For replication across dispersed machines, most platforms offer some form of weak consistency. There are also a range of research prototypes which give the user a choice between several consistency models; in general the user sees a tradeoff, where improved performance comes from accepting weaker consistency models.

Future Directions

Effective database replication is not yet a solved problem; the existing proposals compromise somehow among many desired properties, such as scalability for read-heavy workloads, scalability for update-heavy workloads, availability in face of failures or partitions, generality of the clients supported, ease of system programming, capacity to use varied local databases as black boxes, and the consistency provided. Thus the design space of possible systems is still being actively explored, and sometimes a new design achieves a consistency model different from those previously seen. One topic for ongoing research is how users can express their requirements for performance and for different levels of consistency, and how a system can then choose the appropriate replica control mechanism to provide the user with what they need. Research is also likely in consistency models that deal, to some extent, with malicious (often called "Byzantine") sites.

Cross-references

- ▶ Data Replication
- ▶ Strong Consistency Models for Replicated Data
- ▶ Weak Consistency Models for Replicated Data

Recommended Reading

1. Alonso R., Barbará D., and Garcia-Molina H. Data caching issues in an information retrieval system. ACM Trans. Database Syst., 15(3):359–384, 1990.
2. Attar R., Bernstein P.A., and Goodman N. Site initialization, recovery, and backup in a distributed database system. IEEE Trans. Software Eng., 10(6):645–650, 1984.
3. Demers A.J., Greene D.H., Hauser C., Irish W., Larson J., Shenker S., Sturgis H.E., Swinehart D.C., and Terry D.B. Epidemic algorithms for replicated database maintenance. In Proc. ACM SIGACT-SIGOPS 6th Symp. on the Principles of Dist. Comp., 1987, pp. 1–12.
4. Plattner C. and Ganymed G.A. Scalable replication for transactional web applications. In Proc. ACM/IFIP/USENIX Int. Middleware Conf., 2004, pp. 155–174.
5. Sheth A.P. and Rusinkiewicz M. Management of interdependent data: specifying dependency and consistency requirements. In Proc. Workshop on the Management of Replicated Data, 1990, pp. 133–136.

Consistency Preservation

- ▶ ACID Properties

Consistent Facts

► Possible Answers

Consistent Query Answering

Leopoldo Bertossi
Carleton University, Ottawa, ON, Canada

Definition

Consistent query answering (CQA) is the problem of querying a database that is inconsistent, i.e., that fails to satisfy certain integrity constraints, in such a way that the answers returned by the database are consistent with those integrity constraints. This problem involves a characterization of the semantically correct or consistent answers to queries in an inconsistent database.

Key Points

Databases may be inconsistent in the sense that certain desirable integrity constraints (ICs) are not satisfied. However, it may be necessary to still use the database, because it contains useful information, and, most likely, most of the data is still consistent, in some sense. CQA, as introduced in [1], deals with two problems. First, with the logical characterization of the portions of data that are consistent in the inconsistent database. Secondly, with developing computational mechanisms for retrieving the consistent data. In particular, when queries are posed to the database, one would expect to obtain as answers only those answers that are semantically correct, i.e., that are consistent with the ICs that are violated by the database as a whole.

The consistent data in the database is characterized [1] as the data that is invariant under all the database instances that can be obtained after making minimal changes in the original instance with the purpose of restoring consistency. These instances are the so-called (minimal) *repairs*. In consequence, what is consistently true in the database is what is *certain*, i.e., true in the collection of possible worlds formed by the repairs. Depending on the queries and ICs, there are different algorithms for computing consistent answers. Usually, the original query is transformed into a new query, possibly written in a different language, to be posed to the database at hand, in such a way that the usual answers to the latter are the consistent answers to the former [1]. For surveys of CQA and specific references, c.f. [2,3].

Cross-references

► Database Repairs
► Inconsistent Databases

Recommended Reading

1. Arenas M., Bertossi L., and Chomicki J. Consistent query answers in inconsistent databases. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1999, pp. 68–79.
2. Bertossi L. Consistent query answering in databases. ACM SIGMOD Rec., 35(2):68–76, 2006.
3. Chomicki J. Consistent query answering: five easy pieces. In Proc. 11th Int. Conf. on Database Theory, 2007, pp. 1–17.

Constant Span

► Fixed Time Span

Constrained Frequent Itemset Mining

► Frequent Itemset Mining with Constraints

Constraint Databases

Floris Geerts
University of Edinburgh, Edinburgh, UK

Definition

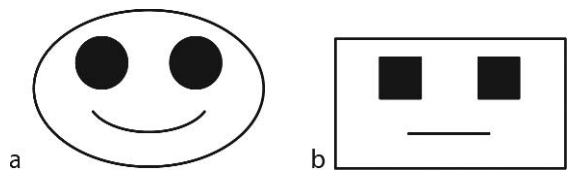
Constraint databases are a generalization of relational databases aimed to store possibly infinite-sized sets of data by means of a finite representation (*constraints*) of that data. In general, constraints are expressed by *quantifier-free first-order formulas* over some fixed vocabulary Ω and are interpreted in some Ω-structure $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$. By varying Ω and $\mathcal{M}$, constraint databases can model a variety of data models found in practice including traditional relational databases, spatial and spatio-temporal databases, and databases with text fields (strings). More formally, let Ω be a fixed vocabulary consisting of function, predicate and constant symbols, and let $\mathcal{R} = \{R_1,\ldots,R_\ell.\}$ be a relational schema, where each relation name R_i is of arity

$n_i > 0$. An Ω-*constraint database* $\mathbf{D}$ with schema $\mathcal{R}$ maps each relation $R_i \in \mathcal{R}$ to a quantifier-free formula $\varphi^{\mathbf{D}}_{R_i}(x_1,\ldots,x_{n_i})$ (with n_i free variables $x_1,\ldots,x_{n_i}$) in *first-order logic* over Ω. When interpreted over an Ω-structure $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$, an Ω-constraint database $\mathbf{D}$ with schema $\mathcal{R}$ corresponds to the collection of the $\mathcal{M}$-*definable sets* $[\![R_i]\!]^{\mathbf{D}}_{\mathcal{M}} = \{(a_1,\ldots,a_{n_i}) \in \mathbb{U}^{n_i} \mid \mathcal{M} \models \varphi^{\mathbf{D}}_{R_i}(a_1,\ldots,a_{n_i})\}$, for $R_i \in \mathcal{R}$. Constraint query languages have been devised to manipulate and query constraint databases.

Key Points

The primary motivation for constraint databases comes from the field of spatial and spatio-temporal databases where one wants to store an infinite set of points in the real Euclidean space and query it as if all (infinitely) many points are present [3,4,5]. In the spatial context, the constraints used to finitely represent data are *Boolean combinations of polynomial inequalities*. For instance, the infinite set of points in the real plane $\mathbb{R}^2$ depicted in Fig. 1(a) can be described by means of a disjunction of polynomial inequalities with integer coefficients as follows: $\varphi(x, y) = (x^2/25 + y^2/16 = 1) \vee (x^2 + 4x + y^2 - 2y \leq 4) \vee (x^2 - 4x + y^2 - 2y \leq -4) \vee (x^2 + y^2 - 2y = 8 \wedge y < -1)$. In the language of constraint databases, $\varphi(x, y)$ is a quantifier-free first-order formula over $\Omega = (+,\cdot,0,1,<)$ and Fig. 1(a) represents the $\mathcal{M}$-definable set in $\mathbb{R}^2$ corresponding to the formula φ for the Ω-structure $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$. If $\mathcal{R}$ is a relational schema consisting of a binary relation R, then the Ω-constraint database $\mathbf{D}$ with schema $\mathcal{R}$ defined by $R \mapsto \varphi(x, y)$ "stores" the set in Fig. 1(a). In this case, the $\mathcal{M}$-definable sets are also known as semi-algebraic sets [2].

When *Boolean combination of linear inequalities* suffice, such as in geographical information systems (GIS), one considers constraint databases over $\Omega = (+,0,1,<)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$. Fig. 1(b) shows an example of a set defined by means of a first-order formula over $\Omega = (+,0,1,<)$. The advantage of the constraint approach to represent spatial data is the uniform representation of the various spatial entities. Whereas in GIS one normally defines a special data-type for each spatial object such as line, poly-line, circle,..., each of those are now represented by constraints in the same constraint language.



Constraint Databases. Figure 1. Example of set definable by (a) polynomial constraints and (b) linear constraints.

Other common scenarios of constraint databases include: *dense order constraints over the rationals*, where $\Omega = (<,(c)_{c\in\mathbb{Q}})$ and $\mathcal{M} = \langle \mathbb{Q}, \Omega \rangle$. That is, rational numbers with order and constants for every $c \in \mathbb{Q}$; and *constraints over strings*, where $\Omega = ((f_a)_{a\in\Sigma},\prec,\mathrm{el})$ and $\mathcal{M} = \langle \Sigma^*, \Omega \rangle$ [1]. Here, Σ is a finite alphabet, f_a is a function that adds a at the end of its argument, $\prec$ is the prefix relation and $\mathrm{el}(x, y)$ is a binary predicate that holds if $|x| = |y|$, where $|\cdot|$ stands for the length of a finite string. In the latter case, the $\mathcal{M}$-definable sets are precisely the regular languages over Σ.

Finally, standard relational databases with schema $\mathcal{R}$ can be considered as constraint databases over *equality constraints over an arbitrary infinite domain* $\mathbb{U}$, where $\Omega = ((c)_{c\in\mathbb{U}})$ and $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$. Indeed, consider a tuple $t = (a_1,\ldots,a_n)$ consisting of some constants $a_i \in \mathbb{U}$, for $i \in [1,n]$. The tuple t can be expressed by the formula $\varphi_t(x_1,\ldots,x_n) = (x_1 = a_1) \wedge \ldots \wedge (x_n = a_n)$ over the signature $\Omega = ((c)_{c\in\mathbb{U}})$. More generally, an instance $I = \{t_1,\ldots,t_N\}$ over $R \in \mathcal{R}$ corresponds to $\varphi_I = \bigvee_{i=1}^{N} \varphi_{t_i}$. Therefore, a relational instance $(I_1,\ldots,I_\ell)$ over $\mathcal{R}$ can be represented as the constraint database $\mathbf{D}$ defined by $R_i \mapsto \varphi_{I_i}(x_1,\ldots,x_{n_i})$, for $i \in [1,\ell]$. This shows that constraint databases indeed generalize standard relational databases.

Cross-references

- ▶ Constraint query languages
- ▶ Geographic information system
- ▶ Relational model
- ▶ Spatial Data Types

Recommended Reading

1. Benedikt M., Libkin L., Schwentick T., and Segoufin L. Definable relations and first-order query languages over strings. J. ACM, 50(5):694–751, 2003.
2. Bochnak J., Coste M., and Roy M.F. Real Algebraic Geometry. Springer, Berlin, 1998.
3. Kanellakis P.C., Kuper G.M., and Revesz P.Z. Constraint query languages. J. Comput. Syst. Sci., 51(1):26–52, 1995.
4. Kuper G.M., Libkin L., and Paredaens J. (eds.) Constraint databases. Springer, Berlin, 2000.
5. Revesz P.Z. Introduction to Constraint Databases. Springer, Berlin, 2002.

Constraint Query Languages

Floris Geerts
University of Edinburgh, Edinburgh, UK

Definition

A constraint query language is a query language for constraint databases.

Historical Background

The field of constraint databases was initiated in 1990 in a paper by Kanellakis, Kuper and Revesz [9]. The goal was to obtain a database-style, optimizable version of constraint logic programming. It grew out of the research on DATALOG and constraint logic programming. The key idea was that the notion of tuple in a relational database could be replaced by a conjunction of constraints from an appropriate language, and that many of the features of the relational model could then be extended in an appropriate way. In particular, standard query languages such as those based on first-order logic and DATALOG could be extended to such a model.

It soon became clear, however, that recursive constraint query languages led to non-effective languages. The focus therefore shifted to non-recursive constraint query languages. The standard query language is the constraint relational calculus (or equivalently, the constraint relational algebra). The study of this query language turned out to lead to many interesting research problems. During the period from 1990 to 2000, the constraint setting has been studied in great generality which led to deep connections between constraint databases and embedded finite model theory. Also, the potential application of constraint databases in the spatial context led to numerous theoretical results and concrete implementations such as the DEDALE and the DISCO systems. The connection with so-called o-minimal geometry underlies many of the results in the spatial setting. The success of this research led to the publication of a comprehensive survey of the area in 2000 [11] and a textbook in 2002 [13].

In recent years, constraint query languages have been studied in new application domains such a strings, spatio-temporal and moving objects.

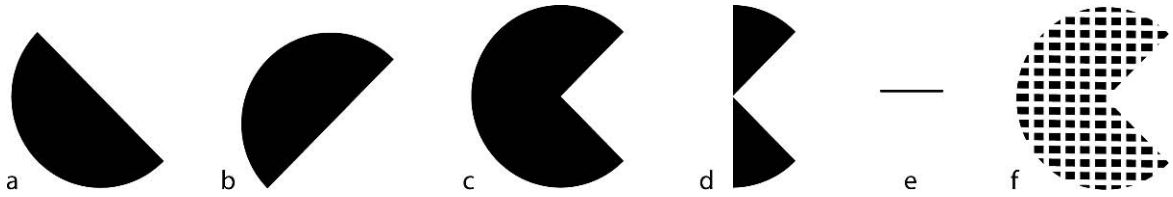
Foundations

In the constraint model, a database is viewed as a collection of constraints specified by quantifier-free first-order logic formulas over some fixed vocabulary Ω. When interpreted over an Ω-structure $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$, each constraint corresponds to an $\mathcal{M}$-definable set. Consequently, when interpreted over $\mathcal{M}$, an Ω-constraint database corresponds to a collection of $\mathcal{M}$-definable sets. For instance, consider the vocabulary $\Omega = (+, \cdot, 0, 1, <)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$. Constraints in first-order logic over Ω, denoted by FO(Ω), correspond to Boolean combinations of polynomial inequalities with integer coefficients. The corresponding $\mathcal{M}$-definable sets are better known as semi-algebraic sets. Let $\mathcal{R} = \{R, S\}$ be a relational schema consisting of two binary relations R and S and let $\mathbf{D}$ be the constraint database that maps $R \mapsto \varphi_R(x, y) = (x^2 + y^2 \leq 1) \wedge (y - x \geq 0)$ and $S \mapsto \varphi_S(x, y) = (x^2 + y^2 \leq 1) \wedge (-y - x \geq 0)$. The two $\mathcal{M}$-definable sets in $\mathbb{R}^2$ corresponding to φ_R and φ_S are shown in Figs. 1(a) and (b) respectively.

A constraint database can therefore be viewed from two different perspectives: First, one can simply look at the finite representations (constraints) stored in them; Second, one can regard them as a set of definable sets. Whereas in traditional relational databases, a query is simply a mapping that associates with each database an answer relation, in the constraint setting the two different perspectives give rise to two different notions of queries.

Indeed, for a fixed vocabulary Ω, relational schema $\mathcal{R}$ consisting of relation names R_1,...,R_ℓ, where each relation R_i is of arity $n_i > 0$, and natural number k, a *k-ary constraint query* with schema $\mathcal{R}$ over Ω, is a (partial) function Q that maps each Ω-constraint databases $\mathbf{D}$ with schema $\mathcal{R}$ to a k-ary Ω-constraint relation $Q(\mathbf{D})$. That is, a constraint query works entirely on the representational (constraint) level. On the other hand, given an additional Ω-structure $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$, a *k-ary unrestricted query* with schema $\mathcal{R}$ over $\mathcal{M}$ is a (partial) function Q that maps each collection $\mathbf{D}$ of sets in $\mathbb{U}^{n_i}$, for $i \in [1, \ell]$, to a set $Q(\mathbf{D})$ in $\mathbb{U}^k$. Such a collection of sets $\mathbb{U}^{n_i}$, for $i \in [1, \ell]$, is called an *unrestricted database* with schema $\mathcal{R}$ over $\mathcal{M}$.

For instance, consider again $\Omega = (+, \cdot, 0, 1, <)$ and $\mathcal{R} = \{R, S\}$. The mapping Q_1 that associates each Ω-constraint database $\mathbf{D}$ over $\mathcal{R}$ with the binary Ω-constraint relation defined by taking the disjunction of the constraints in R and S, is an example of a 2-ary constraint query over Ω. When applied on the database $\mathbf{D}$ given above, $Q_1(\mathbf{D})$ is mapped to $\varphi_R(x, y) \vee \varphi_S(x, y)$. Similarly, the mapping Q_2 that maps $\mathbf{D}$ to the



Constraint Query Languages. Figure 1. Sets in $\mathbb{R}^2$ defined by $\varphi_R(x, y)$ (a); by $\varphi_S(x, y)$ (b); by $\varphi_R(x, y) \vee \varphi_S(x, y)$ (c); and by $\varphi_1(x, y)$ (d). The set in $\mathbb{R}$ defined by $\varphi_2(x)$ (e). An example of a non-definable set in $\mathbb{R}^2$ (f).

constraint in R or S that contains the polynomial with the largest coefficient (if there is no such unique constraint then Q_2 is undefined) is also a constraint query. It will be undefined on the example database $\mathbf{D}$ since both R and S consist of a polynomial with coefficient one.

So far, only constraint queries have been considered. To relate constraint and unrestricted queries requires some care. Clearly, a constraint query only makes sense if it corresponds to an unrestricted query. In this case, a constraint query is called *consistent*. More formally, a constraint query Q is called consistent if there exists an unrestricted query Q_0 such that for any constraint database $\mathbf{D}$ and any unrestricted database $\mathbf{D}_0$, if $\mathbf{D}$ represents $\mathbf{D}_0$, then $Q(\mathbf{D})$ is defined if and only if $Q_0(\mathbf{D}_0)$ is defined and furthermore, $Q(\mathbf{D})$ represent $Q_0(\mathbf{D}_0)$. One also says that Q *represents* Q_0.

For instance, consider again $\Omega = (+,\cdot,0,1,<)$, $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ and $\mathcal{R} = \{R, S\}$. The mapping $\tilde{Q}_1$ that assigns to any two sets $A \subseteq \mathbb{R}^2$ and $B \subseteq \mathbb{R}^2$, corresponding to R and S, respectively, their union $A \cup B \subseteq \mathbb{R}^2$ is an unrestricted query. It is clear that Q_1 and $\tilde{Q}_1$ satisfy the condition of consistency and therefore Q_1 is consistent. Fig. 1(c) shows $\tilde{Q}_1(\mathbf{D}_0)$ for the unrestricted database $\mathbf{D}_0$ shown in Fig. 1(a) and (b). This set is indeed represented by the constraint relation $Q_1(\mathbf{D}) \mapsto \varphi_R(x, y) \vee \varphi_S(x, y)$. On the other hand, it is easily verified that Q_2 is not consistent. Indeed, it suffices to consider the behavior of Q_2 on $\mathbf{D}$ defined above and $\mathbf{D}'$ defined by $R \mapsto \varphi'_R(x, y) = (x^2 + y^2 \leq 1) \wedge (6(y - x) \geq 0)$ and $S \mapsto \varphi'_S(x, y) = (x^2 + y^2 \leq 1) \wedge (-y - x \geq 0)$. While both $\mathbf{D}$ and $\mathbf{D}'$ represent the same unrestricted database, note that $Q_2(\mathbf{D})$ is undefined while $Q_2(\mathbf{D}') \mapsto \varphi_R$. Hence, no unrestricted query that is consistent with Q_2 can exist.

Finally, unrestricted queries are defined without any reference to the class of $\mathcal{M}$-definable sets. A desirable property, however, is that when an unrestricted query Q is defined on an unrestricted database $\mathbf{D}$ that consists of $\mathcal{M}$-definable sets, then also $Q(\mathbf{D})$ is an $\mathcal{M}$-definable set. Such unrestricted queries are called *closed*. Note that an unrestricted query that is represented by a consistent constraint query is uniquely defined and moreover is trivially closed. An example of an unrestricted query for $\Omega = (+,\cdot,0,1,<)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ that is not closed is the query Q that maps any $\mathcal{M}$-definable set A in $\mathbb{R}^2$ to its intersection $A \cap \mathbb{Q}^2$. Fig. 1(f) shows (approximately) the result of this query on $\tilde{Q}_1(\mathbf{D}_0)$ (i.e., Fig. 1(c)). Since this is not a semi-algebraic set in $\mathbb{R}^2$, it cannot be defined by means of a quantifier-free FO(Ω)-formula. As a consequence, Q is not closed.

Now that the notion of query is defined in the setting of constraint databases, the basic *constraint query language* is introduced. This language, in the same spirit as the relational calculus for traditional relational databases, is the *relational calculus* or first-order logic of the given class of constraints. More specifically, given a vocabulary Ω and relational schema $\mathcal{R}$, a *relational calculus formula* over Ω is a first-order logic formula over the expanded vocabulary $(\Omega, \mathcal{R})$ obtained by expanding Ω with the relation names (viewed as predicate symbols) of the schema $\mathcal{R}$. This class of queries is denoted by FO($\Omega, \mathcal{R}$), or simply FO(Ω) when $\mathcal{R}$ is understood from the context.

For instance, for $\Omega = (+, \cdot, 0, 1, <)$ and $\mathcal{R} = \{R, S\}$, the expressions $\varphi_1(x, y) = (R(x, y) \vee S(x, y)) \wedge x > 0$ and $\varphi_2(x) = \exists y \varphi_1(x, y)$ are formulas in FO($+, \cdot, 0, 1, <, R, S$).

Given an Ω-structure $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$, formulas in FO($\Omega, \mathcal{R}$) express (everywhere defined) unrestricted queries with schema $\mathcal{R}$ over $\mathcal{M}$. Indeed, a formula $\varphi(x_1,\ldots,x_k) \in$ FO($\Omega, \mathcal{R}$) defines the k-ary unrestricted query Q over $\mathcal{M}$ as follows: consider the expansion of $\mathcal{M}$ to a structure $\langle \mathcal{M}, \mathbf{D} \rangle = \langle \mathbb{U}, \Omega, \mathbf{D} \rangle$ over the expanded vocabulary $(\Omega, \mathcal{R})$ by adding the sets in the unrestricted database $\mathbf{D}$ to $\mathcal{M}$ for each $R_i \in \mathcal{R}$. Then, $Q(\mathbf{D}) = \{(a_1,\ldots,a_k) \in \mathbb{U}^k \mid 0\langle \mathcal{M}, \mathbf{D} \rangle \models \varphi(a_1,\ldots,a_k)\}$.

For instance, for $\Omega = (+,\cdot,0,1,<)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$, the formula $\varphi_1(x, y)$ defined above

corresponds to the unrestricted query Q_1 that takes the union of the two sets in $\mathbb{R}^2$ corresponding to R and S, respectively, restricted to those points in $\mathbb{R}^2$ with strictly positive x-coordinate. Similarly for φ_2, but with an additional projection on the x-axis. The results of these two unrestricted queries have been shown in Figs. 1(d), (e), respectively.

The previous example raises the following two questions: (i) are the unrestricted queries expressed by formulas in first-order logic closed, and (ii) if so, can one find a corresponding constraint query that is effectively computable? The fundamental mechanism underlying the use of first-order logic as a constraint query language is the following observation that provides an answer to both questions:

- Every relational calculus formula φ expresses a consistent, effectively computable, total constraint query that represents the unrestricted query expressed by φ, if and only if $\mathcal{M}$ admits *effective quantifier elimination*.

Here, an Ω-structure $\mathcal{M}$ admits effective quantifier elimination if there exists an effective algorithm that transforms any first-order formula in FO(Ω) to an equivalent (in the structure $\mathcal{M}$) *quantifier-free* first-order formula in FO(Ω).

Consider the two FO($+,\cdot,0,1,<$, R, S)-formulas φ_1 and φ_2 given above. It is known that the structure $\langle \mathbb{R}, +, \cdot, 0, 1, < \rangle$ admits effective quantifier-elimination. In case of φ_1 it is easy to see that the result of corresponding constraint query is obtained by "plugging" in the constraints for R (resp. S) as given by the constraint database into the expression for φ_1. That is, on the example database $\mathbf{D}$, φ_1 corresponds to the constraint query that maps $\mathbf{D}$ to $(\varphi_R(x, y) \vee \varphi_S(x, y)) \wedge (x > 0)$, which is a 2-ary Ω-constraint relation. In case of φ_2, however, first plug in the descriptions of the constraints as before, resulting in $\exists y\,(\varphi_R(x, y) \vee \varphi_S(x, y)) \wedge (x > 0)$. In order to obtain an Ω-constraint relation, one needs perform quantifier-elimination. It is easily verified that in this example, a corresponding constraint query is one that maps $\mathbf{D}$ to $(0 < x) \wedge (x \leq 1)$ which is consistent with Fig. 1(e).

For Ω-structures $\mathcal{M}$ that admit effective quantifier-elimination, this suggests the following effective evaluation mechanism for constraint relational calculus queries φ on a constraint database $\mathbf{D}$: (i) plug in the contents of $\mathbf{D}$ in the appropriate slots (relations). Denote the resulting formula by plug(φ, $\mathbf{D}$); and (ii) eliminate the quantifiers in plug(φ, $\mathbf{D}$). Since $\mathbf{D}$ consists of quantifier-free formulas, the number of quantifiers that need to be eliminated is the same as in φ and is therefore independent of $\mathbf{D}$. For many structures $\mathcal{M}$ this implies that the evaluation of constraint queries can be done in polynomial data complexity, which is a desirable property for any query language.

It is important to point out that the classical equivalence between the relational calculus and the relational algebra can be easily extended to the constraint setting. That is, for a fixed Ω and schema $\mathcal{R}$, one can define a *constraint relational algebra* and show that every constraint relational calculus formula can be effectively converted to an equivalent constraint relational algebra expression, and vice versa. This equivalence is useful for concrete implementations of constraint database systems.

The study of expressivity of FO($\Omega, \mathcal{R}$) for various Ω-structures $\mathcal{M}$ has led to many interesting results. In particular, the impact of the presence of the "extra" structure on the domain elements in $\mathbb{U}$ has been addressed when $\mathbf{D}$ consists of an ordinary finite relational database that takes values from $\mathbb{U}$ [3]. In particular, the correspondence between natural and active-domain semantics has been revisited. That is, conditions are identified for $\mathcal{M} = \langle \mathbb{U}, \Omega \rangle$ such that the language FO($\Omega, \mathcal{R}$) is equal to $\mathrm{FO}_{\mathrm{act}}(\Omega, \mathcal{R})$, the query language obtained by interpreting $\forall x$ and $\exists x$ over the active domain of $\mathbf{D}$ instead of over $\mathbb{U}$. Such structures are said to admit the *natural-active collapse*. Similarly, ordered structures $\mathcal{M}$ are identified that admit the *active-generic collapse*. That is, $\mathrm{FO}_{\mathrm{act}}(\Omega, \mathcal{R})$ is equal to $\mathrm{FO}_{\mathrm{act}}(<, \mathcal{R})$ with respect to the class of generic queries. In other words, every generic query definable under active domain semantics with Ω-constraints is already definable with just order constraints. Finally, structures $\mathcal{M}$ are considered that allow the *natural-generic collapse*. This is the same as the active-generic collapse but with natural domain semantics instead of active domain semantics. The study of these collapse properties for various structures not only sheds light on the interaction of the structure on $\mathbb{U}$ and the query language, it is also helpful to understand the expressiveness of constraint query languages [3,11].

Indeed, let $\Omega = (+, \times, 0, 1, <)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$. It can be shown that $\mathcal{M}$ admits all three collapses because it is a so-called o-minimal structure. As a consequence, the query EVEN that returns **yes** if the

cardinality of **D** is even and **no** otherwise, is not expressible in FO($\Omega, \mathcal{R}$). Indeed, if it would be expressible by a query φ in FO($\Omega, \mathcal{R}$) it would already have been expressible by a query in $\text{FO}_{\text{act}}(<, \mathcal{R})$, which is known not to be true in the traditional database setting.

The expressivity of FO($\Omega, \mathcal{R}$) has been studied extensively as well when **D** corresponds to sets of infinite size. In particular, expressiveness questions have been addressed in the spatial setting where $\Omega = (+, \cdot, 0, 1, <)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega \rangle$ (polynomial constraints); and $\Omega' = (+, 0, 1, <)$ and $\mathcal{M} = \langle \mathbb{R}, \Omega' \rangle$ (linear constraints). In this setting, many reductions are presented in [7] to expressiveness questions in the finite case. Combined with the collapse results mentioned above, these reductions were used to show that, for example, topological connectivity of Ω- (resp. Ω'-) constraint databases is not expressible in first-order logic. Indeed, a proof of this results relies on the fact that the EVEN-query is not expressible in FO($\Omega, \mathcal{R}$) (resp. FO($\Omega', \mathcal{R}$)) [7].

An interesting line of work in the spatial context concerns the expressive power of FO($\Omega, \mathcal{R}$) with respect to queries that preserve certain geometrical properties. More formally, let $\mathcal{G}$ be a group of transformations of $\mathbb{R}^k$. A query Q is called $\mathcal{G}$-generic if, for every transformation $g \in \mathcal{G}$, and for any two databases **D** and **D**′, g(**D**) = **D**′ implies g(Q(**D**)) = Q(**D**′). Transformation groups and properties of the corresponding generic queries have been studied for the group of homeomorphisms, affinities, similarities, isometries, among others [8]. Especially the study of the topologically queries (those that are generic under homeomorphisms) has received considerable attention [10,2].

To conclude, both for the historical reasons mentioned above and in view of the limited expressive power of FO($\Omega, \mathcal{R}$), various recursive extensions of FO($\Omega, \mathcal{R}$) have been proposed such as: constraint transitive-closure logic [5], constraint DATALOG [9], and FO($\Omega, \mathcal{R}$) extended with a WHILE-loop [8]. The interaction of recursion with the structure on $\mathbb{U}$ imposed by Ω leads in most cases to computationally complete query languages. Worse still, queries defined in these languages may not be closed or even terminate. To remedy this, special-purpose extensions of FO($\Omega, \mathcal{R}$) have been proposed that guarantee both termination and closure. Characteristic examples include FO($\Omega, \mathcal{R}$) + AVG and FO($\Omega, \mathcal{R}$) + SUM in the context of aggregation [4]. In the spatial setting, extensions of FO($\Omega, \mathcal{R}$) with various connectivity operators have been proposed [1].

Results concerning constraint query languages have been both extended to great generality and applied to concrete settings. Refer to [14] for a gentle introduction and to [11] for a more detailed survey of this research area up to 2000. Some more recent results are included in Chapter 5 of [12] for the general constraint setting and Chapter 12 in [6] for the spatial setting.

Key Applications

Manipulation and querying of constraint databases, querying of spatial data.

Cross-references

- ▶ Computationally Complete Relational Query Languages
- ▶ Constraint Databases
- ▶ FOL
- ▶ FOL Modeling of Integrity Constraints (Dependencies)
- ▶ Query Language
- ▶ Relational Calculus
- ▶ Relational Model

Recommended Reading

1. Benedikt M., Grohe M., Libkin L., and Segoufin L. Reachability and connectivity queries in constraint databases. J. Comput. Syst. Sci., 66(1):169–206, 2003.
2. Benedikt M., Kuijpers B., Christ of Löding, Van den Bussche J., and Wilke T. A characterization of first-order topological properties of planar spatial data. J. ACM, 53(2):273–305, 2006.
3. Benedikt M. and Libkin L. Relational queries over interpreted structures. J. ACM, 47(4):644–680, 2000.
4. Benedikt M. and Libkin L. Aggregate operators in constraint query languages. J. Comput. Syst. Sci., 64(3):628–654, 2002.
5. Geerts F., Kuijpers B., and Van den Bussche J. Linearization and completeness results for terminating transitive closure queries on spatial databases. SIAM J. Comput., 35(6):1386–1439, 2006.
6. Geerts F. and Kuijpers B. Real algebraic geometry and constraint databases. In M. Aiello, I. Pratt-Hartmann, and J. Van Benthem, editors, Handbook of Spatial Logics. Springer 2007.
7. Grumbach S. and Su J. Queries with arithmetical constraints. Theor. Comput. Sci., 173(1):151–181, 1997.
8. Gyssens M., Van den Bussche J., and Van Gucht D. Complete geometric query languages. J. Comput. Syst. Sci., 58(3):483–511, 1999.
9. Kanellakis P.C., Kuper G.M., and Revesz P.Z. Constraint Query Languages. J. Comput. Syst. Sci., 51(1):26–52, 1995.
10. Kuijpers B., Paredaens J., and Van den Bussche J. Topological elementary equivalence of closed semi-algebraic sets in the real plane. J. Symb. Log., 65(4):1530–1555, 2000.
11. Kuper G.M., Libkin L., and Paredaens J. Constraint Databases. editors. Springer, 2000.

12. Libkin L. Embedded finite models and constraint databases. In Grädel E., Kolaitis P.G., Libkin L., Marx M., Spencer J., Vardi M.Y., Venema Y. and Weinstein S., editors, Finite Model Theory and Its Applications. Springer, 2007.
13. Revesz P.Z. Introduction to Constraint Databases. Springer, 2002.
14. Van den Bussche J. Constraint databases. A tutorial introduction. ACM SIGMOD Record, 29(3):44–51, 2000.

Constraint-Driven Database Repair

Wenfei Fan[1,2]
[1]University of Edinburgh, Edinburgh, UK
[2]Bell Laboratories, Murray Hill, NJ, USA

Synonyms

Data reconciliation; Minimal-change integrity maintenance; Data standardization

Definition

Given a set Σ of integrity constraints and a database instance D of a schema R, the problem of *constraint-driven database repair* is to find an instance D' of the same schema R such that (i) D' is *consistent*, i.e., D' satisfies Σ, and moreover, (ii) D' *minimally differs* from the original database D, i.e., it takes a minimal number of repair operations or incurs minimal cost to obtain D' by updating D.

Historical Background

Real life data is often dirty, i.e., inconsistent, inaccurate, stale or deliberately falsified. While the prevalent use of the Web has made it possible, on an unprecedented scale, to extract and integrate data from diverse sources, it has also increased the risks of creating and propagating dirty data. Dirty data routinely leads to misleading or biased analytical results and decisions, and incurs loss of revenue, credibility and customers. With this comes the need for finding repairs of dirty data, and editing the data to make it consistent. This is the data cleaning approach that US national statistical agencies, among others, has been practicing for decades [10].

The notion of constraint-based database repairs is introduced in [1], highlighting the use of integrity constraints for characterizing the consistency of the data. In other words, constraints are used as data quality rules, which detect inconsistencies as violations of the constraints. Prior work on constraint-based database repairs has mostly focused on the following issues. (i) Integrity constraints used for repair. Earlier work considers traditional functional dependencies, inclusion dependencies and denial constraints [1,2,4,6,12]. Extensions of functional and inclusion dependencies, referred to as conditional functional and inclusion dependencies, are recently proposed in [3,9] for data cleaning. (ii) Repair semantics. Tuple deletion is the only repair operation used in [6], for databases in which the information is complete but not necessarily consistent. Tuple deletion and insertion are considered in [1,4] for databases in which the information may be neither consistent nor complete. Updates, i.e., attribute-value modification are proposed as repair operations in [12]. Cost models for repairs are studied in [2,8]. (iii) Algorithms. The first algorithms for finding repairs are developed in [2], based on traditional functional and inclusion dependencies. Algorithms for repairing and incrementally repairing databases are studied in [8], using conditional functional dependencies. The repair model adopted by these algorithms supports updates as repair operations. (iv) Fundamental issues associated with constraint-based repairs. One issue concerns the complexity bounds on the database repair problem [2,6]. Another issue concerns the static analysis of constraint consistency [3,9] for determining whether a given set of integrity constraints is dirty or not itself.

Constraint-based database repairs are one of the two topics studied for constraint-based data cleaning. The other topic, also introduced in [1], is *consistent query answers*. Given a query Q posed on an inconsistent database D, it is to find tuples that are in the answer of Q over every repair of D. There has been a host of work on consistent query answers [1,4,6,11,12] (see [5,7] for comprehensive surveys).

Foundations

The complexity of the constraint-based database repair problem is highly dependent upon what integrity constraints and repair model are considered.

Integrity Constraints for Characterizing Data Consistency

A central technical issue for data cleaning concerns how to tell whether the data is dirty or clean. Constraint-based database repair characterizes inconsistencies in terms of violations of integrity constraints. Constraints employed for data cleaning include functional dependencies,

inclusion dependencies, denial constraints, conditional functional dependencies and conditional inclusion dependencies. To illustrate these constraints, consider the following relational schema R, which consists of three relation schemas:

customer(name, country-code, area-code, phone, city, street, zip)

order(name, country-code, area-code, phone, item-id, title, price, item-type)

book(isbn, title, price, format)

Traditional functional dependencies and inclusion dependencies defined on the schema R include:

FD: customer(country-code, area-code, phone $\rightarrow$ city, street, zip)

IND: order(name, country-code, area-code, phone) $\subseteq$ customer(name, country-code, area-code, phone)

The functional dependency FD asserts that the phone number (country-code, area-code and phone) of a customer uniquely determines her address (state, city, street, zip). That is, for any two customer tuples, if they have the same country-code, area-code and phone number, then they must have the same state, city, street, zip code. The inclusion dependency IND asserts that for any order tuple t, there must exist a customer tuple t' such that t and t' match on their name, country-code, area-code and phone attributes. In other words, an item cannot be ordered by a customer who does not exist.

Consider a set Σ consisting of FD and IND . One may want to use Σ to specify the consistency of database instances of R. An example instance D of R is shown in Fig. 1. This database is inconsistent, because tuples t_3 and t_4 in D violate the functional dependency FD . Indeed, while t_3 and t_4 have the same country-code, area-code and phone number, they differ in their street attributes. In other words, t_3, t_4 or both of them may be dirty.

One may also want to add a denial constraint to the set Σ:

DC : $\forall$nm, cc, ac, ph, id, tl, tp, pr$\neg$ (order(nm, cc, ac, ph, id, tl, pr, tp) $\wedge$pr $>$ 100)

Here nm, cc, ac, ph, id, tl, tp, pr stand for name, country-code, area-code, phone number, item-id, title, item type and price, respectively. This constraint says that no items in the order table may have a price higher than 100. In the database D of Fig. 1, tuple t_6 violates the constraint DD : the price of the CD is too high to be true. In general denial constraints can be expressed as universally quantified first-order logic sentences of the form:

$$\forall \bar{x}_1,\ldots,\bar{x}_m \neg(R_1(\bar{x}_1) \wedge \ldots \wedge R_m(\bar{x}_m) \wedge \varphi(\bar{x}_1,\ldots,\bar{x}_m)),$$

where R_i is a relation symbol for $i \in [1,m]$, and φ is a conjunction of built-in predicates.

Now consider an instance D' of D by removing t_3 from D and changing t_6[pr] to, e.g., 7.99. Then the database D' satisfies Σ. However, D' is not quite clean: it violates each of the following constraints. In other

	Name	Country-code	Area-code	Phone	City	Street	Zip
t_1:	M. Smith	44	131	1233444	NYC	Mayfield	EH8 8LE
t_2:	L. Webber	44	131	2344455	NYC	Crichton	EH8 8LE
t_3:	M. Hull	01	908	3456788	NYC	Mountain Ave	07974
t_4:	R. Xiong	01	908	3456788	NYC	Main St	07974

a Example customer data

	Name	Country-code	Area-code	Phone	Item-id	Title	Price	Item-type
t_5:	M.Smith	44	131	1233444	b23	H. Porter	17.99	book
t_6:	M.Hull	01	908	3456788	c68	John Denver	2000	CD

b Example order data

	isbn	Title	Price	Format
t_7:	b23	Harry porter	17.99	hard-cover
t_8:	b88	Snow white	7.94	audio

c Example book data

Constraint-Driven Database Repair. Figure 1. Example database instance.

words, if one further extends Σ by including the following constraints, then D' no longer satisfies Σ.

CFD1 : customer(country-code = 44, zip → street)
CFD2 : customer(country-code = 44, area-code = 131, phone → city = EDI, street, zip)
CFD3 : customer(country-code = 01, area-code = 908, phone → city = MH, street, zip)
CIND1 : order(id, title, price; item-type = book) ⊆ book(isbn, title, price)

Here CFD1, CFD2 and CFD3 are conditional functional dependencies defined on the customer relation. The constraint CFD1 asserts that for each customer in the UK, i.e., when the country code is 44, her zip code uniquely determines her street. In contrast to traditional functional dependencies, CFD1 does not hold on the entire customer relation. Indeed, it does not hold on customer tuples with, e.g., country-code = 01. Instead, it is applicable only to the set of customer tuples with country-code = 44. Constraints CFD2 and CFD3 refine the traditional functional dependency FD given earlier: CFD2 requires that when the country-code is 44 and area-code is 131, the city must be Edinburgh (EDI); similarly for CFD3. None of these can be expressed as traditional functional dependencies.

The constraint CIND1 is a conditional inclusion dependency, asserting that when the type of an item t in the order table is book, there must exist a corresponding tuple t' in the book table such that t and t' match on their id, title and price. Again this is a constraint that only holds conditionally. Indeed, without the condition item-type = book, a traditional inclusion dependency order(id, title, price) ⊆ book (isbn, title, price) does not make sense since, among other things, it is unreasonable to require each CD item in the order table to match a tuple in the book table.

These conditional dependencies tell us that the database D' is not clean after all. Indeed, tuples t_1,t_2 in the customer table violate CFD1: while they both represent customers in the UK and have the same zip code, they differ in their street attributes. Furthermore, each of t_1 and t_2 violates CFD2: while its area-code is 131, its city is NYC instead of EDI. Similarly, t_4 violates CFD3. From these one can see that while it takes two tuples to violate a traditional functional dependency, a single tuple may violate a conditional functional dependency. The inconsistencies in D' are not limited to the customer table: while tuple t_5 in the order table has item-type = book, there exists no tuple t' in the book table such that t_5 and t' match on their id, title and price attributes. Thus either t_5 in the order table is not error-free, or the book table is incomplete or inconsistent.

Conditional functional and inclusion dependencies are extensions of traditional functional and inclusion dependencies, respectively. In their general form each conditional functional (resp. inclusion) dependency is a pair comprising of (i) a traditional functional (resp. inclusion) dependency and (ii) a pattern tableau consisting of tuples that enforce binding of semantically related data values. Traditional functional (resp. inclusion) dependencies are a special case of conditional functional (resp. inclusion) dependencies, in which the tableaux do not include tuples with patterns of data values. As opposed to traditional functional and inclusion dependencies that were developed mainly for schema design, conditional functional and inclusion dependencies aim to capture the consistency of the data, for data cleaning. As shown by the example above, conditional dependencies are capable of detecting more errors and inconsistencies than what their traditional counterparts can find.

In summary, integrity constraints specify a fundamental part of the semantics of the data. Indeed, errors and inconsistencies in real-world data often emerge as violations of integrity constraints. The more expressive the constraints are, the more errors and inconsistencies can be caught. On the other hand, as will be seen shortly, the expressive power of the constraints often comes with the price of extra complexity for finding database repairs.

Repair Models

Consider functional dependencies, inclusion dependencies, denial constraints, conditional functional dependencies and inclusion dependencies. Given a database instance D of a schema R, if D violates a set Σ consisting of these constraints, one can always edit D to obtain a consistent instance D' of R, such that D' satisfies Σ. An extreme case is to delete all tuples from D and thus get an empty D'. Such a fix is obviously impractical: it removes inconsistencies as well as correct information. Apparently database repairs should not be conducted with the price of losing information of the original data. This motivates the criterion for database repairs to minimally differ from the original data.

Several repair models have been proposed [1,6]. One model allows tuple deletions only, assuming that the information in the database D is inconsistent but is complete. In this model, a repair D' is a maximal subset of D that satisfies Σ. For example, consider Σ consisting of FD given above, and the database D shown in Fig. 1. Then a repair of D can be obtained by removing either t_3 or t_4 from the customer table.

