M

MAC

► Message Authentication Codes

Machine Learning in Bioinformatics

► Machine Learning in Computational Biology

Machine Learning in Computational Biology

Cornelia Caragea, Vasant Honavar
Iowa State University, Ames, IA, USA

Synonyms

Data mining in computational biology; Data mining in bioinformatics; Machine learning in bioinformatics; Machine learning in systems biology; Data mining in systems biology

Definition

Advances in high throughput sequencing and "omics" technologies and the resulting exponential growth in the amount of macromolecular sequence, structure, gene expression measurements, have unleashed a transformation of biology from a data-poor science into an increasingly data-rich science. Despite these advances, biology today, much like physics was before Newton and Leibnitz, has remained a largely descriptive science. Machine learning [6] currently offers some of the most cost-effective tools for building predictive models from biological data, e.g., for annotating new genomic sequences, for predicting macromolecular function, for identifying functionally important sites in proteins, for identifying genetic markers of diseases, and for discovering the networks of genetic interactions that orchestrate important biological processes [3]. Advances in machine learning e.g., improved methods for learning from highly unbalanced datasets, for learning complex structures of class labels (e.g., labels linked by directed acyclic graphs as opposed to one of several mutually exclusive labels) from richly structured data such as macromolecular sequences, three-dimensional molecular structures, and reliable methods for assessing the performance of the resulting models, are critical to the transformation of biology from a descriptive science into a predictive science.

Historical Background

Large scale genome sequencing efforts have resulted in the availability of hundreds of complete genome sequences. More importantly, the GenBank repository of nucleic acid sequences is doubling in size every 18 months [4]. Similarly, structural genomics efforts have led to a corresponding increase in the number of macromolecular (e.g., protein) structures [5]. At present, there are over a thousand databases of interest to biologists [16]. The emergence of high-throughput "omics" techniques, e.g., for measuring the expression of thousands of genes under different perturbations, has made possible system-wide measurements of biological variables [8]. Consequently, discoveries in biological sciences are increasingly enabled by machine learning.

Some representative applications of machine learning in computational and systems biology include: identifying the protein-coding genes (including gene boundaries, intron-exon structure) from genomic DNA sequences; predicting the function(s) of a protein from its primary (amino acid) sequence (and when available, structure and its interacting partners); identifying functionally important sites (e.g., protein-protein, protein-DNA, protein-RNA binding sites, post-translational modification sites) from the protein's amino acid sequence and, when available, from the protein's structure; classifying protein sequences (and structures) into structural classes; Identifying functional modules (subsets of genes that function together) and genetic networks from gene expression data.

These applications collectively span the entire spectrum of machine learning problems including supervised learning, unsupervised learning (or cluster analysis), and system identification. For example, protein function prediction can be formulated as a supervised learning problem: given a dataset of protein sequences with experimentally determined function labels, induce a classifier that correctly labels a novel protein sequence. The problem of identifying functional modules from gene expression data can be formulated as an unsupervised learning problem: given expression measurements of a set of genes under different conditions (e.g., perturbations, time points), and a distance metric for measuring the similarity or distance between expression profiles of a pair of genes, identify clusters of genes that are co-expressed (and hence are likely to be co-regulated). The problem of constructing gene networks from gene expression data can be formulated as a system identification problem: given expression measurements of a set of genes under different conditions (e.g., perturbations, time points), and available background knowledge or assumptions, construct a model (e.g., a boolean network, a bayesian network) that explains the observed gene expression measurements and predicts the effects of experimental perturbations (e.g., gene knockouts).

Foundations

Challenges presented by computational and systems biology applications have driven, and in turn benefited from, advances in machine learning. Some of these developments are described below.

Multi-Label Classification: In the traditional classification problem, an instance $\mathbf{x}_i$, i = 1,...,n, is associated with a single class label y_j from a finite, disjoint set of class labels Y , j = 1,...,k, k = |Y| (*single-label classification problem*). If the set Y has only two elements, then the problem is referred to as the *binary classification problem*. Otherwise, if Y has more than two elements, then it is referred to as *multi-class classification problem*. However, in many biological applications, an instance $\mathbf{x}_i$ is associated with a subset of, not necessarily disjoint, class labels in Y (*multi-label classification problem*). For example, many genes and proteins are multi-functional. Most of the existing algorithms cannot simultaneously label a gene or protein with several, not necessarily mutually exclusive functions. Each instance is then assigned to a subset of nodes in the hierarchy, yielding a *hierarchical multi-label classification problem* or a *structured output classification problem*. The most common approach to dealing with *multi-label classification problem* [7] is to transform the problem into k binary classification problems, one for each different label $y_j \in Y$, j = 1,...,k. The transformation consists of constructing k datasets, D_j, each containing all instances of the original dataset, such that an instance in D_j, j = 1,...,k, is labeled with 1 if it has label y_j in the original dataset, and 0 otherwise. During classification, for a new unlabeled instance $\mathbf{x}_{test}$, each individual classifier C_j,j = 1,...,k, returns a prediction that $\mathbf{x}_{test}$ belongs to the class label y_j or not. However, the transformed datasets that result from this approach are highly unbalanced, typically, with the number of positively labeled instances being significantly smaller than the number of negatively labeled instances, requiring the use of methods that can cope with unbalanced data. Alternative evaluation metrics need to be developed for assessing the performance of multi-label classifiers. This task is complicated by correlations among the class labels.

Learning from Unbalanced Data: Many of the macromolecular sequence classification problems present the problem of learning from highly *unbalanced* data. For example, only a small fraction of amino acids in an RNA-binding protein binds to RNAs. Classifiers that are trained to optimize accuracy generally perform rather poorly on the minority class. Hence, if accurate classification of instances from the minority class is important (or equivalently, the false positives and false negatives have unequal costs or risks associated with them), it is necessary to change the distribution of positive and negative instances *during training* by randomly selecting a subset of the training data for the majority class, or alternatively, assigning different *weights* to positive and negative samples (and learn from the resulting weighted samples). More recently, *ensemble classifiers* [11] have been shown to improve the performance of sequence classifiers on unbalanced datasets. Unbalanced datasets also complicate both the training and the assessment of the predictive performance of classifiers. *Accuracy* is not a useful performance measure in such scenarios. Indeed, no single performance measure provides a complete picture of the classifier's performance. Hence, it is much more useful to examine ROC (Receiver Operating Characteristic) or precision-recall curves [3]. Of particular interest are methods that can directly optimize alternative performance measures that take

into account the unbalanced nature of the dataset and user-specified tradeoff between false positive and false negative rates.

Data Representation: Many computational and systems biology applications of machine learning present challenges in data representation. Consider for example, the problem of identifying functionally important sites (e.g., RNA-binding residues) from amino acid sequences. In this case, given an amino acid sequence, the classifier needs to assign a binary label (1 for an RNA-binding residue and 0 for a non RNA-binding residue) to each letter of the sequence. To solve this problem using standard machine learning algorithms that work with a fixed number of input features, it is fairly common to use a *sliding window* approach [12] to generate a collection of fixed length windows, where each window corresponds to the target amino acid and an equal number of its sequence neighbors on each side. The classifier is trained to label the target residue. Similarly, identifying binding sites from a three-dimensional structure of the protein requires transforming the problem into one that can be handled by a traditional machine learning method. Such transformations, while they allow the use of existing machine learning methods on macromolecular sequence and structure labeling problems, complicate the task of assessing the performance of the resulting classifier (see below).

Performance Assessment: Standard approaches to assessing the performance of classifiers rely on k-fold cross-validation wherein a dataset is partitioned into k disjoint subsets (folds). The performance measure of interest is estimated by averaging the measured performance of the classifier on k runs of a cross-validation experiment, each using a different choice of the $k - 1$ subsets for training and the remaining subset for testing the classifier. The fixed length window representation described above complicates this procedure on macromolecular sequence labeling problems. The training and test sets obtained by random partitioning of the dataset of labeled windows can contain windows that originate from the same sequence, thereby violating a critical requirement for cross-validation, namely, that the training and test data be disjoint. The resulting overlap between training and test data can yield overly optimistic estimates of performance of the classifier. A better alternative is to perform sequence-based (as opposed to window-based) cross-validation by partitioning the set of sequences (instead of windows) into disjoint folds. This procedure guarantees that training and test sets are indeed disjoint [9]. Obtaining realistic estimates of performance in sequence classification and sequence labeling problems also requires the use of *non-redundant* datasets [13].

Learning from Sparse Datasets: In gene expression datasets, the number of genes is typically in the hundreds or thousands, whereas the number of measurements (conditions, perturbations) is typically fewer than ten. This presents significant challenges in inferring genetic network models from gene expression data because the number of variables (genes) far exceeds the number of observations or data samples. Approaches to dealing with this challenge require reducing the effective number of variables via variable selection [17] or abstraction i.e., by grouping variables into clusters that behave similarly under the observed conditions. Another approach to dealing with sparsity of data in such settings is to incorporate information from multiple datasets [18].

Key Applications

Protein Function Prediction: Proteins are the principal catalytic agents, structural elements, signal transmitters, transporters and molecular machines in cells. Understanding protein function is critical to understanding diseases and ultimately in designing new drugs. Until recently, the primary source of information about protein function has come from biochemical, structural, or genetic experiments on individual proteins. However, with the rapid increase in number of genome sequences, and the corresponding growth in the number of protein sequences, the numbers of experimentally determined structures and functional annotations has significantly lagged the number of protein sequences. With the availability of datasets of protein sequences with experimentally determined functions, there is increasing use of sequence or structural homology-based transfer of annotation from already annotated sequences to new protein sequences. However, the effectiveness of such homology-based methods drops dramatically when the sequence similarity between the target sequence and the reference sequence falls below 30%. In many instances, the function of a protein is determined by conserved local sequence motifs. However, approaches that assign function to a protein based on the presence of a single motif (the so-called characteristic motif) fail to take advantage of multiple sequence motifs that

are correlated with critical structural features (e.g., binding pockets) that play a critical role in protein function. Against this background, machine learning methods offer an attractive approach to training classifiers to assign putative functions to protein sequences. Machine learning methods have been applied, with varying degrees of success, to the problem of protein function prediction. Several studies have demonstrated that machine learning methods, used in conjunction with traditional sequence or structural homology based techniques and sequence motif-based methods outperform the latter in terms of accuracy of function prediction (based on cross-validation experiments). However, the efficacy of alternative approaches in genome-wide prediction of functions of protein-coding sequences from newly sequenced genomes remains to be established. There is also significant room for improving current methods for protein function prediction.

Identification of Potential Functional Annotation Errors in Genes and Proteins: As noted above, to close the sequence-function gap, there is an increasing reliance on automated methods in large-scale genome-wide annotation efforts. Such efforts often rely on transfer of annotations from previously annotated proteins, based on sequence or structural similarity. Consequently, they are susceptible to several sources of error including errors in the original annotations from which new annotations are inferred, errors in the algorithms, bugs in the software used to process the data, and clerical errors on the part of human curators. The effect of such errors can be magnified because they can propagate from one set of annotated sequences to another. Because of the increasing reliance of biologists on reliable functional annotations for formulation of hypotheses, design of experiments, and interpretation of results, incorrect annotations can lead to wasted effort and erroneous conclusions. Hence, there is an urgent need for computational methods for checking consistency of such annotations against independent sources of evidence and detecting potential annotation errors. A recent study has demonstrated the usefulness of machine learning methods to *identify and correct* potential annotation errors [1].

Identification of Functionally Important Sites in Proteins: Protein-protein, protein-DNA, and protein-RNA interactions play a pivotal role in protein function. Reliable identification of such interaction sites from protein sequences has broad applications ranging from rational drug design to the analysis of metabolic and signal transduction networks. Experimental detection of interaction sites must come from determination of the structure of protein-protein, protein-DNA and protein-RNA complexes. However, experimental determination of such complexes lags far behind the number of known protein sequences. Hence, there is a need for development of reliable computational methods for identifying functionally important sites from a protein sequence (and when available, its structure, but not the complex). This problem can be formulated as a sequence (or structure) labeling problem. Several groups have developed and applied, with varying degrees of success, machine learning methods for identification of functionally important sites in proteins (see [21,14,22] for some examples). However, there is significant room for improving such methods.

Discovery and Analysis of Gene and Protein Networks: Understanding how the parts of biological systems (e.g., genes, proteins, metabolites) work together to form dynamic functional units, e.g., how genetic interactions and environmental factors orchestrate development, aging, and response to disease, is one of the major foci of the rapidly emerging field of systems biology [8]. Some of the key challenges include the following: uncovering the biophysical basis and essential macromolecular sequence and structural features of macromolecular interactions; comprehending how temporal and spatial clusters of genes, proteins, and signaling agents correspond to genetic, developmental and regulatory networks [10]; discovering topological and other characteristics of these networks [19]; and explaining the emergence of systems-level properties of networks from the interactions among their parts. Machine learning methods have been developed and applied, with varying degrees of success, in learning predictive models including boolean networks [20] and bayesian networks [15] from gene expression data. However, there is significant room for improving the accuracy and robustness of such algorithms by taking advantage of multiple types of data and by using active learning.

Future Directions

Although many machine learning algorithms have had significant success in computational biology, several challenges remain. These include the development of: efficient algorithms for learning predictive models from distributed data; cumulative learning algorithms

that can efficiently update a learned model to accommodate changes in the underlying data used to train the model; effective methods for learning from sparse, noisy, high-dimensional data; and effective approaches to make use of the large amounts of unlabeled or partially labeled data; algorithms for learning predictive models from disparate types of data: macromolecular sequence, structure, expression, interaction, and dynamics; and algorithms that leverage optimal experiment design with active learning in settings where data is expensive to obtain.

Cross-references

- ▶ Biological Networks
- ▶ Biostatistics and Data Analysis
- ▶ Classification
- ▶ Clustering
- ▶ Data Mining
- ▶ Graph Database Mining

Recommended Reading

1. Andorf C., Dobbs D., and Honavar V. Exploring inconsistencies in genome-wide protein function annotations: a machine learning approach. BMC Bioinform., 8:284, 2007.
2. Ashburner M., Ball C.A., Blake J.A., Botstein D., Butler H., Cherry J.M., Davis A.P., Dolinski K., Dwight S.S., Eppig J.T., Harris M.A., Hill D.P., Issel-Tarver L., Kasarskis A., Lewis S., Matese J.C., Richardson J.E., Ringwald M., Rubin G.M., and Sherlock G. Gene ontology: tool for the unification of biology. Nat. Gene., 25:25–29, 2000.
3. Baldi P. and Brunak S. Bioinformatics: the machine learning approach. MIT, Cambridge, MA, 2001.
4. Benson D.A., Karsch-Mizrachi I., Lipman D.J., Ostell J., and Wheeler D.L. Genbank. Nucleic Acids Res., 35D (Database issue):21–D25, 2007.
5. Berman H.M., Westbrook J., Feng Z., Gilliland G., Bhat T.N., Weissig H., Shindyalov I.N., and Bourne P.E. The protein data bank. Nucleic Acids Res., 28:235–242, 2000.
6. Bishop C.M. Pattern Recognition and Machine Learning. Springer, Berlin, 2006.
7. Boutell M.R., Luo J., Shen X., and Brown C.M. Learning multi-label scene classification. Pattern Recogn., 37:1757–1771, 2004.
8. Bruggeman F.J. and Westerhoff H.V. The nature of systems biology. Trends Microbiol., 15:15–50, 2007.
9. Caragea C., Sinapov J., Dobbs D., and Honavar V. Assessing the performance of macromolecular sequence classifiers. In Proc. IEEE 7th Int. Symp. on Bioinformatics and Bioengineering, 2007, pp. 320–326.
10. de Jong H. Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol., 9:67–103, 2002.
11. Diettrich T.G. Ensemble methods in machine learning. Springer, Berlin, In Proc. 1st Int. Workshop on Multiple Classifier Systems, 2000, pp. 1–15.
12. Diettrich T.G. Machine learning for sequential data: a review. In Proc. Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2002, pp. 15–30.
13. El-Manzalawy Y., Dobbs D., and Honavar V. On evaluating MHC-II binding peptide prediction methods, PLoS One, 3(9): e3268, 2008.
14. El-Manzalawy Y., Dobbs D., and Honavar V. Predicting linear B-cell epitopes using string kernels. J. Mole. Recogn., 21243–255, 2008.
15. Friedman N., Linial M., Nachman I., and Pe'er D. Using bayesian networks to analyze expression data. J. Comput. Biol., 7:601–620, 2000.
16. Galperin M.Y. The molecular biology database collection: 2008 update. Nucleic Acids Res., 36:D2–D4, 2008.
17. Guyon I. and Elisseeff A. An introduction to variable and feature selection. J. Mach. Learn. Res., 3:1157–1182, 2003.
18. Hecker L., Alcon T., Honavar V., and Greenlee H. Querying multiple large-scale gene expression datasets from the developing retina using a seed network to prioritize experimental targets. Bioinform. Biol. Insights, 2:91–102, 2008.
19. Jeong H., Tombor B., Albert R., Oltvai Z.N., and Barabasi A.-L. The large-scale organization of metabolic networks. Nature, 407:651–654, 1987.
20. Lahdesmaki H., Shmulevich I., and Yli-Harja O. On learning gene regulatory networks under the boolean network model. Mach. Learn., 52:147–167, 2007.
21. Terribilini M., Lee J.-H., Yan C., Jernigan R.L., Honavar V, and Dobbs D. Predicting RNA-binding sites from amino acid sequence. RNA J., 12:1450–1462, 2006.
22. Yan C., Terribilini M., Wu F., Jernigan R.L., Dobbs D., and Honavar V. Identifying amino acid residues involved in protein-DNA interactions from sequence. BMC Bioinform., 7:262, 2006.