Another model allows both tuple deletions and insertions. In this model, a repair D' is an instance of R such that (i) D' satisfies Σ, and (ii) the difference between D and D', i.e., $(D \setminus D') \cup (D' \setminus D)$, is minimal when D' ranges over all instances of R that satisfy Σ. As an example, let Σ consist of FD and CIND given above, and D be the database of Fig. 1. Then one can obtain a repair of D either by removing both t_3 and t_5, or by removing t_3 but inserting a tuple t' to the book table such that t_5 and t' agree on their id, title and price attributes.

A more practical model is based on updates, i.e., attribute-value modifications. To illustrate this, let us consider Σ consisting of all the constraints that have been encountered, i.e., FD, IND, DD, CFD1, CFD2, CFD3 and CIND given above, and the database D of Fig. 1. Observe that every tuple in the customer relation violates at least one of the (conditional) functional dependencies in Σ. In the two models mentioned above, the only way to find a repair is by removing all tuples from the customer table. However, it is possible that only some fields in a customer tuple are not correct, and thus it is an overkill to remove the entire tuple. A more reasonable fix is to update the tuples by, e.g., changing t_1[city] and t_2[city] to EDI (for CFD2), t_1[street] to Crichton (forCFD1), t_3[city] and t_4[city] to MH (for CFD3), t_3[street] to Mountain Ave (for FD), t_6[price] to 7.99 (for DD), and t_5[title] to Harry Porter (for CIND). This yields a repair in the update model.

An immediate question about the update model concerns what values should be changed and what values should be chosen to replace the old values. One should make the decisions based on both the accuracy of the attribute values to be modified, and the "closeness" of the new value to the original value. Following the practice of US national statistical agencies [10], one can define a cost metric as follows [2,8]. Assuming that a *weight* in the range [0,1] is associated with each attribute A of each tuple t in D, denoted by $w(t,A)$ (if $w(t,A)$ is not available, a default weight can be used instead). The weight reflects the confidence of the *accuracy* placed by the user in the attribute $t[A]$, and can be propagated via data provenance analysis in data transformations. For two values v,v' in the same domain, assume that a *distance function* $\text{dis}(v,v')$ is in place, with lower values indicating greater similarity. The cost of changing the value of an attribute $t[A]$ from v to v' can be defined to be:

$$\text{cost}\ (v, v') = w(t, A) \cdot \text{dis}\ (v, v')/\max\ (|v|, |v'|),$$

Intuitively, the more accurate the original $t[A]$ value v is and more distant the new value v' is from v, the higher the cost of this change. The similarity of v and v' is measured by $\text{dis}(v,v')/\max(|v|,|v'|)$, where $|v|$ is the length of v, such that longer strings with 1-character difference are closer than shorter strings with 1-character difference. The cost of changing the value of a tuple t to t' is the sum of $\text{cost}(t[A],t'[A])$ when A ranges over all attributes in t for which the value of $t[A]$ is modified. The cost of changing D to D', denoted by $\text{cost}(D',D)$, is the sum of the costs of modifying tuples in D. A repair of D in the update model is an instance D' of R such that (i) D' satisfies Σ, and (ii) $\text{cost}(D',D)$ is minimal when D' ranges over all instances of R that satisfy Σ.

The accuracy of a repair can be measured by precision and recall metrics, which are the ratio of the number of errors correctly fixed to the total number of changes made, and the ratio of the number of errors correctly fixed to the total number of errors in the database, respectively.

Methods for Finding Database Repairs

It is prohibitively expensive to find a repair of a dirty database D by manual effort. The objective of constraint-based database repair is to automatically find candidate repairs of D. These candidate repairs are subject to inspection and change by human experts. There have only been preliminary results on methods for finding quality candidate repairs, as outlined below.

Given a set Σ of integrity constraints, either defined on a schema R or discovered from sample instances of R, one first wants to determine whether or not Σ is dirty itself. That is, before Σ is used to find repairs of D, one has to check, at compile time, whether or not Σ is consistent or makes sense, i.e., whether or not there exists a nonempty database instance of R that satisfies Σ. For traditional functional and inclusion dependencies, this is not an issue: one can specify

arbitrary functional and inclusion dependencies without worrying about their consistency. While conditional inclusion dependencies alone retain this nice property, it is no longer the case when it comes to conditional functional dependencies. For example, consider the following conditional functional dependencies: $\psi_1 = R(A = \text{true} \rightarrow B = b_1)$, $\psi_2 = R(A = \text{false} \rightarrow B = b_2)$, $\psi_3 = R(B = b_1 \rightarrow A = \text{false})$, and $\psi_4 = R(B = b_2 \rightarrow A = \text{true})$, where the domain of attribute A is Boolean. While each of these constraints can be separately satisfied by a nonempty database instance, there exists no nonempty instance that satisfies all of these constraints. Indeed, for any tuple t in an instance, no matter what Boolean value $t[A]$ has, these constraints force $t[A]$ to take the other value from the Boolean domain.

The consistency problem is already NP-complete for conditional functional dependencies alone [9], and it becomes undecidable for conditional functional and inclusion dependencies taken together [3]. In light of the complexity, the consistency analysis is necessarily conducted by effective heuristic methods, ideally with performance guarantee. There has been approximate algorithms developed for checking the consistency of conditional functional dependencies, and heuristic algorithms for conditional functional and inclusion dependencies taken together.

After Σ is confirmed consistent, one needs to detect the inconsistencies in the database D, i.e., to find all tuples in D that violate one or more constraints in Σ. It has been shown that it is possible to automatically generate a fixed number of SQL queries from Σ, such that these queries can find all violations in D. This strategy works when Σ consists of functional dependencies, inclusion dependencies, conditional functional and inclusion dependencies [9]. Better yet, the size of the queries is dependent upon neither the number of constraints in Σ nor the pattern tableaux in the conditional dependencies in Σ.

After all violations are identified, the next step is to find an accurate repair of D by fixing these violations. This is challenging: in the repair model based on attribute-value updates, the problem of finding a database repair is already NP-complete even when the database schema is fixed and only a fixed number of traditional functional (or inclusion) dependencies are considered [2].

To cope with the tractability of the problem, several heuristic algorithms have been developed, based on the cost model given above. A central idea of these algorithms is to separate the decision of which attribute values should be equal from the decision of what value should be assigned to these attributes. Delaying value assignment allows a poor local decision to be improved in a later stage of the repairing process, and also allows a user to inspect and modify a repair. To this end an equivalence class $\text{eq}(t,A)$ can be associated with each tuple t in the dirty database D and each attribute A in t. The repairing is conducted by merging and modifying the equivalence classes of attributes in D. For example, if tuples t_1, t_2 in D violate a functional dependency $R(X \rightarrow A)$, one may want to fix the inconsistency by merging $\text{eq}(t_1,A)$ and $\text{eq}(t_2,A)$ into one, i.e., by forcing t_1 and t_2 to match on their A attributes. If a tuple t_1 violates an inclusion dependency $R_1[X] \subseteq R_2[Y]$, one may want to resolve the conflict by finding an existing tuple t_2 in the R_2 relation or inserting a new tuple t_2 into the R_2 table, such that for each corresponding attribute pair (A,B) in $[X]$ and $[Y]$, $t_1[A] = t_2[B]$ by merging $\text{eq}(t_1,A)$ and $\text{eq}(t_2,B)$ into one. A target value is assigned to each equivalence class when no more merging is possible.

Based on this idea, effective heuristic algorithms have been developed for repairing databases using traditional functional and inclusion dependencies (e.g., [8]). The algorithms modify tuple attributes in the right-hand side of a functional or inclusion dependency in the presence of a violation. This strategy, however, no longer works for conditional functional dependencies: the process may not even terminate if only tuple attributes in the right-hand side of a conditional functional dependency can be modified. Heuristic algorithms for repairing conditional functional dependencies have been developed [8], which are also based on the idea of equivalence classes but may modify tuple attributes in either the left-hand side or right-hand side of a conditional functional dependency in the presence of a violation.

Key Applications

Constraint-based database repairs have a wide range of applications in, e.g., data standardization, data quality tools, data integration systems, master data management, and credit-card fraud detection.

Future Directions

The study of constraint-based database repair is still in its infancy. There is naturally much more to be done. One topic for future research is to identify new integrity constraints that are capable of detecting inconsistencies and errors commonly found in practice, without incurring extra complexity. The second topic

is to develop more accurate and practical repair models. The third topic is to find heuristic methods, with performance guarantees, for reasoning about integrity constraints used for data cleaning, such as their consistency and implication analyses. The fourth yet the most challenging topic is to develop scalable algorithms for finding database repairs with performance guarantee, such as to guarantee that the precision and recall of the repairs found are above a predefined bound with a high confidence.

Cross-references

- ▶ Data Cleaning
- ▶ Data Quality Models
- ▶ Database Dependencies
- ▶ Database Repair
- ▶ Functional Dependency
- ▶ Inconsistent Databases
- ▶ Record Linkage
- ▶ Relational Integrity Constraints
- ▶ Uncertain Databases

Recommended Reading

1. Arenas M., Bertossi L.E., and Chomicki J. Consistent query answers in inconsistent databases. In Proc. 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1999, pp. 68–79.
2. Bohannon P., Fan W., Flaster M., and Rastogi R. A cost-based model and effective heuristic for repairing constraints by value modification. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005.
3. Bravo L., Fan W., and Ma S. Extending dependencies with conditions. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 243–254.
4. Calì A., Lembo D., and Rosati R. On the decidability and complexity of query answering over inconsistent and incomplete databases. In Proc. 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2003, pp. 260–271.
5. Chomicki J. Consistent query answering: Five easy pieces. In Proc. 11th Int. Conf. on Database Theory, 2007, pp. 1–17.
6. Chomicki J. and Marcinkowski J. Minimal-change integrity maintenance using tuple deletions. Inf. Comput., 197(1–2): 90–121, 2005.
7. Chomicki J. and Marcinkowski J. On the computational complexity of minimal-change integrity maintenance in relational databases. Inconsistency Tolerance :119–150, 2005.
8. Cong G., Fan W., Geerts F., Jia X., and Ma S. Improving data quality: Consistency and accuracy. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 315–326.
9. Fan W., Geerts F., Jia X., and Kementsietsidis A. Conditional functional dependencies for capturing data inconsistencies. ACM Trans. Database Syst., 33(2), 2008.
10. Fellegi I. and Holt D. A. systematic approach to automatic edit and imputation. J. Am. Stat. Assoc., 71(353):17–35, 1976.
11. Lopatenko A. and Bertossi L.E. Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In Proc. 11th Int. Conf. on Database Theory, 2007, pp. 179–193.
12. Wijsen J. Database repairing using updates. ACM Trans. Database Syst., 30(3):722–768, 2005.

Content Delivery Networks

▶ Storage Grid

Content-and-Structure Query

Thijs Westerveld[1,2]
[1]Teezir Search Solutions, Ede, Netherlands
[2]CWI, Amsterdam, Netherlands

Synonyms

CAS query; CO+S query

Definition

A content-and-structure query is a formulation of an information need in XML retrieval or, more generally, in semi-structured text retrieval that includes explicit information about the structure of the desired result.

Key Points

Content-and-structure query is a term from semi-structured text retrieval, used predominantly for XML retrieval. The term refers to a specific way of querying a structured document collection. In addition to describing the (topical) content of the desired result, content-and-structure queries include explicit hints about the structure of the desired result or the structure of the context it appears in. Content-and-structure queries are useful for users who have knowledge about the collection structure and want to express the precise structure of the information they are after. For example, they can express the granularity of the desired results, e.g., return *sections* about architecture, or they can express the structural context of the information they are looking for, e.g., return *sections* about architecture within documents about *Berlin*. It is up to the retrieval system to decide how to use the structural hints in locating the most relevant information. In INEX, the Initiative for the Evaluation of XML Retrieval [1], content-and-structure queries are known as *CAS* queries or CO+S queries (*Content-Only queries* with

structural hints) and expressed in the NEXI language [2]. More information on query languages, including content-only and content-and-structured queries in the field of XML search can be found in [1].

Cross-references

- ▶ Content-Only Query
- ▶ NEXI
- ▶ XML Retrieval

Recommended Reading

1. Amer-Yahia S. and Lalmas M. XML search: languages, INEX and scoring. ACM SIGMOD Rec., 35(4):16–23, 2006.
2. Trotman A. and Sigurbjörnsson B. Narrowed extended xpath i (NEXI). In Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004. N. Fuhr, M. Lalmas, S. Malik, and Z. Szlavik (eds). Dagstuhl Castle, Germany, December 6–8, 2004, Revised Selected Papers , Vol. 3493. Springer, Berlin Heidelberg New York, GmbH, May 2005. http://www.springeronline.com/3-540-26166-4.

Content-based Image Retrieval (CBIR)

- ▶ Image Database

Content-Based Publish/Subscribe

Hans-Arno Jacobsen
University of Toronto, Toronto, ON, Canada

Definition

Content-based publish/subscribe is a communication abstraction that supports selective message dissemination among many sources and many sinks. The publication message content is used to make notification decisions. Subscribers express interest in receiving messages based on specified filter criteria that are evaluated against publication messages. Content-based publish/subscribe is an instance of the publish/subscribe concept.

Key Points

Content-based publish/subscribe is an instance of the publish/subscribe concept. In the content-based model clients interact by publishing messages and subscribing to messages through the publish/subscribe system. The key difference between the content-based and other publish/subscribe models is that the content of publication messages is used as the basis for disseminating publications to subscribers. Subscriptions consist of filters that specify subscriber interests making reference to publication message content. The publish/subscribe system matches publications against subscriptions by evaluating the publication message content against the filters expressed in subscriptions. The kind of content to subscribe to that exists in content-based publish/subscribe systems is either out-of-band information and must be know to clients, or is dynamically discoverable by clients based on additional support provided by the system.

A publication message published to the content-based publish/subscribe system is delivered to all subscribers with matching subscriptions. A subscription is a Boolean function over predicates. A publication matches a subscription, if the Boolean function representing it evaluates to true, otherwise the publication does not match the subscription.

As in the other publish/subscribe models, the content-based publish/subscribe model decouples the interaction among publishing data sources and subscribing data sinks. The same decoupling characteristics as discussed under the general publish/subscribe concept apply here as well. Specific realizations of this model found in practice vary in the exact decoupling offered. To properly qualify as publish/subscribe, at least the anonymous communication style must exist. That is publishing clients must not be aware of who the subscribing clients are and how many subscribing clients exist, and vice versa. Thus, content-based publish/subscribe enables the decoupled interaction of n sources with m sinks for $n, m \geq 1$.

In content-based publish/subscribe, the publication data model is defined by the data model underlying the definition of publication messages. The publication data model defines the structure and the type of publication messages processed by the system. In many approaches, a publication is a set of attribute-value pairs, where values are explicitly or implicitly typed. In explicit typing, each attribute-value pair has an additional type component specifying the type of the value. In implicit typing, no type is specified, and the type interpretation for matching is conveyed by the operator specified in the subscription referencing the

attribute. The type-based publish/subscribe concept is a refinement of this based on programming language type theory.

Some content-based publish/subscribe approaches exist that define multi-valued attributes, where an attribute may have more than one value. Also, publication schemas and patterns have been introduced that specify certain attributes as required, while others are optional. Besides representing publications as attribute-value pairs, many other representations of publications have been introduced in the literature, such as XML, RDF, and strings.

The subscription language model is closely tied to the publication data model and defines the subscriptions the publish/subscribe system processes. The subscription language model that corresponds to the above described attribute-value pair-based publication data model, represents subscriptions as Boolean functions over predicates. Most content-based publish/subscribe systems process conjunctions of predicates only. In these systems, more general subscriptions must be represented as separate conjunctive subscriptions. A predicate is an attribute- operator-value triple that evaluates to true or false. Besides representing subscriptions as Boolean functions over predicates, many other representations of publications have been introduced in the literature, such as XPath, RQL, regular expressions, and keywords.

Subscription language and publication data model define subscriptions and publications processed by the publish/subscribe system. The matching semantic defines when a publication matches a subscription. Commonly the matching semantic is crisp; that is a publication matches a subscription or it does not. However, other semantics have been explored, such as an approximate match, a similarity-based match, or even a probabilistic match.

The publish/subscribe matching problem is stated as follows: Given a set of subscriptions and a publication, determine the subscriptions that match for the given publication. The publish/subscribe system can be interpreted as a filter that based on the subscriptions it stores, publications that do not match are filtered out, while those that match are forwarded to subscribers that have expressed interest in receiving information by registering subscriptions. The challenge is to efficiently solve this problem without computing a separate match between all subscriptions and the given publication. This is possible since in many applications, subscriptions share predicates, subsumption relationships exist among different subscriptions, and the evaluation of one predicate may allow to determine the result of other predicates without requiring explicit computation.

Content-based publish/subscribe systems differ in the publication data model, the subscription language model, the matching semantic, and the system architecture. In a system based on a centralized architecture, all publishers and subscribers connect to one and the same publish/subscribe system. In a system based on a distributed architecture, publishers and subscribers connect to one of many publish/subscribe systems that are interconnected in a federation. The federated publish/subscribe system offers the same functionality and solves the same matching problem as the centralized one.

Content-based publish/subscribe differs from topic-based publish/subscribe in that the entire message content is used for matching, while in a topic-based approach only the topic associated with a message is used.

Content-based publish/subscribe differs from database stream processing in that the publish/subscribe system processes publications of widely varying schemas. In the extreme case, every publication processed by the system could be based on a different schema. In stream processing, each data stream follows one and the same schema, which is an important assumption in the design of the stream query engine.

Content-based publish/subscribe has been an active area of research, since at least the late 1990s. The early work in the area was influenced from approaches in active databases, network management, and distributed systems. Many academic and industry research projects have developed content-based publish/subscribe systems. Various standards exhibit elements of the above described content-based publish/subscribe model. These standards are the CORBA Notification Service [3], the OMG Data Dissemination Service [4], the OGF's Info-D specification [2], and the Advanced Message Queuing Protocol [1].

Content-based publish/subscribe intends to support applications that need to highly selectively disseminate messages from one or more data sources to several data sinks. The mapping of sources to sinks can change dynamically with every message published and is fully determined by the publication content and the at publication time existing subscriptions. Given no change in

the subscription set, one and the same message published twice, is sent to the same recipient set. Applications that require fine-grained filtering capabilities are ideally suited for realization with content-based publish/subscribe. Most existing publish/subscribe systems allow the application to dynamically change subscriptions at run-time. There are many applications that follow these characteristics. Examples include selective information dissemination, information filtering, database trigger processing, application-level firewalls, intrusion detection systems, and notification and altering services. Furthermore, recently it was demonstrated how higher-level applications can be build effectively with content-based publish/subscribe. Scenarios in this category are business activity monitoring, business process execution, monitoring and control of service level agreements, and automatic service discovery.

In the literature, the term content-based publish/subscribe refers to the above-described model and encompasses the centralized as well as the distributed realization of the publish/subscribe concept. In this context the terms matching and filtering are used interchangeably. The term content-based routing is reserved for the distributed realization of the model, where the publish/subscribe system is also referred to as a router, a broker, or a publish/subscribe message broker. In information retrieval, the publish/subscribe matching problem is referred to as information filtering. Subscriptions are then referred to as profiles or filters.

Cross-references

- ► Content-Based Routing
- ► Publish/Subscribe
- ► Type-Based Publish/Subscribe

Recommended Reading

1. AMQP Consortium. Advanced Message Queuing Protocol Specification, version 0–10 edition, 2008.
2. OGF. Information Dissemination in the Grid Environment Base Specifications, 2007.
3. OMG. Notification Service Specification, version 1.1, formal/04–10–11 edition, October 2004.
4. OMG. Data Distribution Service for Real-time Systems, version 1.2, formal/07–01–01 edition, January 2007.

Content-based Retrieval

- ► Multimedia Information Retrieval Model

Content-Based Video Retrieval

Cathal Gurrin
Dublin City University, Dublin, Ireland

Synonyms

Digital video search; Digital video retrieval

Definition

Content-based Video Retrieval refers to the provision of search facilities over archives of digital video content, where these search facilities are based on the outcome of an analysis of digital video content to extract indexable data for the search process.

Historical Background

As the volume of digital video data in existence constantly increases, the resulting vast archives of professional video content and UCC (User Created Content) are presenting an opportunity for the development of content-based video retrieval systems. Content-based video retrieval system development was initially lead by academic research such as the Informedia Digital Video Library [3] from CMU and the Físchlár Digital Video Suite [6] from DCU (Dublin City University). Both of these systems operated over thousands of hours of content, however digital video search has now become an everyday WWW phenomenon, with millions of items of digital video being indexed by the major WWW search engines and video upload sites. The early research focused content-based digital video systems, such as the offerings from CMU and DCU, exploited aspects of text search and content-based image search in order to provide intelligent indexing, retrieval, summarization and visualization of digital video content. In recent years, the emerging WWW search engines have focused on the textual indexing of large quantities of digital video, at the expense of performing complex and time-consuming visual content analysis.

Foundations

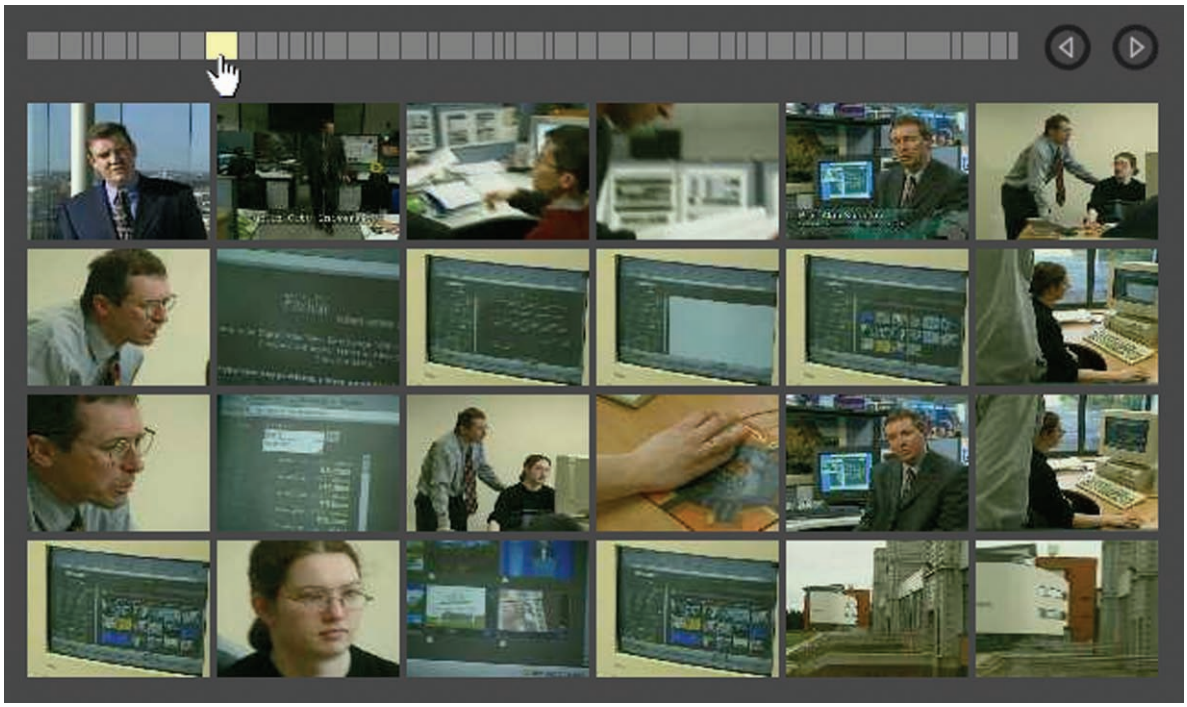
The aim of a content-based video search system is to answer user queries with a (ranked) list of appropriate video content. Many different sources of video content exist and each needs to be treated differently. Firstly there is professional created content such as TV news content, documentaries, TV programmes, sports video and many others. Professional content is directed and

polished content with many visual effects, such as a movie, music video or TV programme. Secondly there is the increasing quantities of UCC (User Created Content), such as home movie content or amateur/semi-professional content and finally there is security video content, which is increasingly being captured as the number of surveillance cameras in use increases. In addition to the type of content, another factor of key importance for content-based video search is the unit of retrieval.

Unit of Retrieval

Different content types will require different units of retrieval. In most WWW video search systems such as YouTube or Google Video, the unit of retrieval is the entire video content, which is sensible because most of the video content is short UCC or UUC (User Uploaded Content) clips. Retrieval of entire video units is not ideal for other types of content, for example TV news video, where the logical unit of retrieval would be a news story. There are a number of units of retrieval that are typically employed in addition to the entire video unit.

A *shot* in digital video is a sequence of continuous images (frames) from a single camera. A shot boundary is crossed when a recording instance ends and a new one begins. In many content-based video search systems the (automatically segmented) shot is the preferred unit of retrieval due to the fact that it is relatively easy to split a video file into its constituent shot in an automatic process called shot boundary detection [1]. Once a video stream has been segmented into shots, it can be browsed or indexed for subsequent search and retrieval, as shown in Fig. 1. A *scene* in digital video is a logical combination of video shots that together comprise some meaningful semantic unit. A *news story* is a special type of scene that is found in the context of news video. News stories can be automatically segmented from news video in a process called story-segmentation. This, like scene segmentation, is not a simple process, though reasonable accuracy can be achieved by replying on a number of individual cues from the video content and can be improved by exploiting the unique video production techniques of a particular broadcaster. For some video content, generating a *summary* or a *video skim* is a logical unit of retrieval. These summaries can be independent of any user need (context or query) or can be generated in response to a user need. Summaries have been successfully employed in the domain of field



Content-Based Video Retrieval. Figure 1. StoryBoard Interface from the Físchlár Video Retrieval System [2].

sports [4] or news summaries of reoccurring news topics [2]. Finally, as mentioned earlier, the entire video content may be returned in response to a user query, as is the case on many WWW video search engines in 2008.

Representing Video Content Visually on Screen

The quality of the interface to a content-based video retrieval system has a great effect on the usefulness of the system. In order to represent digital video visually, one or more keyframes (usually JPEG images) are usually extracted from the video to represent the content. These keyframes can then be employed for visual analysis of the video content by representing a video clip (typically a shot) by its keyframe and applying visual analysis techniques to the keyframe. In addition, these keyframes may be employed for visual presentation of video contents to support a degree of random access into the content. By processing video into a sequence of shots/scenes, and representing each shot/scene with one or more keyframes allows for the display of an entire video as a sequence of keyframes in what is called a StoryBoard interface, as shown in Fig. 1. Clicking on any keyframe would typically begin video playback from that point.

However, relying on simply presenting keyframes in screen can still require browsing through a very large information entity for long videos, maybe having over a hundred keyframes (shots) per hour. Therefore the ability to search within video content to locate a desired section of the video is desirable.

Searching Archives of Digital Video

The goals of supporting search through digital video archives are to (i) understand video content and (ii) understand how relevant content is likely to be to a user's query and to (iii) present the most highest ranked content for user consideration. We try to achieve these goals by indexing video content utilizing a number of sources (textual, audio and visual), either alone or in any combination. The unit of retrieval can be shots, scenes, stories, entire video units or any other unit of retrieval required.

Content-Based Retrieval using Text Sources The most common searching technology used for video retrieval in the WWW is content searching using proven text search techniques. This implies that it is possible to generate textual content (often referred to as a text surrogate) for the video. There are a number of sources of text content that can be employed to generate these textual surrogates, for example sources based on analyzing the digital video or the broadcast video stream:

- *Spoken words*, generated by utilizing a speech-to-text tool. The spoken words will provide an indication of the content of the video.
- *Written words*, extracted using a process of OCR (OCR – Optical Character Recognition) from the actual visual content of the video frames.
- *Professional closed caption annotation,* which are the closed caption (teletext) transcripts of video that accompanies much broadcast video content.

In addition there are many sources of textual evidence that can be employed that do not directly rely on the content of the digital video stream, and typically, these would be available for publicly available WWW digital video content:

- *Professionally annotated metadata* from the content provider which, if available, provides a valuable source of content for the textual surrogate.
- *Community annotated metadata* from general users of the content. On WWW video sharing sites users are encouraged to annotate comments about the video content and these annotations can be a valuable source of indexable content.

All of these sources of textual data can be employed alone, or in any combination to generate textual surrogates for video content (shots, scenes, stories or entire videos). Users can query such systems with conventional text queries and this is the way that most WWW video search engines operate. Text search through video archives is a very effective way to support search and retrieval and relies on well-known and proven text search techniques.

Content-Based Retrieval using Visual Sources

Digital Video, being a visual medium, can also be analyzed using visual analysis tools, which typically operate over individual keyframes to visually index each clip (typically a shot or scene). The visual content analysis tools are often borrowed from the domain of visual image analysis. The first generation of video analysis systems relied on modeling video with easily extractable low-level visual features such as color, texture and edge detection. However a significant

'semantic gap' exists between these low-level visual features and the semantic meaning of the video content, which is how a typical user would like to query a video search system. To help bridge this semantic gap, video content in the current generation of video search systems is processed to seek more complex semantic (higher-level or derived) visual concepts, such as people (faces, newsreaders), location (indoor/outdoor, cityscape/landscape), objects (buildings, cars, airplanes), events (explosions, violence) and production techniques such as camera motion. The output of these higher-level concept detectors can, with sufficient development and training, be successfully integrated (mainly as filters) into content-based video retrieval systems. However, the development of these concept detectors can be a difficult process and it is not reasonable to assume the development of tens of thousands of concept detectors to cover all concepts for the video archive. Research carried out by the Informedia team at CMU suggest that "concept-based" video retrieval with fewer than 5,000 concepts, detected with minimal accuracy of 10% mean average precision is likely to provide high accuracy results, comparable to text retrieval on the web, for a typical broadcast news video archive. Extending into other domains besides broadcast news may require some additional concepts. A review of image analysis techniques will provide more details of these semantic visual concept detectors and how they can be developed.

The output of easily extracted low-level feature analysis can be also employed in a content-based video retrieval system to support linking between visually similar content, though it is unlikely to be used to support direct user querying. Semantic features can form part of a user query, whereby a user, knowing the semantic factures extracted from a video archive, can specify semantic features that are required/not required in the result of a video search. For example, a user may request video content concerning forest fires, that also contains the feature 'fire'.

Content-Based Retrieval using Audio Sources

Apart from the speech-to-text there are other uses of audio sources for content-based video retrieval. For example, security video to identify non-standard audio events, such as a window breaking to provide a special access point to security video at this point. Key events in sports video can be identified using visual analysis (e.g., goal-mouth detection, or onscreen scoreboard changing) but also using audio analysis, for example crowd noise level or commentator excitement level.

Effective Retrieval

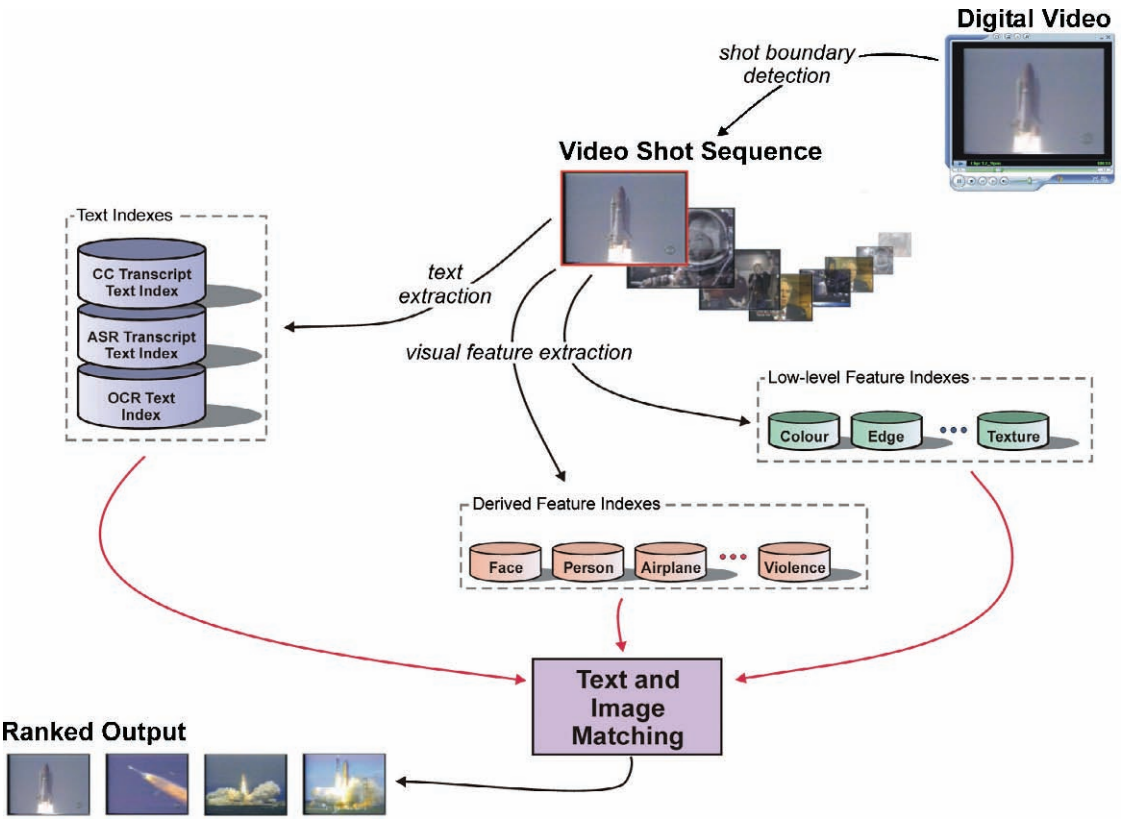
As can be seen from the current generation of WWW video search engines, most content-based video retrieval relies on user text queries to operate over text surrogates of video content. In typical situations the use of visual sources does not achieve any noticeable improvement in performance over using textual sources (such as CC text or ASR text). However, combining both sources of evidence can lead to higher performance than using either source alone. Figure 2 summarizes a typical shot-level content-based indexing process for digital video and illustrates some of the indexing options available.

Often successful academic video search systems allow the user to search to find the location in a piece of video which is likely to be of interest, with the user being encouraged to browse this area of the video by presenting keyframes from shots in the general video area (before and after).

Key Applications

The key application areas can be broadly divided into two categories; domain dependent video retrieval and generic (non-domain) video retrieval. In *domain dependent video retrieval*, the domain of the search system is limited, thereby allowing the search tool to exploit any domain dependent knowledge to develop a tailored and more effective content-based video retrieval system. Typical domains include news video where the unit of retrieval would a news story. Domain dependent additions for news video retrieval include anchor person detection to aid in the identification of news story bounds, inter-story linkage, story trails and timeline story progression. An example of a typical news story retrieval system is the Físchlár-News Digital Video Library that was operational from 2001 to 2004 at Dublin City University, and shown in Fig. 3.

Another example domain dependent application area is sports video, where research has been progressing on generating automatic summaries of field sports events and a third example is security video where research is ongoing into the automatic analysis of security footage to identify events of interest or even to identify and track individuals and objects in the video streams from many cameras in a given location.

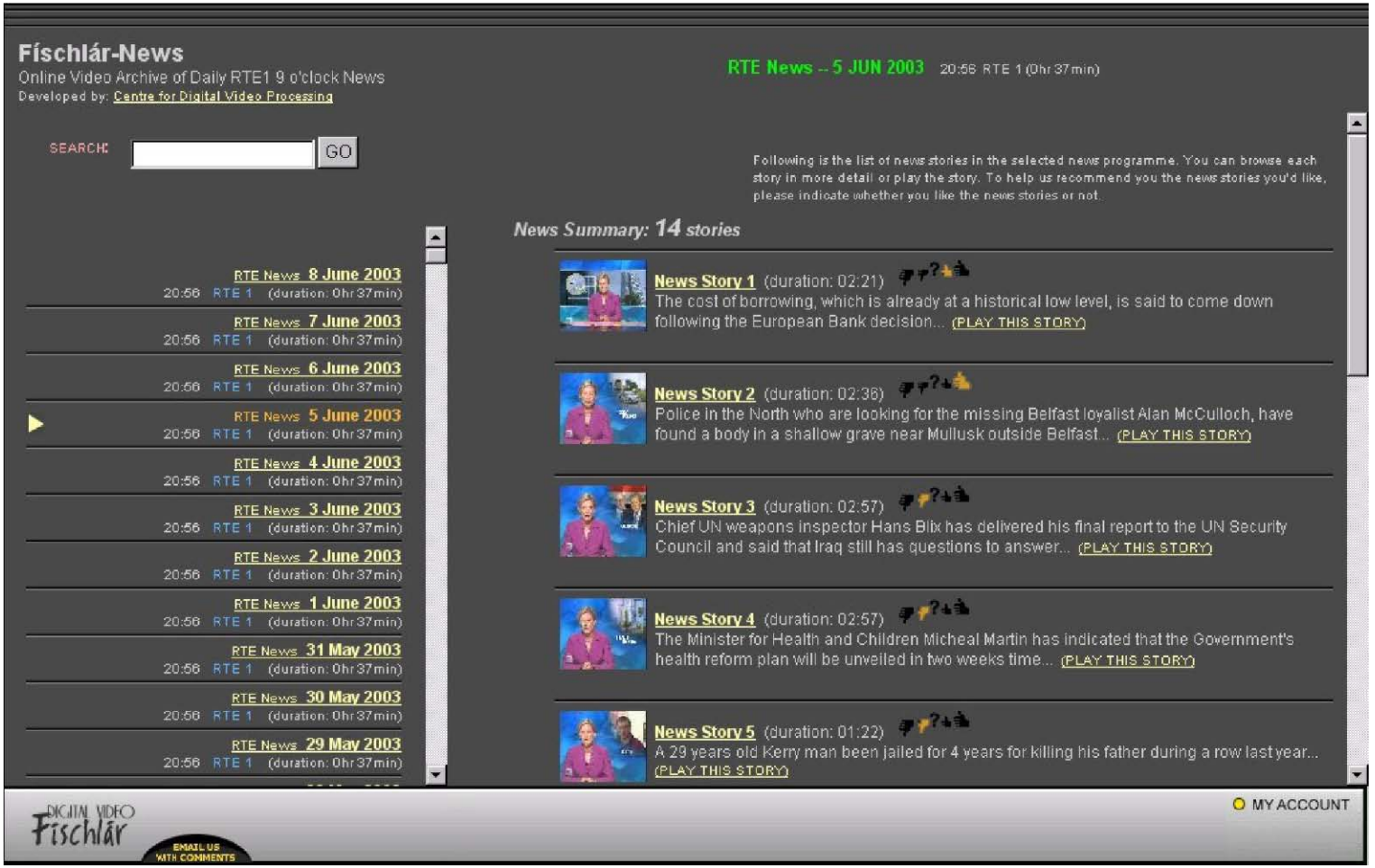


Content-Based Video Retrieval. Figure 2. The content-based indexing process for digital video (from Físchlár system at TRECVid in 2004) showing some text extraction, some low-level features and some higher level (derived) features (concepts).

Domain independent video retrieval attempts to index all types of content, such as general TV programmes or generic UCC. Given that there are no domain specific clues to exploit, retrieval is usually on textual indexing of a textual surrogate or extracted visual concepts, such as objects [5], locations, people, etc. The unit of retrieval would typically be a shot or an entire video clip, but could also be a non-shot unit that matches the user request. Domain independent video retrieval is most commonly seen in WWW video search engines, which index content based on text surrogates and returns entire videos in the result set.

Future Directions Future applications of, and research into content-based video search will likely focus on developing techniques for providing access to large archives of digital video content as broadcasters continue the process of digitizing their huge archives of programmes and the raw video content (rushes) that is used in the making of TV programmes. For rushes content especially, one will not be able to rely on text transcripts for indexing purposes. In addition, the ever increasing volume of UCC requires content-based retrieval techniques to be developed that will provide an improved semantic search facility over this content. Finally, the third point of research into the future will likely be in migrating content-based retrieval tools onto consumer devices (PVRs for example), which themselves are becoming capable of storing hundreds of hours of recorded video and UCC.

Experimental Results In the field of content-based video retrieval there exists an annual, worldwide forum for the evaluation of techniques for video search, called TRECVid [7] which began in 2001. The TRECVid workshop is (2007) part of the TREC [8] conference series is sponsored by the National Institute of Standards and Technology (NIST). In 2007, 54 teams from Europe, the Americas, Asia, and Australia participated in TRECVid. Over the course of the TRECVid evaluations, data employed has been either TV news, documentaries, educational video and rushes



Content-Based Video Retrieval. Figure 3. Fischlár-News, a domain dependent content-based video retrieval system.

(Rushes content, is the unproduced content that is used to prepare TV programming.) content. TRECVid has organized a number of tasks for the annual evaluations which may change each year. The tasks evaluated, 2001, have included shot boundary determination, interactive and automatic (no query modification or browsing) video search, high-level concept detection, story boundary determination for TV news and camera motion analysis, among others.

In addition to TRECVid, other evaluation forums also exist such as Video Analysis and Content Extraction (VACE) which is a US program that addresses the lack of tools to assist human analysts monitor and annotate video for indexing. The video data used in VACE is broadcast TV news, surveillance, Unmanned Aerial Vehicle, meetings, and ground reconnaissance video. Other evaluation forums such as the French ETISEO and EU PETS evaluations have evaluated content-based retrieval (event detection and object detection) from surveillance video. ARGOS, sponsored by the French government, evaluated tasks similar to TRECVid and employed video data from TV news, scientific documentaries and surveillance video archives.

Some summary findings from content-based video retrieval research are that employing visual analysis of the video content does not provide a significant increase in search performance over using text transcripts, that text transcripts provide the single most important clue for searching content, that employing as many text sources as possible aids text search quality, and finally that, incorporating visual content search can improve retrieval over that of text indexing alone. The performance of visual indexing tools suggests that this is an unsolved problem with much research needed. As an example, the highest accuracy attained (in terms of Inferred Average Precision) for the twenty visual concepts evaluated at TRECVid in 2007 are shown in Table 1.

Finally, it should be noted that the interface to an interactive video search system (for example [3,6]) can make a huge difference for effective content-based video search and retrieval. Content searching through text transcripts can locate the area of the video, but a good storyboard interface to find the exact video shot of interest is a valuable addition.

Data Sets The TRECVid evaluation framework provides a number of datasets to support the comparative and repeatable evaluation of TREC. Since 2001, these datasets, along with the associated queries and relevance judgements are available. The video data employed in these datasets comes from various sources, such as the video from the Movie Archive of the Internet Archive, news video data in a number of languages (English, Arabic and Chinese) and rushes content. Datasets used in other evaluation forums are also available

Content-Based Video Retrieval. Table 1. Inferred Average Precision (infAP (In terms of infAP, a value of 1.0 infers that the technique locates only correct examples of the concept, whereas a value of 0.0 infers that the technique only locates incorrect examples)) measurement for the top performing techniques for visual concept detection at the TRECVid workshop in 2007

CONCEPT	infAP	CONCEPT	infAP
Sports	0.144	Computer/TV screen	0.209
Weather	0.062	US flag	0.041
Office	0.222	Airplane	0.226
Meeting	0.279	Car	0.265
Desert	0.155	Truck	0.108
Mountain	0.12	Boat/ship	0.212
Waterscape/ waterfront	0.374	People marching	0.104
Police/security	0.046	Explosion/fire	0.069
Military personnel	0.081	Maps	0.236
Animal	0.249	Charts	0.225

Cross-references

- ▶ Video Abstraction
- ▶ Video Content Analysis
- ▶ Video Content Modeling
- ▶ Video Metadata
- ▶ Video Representation
- ▶ Video Scene and Event Detection
- ▶ Video Segmentation
- ▶ Video Shot Detection
- ▶ Video Skimming
- ▶ Video Summarization

Recommended Reading

1. Browne P., Smeaton A.F., Murphy N., O'Connor N., Marlow S., and Berrut C. Evaluating and combining digital video shot boundary detection algorithms. In Proc. IMVIP 2000 – Irish Machine Vision and Image Processing Conference, 2000, pp. 93–100.