Machine Learning in Systems Biology

▶ Machine Learning in Computational Biology

Machine-Readable Dictionary (MRD)

▶ Electronic Dictionary

Macro

▶ Snippet

Magnetic Disk

► Disk

Maid

► Massive Array of Idle Disks

Main Memory

Peter Boncz
CWI, Amsterdam, The Netherlands

Synonyms

Primary memory; Random access memory (RAM)

Definition

Primary storage, presently known as main memory, is the largest memory directly accessible to the CPU in the prevalent Von Neumann model and stores both data and instructions (program code). The CPU continuously reads instructions stored there and executes them. Also called Random Access Memory (RAM), to indicate that load/store instructions can access data at any location at the same cost, it is usually implemented using DRAM chips, which are connected to the CPU and other peripherals (disk drive, network) via a bus.

Key Points

The earliest computers used tubes, then transistors and since the 1970s in integrated circuits. RAM chips generally store a bit of data in either the state of a flip-flop, as in SRAM (static RAM), or as a charge in a capacitor (or transistor gate), as in DRAM (dynamic RAM). Some types have circuitry to detect and/or correct random faults called memory errors in the stored data, using parity bits or error correction codes (ECC). RAM of the read-only type, ROM, instead uses a metal mask to permanently enable/disable selected transistors, instead of storing a charge in them.

The main memory available to a program in most operating systems, while primarily relying on RAM, can be increased by disk memory. That is, the memory access instructions supported by a CPU work on so-called virtual memory, where an abstract virtual memory space is divided into pages. At any time, a page either resides in a swap-file on disk or in RAM, where it must be in order for the CPU to access it. When memory is accessed, the Memory Management Unit (MMU) of the CPU transparently translates the virtual address into its current physical address. If the memory page is not in RAM, it generates a page fault, to be a handled by the OS which then has to perform I/O to the swap file. If a high percentage of the memory access generates a page fault, this is called thrashing, and severely lowers performance.

Over the past decades, the density of RAM chips has increased, following a planned evolution of finer chip production process sizes, popularly known as "Moore's Law." This has led to an increase in RAM capacity as well as bandwidth. Access latency has also decreased, however, the physical distance on the motherboard between DRAM chips and CPU results in a minimum access latency of around 50ns (real RAM latencies are often higher). In current multi-GHz CPUs this means that a memory access instruction takes hundreds of cycles to execute. Typically, a high percentage of instructions in a program can be memory access instructions (up to 33%) and the RAM latency can seriously impact performance. This problem is known as the "memory wall."

To counter the performance problems of the memory wall, modern computer architecture now features a memory hierarchy that besides DRAM also includes SRAM cache memories, typically located on the CPU chip. Memory access instructions transfer memory in units of cache-lines, typically 64 bytes at a time (this cache line size is also related to the width of the memory bus). Memory access instruction first checks whether the accessed cache line is in the highest (fastest/smallest) L1 cache. This takes just a few CPU cycles. If a cache miss occurs, the memory access instruction checks the next cache level. Only if no cache contains the cache line, memory access is performed. Therefore, like virtual memory page thrashing, the CPU cache hit ratio achieved by a program now materially affects performance.

While in the past access to the DRAM chips over the bus was typically performed by a chipset, in between CPU and memory, some modern CPU architectures have moved the memory controller logic onto the CPU chip itself, which tends to reduce access latency. Also, to better serve the memory bandwidth requirement multi-CPU systems, modern architectures often

have a dedicated memory bus between the CPU and DRAM. In a Symmetric Multi-Processing (SMP) this leads to a so-called Non-Uniform Memory Access architecture (NUMA), where access to the memory directly connected to a CPU is faster than access to the memory connected to another CPU.

While database systems traditionally focus on the disk access pattern (i.e., I/O), modern database systems, as well as main-memory database systems (that do not rely on I/O in the first place) now must carefully plan the in-memory data storage format used as well as the memory access patterns caused by query processing algorithms, in order to optimize the use of the CPU caches and avoid high cache miss ratios. The increased RAM sizes as well as the increased impact of I/O latency also leads to a trend to rely more on main memory as the preferred storage medium in database processing.

Cross-references

- ► Cache Memory
- ► CPU
- ► Disk

Main Memory DBMS

Peter Boncz
CWI, Amsterdam, The Netherlands

Synonyms

In-memory DBMS; MMDBMS

Definition

A main memory database system is a DBMS that primarily relies on main memory for computer data storage. In contrast, conventional database management systems typically employ hard disk based persistent storage.

Key Points

The main advantage of MMDBMS over normal DBMS technology is superior performance, as I/O cost is no more a performance cost factor. With I/O as main optimization focus eliminated, the architecture of main memory database systems typically aims at optimizing CPU cost and CPU cache usage, leading to different data layout strategies (avoiding complex tuple representations) as well as indexing structures (e.g., B-trees with lower-fan-outs with nodes of one or a few CPU cache lines).

While built on top of volatile storage, most MMDB products offer ACID properties, via the following mechanisms: (i) Transaction Logging, which records changes to the database in a journal file and facilitates automatic recovery of an in-memory database, (ii) Non-volatile RAM, usually in the form of static RAM backed up with battery power (battery RAM), or an electrically erasable programmable ROM (EEPROM). With this storage, the MMDB system can recover the data store from its last consistent state upon reboot, (iii) High availability implementations that rely on database replication, with automatic failover to an identical standby database in the event of primary database failure.

Main-memory database systems were originally popular in real-time systems (used in e.g., telecommunications) for their fast and more predictable performance, and this continues to be the case. However, with increasing RAM sizes allowing more problems to be addressed using a MMDBMS, this technology is proliferating into many other areas, such as on-line transaction systems, and recently in decision support. Main memory database systems are also deployed as drop-in systems that intercept read-only queries on cached data from an existing disk-based DBMS, thus reducing its workload and providing fast answers to a large percentage of the workload.

Examples of main-memory database systems are MonetDB, SolidDB, TimesTen and DataBlitz. MySQL offers a main-memory backend based on Heap tables. The MySQL Cluster product is a parallel main memory system that offers ACID properties through high availability.

Cross-references

- ► Disk
- ► Main Memory
- ► Processor Cache

Recommended Reading

1. Bohannon P., Lieuwen D.F., Rastogi R., Silberschatz A., Seshadri S., and Sudarshan S. The architecture of the dalí main-memory storage manager. Multimedia Tools Appl., 4(2):115–151, 1997.
2. Boncz P.A. and Kersten M.L. MIL primitives for querying a fragmented world. VLDB J., 8(2):101–119, 1999.
3. DeWitt D.J., Katz R.H., Olken F., Shapiro L.D., Stonebraker M., and Wood D.A. Implementation techniques for main memory

database systems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1984, pp. 1–8.
4. Hvasshovd S-O., Torbjørnsen Ø., Bratsberg S.E., and Holager P. The ClustRa telecom database: high availability, high throughput, and real-time response. In Proc. 21th Int. Conf. on Very Large Data Bases, 1995, pp. 469–477.

Maintenance of Materialized Views with Outer-Joins

Per-Åke Larson
Microsoft Corporation, Redmond, WA, USA

Definition

An materialized outer-join view is a materialized view whose defining expression contains at least one outer join. View maintenance refers to the process of bringing the view up to date after one or more of the underlying base tables has been updated. View maintenance can always be done by recomputing the result, known as a full refresh, but this is usually prohibitively expensive. Incremental view maintenance, that is, only applying the minimal changes required to bring the view up to date, is normally more efficient.

Historical Background

Full outer join (called generalized join) was proposed by Lacroix and Pirotte in 1976 [4]. During the 1980's, there was considerable discussion in the research literature about the use and power of outer joins. Commercial systems began supporting outer joins in the late 1980's and at the time of writing (2007) all major commercial systems do. Optimization of outer-join queries was an active research area during the 1990's. Outer join was first included in the 1992 SQL standard. The first view matching algorithm for outer-join views was published by Larson and Zhou [5] in 2005.