2. Christel M.G., Hauptmann A.G., Wactlar H.D., and Ng T.D., Collages as dynamic summaries for news video. In Proc. 10th ACM Int. Conf. on Multimedia, 2002, pp. 561–569.
3. Hauptmann A. lessons for the future from a decade of informedia video analysis research, image and video retrieval. In Proc. 4th Int. Conf. Image and Video Retrieval, 2005, pp. 1–10.
4. Sadlier D. and O'Connor N. Event detection in field sports video using audio-visual features and a support vector machine. IEEE Trans. Circuits Syst. Video Technol., 15(10):1225–1233, 2005.
5. Sivic J. AND Zisserman A. Video Google: a text retrieval approach to object matching in videos. In Proc. 9th IEEE Conf. Computer Vision, Vol. 2, 2003, pp. 1470–1477.
6. Smeaton A.F., Lee H., and Mc Donald K. Experiences of creating four video library collections with the Físchlár system. Int. J. Digit. Libr., 4(1):42–44, 2004.
7. Smeaton A.F., Over P., and Kraaij W. Evaluation campaigns and TRECVid. In Proc. 8th ACM SIGMM Int. Workshop on Multimedia Information Retrieval, 2006, pp. 321–330.
8. http://trec.nist.gov Last visited June '08.

Content-Only Query

Thijs Westerveld[1,2]
[1]Teezir Search Solutions, Ede, The Netherlands
[2]CWI, Amsterdam, The Netherlands

Synonyms
Content-only query; CO query

Definition
A content-only query is a formulation of an information need in XML retrieval or, more generally, in semi-structured text retrieval that does not contain information regarding the structure of the desired result.

Key Points
Content-only query or CO query is a term from semi-structured text retrieval, used predominantly for XML retrieval. The term refers to a specific way of querying a semi-structured document collection. Content-only queries ignore the structure of the collection and only refer to the (topical) content of the desired result. In that sense, they are similar to the keyword queries typically used in traditional information retrieval systems or in web search engines. The fact that structural information is lacking from the query formulation does not mean structure does not play a role. When a content-only query is posed, it is up to the retrieval system to decide the appropriate level of granularity to satisfy the information need. This contrasts so-called *content-and-structure queries* where the user specifies structural clues regarding the desired result. More information on query languages, including content-only and content-and-structured queries in the field of XML search can be found in [1].

Cross-references
▶ Content-and-structure query
▶ NEXI
▶ Xml Retrieval

Recommended Reading
1. Amer-Yahia S. and Lalmas M. XML search: languages, INEX and scoring. ACM SIGMOD Rec., 35(4):16–23, 2006.

Content-oriented XML Retrieval

▶ XML Retrieval

Context

Opher Etzion
IBM Research Lab in Haifa, Haifa, Israel

Synonyms
Life-span (in part); Space-span (in part)

Definition
A context is a collection of semantic dimensions within which the event occurs. These dimensions may include: temporal context, spatial context, state-related context and reference-related context.

Key Points
Event processing is being done within context, which means that an event is interpreted differently in different contexts, and may trigger different actions in different contexts, or be irrelevant in certain context. In the event processing network, each agents operates within a single context. While the term context has been associated with the spatial dimension, in event processing it is most strongly associated with the temporal dimension.

Each context-dimension may be specified either explicitly, or by using higher level abstractions.

Examples are:

- Temporal context:
  - Explicit: Everyday between 8AM–5PM EST.
  - Implicit: From sunrise to sunset.
  - Mixed: Within two hours from admission to the hospital.
- Spatial context:
  - Explicit: Within 1 KM from coordinate + 51° 3′ 45.71″, −1° 18′ 25.56″.
  - Implicit: Within the borders of the city of Winchester.
  - Mixed: Within 1 KM north of the border between Thailand and Laos.
- State-oriented context:
  - Explicit: When "red alert" is present.
  - Implicit: During traffic jam in the area.
- Reference-oriented context:
  - Explicit: Context-instance for each platinum-customer with credit-limit > $1M.
  - Implicit: Context-instance for each "angry customer."

Note that the state-oriented dimension is different, since it does not relate to the event itself, and is global in nature. A context may consist of one dimension only or combination of dimensions. The reference-oriented context is mainly used to partition the event space. Context instances may or may not cover the entire space of possibilities, a context can also be created from binary operations on contexts (union, intersection, difference).

Cross-references

► Complex Event Processing
► Event Processing Network
► Retrospective Event Processing

Recommended Reading

1. Adi A., Biger A., Botzer D., Etzion O., and Sommer Z. Context awareness in Amit. In Proc. 5th Annual Workshop on Active Middleware Services, 2003, pp. 160–167.
2. Barghouti N.S. and Krishnamurthy B. Using event contexts and matching constraints to monitor software processes. In Proc. 17th Int. Conf. on Software Eng., 1995, pp. 83–92.
3. Buvac S. Quantificational logic of context. In Proc. 10th National Conf. on AI, 1996, pp. 600–606.
4. Hong C., Lee K., Suh Y., Kim H., Kim H., and Lee H. Developing context-aware system using the conceptual context model. In Proc. 6th IEEE Int. Conf. on Information Technology, 2006, pp. 238.
5. Rakotonirainy A., Indulska J., Loke S.W., and Zaslavsky A. Middleware for reactive components: An integrated use of context, roles, and event based coordination. In Proc. IFIP/ACM Int. Conf. on Dist. Syst. Platforms, 2001, pp. 77–98.

Context-aware Interfaces

► Adaptive Interfaces

Contextual Advertising

► Web Advertising

Contextualization

Jaana Kekäläinen, Paavo Arvola, Marko Junkkari
University of Tampere, Tampere, Finland

Definition

In relation to structured text retrieval, contextualization means estimating the relevance of a given structural text unit with information obtainable from – besides the unit itself – the surrounding structural text units, that is, from the context of the unit. From now on, structural text units are referred to as elements in accordance with [4]. In other words, in contextualization it is assumed that the context of an element gives hints about the relevance of the element.

Historical Background

Structured information retrieval typically addresses documents marked-up with, for instance, SGML or XML. In this article, XML documents are used as a sample case of structured documents. These documents have a hierarchical structure, which is often represented as a tree. In structured text retrieval, like in information retrieval (IR) in general, querying is based on words representing the information needed. Structural conditions, concerning the tree, may or may not be added to the query. An information retrieval system (IRS) returns a list of elements ranked by their retrieval status values (RSV), which are scores given by the IRS. Typically, RSVs are based on the statistics of

words (cf. definitional entry *Term statistics*) appearing in the element and the query, although other features of the document or its structure may be used in addition.

The idea of XML retrieval is not to return whole documents but those elements that are best matches to the query–relevant elements with the least irrelevant content. The length of the textual content varies as the size of the elements varies in the hierarchy, so that descendant elements often have less text than their ancestors. As a consequence, small elements down in the hierarchy may have too few words in common with the query, that is, too little evidence to be matched with the query, although they might be more exact matches than their ancestors. This problem, known as the vocabulary mismatch, is typical for text retrieval, and is caused by natural language allowing several ways to refer to objects. However, elements are often dependent on each other because of textual cohesion. Thus, one solution is to use the context of the element to give more evidence about the subject of the element. One could say that "good elements" appear in "good company." This approach was first proposed by Sigurbjörnsson, Kamps, and de Rijke in [8].

Another problem related to the nested structure of structured texts is the calculation of word statistics. There are no obvious indexing units like in nonstructured (flat) text retrieval (cf. entry *Indexing units*). The calculation of word weights is challenging since the length of the elements vary, which has effects on word frequencies. Moreover, inverted element frequencies, corresponding to inverted document frequencies (*idf*s) in weight calculation, vary depending on the indexing unit. As a solution for this, the concept of *document pivot* factor (originally introduced in [9] for classical document retrieval) is suggested by Mass and Mandelbrod [6] to scale the final relevance status value of an element in XML IR. In [6] the scaling is based on the document pivot factor, RSV of the topmost ancestor (the root element), and the RSV of the element. This can be regarded as contextualization though for different reasons than in the first mentioned case.

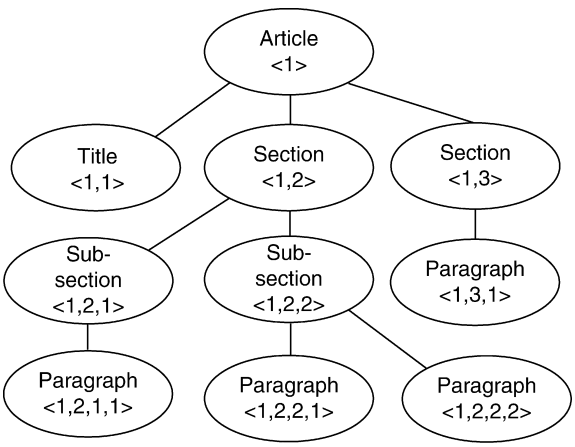
Foundations

An XML document consists of elements, which in turn may contain smaller elements. If an element x contains immediately another element y, then x is called the parent of y, whereas y is called a child of x. Any element containing x is an ancestor of y, and y is a descendant of those elements. A sample XML document is represented as a tree in Fig. 1. The document is an article consisting of a title, sections, subsections, and paragraphs. All elements are labeled with Dewey indices for reference.

Depending on how a collection is organized, an element may be viewed at various levels of context. For example, assuming that documents follow an article-section-subsection-paragraph division as in the sample, then the article, the section and the subsection form different levels of context for a paragraph. Further, a subsection can be viewed in the contexts of the section or article. The length of the path from the context element to the element at hand determines the level of context. For example, the parent of an element determines the first level context; the ancestor with the path length 2 determines the second level context, etc. The root element forms the topmost context.

As an example, the paragraph labeled <1,2,2,1> in Fig. 1 is examined. Now Subsection <1,2,2> forms the first level context and Section <1,2> the second level context of this paragraph. The article is the root element, or it determines the topmost context of this paragraph. In turn, Section <1,2> forms the first level context, and the article the second level (or topmost) context of Subsection<1,2,2>. The article possesses no context.

The idea of contextualization is based on the assumption that an element in a relevant context should be ranked higher than a similar element in a nonrelevant context. In contextualization, the RSV of an element is tuned by the RSV of its context element(s). If the RSV of the context is low (predicting nonrelevance), the RSV



Contextualization. Figure 1. A sample document tree.

of the element should be decreased; if the RSV of the context is high (predicting relevance), the RSV of the element should be increased. Here low and high are relative to the element RSV and the RSVs of other contexts.

As an example three special contextualization cases are considered – parent, root or all ancestors as a context. In defining contextualization function the following notations, presented in [1], are used:

parent(e) yields the parent element of the element e,
root(e) yields the root element of the element e,
$\langle e_1,e_2,...,e_n\rangle$ means the path from the root element e_1 to its descendant e_n such that $\forall i \in \{1,...,n-1\}$ holds $e_i = parent(e_{i+1})$, and
$w(q, e)$ denotes the RSV of the element e with respect to the query q.

With relation to the query expression q, contextualization based on the nearest context (parent) of the element e can be calculated by averaging the RSVs of the element e and its parent element. Averaging is applied as an example because it increases the RSV of the element whose context's RSV is higher, and decreases the RSV of such elements whose context's RSV is lower. The function for this is denoted by the symbol c_p:

$$c_p(q, e) = \frac{w(q, e) + w(q, parent(e))}{2}.$$

The contextualization by the topmost context is denoted by the function symbol c_r and it can be calculated by averaging the RSVs of the element e and its root element.

$$c_r(q, e) = \frac{w(q, e) + w(q, root(e))}{2}.$$

The contextualization function c_t is called tower contextualization and it yields the average of the RSVs of all the elements within the path from the root element to the element e, that is, all ancestors.

$$c_t(q, e) = \frac{\sum_{i=1}^{n} w(q, e_i)}{n},$$

when $e_1 = root(e)$ and $e_n = e$ in the path $\langle e_1,...,e_n\rangle$.

Now, Dewey indices are utilized as a method for handling the XML tree structure. In the following, the contextualization for any path between the element and the root is generalized. The notations used are as follows:

- The symbol ξ is used for denoting a Dewey index labeling an element. An element possessing the index ξ is called the ξ *element*.
- The set of indices related to the XML collection at hand is denoted by *IS*.
- The length of an index ξ is denoted by $len(\xi)$. For example $len(\langle 1,2,2,3\rangle)$ is 4.
- The index $\langle i\rangle$ consisting of an integer i (i.e., its length is 1) is called *root index* and it is associated with the whole document.
- Let ξ be an index and i a positive integer, then the *cutting operation* $\delta_i(\xi)$ selects the sub-index of the index ξ consisting of its i first integers. For example if $\xi = \langle a,b,c\rangle$ then $\delta_2(\xi) = \langle a,b\rangle$. In terms of the cutting operation the root index at hand is denoted by $\delta_1(\xi)$ whereas the index of the parent element can be denoted by $\delta_{len(\xi)-1}(\xi)$.

A general contextualization function C has the following arguments: q, ξ, and g. The arguments q (query) and ξ (index) are defined above. The argument g is called *contextualization vector* and is set-theoretically represented as a tuple, consisting of values by which elements between the root element and ξ element are weighted in contextualization. The length of g is $len(\xi)$. When referring to the ith position of the contextualization vector g, the notation $g[i]$ is used. For example, if $g = \langle a,b,c\rangle$ then $g[2] = b$. The value in $g[i]$ relates to the element with the index $\delta_i(\xi)$. For example, if $\xi = \langle 1,2,2\rangle$ then g is the 3-tuple $\langle a,b,c\rangle$ where a is the contextualization weight of the root element $\langle 1\rangle$ (i.e., the element with index $\delta_1(\xi)$), b is the contextualization weight of the $\langle 1,2\rangle$ element (i.e., the element with index $\delta_2(\xi)$), and c is the weight of the $\langle 1,2,2\rangle$ element (i.e., the element with the index $\delta_{len\ (\xi)}(\xi)$). The contextualized RSVs of elements are calculated by weighted average based on the contextualization vector and the index at hand. In the sample case above the contextualized RSV is calculated as $(a * w(q, \langle 1\rangle) + b * w(q, \langle 1,2\rangle) + c * w(q, \langle 1,2,2\rangle))/(a + b + c)$. Contextualization is applied only to those elements whose basic RSV is not zero. The general contextualization function C is formally defined:

$$C(q, \xi, g) = \begin{cases} 0, & \text{if } w(q, \xi) = 0 \\ \dfrac{\sum_{i=1}^{len(\xi)} g[i] \cdot w(q, \delta_i(\xi))}{\sum_{i=1}^{len(\xi)} g[i]}, & \text{otherwise.} \end{cases}$$

The values in g are not bound to any range. This means that in terms of g, different levels of the context can be weighted in various ways. For example, weighting may increase or decrease toward the topmost context (root element).

Now, parent and root contextualization with the given generalized notation are considered. For simplicity, binary contextualization weights are used, that is, only such cases where the values of g are either 1 or 0. Zero value means that the corresponding element is not taken into account in the contextualization. With relation to a query expression q, the contextualization based on the first level (parent) context of the ξ element is calculated using the contextualization vector where two last elements have the value 1 and the others zero value. This function is denoted $c_p(q, \xi)$ and defined as follows:

$$c_p(q, \xi) = C(q, \xi, g)$$

where g =

$$\begin{cases} g[len(\xi)] = 1 \\ g[len(\xi) - 1] = 1, \text{when len}(\xi) > 1 \\ g\ \underset{i=1}{[i]}\ = 0, \text{when len}(\xi) > 2 \end{cases}$$

The contextualization by the topmost context (or by the root element) is denoted by the function symbol c_r. In this case the weights for the first and the last element are 1 and the other weights are 0 in the contextualization vector.

$$c_r(q, \xi) = C(q, \xi, g) \text{ where } g = \begin{cases} g[len(\xi)] = 1 \\ g[1] = 1 \\ g\ \underset{i=2}{\overset{len(\xi)-1}{[i]}}\ = 0, \\ \text{when } len(\xi) > 2. \end{cases}$$

There are alternative interpretations and presentations of the idea of contextualization, for example [8,6,10]. The idea of mixing evidence from the element itself and its surrounding elements was presented for the first time by Sigurbjörnsson, Kamps, and de Rijke [8]. They apply language modeling based on a mixture model. The final RSV for the element is combined from the RSV of the element itself and the RSV of the root element as follows:

$$w_{mix}(q, e) = lp(e) + \alpha \cdot w(q, root(e)) + (1 - \alpha) \cdot w(q, e),$$

where the notation is as given above; $w_{mix}(q,e)$ is the combined RSV for the element, $lp(e)$ is the length prior, and α is a tuning parameter.

In [6] independent indices are created for different – selected – element types. Some indices have sparse data compared with others, that is, all the words of root elements are not contained in lower level elements. This has effects on inverted element frequencies and comparability of the word weights across the indices. To resolve the problem the final element RSV is tuned by a scaling factor and the RSV of the root element. With the notation explained above this can be represented as follows:

$$w_p(q, e) = DocPivot \cdot w(q, root(e)) + (1 - DocPivot) \cdot w(q, e)$$

where $w_p(q,e)$ is the pivoted RSV of the element, and *DocPivot* is the scaling factor.

In [10], the structural context of the element is utilized for scoring elements. Original RSVs are obtained with the Okapi model [7]. For this, word frequencies are calculated for elements rather than documents, and normalized for each element type. The combined RSV for the element is calculated as a sum of the RSV of the context and the original RSV for the element, each score multiplied by a parameter. Machine learning approach is applied for learning the parameters. Let $x = \{t_1, t_2,...,t_d\}$ be a vector of features representing the element e. The features are RSVs of the element e and its different contexts. Then the combined RSV is

$$f\omega(x) = \sum_{j=1}^{d} \omega_j t_j,$$

where $\omega = \{\omega_1, \omega_2,...,\omega_d\}$ are the parameters to be learned. The approach was tested with the parent and root as the contexts.

Key Applications

Contextualization and similar approached have been applied in XML retrieval with different retrieval models: tf-idf based ranking and structural indices [1], the vector space model [6], and the language modeling [8]. The key application is element ranking. The results obtained with different retrieval systems seem to indicate that the root contextualization is the best alternative.

In traditional text retrieval a similar approach has been applied to text passages [3,5]. The idea of contextualization is applicable to all structured text documents; yet the type of the documents to be retrieved has effects on contextualization. Lengthy documents with one or few subjects are more amenable for the method than short documents, or documents with diverse subjects; contextualization might not work in an encyclopedia

with short entries, but can improve effectiveness, say, in the retrieval of the elements of scientific articles.

Experimental Results

The effectiveness of contextualization in XML retrieval has been experimented with the INEX test collection consisting of IEEE articles [1,2]. All three contextualization types mentioned above (parent, root, and tower contextualization) were tested. The results show clear improvement over a non-contextualized baseline; the best results were obtained with the root and tower contextualization. Additionally, approaches suggested in [8,6,10] were tested with the same INEX collection and reported to be effective compared with non-contextualized baselines, that is, compared with ranking based on elements' basic RSVs.

Cross-references

- ▶ Indexing Units
- ▶ Term Statistics for Structured Text Retrieval
- ▶ XML Retrieval

Recommended Reading

1. Arvola P., Junkkari M., and Kekäläinen J. Generalized contextualization method for XML information retrieval. In Proc. Int. Conf. on Information and Knowledge Management, 2005, pp. 20–27.
2. Arvola P., Junkkari M., and Kekäläinen J. Query evaluation with structural indices. In Proc. 4th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005, pp. 134–145.
3. Callan J.P. Passage-level evidence in document retrieval. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 302–310.
4. Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation 16 August 2006. Available at: http://www.w3.org/TR/xml/[retrieved 17.8.2007].
5. Kaszkiel M., Zobel J., and Sacks-Davis R. Efficient passage ranking for document databases. ACM Trans. Infor. Syst., 17(4):406–439, 1999.
6. Mass Y. and Mandelbrod M. Component ranking and automatic query refinement for XML retrieval. In Proc. 4th Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2005, pp. 73–84.
7. Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M.M., and Gatford M. Okapi at TREC-3. In Proc. The 3rd Text Retrieval Conf., 1994, pp. 500–226.
8. Sigurbjörnsson B, Kamps J., and De Rijke M. An element-based approach to XML retrieval. In Proc. 2nd Int. Workshop of the Initiative for the Evaluation of XML Retrieval, 2003, 19–26. Available at: http://inex.is.informatik.uni-duisburg.de:2003/proceedings.pdf [retrieved 29.8.2007].
9. Singhal A., Buckley C., and Mitra M. Pivoted document length normalization. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996, 21–29.
10. Vittaut J.-N. and Gallinari P. Machine learning ranking for structured information retrieval. In Proc. 28th European Conf. on IR Research, 2006, 338–349.

Continuous Backup

▶ Continuous Data Protection (CDP)

Continuous Data Feed

▶ Data Stream

Continuous Data Protection

Kenichi Wada
Hitachi, Ltd, Tokyo, Japan

Synonyms

Continuous backup
CDP

Definition

CDP is a data protection service capturing data changes to storage, often providing the capability of restoring any point in time copies.

Key Points

CDP differs from usual backups in that users do not need to specify the point in time until they recover data from backups. From an application point of view, every time when it updates data in an original volume, CDP keeps updates. In case of recovery, when users specify the point in time, CDP creates the point in time copy from an original volume and updates.

In several CDP implementations, users can specify the granularities of restorable objects which help them to specify the point in time easily. For example, restorable objects range from crash-consistent images to logical objects such as files, mail boxes, messages, database files, or logs.

Cross-references

- ▶ Backup and Restore

Recommended Reading

1. Laden G., et al. Architectures for Controller Based CDP. In Proc. 5th USENIX conf. on File and Storage Technologies, 2007, pp. 107–121.

Continuous Monitoring of Spatial Queries

Kyriakos Mouratidis
Singapore Management University, Singapore, Singapore

Synonyms

Spatio-temporal stream processing

Definition

A continuous spatial query runs over long periods of time and requests constant reporting of its result as the data dynamically change. Typically, the query type is range or nearest neighbor (NN), and the assumed distance metric is the Euclidean one. In general, there are multiple queries being processed simultaneously. The query points and the data objects move frequently and arbitrarily, i.e., their velocity vectors and motion patterns are unknown. They issue location updates to a central server, which processes them and continuously reports the current (i.e., updated) query results. Consider, for example, that the queries correspond to vacant cabs, and that the data objects are pedestrians that ask for a taxi. As cabs and pedestrians move, each free taxi driver wishes to know his/her closest client. This is an instance of continuous NN monitoring. Spatial monitoring systems aim at minimizing the processing time at the server and/or the communication cost incurred by location updates. Due to the time-critical nature of the problem, the data are usually stored in main memory to allow fast processing.

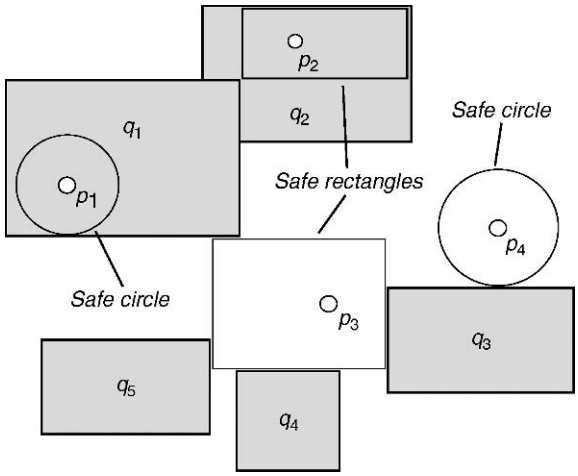
Historical Background

The first algorithms in the spatial database literature process snapshot (i.e., one-time) queries over static objects. They assume disk-resident data and utilize an index (e.g., an *R-tree*) to restrict the search space and reduce the I/O cost. Subsequent research considered spatial queries in client-server architectures. The general idea is to provide the user with extra information (along with the result at query-time) in order to reduce the number of subsequent queries as he/she moves (see entry *Nearest Neighbor Query*). These methods assume that the data objects are either static or moving linearly with known velocities. Due to the wide availability of positioning devices and the need for improved location-based services, the research focus has recently shifted to continuous spatial queries. In contrast with earlier assumed contexts, in this setting (i) there are multiple queries being evaluated simultaneously, (ii) the query results are continuously updated, and (iii) both the query points and the data objects move unpredictably.

Foundations

The first spatial monitoring method is called *Q-index* [13] and processes static range queries. Based on the observation that maintaining an index over frequently moving objects is very costly, *Q-index* indexes the queries instead of the objects. In particular, the monitored ranges are organized by an *R-tree*, and moving objects probe this tree to find the queries that they influence. Additionally, *Q-index* introduces the concept of *safe regions* to reduce the number of location updates. Specifically, each object p is assigned a circular or rectangular region, such that p needs to issue an update only if it exits this area (because, otherwise, it does not influence the result of any query). Figure 1 shows an example, where the current result of query q_1 contains object p_1, that of q_2 contains p_2, and the results of q_3, q_4, and q_5 are empty. The safe regions for p_1 and p_4 are circular, while for p_2 and p_3 they are rectangular. Note that no query result can change unless some objects fall outside their assigned safe regions. Kalashnikov et al. [4] show that a grid implementation of *Q-index* is more efficient (than *R-trees*) for main memory evaluation.

Monitoring Query Management (MQM) [1] and *MobiEyes* [2] also monitor range queries. They further exploit the computational capabilities of the objects to reduce the number of updates and the processing load of the server. In both systems, the objects store locally the queries in their vicinity and issue updates to the server only when they cross the boundary of any of these queries. To save their limited computational capabilities, the objects store and monitor only the queries they may affect when they move. MQM and *MobiEyes* employ different strategies to identify these queries. The former applies only to static queries. The latter can also handle moving ones, making however the assumption that they move linearly with fixed velocity.



Continuous Monitoring of Spatial Queries. Figure 1. Circular and rectangular safe regions.

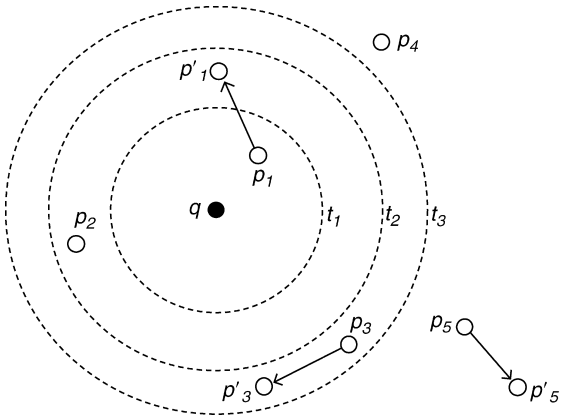
Mokbel et al. [7] present *Scalable INcremental hash-based Algorithm* (SINA), a system that monitors both static and moving ranges. In contrast with the aforementioned methods, in SINA the objects do not perform any local processing. Instead, they simply report their locations whenever they move, and the objective is to minimize the processing cost at the server. SINA is based on *shared execution* and *incremental evaluation*. Shared execution is achieved by implementing query evaluation as a spatial join between the objects and the queries. Incremental evaluation implies that the server computes only updates (i.e., object inclusions/exclusions) over the previously reported answers, as opposed to re-evaluating the queries from scratch.

The above algorithms focus on ranges, and their extension to NN queries is either impossible or non-trivial. The systems described in the following target NN monitoring. Hu et al. [3] extend the safe region technique to NN queries; they describe a method that computes and maintains rectangular safe regions subject to the current query locations and *k*NN results. Mouratidis et al. [11] propose *Threshold-Based algorithm* (TB), also aiming at communication cost reduction. To suppress unnecessary location updates, in TB the objects monitor their distance from the queries (instead of safe regions). Consider the example in Fig. 2, and assume that q is a continuous 3-NN query (i.e., $k = 3$). The initial result contains p_1, p_2, p_3. TB computes three thresholds (t_1, t_2, t_3) which define a range for each object. If every object's distance from q lies within its respective range, the result of the query is guaranteed to remain unchanged. Each threshold is set in the middle of the distances of two consecutive objects from the query. The distance range for p_1 is $[0, t_1)$, for p_2 is $[t_1, t_2)$, for p_3 is $[t_2, t_3)$, and for p_4, p_5 is $[t_3, \infty)$. Every object is aware of its distance range, and when there is a boundary violation, it informs the server about this event. For instance, assume that p_1, p_3, and p_5 move to positions p'_1, p'_3 and p'_5, respectively. Objects p_3 and p_5 compute their new distances from q, and avoid sending an update since they still lie in their permissible ranges. Object p_1, on the other hand, violates its threshold and updates its position to the server. Since the order between the first two NNs may have changed, the server requests for the current location of p_2, and updates accordingly the result and threshold t_1. In general, TB processes all updates issued since the last result maintenance, and (if necessary) it decides which additional object positions to request for, updates the k NNs of q, and sends new thresholds to the involved objects.

All the following methods aim at minimizing the processing time. Koudas et al. [6] describe *aDaptive Indexing on Streams by space-filling Curves* (DISC), a technique for *e*-approximate *k*NN queries over streams of multi-dimensional points. The returned (*e*-approximate) *k*th NN lies at most *e* distance units farther from q than the actual *k*th NN of q. DISC partitions the space with a regular grid of granularity such that the maximum distance between any pair of points in a cell is at most *e*. To avoid keeping all arriving data in the system, for each cell *c* it maintains only K points and discards the rest. It is proven that an exact *k*NN search in the retained points corresponds to a valid *ek*NN answer over the original dataset provided that $k \leq K$. DISC indexes the data points with a *B-tree* that uses a *space filling curve* mechanism to facilitate fast updates and query processing. The authors show how to adjust the index to: (i) use the minimum amount of memory in order to guarantee a given error bound *e*, or (ii) achieve the best possible accuracy, given a fixed amount of memory. DISC can process both snapshot and continuous *ek*NN queries.

Yu et al. [17] propose a method, hereafter referred to as YPK-CNN, for continuous monitoring of exact *k*NN queries. Objects are stored in main memory and indexed with a regular grid of cells with size $\delta \times \delta$. YPK-CNN does not process updates as they arrive, but directly applies them to the grid. Each NN

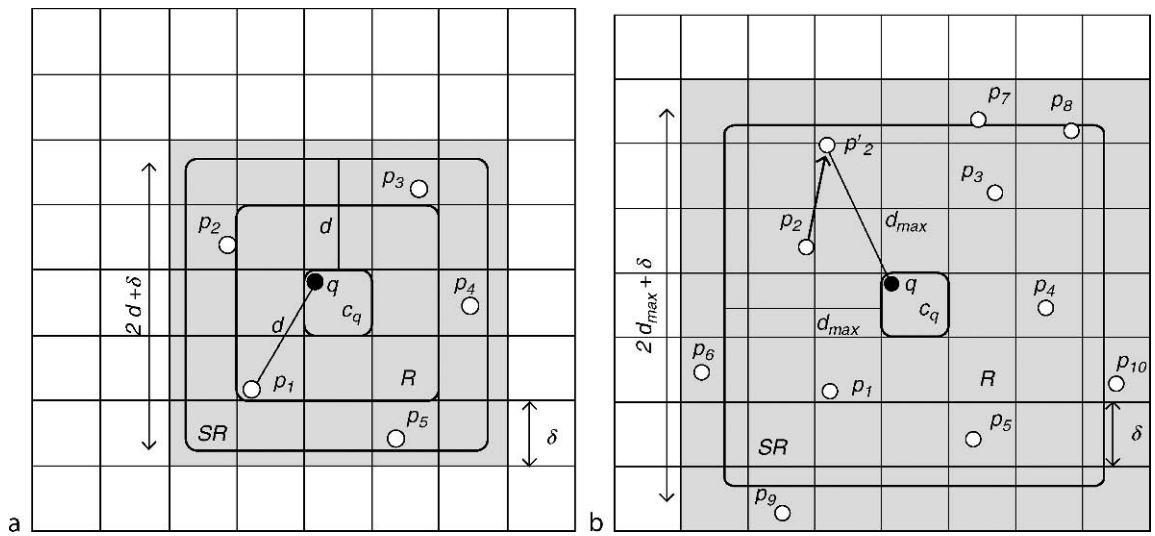
query installed in the system is re-evaluated every T time units. When a query q is evaluated for the first time, a two-step NN search technique retrieves its result. The first step visits the cells in an iteratively enlarged square R around the cell c_q of q until k objects are found. Figure 3a shows an example of a single NN query where the first candidate NN is p_1 with distance d from q; p_1 is not necessarily the actual NN since there may be objects (e.g., p_2) in cells outside R with distance smaller than d. To retrieve such objects, the second step searches in the cells intersecting the square SR centered at c_q with side length $2{\cdot}d + \delta$, and determines the actual kNN set of q therein. In Fig. 3a, YPK-CNN processes p_1 up to p_5 and returns p_2 as the actual NN. The accessed cells appear shaded.



Continuous Monitoring of Spatial Queries. Figure 2. TB example (k = 3).

When re-evaluating an existing query q, YPK-CNN makes use of its previous result in order to restrict the search space. In particular, it computes the maximum distance d_{max} among the current locations of the previous NNs (i.e., d_{max} is the distance of the previous neighbor that currently lies furthest from q). The new SR is a square centered at c_q with side length $2{\cdot}d_{max} + \delta$. In Fig. 3b, assume that the current NN p_2 of q moves to location p'_2. Then, the rectangle defined by $d_{max} = dist(p'_2, q)$ is guaranteed to contain at least one object (i.e., p_2). YPK-CNN collects all objects (p_1 up to p_{10}) in the cells intersecting SR and identifies p_1 as the new NN. Finally, when a query q changes location, it is handled as a new one (i.e., its NN set is computed from scratch).

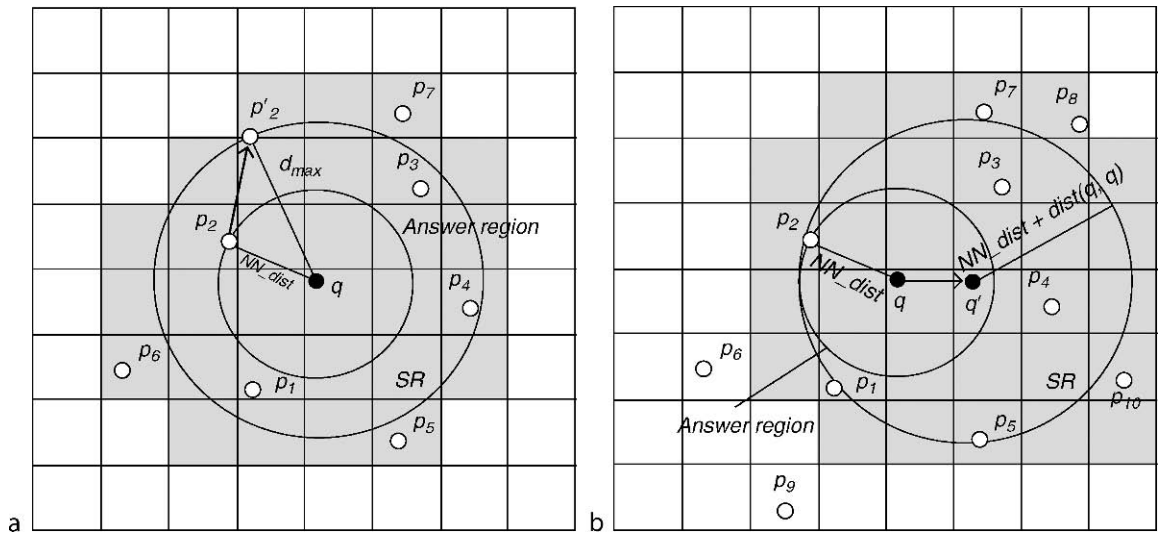
Xiong et al. [16] propose *Shared Execution Algorithm for Continuous NN queries* (SEA-CNN). SEA-CNN focuses exclusively on monitoring the NN changes, without including a module for the first-time evaluation of an arriving query q (i.e., it assumes that the initial result is available). Objects are stored in secondary memory, indexed with a regular grid. The *answer region* of a query q is defined as the circle with center q and radius NN_dist (where NN_dist is the distance of the current kth NN). Book-keeping information is stored in the cells that intersect the answer region of q to indicate this fact. When updates arrive at the system, depending on which cells they



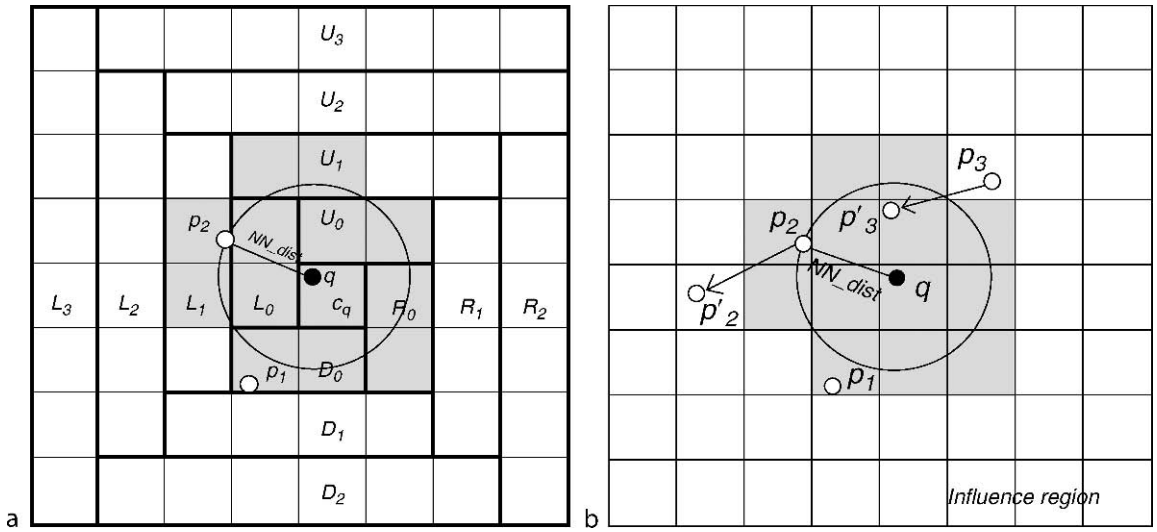
Continuous Monitoring of Spatial Queries. Figure 3. YPK-CNN examples.

affect and whether these cells intersect the answer region of the query, SEA-CNN determines a circular search region *SR* around q, and computes the new kNN set of q therein. To determine the radius r of *SR*, the algorithm distinguishes the following cases: (i) If some of the current NNs move within the answer region or some outer objects enter the answer region, SEA-CNN sets $r = NN\_dist$ and processes all objects falling in the answer region in order to retrieve the new NN set. (ii) If any of the current NNs moves out of the answer region, processing is similar to YPK-CNN; i.e., $r = d_{max}$ (where d_{max} is the distance of the previous NN that currently lies furthest from q), and the NN set is computed among the objects inside *SR*. Assume that in Fig. 4a the current NN p_2 issues an update reporting its new location p'_2. SEA-CNN sets $r = d_{max} = dist(p'_2, q)$, determines the cells intersecting *SR* (these cells appear shaded), collects the corresponding objects (p_1 up to p_7), and retrieves p_1 as the new NN. (iii) Finally, if the query q moves to a new location q', then SEA-CNN sets $r = NN\_dist + dist(q, q')$, and computes the new kNN set of q by processing all the objects that lie in the circle centered at q' with radius r. For instance, in Fig. 4b the algorithm considers the objects falling in the shaded cells (i.e., objects from p_1 up to p_{10} except for p_6 and p_9) in order to retrieve the new NN (p_4).

Mouratidis et al. [9] propose another NN monitoring method, termed *Conceptual Partitioning Monitoring* (CPM). CPM assumes the same system architecture and uses similar indexing and book-keeping structures as YPK-CNN and SEA-CNN. When a query q arrives at the system, the server computes its initial result by organizing the cells into conceptual rectangles based on their proximity to q. Each rectangle *rect* is defined by a *direction* and a *level number*. The direction is U, D, L, or R (for up, down, left and right), and the level number indicates how many rectangles are between *rect* and q. Figure 5a illustrates the conceptual space partitioning around the cell c_q of q. If $mindist(c,q)$ is the minimum possible distance between any object in cell c and q, the NN search considers the cells in ascending $mindist(c, q)$ order. In particular, CPM initializes an empty heap H and inserts (i) the cell of q with key equal to 0, and (ii) the level zero rectangles for each direction *DIR* with key $mindist(DIR_0, q)$. Then, it starts de-heaping entries iteratively. If the de-heaped entry is a cell, it examines the objects inside and updates accordingly the NN set (i.e., the list of the k closest objects found so far). If the de-heaped entry is a rectangle DIR_{lvl}, it inserts into H (i) each cell $c \in DIR_{lvl}$ with key $mindist(c, q)$ and (ii) the next level rectangle DIR_{lvl+1} with key $mindist(DIR_{lvl+1}, q)$. The algorithm terminates when the next entry in H (corresponding either to a cell or a rectangle) has key greater than the distance NN_dist of the kth NN found. It can be easily verified that the server processes only the cells that intersect the circle with center at q and radius equal to NN_dist. This is the minimal set of cells to visit in order to guarantee correctness. In Fig. 5a, the search processes the shaded cells and returns p_2 as the result.



Continuous Monitoring of Spatial Queries. Figure 4. SEA-CNN update handling examples.



Continuous Monitoring of Spatial Queries. Figure 5. CPM examples.

The encountered cells constitute the *influence region* of q, and only updates therein can affect the current result. When updates arrive for these cells, CPM monitors how many objects enter or leave the circle centered at q with radius *NN_dist*. If the *outgoing* objects are more than the *incoming* ones, the result is computed from scratch. Otherwise, the new NN set of q can be inferred by the previous result and the update information, without accessing the grid at all. Consider the example of Fig. 5b, where p_2 and p_3 move to positions p'_2 and p'_3, respectively. Object p_3 moves closer to q than the previous *NN_dist* and, therefore, CPM replaces the outgoing NN p_2 with the incoming p_3. The experimental evaluation in [11] shows that CPM is significantly faster than YPK-CNN and SEA-CNN.

Key Applications

Location-Based Services

The increasing trend of embedding positioning systems (e.g., GPS) in mobile phones and PDAs has given rise to a growing number of location-based services. Many of these services involve monitoring spatial relationships among mobile objects, facilities, landmarks, etc. Examples include location-aware advertising, enhanced 911 services, and mixed-reality games.

Traffic Monitoring

Continuous spatial queries find application in traffic monitoring and control systems, such as on-the-fly driver navigation, efficient congestion detection and avoidance, as well as dynamic traffic light scheduling and toll fee adjustment.

Security Systems

Intrusion detection and other security systems rely on monitoring moving objects (pedestrians, vehicles, etc.) around particular areas of interest or important people.

Future Directions

Future research directions include other types of spatial queries (e.g., *reverse nearest neighbor* monitoring [15,5]), different settings (e.g., NN monitoring over sliding windows [10]), and alternative distance metrics (e.g., NN monitoring in *road networks* [12]). Similar techniques and geometric concepts to the ones presented above also apply to problems of a non-spatial nature, such as continuous skyline [14] and top-k queries [8,18].

Experimental Results

The methods described above are experimentally evaluated and compared with alternative algorithms in the corresponding reference.