In 1998 Griffin and Kumar [2] published the first paper covering incremental maintenance of materialized outer-join views. A paper 2006 by Gupta and Mumick [3] described a more efficient procedure but, unfortunately, it does not always produce the correct result. In 2007, Larson and Zhou [6] introduced an method for efficient incremental maintenance of outer-join views. At the time of writing, only Oracle allows (a limited form of) materialized outer-join views.

Foundations

Larson and Zhou [6] showed that incremental maintenance of an outer-join view can be divided into two steps: computing and applying a *primary delta* and a *secondary delta*. The first step is very similar to maintaining an inner-join view while the second step is a "clean-up" step.

This entry describes Larson's and Zhou's maintenance procedure for a view without aggregation when the update consists of insertions into one of its base tables. The reader is referred to the original paper [6] for a more complete description of how to handle deletions, views with aggregation, and how to exploit foreign-key constraints to simplify maintenance. Examples illustrating the procedure use a database consisting of the following three tables. Primary keys are underlined.

```
O(Okey, Odate, Ocustomer),
L(okey, pkey, Qty, Price),
P(Pkey, Pname).
```

The following materialized view consisting of two full outer joins will be used as a running example.

$$MV = \left(L \overset{fo}{\underset{p(l,p)}{\bowtie}} P\right) \overset{fo}{\underset{p(l,o)}{\bowtie}} O$$

where the join predicates are defined as $p(l,p) \equiv (l.pkey = p.pkey)$ and $p(l,o) \equiv (l.okey = o.okey)$.

Join-Disjunctive Normal Form

The view maintenance procedure builds on the join-disjunctive normal form for outer-join expressions introduced by Galindo-Legaria [6]. The normal form is described in this section by an example; more details can be found in [6,1].

Let T_1 and T_2 be tables with schemas S_1 and S_2, respectively. The *outer union*, denoted by $T_1 \uplus T_2$, first null-extends (pads with nulls) the tuples of each operand to schema $S_1 \cup S_2$ and then takes the union of the results (without duplicate elimination).

Let t_1 and t_2 be tuples with the same schema. Tuple t_1 is said to *subsume* tuple t_2 if t_1 agrees with t_2 on all columns where they both are non-null and t_1 contains fewer null values than t_2. The operator *removal of subsumed tuples* of T, denoted by $T\downarrow$, returns the tuples of T that are not subsumed by any other tuple in T.

The *minimum union* of tables T_1 and T_2 is defined as $T_1 \oplus T_2 = (T_1 \uplus T_1)\downarrow$. Minimum union is both commutative and associative.

Left outer join can be rewritten as $T_1 \bowtie_p^{lo} T_2 = T_1 \bowtie {}_pT_2 \oplus T_1$ and right outer join as $T_1 \bowtie_p^{ro} T_2 = T_1 \bowtie {}_pT_2 \oplus T_2$. Full outer join can be rewritten as $T_1 \bowtie_p^{fo} T_2 = T_1 \bowtie {}_pT_2 \oplus T_1 \oplus T_2$.

The example view was defined as

$$MV = \left(L \bowtie_{p(l,p)}^{fo} P\right) \bowtie_{p(l,p)}^{fo} O.$$

Conversion to normal form is done bottom up by applying the rewrite rules above. First rewrite the join between L and P in terms of inner joins and minimum union, which yields

$$MV = (\sigma_{p(l,p)}(L \times P) \oplus L \oplus P) \bowtie_{p(l,o)}^{fo} O.$$

Then apply the same rewrite to the second outer join, which produces

$$MV = ((\sigma_{p(l,p)}(L \times P) \oplus L \oplus P) \bowtie_{(l,o)} O) \oplus (\sigma_{p(l,p)}(L \times P) \oplus L \oplus P) \oplus O.$$

Inner join distributes over minimum union in the same way as over regular union. Applying this transformation to the join with O produces

$$MV = \sigma_{p(l,p)\wedge p(l,o)}(O \times L \times P) \oplus \sigma_{p(l,o)}(O \times L) \oplus \sigma_{p(l,o)}(O \times P) \oplus \sigma_{p(l,p)}(L \times P) \oplus L \oplus P \oplus O.$$

The view expression is now in join-disjunctive form but it can be further simplified. The term $\sigma_{p(l,o)}(O \times P)$ can be eliminated because the join predicate will never be satisfied.

$$MV = \sigma_{P(l,p)\wedge P(l,o)}(O \times L \times P) \oplus \sigma_{p(l,o)}(O \times L) \oplus \sigma_{p(l,p)}(L \times P) \oplus L \oplus P \oplus O.$$

The normal form shows what form of tuples are found in MV. For example, it "contains" all tuples in the join of O and L. Most such tuples are represented implicitly by being included in a wider tuple composed of tuples from O, L and P; only the non-subsumed tuples are stored explicitly in the view.

As illustrated by this example, an outer-join expression E over a set of tables $\mathcal{U}$ can be converted to a normal form consisting of the minimum union of terms composed from selections and inner joins (but no outer joins). More formally, the join-disjunctive normal form of E equals

$$E = E_1 \oplus E_2 \oplus \cdots \oplus E_n$$

where each term E_i is of the form $E_i = \sigma_{pi}(T_{i1} \times T_{i2} \times \ldots \times T_{im})$. $\mathcal{T}_i = \{T_{i1}, T_{i2} \ldots T_{im}\}$ is a (unique) subset of the tables in $\mathcal{U}$. Predicate p_i is the conjunction of a subset of the selection and join predicates found in the original form of the query.

The Subsumption Graph

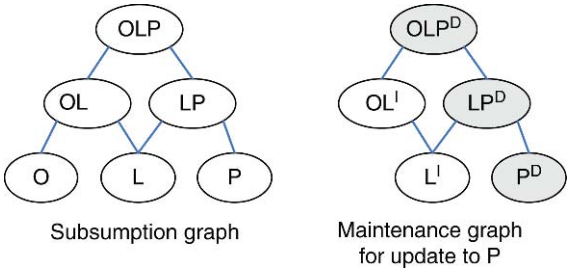
Every term in the normal form of the view has a unique set of source tables drawn from $\mathcal{U}$ and is null-extended on all other tables in the view. The set of source tables of term E_i is denoted by $\mathcal{T}_i$ and the set of tables on which it is is null-extended by $\mathcal{S}_i, \mathcal{S}_i = \mathcal{U} - \mathcal{T}i$.

A tuple produced by a term with source tables $\mathcal{T}_i$ can only be subsumed by tuples produced by terms whose source set is a superset of $\mathcal{T}_i$. The subsumption relationships among terms can be modeled by a DAG called the subsumption graph.

The *subsumption graph* of E contains a node n_i for each term E_i in the normal form and the node is labeled with the source table set $\mathcal{T}_i$ of E_i. There is an edge from a node n_i to a node n_j, if $\mathcal{T}_i$ is a minimal superset of $\mathcal{T}_j$. $\mathcal{T}_i$ is a minimal superset of $\mathcal{T}_j$ if there does not exist a node n_k in the graph such that $\mathcal{T}_j \subset \mathcal{T}_k \subset \mathcal{T}_i$.

The subsumption graph for view MV is shown to the left in Fig. 1. The importance of the subsumption graph lies in the following observation: when checking whether a tuple of a term is subsumed, it is sufficient to check against tuples in the term's immediate parent terms. For example, to determine whether a P tuple p_1 is subsumed, all that is needed is to check whether it joins with an L tuple. If it does, the resulting tuple, which subsumes p_1, is included in the LP term.

The result of an outer-join expression is represented in a minimal form. Only the non-subsumed



Maintenance of Materialized Views with Outer-Joins.
Figure 1. Subsumption graph and maintenance graph for view *MV*.

tuples produced by a term E_i in the normal form are explicitly represented. A subsumed tuple is represented implicitly by being included in a subsuming tuple. The *net contribution* of a term, denoted by D_i, is the set of non-subsumed tuples of term E_i in the normal form of expression E. Then E can then be written in the form

$$E = D_1 \uplus D_2 \uplus \cdots \uplus D_n.$$

Consider a view V and suppose one of the its base tables T is modified. This may affect the net contribution of a term D_i in one of three ways:

1. Directly, which occurs if T is among the tables in $\mathcal{T}_i$;
2. Indirectly, which occurs if T is not among the tables in $\mathcal{T}_i$ but it is among the source tables of at least one of its parent nodes;
3. No effect, otherwise.

Based on this classification of how terms are affected, a *view maintenance graph* is created as follows.

1. Eliminate from the subsumption graph all nodes that are unaffected by the update of T.
2. Mark the remaining nodes by D or I depending on whether the node is affected directly or indirectly.

The maintenance graph for view when updating P is shown to the right in Fig. 1. The maintenance graph is used primarily to identify which terms of a view are indirectly affected and thus may require maintenance.

Maintenance Procedure

Suppose table T has been updated. If so, any view V that references T needs to be maintained. The first step is to compute the view's maintenance graph and classify the terms as directly affected, indirectly affected, and unaffected. Without loss of generality, assume that the view has n terms, of which terms $1, 2,...,k$ are directly affected, terms $k + 1$, $k + 2,...,k+ m$ are indirectly affected, and terms $k + m + 1$, $k + m + 2,...,n$ are not affected. The view expression can then be rewritten in the form

$$V = V^D \uplus V^I \uplus V^U \text{where}$$
$$V^D = \uplus_{i=1}^{k} D_i,\ V^I = \uplus_{i=k+1}^{k+m} D_i,$$
$$V^U = \uplus_{i=k+m+1}^{n} D_i.$$

From this form of the expression, one can see that to update the view, two delta expressions must be evaluated and applied to the view

$$\Delta V^D = \uplus_{i=1}^{k} \Delta D_i,\ \Delta V^I = \uplus_{i=k+1}^{k+m} \Delta D_i.$$

ΔV^D is called the *primary delta* and ΔV^I the *secondary delta*. In summary, maintenance of a view V after insertions into one of its underlying base tables can be performed in two steps.

1. Compute the primary delta ΔV^D and insert the resulting tuples into the view.
2. If there are indirectly affected terms, compute the secondary delta ΔV^I and delete the resulting tuples from the view.

Computing the Primary Delta

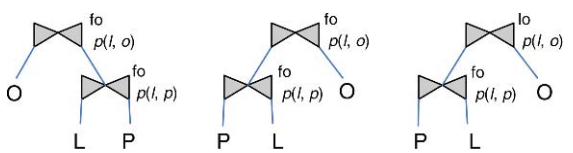
An expression that computes the primary delta, ΔV^D, can be constructed by the following simple algorithm.

1. Traverse the operator tree for V along the path from T to the root. On any join operator encountered, apply commutativity rules to ensure that the input referencing T is on the left.
2. Traverse the path from T to the root of V. Convert any full outer join operator encountered to a left outer join and any right outer join operator to an inner join.
3. Substitute T by ΔT.

Step 1 is a normal rewrite of the view expression and does not change the result. Step 2 modifies the expression so that it computes only V^D. After Step 2, the operators on the path from T to the root consists only of selects, inner joins, and left outer joins and the delta expression is always the left input.

Figure 2 illustrates the transformation process for the example view MV when table P is updated. The resulting expression for computing the primary delta is

$$\Delta MV^D = \left(\Delta P \bowtie^{lo}_{p(l,p)} L\right) \bowtie^{lo}_{p(l,o)} O$$



Maintenance of Materialized Views with Outer-Joins. Figure 2. Constructing primary-delta expression for insertions into table P.

Computing the Secondary Delta

The secondary delta can be computed efficiently from the primary delta and either the view or base tables. Only the case when using the view is described here. Recall that the base tables have already been updated and the primary delta has been applied to the view.

The primary delta ΔV^D contains the union of the deltas for all directly affected terms. However, deltas for individual terms are needed to compute the secondary delta. Each term is defined over a unique set of tables and null extended on all others so tuples from a particular term are easily identified and can be extracted from ΔV^D by simple selection predicates.

Let *null*(T) denote a predicate that evaluates to true if a tuple is null-extended on table T. *null*(T) can be implemented in SQL as "$T.c$ is null" where c is any column of T that does not contain nulls, for example, a column of a key. When applying *null* and $\neg$*null* to a set of tables $T = \{T_1, T_2..., T_n\}$, the shorthand notations $n(\mathcal{T}) = \bigwedge_{T_i \in \mathcal{T}} null(T_i)$ and $nn(\mathcal{T}) = \bigwedge_{T_i \in \mathcal{T}} \neg null(T_i)$ are used.

For the example view, MV, the primary delta contains deltas of three directly affected terms, see Fig. 1. Non-subsumed tuples from, for example, the LP-term are uniquely identified by the fact that they are composed of a real tuple from L and from P but are null extended on O. Hence, ΔD_{LP} can be extracted from ΔV^D as follows:

$$\Delta D_{LP} = \pi_{(LP).*}\sigma_{nn(LP)\wedge n(o)}\Delta MV^D$$

where $nn(LP) = \neg null(L) \wedge \neg null(P)$ and $n(O) = null(O)$.

ΔD_{LP} contains only the delta of the net contribution of the term. ΔE_{LP} contains the complete delta of the term, including both subsumed and non-subsumed tuples. Tuples in ΔE_{LP} are composed of real tuples from L, and from P, and may or may not be null extended on O. Hence, ΔE_{LP} can be extracted from ΔV^D as follows:

$$\Delta E_{LP} = \delta\pi_{(LPo).*}\sigma_{nn}(LP)\Delta MV^D.$$

The duplicate elimination (δ) is necessary because an LP tuple may have joined with multiple O tuples.

Continuing with the running example, the secondary delta consists of ΔD_{OL} and ΔD_L. ΔD_{OL} is null extended on P and the OLP-term is its only parent so it can be computed as:

$$\Delta D_{OL} = \sigma_{nn(OL)\wedge n(P)}(MV + \Delta MV^D) \ltimes^{ls}_{eq(OL)} \sigma_{nn(OLP)}\Delta MV^D.$$

This expression makes sense intuitively. The first part selects from the view all orphaned (non-subsumed) tuples of term E_{OL} contained in the view after the primary delta has been applied. The second part extracts from the primary delta all tuples added to the parent term E_{OLP}. The join is a left semijoin and outputs every tuple from the left operand that joins with one or more tuples in the right operand. The complete expression thus amounts to finding all currently orphaned tuples of the term and retaining those that cease to be orphans because of the insert. Those tuples should be deleted from the view.

D_L is null extended on O, and P and has one directly affected parent, the LP-term. ΔD_L can be computed as:

$$\Delta D_L = \sigma_{nn(L)\wedge n(OP)}(MV + \Delta MV^D) \ltimes^{ls}_{eq(L)} \sigma_{nn}(LP)\Delta MV^D.$$

Summary

In summary, after insertion into table P of a set of tuples ΔP, the example view MV can be brought up to date as follows. First compute the primary delta

$$\Delta MV^D = (\Delta P \bowtie^{lo}_{p(l,p)} L) \bowtie^{lo}_{p(l,o)} O$$

and insert the resulting tuples into the view, resulting in $MV+\Delta MV^D$. Then compute the secondary delta

$$\begin{aligned}\Delta MV^I &= \Delta D_{OL} \uplus \Delta D_L \\ &= \sigma_{nn(OL)\wedge n(P)}(MV + \Delta MV^D) \\ &\quad \ltimes^{ls}_{eq(OL)\,\sigma_{nn}(OLP)} \Delta MV^D \uplus \\ &\quad \sigma_{nn(L)\wedge n(OP)}(MV + \Delta MV^D) \\ &\quad \ltimes^{ls}_{eq(L)\,\sigma_{nn}(LP)} \Delta MV^D\end{aligned}$$

and delete the resulting tuples from the view.

Key Applications

Queries containing outer joins are often used in analysis queries over large tables in data warehouses. Materialized outer-join views, especially when aggregated, can be very beneficial in such scenarios.

Cross-references

- ▶ Materialized Views
- ▶ Maintenance of Materialized Views with Outer-Joins
- ▶ Views

Recommended Reading

1. Galindo-Legaria C. Outerjoins as disjunctions. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1994, pp. 348–358.
2. Griffin T. and Kumar B. Algebraic change propagation for semijoin and outerjoin queries. ACM SIGMOD Rec., 27(3):22–27, 1998.
3. Gupta A. and Mumick I.S. Incremental maintenance of aggregate and outerjoin expressions. Inf. Syst., 31(6):435–464, 2006.
4. Lacroix M. and Pirotte A. Generalized joins. ACM SIGMOD Rec., 8(3):14–15, 1976.
5. Larson P. and Zhou J. View matching for outer-join views. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 445–456.
6. Larson P. and Zhou J. Efficient maintenance of materialized outer-join views. In Proc. 23rd Int. Conf. on Data Engineering, 2007, pp. 56–65.