Cross-references

- ▶ B+-Tree
- ▶ Nearest Neighbor Query
- ▶ R-tree (and Family)
- ▶ Reverse Nearest Neighbor Query

► Road Networks
► Space-Filling Curves for Query Processing

Recommended Reading

1. Cai Y., Hua K., and Cao G. Processing range-monitoring queries on heterogeneous mobile objects. In Proc. 5th IEEE Int. Conf. on Mobile Data Management, 2004, pp. 27–38.
2. Gedik B. and Liu L. MobiEyes: Distributed processing of continuously moving queries on moving objects in a mobile system. In Advances in Database Technology, Proc. 9th Int. Conf. on Extending Database Technology, 2004, pp. 67–87.
3. Hu H., Xu J., and Lee D. A generic framework for monitoring continuous spatial queries over moving objects. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 479–490.
4. Kalashnikov D., Prabhakar S., and Hambrusch S. Main memory evaluation of monitoring queries over moving objects. Distrib. Parallel Databases, 15(2):117–135, 2004.
5. Kang J., Mokbel M., Shekhar S., Xia T., and Zhang D. Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 806–815.
6. Koudas N., Ooi B., Tan K., and Zhang R. Approximate NN queries on streams with guaranteed error/performance bounds. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 804–815.
7. Mokbel M., Xiong X., and Aref W. SINA: Scalable incremental processing of continuous queries in spatio-temporal databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 623–634.
8. Mouratidis K., Bakiras S., Papadias D. Continuous monitoring of top-*k* queries over sliding windows. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 635–646.
9. Mouratidis K., Hadjieleftheriou M., and Papadias D. Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 634–645.
10. Mouratidis K. and Papadias D. Continuous nearest neighbor queries over sliding windows. IEEE Trans. Knowledge and Data Eng., 19(6):789–803, 2007.
11. Mouratidis K., Papadias D., Bakiras S., and Tao Y. A threshold-based algorithm for continuous monitoring of *k* nearest neighbors. IEEE Trans. Knowledge and Data Eng., 17(11):1451–1464, 2005.
12. Mouratidis K., Yiu M., Papadias D., and Mamoulis N. Continuous nearest neighbor monitoring in road networks. In Proc. 32nd Int. Conf. on Very Large Data Bases, 2006, pp. 43–54.
13. Prabhakar S., Xia Y., Kalashnikov D., Aref W., and Hambrusch S. Query indexing and velocity constrained indexing: scalable techniques for continuous queries on moving objects. IEEE Trans. Comput., 51(10):1124–1140, 2002.
14. Tao Y. and Papadias D. Maintaining sliding window skylines on data Streams. IEEE Trans. Knowledge and Data Eng., 18(3): 377–391, 2006.
15. Xia T. and Zhang D. Continuous reverse nearest neighbor monitoring. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
16. Xiong X., Mokbel M., and Aref W. SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In Proc. 21st Int. Conf. on Data Engineering, 2005, pp. 643–654.
17. Yu X., Pu K., and Koudas N. Monitoring k-nearest neighbor queries over moving objects. In Proc. 21st Int. Conf. on Data Engineering, 2005, pp. 631–642.
18. Zhang D., Du Y., and Hu L. On monitoring the top-k unsafe places, In Proc. 24th Int. Conf. on Data Engineering, 2008, pp. 337–345.

Continuous Multimedia Data Retrieval

Jeffrey Xu Yu
Chinese University of Hong Kong, Hong Kong, China

Definition

Continuous multimedia is widely used in many applications nowadays. Continuous multimedia objects, such as audio and video streams, being stored on disks with different requirements of bandwidths, are required to be retrieved continuously without interruption. The response time is an important measurement in supporting continuous multimedia streams. Several strategies are proposed in order to satisfy the requirements of all users in a multi-user environment where multiple users are trying to retrieve different continuous multimedia streams together.

Historical Background

Several multimedia data retrieval techniques are proposed to support the real-time display of continuous multimedia objects. There are three categories [6]. The first category is to sacrifice the quality of the data in order to guarantee the required bandwidth of multimedia objects. The existing techniques either use lossy compression techniques (such as predictive [15], frequency oriented [11], and importance oriented [10]), or use a low resolution device. The second category is to use the placement techniques to satisfy the continuous requirement by arranging the data to appropriate disk locations. In other words, it is to organize multimedia data across the surface of a disk drive to maximize its bandwidth when it is retrieved [4,5,16,22,20]. The third category is to increase the bandwidth of storage device by using parallelism. The basic idea is to employ the aggregate bandwidth of several disk drives by putting an object across multiple disks, for

example, a Redundant Arrays of Inexpensive Disk (RAID) [17]. The existing works [9,19] focus on this direction.

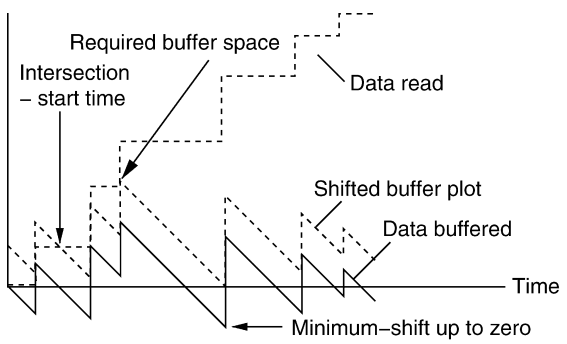
Foundations

This section focuses on the second and third categories, and discusses multimedia data retrieval regarding single/multiple stream(s) and single/multiple disk(s).

Retrieval of a Single Stream on a Single Disk

For the retrieval of a single multimedia stream on a single disk, the stream data is read into a first-in-first-out queue (FIFO) continuously first, and then is sent to the display devices, possibly via a network, at the appropriate rate. In order to satisfy the real-time requirements – to display multimedia data continuously on a display, it is required to keep the FIFO non empty. In other words, there is some multimedia data to be displayed in the FIFO in the duration of the playback. As pointed in [6], pre-fetching all the data into the FIFO before playback is not a feasible solution because the size of the stream can be very large.

Suppose that a read request of a large multimedia data is issued. The starting time and the minimum buffer space, to display the retrieved multimedia data continuously, are determined as follows, under the following conditions: (i) the timing of data retrieval is known in advance, (ii) both the transfer rate and consumption rate are constant, and (iii) the transfer rate of the storage device is at least as great as the consumption rate. Consider Fig. 1. First, the amount of data, that needs to be consumed by a display, is illustrated as the dotted line marked `data read`. The vertical line segments show the amount of data that needs to be consumed in order to continuously display, and the horizontal line shows the time periods such amount of data is consumed on a display. Second, the solid zigzag line, marked `data buffered`, shows the data to be accessed in the data buffers. The vertical line segments show the data to be read into the buffers followed by the line segments that show data is consumed in the buffer during a certain time interval. Here, in the solid zigzag line, there is a minimum point (marked `minimum-shift up to zero`), which is a possible negative value and is denoted as $z(< 0)$. Third, the dotted zigzag line (marked `shifted buffer plot`) is the line by shifting the entire solid zigzag line up by the amount of $|z|$ where $z < 0$. Finally, the starting time to display is determined as the point at which the shifted-up dotted zigzag line (`shifted buffer plot`) and the dotted line (`data read`) intersect, which is indicated as `intersection - start time` in Fig. 1. Also, the minimum buffer size is the maximum value of in the line of `shifted buffer plot`, which is indicated as `required buffer space` in Fig. 1. Details are discussed in [7].

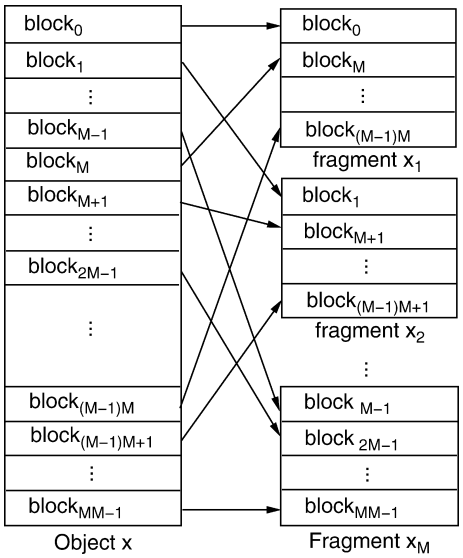


Continuous Multimedia Data Retrieval. Figure 1.
Finding minimum buffer space and start time (Fig. 2 in [6]).

Retrieval of a Single Stream on Multiple Disks

The multimedia data retrieval using multiple disks is a technique to retrieve a data stream continuously at the required bandwidth. The main idea behind is to de-cluster the data stream into several fragments [14,2], and distribute these fragments across multiple processors (and disks). By combining the I/O bandwidths of several disks, a system can provide the required retrieval rate to display a continuous multimedia stream in real-time. Assume that the required retrieval rate is B and the bandwidth of each disk is B_D. The degree of de-clustering can be calculated as $M = \lceil \frac{B}{B_D} \rceil$, which implies the number of disks that is needed to satisfy the required retrieval rate.

When the degree of de-clustering is determined, the fragments can be formed using a round-robin partitioning strategy as illustrated in Fig. 2, where an object x is partitioned into M fragments stored on M disks. The round-robin partitioning is conducted as follows. First, the object x is divided in N blocks (disk pages) depending on the disk-page size allowed on disks. In Fig. 2, the number of blocks is $N = M \cdot M$. The first block_0 is assigned to first fragment indicated as x_1 in Fig. 2, and the second block_1 is assigned to the second fragment indicated as x_2. The first M blocks from block_0 to block_{M-1} are assigned to the M fragments one by one. In next run, the next set of blocks,



Continuous Multimedia Data Retrieval. Figure 2. Round-robin partitioning of object *x* (Fig. 5 in [9]).

from block$_M$ to block$_{2M-1}$ will be assigned to the M fragments in the similar fashion. The process repeats until all data blocks are assigned to the fragments in a round-robin fashion.

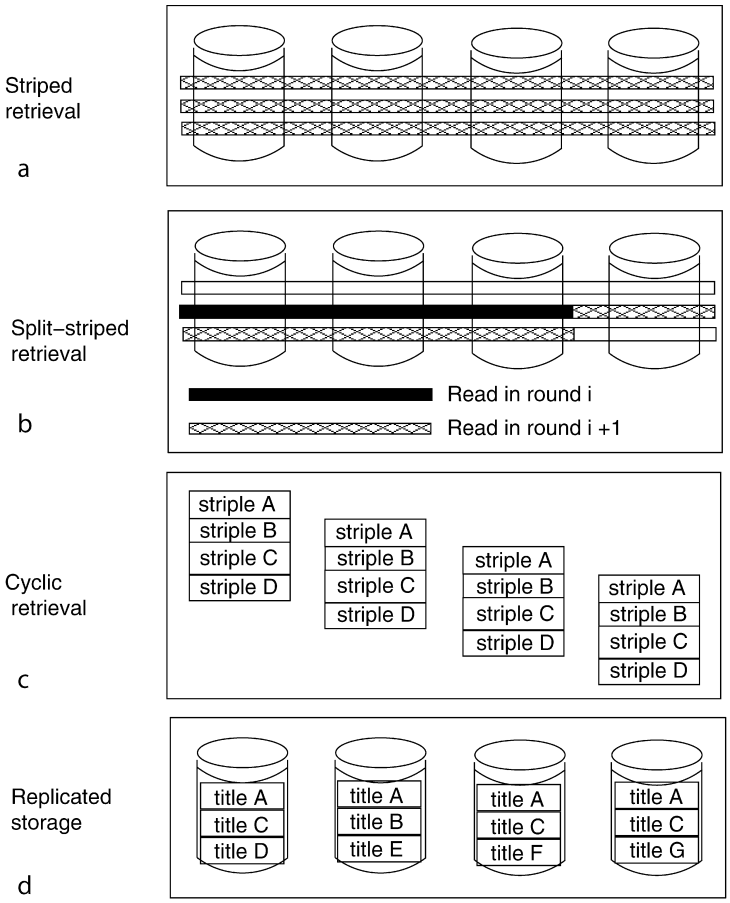
Retrieval of Multiple Streams on a Single Disk

In a multi-user environment, several users may retrieve data streams simultaneously. Therefore, there are multiple data streams requested on a single disk. The data streams are retrieved in rounds, and each stream is allowed a disk access or a fixed number of disk accesses at one time. All data retrieval requests need to be served in turn. Existing solutions include SCAN, round-robin, EDF, and Sorting-Set algorithms. The round-robin algorithm retrieves data for each data retrieval request, in turn, in a predetermined order. The SCAN algorithm moves the disk head back and forth, and retrieves the requested blocks when the disk head passes over the requested blocks [18]. The EDF (earliest-deadline-first) algorithm serves the request with the earliest deadline first, where a deadline is given to a data stream [13]. The sorting-set algorithm is designed to exploit the trade-off between the number of rounds between successive reads for a data stream and the length of the round [8,21], by assigning each data stream to a sorting set. Fixed time slots are allocated to a sorting set in a round during which its requests are possibly processed.

Retrieval of Multiple Streams on Multiple Disks

In order to support multiple stream retrieval, making use of parallel disks is an effective method, where a data stream is striped across the parallel disks. There are several approaches to retrieve data streams when they are stored on parallel disks (Fig. 3). It is important to note that the main issue here is to increase the number of data streams to be retrieved simultaneously. It is not to speed up retrieval for an individual data stream using multiple disks. Consider the striped retrieval as shown in Fig. 3a, where a data stream is striped across m parallel disks. Suppose that each disk has r_c bandwidth, m parallel disks can be together used to increase the bandwidth up to $m \cdot r_c$. However, the issue is the system capacity in terms of the number of data streams it can serve, for example, from n data streams to $m \cdot n$ data streams using m parallel disks. Suppose that each data stream will be served in turn. When it increases the number of data streams from n to $m \cdot n$, in the striped retrieval, the round length (or in other words consecutive reads for a single data stream) increases proportionally from n to $m \cdot n$. It implies that, in order to satisfy the required retrieval rate, it needs to use a larger buffer, which also implies a larger startup delay. An improvement over striped retrieval is to use split-stripe retrieval which allows partial stripes to be used (Fig. 3b), in order to reduce the buffer size required in the striped retrieval. But, it has its limit to significantly reduce startup delay and buffer space.

Observe the data transfer patterns in the striped retrieval and the split-striped retrieval, which show busty patterns for data to be read into the buffer. For instance, consider Fig. 3a, an entire strip for a single data stream will be read in and be consumed in a period which is related to the round length. It requests larger buffer sizes, because it needs to keep the data to be displayed continuously until the next read, in particular when the number of streams increases from n to $m \cdot n$. Instead, an approach is proposed to read small portion of data frequently, in order to reduce the buffer space required. The approach is called cyclic retrieval. As shown in Fig. 3c, the cyclic retrieval tries to read multiple streams rather than one stream at one time. Rather than retrieving an entire stripe at once, the cyclic retrieval retrieves each striping unit of a stripe consecutively [1,3]. Using this approach, the buffer space is significantly reduced. But the reduction comes with cost. The buffer space reduction is achieved at the expense of cuing (*a stream is said to*



Continuous Multimedia Data Retrieval. Figure 3. Retrieval of multiple streams on multiple disks [6].

be cued if it is paused and playback may be initiated instantaneously) and clock skew tolerance.

As an alternative to striped (split-striped) or cyclic retrieval, it can deal with each disk independently rather than treating them as parallel disks. Here, each disk stores a number of titles (data streams). When there is a multimedia data retrieval request, a disk that contains the data stream will respond. The data streams that are frequently requested may be kept in multiple disks using replication. The number of replications can be determined based on the retrieval frequency of data streams [12], as shown in Fig. 3d. Based on the replicated retrieval, both the startup delay time and buffer space can be reduced significantly. It is shown that it is easy to scale when the number of data streams increase at the expense of more disk space required. [9] discusses data replication techniques.

A comparison among striped-retrieval, cyclic retrieval, and replicated retrieval in supporting n streams is shown in Table 1.

Continuous Multimedia Data Retrieval. Table 1. A comparison of multi-disk retrieval strategies supporting n streams (Table 1 in [6])

	Striped	Cyclic	Replicated
Instant restart	yes	no	yes
Clock skew tolerance	yes	no	yes
Easy scaling	no	no	yes
Capacity	per-system	per-system	per-title
Startup delay	$O(n)$	$O(n)$	$O(1)$
Buffer space	$O(n^2)$	$O(n)$	$O(n)$

Key Applications

Continuous multimedia data retrieval is used in many real-time continuous multimedia streams such as audio and video through the network. Especially in a multi-user environment, the continuous multimedia data retrieval techniques are used to support simultaneous display of several multimedia objects in real-time.

Cross-references

- ▶ Buffer Management
- ▶ Buffer Manager
- ▶ Multimedia Data Buffering
- ▶ Multimedia Data Storage
- ▶ Multimedia Resource Scheduling
- ▶ Scheduling Strategies for Data Stream Processing
- ▶ Storage Management
- ▶ Storage Manager

Recommended Reading

1. Berson S., Ghandeharizadeh S., Muntz R., and Ju X. Staggered striping in multimedia information systems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1994, pp. 79–90.
2. Carey M.J. and Livny M. Parallelism and concurrency control performance in distributed database machines. ACM SIGMOD Rec., 18(2):122–133, 1989.
3. Chen M.S., Kandlur D.D., and Yu P.S. Storage and retrieval methods to support fully interactive playout in a disk-array-based video server. Multimedia Syst., 3(3):126–135, 1995.
4. Christodoulakis S. and Ford D.A. Performance analysis and fundamental performance tradeoffs for CLV optical disks. ACM SIGMOD Rec., 17(3):286–294, 1988.
5. Ford D.A. and Christodoulakis S. Optimizing random retrievals from CLV format optical disks. In Proc. 17th Int. Conf. on Very Large Data Bases, 1991, pp. 413–422.
6. Gemmell D.J. Multimedia information storage and management, chap. 1. Disk Scheduling for Continuous Media. Kluwer, Norwell, MA, USA, 1996.
7. Gemmell J. and Christodoulakis S. Principles of delay-sensitive multimedia data storage retrieval. ACM Trans. Inf. Syst., 10(1):51–90, 1992.
8. Gemmell D.J. and Han J. Multimedia network file servers: multichannel delay-sensitive data retrieval. Multimedia Syst., 1(6):240–252, 1994.
9. Ghandeharizadeh S. and Ramos L. Continuous retrieval of multimedia data using parallelism. IEEE Trans. on Knowl. and Data Eng., 5(4):658–669, 1993.
10. Green J.L. The evolution of DVI system software. Commun. ACM, 35(1):52–67, 1992.
11. Lippman A. and Butera W. Coding image sequences for interactive retrieval. Commun. ACM, 32(7):852–860, 1989.
12. Little T.D.C. and Venkatesh D. Popularity-based assignment of movies to storage devices in a video-on-demand system. Multimedia Syst., 2(6):280–287, 1995.
13. Liu C.L. and Layland J.W. Scheduling algorithms for multiprogramming in a hard real-time environment. In Tutorial: Hard Real-Time Systems. IEEE Computer Society, Los Alamitos, CA, USA, 1989, pp. 174–189.
14. Livny M., Khoshafian S., and Boral H. Multi-disk management algorithms. SIGMETRICS Perform. Eval. Rev., 15(1):69–77, 1987.
15. Luther A.C. Digital video in the PC environment, (2nd edn.). McGraw-Hill, New York, NY, USA, 1991.
16. McKusick M.K., Joy W.N., Leffler S.J., and Fabry R.S. A fast file system for UNIX. Comput. Syst., 2(3):181–197, 1984.
17. Patterson D.A., Gibson G.A., and Katz R.H. A case for redundant arrays of inexpensive disks (RAID). In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1988, pp. 109–116.
18. Teorey T.J. and Pinkerton T.B. A comparative analysis of disk scheduling policies. In Proc. 3rd ACM Symp. on Operating System Principles, 1971, pp. 114.
19. Tsai W.J. and Lee S.Y. Storage design and retrieval of continuous multimedia data using multi-disks. In Proc. 1994 Int. Conf. on Parallel and Distributed Systems, 1994, pp. 148–153.
20. Wong C.K. Minimizing expected head movement in one-dimensional and two-dimensional mass storage systems. ACM Comput. Surv., 12(2):167–178, 1980.
21. Yu P.S., Chen M.S., and Kandlur D.D. Grouped sweeping scheduling for DASD-based multimedia storage management. Multimedia Syst., 1(3):99–109, 1993.
22. Yue P.C. and Wong C.K. On the optimality of the probability ranking scheme in storage applications. J. ACM, 20(4):624–633, 1973.

Continuous Queries in Sensor Networks

Yong Yao, Johannes Gehrke
Cornell University, Ithaca, NY, USA

Synonyms

Long running queries

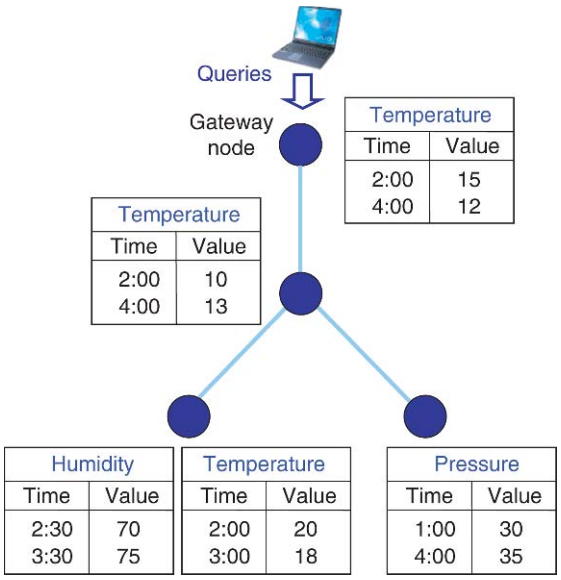
Definition

A powerful programming paradigm for data acquisition and dissemination in sensor networks is a declarative query interface. With a declarative query interface, the sensor network is programmed for long term monitoring and event detection applications through continuous queries, which specify what data to retrieve at what time or under what conditions. Unlike snapshot queries which execute only once, continuous queries are evaluated periodically until the queries expire. Continuous queries are expressed in a high-level language, and are compiled and installed on target sensor nodes, controlling when, where, and what data is sampled, possibly filtering out unqualified data through local predicates. Continuous queries can have a variety of optimization goals, from improving result quality and response time to reducing energy consumption and prolonging network lifetime.

Historical Background

In recent years sensor networks have been deployed successfully for a wide range of applications from environmental sensing to process monitoring. A database approach to programming sensor networks has gained much importance: Clients program the network through queries without knowing how the results are generated, processed, and returned to the client. Sophisticated catalog management, query optimization, and query processing techniques abstract the client from the physical details of contacting the relevant sensors, processing the sensor data, and sending the results to the client. The concept of a sensor network as a database was first introduced in [3]. A number of research projects, including TinyDB [9] and Cougar [14] have implemented continuous queries as part of their database languages for sensor networks. In these systems time is divided into epochs of equal size, and continuous queries are evaluated once per epoch during their lifetime. Figure 1 shows this database view of sensor networks.

Two properties are significant to continuous query processing in sensor networks: energy conservation and fault-tolerance in case of failures of sensors, both topics that are not of importance in traditional database systems or data stream systems. Advanced query processing techniques have been proposed to enable energy-efficient query processing in the presence of frequent node and communication failures. For example, a lot of research has been dedicated to in-network query processing [6,9,14] to reduce the amount of data to be transmitted inside the network. Another approach is to permit approximate query processing [4,5], which produces approximate query answers within a predefined accuracy range, but consumes much less energy. Sensor data is correlated in time and space. Data compression in sensor networks and probabilistic data models [1,7,8] exploit data correlation and remove redundant data from intermediate results.



Continuous Queries in Sensor Networks. Figure 1. Database view of sensor networks.

Next generation sensor network may consist of media-rich and mobile sensor nodes, which result in new challenges arise for continuous query processing such as mobility and high data rates. ICEDB [15] describes a new framework for continuous query processing in sensor networks with intermittent network connectivity and large amount of data to transfer.

Foundations

Continuous queries are a natural approach for data fusion in sensor networks for long running applications as they provide a high-level interface that abstracts the user from the physical details of the network. The design and implementation of continuous queries needs to satisfy several requirements. First, it has to preserve the scarce resources such as energy and bandwidth in battery-powered sensor networks. Thus the simple approach of transmitting all relevant data back to a central node for query evaluation is prohibitive for sensor networks of non-trivial size, as communication using the wireless medium consumes a lot of energy. Since sensor nodes have the ability to perform local computation, communication can be traded for computation by moving computation from the clients into the sensor network, aggregating partial results or eliminating irrelevant data. Second, sensor network applications usually have different QoS requirements, from accuracy, energy consumption to delay. Therefore the continuous query model needs to be flexible enough to adopt various processing techniques in different scenarios.

Sensor Data Model

In the view of a sensor network as a database, each sensor node is modeled as a separate data source that generates records with several fields such as the sensor type, location of the sensor node, a time stamp, and the value of the reading. Records of the same sensor type from different nodes have the same schema, and

these records collectively form a distributed table of sensor readings. Thus the sensor network can be considered as a large distributed database system consisting of several tables of different types of sensors.

Sensor readings are samples of physical signals whose values change continuously over time. For example, in environmental monitoring applications, sensor readings are generated every few seconds (or even faster). For some sensor types (such as PIR sensors that sense the presence of objects) their readings might change rapidly and thus may be outdated rather quickly, whereas for other sensors, their value changes only slowly over time as for temperature sensors that usually have a small derivative. Continuous queries recompute query results periodically and keep query results up-to-date. For applications that require only approximate results, the system can cache previous results and lower the query update rate to save energy.

Instead of querying raw sensor data, most applications are more interested in composite data which captures high-level events monitored by sensor networks. Such composite data is produced by complex signal processing algorithms given raw sensor measurements as inputs. Composite data usually has a compact structure and is easier to query.

Continuous Query Models

In TinyDB and Cougar, continuous queries are represented as a variant of SQL with a few extensions. A simple query template in Cougar is shown in the figure below. (TinyDB uses a very similar query structure.)

```
SELECT    {attribute, aggregate}
FROM      {Sensordata S}
WHERE     {predicate}
GROUP BY  {attribute}
HAVING    {predicate}
DURATION  time interval
EVERY     time span e
```

The template can be extended to support nested queries, where the basic query block shown below can appear within the WHERE or HAVING clause of another query block. The query template has an obvious semantics: the SELECT clause specifies attributes and aggregates from sensor records, the FROM clause specifies the distributed relation describing the sensor type, the WHERE clause filters sensor records by a predicate, the GROUP BY clause classifies sensor records into different partitions according to some attributes, and the HAVING clause eliminates groups by a predicate. Join queries between external tables and sensor readings are constructed by including the external tables and sensor readings in the FROM clause and join predicates in the WHERE clause.

Two new clauses introduced for continuous queries are DURATION and EVERY; The DURATION clause specifies the lifetime of the continuous query, and the EVERY or clause determines the rate of query answers. TinyDB has two related clauses: LIFETIME and SAMPLE INTERVAL, specifying the lifetime of the query and the sample interval, respectively. The LIFETIME clause will be discussed in more detail a few paragraphs later.

In event detection applications, sensor data is collected only when particular events happen. The above query template can be extended with a condition clause as a prerequisite to determine when to start or stop the main query. Event-based queries have the following structure in TinyDB:

```
ON EVENT    {event(arguments)}:
  {query body}
```

Another extension to the basic query template is lifetime-based queries, which have no explicit EVERY or SAMPLE INTERVAL clause; only the query lifetime is specified through a LIFETIME clause [9]. The system automatically adjusts the sensor sampling rate to the highest rate possible with the guarantee that the sensor network can process the query for the specified lifetime. Lifetime-based queries are more intuitive in some mission critical applications where user queries have to run for a given period of time, but it is hard to predict the optimal sampling rate in advance. Since the sampling rate is adjusted continuously according to the available power and the energy consumption rate in the sensor network, lifetime-based queries are more adaptive to unpredictable changes in sensor networks deployed in a harsh environment.

Common Types of Continuous Queries in Sensor Networks Select-All Queries

Recent sensor network deployments indicate that a very common type of continuous queries is a select-all query, which extracts all relevant data from the sensor network and stores the data in a central place for further processing and analysis. Although select-all queries are simple to express, efficient processing of select-all queries is a big challenge. Without optimization, the size of the transmitted data explodes

quickly, and thus the power of the network would be drained in a short time, especially for those nodes acting as bridge to the outside world; this significantly decreases the lifetime of the sensor network.

One possible approach is to apply model-based data compression at intermediate sensor nodes [7]. For many types of signals, e.g., temperature and light, sensor readings are highly correlated in both time and space. Data compression in sensor networks can significantly reduce the communication overhead and increase the network lifetime. Data compression can also improve the signal quality by removing unwanted noise from the original signal. One possible form of compression is to construct and maintain a model of the sensor data in the network; the model is stored both on the server and on sensor nodes in the network. The model on the server can be used to predicate future values within a pre-defined accuracy range. Data communication happens to synchronize the data model on the server with real sensor measurements [7].

Aggregate Queries

Aggregate queries return aggregate values for each group of sensor nodes specified by the `GROUP BY` clause. Below is is an example query that computes the average concentration in a region every 10 seconds for the next hour:

```
SELECT    AVG(R.concentration)
FROM      ChemicalSensor R
WHERE     R.loc IN region
HAVING    AVG(R.concentration) > T
DURATION  (now,now+3600)
EVERY     10
```

Data aggregation in sensor networks is well-studied because it scales to sensor networks with even thousands of nodes. Query processing proceeds along a spanning tree of sensor nodes towards a gateway node. During query processing, partial aggregate results are transmitted from a node to its parent in the spanning tree. Once an intermediate node in the tree has received all data from nodes below it in a round, the node compute a partial aggregate of all received data and sends that output to the next node. This solution works for aggregate operators that are incrementally computable, such as avg, max, and moments of the data. The only caveat is that this in-network computation requires synchronization between sensor nodes along the communication path, since a node has to "wait" to receive results to be aggregated. In networks with high loss rates, broken links are hard to differentiate from long delays due to high loss rates, making synchronization a non-trivial problem [13].

Join Queries

In a wide range of event detection applications, sensor readings are compared to a large number of time and location varying predicates to determine whether a user-interesting event is detected [1]. The values of these predicates are stored in a table. Continuous queries with a *join* operator between sensor readings and the predicate table are suitable for such applications. Similar join queries can be used to detect defective sensor nodes whose readings are inaccurate by checking their readings against readings from neighboring sensors (again assuming spatial correlation between sensor readings). Suitable placement of the join operator in a sensor network has also been examined [2].

Key Applications

Habitat Monitoring

In the Great Duck Island experiment, a network of sensors was deployed to monitor the microclimate in and around nesting burrows used by birds, with the goal of developing a habitat monitoring kit that would enable researchers worldwide to engage in non-intrusive and non-disruptive monitoring of sensitive wildlife and habitats [10]. In a more recent experiment, a sensor network was deployed to densely record the complex spatial variations and the temporal dynamics of the microclimate around a 70-meter tall redwood tree [12].

The Intelligent Building

Sensor networks can be deployed in intelligent buildings for the collection and analysis of structural responses to ambient or forced excitation of the building's structure, for control of light and temperature to conserve energy, and for monitoring of the flow of people in critical areas. Continuous queries are used both for data collection and for event-based monitoring of sensitive areas and to enforce security policies.

Industrial Process Control

Idustrial manufacturing processes often have strict requirements on temperature, humidity, and other environmental parameters. Sensor networks can be

deployed to monitor the production environment without expensive wires to be installed. Continuous join queries compare the state of the environment to a range of values specified in advance and send an alert when an exception is detected [1].

Cross-references

- ▶ Approximate Query Processing
- ▶ Data Acquisition and Dissemination in Sensor Networks
- ▶ Data Aggregation in Sensor networks
- ▶ Data Compression in Sensor Networks
- ▶ Data Fusion in Sensor Networks
- ▶ Database Languages for Sensor Networks
- ▶ Distributed Database Systems
- ▶ In-Network Query Processing
- ▶ Sensor Networks

Recommended Reading

1. Abadi D., Madden S., and Lindner W. REED: robust, efficient filtering and event detection in sensor networks. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 768–780.
2. Bonfils B. and Bonnet P. Adaptive and decentralized operator placement for in-network query processing. In Proc. 2nd Int. Workshop Int. Proc. in Sensor Networks, 2003, pp. 47–62.
3. Bonnet P., Gehrke J., and Seshadri P. Towards sensor database systems. In Proc. 2nd Int. Conf. on Mobile Data Management, 2001, pp. 3–14.
4. Chu D., Deshpande A., Hellerstein J., and Hong W. Approximate data collection in sensor networks using probabilistic models. In Proc. 22nd Int. Conf. on Data Engineering, 2006.
5. Considine J., Li F., Kollios G., and Byers J. Approximate aggregation techniques for sensor databases. In Proc. 20th Int. Conf. on Data Engineering, 2004, pp. 449–460.
6. Deligiannakis A., Kotidis Y., and Roussopoulos N. Hierarchical in-network data aggregation with quality guarantees. In Advances in Database Technology, Proc. 9th Int. Conf. on Extending Database Technology, 2004, pp. 658–675.
7. Deshpande A., Guestrin C., Madden S., Hellerstein J., and Hong W. Model-driven data acquisition in sensor networks. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 588–599.
8. Kanagal B. and Deshpande A. Online filtering, smoothing and probabilistic modeling of streaming data. In Proc. 24th Int. Conf. on Data Engineering, 2008, pp. 1160–1169.
9. Madden S., Franklin M., Hellerstein J., and Hong W. The design of an acquisitional query processor for sensor networks. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 491–502.
10. Mainwaring A., Polastre J., Szewczyk R., Culler D., and Anderson J. Wireless sensor networks for habitat monitoring. In Proc. 1st ACM Int. Workshop on Wireless Sensor Networks and Applications, 2002, pp. 88–97.
11. Stoianov I., Nachman L., Madden S., and Tokmouline T. PIPENET: a wireless sensor network for pipeline monitoring. In Proc. 6th Int. Symp. Inf. Proc. in Sensor Networks, 2007, pp. 264–273.
12. Tolle G., Polastre J., Szewczyk R., Culler D., Turner N., Tu K., Burgess S., Dawson T., Buonadonna P., Gay D., and Hong W. A macroscope in the redwoods. In Proc. 3rd Int. Conf. on Embedded Networked Sensor Systems, 2005.
13. Trigoni N., Yao Y., Demers A.J., Gehrke J., and Rajaraman R. Wave scheduling and routing in sensor networks. ACM Trans. Sensor Netw., 3(1):2, 2007.
14. Yao Y. and Gehrke J. Query processing in sensor networks. In Proc. 1st Biennial Conf. on Innovative Data Systems Research, 2003.
15. Zhang Y., Hull B., Balakrishnan H., and Madden S. ICEDB: intermittently connected continuous query processing. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 166–175.

Continuous Query

Shivnath Babu
Duke University, Durham, NC, USA

Synonyms

Standing query

Definition

A continuous query Q is a query that is issued once over a database D, and then logically runs continuously over the data in D until Q is terminated. Q lets users get new results from D without having to issue the same query repeatedly. Continuous queries are best understood in contrast to traditional SQL queries over D that run once to completion over the current data in D.

Key Points

Traditional database systems expect all data to be managed within some form of persistent data sets. For many recent applications, where the data is changing constantly (often exclusively through insertions of new elements), the concept of a continuous data stream is more appropriate than a data set. Several applications generate data streams naturally as opposed to data sets, e.g., financial tickers, performance measurements in network monitoring, and call detail records in telecommunications. Continuous queries are a natural interface for monitoring data streams. In network monitoring, e.g., continuous queries may be used to monitor whether all routers and links are functioning efficiently.

The Tapestry system [3] for filtering streams of email and bulletin-board messages was the first to make continuous queries a core component of a database system. Continuous queries in Tapestry were expressed using a subset of SQL. Barbara [2] later formalized continuous queries for a wide spectrum of environments. With the recent emergence of general-purpose systems for processing data streams, continuous queries have become the main interface that users and applications use to query data streams [1].

Materialized views and triggers in traditional database systems can be viewed as continuous queries. A materialized view V is a query that needs to be reevaluated or incrementally updated whenever the base data over which V is defined changes. Triggers implement event-condition-action rules that enable database systems to take appropriate actions when certain events occur.

Cross-references

- ▶ Database Trigger
- ▶ ECA-Rule
- ▶ Materialized Views
- ▶ Processing

Recommended Reading

1. Babu S. and Widom J. Continuous queries over data streams. ACM SIGMOD Rec., 30(3):109–120, 2001.
2. Barbara D. The characterization of continuous queries. Int. J. Coop. Inform. Syst., 8(4):295–323, 1999.
3. Terry D., Goldberg D., Nichols D., and Oki B. Continuous queries over append-only databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1992, pp. 321–330.

Continuous Query Languages

- ▶ Stream-oriented Query Languages and Operators

Continuous Query Processing Applications

- ▶ Streaming Applications

Continuous Query Scheduling

- ▶ Scheduling Strategies for Data Stream Processing

ConTract

Andreas Reuter[1,2]
[1]EML Research aGmbH Villa Bosch, Heidelberg, Germany
[2]Technical University Kaiserslautern, Kaiserslautern, Germany

Definition

A ConTract is an extended transaction model that employs transactional mechanisms in order to provide a run-time environment for the reliable execution of long-lived, workflow-like computations. The focus is on durable execution and on correctness guarantees with respect to the effects of such computations on shared data.

Key Points

The notion of a ConTract (concatenated transactions) combines the principles of workflow programing with the ideas related to long-lived transactions. The ConTract model is based on a two-tier programing approach. At the top level, each ConTract is a script describing a (long-lived) computation. The script describes the order of execution of so-called steps. A step is a predefined unit of execution (e.g., a service invocation) with no visible internal structure. A step can access shared data in a database, send messages, etc.

A ConTract, once it is started, will never be lost by the system, no matter which technical problems (short of a real disaster) will occur during execution. If completion is not possible, all computations performed by a ConTract will be revoked, so in a sense ConTracts have transactional behaviour in that they will either be run to completion, or the impossibility of completion will be reflected in the invocation of appropriate recovery measures.

The ConTract model draws on the idea of Sagas, where the notion of compensation is employed as a means for revoking the results of computations beyond the boundaries of ACID transactions. In a ConTract, by default each step is an ACID transaction. But it is possible to group multiple steps (not just linear sequences) into a transaction. Compensation steps must be supplied by the application explicitly.

The ideas of the ConTract model have selectively been implemented in some academic prototypes, but a full implementation has never been attempted. It has

influenced many later versions of "long-lived transaction" schemes, and a number of its aspects can be found in commercial systems such as BizTalk.

Cross-references

▶ Extended Transaction Models
▶ Persistent Execution
▶ Sagas
▶ Workflow

Recommended Reading

1. Reuter A. and Waechter H. The ConTract model. In Readings in Database in Database Systems, (2nd edn.), M. Stonebraker, J. Hellerstein, (eds.). Morgan Kaufmann, Los Altos, CA, 1992, pp. 219–263.

ConTracts

▶ Flex Transactions

Contrast Pattern

▶ Emerging Patterns

Contrast Pattern Based Classification

▶ Emerging Pattern Based Classification

Control Data

Nathaniel Palmer
Workflow Management Coalition, Hingham, MA, USA

Synonyms

Workflow control data; Workflow engine state data; Workflow enactment service state data

Definition

Data that is managed by the Workflow Management System and/or a Workflow Engine. Such data is internal to the workflow management system and is not normally accessible to applications.

Key Points

Workflow control data represents the dynamic state of the workflow system and its process instances.

Workflow control data examples include:

- State information about each workflow instance.
- State information about each activity instance (active or inactive).
- Information on recovery and restart points within each process, etc.

The workflow control data may be written to persistent storage periodically to facilitate restart and recovery of the system after failure. It may also be used to derive audit data.

Cross-references

▶ Activity
▶ Process Life Cycle
▶ Workflow Management and Workflow Management System
▶ Workflow Model

Control Flow Diagrams

▶ Activity Diagrams

Controlled Vocabularies

▶ Lightweight Ontologies

Controlling Overlap

▶ Processing Overlaps

Convertible Constraints

Carson Kai-Sang Leung
University of Manitoba, Winnipeg, MB, Canada

Definition

A constraint C is *convertible* if and only if C is convertible anti-monotone or convertible monotone.

A constraint C is *convertible anti-monotone* provided there is an order $\mathcal{R}$ on items such that when an ordered itemset S satisfies constraint C, so does any prefix of S. A constraint C is *convertible monotone* provided there is an order $\mathcal{R}'$ on items such that when an ordered itemset S' violates constraint C, so does any prefix of S'.

Key Points

Although some constraints are neither anti-monotone nor monotone in general, several of them can be converted into anti-monotone or monotone ones by properly ordering the items. These *convertible constraints* [1-3] possess the following nice properties. By arranging items according to some proper order $\mathcal{R}$, if an itemset S satisfies a *convertible anti-monotone constraint* C, then all prefixes of S also satisfy C. Similarly, by arranging items according to some proper order $\mathcal{R}'$, if an itemset S violates a *convertible monotone constraint* C', then any prefix of S also violates C'. Examples of convertible constraints include $avg(S.Price) \geq 50$, which expresses that the average price of all items in an itemset S is at least \$50. By arranging items in non-ascending order $\mathcal{R}$ of price, if the average price of items in an itemset S is at least \$50, then the average price of items in any prefix of S would not be lower than that of S (i.e., all prefixes of S satisfying a convertible anti-monotone constraint C also satisfy C). Similarly, by arranging items in non-descending order $\mathcal{R}^{-1}$ of price, if the average price of items in an itemset S falls below \$50, then the average price of items in any prefix of S would not be higher than that of S (i.e., any prefix of S violating a convertible monotone constraint C also violates C). Note that (i) any *anti-monotone constraint* is also convertible anti-monotone (for any order $\mathcal{R}$) and (ii) any *monotone constraint* is also convertible monotone (for any order $\mathcal{R}'$).

Cross-references

► Frequent Itemset Mining with Constraints

Recommended Reading

1 Pei J. and Han J. Can we push more constraints into frequent pattern mining? In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 350–354.

2. Pei J., Han J., and Lakshmanan L.V.S. Mining frequent item sets with convertible constraints. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 433–442.

3. Pei J., Han J., and Lakshmanan L.V.S. Pushing convertible constraints in frequent itemset mining. Data Mining Knowl. Discov. 8(3):227–252, 2004.

C

Cooperative Classification

► Visual Classification

Cooperative Content Distribution

► Peer-To-Peer Content Distribution

Cooperative Storage Systems

► Peer-to-Peer Storage

Coordination

W.M.P. van der Aalst
Eindhoven University of Technology, Eindhoven, The Netherlands

Definition

Coordination is about managing dependencies between activities, processes, and components. Unlike the classical computation models, a coordination model puts much more emphasis on communication and cooperation than computation.

Key Points

Turing machines are a nice illustration of the classical "computation-oriented" view of systems. However, this view is too limited for many applications (e.g., web services). Many systems can be viewed as a collection of interacting entities (e.g., communicating Turing machines). For example, in the context of a service oriented architecture (SOA) coordination is more important than computation. There exist many approaches to model and support coordination. Linda is an example

of a language to model coordination and communication among several parallel processes operating upon objects stored in and retrieved from a shared, virtual, associative memory [1]. Linda attempts to separate coordination from computation by only allowing interaction through tuplespaces. However, one could argue that this is also possible in classical approaches such as Petri nets (e.g., connect processes through shared places), synchronized transition systems/automata, process algebra, etc. Coordination also plays an important role in agent technology [2].

Some authors emphasize the interdisciplinary nature of coordination [3]. Coordination is indeed not a pure computer science issue and other disciplines like organizational theory, economics, psychology, etc. are also relevant.

Cross-references

- ▶ Business Process Management
- ▶ Choreography
- ▶ Web Services
- ▶ Workflow Management

Recommended Reading

1. Gelernter D. and Carriero N. Coordination languages and their significance. Commun. ACM, 35(2):97–107, 1992.
2. Jennings N.R. Commitments and conventions: the foundation of coordination in multi-agent systems. Knowl. Eng. Rev., 8(3):223–250, 1993.
3. Malone T.W. and Crowston K. The interdisciplinary study of coordination. ACM Comput. Surv., 26(1):87–119, 1994.

||-Coords

- ▶ Parallel Coordinates

Copy Divergence

- ▶ Weak Consistency Models for Replicated Data

Copy Transparency

- ▶ Strong Consistency Models for Replicated Data

Copyright Issues in Databases

Michael W. Carroll
Villanova University School of Law, Villanova, PA, USA

Synonyms

Intellectual property; License

Definition

Copyright is a set of exclusive rights granted by law to authors of original works of authorship. It applies automatically as soon as an original work is created and fixed in a tangible medium of expression, such as when it is stored on a hard disk. Originality requires independent creation by the author and a modicum of creativity. Copyright covers only an author's original expression. Facts and ideas are not copyrightable. Copyright usually applies only partially to databases. Copyrightable expression usually is found in database structures, such as the selection and arrangement of field names, unless these do not reflect any creativity or are standard within an area of research. Copyright will also apply to creative data, such as photographs or expressive and sufficiently long text entries. By and large, the rule on facts and ideas means that most numerical data, scientific results, other factual data, and short text entries are not covered by copyright.