Maintenance of Recursive Views

Suzanne W. Dietrich
Arizona State University, Phoenix, AZ, USA

Synonyms

Incremental maintenance of recursive views; Recursive view maintenance

Definition

A view is a derived or virtual table that is typically defined by a query, providing an abstraction or an alternate perspective of the data that allows for more intuitive query specifications using these views. Each reference to the view name results in the retrieval of the view definition and the recomputation of the view to answer the query in which the view was referenced. When views are materialized, the tuples of the computed view are stored in the database with appropriate index structures so that subsequent access to the view can efficiently retrieve tuples to avoid the cost of recomputing the entire view on subsequent references to the view. However, the materialized view must be updated if any relation that it depends on has changed. Rather than recomputing the entire view on a change, an incremental view maintenance algorithm uses the change to incrementally compute updates to the materialized view in response to that change. A recursive view is a virtual table definition that depends on itself. A canonical example of a recursive view is the transitive closure of a relationship stored in the database that can be modeled as directed edges in a graph. The transitive closure essentially determines the reachability relationship between the nodes in the graph. Typical examples of transitive closure include common hierarchies such as employee-supervisor, bill-of-materials (parts-subparts), ancestor, and course prerequisites. The incremental view maintenance algorithms for the maintenance of recursive views have additional challenges posed by the recursive nature of the view definition.

Historical Background

A view definition relates a view name to a query defined in the query language of the database. Initially, incremental view maintenance algorithms were explored in the context of non-recursive view definitions involving select, project, and join query expressions, known as SPJ expressions in the literature. The power of recursive views was first introduced in the *Datalog* query language, which is a declarative logic programming language established as the database query language for deductive databases in the 1980s. Deductive databases assume the theoretical foundations of relational data but use Datalog as the query language. Since its relational foundations assume first normal form, Datalog looks like a subset of the Prolog programming language without function symbols. However, Datalog does not assume Prolog's top-down left-to-right programming language evaluation strategy. The evaluation of Datalog needed to be founded on the fundamentals of database query optimization. In a database system, a user need only specify a correct declarative query, and it is the responsibility of the database system to efficiently execute that specification. The evaluation of Datalog was further complicated by the fact that Datalog allows for relational views that include union and recursion in the presence of negation. Therefore, the view definitions in Datalog were more expressive than the traditional select-project-join views available in relational databases at that time. Therefore, the incremental view maintenance algorithms for recursive views in the early 1990s are typically formulated in the context of the evaluation of Datalog. The power to define a recursive union in SQL was added in the SQL:1999 standard.

Historically, it is important to note that the incremental maintenance of recursive views is related to the areas of integrity constraint checking and condition monitoring in active databases. These three areas were being explored in the research literature at around the same time. In integrity constraint checking, the

database is assumed to be in a consistent state and when a change occurs in the database, it needs to incrementally determine whether the database is still in a consistent state. In active databases, the database is responsible for actively checking whether a condition that it is responsible for monitoring is now satisfied by incrementally evaluating condition specifications affected by changes to the database. Although closely related, there are differences in the underlying assumptions for these problems.

Foundations

Recursive View Definition

A canonical example of a recursive view definition is the reachability of nodes in a directed graph. In Datalog, the reach view consists of two rules. The first non-recursive rule serves as the base or seed case, and indicates that if the stored or base table edge defines a directed edge from the source node to the destination node, then the destination can be reached from the source. The second rule is recursive. If the source node can reach some intermediate node and there is an edge from that intermediate node to a destination node, then the source can reach the destination.

```
reach(Source, Destination):-
   edge(Source, Destination).
reach(Source, Destination):-
   reach(Source, Intermediate),
   edge(Intermediate, Destination).
```

Intuitively, one can think of the recursive rule as an unfolding of the joins required to compute the reachability of paths of length two, then paths of length three, and so on until the data of the underlying graph is exhausted.

In SQL, this recursive view is defined with the following recursive query expression:

```
with recursive reach(source, destination) as
(select E.source, E.destination
from edge E)
union
(select S.source, D.destination
from reach S, edge D
where S.destination = D.source)
```

SQL limits recursive queries to linear recursions, which means that there is at most one direct invocation of a recursive item. The specification of the reach view above is an example of a linear recursion. There is another linear recursive specification of reach where the direct recursive call appears on the right side of the join versus the left side of the join:

```
reach(Source, Destination):-
   edge(Source, Intermediate),
   reach(Intermediate, Destination).
```

However, there is a logically equivalent specification of reach that is non-linear:

```
reach(Source, Destination):-
   reach(Source, Intermediate),
   reach(Intermediate, Destination).
```

The goal of Datalog evaluation is to allow the user to specify the recursive view declaratively in a logically correct way, and it is the system's responsibility to optimize the evaluation of the query.

SQL also restricts recursions to those defined in deductive databases as stratified *Datalog with negation*. Without negation, a recursive Datalog program has a unique solution that corresponds to the theoretical fixpoint semantics or meaning of the logical specification. In the computation of the reach view, each unfolding of the recursion joins the current instance of the recursive view with the edge relation until no new tuples can be added. The view instance has reached a fixed point and will not change. When negation is introduced, the interaction of recursion and negation must be considered. The concept of stratified negation means that there can be no negation through a recursive computation, i.e., a view cannot be defined in terms of its own negation. Recursive views can contain negation but the negation must be in the context of relations that are either stored or completely computed before the application of the negation. This imposed level of evaluation with respect to negation and recursion are called strata. For stratified Datalog with negation, there also exists a theoretical fixpoint that represents the intuitive meaning of the program.

Consider an example of a view defining a peer as two employees that are not related in the employee-supervisor hierarchy:

```
peer(A, B):- employee(A,...), employee(B,...),
   not (supervisor(A,B)), not(supervisor(B,A)).
supervisor(Emp, Sup):-
   immediateSupervisor(Emp,Sup).
```

```
supervisor(Emp, Sup):-
  supervisor(Emp, S),
  immediateSupervisor(S, Sup).
```

Since peer depends on having supervisor materialized for the negation, peer is in a higher stratum than supervisor. Therefore, the strata provide the levels in which the database system needs to compute views to answer a query.

Evaluation of Recursive Queries

Initial research in the area emphasized the efficient and complete evaluation of recursive queries. The intuitive evaluation of the recursive view that unions the join of the current view instance with the base data at each unfolding is known as a naïve bottom-up algorithm. In a bottom-up approach to evaluating a rule, the known collection of facts is used to satisfy the subgoals on the right-hand side of the rule, generating new facts for the relation on the left-hand side of the rule. To improve the efficiency of the naïve algorithm, a semi-naïve approach can be taken that only uses the new tuples for the recursive view from the last join to use in the join at the next iteration. A disadvantage of this bottom-up approach for evaluating a query is that the entire view is computed even when a query may be asking for a small subset of the data. This eager approach is not an issue in the context of materializing an entire view.

Another recursive query evaluation approach considered a top-down strategy as in Prolog's evaluation. In a top-down approach to evaluation, the evaluation starts with the query and works toward the collection of facts in the database. In the context of the reach recursive view, the reach query is unified with the left-hand side of the non-recursive rule and rewritten as a query involving edge. The edge facts are then matched to provide answers. The second recursive rule is then used to rewrite the reach query with the query consisting of the goals on the right-hand side of the rule. This evaluation process continues, satisfying the goals with facts or rewriting the goals using the rules. The unification of a goal with the left-hand side of a rule naturally filters the evaluation by binding variables in the rule to constants that appear in the query. However, the evaluation of a left-recursive query using Prolog's evaluation strategy enters an infinite loop on cyclic data by attempting to prove the same query over and over again. A logic programmer would not write a logic program that enters an infinite loop, but the deductive database community was interested in the evaluation of truly declarative query specifications.

The resulting evaluation approaches combine the best of top-down filtering with bottom-up materialization. The magic sets technique added top-down filtering by cleverly rewriting the original rules so that a bottom-up evaluation would take advantage of constants appearing in the query [1]. Memoing was added to a top-down evaluation strategy to achieve the duplicate elimination feature that is inherent in a bottom-up evaluation of sets of tuples [3]. This duplicate elimination feature avoids the infinite loops on cyclic data. Top-down memoing is complete for subsets of Datalog on certain types of queries [4]. For stratified Datalog with negation, top-down memoing still requires iteration to guarantee complete evaluation. Further research explored additional optimizations as well as implementations of deductive database systems [12] and led to research in active databases and materialized view maintenance.

Incremental Evaluation of Recursive Views

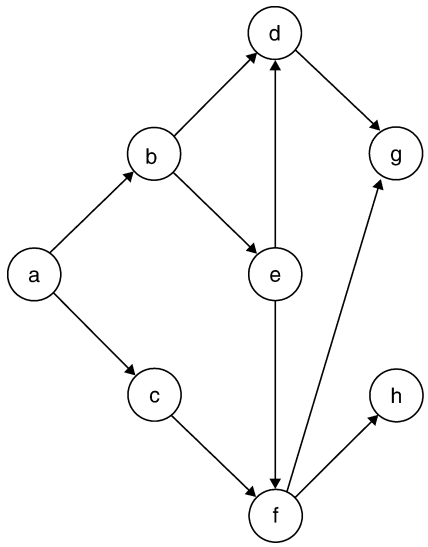
A view maintenance algorithm uses the change to incrementally determine updates to the view. Consider a change in the underlying graph for the transitive closure example. If a new edge is inserted, this edge may result in a change to the materialized reach view by adding a connection between two nodes that did not exist before. However, another possibility is that the new edge added another path between two nodes that were already in the materialized view. A similar situation applies on the removal of an edge. The deletion could result in a change in the reachability between nodes or it could result in the removal of a path but the nodes are still connected via another route. In addition, in the general case, a view may depend on many relations including other (recursive) views in the presence of negation. Therefore, the approaches for the incremental maintenance of recursive views typically involve a propagation or derivation phase that determines an approximation or overestimate of the changes, and a filtering or rederivation phase that checks whether the potential change represents a change to the view. There are differences in the underlying details of how these phases are performed.

The two incremental view maintenance algorithms that will be presented by example are the DRed algorithm [7] and the PF Algorithm [8]. Both the DRed

and PF algorithms handle recursive stratified Datalog programs with negation. There are other algorithms developed for special cases of Datalog programs and queries, such as the counting technique for nonrecursive programs, but this exposition will explore these more general approaches for incremental view maintenance. Historically, the PF algorithm was developed in the context of top-down memoing whereas DRed assumes a bottom-up semi-naïve evaluation. To assist with the comparison of the approaches, the notation introduced for the DRed algorithm [7] will be used to present both algorithms in the context of the transitive closure motivational example.

Figure 1 provides a graphical representation of an edge relation. Assume that the view for reach is materialized, and the edge (e,f) is deleted from the graph. The potential deletions or overestimates for reach, denoted by δ^-(reach), are computed by creating Δ^- rules for each rule computing reach. Each reach rule has k Δ^- rules where k corresponds to the number of subgoals in the body of the rule. The i^{th} Δ^- rule uses the current estimate of deleted tuples (δ^-) for the i^{th} subgoal. For the nonrecursive rule, there is only one subgoal. Therefore, there is only one Δ^- rule indicating that potential edge deletions generate potential deletions to the reach view.

Δ^-(r1): δ^-(reach(S, D)):- δ^-(edge(S, D)).



Maintenance of Recursive Views. Figure 1. Sample Graph.

Since the recursive rule has two subgoals, there are two Δ^- rules:

Δ^-(r21): δ^-(reach(S, D)):- δ^-(reach(S, I)), edge(I, D).
Δ^-(r22): δ^-(reach(S, D)):- reach(S, I), δ^-(edge(I, D)).

Potential deletions to the reach view as well as the edge relation can generate potential deletions to the view.

These potential deletions need to be filtered by determining whether there exist alternative derivations or paths between the nodes computed in the potential deletion. There is a Δ^r rule defined for each reach rule that determines the rederivation of the potential deletions, which is denoted by δ^+ (reach):

Δ^r(r1): δ^+ (reach(S, D)):- δ^-(reach(S, D)), $edge^v$(S, D).
Δ^r(r2): δ^+ (reach(S, D)):- δ^-(reach(S, D)), $reach^v$(S, I), $edge^v$(I, D).

The superscript v on the subgoals in the rule indicates the use of the current instance of the relation corresponding to the subgoal. If the potential deletion is still reachable in the new database instance, then there exists another route between the source and destination, and it should not be removed from the materialized view. The actual removals to reach, indicated by Δ^-(reach), is the set of potential deletions minus the set of alternative derivations:

$$\Delta^-(\text{reach}) = \delta^-(\text{reach}) - \delta^+(\text{reach})$$

Table 1 illustrates the evaluation of the DRed algorithm for incrementally maintaining the reach view on the deletion of edge(e,f) from Fig. 1. The DRed algorithm uses a bottom-up evaluation of the given rules, starting with the deletion δ^-(edge(e, f)). In the first step, the Δ^- rules compute the overestimate of the deletions to reach. The result of the Δ^- rules are shown in the right column, which indicates the potential deletions to reach as δ^-(reach). The second step uses the Δ^r rules to filter the potential deletions. The right column illustrates the source destination pairs that are still reachable after the deletion of edge(e,f) as δ^+ (reach). The tuples that must be removed from the materialized view are indicated by Δ^-(reach): {(e,f) (e,h) (b,f) (b,h)}.

The PF (Propagate Filter) algorithm on the same example is shown in Table 2. PF starts by propagating the edge deletion using the nonrecursive rule, which generates a potential deletion of reach(e,f). This approximation is immediately filtered to determine whether there exists another path between e and f.

Maintenance of Recursive Views. Table 1. DRed algorithm on deletion of edge (e,f) on materialized reach view

DRed algorithm		
Step 1	*Compute overestimate of potential deletions*	δ^-(reach)
	Δ^-(r1): δ^-(reach(S, D)):- δ^-(edge(S, D)).	(e,f)
	Δ^-(r21): δ^-(reach(S, D)):- δ^-(reach(S, I)), edge(I, D).	(e,g) (e,h)
	Δ^-(r22): δ^-(reach(S, D)):- reach(S, I), δ^-(edge(I, D)).	(a,f) (b,f)
	Repeat until no change: No new tuples for Δ^-(r1) and Δ^-(r22)	
	Δ^-(r21): δ^-(reach(S, D)):- δ^-(reach(S, I)), edge(I, D).	(a,g) (a,h) (b,g) (b,h)
	Last iteration does not generate any new tuples	
Step 2	*Find alternative derivations to remove potential deletions*	δ^+ (reach)
	Δ^r(r1): δ^+ (reach(S, D)):- δ^-(reach(S, D)), edge(S, D).	
	Δ^r(r2): δ^+ (reach(S, D)):- δ^-(reach(S, D)), reach^v(S, I), edge^v(I, D).	(e,g) (a,f) (a,g) (a,h) (b,g)
Step 3	*Compute actual changes to reach*	Δ^-(reach)
	Δ^-(reach) = δ^-(reach) – δ^+ (reach)	(e, f) (e,h) (b,f) (b,h)

Maintenance of Recursive Views. Table 2. PF algorithm on deletion of edge (e,f) on materialized reach view

PF algorithm				
Propagate			*Filter*	
	Rule	δ^-(reach)	δ^+ (reach)	Δ^-(reach)
δ^-(edge):{(e, f)}	Δ^-(r1)	(e,f)	{}	(e,f)
Δ^-(reach): {(e,f)}	Δ^-(r21)	(e,g) (e,h)	(e,g)	(e,h)
Δ^-(reach): {(e,h)}	Δ^-(r21)	{}		{}
δ^-(edge): {(e, f)}	Δ^-(r22)	(a,f) (b,f)	(a,f)	(b,f)
Δ^-(reach): {(b,f)}	Δ^-(r21)	(b,g) (b,h)	(b,g)	(b,h)
Δ^-(reach): {(b,h)}	Δ^-(r21)	{}		{}

Since there is no alternate route, the tuple (e,f) is identified as an actual change, and is then propagated. The propagation of Δ^-(reach): {(e,f)} identifies (e,g) and (e,h) as potential deletions. However, the filtering phase identifies that there is still a path from e to g, so (e, h) is identified as a removal to reach. The propagation of (e,h) does not identify any potential deletions. The propagation of the initial edge deletion δ^-(edge): {(e, f)} must be propagated through the recursive rule for reach using Δ^-(r22). The potential deletions are immediately filtered, and only actual changes are propagated. The PF algorithm also identifies the tuples {(e,f) (e,h) (b,f) (b,h)} to be removed from the materialized view.

As shown in the above deletion example, the DRed and PF algorithms both compute overestimates or approximations of tuples to be deleted from the recursive materialized view. The PF algorithm eagerly filters the potential deletions before propagating them. The DRed algorithm propagates the potential deletions within a stratum but filters the overestimates before propagating them to the next stratum. There are scenarios in which the DRed algorithm outperforms the PF algorithm and others in which the PF algorithm outperforms the DRed algorithm.