Historical Background

Copyright has evolved from a limited right to control the unauthorized distribution of a limited class of works, primarily books, to a more expansive set of rights that attach automatically to any original work of authorship. Copyright law has always been national in scope, but through international treaties most nations now extend copyright to non-resident copyright owners. To comply with these treaties, copyright is now also automatic in the USA, which has abandoned requirements that a copyright owner register the work with the Copyright Office or publish the work with the copyright symbol – © – in order to retain copyright.

Foundations

Copyright

Copyright attaches to an original work of authorship that has been embodied in a fixed form. The "work" to

which copyright attaches can be the structure of the database or a relatively small part of a database, including an individual data element, such as a photograph. It is therefore possible for a database to contain multiple overlapping copyrighted works or elements. To the extent that a database owner has a copyright, or multiple copyrights, in elements of a database, the rights apply only to those copyrighted elements. The rights are to reproduce, publicly distribute or communicate, publicly display, publicly perform, and prepare adaptations or derivative works.

Standards for Obtaining Copyright

Originality Copyright protects only an author's "original" expression, which means expression independently created by the author that reflects a minimal spark of creativity. A database owner may have a copyright in the database structure or in the user interface with the database, whether that be a report form or an electronic display of field names associated with data. The key is whether the judgments made by the person(s) selecting and arranging the data require the exercise of sufficient discretion to make the selection or arrangement "original." In *Feist Publications, Inc. v. Rural Telephone Service Company*, the US Supreme Court held that a white pages telephone directory could not be copyrighted. The data—the telephone numbers and addresses – were "facts" which were not original because they had no "author." Also, the selection and arrangement of the facts did not meet the originality requirement because the decision to order the entries alphabetically by name did not reflect the "minimal spark" of creativity needed.

As a practical matter, this originality standard prevents copyright from applying to complete databases – i.e., those that list all instances of a particular phenomenon – that are arranged in an unoriginal manner, such as alphabetically or by numeric value. However, courts have held that incomplete databases that reflect original selection and arrangement of data, such as a guide to the "best" restaurants in a city, are copyrightable in their selection and arrangement. Such a copyright would prohibit another from copying and posting such a guide on the Internet without permission. However, because the copyright would be limited to that particular selection and arrangement of restaurants, a user could use such a database as a reference for creating a different selection and arrangement of restaurants without violating the copyright owner's copyright.

Copyright is also limited by the merger doctrine, which appears in many database disputes. If there are only a small set of practical choices for expressing an idea, the law holds that the idea and expression merge and the result is that there is no legal liability for using the expression.

Under these principles, metadata is copyrightable only if it reflects an author's original expression. For example, a collection of simple bibliographic metadata with fields named "author," "title," "date of publication," would not be sufficiently original to be copyrightable. More complex selections and arrangements may cross the line of originality. Finally, to the extent that software is used in a databases, software is protectable as a "literary work." A discussion of copyright in executable code is beyond the scope of this entry.

Fixation A work must also be "fixed" in any medium permitting the work to be perceived, reproduced, or otherwise communicated for a period of more than a transitory duration. The structure and arrangement of a database may be fixed any time that it is written down or implemented. For works created after January 1, 1978 in the USA, exclusive rights under copyright shower down upon the creator at the moment of fixation.

The Duration of Copyright

Under international treaties, copyright must last for at least the life of the author plus 50 years. Some countries, including the USA, have extended the length to the life of the author plus 70 years. Under U.S. law, if a work was made as a "work made for hire," such as a work created by an employee within the scope of employment, the copyright lasts for 120 years from creation if the work is unpublished or 95 years from the date of publication.

Ownership and Transfer of Copyright

Copyright is owned initially by the author of the work. If the work is jointly produced by two or more authors, such as a copyrightable database compiled by two or more scholars, each has a legal interest in the copyright. When a work is produced by an employee, ownership differs by country. In the USA, the employer is treated as the author under the "work made for hire" doctrine and the employee has no rights in the resulting

work. Elsewhere, the employee is treated as the author and retains certain moral rights in the work while the employer receives the economic rights in the work. Copyrights may be licensed or transferred. A non-exclusive license, or permission, may be granted orally or even by implication. A transfer or an exclusive license must be done in writing and signed by the copyright owner. Outside of the USA, some or all of the author's moral rights cannot be transferred or terminated by agreement. The law on this issue varies by jurisdiction.

The Copyright Owner's Rights

The rights of a copyright owner are similar throughout the world although the terminology differs as do the limitations and exceptions to these rights.

Reproduction As the word "copyright" implies, the owner controls the right to reproduce the work in copies. The reproduction right covers both exact duplicates of a work and works that are "substantially similar" to the copyrighted work when it can be shown that the alleged copyist had access to the copyrighted work. In the USA, some courts have extended this right to cover even a temporary copy of a copyrighted work stored in a computer's random access memory ("RAM").

Public Distribution, Performance, Display or Communication The USA divides the rights to express the work to the public into rights to distribute copies, display a copy, or publicly perform the work. In other parts of the world, these are subsumed within a right to communicate the work to the public.

Within the USA, courts have given the distribution right a broad reading. Some courts, including the appeals court in the Napster case, have held that a download of a file from a server connected to the internet is both a reproduction by the person requesting the file and a distribution by the owner of the machine that sends the file. The right of public performance applies whenever the copyrighted work can be listened to or watched by members of the public at large or a subset of the public larger than a family unit or circle of friends. Similarly, the display right covers works that can be viewed at home over a computer network as long as the work is accessible to the public at large or a subset of the public.

Right of Adaptation, Modification or Right to Prepare Derivative Works A separate copyright arises with respect to modifications or adaptations of a copyrighted work so long as these modifications or adaptations themselves are original. This separate copyright applies only to these changes. The copyright owner has the right to control such adaptations unless a statutory provision, such as fair use, applies.

Theories of Secondary Liability

Those who build or operate databases also have to be aware that copyright law holds liable certain parties that enable or assist others in infringing copyright. In the USA, these theories are known as contributory infringement or vicarious infringement.

Contributory Infringement Contributory copyright infringement requires proof that a third party intended to assist a copyright infringer in that activity. This intent can be shown when one supplies a means of infringement with the intent to induce another to infringe or with knowledge that the recipient will infringe. This principle is limited by the so-called *Sony* doctrine, by which one who supplies a service or technology that enables infringement, such as a VCR or photocopier, will be deemed not to have knowledge of infringement or intent to induce infringement so long as the service or technology is capable of substantial non-infringing uses.

Two examples illustrate the operation of this rule. In *A&M Records, Inc. v. Napster, Inc.*, the court of appeals held that peer-to-peer file sharing is infringing but that Napster's database system for connecting users for peer-to-peer file transfers was capable of substantial non-infringing uses and so it was entitled to rely on the *Sony* doctrine. (Napster was held liable on other grounds.) In contrast, in *MGM Studios, Inc. v. Grokster, Ltd.*, the Supreme Court held that Grokster was liable for inducing users to infringe by specifically advertising its database service as a substitute for Napster's.

Vicarious Liability for Copyright Infringement Vicarious liability in the USA will apply whenever (i) one has control or supervisory power over the direct infringer's infringing conduct and (ii) one receives a direct financial benefit from the infringing conduct. In the Napster case, the court held that Napster had control over its users because it could refuse them access to the Napster server and, pursuant to the Terms of Service Agreements entered into with users, could terminate access if infringing conduct was discovered. Other courts have

required a greater showing of actual control over the infringing conduct.

Similarly, a direct financial benefit is not limited to a share of the infringer's profits. The Napster court held that Napster received a direct financial benefit from infringing file trading because users' ability to obtain infringing audio files drew them to use Napster's database. Additionally, Napster could potentially receive a financial benefit from having attracted a larger user base to the service.

Limitations and Exceptions

Copyrights' limitations and exceptions vary by jurisdiction. In the USA, the broad "fair use" provision is a fact-specific balancing test that permits certain uses of copyrighted works without permission. Fair use is accompanied by some specific statutory limitations that cover, for example, certain uses in the classroom use and certain uses by libraries. The factors to consider for fair use are: (i) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (ii) the nature of the copyrighted work; (iii) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (iv) the effect of the use upon the potential market for or value of the copyrighted work. The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

Countries whose copyright law follows that of the United Kingdom, a more limited "fair dealing" provision enumerates specific exceptions to copyright. In Europe, Japan, and elsewhere, the limitations and exceptions are specified legislatively and cover some private copying and some research or educational uses.

Remedies and Penalties

In general, a copyright owner can seek an injunction against one who is either a direct or secondary infringer of copyright. The monetary consequences of infringement differ by jurisdiction. In the USA, the copyright owner may choose between actual or statutory damages. Actual damages cover the copyright owner's lost profits as well as a right to the infringer's profits derived from infringement. The range for statutory damages is $750–$30,000 per copyrighted work infringed. If infringement is found to have been willful, the range increases to $150,000. The amount of statutory damages in a specific case is determined by the jury. There is a safe harbor from statutory damages for non-profit educational institutions if an employee reproduces a copyrighted work with a good faith belief that such reproduction is a fair use.

A separate safe harbor scheme applies to online service providers when their database is comprised of information stored at the direction of their users. An example of such a database would be YouTube's video sharing database. The service provider is immune from monetary liability unless the provider has knowledge of infringement or has control over the infringer and receives a direct financial benefit from infringement. The safe harbor is contingent on a number of requirements, including that the provider have a copyright policy that terminates repeat infringers, that the provider comply with a notice-and-takedown procedure, and that the provider have an agent designated to receive notices of copyright infringement.

Key Applications

In cases arising after the *Feist* decision, the courts have faithfully applied the core holding that facts are in the public domain and free from copyright even when substantial investments are made to gather such facts. There has been more variation in the characterization of some kinds of data as facts and in application of the modicum-of-creativity standard to the selections and arrangements in database structures.

On the question of when data is copyrightable, a court of appeals found copyrightable expression in the "Red Book" listing of used car valuations. The defendant had copied these valuations into its database, asserting that it was merely copying unprotected factual information. The court disagreed, likening the valuations to expressive opinions and finding a modicum of originality in these. In addition, the selection and arrangement of the data, which included a division of the market into geographic regions, mileage adjustments in 5,000-mile increments, a selection of optional features for inclusion, entitled the plaintiff to a thin copyright in the database structure.

Subsequently, the same court found that the prices for futures contracts traded on the New York Mercantile Exchange (NYMEX) probably were not expressive data even though a committee makes some judgments in the setting of these prices. The court concluded that even if such price data were expressive, the merger doctrine applied because there was no other practicable way of expressing the idea other than through a

numerical value and a rival was free to copy price data from NYMEX's database without copyright liability.

Finally, where data are comprised of arbitrary numbers used as codes, the courts have split. One court of appeals has held that an automobile parts manufacturer owns no copyright in its parts numbers, which are generated by application of a numbering system that the company created. In contrast, another court of appeals has held that the American Dental Association owns a copyright in its codes for dental procedures.

On the question of copyright in database structures, a court of appeals found that the structure of a yellow pages directory including listing of Chinese restaurants was entitled to a "thin" copyright, but that copyright was not infringed by a rival database that included 1,500 of the listings because the rival had not copied the plaintiff's data structure. Similarly, a different court of appeals acknowledged that although a yellow pages directory was copyrightable as a compilation, a rival did not violate that copyright by copying the name, address, telephone number, business type, and unit of advertisement purchased for each listing in the original publisher's directory. Finally, a database of real estate tax assessments that arranged the data collected by the assessor into 456 fields grouped into 34 categories was sufficiently original to be copyrightable.

Cross-references

► European Law in Databases
► Licensing and Contracting Issues in Databases

Recommended Reading

1. American Dental Association v. Delta Dental Plans Ass'n, 126 F.3d 977 (7th Cir.1997).
2. Assessment Technologies of WI, LLC v. WIRE data, Inc., 350 F.3d 640 (7th Cir. 2003).
3. Bellsouth Advertising & Publishing Corp. v. Donnelly Information Publishing, Inc., 999 F.2d 1436 (11th Cir. 1993) (en banc).
4. CCC Information Services, Inc. v. MacLean Hunter Market Reports, Inc., 44 F.3d 61 (2d Cir. 1994).
5. Feist Publications, Inc. v. Rural Telephone Service Co., 499 U.S. 340 (1991).
6. Ginsburg J.C. Copyright, common law, and sui generis protection of databases in the United States and abroad, University of Cincinnati Law Rev., 66:151–176, 1997.
7. Key Publications, Inc. v. Chinatown Today Publishing Enterprises, Inc., 945 F.2d 509 (2d Cir. 1991).
8. New York Mercantile Exchange, Inc. v. Intercontinental-Exchange, Inc., 497 F.3d 109, (2d Cir. 2007).
9. Southco, Inc. v. Kanebridge Corp., 390 F.3d 276 (3d Cir. 2004) (en banc).

CORBA

Aniruddha Gokhale
Vanderbilt University, Nashville, TN, USA

Synonyms

Object request broker; Common object request broker architecture

Definition

The Common Object Request Broker Architecture (CORBA) [2,3] is standardized by the Object Management Group (OMG) for distributed object computing.

Key Points

The CORBA standard specifies a platform-independent and programming language-independent architecture and a set of APIs to simplify distributed application development. The central idea in CORBA is to decouple the interface from the implementation. Applications that provide services declare their interfaces and operations in the Interface Description Language (IDL). IDL compilers read these definitions and synthesize client-side stubs and server-side skeletons, which provide data marshaling and proxy capabilities.

CORBA provides both a type-safe RPC-style object communication paradigm called the Static Invocation Interface (SII), and a more dynamic form of communication called the Dynamic Invocation Interface (DII), which allows creation and population of requests dynamically via reflection capabilities. The DII is often used to bridge different object models. CORBA defines a binary format for on-the-wire representation of data called the Common Data Representation (CDR). CDR has been defined to enable programming language-neutrality.

The CORBA 1.0 specification (October 1991) and subsequent revisions through version 1.2 (December 1993) defined these basic capabilities, however, they lacked any support for interoperability across different CORBA implementations.

The CORBA 2.0 specification (August 1996) defined an interoperability protocol called the General Inter-ORB Protocol (GIOP), which defines the packet formats for data exchange between communicating CORBA entities. GIOP is an abstract specification and must be mapped to the underlying transport protocol. The most widely used concrete mapping of GIOP is

called the Internet Inter-ORB Protocol (IIOP) used for data exchange over TCP/IP networks.

Despite these improvements, the earlier versions of CORBA focused only on the client-side portability and lacked any support for server-side portability. This limitation was addressed in the CORBA 2.2 specification (August 1996) through the Portable Object Adapter (POA) concept. The POA enables server-side transparency to applications and server-side portability. The POA provides a number of policies that can be used to manage the server-side objects.

The CORBA specification defines compliance points for implementations to ensure interoperability. The CORBA specification has also been enhanced with additional capabilities that are available beyond the basic features, such as the Real-time CORBA specification [1]. Implementations of these specifications must provide these additional capabilities.

In general, CORBA enhances conventional procedural RPC middleware by supporting object oriented language features (such as encapsulation, interface inheritance, parameterized types, and exception handling) and advanced design patterns for distributed communication. The most recent version of CORBA specification at the time of this writing is 3.3 (January 2008), which also includes support for a component architecture.

Cross-references

- ▶ Client-Server Architecture
- ▶ DCE
- ▶ DCOM
- ▶ J2EE
- ▶ Java RMI
- ▶ .NET Remoting
- ▶ Request Broker
- ▶ SOAP

Recommended Reading

1. Object Management Group, Real-Time CORBA Specification, Version 1.2, OMG Document No. formal/2005-01-04, January 2005.
2. Object Management Group, Common Object Request Broker Architecture (CORBA), Version 3.1, OMG Document No. formal/2008-01-08, January 2008.
3. Soley R.M. and Stone C.M. Object Management Architecture Guide, 3rd edn., Object Management Group, June 1995.

Corpora

- ▶ Document Databases

Corpus

- ▶ Test Collection

Correctness Criteria Beyond Serializability

Mourad Ouzzani[1], Brahim Medjahed[2], Ahmed K. Elmagarmid[1]
[1]Purdue University, West Lafayette, IN, USA
[2]The University of Michigan – Dearborn, Dearborn, MI, USA

Synonyms

Concurrency control; Preserving database consistency

Definition

A *transaction* is a logical unit of work that includes one or more database access operations such as insertion, deletion, modification, and retrieval [8]. A *schedule* (or history) S of n transactions $T_1,...,T_n$ is an ordering of the transactions that satisfies the following two conditions: (i) the operations of T_i (i = 1,...,n) in S must occur in the same order in which they appear in T_i, and (ii) operations from T_j ($j \neq i$) may be interleaved with T_i's operations in S. A schedule S is *serial* if for every two transactions T_i and T_j that appear in S, either all operations of T_i appear before all operations of T_j, or vice versa. Otherwise, the schedule is called *nonserial* or *concurrent*. Non-serial schedules of transactions may lead to concurrency problems such as lost update, dirty read, and unrepeatable read. For instance, the lost update problem occurs whenever two transactions, while attempting to modify a data item, both read the item's old value before either of them writes the item's new value [2].

The simplest way for controlling concurrency is to allow only serial schedules. However, with no concurrency, database systems may make poor use of their resources and hence, be inefficient, resulting in smaller transaction execution rate for example. To broaden the class of allowable transaction schedules, *serializability* has been proposed as the major correctness criterion for concurrency control [7,11]. Serializability ensures that a concurrent schedule of transactions is equivalent to some serial schedule of the same transactions [12]. While serializability has been successfully used in

traditional database applications, e.g., airline reservations and banking, it has been proven to be restrictive and hardly applicable in advanced applications such as Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), office automation, and multidatabases. These applications introduced new requirements that either prevent the use of serializability (e.g., violation of local autonomy in multidatabases) or make the use of serializability inefficient (e.g., long-running transactions in CAD/CAM applications). These limitations have motivated the introduction of more flexible correctness criteria that go beyond the traditional serializability.

Historical Background

Concurrency control began appearing in database systems in the early to mid 1970s. It emerged as an active database research thrust starting from 1976 as witnessed by the early influential papers published by Eswaren et al. [5] and Gray et al. [7]. A comprehensive coverage of serializability theory has been presented in 1986 by Papadimitriou in [12]. Simply put, serializability theory is a mathematical model for proving whether or not a concurrent execution of transactions is correct. It gives precise definitions and properties that non-serial schedules of transactions must satisfy to be serializable. Equivalence between a concurrent and serial schedule of transactions is at the core of the serializability theory. Two major types of equivalence have then been defined: *conflict* and *view* equivalence. If two schedules are conflict equivalent then they are view equivalent. The converse is not generally true.

Conflict equivalence has initially been introduced by Gray et al. in 1975 [7]. A concurrent schedule of transactions is *conflict equivalent* to a serial schedule of the same transactions (and hence *conflict serializable*) if they order conflicting operations in the same way, i.e., they have the same precedence relations of conflicting operations. Two operations are *conflicting* if they are from different transactions upon the same data item, and at least one of them is `write`. If two operations conflict, their execution order matters. For instance, the value returned by a `read` operation depends on whether or not that operation precedes or follows a particular `write` operation on the same data item. Conflict serializability is tested by analyzing the acyclicity of the graph derived from the execution of the different transactions in a schedule. This graph, called *serializability graph*, is a directed graph that models the precedence of conflicting operations in the transactions.

View equivalence has been proposed by Yannakakis in 1984 [15]. A concurrent schedule of transactions is *view equivalent* to a serial schedule of the same transactions (and hence *view serializable*) if the respective transactions in the two schedules read and write the same data values. View equivalence is based on the following two observations: (i) if each transaction reads each of its data items from the same `write`s, then all `write`s write the same value in both schedules; and (ii) if the final `write` on each data item is the same in both schedules, then the final value of all data items will be the same in both schedules. View serializability is usually expensive to check. One approach is to check the acyclicity of a special graph called *polygraph*. A polygraph is a generalization of the precedence graph that takes into account all precedence constraints required by view serializability.

Foundations

The limitations of the traditional serializability concept combined with the requirement of advanced database applications triggered a wave of new correctness criteria that go beyond serializability. These criteria aim at achieving one or several of the following goals: (i) accept non serializable but correct executions by exploiting the semantics of transactions, their structure, and integrity constraints (ii) allow inconsistencies to appear in a controlled manner which may be acceptable for some transactions, (iii) limit conflicts by creating a new version of the data for each update, and (iv) treat transactions accessing more than one database, in the case of multidatabases, differently from those accessing one single database and maintain overall correctness. While a large number of correctness criteria have been presented in the literature, this entry will focus on the major criteria which had a considerable impact on the field. These criteria will be presented as described in their original versions as several of these criteria have been either extended, improved, or applied to specific contexts. Table 1 summarizes the correctness criteria outlined in this section.

Multiversion Serializability

Multiversion databases aim at increasing the degree of concurrency and providing a better system recovery. In such databases, whenever a transaction writes a data

Correctness Criteria Beyond Serializability. Table 1. Representative correctness criteria for concurrency control

Correctness criterion	Basic idea	Examples of application domains	Reference
Multiversion serializability	Allows some schedules as serializable if a read is performed on some older version of a data item instead of the newer modified version.	Multiversion database systems	[1]
Semantic consistency	Uses semantic information about transactions to accept some non-serializable but correct schedules.	Applications that can provide some semantic knowledge	[6]
Predicatewise serializability	Focuses on data integrity constraints.	CAD database and office information systems	[9]
Epsilon-serializability	Allows inconsistencies to appear in a controlled manner by attaching a specification of the amount of permitted inconsistency to each transaction.	Applications that tolerate some inconsistencies	[13]
Eventual consistency	Requires that duplicate copies are consistent at certain times but may be inconsistent in the interim intervals.	Distributed databases with replicated or interdependent data	[14]
Quasi serializability	Executes global transactions in a serializable way while taking into account the effect of local transactions.	Multidatabase systems	[4]
Two-level serializability	Ensures consistency by exploiting the nature of integrity constraints and the nature of transactions in multidatabase environments.	Multidatabase systems	[10]

item, it creates a new version of this item instead of overwriting it. The basic idea of *multiversion serializability* [1] is that some schedules can be still seen as serializable if a read is performed on some older version of a data item instead of the newer modified version. Concurrency is increased by having transactions read older versions while other concurrent transactions are creating newer versions. There is only one type of conflict that is possible; when a transactions reads a version of a data item that was written by another transaction. The two other conflicts (write, write) and (read, write) are not possible since each write produces a new version and a data item cannot be read until it has been produced, respectively. Based on the assumption that users expect their transactions to behave as if there were just one copy of each data item, the notion of *one-copy serial* schedule is defined. A schedule is one-copy serial if for all i, j, and x, if a transaction T_j reads x from a transaction T_i, then either $i = j$ or T_i is the last transaction preceding t_j that writes into any version of x. Hence, a schedule is defined as *one-copy serializable* (1-SR) if it is equivalent to a 1-serial schedule. 1-SR is shown to maintain correctness by proving that a multiversion schedule behaves like a serial non-multiversion schedule (there is only one version for each data item) if the multiversion schedule is one-serializable. The one-copy serializability of a schedule can be verified by checking the acyclicity of the multiversion serialization graph of that schedule.

Semantic Consistency

Semantic consistency uses semantic information about transactions to accept some non-serializable but correct schedules [6]. To ensure that users see consistent data, the concept of *sensitive transactions* has been introduced. Sensitive transactions output only consistent data and thus must see a consistent database state. A semantically consistent schedule is one that transforms the database from a consistent state to another consistent state and where all sensitive transactions obtain a consistent view of the database with respect to the data accessed by these transactions, i.e., all data consistency constraints of the accessed data are evaluated to True. Enforcing semantic consistency requires knowledge about the application which must be provided by the user. In particular, users will need to group actions of the transactions into steps and specify which steps of a transaction of a given type can be interleaved with the steps of another type of transactions without violating consistency. Four types of semantic knowledge are defined: (i) transaction semantic types, (ii) compatibility sets associated with each type, (iii) division of transactions into steps,

and (iv) counter-steps to (semantically) compensate the effect from some of the steps executed within the transaction.

Predicatewise Serializability

Predicatewise serializability (PWSR) has been introduced as a correctness criterion for CAD database and office information systems [9]. PWSR focuses solely on data integrity constraints. In a nutshell, if database consistency constraints can be expressed in a conjunctive normal form, a schedule is said to be PWSR if all projections of that schedule on each group of data items that share a disjunctive clause (of the conjunctive form representing the integrity constraints) are serializable. There are three different types of restrictions that must be enforced on PWSR schedules to preserve database consistency: (i) force the transactions to be of *fixed structure*, i.e., they are independent of the database state from which they execute, (ii) force the schedules to be *delayed read*, i.e., a transaction T_i cannot read a data item written by a transaction T_j until after T_j has completed all of its operations, or (iii) the conjuncts of the integrity constraints can be ordered in a way that no transaction reads a data item belonging to a higher numbered conjunct and writes a data item belonging to a lower numbered conjunct.

Epsilon-Serializability

Epsilon-serializability (ESR) [13] has been introduced as a generalization to serializability where a limited amount of inconsistency is permitted. The goal is to enhance concurrency by allowing some non serializable schedules. ESR introduces the notion of *epsilon transactions* (ETs) by attaching a specification of the amount of permitted inconsistency to each (standard) transaction. ESR distinguishes between transactions that contain only read operation, called query epsilon transaction or query ET, and transactions with at least one update operation, called update epsilon transaction or update ET. Query ETs may view uncommitted, possibly inconsistent, data being updated by update ETs. Thus, update ETs are seen as exporting some inconsistencies while query ETs are importing these inconsistencies. ESR aims at bounding the amount of imported and exported inconsistency for each ET. An *epsilon-serial* schedule is defined as a schedule where (i) the update ETs form a serial schedule if considered alone without the query ET and (ii) the entire schedule consisting of both query ETs and update ETs is such that the non serializable conflicts between query ETs and update ETs are less than the permitted limits specified by each ET. An epsilon-serializable schedule is one that is equivalent to an epsilon-serial schedule. If the permitted limits are set to zero, ESR corresponds to the classical notion of serializability.

Eventual Consistency

Eventual consistency has been proposed as an alternative correctness criterion for distributed databases with replicated or interdependent data [14]. This criterion is useful is several applications like mobile databases, distributed databases, and large scale distributed systems in general. Eventual consistency requires that duplicate copies are consistent at certain times but may be inconsistent in the interim intervals. The basic idea is that duplicates are allowed to diverge as long as the copies are made consistent periodically. The times where these copies are made consistent can be specified in several ways which could depend on the application, for example, at specified time intervals, when some events occur, or at some specific times. A correctness criterion that ensures eventual consistency is the *current copy serializability*. Each update occurs on a current copy and is asynchronously propagated to other replicas.

Quasi Serializability

Quasi Serializability (QSR) is a correctness criterion that has been introduced for multidatabase systems [4]. A multidatabase system allows users to access data located in multiple autonomous databases. It generally involves two kinds of transactions: (i) Local transactions that access only one database; they are usually outside the control of the multidatabase system, and (ii) global transactions that can access more than one database and are subject to control by both the multidatabase and the local databases. The basic premise is that to preserve global database consistency, global transactions should be executed in a serializable way while taking into account the effect of local transactions. The effect of local transactions appears in the form of indirect conflicts that these local transactions introduce between global transactions which may not necessarily access (conflict) the same data items. A *quasi serial* schedule is a schedule where global transactions are required to execute serially and local schedules are required to be serializable. This is in contrast to global serializability where all transactions,

both local and global, need to execute in a (globally) serializable way. A global schedule is said to be quasi serializable if it is (conflict) equivalent to a quasi serial schedule. Based on this definition, a quasi serializable schedule maintains the consistency of multidatabase systems since (i) a quasi serial schedule preserves the mutual consistency of globally replicated data items, based on the assumptions that these replicated data items are updated only by global transactions, and (ii) a quasi serial schedule preserves the global transaction consistency constraints as local schedules are serializable and global transactions are executed following a schedule that is equivalent to a serial one.

Two-Level Serializability

Two-level serializability (2LSR) has been introduced to relax serializability requirements in multidatabases and allow a higher degree of concurrency while ensuring consistency [10]. Consistency is ensured by exploiting the nature of integrity constraints and the nature of transactions in multidatabase environments. A global schedule, consisting of both local and global transactions, is 2LSR if all local schedules are serializable and the projection of that schedule on global transactions is serializable. Local schedules consist of all operations, from global and local transactions, that access the same local database. Ensuring that each local schedule is serializable is already taken care of by the local database. Furthermore, ensuring that the global transactions are executed in a serializable way can be done by the global concurrency controller using any existing technique from centralized databases like the Two-phase-locking (2PL) protocol. This is possible since the global transactions are under the full control of the global transaction manager. [10] shows that under different scenarios 2LSR preserves a strong notion of correctness where the multidatabase consistency is preserved and all transactions see consistent data. These different scenarios differ depending on (i) which kind of data items, local or global, global and local transactions are reading or writing, (ii) the existence of integrity constraints between local and global data items, and (iii) whether all transaction are preserving the consistency of local databases when considered alone.

Key Applications

The major database applications behind the need for new correctness criteria include distributed databases, mobile databases, multidatabases, CAD/CAM applications, office automation, cooperative applications, and software development environments. All of these advanced applications introduced requirements and limitations that either prevent the use of serializability like the violation of local autonomy in multidatabases, or make the use of serializability inefficient like blocking long-running transactions.

Future Directions

A recent trend in transaction management focuses on adding transactional properties (e.g., isolation, atomicity) to business processes [3]. A business process (BP) is a set of tasks which are performed collaboratively to realize a business objective. Since BPs contain activities that access shared and persistent data resources, they have to be subject to transactional semantics. However, it is not adequate to treat an entire BP as a single "traditional" transaction mainly because BPs: (i) are of long duration and treating an entire process as a transaction would require locking resources for long periods of time, (ii) involve many independent database and application systems and enforcing transactional properties across the entire process would require expensive coordination among these systems, and (iii) have external effects and using conventional transactional rollback mechanisms is not feasible. These characteristics open new research issues to take the concept of correctness criterion and how it should be enforced beyond even the correctness criteria discussed here.

Cross-references

- ▶ ACID Properties
- ▶ Concurrency Control
- ▶ Distributed
- ▶ Parallel and Networked Databases
- ▶ System Recovery
- ▶ Transaction Management
- ▶ Two-Phase Commit
- ▶ Two-Phase Locking

Recommended Reading

1. Bernstein P.A. and Goodman N. Multiversion concurrency control – theory and algorithms. ACM Trans. Database Syst., 8(4):465–483, 1983.
2. Bernstein P.A., Hadzilacos V., and Goodman N. Concurrency control and recovery in database systems. Addison-Wesley, Reading, MA, 1987.
3. Dayal U., Hsu M., and Ladin R. Business process coordination: state of the art, trends, and open issues. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 3–13.

4. Du W. and Elmagarmid A.K. Quasi serializability: a correctness criterion for global concurrency control in Interbase. In Proc. 15th Int. Conf. on Very Large Data Bases, 1989, pp. 347–355.
5. Eswaran K.P., Gray J., Lorie R.A., and Traiger I.L. The notions of consistency and predicate locks in a database system. Commun. ACM, 19(11):624–633, 1976.
6. Garcia-Molina H. Using semantic knowledge for transaction processing in a distributed database. ACM Trans. Database Syst., 8(2):186–213, 1983.
7. Gray J., Lorie R.A., Putzolu G.R., and Traiger I.L. Granularity of locks in a large shared data base. In Proc. 1st Int. Conf. on Very Data Bases, 1975, pp. 428–451.
8. Gray J. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, Los Altos, CA, 1993.
9. Korth H.F. and Speegle G.D. Formal model of correctness without serializability. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1988, pp. 379–386.
10. Mehrotra S., Rastogi R., Korth H.F., and Silberschatz A. Ensuring consistency in multidatabases by preserving two-level serializability. ACM Trans. Database Syst., 23(2):199–230, 1998.
11. Papadimitriou C.H. The serializability of concurrent database updates. J. ACM, 26(4):631–653, 1979.
12. Papadimitriou C.H. The Theory of Database Concurrency Control. Computer Science, Rockville, MD, 1986.
13. Ramamritham K. and Pu C. A formal characterization of epsilon serializability. IEEE Trans. Knowl. Data Eng., 7(6):997–1007, 1995.
14. Sheth A., Leu Y., and Elmagarmid A. Maintaining Consistency of Interdependent Data in Multidatabase Systems. Tech. Rep. CSD-TR-91-016, Purdue University, http://www.cs.toronto.edu/georgem/ws/ws.ps, 1991.
15. Yannakakis M. Serializability by locking. J. ACM, 31(2): 227–244, 1984.

Correctness Criterion for Concurrent Executions

▶ Serializability

Correlated Data Collection

▶ Data Compression in Sensor Networks

Correlation

▶ Similarity and Ranking Operations

Correlation Clustering

▶ Subspace Clustering Techniques

Cost Estimation

Stefan Manegold
CWI, Amsterdam, The Netherlands

Definition

Execution costs, or simply *costs*, is a generic term to collectively refer to the various goals or objectives of database query optimization. Optimization aims at finding the "cheapest" ("best" or at least a "reasonably good") query execution plan (QEP) among semantically equivalent alternative plans for the given query. Cost is used as a metric to compare plans. Depending on the application different types of costs are considered. Traditional optimization goals include minimizing response time (for the first answer or the complete result), minimizing resource consumption (like CPU time, I/O, network bandwidth, or amount of memory required), or maximizing throughput, i.e., the number of queries that the system can answer per time. Other, less obvious objectives – e.g., in a mobile environment – may be to minimize the power consumption needed to answer the query or the on-line time being connected to a remote database server.

Obviously, evaluating a QEP to measure its execution cost does not make sense. *Cost estimation* refers to the task of predicting the (approximate) costs of a given QEP a priori, i.e., without actually evaluating it. For this purpose, mathematical algorithms or parametric equations, commonly referred to as *cost models*, provide a simplified "idealized" abstract description of the system, focusing on the most relevant components. In general, the following three cost components are distinguished.

1. *Logical costs* consider only the data distributions and the semantics of relational algebra operations to estimate intermediate result sizes of a given (logical) query plan.
2. *Algorithmic costs* extend logical costs by taking also the computational complexity (expressed in terms of *O*-classes) of the algorithms into account.
3. *Physical costs* finally combine algorithmic costs with system/hardware specific parameters to predict the total costs, usually in terms of execution time.

Next to query optimization, cost models can serve another purpose. Especially algorithmic and physical cost models can help database developers to understand

and/or predict the performance of existing algorithms on new hardware systems. Thus, they can improve the algorithms or even design new ones without having to run time and resource consuming experiments to evaluate their performance.

Since the quality of query optimization strongly depends on the quality of cost estimation, details of cost estimation in commercial database products are usually well kept secrets of their vendors.

Historical Background

Not all aspects of database cost estimation are treated as independent research topic of their own. Mainly selectivity estimation and intermediate result size estimation have received intensive attention yielding a plethora of techniques proposed in database literature. Discussion of algorithmic costs usually occurs with the proposal of new or modified database algorithms. Given its tight coupling with query optimization, physical cost estimation has never been an independent research topic of its own. Apart from very few exceptions, new physical cost models and estimation techniques are usually published as "by-products" in publications that mainly deal with novel optimization techniques.

The first use of (implicit) cost estimation were complexity analyses that led to heuristic optimization rules. For instance, a join is always considered cheaper than calculating first the Cartesian product, followed by a selection. Likewise, linear operations that tend to reduce the data stream (selections, projections) should be evaluated as early as data dependencies allow, followed by (potentially) quadratic operations that do not "blow-up" the intermediate results (semijoins, foreign-key joins). More complex, and hence expensive, operations (general joins, Cartesian products) should be executed as late as possible.

Since a simple complexity metric does not necessarily reflect the same ranking of plans as the actual execution costs, first explicit cost estimation in database query optimization aimed at estimating intermediate result sizes. Initial works started with simplifications such as assuming uniform data distributions and independence of attribute values. Over time, the techniques have been improved to model non-uniform data distributions. Til date, effective treatment of (hidden) correlations is still an open research topic.

The following refinement was the introduction of physical costs. With I/O being the dominating cost factor in the early days of database management systems, the first systems assessed query plans by merely estimating the number of I/O operations required. However, I/O systems exhibit quite different performance for sequential and randomly placed I/O operations. Hence, the models were soon refined to distinguish between sequential and random accesses, weighing them with their respective costs, i.e., time to execute one operation.

With main memory sizes growing, more and more query processing work is done within main memory, minimizing disk accesses. Consequently, CPU and memory access costs can no longer be ignored. Assuming uniform memory access costs, memory access has initially been covered by CPU costs. CPU costs are estimated in terms of CPU cycles. Scoring them with the CPU's clock speed yields time, the common unit to combine CPU and I/O costs to get the overall physical costs.

Only recently with the advent of CPU caches and extended memory hierarchies, the impact of memory access costs has become so significant that it needs to be modeled separately [15,16]. Similarly to I/O costs, memory access costs are estimated in terms of number of memory accesses (or cache misses) and scored by their penalty to achieve time as common unit.

In parallel and distributed database systems, also network communication costs are considered as contributing factors to the overall execution costs.

Foundations

Different query execution plans require different amounts of effort to be evaluated. The objective function for the query optimization problems assigns every execution plan a single non-negative value. This value is commonly referred to as *costs* in the query optimization business.

Cost Components

Logical Costs/Data Volume The most important cost component is the amount of data that is to be processed. Per operator, three data volumes are distinguished: input (per operand), output, and temporary data. Data volumes are usually measured as cardinality, i.e., number of tuples. Often, other units such as number of I/O blocks, number of memory pages, or total size in bytes are required. Provided that the respective tuple sizes, page sizes, and block sizes are known, the cardinality can easily be transformed into the other units.

The amount of input data is given as follows: For the leaf nodes of the query graph, i.e., those operations that directly access base tables stored in the database, the input cardinality is given by the cardinality of the base table(s) accessed. For the remaining (inner) nodes of the query graph, the input cardinality is given by the output cardinality of the predecessor(s) in the query graph.

Estimating the output size of database operations – or more generally, their *selectivity* – is anything else but trivial. For this purpose, DBMSs usually maintain statistic about the data stored in the database. Typical statistics are

1. Cardinality of each table,
2. Number of distinct values per column,
3. Highest/lowest value per column (where applicable).

Logical cost functions use these statistics to estimate output sizes (respectively selectivities) of database operations. The simplest approach is to assume that attribute values are uniformly distributed over the attribute's domain. Obviously, this assumption virtually never holds for "real-life" data, and hence, estimations based on these assumption will never be accurate. This is especially severe, as the estimation errors compound exponentially throughout the query plan [9]. This shows, that more accurate (but compact) statistics on data distributions (of base tables as well as intermediate results) are required to estimate intermediate results sizes.

The importance of statistics management has led to a plethora of approximation techniques, for which [6] have coined the general term "*data synopses*". Such techniques range from advanced forms of *histograms* (most notably, *V-optimal histograms* including multi-dimensional variants) [7,10] over *spline synopses* [12,11], *sampling* [3,8], and *parametric curve-fitting techniques* [4,20] all the way to highly sophisticated methods based on *kernel estimators* [1] or *Wavelets* and other transforms [2,17].

A logical cost model is a prerequisite for the following two cost components.

Algorithmic Costs/Complexity

Logical costs only depend on the data and the query (i.e., the operators' semantics), but they do not consider the algorithms used to implement the operators' functionality. Algorithmic costs extend logical costs by taking the properties of the algorithms into account.

A first criterion is the algorithm's complexity in the classical sense of complexity theory. Most unary operator are in $O(n)$, like selections, or $O(n \log n)$, like sorting; n being the input cardinality. With proper support by access structures like indices or hash tables, the complexity of selection may drop to $O(\log n)$ or $O(1)$, respectively. Binary operators can be in $O(n)$, like a union of sets that does not eliminate duplicates, or, more often, in $O(n^2)$, as for instance join operators.

More detailed algorithmic cost functions are used to estimate, e.g., the number of I/O operations or the amount of main memory required. Though these functions require some so-called "physical" information like I/O block sizes or memory pages sizes, they are still considered algorithmic costs and not physical cost, as these informations are system specific, but not hardware specific. The standard database literature provides a large variety of cost formulas for the most frequently used operators and their algorithms. Usually, these formulas calculate the costs in terms of I/O operations as this still is the most common objective function for query optimization in database systems [5,13].

Physical Costs/Execution Time

Logical and algorithmic costs alone are not sufficient to do query optimization. For example, consider two algorithms for the same operation, where the first algorithm requires slightly more I/O operations than the second, while the second requires significantly more CPU operations than the first one. Looking only at algorithmic costs, both algorithms are not comparable. Even assuming that I/O operations are more expensive than CPU operations cannot in general answer the question which algorithm is faster. The actual execution time of both algorithms depends on the speed of the underlying hardware. The physical cost model combines the algorithmic cost model with an abstract hardware description to derive the different cost factors in terms of time, and hence the total execution time. A hardware description usually consists of information such as CPU speed, I/O latency, I/O bandwidth, and network bandwidth. The next section discusses physical cost factors on more detail.

Cost Factors

In principle, physical costs are considered to occur in two flavors, *temporal* and *spatial*. Temporal costs cover all cost factors that can easily be related to execution time, e.g., by multiplying the number of certain events with there respective cost in terms of some time unit. Spatial costs contain resource consumptions that cannot directly (or not at all) be related to time. The

following briefly describes the most prominent cost factors of both categories.

Temporal Cost Factors

Disk-I/O This is the cost of searching for, reading, and writing data blocks that reside on secondary storage, mainly on disk. In addition to accessing the database files themselves, temporary intermediate files that are too large to fit in main memory buffers and hence are stored on disk also need to be accessed. The cost of searching for records in a database file or a temporary file depends on the type of access structures on that file, such as ordering, hashing, and primary or secondary indexes. I/O costs are either simply measured in terms of the number of block-I/O operations, or in terms of the time required to perform these operations. In the latter case, the number of block-I/O operations is multiplied by the time it takes to perform a single block-I/O operation. The time to perform a single block-I/O operation is made up by an initial seek time (*I/O latency*) and the time to actually transfer the data block (i.e., block size divided by *I/O bandwidth*). Factors such as whether the file blocks are allocated contiguously on the same disk cylinder or scattered across the disk affect the access cost. In the first case (also called *sequential I/O*), I/O latency has to be counted only for the first of a sequence of subsequent I/O operations. In the second case (*random I/O*), seek time has to be counted for each I/O operation, as the disk heads have to be repositioned each time.

Main-Memory Access These are the costs for reading data from or writing data to main memory. Such data may be intermediate results or any other temporary data produced/used while performing database operations.

Similar to I/O costs, memory access costs can be modeled be estimating the number of memory accesses (i.e., cache misses) and scoring them with their respective penalty (latency) [16].

Network Communication In centralized DBMSs, communication costs cover the costs of shipping the query from the client to the server and the query's result back to the client. In distributed, federated, and parallel DBMSs, communication costs additionally contain all costs for shipping (sub-)queries and/or (intermediate) results between the different hosts that are involved in evaluating the query.

Also with communication costs, there is a latency component, i.e., a delay to initiate a network connection and package transfer, and a bandwidth component, i.e., the amount of data that can be transfer through the network infrastructure per time.

CPU Processing This is the cost of performing operations such as computations on attribute values, evaluating predicates, searching and sorting tuples, and merging tuples for join. CPU costs are measured in either CPU cycles or time. When using CPU cycles, the time may be calculated by simply dividing the number of cycles by the CPU's clock speed. While allowing limited portability between CPUs of the same kind, but with different clock speeds, portability to different types of CPUs is usually not given. The reason is, that the same basic operations like adding two integers might require different amounts of CPU cycles on different types of CPUs.

Spatial Cost Factors

Usually, there is only one spatial cost factor considered in database literature: *memory size*. This cost it the amount of main memory required to store intermediate results or any other temporary data produced/used while performing database operations.

Next to not (directly) being related to execution time, there is another difference between temporal and spatial costs that stems from the way they share the respective resources. A simple example shall demonstrate the differences. Consider to operations or processes each of which consumes 50% of the available resources (i.e., CPU power, I/O-, memory-, and network bandwidth). Further, assume that when run one at a time, both tasks have equal execution time. Running both tasks concurrently on the same system (ideally) results in the same execution time, now consuming all the available resources. In case each individual process consumes 100% of the available resources, the concurrent execution time will be twice the individual execution time. In other words, if the combined resource consumption of concurrent tasks exceed 100%, the execution time extends to accommodate the excess resource requirements. With spatial cost factors, however, such "stretching" is not possible. In case two tasks together would require more than 100% of the available memory, they simply cannot be executed at the same time, but only after another.

Types of (Cost) Models

According to their degree of abstraction, (cost) models can be classified into two classes: *analytical models* and *simulation models*.

Analytical Models In some cases, the assumptions made about the real system can be translated into

mathematical descriptions of the system under study. Hence, the result is a set of mathematical formulas that is called an analytical model. The advantage of an analytical model is that evaluation is rather easy and hence fast. However, analytical models are usually not very detailed (and hence not very accurate). In order to translate them into a mathematical description, the assumptions made have to be rather general, yielding a rather high degree of abstraction.

Simulation Models Simulation models provide a very detailed and hence rather accurate description of the system. They describe the system in terms of (a) simulation experiment(s) (e.g., using event simulation). The high degree of accuracy is charged at the expense of evaluation performance. It usually takes relatively long to evaluate a simulation base model, i.e., to actually perform the simulation experiment(s). It is not uncommon, that the simulation actually takes longer than the execution in the real system would take.

In database query optimization, though it would appreciate the accuracy, simulation models are not feasible, as the evaluation effort is far to high. Query optimization requires that costs of numerous alternatives are evaluated and compared as fast as possible. Hence, only analytical cost models are applicable in this scenario.

Architecture and Evaluation of Database Cost Models

The architecture and evaluation mechanism of database cost models is tightly coupled to the structure of query execution plans. Due to the strong encapsulation offered by relational algebra operators, the cost of each operator, respectively each algorithm, can be described individually. For this purpose, each algorithm is assigned a set of *cost functions* that calculate the three cost components as described above. Obviously, the physical cost functions depend on the algorithmic cost functions, which in turn depend on the logical cost functions. Algebraic cost functions use the data volume estimations of the logical cost functions as input parameters. Physical cost functions are usually specializations of algorithmic cost functions that are parametrized by the hardware characteristics.

The cost model also defines how the single operator costs within a query have to be combined to calculate the total costs of the query. In traditional sequential DBMSs, the single operators are assumed to have no performance side-effects on each other. Thus, the cost of a QEP is the cumulative cost of the operators in the QEP [18]. Since every operator in the QEP is the root of a sub-plan, its cost includes the cost of its input operators. Hence, the cost of a QEP is the cost of the topmost operator in the QEP. Likewise, the cardinality of an operator is derived from the cardinalities of its inputs, and the cardinality of the topmost operator represents the cardinality of the query result.

In non-sequential (e.g., distributed or parallel) DBMSs, this subject is much more complicated, as more issues such as scheduling, concurrency, resource contention, and data dependencies have to considered. For instance, in such environments, more than one operator may be executed at a time, either on disjoint (hardware) resources, or (partly) sharing resources. In the first case, the total cost (in terms of time) is calculated as the maximum of the costs (execution times) of all operators running concurrently. In the second case, the operators compete for the same resources, and hence mutually influence their performance and costs. More sophisticated cost function and cost models are required here to adequately model this resource contention [14,19].

Cross-references

- ▶ Distributed Query Optimization
- ▶ Multi-Query Optimization
- ▶ Optimization and Tuning in Data Warehouses
- ▶ Parallel Query Optimization
- ▶ Process Optimization
- ▶ Query Optimization
- ▶ Query Optimization (in Relational Databases)
- ▶ Query Optimization in Sensor Networks
- ▶ Query Plan
- ▶ Selectivity Estimation
- ▶ Spatio-Temporal Selectivity Estimation
- ▶ XML Selectivity Estimation

Recommended Reading

1. Blohsfeld B., Korus D., and Seeger B. A comparison of selectivity estimators for range queries on metric attributes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 239–250.
2. Chakrabarti K., Garofalakis M.N., Rastogi R., and Shim K. Approximate query processing using wavelets. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 111–122.
3. Chaudhuri S., Motwani R., and Narasayya V.R. On random sampling over joins. In Proc. ACM SIGMOD Int. Conf.

on Management of Data, Philadephia, PA, USA, June 1999, pp. 263–274.
4. Chen C.M. and Roussopoulos N. Adaptive selectivity estimation using query feedback. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1994, pp. 161–172.
5. Garcia-Molina H., Ullman J.D., and Widom J. Database Systems: The Complete Book. Prentice Hall, Englewood Cliffs, NJ, USA, 2002.
6. Gibbons P.B. and Matias Y. Synopsis data structures for massive data sets. In Proc. 10th Annual ACM-SIAM Symp. on Discrete Algorithms, 1999, pp. 909–910.
7. Gibbons P.B., Matias P.B., and Poosala V. Fast incremental maintenance of approximate histograms. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 466–475.
8. Haas P.J., Naughton J.F., Seshadri S., and Swami A.N. Selectivity and cost estimation for joins based on random sampling. J. Comput. Syst. Sci., 52(3):550–569, 1996.
9. Ioannidis Y.E. and Christodoulakis S. On the propagation of errors in the size of join results. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 268–277.
10. Ioannidis Y.E. and Poosala V. Histogram-based approximation of set-valued query-answers. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 174–185.
11. König A.C. and Weikum G. Combining histograms and parametric curve fitting for feedback-driven query result-size estimation. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 423–434.
12. König A.C. and Weikum G. Auto-tuned spline synopses for database statistics management. In Proc. Int. Conf. on Management of Data, 2000.
13. Korth H. and Silberschatz A. Database Systems Concepts. McGraw-Hill, Inc., New York, San Francisco, Washington, DC, USA, 1991.
14. Lu H., Tan K.L., and Shan M.C. Hash-based join algorithms for multiprocessor computers. In Proc. 16th Int. Conf. on Very Large Data Bases, 1990, pp. 198–209.
15. Manegold S. Understanding, Modeling, and Improving Main-Memory Database Performance. PhD Thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, December 2002.
16. Manegold S., Boncz P.A., and Kersten M.L. Generic database cost models for hierarchical memory systems. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 191–202.
17. Matias Y., Vitter J.S., and Wang M. Wavelet-based histograms for selectivity estimation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 448–459.
18. Selinger P.G., Astrahan M.M., Chamberlin D.D., Lorie R.A., and Price T.G. Access path selection in a relational database management system. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1979, pp. 23–34.
19. Spiliopoulou M. and Freytag J.-C. Modelling resource utilization in pipelined query execution. In Proc. European Conference on Parallel Processing, 1996, pp. 872–880.
20. Sun W., Ling Y., Rishe N., and Deng Y. An instant and accurate size estimation method for joins and selection in a retrieval-intensive environment. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 79–88.

Count-Min Sketch

Graham Cormode
AT&T Labs–Research, Florham Park, NJ, USA

C

Synonyms

CM Sketch

Definition

The Count-Min (CM) Sketch is a compact summary data structure capable of representing a high-dimensional vector and answering queries on this vector, in particular point queries and dot product queries, with strong accuracy guarantees. Such queries are at the core of many computations, so the structure can be used in order to answer a variety of other queries, such as frequent items (heavy hitters), quantile finding, join size estimation, and more. Since the data structure can easily process updates in the form of additions or subtractions to dimensions of the vector (which may correspond to insertions or deletions, or other transactions), it is capable of working over streams of updates, at high rates.

The data structure maintains the linear projection of the vector with a number of other random vectors. These vectors are defined implicitly by simple hash functions. Increasing the range of the hash functions increases the accuracy of the summary, and increasing the number of hash functions decreases the probability of a bad estimate. These tradeoffs are quantified precisely below. Because of this linearity, CM sketches can be scaled, added and subtracted, to produce summaries of the corresponding scaled and combined vectors.

Historical Background

The Count-Min sketch was first proposed in 2003 [5] as an alternative to several other sketch techniques, such as the Count sketch [3] and the AMS sketch [1]. The goal was to provide a simple sketch data structure with a precise characterization of the dependence on the input parameters. The sketch has also been viewed as a realization of a counting *Bloom filter* or Multistage-Filter [8], which requires only limited independence randomness to show strong, provable guarantees. The simplicity of creating and probing the sketch has led to its wide use in disparate areas since its initial description.

Foundations

The CM sketch is simply an array of counters of width w and depth d, $CM\,[1,\,1]\,...\,CM\,[d,\,w]$. Each entry of the array is initially zero. Additionally, d hash functions

$$h_1 ... h_d : \{1 ... n\} \rightarrow \{1 ... w\}$$

are chosen uniformly at random from a pairwise-independent family. Once w and d are chosen, the space required is fixed: the data structure is represented by wd counters and d hash functions (which can each be represented in $O(1)$ machine words [14]).

Update Procedure

Consider a vector a, which is presented in an implicit, incremental fashion (this abstract model captures a wide variety of data stream settings, see entries on *Data Stream Models* for more details). This vector has dimension n, and its current state at time t is $a(t) = [a_1(t),...a_i(t),...a_n(t)]$. Initially, $a(0)$ is the zero vector, 0, so $a_i(0)$ is 0 for all i. Updates to individual entries of the vector are presented as a stream of pairs. The tth update is (i_t, c_t), meaning that

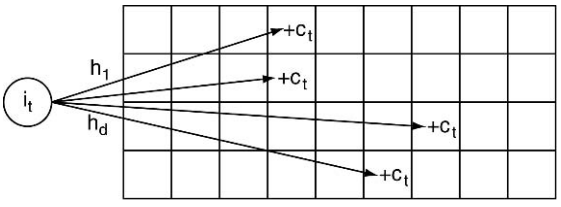
$$a_{i_t}(t) = a_{i_t}(t-1) + c_t$$
$$a_{i'}(t) = a_{i'}(t-1) \qquad i' \neq i_t$$

This procedure is illustrated in Fig. 1. In the remainder of this article, t is dropped, and the current state of the vector is referred to as just a for convenience. It is assumed throughout that although values of a_i increase and decrease with updates, each $a_i \geq 0$. The Count-Min sketch also applies to the case where a_is can be less than zero, with small factor increases in space. Here, details of these extensions are omitted for simplicity of exposition (full details are in [5]).

When an update (i_t, c_t) arrives, c_t is added to one count in each row of the Count-Min sketch; the counter is determined by h_j . Formally, given (i_t, c_t), the following modifications are performed:

$$\forall 1 \leq j \leq d : CM\,[j, h_j(i_t)] \leftarrow CM[j, h_j(i_t)] + c_t$$

Because computing each hash function takes $O(1)$ (constant) time, the total time to perform an update is $O(d)$, independent of w. Since d is typically small in practice (often less than 10), updates can be processed at high speed.



Count-Min Sketch. Figure 1. Each item i is mapped to one cell in each row of the array of counts: when an update of c_t to item i_t arrives, c_t is added to each of these cells.

Point Queries

A *point query* is to estimate the value of an entry in the vector a_i. The point query procedure is similar to updates: given a query point i, an estimate is found as $\hat{a}_i = \min_{1 \leq j \leq d} CM[j, h_j(i)]$. Since the space used by the sketch is typically much smaller than that required to represent the vector exactly, there is necessarily some approximation in the estimate, which is quantified as follows:

Theorem 1 (Theorem 1 from [5]). *If* $w = \lceil \frac{e}{\varepsilon} \rceil$ *and* $d = \lceil \ln \frac{1}{\delta} \rceil$, *the estimate* $\hat{a}_i$ *has the following guarantees:* $a_i \leq \hat{a}_i$; *and, with probability at least* $1 - \delta$,

$$\hat{a}_i \leq a_i + \varepsilon \|a\|_1 .$$

The proof follows by considering the estimate in each row, and observing that the expected error in using $CM\,[j,\,h_j(i)]$ as an estimate has expected (non-negative) error $\|a\|_1 / w$. By the Markov inequality [14], the probability that this error exceeds $\epsilon \|a\|_1$ is at most $\frac{1}{e}$ (where e is the base of the natural logarithm, i.e., 2.71828 . . ., a constant chosen to optimize the space for fixed accuracy requirements). Taking the smallest estimate gives the best estimator, and the probability that this estimate has error exceeding $\epsilon \|a\|_1$ is the probability that *all* estimates exceed this error, i.e., $e^{-d} \leq \delta$.

This analysis makes no assumption about the distribution of values in a. However, in many applications there are Zipfian, or power law, distributions of item frequencies. Here, the (relative) frequency of the ith most frequent item is proportional to i^{-z}, for some parameter z, where z is typically in the range 1–3 ($z = 0$ gives a perfectly uniform distribution). In such cases, the skew in the distribution can be used to show a stronger space/accuracy tradeoff:

Theorem 2 (Theorem 5.1 from [7]). *For a Zipf distribution with parameter z, the space required to*

answer point queries with error $\epsilon\|a\|_1$ *with probability at least* $1-\delta$ *is given by* $O(\varepsilon^{-\min\{1,1/z\}}\ln 1/\delta)$.

Moreover, the dependency of the space on z is optimal:

Theorem 3 (Theorem 5.2 from [7]). *The space required to answer point queries correctly with any constant probability and error at most* $\epsilon\|a\|_1$ *is* $\Omega(\varepsilon^{-1})$ *over general distributions, and* $\Omega(\varepsilon^{-1/z})$ *for Zipf distributions with parameter z, assuming the dimension of a, n is* $\Omega(\varepsilon^{-\min\{1,1/z\}})$.

Range, Heavy Hitter and Quantile Queries

A range query is to estimate $\sum_{i=l}^{r} a_i$ for a range $[l...r]$. For small ranges, the range sum can be estimated as a sum of point queries; however, as the range grows, the error in this approach also grows linearly. Instead, log$n$ sketches can be kept, each of which summarizes a derived vector a^k where

$$a^k[j] = \sum_{i=j2^k}^{(j+1)2^k-1} a_i$$

for $k = 1...\log n$. A range of the form $j2^k...(j+1)2^k - 1$ is called a *dyadic range*, and any arbitrary range $[l...r]$ can be partitioned into at most 2log n dyadic ranges. With appropriate rescaling of accuracy bounds, it follows that:

Theorem 4 (Theorem 4 from [5]). *Count-Min sketches can be used to find an estimate* $\hat{r}$ *for a range query on l...r such that*

$$\hat{r} - \epsilon\|a\|_1 \leq \sum_{i=l}^{r} a_i \leq \hat{r}$$

The right inequality holds with certainty, and the left inequality holds with probability at least $1-\delta$. *The total space required is* $O(\frac{\log^2 n}{\epsilon}\log\frac{1}{\delta})$.

Closely related to the range query is the ϕ-*quantile query*, which is to find a point j such that

$$\sum_{i=1}^{j} a_i \leq \phi\|a\|_1 \leq \sum_{i=1}^{j+1} a_i.$$

A natural approach is to use range queries to binary search for a j which satisfies this requirement approximately (i.e., tolerates up to $\epsilon\|a\|_1$ error in the above expression) given ϕ. In order to give the desired guarantees, the error bounds need to be adjusted to account for the number of queries that will be made.

Theorem 5 (Theorem 5 from [5]). ε-*approximate* ϕ-*quantiles can be found with probability at least* $1-\delta$ *by keeping a data structure with space* $O\left(\frac{1}{\epsilon}\log^2(n)\log\left(\frac{\log n}{\delta}\right)\right)$. *The time for each insert or delete operation is* $O\left(\log(n)\log\left(\frac{\log n}{\delta}\right)\right)$, *and the time to find each quantile on demand is* $O\left(\log(n)\log\left(\frac{\log n}{\delta}\right)\right)$.

Heavy Hitters are those points i such that $a_i \geq \phi\|a\|_1$ for some specified ϕ. The range query primitive based on Count-Min sketches can again be used to find heavy hitters, by recursively splitting dyadic ranges into two and querying each half to see if the range is still heavy, until a range of a single, heavy, item is found. Formally,

Theorem 6 (Theorem 6 from [5]). *Using space* $O\left(\frac{1}{\epsilon}\log(n)\log\left(\frac{2\log(n)}{\delta\phi}\right)\right)$, *and time* $O\left(\log(n)\log\left(\frac{2\log n}{\delta\phi}\right)\right)$ *per update, a set of approximate heavy hitters can be output so that every item with frequency at least* $(\phi+\epsilon)\|a\|_1$ *is output, and with probabilitye* $1-\delta$ *no item whose frequency is less than* $\phi\|a\|_1$ *is output.*

For skewed Zipfian distributions, as described above, with parameter $z > 1$, it is shown more strongly that the top-k most frequent items can be found with relative error ε using space only $\tilde{O}(\frac{k}{\epsilon})$ [7].

Inner Product Queries

The Count-Min sketch can also be used to estimate the inner product between two vectors; in database terms, this captures the (equi)join size between relations. The inner product $a \cdot b$, can be estimated by treating the Count-Min sketch as a collection of d vectors of length w, and finding the minimum inner product between corresponding rows of sketches of the two vectors. With probability $1-\delta$, this estimate is at most an additive quantity $\epsilon\|a\|_1\|b\|_1$ above the true value of $a \cdot b$. This is to be compared with AMS sketches which guarantee $\epsilon\|a\|_2\|b\|_2$ additive error, but require space proportional to $\frac{1}{\epsilon^2}$ to make this guarantee.

Interpretation as Random Linear Projection

The sketch can also be interpreted as a collection of inner-products between a vector representing the input and a collection of random vectors defined by the hash functions. Let a denote the vector representing the input, so that $a[i]$ is the sum of the updates to the ith location in the input. Let $r_{j,k}$ be the binary vector such

that $r_{j,k}[i] = 1$ if and only if $h_j(i) = k$. Then it follows that $CM[j,k] = a \cdot r_{j,k}$. Because of this linearity, it follows immediately that if sketches of two vectors, a and b, are built then (i) the sketch of $a + b$ (using the same w,d,h_j) is the (componentwise) sum of the sketches (ii) the sketch of λa for any scalar λ is λ times the sketch of a. In other words, the sketch of any linear combination of vectors can be found. This property is useful in many applications which use sketches. For example, it allows distributed measurements to be taken, sketched, and combined by only sending sketches instead of the whole data.

Conservative Update

If only positive updates arrive, then an alternate update methodology may be applied, known as conservative update (due to Estan and Varghese [8]). For an update (i,c), $\hat{a}_i$ is computed, and the counts are modified according to $\forall 1 \leq j \leq d : CM[j, h_j(i)] \leftarrow \max(CM[j, h_j(i)], \hat{a}_i + c)$. It can be verified that procedure still ensures for point queries that $a_i \leq \hat{a}_i$, and that the error is no worse than in the normal update procedure; it is remarked that this can improve accuracy "up to an order of magnitude" [8]. Note however that deletions or negative updates can no longer be processed, and the additional processing that must be performed for each update could effectively halve the throughput.

Key Applications

Since its description and initial analysis, the Count-Min Sketch has been applied in a wide variety of situations. Here is a list of some of the ways in which it has been used or modified.

- Lee et al. [13] propose using least-squares optimization to produce estimates from Count-Min Sketches for point queries (instead of returning the minimum of locations where the item was mapped). It was shown that this approach can give significantly improved estimates, although at the cost of solving a convex optimization problem over n variables (where n is the size of the domain from which items are drawn, typically 2^{32} or higher).
- The "skipping" technique, proposed by Bhattacharrya et al. [2] entails avoiding adding items to the sketch (and saving the cost of the hash function computations) when this will not affect the accuracy too much, in order to further increase throughout in high-demand settings.
- Indyk [9] uses the Count-Min Sketch to estimate the residual mass after removing a subset of items. That is, given a (small) set of indices I, to estimate $\sum_{i \notin I} a_i$. This is needed in order to find clusterings of streaming data.
- The *entropy* of a data stream is a function of the relative frequencies of each item or character within the stream. Using Count-Min Sketches within a larger data structure based on additional hashing techniques, Lakshminath and Ganguly [12] showed how to estimate this entropy to within relative error.
- Sarlós et al. [17] gave approximate algorithms for personalized page rank computations which make use of Count-Min Sketches to compactly represent web-size graphs.
- In describing a system for building selectivity estimates for complex queries, Spiegel and Polyzotis [18] use Count-Min Sketches in order to allow clustering over a high-dimensional space.
- Rusu and Dobra [16] study a variety of sketches for the problem of inner-product estimation, and conclude that Count-Min sketch has a tendency to outperform its theoretical worst-case bounds by a considerable margin, and gives better results than some other sketches for this problem.
- Many applications call for tracking *distinct* counts: that is, a_i should represent the number of distinct updates to position i. This can be achieved by replacing the counters in the Count-Min sketch with approximate Count-Distinct summaries, such as the *Flajolet-Martin sketch*. This is described and evaluated in [6,10].
- Privacy preserving computations ensure that multiple parties can cooperate to compute a function of their data while only learning the answer and not anything about the inputs of the other participants. Roughan and Zhang demonstrate that the Count-Min Sketch can be used within such computations, by applying standard techniques for computing privacy preserving sums on each counter independently [15].

Related ideas to the Count-Min Sketch have also been combined with group testing to solve problems in the realm of Compressed Sensing, and finding significant changes in dynamic streams.

Future Directions

As is clear from the range of variety of applications described above, Count-Min sketch is a versatile data structure which is finding applications within Data Stream systems, but also in Sensor Networks, Matrix Algorithms, Computational Geometry and Privacy-Preserving Computations. It is helpful to think of the structure as a basic primitive which can be applied wherever approximate entries from high dimensional vectors or multisets are required, and one-sided error proportional to a small fraction of the total mass can be tolerated (just as a Bloom filter should be considered in order to represent a set wherever a list or set is used and space is at a premium). With this in mind, further applications of this synopsis can be expected to be seen in more settings.

As noted below, sample implementations are freely available in a variety of languages, and integration into standard libraries will further widen the availability of the structure. Further, since many of the applications are within high-speed data stream monitoring, it is natural to look to hardware implementations of the sketch. In particular, it will be of interest to understand how modern multi-core architectures can take advantage of the natural parallelism inherent in the Count-Min Sketch (since each of the d rows are essentially independent), and to explore the implementation choices that follow.

Experimental Results

Experiments performed in [7] analyzed the error for point queries and F_2 (self-join size) estimation, in comparison to other sketches. High accuracy was observed for both queries, for sketches ranging from a few kilobytes to a megabyte in size. The typical parameters of the sketch were a depth d of 5, and a width w of a few hundred to thousands. Implementations on desktop machines achieved between and two and three million updates per second. Other implementation have incorporated Count-Min Sketch into high speed streaming systems such as Gigascope [4], and tuned it to process packet streams of multi-gigabit speeds.

Lai and Byrd report on an implementation of Count-Min sketches on a low-power stream processor [18], capable of processing 40 byte packets at a throughput rate of up to 13 Gbps. This is equivalent to about 44 million updates per second.

URL To Code

Several example implementations of the Count-Min sketch are available. C code is given by the MassDal code bank: http://www.cs.rutgers.edu/~muthu/massdal-code-index.html. C++ code due to Marios Hadjieleftheriou is available from http://research.att.com/~marioh/sketches/index.html.

Cross-references

- ▶ AMS Sketch
- ▶ Data sketch/synopsis
- ▶ FM Synopsis
- ▶ Frequent items on streams
- ▶ Quantiles on streams

Recommended Reading

1. Alon N., Matias Y., and Szegedy M. The space complexity of approximating the frequency moments. In Proc. 28th Annual ACM Symp. on Theory of Computing, 1996, pp. 20–29. Journal version in J. Comput. Syst. Sci., 58:137–147, 1999.
2. Bhattacharrya S., Madeira A., Muthukrishnan S., and Ye T. How to scalably skip past streams. In Proc. 1st Int. Workshop on Scalable Stream Processing Syst., 2007, pp. 654–663.
3. Charikar M., Chen K., and Farach-Colton M. Finding frequent items in data streams. In 29th Int. Colloquium on Automata, Languages, and Programming, 2002, pp. 693–703.
4. Cormode G., Korn F., Muthukrishnan S., Johnson T., Spatscheck O., and Srivastava D. Holistic UDAFs at streaming speeds. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 35–46.
5. Cormode G. and Muthukrishnan S. An improved data stream summary: the count-min sketch and its applications. J. Algorithms, 55(1):58–75, 2005.
6. Cormode G. and Muthukrishnan S. Space efficient mining of multigraph streams. In Proc. 24th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2005, pp. 271–282.
7. Cormode G. and Muthukrishnan S. Summarizing and mining skewed data streams. In Proc. SIAM International Conference on Data Mining, 2005.
8. Estan C. and Varghese G. New directions in traffic measurement and accounting. In Proc. ACM Int. Conf. of the on Data Communication, 2002, pp. 323–338.
9. Indyk P. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2003.
10. Kollios G., Byers J., Considine J., Hadjieleftheriou M., and Li F. Robust aggregation in sensor networks. Q. Bull. IEEE TC on Data Engineering, 28(1):26–32, 2005.
11. Lai Y.-K. and Byrd G.T. High-throughput sketch update on a low-power stream processor. In Proc. ACM/IEEE Symp. on Architecture for Networking and Communications Systems, 2006, pp. 123–132.

12. Lakshminath B. and Ganguly S. Estimating entropy over data streams. In Proc. 14th European Symposium on Algorithms, 2006, pp. 148–159.
13. Lee G.M., Liu H., Yoon Y., and Zhang Y. Improving sketch reconstruction accuracy using linear least squares method. In Proc. 5th ACM SIGCOMM Conf. on Internet Measurement, 2005, pp. 273–278.
14. Motwani R. and Raghavan P. Randomized Algorithms. Cambridge University Press, 1995.
15. Roughan M. and Zhang Y. Secure distributed data mining and its application in large-scale network measurements. Computer Communication Review, 36(1):7–14, 2006.
16. Rusu F. and Dobra A. Statistical analysis of sketch estimators. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 187–198.
17. Sarlós T., Benzúr A., Csalogány K., Fogaras D., and Rácz B. To randomize or not to randomize: space optimal summaries for hyperlink analysis. In Proc. 15th Int. World Wide Web Conference, 2006, pp. 297–306.
18. Spiegel J. and Polyzotis N. Graph-based synopses for relational selectivity estimation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 205–216.

Coupling and De-coupling

Serguei Mankovskii
CA Labs, CA, Inc., Thornhill, ON, Canada

Definition

Coupling is a measure of dependence between components of software system.

De-coupling is a design or re-engineering activity aiming to reduce coupling between system elements.

Key Points

Coupling of system components refers to a measure of dependency among them. Coupled components might depend on each other in different ways. Some examples of the dependencies are:

- One component might depend on syntax, format, or encoding of data produced by another component.
- One component might depend on the execution time within another component.
- One component might depend on state of another component.

Notion of coupling is connected to notion of cohesion. Cohesion is a measure of how related and focused are responsibilities of a software component. For example a highly cohesive component might group responsibilities

- using the same syntax, format or encoding of data.
- performed at the same time.
- executed in the same state.

Highly cohesive components lead to fewer dependencies between components and voice versa.

Notions of coupling and cohesion were studied in structured and object oriented programming. The research developed software tools to calculate coupling and cohesion metrics.

Low coupling is often desirable because it leads to reliability, easy of modification, low maintenance costs, understandability, and reusability. Low coupling can be achieved by deliberately designing system with low values of coupling metric. It can also be achieved by re-engineering of existing software system through restructuring of system into a set of more cohesive components. These activates are called de-coupling.

Cross-references

► Cohesion
► OODB (Object-Oriented Database)
► Structured Programming

Coverage

► Specificity

Covering Index

Donghui Zhang
Northeastern University, Boston, MA, USA

Definition

Given an SQL query, a covering index is a composite index that includes all of the columns referenced in SELECT, JOIN, and WHERE clauses of this query. Because the index contains all the data needed by the query, to execute the query the actual data in the table does not need to be accessed.

Key Points

Covering indexes [1] support index-only execution plans. In general, having everything indexed tends to increase the query performance (in number of I/Os). However, using a covering index with too many columns

can actually degrade performance. Typically, multi-dimensional index structures, e.g., the R-tree, perform poorer than linear scan with high dimensions. Some guidelines of creating a covering index are: (i) Create a covering index on frequently used queries. There are overheads in creating a covering index, which is often more significant than creating a regular index with fewer columns. Hence, if a query is seldom used, the overhead to create a covering index on it is not substantiated. This corresponds to Amdahl's law: improve the "interesting" part to receive maximum overall benefit of a system. (ii) Try to build a covering index by expanding an existing index. For instance, if there already exists an index on "age" and "salary," and one needs a covering index on "age," "salary," and "income," it is often better to expand the existing index rather than building a new index, which would share two columns with the existing index.

The term "covering index" is sometimes used to mean the collection of single-column, non-clustered indexes on all the columns in a table. This is due to the "index intersection" technique incorporated into the Microsoft SQL Server's query optimizer [1]. In particular, the query optimizer can build, at run time, a hash-based "covering index" to speedup queries on a frequently used table. This covering index is really a hash table, which is built based on multiple existing indexes. Creating single-column indexes on all columns encourages the query optimizer to perform index intersection, i.e., to build dynamic covering indexes.

Cross-references

► Access Methods
► Indexing

Recommended Reading

1. McGehee B. Tips on Optimizing Covering Indexes. http://www.sql-server-performance.com/tips/covering_indexes_p1.aspx, 2007.

Covert Communication

► Steganography

CPU Cache

► Processor Cache

Crabbing

► B-Tree Locking

Crash Recovery

Theo Härder
University of Kaiserslautern, Kaiserslautern, Germany

Synonyms

Failure handling; System recovery; Media recovery; Online recovery; Restart processing; Backward recovery

Definition

In contrast to transaction aborts, a crash is typically a major failure by which the state of the current database is lost or parts of storage media are unrecoverable (destroyed). Based on log data from a stable log, also called temporary log file, and the inconsistent and/or outdated state of the permanent database, system recovery has to reconstruct the most recent transaction-consistent database state. Because DBMS restart may take too long to be masked for the user, a denial of service can be observed. Recovery from media failures relies on the availability of (several) backup or archive copies of earlier DB states – organized according to the generation principle – and archive logs (often duplexed) covering the processing intervals from the points of time the backup copies were created. Archive recovery usually causes much longer outages than system recovery.

Historical Background

Log data delivering the needed redundancy to recover from failures was initially stored on nonvolatile core memory to be reclaimed at restart by a so-called log salvager [3] in the "pre-transaction area". Advances in VLSI technology enabled the use of cheaper and larger, but volatile semiconductor memory as the computers' main memory. This technology change triggered by 1971 in industry – driven by database product adjustments – the development of new and refined concepts of logging such as log sequence numbers (LSNs), write-ahead log protocol (WAL), log duplexing and more. Typically, these concepts were not published, nevertheless they paved the way towards the use

of ACID transactions. As late as 1978, Jim Gray documented the design of such a logging system implemented in IMS in a widely referenced publication [5].

Many situations and dependencies related to failures and recovery from those in databases have been thoroughly explored by Lawrence Bjork and Charles Davies in their studies concerning DB/DC systems back in 1973 leading to the so-called "spheres of control" [2]. The first published implementation of the transaction concept by a full-fledged DBMS recovery manager was that of System R, started in 1976 [4]. It refined the Do-Undo-Redo protocol and enabled automatic recovery for new recoverable types and operations. In 1981, Andreas Reuter presented in his Ph.D. dissertation further investigations and refinements of concepts related to failure handling in database systems [9]. Delivering a first version of the principles of transaction-oriented database recovery [Härder and Reuter 1979], including the *Ten Commandments* [6], this classification framework, defining the paradigm of transaction-oriented recovery and coining the acronym ACID for it [7], was finally published in 1983. The most famous and most complete description of recovery methods and their implementation was presented by C. Mohan et al. in the ARIES paper [8] in 1992, while thorough treatment of all questions related to this topic appeared in many textbooks, especially those of Bernstein et al. [1], Gray and Reuter [3], and Weikum and Vossen [11]. All solutions implemented for crash recovery in industrial-strength DBMSs are primarily disk-based. Proposals to use "safe RAM", for example, were not widely accepted.

Foundations

The most difficult failure type to be recovered from is the system failure or system crash (see Logging and Recovery). Due to some (expected, but) unplanned failure event (a bug in the DBMS code, an operating system fault, a power or hardware failure, etc.), the *current database* – comprising all objects accessible to the DBMS during normal processing – is not available anymore. In particular, the in-memory state of the DBMS (lock tables, cursors and scan indicators, status of all active transactions, etc.) and the contents of the database buffer and the log buffer are lost. Furthermore, the state lost may include information about LSNs, ongoing commit processing with participating coordinators and participants as well as commit requests and votes. Therefore, restart cannot rely on such information and has to refer to the temporary log file (stable log) and the *permanent (materialized) database*, that is, the state the DBMS finds after a crash at the non-volatile storage devices (disks) without having applied any log information.

Consistency Concerns

According to the *ACID principle*, a database is consistent if and only if it contains the results of successful transactions – called transaction-consistent database. Because a DBMS application must not lose changes of committed transactions and all of them have contributed to the DB state, the goal of crash recovery is to establish the most recent transaction-consistent DB state. For this purpose, redo and undo recovery is needed, in general. Results of committed transactions may not yet be reflected in the database, because execution has been terminated in an uncontrolled manner and the corresponding pages containing such results were not propagated to the permanent DB at the time of the crash. Therefore, they must be repeated, if necessary – typically by means of log information. On the other hand, changes of incomplete transactions may have reached the permanent DB state on disk. Hence, undo recovery has to completely roll back such uncommitted changes.

Because usually many interactive users rely in their daily business on DBMS services, crash recovery is very time-critical. Therefore, crash-related interruption of DBMS processing should be masked for them as far as possible. Although today DBMS crashes are rather rare events and may occur several times a month or a year – depending on the stability of both the DBMS and its operational environment – , their recovery should take no more than a number of seconds or at most a few minutes (as opposed to archive recovery), even if GByte or TByte databases with thousands of users are involved.

Forward Recovery

Having these constraints and requirements in mind, which kind of recovery strategies can be applied? Despite the presence of so-called non-stop systems (giving the impression that they can cope with failures by forward recovery), rollforward is very difficult, if not impossible in any stateful system. To guarantee atomicity in case of a crash, rollforward recovery had to enable all transactions to resume execution so that they can either complete successfully or require to be

aborted by the DBMS. Assume the DB state containing the most recent successful DB operations could be made available, that is, all updates prior to the crash have completely reached the permanent DB state. Even then rollforward would be not possible, because a transaction cannot resume in "forward direction" unless its local state is restored. Moreover in a DBMS environment, the in-memory state lost makes it entirely impossible to resume from the point at the time the crash occurred. For these reasons, a rollback strategy for active transactions is the only choice in case of crash recovery to ensure atomicity (wiping out all traces of such transactions); later on these transactions are started anew either by the user or the DBMS environment. The only opportunities for forward actions are given by redundant structures where it is immaterial for the logical DB content whether or not modifying operations are undone or completed. A typical example is the splitting operation of a B-tree node.

Logging Methods and Rules

Crash recovery – as any recovery from a failure – needs some kind of redundancy to detect invalid or missing data in the permanent database and to "repair" its state as required, i.e., removing modifications effected by uncommitted transactions from it and supplementing it with updates of complete transactions. For this task, the recovery algorithms typically rely on log data collected during normal processing. Different forms of logging are conceivable. *Logical logging* is a kind of operator logging; it collects operators and their arguments at a higher level of abstraction (e.g., for internal operations (actions) or operations of the data management language (DML)). While this method of logging may save I/O to and space in the log file during normal processing, it requires at restart time a DB state that is level-consistent w.r.t. the level of abstraction used for logging, because the logged operations have to be executed using data of the permanent database. For example, action logging and DML-operation logging require action consistency and consistency at the application programming interface (API consistency), respectively [6]. Hence, the use of this kind of methods implies the atomic propagation (see below) of all pages modified by the corresponding operation which can be implemented by shadow pages or differential files. *Physical logging* – in the simplest form collecting the before- and after-images of pages – does not expect any form of consistency at higher DB abstraction levels and, in turn, can be used in any situation, in particular, when non-atomic propagation of modified pages (*update-in-place*) is performed. However, writing before- and after-images of all modified pages to the log file, is very time-consuming (I/O) and not space-economical at all. Therefore, a combination of both kinds leads to the so-called *physiological logging*, which can be roughly characterized as "physical to a page and logical within a page". It enables compact representation of log data (logging of elementary actions confined to single pages, entry logging) and leads to the practically most important logging/recovery method; non-atomic propagation of pages to disk is sufficient for the application of the log data. Together with the use of *log sequence numbers* in the log entries and in the headers of the data pages (combined use of LSNs and PageLSNs, see ARIES Protocol), simple and efficient checks at restart detect whether or not the modifications of elementary actions have reached the permanent database, that is, whether or not undo or redo operations have to be applied.

While, in principle, crash recovery methods do not have specific requirements for forcing pages to the permanent DB, sufficient log information, however, must have reached the stable log. The following rules (for forcing of the log buffer to disk) have to be observed to guarantee recovery to the most recent transaction-consistent DB state:

- Redo log information must be written at the latest in phase 1 of commit.
- WAL (*write ahead logging*) has to be applied to enable undo operations, before uncommitted (dirty) data is propagated to the permanent database.
- Log information must not be discarded from the temporary log file, unless it is guaranteed that it will no longer be needed for recovery; that is, the corresponding data page has reached the permanent DB. Typically, sufficient log information has been written to the archive log, in addition.

Taxonomy of Crash Recovery Algorithms

Forcing log data as captured by these rules yields the necessary and sufficient condition to successfully cope with system crashes. Specific assumptions concerning page propagation to the permanent database only influence performance issues of the recovery process.

When dirty data can reach the permanent DB (steal property), recovery must be prepared to execute undo steps and, in turn, redo steps when data modified by a transaction is not forced at commit or before (no-force property). In contrast, if propagation of dirty data is prevented (no-steal property), the permanent DB only contains clean (but potentially missing or old) data, thus making undo steps unnecessary. Finally, if all transaction modifications are forced at commit (force property), redo is never needed at restart.

Hence, these properties concerning buffer replacement and update propagation are maintained by the buffer manager/transaction manager during normal processing and lead to four cases of crash recovery algorithms which cover all approaches so far proposed:

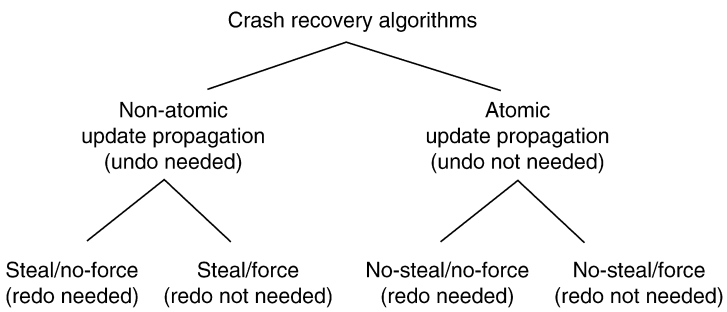
1. *Undo/Redo:* This class contains the *steal/no-force* algorithms which have to observe no other requirements than the logging rules. However, potentially undo and redo steps have to be performed during restart after a crash.
2. *Undo/NoRedo:* The so-called *steal/force* algorithms guarantee at any time that all actions of committed transactions are in the permanent DB. However, because of the steal property, dirty updates may be present, which may require undo steps, but never redo steps during restart.
3. *NoUndo/Redo:* The corresponding class members are known as *no-steal/no-force* algorithms which guarantee that dirty data never reaches the permanent DB. Dirty data pages are either never replaced from the DB buffer or, in case buffer space is in short supply, they are displaced to other storage areas outside the permanent DB. Restart after a crash may require redo steps, but never undo steps.
4. *NoUndo/NoRedo:* This "magic" class of the so-called *no-steal/force* algorithms always guarantees a state of the permanent DB that corresponds to the most recent transaction-consistent DB state. It requires that no modified data of a transaction reaches the permanent DB before commit and that all transaction updates are atomically propagated (forced) at commit. Hence, neither undo nor redo steps are ever needed during restart.

The discussion of these four cases is summarized in Fig. 1 which represents a taxonomy of crash recovery algorithms.

Implementation Implications

The latter two classes of algorithms (NoUndo) require a mechanism which can propagate a set of pages in an atomic way (with regard to the remaining DBMS processing). Such a mechanism needs to defer updates to the permanent DB until or after these updates become committed and can be implemented by various forms of shadowing concepts or differential file approaches.

Algorithms relying on redo steps, i.e., without the need to force committed updates to the permanent DB, have no control about the point of time when committed updates reach the permanent DB. While the buffer manager will propagate back most of the modified pages soon after the related update operations, a few hot-spot pages are modified again and again, and, since they are referenced so frequently, have not been written from the buffer. These pages potentially have accumulated the updates of many committed transactions, and redo recovery will therefore have to go back very far on the temporary log. As a consequence, restart becomes expensive and the DBMS's out-of-service time unacceptably long. For this reason, some form of *checkpointing* is needed to make restart costs independent of mean time between failures. Generating a checkpoint means collecting



Crash Recovery. Figure 1. Taxonomy of crash recovery algorithms.

information related to the DB state in a safe place, which is used to define and limit the amount of redo steps required after a crash. The restart logic can then return to this checkpoint state and attempt to recover the most recent transaction-consistent state.

From a conceptual point of view, the algorithms of class 4 seem to be particularly attractive, because they always preserve a transaction-consistent permanent DB. However in addition to the substantial cost of providing atomic update propagation, the need of forcing all updates at commit, necessarily in a synchronous way which may require a large amount of physical I/Os and, in turn, extend the lock duration for all affected objects, makes this approach rather expensive. Furthermore, with the typical disk-based DB architectures, pages are units of update propagation, which has the consequence that a transaction updating a record in a page cannot share this page with other updaters, because dirty updates must not leave the buffer and updates of complete transactions must be propagated to the permanent DB at commit. Hence, no-steal/force algorithms imply at least *page locking* as the smallest lock granule.

One of these cost factors – either synchronously forced updates at commit or atomic updates for NoUndo – applies to the algorithms of class 2 and 3 each. Therefore, they were not a primary choice for the DBMS vendors competing in the today's market.

Hence, the laissez-faire solution "steal, no-force" with non-atomic update propagation (update-in-place) is today's favorite solution, although it always leaves the permanent DB in a "chaotic state" containing dirty and outdated data pages and keeping the latest version of frequently used pages only in the DB buffer. Hence, with the optimistic expectation that crashes become rather rare events, it minimizes recovery provisions during normal processing. Checkpointing is necessary, but the application of direct checkpoints flushing the entire buffer at a time, is not advisable anymore, when buffers of several GByte are used. To affect normal processing as little as possible, so-called *fuzzy checkpoints* are written; only a few pages with metadata concerning the DB buffer state have to be synchronously propagated, while data pages are "gently" moved to the permanent DB in an asynchronous way.

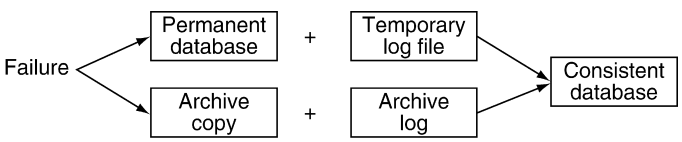
Archive Recovery

So far, data of the permanent DB was assumed to be usable or at least recoverable using the redundant data collected in the temporary log. This is illustrated by the upper path in Fig. 2. If any of the participating components is corrupted or lost because of other hardware or software failure, archive recovery – characterized by the lower path – must be tried. Successful recovery also implies independent failure modes of the components involved.

The creation of an archive copy, that is, copying the online version of the DB, is a very expensive process; for example, creating a transaction-consistent DB copy would interrupt update operation for a long time which is unacceptable for most DB applications. Therefore, two base methods – fuzzy dumping and incremental dumping – were developed to reduce the burden of normal DB operation while an archive copy is created. A fuzzy dump copies the DB on the fly in parallel with normal processing. The other method writes only the changed pages to the incremental dump. Of course, both methods usually deliver inconsistent DB copies such that log-based post-processing is needed to apply incremental modifications. In a similar way, either type of dump can be used to create a new, more up-to-date copy from the previous one, using a separate offline process such that DB operation is not affected.

Archive copies are "hopefully" never or very infrequently used. Therefore, they may be susceptible to magnetic decay. For this reason, redundancy is needed again, which is usually solved by keeping several generations of the archive copy.

So far, all log information was assumed to be written only to the temporary log file during normal processing. To create the (often duplexed) archive log, usually an independent and asynchronously running process copies the redo data from the temporary log. To guarantee successful recovery, failures when using



Crash Recovery. Figure 2. Two ways of DB crash recovery and the components involved.

the archive copies must be anticipated. Therefore, archive recovery must be prepared to start from the oldest generation and hence the archive log must span the whole distance back to this point in time.

Key Applications

Recovery algorithms, and in particular for crash recovery, are a core part of each commercial-strength DBMS and require a substantial fraction of design/implementation effort and of the code base: "A recoverable action is 30% harder and requires 20% more code than a non-recoverable action" (J. Gray). Because the occurrence of failures can not be excluded and all data driving the daily business are managed in databases, mission-critical businesses depend on the recoverability of their data. In this sense, provisions for crash recovery are indispensable in such DBMS-based applications. Another important application area of crash recovery techniques are file systems, in particular their metadata about file existence, space allocation, etc.

Future Directions

So far, crash recovery provisions are primarily disk-based. With "unlimited" memory available, main-memory DBMSs will provide efficient and robust solutions without the need of non-volatile storage for crash recovery. More and more approaches are expected to exploit specialized storage devices such as battery-backed RAM or to use replication in grid-organized memories. Executing online transaction processing sequentially, revolutionary architectural concepts are already proposed which may not require transactional facilities at all [10].

Cross-references

- ▶ ACID Properties
- ▶ Application Recovery
- ▶ B-Tree Locking
- ▶ Buffer Management
- ▶ Logging and Recovery
- ▶ Multi-Level Recovery and the ARIES Algorithm

Recommended Reading

1. Bernstein P.A., Hadzilacos V., and Goodman N. Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading, MA, 1987.
2. Davies C.T. Data processing spheres of control. IBM Syst. J., 17(2):179–198, 1978.
3. Gray H. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA, 1993.
4. Gray J., McJones P., Blasgen M., Lindsay B., Lorie R., Price T., Putzolu F., and Traiger I.L. The recovery manager of the System R database manager. ACM Comput. Surv., 13(2): 223–242, 1981.
5. Gray J, Michael J. Feynn, Jim Gray, Anita K. Jones, Klans Lagally, Holger Opderbeck, Gerald J. Popek, Brian Randell, Jerome H. Saltfer, Hans-Rüdiger Wiehle. Notes on database operating systems. In Operating Systems: An Advanced Course. Springer, LNCS 60, 1978, pp. 393–481.
6. Härder T. DBMS Architecture – Still an Open Problem. In Proc. German National Database Conference, 2005, pp. 2–28.
7. Härder T. and Reuter A. Principles of transaction-oriented database recovery. ACM Comput. Surv., 15(4): 287–317, 1983.
8. Mohan C., Haderle D.J., Lindsay B.G., Pirahesh H., and Schwarz P.M. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst., 17(1):94–162, 1992.
9. Reuter A. Fehlerbehandlung in Datenbanksystemen. Carl Hanser, Munich, 1981, p. 456.
10. Stonebraker M., Madden S., Abadi D.J., Harizopoulos S., Hachem N., and Helland P. The End of an Architectural Era (It's Time for a Complete Rewrite). In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 1150–1160.
11. Weikum G. and Vossen G. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann, San Francisco, CA, 2002.

Crawler

▶ Incremental Crawling

Credulous Reasoning

▶ Possible Answers

Cross Product

▶ Cartesian Product

Cross-language Cross-Language Mining and Retrieval C217 Informational Retrieval

▶ Cross-Language Mining and Retrieval

Cross-Language Mining and Retrieval

Wei Gao[1], Cheng Niu[2]
[1]The Chinese University of Hong Kong, Hong Kong, China
[2]Microsoft Research Asia, Beijing, China

Synonyms

Cross-language text mining; Cross-language web mining; Cross-language informational retrieval; Translingual information retrieval

Definition

Cross-language mining is a task of text mining dealing with the extraction of entities and their counterparts expressed in different languages. The interested entities may be of various granularities from acronyms, synonyms, cognates, proper names to comparable or parallel corpora. Cross-Language Information Retrieval (CLIR) is a sub-field of information retrieval dealing with the retrieval of documents across language boundaries, i.e., the language of the retrieved documents is not the same as the language of the queries. Cross-language mining usually acts as an effective means to improve the performance of CLIR by complementing the translation resources exploited by CLIR systems.

Historical Background

CLIR addresses the growing demand to access large volumes of documents across language barriers. Unlike monolingual information retrieval, CLIR requires query terms in one language to be matched with the indexed keywords in the documents of another language. Usually, the cross-language matching can be done by making use of bilingual dictionary, machine translation software, or statistical model for bilingual words association. CLIR generally takes into account but not limited to the issues like how to translate query terms, how to deal with the query terms nonexistent in a translation resource, and how to disambiguate or weight alternative translations (e.g., to decide that "traitement" in a French query means "treatment" but not "salary" in English, or how to order the French terms "aventure," "business," "affaire," and "liaison" as relevant translations of English query "affair"), etc. The performance of CLIR can be measured by the general evaluation metrics of information retrieval, such as recall precision, average precision, and mean reciprocal rank, etc.

The first workshop on CLIR was held in Zürich during the SIGIR-96 conference. Workshops have been held yearly since 2000 at the meetings of CLEF (Cross Language Evaluation Forum), following its predecessor workshops of TREC (Text Retrieval Conference) cross-language track. The NTCIR (NII Test Collection for IR Systems) workshop is also held each year in Japan for CLIR community focusing on English and Asian languages.

The study of cross-language mining appears relatively more lately than CLIR, partly due to the increasing demands on the quality of CLIR and machine translation, as well as the recent advancement of text/Web mining techniques. A typical early work on cross-lingual mining is believed to be PTMiner [14] that mines parallel text from the Web used for query translation. Other than parallel data mining, people also tried to mine the translations of Out-of-Vocabulary (OOV) terms from search results returned from search engine [5,18] or from web anchor texts and link structures [12]. Based on phonetic similarity, transliteration (the phonetic counterpart of a name in another language, e.g., "Schwarzenegger" is pronounced as "shi wa xin ge" in Chinese pinyin) of foreign names also could be extracted properly from the Web [10]. These methods are proposed to alleviate the OOV problem of CLIR since there is usually lack of appropriate translation resources for new terminologies and proper names, particularly in the scenario of cross-language web search.

Foundations

Most approaches to CLIR perform query translation followed by monolingual retrieval. So the retrieval performance is largely determined by the quality of query translation. Queries are typically translated either using a bilingual dictionary [15], a machine translation (MT) software [7], bilingual word association model learned from parallel corpus [6,14], or recently a query log of a search engine [9]. Despite the types of the resources being used, OOV translation and translation disambiguation are the two major bottlenecks for CLIR. On one hand, translation resources can never be comprehensive. Correctly translating queries, especially Web queries, is difficult since they often contain new words (e.g., new movies, brands, celebrities, etc.) occurring timely and frequently, yet

being OOV to the system; On the other hand, many words are polysemous, or they do not have a unique translation, and sometimes the alternative translations have very different meanings. This is known as translation ambiguity. Selecting the correct translation is not trivial due to the shortage of context provided in a query, and effective techniques for translation disambiguation are necessary.

It should be mentioned that document translation with MT in the opposite direction is an alternative approach to CLIR. However, it is less commonly used than query translation in the literature mainly because MT is computationally expensive and costly to develop, and the document sets in IR are generally very large. For cross-language web search, it is almost impractical to translate all the web pages before indexing. Some large scale attempts to compare query translation and document translation have suggested no clear advantage for either of the approaches to CLIR [12]. But they found that compared with extremely high quality human query translations, it is advantageous to incorporate both document and query translation into a CLIR system.

Cross-Language Web Mining

Mining Parallel Data The approaches of mining parallel text make extensive use of bilingual websites where parallel web pages corresponding to the specified language pair can be identified and downloaded. Then the bilingual texts are automatically aligned in terms of sentences and words by statistical aligning tools, such as GIZA++ [21]. The word translation probabilities can be derived with the statistics of word pairs occurring in the alignments, after which one can resort to statistical machine translation models, e.g., IBM model-1 [4], for translating given queries into the target language. The typical parallel data mining tools include PTMiner [14], STRAND [16] and the DOM-tree-alignment-based system [17].

Mining OOV Term Translation Web pages also contain translations of terms in either the body texts or the anchor texts of hyper-links pointing to other pages. For example, in some language pairs, such as Chinese-English or Japanese-English, the Web contains rich body texts in a mixture of multiple languages. Many of them contain bilingual translations of proper nouns, such as company names and person names. The work of [5,16] exploits this nice characteristic to automatically extract translations from search result for a large number of unknown query terms. Using the extracted bilingual translations, the performance of CLIR between English and Chinese is effectively improved. Both methods select translations based on some variants of co-occurrence statistics.

The anchor text of web pages' hyperlinks is another source for translational knowledge acquisition. This is based on the observation that the anchor texts of hyperlinks pointing to the same URL may contain similar descriptive texts. Lu et al. [11] uses anchor text of different languages to extract the regional aliases of query terms for constructing a translation lexicon. A probabilistic inference model is exploited to estimate the similarity between query term and extracted translation candidates.

Query Translation Disambiguation

Translation disambiguation or ambiguity resolution is crucial to the query translation accuracy. Compared to the simple dictionary-based translation approach without addressing translation disambiguation, the effectiveness of CLIR can be 60% lower than that of monolingual retrieval [3]. Different disambiguation techniques have been developed using statistics obtained from document collections, all resulting in significant performance improvement. Zhang et al. [19] give concise review on three main translation disambiguation techniques. These methods include using term similarity [1], word co-occurrence statistics of the target language documents, and language modeling based approaches [20]. In this subsection, we introduce these approaches following the review of Zhang et al. [19].

Disambiguation by Term Similarity Adriani [1] proposed a disambiguation technique based on the concept of statistical term similarity. The term similarity is measured by the Dice coefficient, which uses the term-distribution statistics obtained from the corpus. The similarity between term x and y, $SIM(x, y)$, is calculated as:

$$SIM(x,y) = 2\sum_{i=1}^{n}(w_{xi}w_{yi}) \Big/ \left(\sum_{i=1}^{n} w_{xi}^2 + \sum_{i=1}^{n} w_{yi}^2\right)$$

where w_{xi} and w_{yi} is the weights of term x and y in document i. This method computes the sum of maximum similarity values between each candidate

translation of a term and the translations of all other terms in the query. For each query term, the translation with the highest sum is selected as its translation. The results of Indonesian-English CLIR experiments demonstrated the effectiveness of this approach. There are many variant term association measures like Jaccard, Cosine, Overlap, etc. that can be applied similarly for calculating their similarity.

Disambiguation by Term Co-occurrence Ballesteros and Croft [3] used co-occurrence statistics obtained from the target corpus for resolving disambiguation. They assume the correct translations of query terms should co-occur in target language documents and incorrect translations tend not to co-occur. Similar approach is studied by Gao et al. [8]. They observed that the correlation between two terms is stronger when the distance between them is shorter. They extended the previous co-occurrence model by incorporating a distance factor $\mathrm{D}(\mathrm{x},\mathrm{y}) = \mathrm{e}^{-\alpha(Dis(x,y)-1)}$. The mutual information between term x and y, $MI(x, y)$, is calculated as:

$$MI(x,y) = \log\left(\frac{f_w(x,y)}{f_x f_y} + 1\right) \times D(x,y)$$

where $f_w(x, y)$ is the co-occurrence frequency of x and y that occur simultaneously within a window size of w in the collection, f_x is the collection frequency of x, and f_y is the collection frequency of y. $D(x, y)$ decreases exponentially when the distance between the two terms increases, where α is the decay rate, and $D(x, y)$ is the average distance between x and y in the collection. The experiments on the TREC9 Chinese collection showed that the distance factor leads to substantial improvements over the basic co-occurrence model.

Disambiguation by Language Modeling In the work of [20], a probability model based on hidden Markov model (HMM) is used to estimate the maximum likelihood of each sequence of possible translations of the original query. The highest probable translation set is selected among all the possible translation sets. HMM is a widely used for probabilistic modeling of sequence data. In their work, a smoothing technique based on absolute discounting and interpolation method is adopted to deal with the zero-frequency problem during probability estimation. See [20] for details.

Pre-/Post-Translation Expansion

Techniques of OOV term translation and translation disambiguation both aim to translate query correctly. However, it is arguable that precise translation may not be necessary for CLIR. Indeed, in many cases, it is helpful to introduce words even if they are not direct translations of any query word, but are closely related to the meaning of the query. This observation has led to the development of cross-lingual query expansion (CLQE) techniques [2,13]. [2] reported the enhancement on CLIR by post-translation expansion. [13] made performance comparison on various CLQE techniques, including pre-translation expansion, post-translation expansion and their combinations. Relevance feedback, the commonly used expansion technique in monolingual retrieval, is also widely adopted in CLQE. The basic idea is to expand original query by additional terms that are extracted from the relevant retrieval result initially returned. Amongst different relevance feedback methods, explicit feedback requires documents whose relevancy is explicitly marked by human; implicit feedback is inferred from users' behaviors that imply the relevancy of the selected documents, such as which returned documents are viewed or how long they view some of the documents; blind or "pseudo" relevance feedback is obtained by assuming that top n documents in the initial result are relevant.

Cross-Lingual Query Suggestion

Traditional query translation approaches rely on static knowledge and data resources, which cannot effectively reflect the quickly shifting interests of Web users. Moreover, the translated terms can be reasonable translations, but are not popularly used in the target language. For example, the French query "aliment biologique" is translated into "biologic food," yet the correct formulation nowadays should be "organic food." This mismatch makes the query translation in the target language ineffective. To address this problem, Gao et al. [9] proposed a principled framework called Cross-Lingual Query Suggestion (CLQS), which leverages cross-lingual mining and translation disambiguation techniques to suggest related queries found in the query log of a search engine.

CLQS aims to suggest related queries in a language different from the original query. CLQS is closely related to CLQE, but is distinct in that it suggests full queries that have been formulated by users so that the query integrity and coherence are preserved in the

suggested queries. It is used as a new means of query "translation" in CLIR tasks. The use of query log for CLQS stems from the observation that in the same period of time, many search users share the same or similar interests, which can be expressed in different manners in different languages. As a result, a query written in a source language is possible to have an equivalent in the query log of the target language. Especially, if the user intends to perform CLIR, then original query is even more likely to have its correspondent included in the target language log. Therefore, if a candidate for CLQS appears often in the query log, it is more likely to be the appropriate one to suggest. CLQS is testified being able to cover more relevant documents for the CLIR task.

The key problem with CLQS is how to learn a similarity measure between two queries in different languages. They define cross-lingual query similarity based on both translation relation and monolingual similarity. The principle for learning is, for a pair of queries, their cross-lingual similarity should fit the monolingual similarity between one query and the other query's translation. There are many ways to obtain a monolingual similarity between queries, e.g., co-occurrence based mutual information and χ^2. Any of them can be used as the target for the cross-lingual similarity function to fit. In this way, cross-lingual query similarity estimation is formulated as a regression task:

$$sim_{CL}(q_f, q_e) = w \cdot \varphi(f(q_f, q_e)) = sim_{ML}(T_{q_f}, q_e)$$

where given a source language query q_f, a target language query q_e, and a monolingual query similarity between them sim_{ML}, the cross-lingual query similarity sim_{CL} can be calculated as an inner product between a weight vector and the feature vector in the kernel space, and φ is the mapping from the input feature space onto the kernel space, and w is the weight vector which can be learned by support vector regression training. The monolingual similarity is measured by combining both query content-based similarity and click-through commonality in the query log.

This discriminative modeling framework can integrate arbitrary information sources to achieve an optimal performance. Multiple feature functions can be incorporated easily into the framework based on different translation resources, such as bilingual dictionaries, parallel data, web data, and query logs. They work uses co-occurrence-based dictionary translation disambiguation, IBM translation model-1 based on parallel corpus, and Web-based query translation mining as means to discover related candidate queries in the query log. Experiments on TREC6 French-English CLIR task demonstrate that CLQS-based CLIR is significantly better than the traditional dictionary-based query translation with disambiguation and machine translation approaches.

Latent Semantic Index (LSI) for CLIR

Different from most of the alternative approaches discussed above, LSI for CLIR [6] provides a method for matching text segments in one language with the segments of similar meaning in another language without having to translate either. Using a parallel corpus, LSI can create a language-independent representation of words. The representation matrix reflects the patterns of term correspondences in the documents of two languages. The matrix is factorized by Singular Value Decomposition (SVD) for deriving a latent semantic space with a reduced dimension, where similar terms are represented by similar vectors. In latent semantic space, therefore, the monolingual similarity between synonymous terms from one language and the cross-lingual similarity between translation pairs from different languages tend to be higher than the similarity with irrelevant terms. This characteristic allows relevant documents to be retrieved even if they do not share any terms in common with the query, which makes LSI suitable for CLIR.

Key Applications

Cross-language mining and retrieval is the foundation technology for searching web information across multiple languages. It can also provide the cross-lingual functionality for the retrieval of structured, semi-structured and un-structured document databases of specific domains or in large multinational enterprises.

Experimental Results

In general, for every presented work, there is an accompanying experimental evaluation in the corresponding reference. Especially, the three influential international workshops held annually, i.e., CLEF, NTCIR and TREC, defines many evaluation tasks for CLIR, and there are a large number of experimental results being published based on these benchmark specifications.

Data Sets

Data sets for benchmarking CLIR are released to the participants of TREC, CLEF and NTCIR workshops annually with license agreements.

Cross-references

- ▶ Anchor Text
- ▶ Average Precision
- ▶ Document databases
- ▶ Document Links and Hyperlinks
- ▶ Evaluation Metrics for Structured Text Retrieval
- ▶ Information Extraction
- ▶ Information Retrieval
- ▶ MAP
- ▶ MRR
- ▶ Query Expansion for Information Retreival
- ▶ Query Translation
- ▶ Relevance Feedback
- ▶ Singular Value Decomposition
- ▶ Snippet
- ▶ Stemming
- ▶ Stoplists
- ▶ Term Statistics for Structuerd Text Retrieval
- ▶ Term Weighting
- ▶ Text Indexing and Retrieval
- ▶ Text Mining
- ▶ Web Information Extraction
- ▶ Web Search Relevance Feedback

Recommended Reading

1. Adriani M. Using statistical term similarity for sense disambiguation in cross-language information retrieval. Inform. Retr., 2(1):71–82, 2000.
2. Ballestors L.A. and Croft W.B. Phrasal translation and query expansion techniques for cross-language information retrieval. In Proc. 20th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1997, pp. 84–91.
3. Ballestors L.A. and Croft W.B. Resolving ambiguity for cross-language information retrieval. In Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 64–71.
4. Brown P.F., Pietra S.A.D., Pietra V.D.J., and Mercer R.L. The mathematics of machine translation: parameter estimation. Comput. Linguist., 19:263–312, 1992.
5. Cheng P.-J., Teng J.-W., Chen R.-C., Wang J.-H., Lu W.-H., and Chien L.-F. Translating unknown queries with Web corpora for cross-language information retrieval. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 146–153.
6. Dumais S.T., Landauer T.K., and Littman M.L. Automatic cross-linguistic information retrieval using latent semantic indexing. ACM SIGIR Workshop on Cross-Linguistic Information Retrieval, 1996, pp. 16–23.
7. Fujii A. and Ishikawa T. Applying machine translation to two-stage cross-language information retrieval. In Proc. 4th Conf. Association for Machine Translation in the Americas, 2000, pp. 13–24.
8. Gao J., Zhou M., Nie, J.-Y., He H., and Chen W. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In Proc. 25th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2002, pp. 183–190.
9. Gao W., Niu C., Nie J.-Y., Zhou M., Hu J., Wong K.-F., and Hon H.-W.: Cross-lingual query suggestion using query logs of different languages. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 463–470.
10. Jiang L., Zhou M., Chien L.-F., and Niu C. Named entity translation with Web mining and transliteration. In Proc. 20th Int. Joint Conf. on AI, 2007, pp. 1629–1634.
11. Lu W.-H., Chien L.-F., and Lee H.-J. Translation of Web queries using anchor text mining. ACM Trans. Asian Lang. Information Proc., 1(2):159–172, 2002.
12. McCarley J.S. Should we translate the documents or the queries in cross-language information retrieval? In Proc. 27th Annual Meeting of the Assoc. for Computational Linguistics, 1999, pp. 208–214.
13. McNamee P. and Mayfield J. Comparing cross-language query expansion techniques by degrading translation resources. In Proc. 25th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2002, pp. 159–166.
14. Nie J.-Y., Smard M., Isabelle P., and Durand R. Cross-language information retrieval based on parallel text and automatic mining of parallel text from the Web. In Proc. 22nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1999, pp. 74–81.
15. Pirkola A., Hedlund T., Keshusalo H., and Järvelin K. Dictionary-based cross-language information retrieval: problems, methods, and research findings. Inform. Retr., 3(3–4): 209–230, 2001.
16. Resnik P. and Smith N.A. The Web as a parallel corpus. Comput. Linguist., 29(3):349–380, 2003.
17. Shi L., Niu C., Zhou M., and Gao J. A DOM Tree alignment model for mining parallel data from the Web. In Proc. 44th Annual Meeting of the Assoc. for Computational Linguistics, 2006, pp. 489–496.
18. Zhang Y. and Vines P. Using the Web for automated translation extraction in cross-language information retrieval. In Proc. 30th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2004, pp. 162–169.
19. Zhang Y., Vines P., and Zobel J. An empirical comparison of translation disambiguation techniques for Chinese-English Cross-Language Information Retrieval. In Proc. 3rd Asia Information Retrieval Symposium, 2006, pp. 666–672.
20. Zhang Y., Vines P., and Zobel J. Chinese OOV translation and post-translation query expansion in Chinese-English

cross-lingual information retrieval. ACM Trans. Asian Lang. Information Proc., 4(2):57–77, 2005.
21. http://www.fjoch.com/GIZA++.html

Cross-language Text Mining

► Cross-Language Mining and Retrieval

Cross-language Web Mining

► Cross-Language Mining and Retrieval

Cross-lingual Information Retrieval

► Cross-Language Mining and Retrieval

Cross-lingual Text Mining

► Cross-Language Mining and Retrieval

Cross-media Information Retrieval

► Cross-Modal Multimedia Information Retrieval

Cross-Modal Multimedia Information Retrieval

Qing Li, Yu Yang
City University of Hong Kong, Hong Kong, China

Synonyms

Multi-modal information retrieval; Cross-media information retrieval

Definition

Multimedia information retrieval tries to find the distinctive multimedia documents that satisfy people's needs within a huge dataset. Due to the vagueness on the representation of multimedia data, usually the user may only have some clues (e.g., a vague idea, a rough query object of the same or even different modality as that of the intended result) rather than concrete and indicative query objects. In such cases, traditional multimedia information retrieval techniques as Query-By-Example (QBE) fails to retrieve what users really want since their performance depends on a set of specifically defined features and carefully chosen query objects. The cross-modal multimedia information retrieval (CMIR) framework consists of a novel multifaceted knowledge base (which is embodied by a layered graph model) to discover the query results on multiple modalities. Such cross-modality paradigm leads to better query understanding and returns the retrieval result which meets user need better.

Historical Background

Previous works addressing multimedia information retrieval can be classified into two groups: approaches on single-modality, and those on multi-modality integration.

Retrieval Approaches on Single-Modality

The retrieval approach in this group only deals with a single type of media, so that most content-based retrieval (CBR) approaches [2,3,5,8,9] fall into this group. These approaches differ from each other in either the low-level features extracted from the data, or the distance functions used for similarity calculation. Despite the differences, all of them are similar in two fundamental aspects: (i) they all rely on low-level features; (ii) they all use the query-by-example paradigm.

Retrieval Approaches on Multi-Modality Integration

More recently there are some works that investigate the integration of multi-modality data, usually between text and image, for better retrieval performance. For example, iFind [7] proposes a unified framework under which the semantic feature (text) and low-level features are combined for image retrieval, whereas the 2M2Net [12] system extends this framework to the retrieval of video and audio. WebSEEK [9] extracts keywords from the surrounding text of image and videos, which is used as their indexes in the retrieval process. Although these systems involve more than one media, different types of media are not actually integrated but are on different levels. Usually, text is only used as the annotation (index) of other medias. In this regard, cross-modal multimedia information

retrieval (CMIR) enables an extremely high degree of multi-modality integration, since it allows the interaction among objects of any modality in any possible ways (via different types of links).

MediaNet [1] and multimedia thesaurus (MMT) [10] seek to provide a multimedia representation of semantic concept – a concept described by various media objects including text, image, video, etc – and establish the relationships among these concepts. MediaNet extends the notion of relationships to include even perceptual relationships among media objects. Both approaches can be regarded as "concept-centric" approaches since they realize an organization of multi-modality objects around semantic concepts. In contrast, CMIR is "concept-less" since it makes no attempt to identify explicitly the semantics of each object.

Foundations

The cross-modality multimedia information retrieval (CMIR) mechanism shapes a novel scenario for multimedia retrieval: The user starts the search by supplying a set of seed objects as the hints of his intention, which can be of any modality (even different with the intended objects), and are not necessarily the eligible results by themselves. From the seeds, the system figures out the user's intention and returns a set of cross-modality objects that potentially satisfy this intention. The user can give further hints by identifying the results approximating his need, based on which the system improve its estimation about the user intention and refines the results towards it. This scenario can be also interpreted as a cooperative process: the user tries to focus the attention of the system to the objects by giving hints on the intended results, while the system tries to return more reasonable results that allows user to give better hints. A comparison between CMIR and the current CBR approaches is shown in Table 1.

To support all the necessary functionalities for such an ideal scenario, a suite of unique models, algorithms and strategies are developed in CMIR. As shown in Fig. 1, the foundation of the whole mechanism is a multifaceted knowledge base describing the relationships among cross-modality objects. The kernel of the knowledge base is a layered graph model, which characterizes the knowledge on (i) history of user behaviors, (ii) structural relationships among media objects, and (iii) content of media objects, at each of its layers. Link structure analysis—an established technique for web-oriented applications—is tailored to the retrieval of cross-modality data based on the layered graph model. A unique relevant feedback technique that gears with the underlying graph model is proposed, which can enrich the knowledge base by updating the links of the graph model according to user behaviors. The loop in Fig. 1 reveals the hill-climbing nature of the CMIR mechanism, i.e., it enhances its performance by learning from the previously conducted queries and feedbacks.

Layered Graph Model

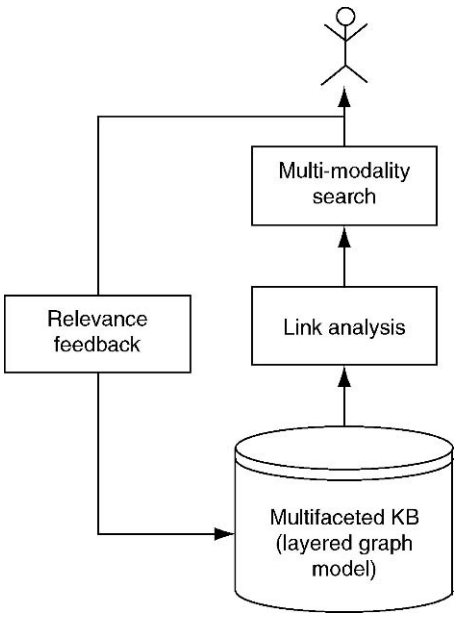
As the foundation of the retrieval capability, the multifaceted knowledge base accommodates a broad range of knowledge indicative of data semantics, mainly in three aspects: (i) user behaviors in the user-system interaction, (ii) structural relationships among media objects, and (iii) content of each media object. The kernel of the knowledge base is a three-layer graph model, with each layer describing the knowledge in one aspect, called knowledge layer. Its formal definition is given as follows.

Definition 1 *A **knowledge layer** is a undirected graph $G = (V, E)$, where V is a finite set of vertices and E is a finite set of edges. Each element in V corresponds to a media object $O_i \in O$, where O is the collection of media*

Cross-Modal Multimedia Information Retrieval. **Table 1.** CBR paradigms, drawbacks, and suggested remedies in CMIR

	CBR paradigms	Drawbacks	Suggested remedies in CMIR
Interaction	Highly representative sample object	Vague idea, or clear idea without appropriate samples	Cross-modality seed objects, only as hints
Data index	Low-level features	Inadequate to capture semantics	Multifaceted knowledge (user behaviors, structure, content)
Results	Single-modality, perceptually similar objects	Looks like or sounds like, but not what user actually needs	Cross-modality, semantically related objects

objects in the database. E is a ternary relation defined on $V \times V \times R$, *where R represents real numbers. Each edge in E has the form of* $<O_i, O_j, r>$, *denoting a semantic link between* O_i *and* O_j *with r as the weight of the link. The graph corresponds to a* $|V| \times |V|$ ***adjacency matrix*** (The adjacency matrix defined here is slightly different from the conventional definition in mathematics, in which each component is a binary value indicating the existence of the corresponding edge.) $M = [m_{ij}]$, *where* $m_{ij} = m_{ji}$ *always holds. Each element* $m_{ij} = r$ *if there is an edge* $<O_i, O_j, r>$, *and* $m_{ij} = 0$ *if there is no edge between* O_i *and* O_j. *The elements on the diagonal are set to zero, i.e.,* $m_{ii} = 0$.

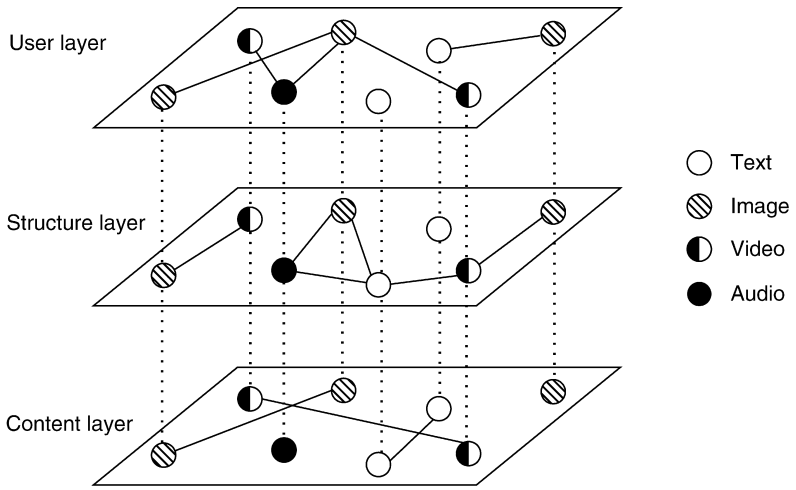


Cross-Modal Multimedia Information Retrieval.
Figure 1. Overview of the CMIR mechanism.

Each semantic link between two media objects may have various interpretations, which corresponds to one of the three cases: (i) a user has implied the relevance between the two objects during the interaction, e.g., designating them as the positive example in the same query session; (ii) there is a structural relationships between them, e.g., they come from the same or linked web page(s); or (iii) they resemble each other in terms of their content. The multifaceted knowledge base seamlessly integrates all these links into the same model while preserving their mutual independence.

Definition 2 *The multifaceted knowledge base is a* ***layered graph model*** *consisting of three superimposed knowledge layers, which from top to bottom are* ***user layer***, ***structure layer***, *and* ***content layer***. *The vertices of the three layers correspond to the same set of media objects, but their edges are different either in occurrences or in interpretations.*

Figure 2 illustrates the layered graph model. Note that the ordering of the three layers is immutable, which reflects their priorities in terms of knowledge reliability. The user layer is placed uppermost since user judgment is assumed most reliable (not necessarily always reliable). Structure links is a strong indicator of relevance, but not as reliable as user links. The lowest layer is the content layer. As a generally accepted fact in



Cross-Modal Multimedia Information Retrieval. **Figure 2.** The Layered graph model as multifaceted knowledge base.

CBR area, content similarity does not entail any well-defined mapping with semantics.

A unique property of the layered graph model is that it stores the knowledge on the links (or relationships) among media objects, rather than on the nodes (media objects) upon which most existing retrieval systems store the data index. All the algorithms based of this model can be interpreted as manipulation of links: to serve the user query, relevant knowledge is extracted from this graph model by analyzing the link structure; meanwhile, user behaviors are studied to enrich the knowledge by updating the links. An advantage of such link-based approach is that the retrieval can be performed in a relatively small locality connected via links instead of the whole database, and therefore it can afford more sophisticated retrieval algorithms.

Link Analysis Based Retrieval

As illustrated in Fig. 3, the retrieval process can be described as a circle: the intended objects are retrieved through the upper semicircle, and the user evaluations are studied and incorporated into the knowledge base though the lower half-circle, which initiates a new circle to refine the previously retrieved results based on the updated knowledge. Consequently, it is a hill-climbing approach in that the performance is enhanced incrementally as the loop is repeated.

The retrieval process consists of five steps (as shown in Fig. 3): (i) generate the seed objects as the hints of the user's intention; (ii) span the seeds to a collection of candidate objects via the links in the layered graph model; (iii) distill the results by ranking the candidates based on link structure analysis, (iv) update the knowledge base to incorporate the user evaluation of the current results, and (v) refine the results based on user evaluations.

Key Applications

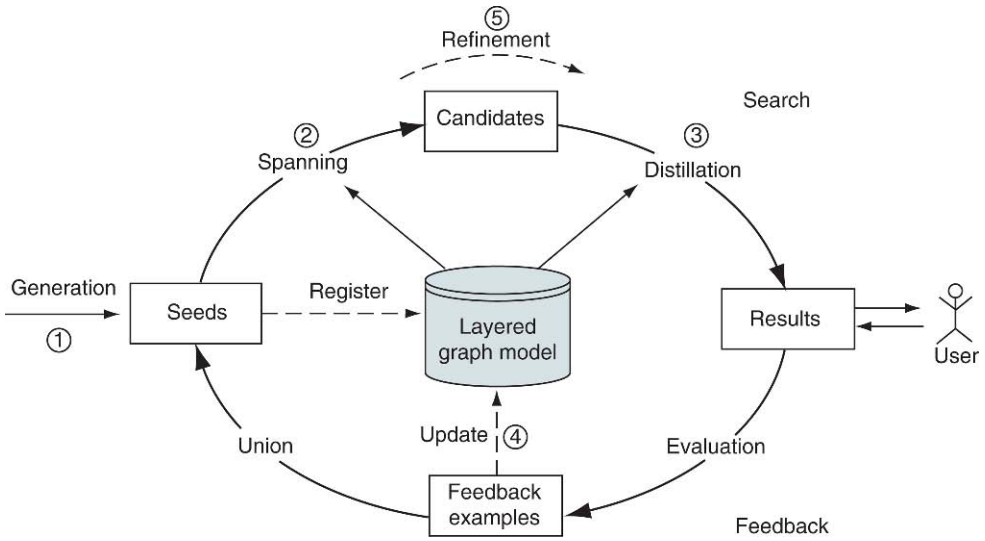
Multimedia Information Retrieval System

For multimedia data, the modalities supported can be texts (surrounding or tagged), images, videos and audios. An ongoing prototype [11] utilizes the primitive features and similarity functions for these media shown in Table 2. The experimental results prove the usefulness of the approach for better query understanding.

Future Directions

Due to the generality and extensibility of the CMIR, there are many potential directions that can be implemented on it:

Navigation. The graph model provides abundant links through which the user can traverse from an object to its related objects. An intuitive scenario for navigation is when the user is looking at a certain object, he is recommended with the objects that are linked to it in the graph model, ranked by their link weights and link types, from which he may select one as the next object he will navigate to.



Cross-Modal Multimedia Information Retrieval. Figure 3. Overview of the link analysis based retrieval algorithm.

Cross-Modal Multimedia Information Retrieval. Table 2. Primitive features and similarity function used in prototype

	Text	Image	Video
Primitive features	Keywords, weighted by TF*IDF	256-d HSV color histogram, 64-d LAB color coherence, 32-d Tamura directionality	First frame of each shot as key-frame, indexing key-frame as an image
Similarity function	Cosine distance	Euclidean distance for each feature, linear combination of different similarities	Key-frame (image) similarity as shot similarity, average pair-wise shot similarity as video similarity

Clustering. Clustering cross-modality objects into semantically meaningful groups is also an important and challenging issue, which requires an underlying *similarity function* among objects, along with a method that produces clusters based on the similarity function. The layered graph model provides knowledgeable and rich links, based on which different similarity functions can be easily formulated. Meanwhile, many existing approaches can be employed as the clustering method, such as simulated and deterministic annealing algorithm [4]. Moreover, CMIR inherently allows the clustering of cross-modality objects, rather than single-modality objects that most previous classification approaches can deal with.

Personalized retrieval. The user layer of the graph model characterizes the knowledge obtained from the behaviors of the whole population of users, and allows a query from a single user to benefit from such common knowledge. However, each user may have his/her personal interests, which may not agree with each other. The "multi-leveled user profile" mechanism [6] leads a good direction for future study.

Cross-references

► Multimedia Data
► Multimedia Information Retrieval

Recommended Reading

1. Benitez A.B., Smith J.R., and Chang S.F. MediaNet: a multimedia information network for knowledge representation. In Proc. SPIE Conf. on Internet Multimedia Management Systems, vol. 4210, 2000, pp. 1–12.
2. Chang S.F., Chen W., Meng H.J., Sundaram H., and Zhong D. VideoQ: an automated content based video search system using visual cues. In Proc. 5th ACM Int. Conf. on Multimedia, 1997.
3. Flickner M., Sawhney H., Niblack W., and Ashley J. Query by image and video content: the QBIC system. IEEE Comput., 28(9):23–32, 1995.
4. Hofmann T. and Buhmann J.M. Pairwise data clustering by deterministic annealing. IEEE Trans. Pattern Anal. Mach. Intell., 19(1):1–14, 1997.
5. Huang T.S., Mehrotra S., and Ramchandran K. Multimedia analysis and retrieval system (MARS) project. In Proc. 33rd Annual Clinic on Library Application of Data Processing-Digital Image Access and Retrieval, 1996.
6. Li Q., Yang J., and Zhuang Y.T. Web-based multimedia retrieval: balancing out between common knowledge and personalized views. In Proc. 2nd Int. Conf. on Web Information Systems Eng., 2001.
7. Lu Y., Hu C.H., Zhu X.Q., Zhang H.J., and Yang Q. A unified framework for semantics and feature based relevance feedback in image retrieval systems. In Proc. 8th ACM Int. Conf. on Multimedia, 2000, pp. 31–38.
8. Smith J.R. and Chang S.F. VisualSEEk: a fully automated content-based image query system. In Proc. 4th ACM Int. Conf. on Multimedia, 1996.
9. Smith J.R. and Chang S.F. Visually searching the web for content. IEEE Multimed. Mag., 4(3):12–20, 1997.
10. Tansley R. The Multimedia Thesaurus: An Aid for Multimedia Information Retrieval and Navigation. Master Thesis, Computer Science, University of Southampton, UK, 1998.
11. Yang J., Li Q., and Zhuang Y. Octopus: Aggressive search of multi-modality data using multifaceted knowledge base. In Proc. 11th Int. World Wide Web Conference, 2002, pp. 54–64.
12. Yang J., Zhuang Y.T., and Li Q. Search for multi-modality data in digital libraries. In Proc. Second IEEE Pacific-Rim Conference on Multimedia, 2001.

Cross-Validation

Payam Refaeilzadeh, Lei Tang, Huan Liu
Arizona State University, Tempe, AZ, USA

Synonyms

Rotation estimation

Definition

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive

rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation. Other forms of cross-validation are special cases of *k*-fold cross-validation or involve repeated rounds of *k*-fold cross-validation.

In *k*-fold cross-validation, the data is first partitioned into *k* equally (or nearly equally) sized segments or folds. Subsequently *k* iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k - 1$ folds are used for learning. Fig. 1 demonstrates an example with $k = 3$. The darker section of the data are used for training while the lighter sections are used for validation. In data mining and machine learning 10-fold cross-validation ($k = 10$) is the most common.

Cross-validation is used to evaluate or compare learning algorithms as follows: in each iteration, one or more learning algorithms use $k - 1$ folds of data to learn one or more models, and subsequently the learned models are asked to make predictions about the data in the validation fold. The performance of each learning algorithm on each fold can be tracked using some predetermined performance metric like accuracy. Upon completion, *k* samples of the performance metric will be available for each algorithm. Different methodologies such as averaging can be used to obtain an aggregate measure from these sample, or these samples can be used in a statistical hypothesis test to show that one algorithm is superior to another.

Historical Background

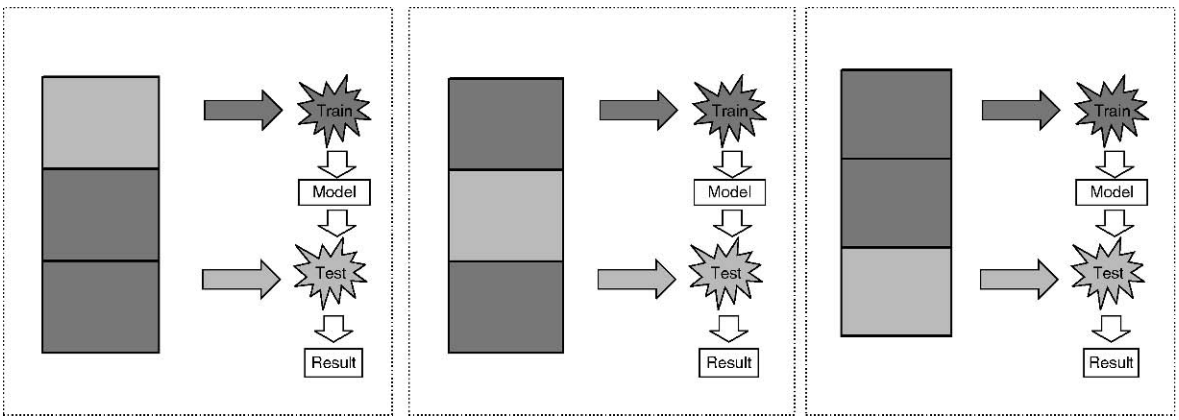
In statistics or data mining, a typical task is to learn a model from available data. Such a model may be a regression model or a classifier. The problem with evaluating such a model is that it may demonstrate adequate prediction capability on the training data, but might fail to predict future unseen data. cross-validation is a procedure for estimating the generalization performance in this context. The idea for cross-validation originated in the 1930s [6]. In the paper one sample is used for regression and a second for prediction. Mosteller and Tukey [9], and various other people further developed the idea. A clear statement of cross-validation, which is similar to current version of *k*-fold cross-validation, first appeared in [8]. In 1970s, both Stone [12] and Geisser [4] employed cross-validation as means for choosing proper model parameters, as opposed to using cross-validation purely for estimating model performance. Currently, cross-validation is widely accepted in data mining and machine learning community, and serves as a standard procedure for performance estimation and model selection.

Foundations

There are two possible goals in cross-validation:

- To estimate performance of the learned model from available data using one algorithm. In other words, to gauge the generalizability of an algorithm.
- To compare the performance of two or more different algorithms and find out the best algorithm for the available data, or alternatively to compare the performance of two or more variants of a parameterized model.

The above two goals are highly related, since the second goal is automatically achieved if one knows the accurate estimates of performance. Given a sample



Cross-Validation. Figure 1. Procedure of three-fold cross-validation.

of N data instances and a learning algorithm A, the average cross-validated accuracy of A on these N instances may be taken as an estimate for the accuracy of A on unseen data when A is trained on all N instances. Alternatively if the end goal is to compare two learning algorithms, the performance samples obtained through cross-validation can be used to perform two-sample statistical hypothesis tests, comparing a pair of learning algorithms.

Concerning these two goals, various procedures are proposed:

Resubstitution Validation

In resubstitution validation, the model is learned from all the available data and then tested on the same set of data. This validation process uses all the available data but suffers seriously from over-fitting. That is, the algorithm might perform well on the available data yet poorly on future unseen test data.

Hold-Out Validation

To avoid over-fitting, an independent test set is preferred. A natural approach is to split the available data into two non-overlapped parts: one for training and the other for testing. The test data is held out and not looked at during training. Hold-out validation avoids the overlap between training data and test data, yielding a more accurate estimate for the generalization performance of the algorithm. The downside is that this procedure does not use all the available data and the results are highly dependent on the choice for the training/test split. The instances chosen for inclusion in the test set may be too easy or too difficult to classify and this can skew the results. Furthermore, the data in the test set may be valuable for training and if it is held-out prediction performance may suffer, again leading to skewed results. These problems can be partially addressed by repeating hold-out validation multiple times and averaging the results, but unless this repetition is performed in a systematic manner, some data may be included in the test set multiple times while others are not included at all, or conversely some data may always fall in the test set and never get a chance to contribute to the learning phase. To deal with these challenges and utilize the available data to the max, k-fold cross-validation is used.

K-Fold Cross-Validation

In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k - 1$ folds are used for learning. Data is commonly stratified prior to being split into k folds. Stratification is the process of rearranging the data as to ensure each fold is a good representative of the whole. For example in a binary classification problem where each class comprises 50% of the data, it is best to arrange the data such that in every fold, each class comprises around half the instances.

Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) is a special case of k-fold cross-validation where k equals the number of instances in the data. In other words in each iteration nearly all the data except for a single observation are used for training and the model is tested on that single observation. An accuracy estimate obtained using LOOCV is known to be almost unbiased but it has high variance, leading to unreliable estimates [3]. It is still widely used when the available data are very rare, especially in bioinformatics where only dozens of data samples are available.

Repeated *K*-Fold Cross-Validation

To obtain reliable performance estimation or comparison, large number of estimates are always preferred. In k-fold cross-validation, only k estimates are obtained. A commonly used method to increase the number of estimates is to run k-fold cross-validation multiple times. The data is reshuffled and re-stratified before each round.

Pros and Cons

Kohavi [5] compared several approaches to estimate accuracy: cross-validation(including regular cross-validation, leave-one-out cross-validation, stratified cross-validation) and bootstrap (sample with replacement), and recommended *stratified 10-fold cross-validation* as the best model selection method, as it tends to provide less biased estimation of the accuracy.

Salzberg [11] studies the issue of comparing two or more learning algorithms based on a performance metric, and proposes using k-fold cross-validation followed by appropriate hypothesis test rather than directly comparing the average accuracy. Paired t-test is one test which takes into consideration the variance of training and test data, and is widely used in machine

learning. Dietterich [2] studied the properties of 10-fold cross-validation followed by a paired t-test in detail and found that such a test suffers from higher than expected type I error. In this study, this high type I error was attributed to high variance. To correct for this Dietterich proposed a new test: 5 × 2-fold cross-validation. In this test 2-fold cross-validation is run five times resulting in 10 accuracy values. The data is re-shuffled and re-stratified after each round. All 10 values are used for average accuracy estimation in the t-test but only values from one of the five 2-fold cross-validation rounds is used to estimate variance. In this study 5 × 2-fold cross-validation is shown to have acceptable type I error but not to be as powerful as 10-fold cross validation and has not been widely accepted in data mining community.

Bouckaert [1] also studies the problem of inflated type-I error with 10-fold cross-validation and argues that since the samples are dependent (because the training sets overlap), the actual degrees of freedom is much lower than theoretically expected. This study compared a large number of hypothesis schemes, and recommend 10 × 10 fold cross-validation to obtain 100 samples, followed with t-test with degree of freedom equal to 10 (instead of 99). However this method has not been widely adopted in data mining field either and 10-fold cross-validation remains the most widely used validation procedure.

A brief summery of the above results is presented in Table 1.

Cross-Validation. Table 1. Pros and Cons of different validation methods

Validation method	Pros	Cons
Resubstitution Validation	Simple	Over-fitting
Hold-out Validation	Independent training and test	Reduced data for training and testing; Large variance
k-fold cross validation	Accurate performance estimation	Small samples of performance estimation; Overlapped training data; Elevated Type I error for comparison; Underestimated performance variance or overestimated degree of freedom for comparison
Leave-One-Out cross-validation	Unbiased performance estimation	Very large variance
Repeated *k*-fold cross-validation	Large number of performance estimates	Overlapped training and test data between each round; Underestimated performance variance or overestimated degree of freedom for comparison

Why 10-Fold Cross-Validation: From Ideal to Reality

Whether estimating the performance of a learning algorithm or comparing two or more algorithms in terms of their ability to learn, an ideal or statistically sound experimental design must provide a *sufficiently large* number of *independent* measurements of the algorithm(s) performance.

To make independent measurements of an algorithm's performance one must ensure that the factors affecting the measurement are independent from one run to the next. These factors are: (i) the training data the algorithm learns from and, (ii) the test data one uses to measure the algorithm's performance. If some data is used for testing in more than one round, the obtained results, for example the accuracy measurements from these two rounds, will be dependent and a statistical comparison may not be valid. In fact, it has been shown that a paired t-test based on taking several random train/test splits tends to have an extremely high probability of Type I error and should never be used [2].

Not only must the datasets be independently controlled across different runs, there must not be any overlap between the data used for learning and the data used for validation in the same run. Typically, a learning algorithm can make more accurate predictions on a data that it has seen during the learning phase than those it has not. For this reason, an overlap between the training and validation set can lead to an over-estimation of the performance metric and is forbidden. To satisfy the other requirement, namely a sufficiently large sample, most statisticians call for 30+ samples.

For a truly sound experimental design, one would have to split the available data into 30 × 2 = 60 partitions to perform 30 truly independent train-test runs. However, this is not practical because the performance of learning algorithms and their ranking is generally not invariant with respect to the number of

samples available for learning. In other words, an estimate of accuracy in such a case would correspond to the accuracy of the learning algorithm when it learns from just 1⁄60 of the available data (assuming training and validation sets are of the same size). However, the accuracy of the learning algorithm on unseen data when the algorithm is trained on all the currently available data is likely much higher since learning algorithms generally improve in accuracy as more data becomes available for learning. Similarly, when comparing two algorithms A and B, even if A is discovered to be the superior algorithm when using 1⁄60 of the available data, there is no guarantee that it will also be the superior algorithm when using all the available data for learning. Many high performing learning algorithms use complex models with many parameters and they simply will not perform well with a very small amount of data. But they may be exceptional when sufficient data is available to learn from.

Recall that two factors affect the performance measure: the training set, and the test set. The training set affects the measurement indirectly through the learning algorithm, whereas the composition of the test set has a direct impact on the performance measure. A reasonable experimental compromise may be to *allow for overlapping training sets, while keeping the test sets independent.* *K*-fold cross-validation does just that.

Now the issue becomes selecting an appropriate value for k. A large k is seemingly desirable, since with a larger k (i) there are more performance estimates, and (ii) the training set size is closer to the full data size, thus increasing the possibility that any conclusion made about the learning algorithm(s) under test will generalize to the case where all the data is used to train the learning model. As k increases, however, the overlap between training sets also increases. For example, with 5-fold cross-validation, each training set shares only 3⁄4 of its instances with each of the other four training sets whereas with 10-fold cross-validation, each training set shares 8⁄9 of its instances with each of the other nine training sets. Furthermore, increasing k shrinks the size of the test set, leading to less precise, less fine-grained measurements of the performance metric. For example, with a test set size of 10 instances, one can only measure accuracy to the nearest 10%, whereas with 20 instances the accuracy can be measured to the nearest 5%. These competing factors have all been considered and the general consensus in the data mining community seems to be that $k = 10$ is a good compromise. This value of k is particularity attractive because it makes predictions using 90% of the data, making it more likely to be generalizable to the full data.

Key Applications

Cross-validation can be applied in three contexts: performance estimation, model selection, and tuning learning model parameters.

Performance Estimation

As previously mentioned, cross-validation can be used to estimate the performance of a learning algorithm. One may be interested in obtaining an estimate for any of the many performance indicators such as accuracy, precision, recall, or F-score. Cross-validation allows for all the data to be used in obtaining an estimate. Most commonly one wishes to estimate the accuracy of a classifier in a supervised-learning environment. In such a setting, a certain amount of labeled data is available and one wishes to predict how well a certain classifier would perform if the available data is used to train the classifier and subsequently ask it to label unseen data. Using 10-fold cross-validation one repeatedly uses 90% of the data to build a model and test its accuracy on the remaining 10%. The resulting average accuracy is likely somewhat of an underestimate for the true accuracy when the model is trained on all data and tested on unseen data, but in most cases this estimate is reliable, particularly if the amount of labeled data is sufficiently large and if the unseen data follows the same distribution as the labeled examples.

Model Selection

Alternatively cross-validation may be used to compare a pair of learning algorithms. This may be done in the case of newly developed learning algorithms, in which case the designer may wish to compare the performance of the classifier with some existing baseline classifier on some benchmark dataset, or it may be done in a generalized model-selection setting. In generalized model selection one has a large library of learning algorithms or classifiers to choose from and wish to select the model that will perform best for a particular dataset. In either case the basic unit of work is pair-wise comparison of learning algorithms. For generalized model selection combining the results of many pair-wise comparisons to obtain a single *best* algorithm may be difficult, but this is beyond the

scope of this article. Researchers have shown that when comparing a pair of algorithms using cross-validation it is best to employ proper two sample hypothesis testing instead of directly comparing the average accuracies. Cross-validation yields k pairs of accuracy values for the two algorithms under test. It is possible to make a null hypothesis assumption that the two algorithms perform equally well and set out to gather evidence against this null-hypothesis using a two-sample test. The most widely used test is the paired t-test. Alternatively the non-parametric sign test can be used.

A special case of model selection comes into play when dealing with non-classification model selection. For example when trying to pick a feature selection [7] algorithm that will maximize a classifier's performance on a particular dataset. Refaeilzadeh et al. [10] explore this issue in detail and explain that there are in fact two variants of cross-validation in this case: performing feature selection before splitting data into folds (OUT) or performing feature selection k times inside the cross-validation loop (IN). The paper explains that there is potential for bias in both cases: With OUT, the feature selection algorithm has looked at the test set, so the accuracy estimate is likely inflated; On the other hand with IN the feature selection algorithm is looking at less data than would be available in a real experimental setting, leading to underestimated accuracy. Experimental results confirm these hypothesis and further show that:

- In cases where the two feature selection algorithms are not statistically differentiable, IN tends to be more truthful.
- In cases where one algorithm is better than another, IN often favors one algorithm and OUT the other.

OUT can in fact be the better choice even if it demonstrates a larger bias than IN in estimating accuracy. In other words, *estimation bias is not necessarily an indication of poor pair-wise comparison*. These subtleties about the potential for bias and validity of conclusions obtained through cross-validation should always be kept in mind, particularly when the model selection task is a complicated one involving pre-processing as well as learning steps.

Tuning

Many classifiers are parameterized and their parameters can be *tuned* to achieve the best result with a particular dataset. In most cases it is easy to learn the proper value for a parameter from the available data. Suppose a Naïve Bayes classifier is being trained on a dataset with two classes: {+, –}. One of the parameters for this classifier is the prior probability p(+). The best value for this parameter according to the available data can be obtained by simply counting the number of instances that are labeled positive and dividing this number by the total number of instances. However in some cases parameters do not have such intrinsic meaning, and there is no good way to pick a best value other than trying out many values and picking the one that yields the highest performance. For example, support vector machines (SVM) use soft-margins to deal with noisy data. There is no easy way of learning the best value for the soft margin parameter for a particular dataset other than trying it out and seeing how it works. In such cases, cross-validation can be performed on the training data as to measure the performance with each value being tested. Alternatively a portion of the training set can be reserved for this purpose and not used in the rest of the learning process. But if the amount of labeled data is limited, this can significantly degrade the performance of the learned model and cross-validation may be the best option.

Cross-references

- ▶ Classification
- ▶ Evaluation Metrics for Structured Text Retrieval
- ▶ Feature Selection for Clustering

Recommended Reading

1. Bouckaert R.R. Choosing between two learning algorithms based on calibrated tests. In Proc. 20th Int. Conf. on Machine Learning, 2003, pp. 51–58.
2. Dietterich T.G. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput., 10(7):1895–1923, 1998.
3. Efron B. Estimating the error rate of a prediction rule: improvement on cross-validation. J. Am. Stat. Assoc., 78: 316–331,1983.
4. Geisser S. The predictive sample reuse method with applications. J. Am. Stat. Assoc., 70(350):320–328,1975.
5. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proc. 14th Int. Joint Conf. on AI, 1995, pp. 1137–1145.
6. Larson S. The shrinkage of the coefficient of multiple correlation. J. Educat. Psychol., 22:45–55, 1931.
7. Liu H. and Yu L. Toward integrating feature selection algorithms for classification and clustering. IEEE Trans. Knowl. Data Eng., 17(4):491–502, 2005.

8. Mosteller F. and Tukey J.W. Data analysis, including statistics. In Handbook of Social Psychology. Addison-Wesley, Reading, MA, 1968.
9. Mosteller F. and Wallace D.L. Inference in an authorship problem. J. Am. Stat. Assoc., 58:275–309, 1963.
10. Refaeilzadeh P., Tang L., and Liu H. On comparison of feature selection algorithms. In Proc. AAAI-07 Workshop on Evaluation Methods in Machine Learing II. 2007, pp. 34–39.
11. Salzberg S. On comparing classifiers: pitfalls to avoid and a recommended approach. Data Min. Knowl. Disc., 1(3):317–328, 1997.
12. Stone M. Cross-validatory choice and assessment of statistical predictions. J. Royal Stat. Soc., 36(2):111–147, 1974.

Cryptographic Hash Functions

► Hash Functions

C-Tables

► Conditional Tables

Cube

Torben Bach Pedersen
Aalborg University, Aalborg, Denmark

Synonyms

Hypercube

Definition

A cube is a data structure for storing and and analyzing large amounts of multidimensional data, often referred to as *On-Line Analytical Processing (OLAP)*. Data in a cube lives in a space spanned by a number of hierarchical *dimensions*. A single point in this space is called a *cell*. A (non-empty) cell contains the values of one or more *measures*.

Key Points

As an example, a three-dimensional cube for capturing sales may have a Product *dimension P*, a Time dimension *T*, and a Store dimension *S*, capturing the product sold, the time of sale, and the store it was sold in, for each sale, respectively. The cube has two *measures*: Dollar Sales and ItemSales, capturing the sales price and the number of items sold, respectively. In a cube, the combinations of a dimension value from each dimension define a *cell* of the cube. The measure value(s), e.g., DollarSales and ItemSales, corresponding to the particular combination of dimension values are then stored stored in the corresponding cells.

Data cubes provide true multidimensionality. They generalize spreadsheets to any number of dimensions, indeed cubes are popularly referred to as "spreadsheets on stereoids." In addition, *hierarchies* in dimensions and formulas are first-class, built-in concepts, meaning that these are supported without duplicating their definitions. A collection of related cubes is commonly referred to as a *multidimensional database* or a *multidimensional data warehouse*.

In a cube, dimensions are first-class concepts with associated domains, meaning that the addition of new dimension values is easily handled. Although the term "cube" implies three dimensions, a cube can have any number of dimensions. It turns out that most real-world cubes have 4–12 dimensions [3]. Although there is no theoretical limit to the number of dimensions, current tools often experience performance problems when the number of dimensions is more than 10–15. To better suggest the high number of dimensions, the term "hypercube" is often used instead of "cube."

Depending on the specific application, a highly varying percentage of the cells in a cube are non-empty, meaning that cubes range from *sparse* to *dense*. Cubes tend to become increasingly sparse with increasing dimensionality and with increasingly finer granularities of the dimension values. A non-empty cell is called a *fact*. The example has a fact for each combination of time, product, and store where at least one sale was made.

Generally, only two or three dimensions may be viewed at the same time, although for low-cardinality dimensions, up to four dimensions can be shown by nesting one dimension within another on the axes. Thus, the dimensionality of a cube is reduced at query time by *projecting* it down to two or three dimensions via *aggregation* of the measure values across the projected-out dimensions. For example, to view sales by Store and Time, data is aggregates over the entire Product dimension, i.e., for all products, for each combination of Store and Time.

OLAP SQL extensions for cubes were pioneered by the proposal of the data cube operators CUBE and ROLLUP [1]. The CUBE operator generalizes GROUP BY, crosstabs, and subtotals using the special "ALL" value that denotes that an aggregation has been performed

over all values for one or more attributes, thus generating a subtotal, or a grand total.

Cross-references

- ▶ Cube Implementations
- ▶ Dimension
- ▶ Hierarchy
- ▶ Measure
- ▶ Multidimensional Modeling
- ▶ On-Line Analytical Processing

Recommended Reading

1. Gray J., Chaudhuri S., Bosworth A., Layman A., Venkatrao M., Reichart D., Pellow F., and Pirahesh H. Data cube: a relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining Knowl. Discov., 1(1):29–54, 1997.
2. Pedersen T.B., Jensen C.S., and Dyreson C.E. A foundation for capturing and querying complex multidimensional data. Inf. Syst., 26(5):383–423, 2001.
3. Thomsen E. OLAP Solutions: Building Multidimensional Information Systems. Wiley, New York, 1997.

Cube Implementations

Konstantinos Morfonios, Yannis Ioannidis
University of Athens, Athens, Greece

Synonyms

Cube materialization; Cube precomputation

Definition

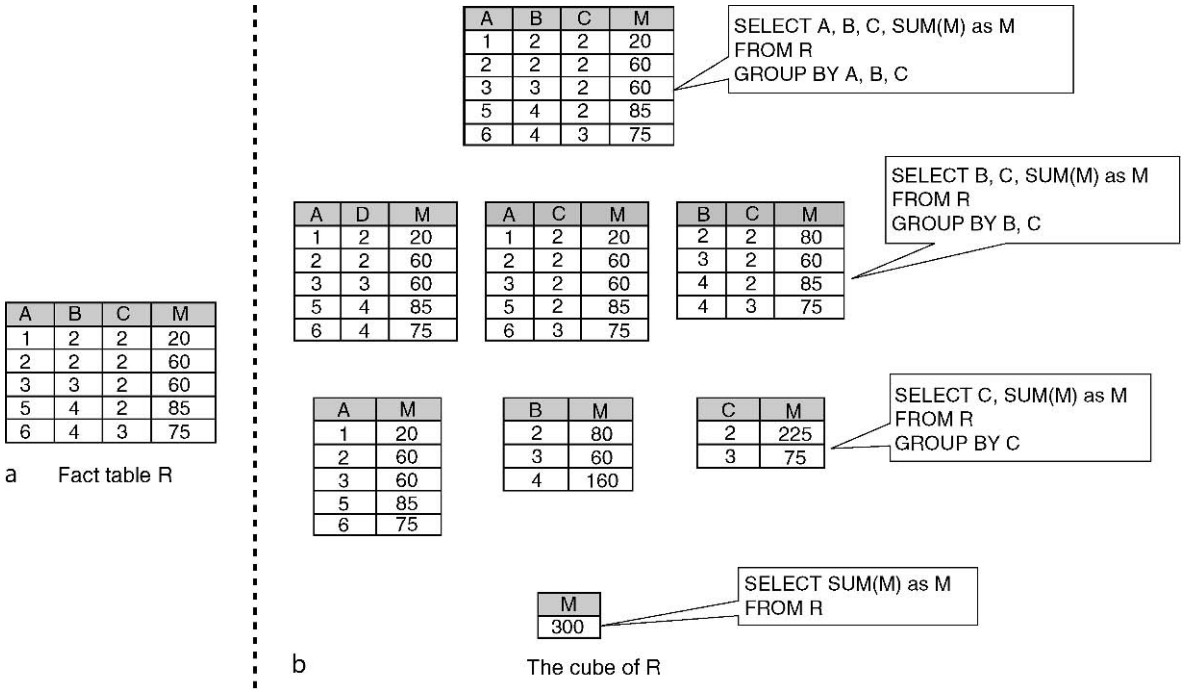
Cube implementation involves the procedures of computation, storage, and manipulation of a data cube, which is a disk structure that stores the results of the aggregate queries that group the tuples of a fact table on all possible combinations of its dimension attributes. For example in Fig. 1a, assuming that R is a fact table that consists of three dimensions (A, B, C) and one measure M (see definitional entry for *Measure*), the corresponding cube of R appears in Fig. 1b. Each cube node (i.e., view that belongs to the data cube) stores the results of a particular aggregate query as shown in Fig. 1b. Clearly, if D denotes the number of dimensions of a fact table, the number of all possible aggregate queries is 2^D; hence, in the worst case, the size of the data cube is exponentially larger with respect to D than the size of the original fact table. In typical applications, this may be in the order of gigabytes or even terabytes, implying that the development of efficient algorithms for the implementation of cubes is extremely important.

Let *grouping attributes* be the attributes of the fact table that participate in the group-by clause of an aggregate query expressed in SQL. A common representation of the data cube that captures the computational dependencies among all the aggregate queries that are necessary for its materialization is the cube lattice [6]. This is a directed acyclic graph (DAG) where each node represents an aggregate query q on the fact table and is connected via a directed edge with every other node whose corresponding group-by part is missing one of the grouping attributes of q. For example, Fig. 2 shows the cube lattice that corresponds to the fact table R (Fig. 1a).

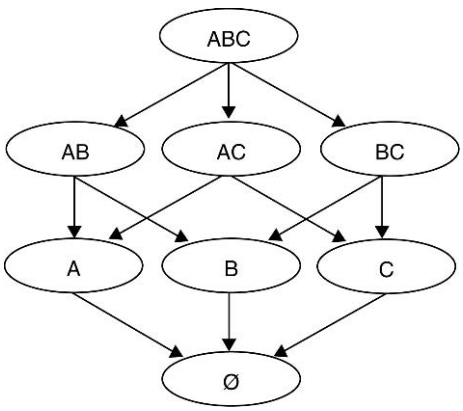
Note that precomputing and materializing parts of the cube is crucial for the improvement of query-response times as well as for accelerating operators that are common in On-Line Analytical Processing (OLAP), such as drill-down, roll-up, pivot, and slice-and-dice, which make an extensive use of aggregation [3]. Materialization of the entire cube seems ideal for efficiently accessing aggregated data; nevertheless, in real-world applications, which typically involve large volumes of data, it may be considerably expensive in terms of storage space, as well as computation and maintenance time. In the existing literature, several efficient methods have been proposed that attempt to balance the aforementioned tradeoff between query-response times and other resource requirements. Their brief presentation is the main topic of this entry.

Historical Background

Most data analysis efforts, whether manual by analysts or automatic by specialized algorithms, manipulate the contents of database systems in order to discover trends and correlations. They typically involve complex queries that make an extensive use of aggregation in order to group together tuples that "behave in a similar fashion." The response time of such queries over extremely large data warehouses can be prohibitive. This problem inspired Gray et al. [3] to introduce the data-cube operator and propose its off-line computation and storage for efficiency at query time. The corresponding seminal publication has been the seed for a plethora of papers thereafter, which have dealt with several different aspects of the lifecycle of a data cube, from cube construction and storage to indexing, query answering, and incremental maintenance.



Cube Implementations. Figure 1. Fact table R and the corresponding data cube.



Cube Implementations. Figure 2. Example of a cube lattice.

Taking into account the format used for the computation and storage of a data cube, the cube-implementation algorithms that have appeared in the literature can be partitioned into four main categories: Relational-OLAP (ROLAP) algorithms exploit traditional materialized views in RDBMSes; Multidimensional-OLAP (MOLAP) algorithms take advantage of multidimensional arrays; Graph-Based methods use specialized graph structures; finally, approximation algorithms use various in-memory representations, e.g., histograms.

The literature also deals with the rest of the cubes lifecycle [12]. Providing fast answers to OLAP aggregate queries is the main purpose of implementing data cubes to begin with, and various algorithms have been proposed to handle different types of queries on the formats above. Moreover, as data stored in the original fact table changes, data cubes must follow suit; otherwise, analysis of obsolete data may result into invalid conclusions. Periodical reconstruction of the entire cube is impractical, hence, incremental-maintenance techniques have been proposed.

The ideal implementation of a data cube must address efficiently all aspects of cube functionality in order to be viable. In the following section, each one of these aspects is further examined separately.

Foundations

In the following subsections, the main stages of the cube lifecycle are analyzed in some detail, including subcube selection, computation, query processing, and incremental maintenance. Note that the references given in this section are only indicative, since the number of related publications is actually very

large. A more comprehensive survey may be found elsewhere [11].

Subcube Selection

In real-world applications, materialization of the entire cube is often extremely expensive in terms of computation, storage, and maintenance requirements, mainly because of the typically large fact-table size and the exponential number of cube nodes with respect to the number of dimensions. To overcome this drawback, several existing algorithms select an appropriate subset of the data cube for precomputation and storage [4,5,6]. Such selection algorithms try to balance the tradeoff between response times of queries (sometimes of a particular, expected workload) and resource requirements for cube construction, storage, and maintenance. It has been shown [6] that selection of the optimum subset of a cube is an NP-complete problem. Hence, the existing algorithms use heuristics in order to find near-optimal solutions.

Common constraints used during the selection process involve constraints on the time available for cube construction and maintenance, and/or on the space available for cube storage. As for the criteria that are (approximately) optimized during selection, they typically involve some form of the benefit gained from the materialization of a particular cube subset.

A particularly beneficial criterion for the selection problem that needs some more attention, since it has been integrated in some of the most efficient cube-implementation algorithms (including Dwarf [17] and CURE [10], which will be briefly presented below) is the so-called redundancy reduction. Several groups of researchers have observed that a big part of the cube data is usually redundant [7,8,10,12,17,20]. Formally, a value stored in a cube is *redundant* if it is repeated multiple times in the same attribute in the cube. For example, in Fig. 1b, tuples $\langle 1, 20\rangle$ of node A, $\langle 1, 2, 20\rangle$ of AB, and $\langle 1, 2, 20\rangle$ of AC are redundant, since they can be produced by properly projecting tuple $\langle 1, 2, 2, 20\rangle$ of node ABC. By appropriately avoiding the storage of such redundant data, several existing cube-implementation algorithms achieve the construction of compressed cubes that can still be considered as fully materialized. Typically, the decrease in the final cube size is impressive, a fact that benefits the performance of computation as well, since output costs are considerably reduced and sometimes, because early identification of redundancy allows pruning of parts of the computation. Furthermore, during query answering, aggregation and decompression are not necessary; instead, some simple operations, e.g., projections, are enough.

Finally, for some applications (e.g., for mining multidimensional association rules), accessing the tuples of the entire cube is not necessary, because they only need those group-by tuples with an aggregate value (e.g. count) above some prespecified minimum support threshold (minsup). For such cases, the concept of *iceberg cubes* has been introduced [2]. Iceberg-cube construction algorithms [2,16] take into consideration only sets of tuples that aggregate together giving a value greater than minsup. Hence, they perform some kind of subcube selection, by storing only the tuples that satisfy the aforementioned condition.

Cube Computation

Cube computation includes scanning the data of the fact table, aggregating on all grouping attributes, and generating the contents of the data cube. The main goal of this procedure is to place tuples that aggregate together (i.e., tuples with identical grouping-attribute values) in contiguous positions in main memory, in order to compute the required aggregations with as few data passes as possible. The most widely used algorithms that accomplish such clustering of tuples are sorting and hashing. Moreover, nodes connected in the cube lattice (Fig. 2) exhibit strong computational dependencies, whose exploitation is particularly beneficial for the performance of the corresponding computation algorithms. For instance, assuming that the data in the fact table R (Fig. 1a) is sorted according to the attribute combination ABC, one can infer that it is also sorted according to both AB and A as well. Hence, the overhead of sorting can be shared by the computation of multiple aggregations, since nodes ABC $\rightarrow$AB $\rightarrow$A $\rightarrow\emptyset$ can be computed with the use of pipelining without reclustering the data. Five methods that take advantage of such node computational dependencies have been presented in the existing literature [1] in order to improve the performance of computation algorithms: smallest-parent, cache-results, amortize-scans, share-shorts, and share-partitions.

Expectedly, both sort-based and hash-based aggregation methods perform more efficiently when the data they process fits in main memory; otherwise,

they are forced to use external-memory algorithms, which generally increase the I/O overhead by a factor of two or three. In order to overcome such problems, most computation methods initially apply a step that partitions data into segments that fit in main memory, called *partitions* [2,10,15]. Partitioning algorithms distribute the tuples of the fact table in accordance with the principle that tuples that aggregate together must be placed in the same partition. Consequently, they can later process each partition independently of the others, since by construction, tuples that belong to different partitions do not share the same grouping-attribute values.

In addition to the above, general characteristics of cube-computation algorithms, there are some further details that are specific to each of four main categories mentioned above (i.e., ROLAP, MOLAP, Graph-Based, and Approximate), which are touched upon below.

ROLAP algorithms store a data cube as a set of materialized relational views, most commonly using either a *star* or a *snowflake schema*. Among these algorithms, algorithm CURE [10] seems to be the most promising, since it is the only solution with the following features: It is purely compatible with the ROLAP framework, hence its integration into any existing relational engine is rather straightforward. Also, it is suitable not only for "flat" datasets but also for processing datasets whose dimension values are hierarchically organized. Furthermore, it introduces an efficient algorithm for external partitioning that allows the construction of cubes over extremely large volumes of data whose size may far exceed the size of main memory. Finally, it stores cubes in a compressed form, removing all types of redundancy from the final result.

MOLAP algorithms store a data cube as a multidimensional array, thereby avoiding to store the dimension values in each array cell, since the position of the cell itself determines these values. The main drawback of this approach comes from the fact that, in practice, cubes have a large number of empty cells (i.e., cubes are sparse), rendering MOLAP algorithms inefficient with respect to their storage-space requirements. To overcome this problem, the so-called chunk-based algorithms have been introduced [21], which avoid the physical storage of most of the empty cells, storing only *chunks*, which are nonempty subarrays. ArrayCube [21] is the most widely accepted algorithm in this category. It has also served as an inspiration to algorithm MM-Cubing [16], which applies similar techniques just to the dense areas of the cube, taking into account the distribution of data in a way that avoids chunking.

Graph-Based algorithms represent a data cube as some specialized graph structure. They use such structures both in memory, for organizing data in a fashion that accelerates computation of the corresponding cube, and on disk, for compressing the final result and reducing storage-space requirements. Among the algorithms in this category, Dwarf [17] seems to be the strongest overall, since it is the only one that guarantees a polynomial time and space complexity with respect to dimensionality [18]. It is based on a highly compressed data structure that eliminates prefix and suffix redundancies efficiently. Prefix redundancy occurs when two or more tuples in the cube share the same prefix, i.e., the same values in the left dimensions; suffix redundancy, which is in some sense complementary to prefix redundancy, occurs when two or more cube tuples share the same suffix, i.e., the same values in the right dimensions and the aggregate measures. An advantage of Dwarf, as well as of the other graph-based methods, is that not only does its data structure store a data cube compactly, but it also serves as an index that can accelerate selective queries.

Approximate algorithms assume that data mining and OLAP applications do not require fine grained or absolutely precise results in order to capture trends and correlations in the data; hence, they store an approximate representation of the cube, trading accuracy for level of compression. Such algorithms exploit various techniques, inspired mainly from statistics, including histograms [14], wavelet transformations [19], and others.

Finally, note that some of the most popular industrial cube implementations include Microsoft SQL Server Analysis Services (http://www.microsoft.com/sql/technologies/analysis/default.mspx) and Hyperion Essbase, which has been bought by ORACLE in 2007 (http://www.oracle.com/hyperion).

Query Processing

The most important motivation for cube materialization is to provide low response times for OLAP queries. Clearly, construction of a highly-compressed cube is useless if the cube format inhibits good query answering performance. Therefore, efficiency during query processing should be taken into consideration as well when selecting a specific cube-construction algorithm and its

corresponding storage format. Note that the latter determines to a great extent the access methods that can be used for retrieving data stored in the corresponding cube; hence, it strongly affects performance of query processing algorithms over cube data.

Intuitively, it seems that brute-force storage of an entire cube in an uncompressed format behaves best during query processing: in this case, every possible aggregation for every combination of dimensions is precomputed and the only cost required is that of retrieving the data stored in the lattice nodes participating in the query. On the other hand, query processing over compressed cubes seems to induce additional overhead for on-line computation or restoration of (possibly redundant) tuples that have not been materialized in the cube.

Nevertheless, the literature has shown that the above arguments are not always valid in practice. This is mostly due to the fact that indexing an uncompressed cube is nontrivial in real-world applications, whereas applying custom indexing techniques for some sophisticated, more compact representations has been found efficient [2]. Furthermore, storing data in specialized formats usually offers great opportunities for unique optimizations that allow a wide variety of query types to run faster over compressed cubes [2]. Finally, recall that several graph-based algorithms, e.g., Dwarf [17], store the cube in a way that is efficient with respect to both storage space and query processing time.

Incremental Maintenance

As mentioned earlier, in general, fact tables are dynamic in nature and change over time, mostly as new records are inserted in them. Aggregated data stored in a cube must follow the modifications in the corresponding fact table; otherwise, query answers over the cube will be inaccurate.

According to the most common scenario used in practice, data in a warehouse is periodically updated in a batch fashion. Clearly, the window of time that is required for the update process must be kept as narrow as possible. Hence, reconstruction of the entire cube from scratch is practically not a viable solution; techniques for incremental maintenance must be used instead.

Given a fact table, its corresponding cube, and a set of updates to the fact table that have occurred since the last cube update, let *delta cube* be the cube formed by the data corresponding to these updates. Most incremental-maintenance algorithms proposed in the literature for the cube follow a common strategy [20]: they separate the update process into the *propagation* phase, during which they construct the delta cube, and the *refresh* phase, during which they merge the delta cube and the original cube, in order to generate the new cube. Most of them identify the refresh phase as the most challenging one and use specialized techniques to accelerate it, taking into account the storage format of the underlying cube (some examples can be found in the literature [12,17]). There is at least one general algorithm, however, that tries to optimize the propagation phase [9]. It selects particular nodes of the delta cube for construction and properly uses them in order to update all nodes of the original cube.

Key Applications

Efficient implementation of the data cube is essential for OLAP applications in terms of performance, since they usually make an extensive use of aggregate queries.

Cross-references

- ▶ Data Warehouse
- ▶ Dimension
- ▶ Hierarchy
- ▶ Measure
- ▶ OLAP
- ▶ Snowflake Schema
- ▶ Star Schema

Recommended Reading

1. Agarwal S., Agrawal R., Deshpande P., Gupta A., Naughton J.F., Ramakrishnan R., and Sarawagi S. On the computation of multidimensional aggregates. In Proc. 22th Int. Conf. on Very Large Data Bases, 1996, pp. 506–521.
2. Beyer K.S. and Ramakrishnan R. Bottom-up computation of sparse and iceberg CUBEs. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 359–370.
3. Gray J., Bosworth A., Layman A., and Pirahesh H. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-total. In Proc. 12th Int. Conf. on Data Engineering, 1996, pp. 152–159.
4. Gupta H. Selection of views to materialize in a data warehouse. In Proc. 6th Int. Conf. on Database Theory, 1997, pp. 98–112.
5. Gupta H. and Mumick I.S. Selection of views to materialize under a maintenance cost constraint. In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 453–470.
6. Harinarayan V., Rajaraman A., and Ullman J.D. Implementing data cubes efficiently. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 205–216.

7. Kotsis N. and McGregor D.R. Elimination of redundant views in multidimensional aggregates. In Proc. 2nd Int. Conf. Data Warehousing and Knowledge Discovery, 2000, pp. 146–161.
8. Lakshmanan L.V.S., Pei J., and Zhao Y. QC-Trees: an efficient summary structure for semantic OLAP. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 64–75.
9. Lee K.Y. and Kim M.H. Efficient incremental maintenance of data cubes. In Proc. 32nd Int. Conf. on Very Large Data Bases, 2006, pp. 823–833.
10. Morfonios K. and Ioannidis Y. CURE for cubes: cubing using a ROLAP engine. In Proc. 32nd Int. Conf. on Very Large Data Bases, 2006, pp. 379–390.
11. Morfonios K., Konakas S., Ioannidis Y., and Kotsis N. ROLAP implementations of the data cube. ACM Comput. Surv., 39(4), 2007.
12. Morfonios K. and Ioannidis Y. Supporting the Data cube Lifecycle: the Power of ROLAP. VLDB J., 17(4):729–764, 2008.
13. Mumick I.S., Quass D., and Mumick B.S. Maintenance of data cubes and summary tables in a warehouse. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 100–111.
14. Poosala V. and Ganti V. Fast approximate answers to aggregate queries on a data cube. In Proc. 11th Int. Conf. on Scientific and Statistical Database Management, 1999, pp. 24–33.
15. Ross K.A. and Srivastava D. Fast computation of sparse datacubes. In Proc. 23th Int. Conf. on Very Large Data Bases, 1997, pp. 116–125.
16. Shao Z., Han J., and Xin D. MM-Cubing: computing iceberg cubes by factorizing the lattice Space. In Proc. 16th Int. Conf. on Scientific and Statistical Database Management, 2004, pp. 213–222.
17. Sismanis Y., Deligiannakis A., Roussopoulos N., and Kotidis Y. Dwarf: shrinking the PetaCube. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 464–475.
18. Sismanis Y. and Roussopoulos N. The complexity of fully materialized coalesced cubes. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 540–551.
19. Vitter J.S. and Wang M. Approximate computation of multidimensional aggregates of sparse data using wavelets. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 193–204.
20. Wang W., Feng J., Lu H., and Yu J.X. Condensed cube: an efficient approach to reducing data cube size. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 155–165.
21. Zhao Y., Deshpande P., and Naughton J.F. An array-based algorithm for simultaneous multidimensional aggregates. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 159–170.

Cube Materialization

► Cube Implementations

Cube Precomputation

► Cube Implementations

Curation

► Biomedical Scientific Textual Data Types and Processing

Current Date

► Now in Temporal Databases

Current Semantics

Michael H. Böhlen[1], Christian S. Jensen[2], Richard T. Snodgrass[3]
[1]Free University of Bozen-Bolzano, Bolzano, Italy
[2]Aalborg University, Aalborg, Denmark
[3]University of Arizona, Tucson, AZ, USA

Synonyms

Temporal upward compatibility

Definition

Current semantics constrains the semantics of non-temporal statements applied to temporal databases. Specifically, current semantics requires that non-temporal statements on a temporal database behave as if applied to the non-temporal database that is the result of taking the timeslice of the temporal database as of the current time.

Key Points

Current semantics [3] requires that queries and views on a temporal database consider the current information only and work exactly as if applied to a non-temporal database. For example, a query to determine who manages the high-salaried employees should consider the current database state only. Constraints and assertions also work exactly as before: they are applied to the current state and checked on database modification.

Database modifications are subject to the same constraint as queries: they should work exactly as if applied

to a non-temporal database. Database modifications, however, also have to take into consideration that the current time is constantly moving forward. Therefore, the effects of modifications must persist into the future (until overwritten by a subsequent modification).

The definition of current semantics assumes a timeslice operator $\tau[t](D^t)$ that takes the snapshot of a temporal database D^t at time t. The timeslice operator takes the snapshot of all temporal relations in D^t and returns the set of resulting non-temporal relations.

Let *now* be the current time [2] and let t be a time point that does not exceed *now*. Let D^t be a temporal database instance at time t. Let $M_1,...,M_n$, $n \geq 0$ be a sequence of non-temporal database modifications.

Let Q be a non-temporal query. Current semantics requires that for all Q, t, D^t, and $M_1,...,M_n$ the following equivalence holds:

$$Q(M_n(M_{n-1}(...(M_1(D^t)...)))) = Q(M_n(M_{n-1}(...(M_1(\tau[now](D^t)))...)))$$

Note that for $n = 0$ there are no modifications, and the equivalence becomes $Q(D^t) = Q(\tau[now](D^t))$, i.e., a non-temporal query applied to a temporal database must consider the current database state only.

An unfortunate ramification of the above equivalence is that temporal query languages that introduce new reserved keywords not used in the non-temporal languages they extend will violate current semantics. The reason is that the user may have previously used such a keyword as an identifier (e.g., a table name) in the database. To avoid being overly restrictive, it is reasonable to consider current semantics satisfied even when reserved words are added, as long as the semantics of all statements that do not use the new reserved words is retained by the temporal query language.

Temporal upward compatibility [1] is a synonym that focuses on settings where the original temporal database is the result of rendering a non-temporal database temporal.

Cross-references

- ▶ Nonsequenced Semantics
- ▶ Now in Temporal Databases
- ▶ Sequenced Semantics
- ▶ Snapshot Equivalence
- ▶ Temporal Database
- ▶ Temporal Data Models
- ▶ Temporal Query Languages
- ▶ Timeslice Operator

Recommended Reading

1. Bair J., Böhlen M.H., Jensen C.S., and Snodgrass R.T. Notions of upward compatibility of temporal query languages. Wirtschaftsinformatik, 39(1):25–34, February 1997.
2. Clifford J., Dyreson C., Isakowitz T., Jensen C.S., and Snodgrass R.T. On the Semantics of "NOW" in Databases. ACM Trans. Database Syst., 22:171–214, June 1997.
3. Snodgrass R.T. Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann, Los Altos, CA, 1999.

Current Time

- ▶ Now in Temporal Databases

Current Timestamp

- ▶ Now in Temporal Databases

Curse of Dimensionality

Lei Chen
Hong Kong University of Science and Technology, Hong Kong, China

Synonyms

Dimensionality curse

Definition

The *curse of dimensionality*, first introduced by Bellman [1], indicates that the number of samples needed to estimate an arbitrary function with a given level of accuracy grows exponentially with respect to the number of input variables (i.e., dimensionality) of the function.

For similarity search (e.g., nearest neighbor query or range query), the *curse of dimensionality* means that the number of objects in the data set that need to be accessed grows exponentially with the underlying dimensionality.

Key Points

The *curse of dimensionality* is an obstacle for solving dynamic optimization problems by backwards induction. Moreover, it renders machine learning problems complicated, when it is necessary to learn a state-of-nature from finite number data samples in a high dimensional feature space. Finally, the *curse of dimensionality* seriously affects the query performance for similarity search over multidimensional indexes because, in high dimensions, the distances from a query to its nearest and to its farthest neighbor are similar. This indicates that data objects tend to be close to the boundaries of the data space with the increasing dimensionality. Thus, in order to retrieve even a few answers to a nearest neighbor query, a large part of the data space should be searched, making the multidimensional indexes less efficient than a sequential scan of the data set, typically with dimensionality greater than 12 [2]. In order to break the *curse of dimensionality*, data objects are usually reduced to vectors in a lower dimensional space via some dimensionality reduction technique before they are indexed.

Cross-references

► Dimensionality Reduction

Recommended Reading

1. Bellman R.E. Adaptive Control Processes. Princeton University Press, Princeton, NJ, 1961.
2. Beyer K.S., Goldstein J., Ramakrishnan R., Shaft U. When is "Nearest Neighbor" Meaningful? In Proc. 7th Int. Conf. on Database Theory, 1999, pp. 217–235.

Cursor

► Iterator

CW Complex

► Simplicial Complex

CWM

► Common Warehouse Metamodel

Cyclic Redundancy Check (CRC)

► Checksum and Cyclic Redundancy Check (CRC) Mechanism