For the case of insertions, the PF algorithm operates in a manner similar to deletions, by approximating the tuples to be added and filtering the potential additions by determining whether the tuple was provable in the old database state. However, the DRed algorithm uses the bottom-up semi-naïve algorithm for Datalog evaluation to provide an inherent mechanism for determining insertions to the materialized view. In semi-naïve evaluation, the original rules are executed once to provide the seed or base answers. Then incremental versions of the rules are executed until a fixpoint is

reached. The incremental rules are formed by creating k rules associated with a rule where k corresponds to the number of subgoals in the right-hand side of the rule. The i^{th} incremental rule uses only the new tuples from the last iteration for the i^{th} subgoal. However, when the i^{th} subgoal is a stored relation, then the corresponding incremental rules are removed since they will not contribute to the incremental evaluation. For the motivational example, the incremental rule for reach is

Δreach(S,I), edge(I, D)

where Δreach represents the new reach tuples computed on the previous iteration. Since a set of tuples is being computed, duplicate proofs are automatically filtered and are not considered new tuples. This is the inherent memoing in bottom-up evaluation that handles cycles in the underlying data.

Key Applications

Query Optimization; Condition Monitoring; Integrity Constraint Checking; Data Warehousing; Data Mining; Network Management; Mobile Systems.

Cross-references

- ▶ Datalog
- ▶ Incremental Maintenance of Views with Aggregates
- ▶ View Maintenance
- ▶ Maintenance of Materialized Views with Outer-Joins

Recommended Reading

1. Bancilhon F., Maier D., Sagiv Y., and Ullman J. Magic sets and other strange ways to implement logic programs. In Proc. 5th ACM SIGACT-SIGMOD Symp. on Principles of Database Systems, 1986, pp. 1–15.
2. Ceri S. and Widom J. Deriving production rules for incremental view maintenance. In Proc. 17th Int. Conf. on Very Large Data Bases, 1991, pp. 577–589.
3. Dietrich S.W. Extension tables: memo relations in logic programming. In 14th Int. Colloquium on Automata, Languages, and Programming, 1987, pp. 264–272.
4. Dietrich S.W. and Fan C. On the completeness of naive memoing in prolog. New Generation Comput., 15:141–162, 1997.
5. Dong G. and Su J. Incremental maintenance of recursive views using relational calculus/SQL. ACM SIGMOD Rec., 29 (1):44–51, 2000.
6. Gupta A. and Mumick I.S. (eds.). Materialized Views: Techniques, Implementations, and Applications, The MIT Press, Cambridge, MA, 1999.
7. Gupta A., Mumick I.S., and Subrahmanian V.S. Maintaining views incrementally. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 157–166.
8. Harrison J.V. and Dietrich S.W. Maintenance of materialized views in a deductive database: an update propagation approach. In Proc. Workshop on Deductive Databases, 1992, pp. 56–65.
9. Küchenhoff V. On the efficient computation of the difference between consecutive database states. In Proc. 2nd Int. Conf. on Deductive and Object-Oriented Databases, 1991, pp. 478–502.
10. Martinenghi D. and Christiansen H. Efficient integrity constraint checking for databases with recursive views. In Proc. 9th East European Conf. Advances in Databases and Information Systems, 2005, pp. 109–124.
11. Ramakrishnan R. (ed.). Applications of Logic Databases. Kluwer, Norwell, MA, 1995.
12. Ramakrishnan R. and Ullman D. A survey of deductive database systems. J. Logic Programming, 23(2):125–149, 1995.
13. Ullman J. Principles of Database and Knowledge Base Systems, Computer Science Press, Rockville, MD, 1989.
14. Urpí T. and Olivé A. A method for change computation in deductive databases. In Proc. 18th Int. Conf. on Very Large Data Bases, 1992, pp. 225–237.

Managing Compressed Structured Text

Gonzalo Navarro
University of Chile, Santiago, Chile

Synonyms

Searching compressed XML; Compressing XML

Definition

Compressing semi-structured text is the problem of creating a reduced-space representation from which the original data can be re-created exactly. Compared to plain text compression, the goal is to take advantage of the structural properties of the data. A more ambitious goal is being able to manipulate this text in compressed form, without decompressing it. This entry focuses on compressing, navigating, and searching semi-structured text, as those are the areas where more advances have been made.

Historical Background

Modeling data using semi-structured text has been a topic of interest at least since the 1980's, with a significant burst of activity in the 1990's [2]. Since then, the widespread adoption of XML (appearing in 1998, see the current version at http://www.w3.org/TR/xml) as

the standard to represent semi-structured data has unified the efforts of the community around this particular format. Very early, however, the same features that made XML particularly appealing for both human and machine processing were pointed out as significant sources of redundancy and wasting of storage space and bandwidth. This was especially relevant for wireless transmission and triggered the proposal of the WAP Binary XML Content Format as early as 1999 (see http://www.w3.org/TR/wbxml), where simple techniques to compress XML prior to its transmission were devised.

In parallel, there has been a growing interest in not only compressing the data for storage or transmission, but in manipulating it in compressed form. The reason is the long-standing tradeoff between faster/smaller/more expensive and slower/larger/cheaper memories. A more compact data representation has the potential of fitting in a faster memory, where manipulating it can be orders of magnitudes faster, even if it requires more operations, than a naive representation fitting only in a slower memory.

Foundations

For concreteness, this entry will focus on the de-facto standard XML, where the structure is a tree or a forest marked with beginning and ending tags in the text. In fact this encompasses many other semi-structured text proposals, hence most of the material of the entry applies to semi-structured text in general, with minimal changes. In XML, the tags can have attributes and associated values, and there might be available a grammar giving the permissible context-free syntax of the semi-structured document.

Compression of Semi-Structured Text

An obvious approach to compressing semi-structured text is to regard it as plain text and use any of the well-known text compression methods [5]. Yet, considering the structure might yield improved compression performance compared to ignoring it. Many compressors have been proposed trying to exploit structure in different ways. Rather than describing them individually, the main principles behind them will be presented.

1. The data is a mix of structure and content. The structure can be regarded as a labeled tree, where the labels are the tag names, and the content as free text, which can appear between every consecutive pair of tree nodes, and within tree leaves. Attribute information can be handled as text as well, or as special data attached to tree nodes.
2. The structure and the content can be compressed separately, which has proved to give good results. Later, encoded tags and contents can be stored in the file in their original order, so that the document can be handled as a plain uncompressed document. Alternatively, structure and content can be stored separately with some pointer information to reconstruct the tree, in which case the structure pointers may help to point out relevant content to scan in the querying process.
3. The text content can be compressed using any text compression method. Semi-static compressors permit accessing the content at random without decompressing all from the beginning, whereas adaptive compressors tend to achieve better compression ratios. Splitting the text into blocks that are compressed adaptively permits trading random access time for compression ratio.
4. The structure can be compressed in several ways, which can range from a simple scheme of assigning numbers to the different tag names, to sophisticated grammar-based compression methods. The latter may take advantage of the explicit grammar when it is available.
5. Structure can be used, in addition, to boost compression. If the text contents are grouped according to the structural path towards the root, and each group is compressed separately, compression ratios improve noticeably. This can be as simple as grouping texts that are under the same tag (that is, considering only the deepest tree node containing the text) or as sophisticated as considering the full path towards the root.

A sample of different open-source systems that compress XML based on diverse combinations of these principles is

XMill [13],
Millau [10],
XMLPPM [6],
XGrind [15],
XCQ [12],
XPress [14], and
SCM [1].

See http://pages.cpsc.ucalgary.ca/~gleighto/research/xml-comp.html for a more exhaustive reference.

Navigating and Searching in Compressed Form

The most popular retrieval operations on semi-structured text are related to *navigating* the tree and to *searching* it. Navigating means moving from a node to its children, parent, and siblings. Searching means various *path matching* operations such as finding all the paths where a node labeled *A* is the parent of another labeled *B* and that one is the ancestor of another labeled *C*, which in turn contains text where word *W* appears. A popular language combining navigation and searching operations is XPath (see http://www.w3.org/TR/xpath20).

Several of the schemes above permit accessing and decompressing any part of the text at random positions. This is because they retain the original order of the components of the document and compress using a semi-static model. Those compression methods are transparent, in the sense that the classical techniques to navigate and search XML data, sequentially or using indexes, can be used almost directly over this compressed representation.

Other techniques, such as *SCM* (Huffman variant) or *XCQ*, allow random access under a slightly more complex scheme, because some work is needed in order to start decompression at a specific point. Finally, techniques based on adaptive compression (such as *XMLPPM* or the PPM variant of *SCM*) usually achieve better compression ratios but need to decompress the whole data before they can operate on it.

Some of these techniques, on the other hand, take some advantage of the separation between structure and content in order to run queries faster than scanning all the data. This is the case of *XCQ*, where the table that points from each different tree path to all the contents compressed under the corresponding model, is useful to avoid traversing those contents if the path does not match a path matching query. Another example is *XPress*, which encodes paths in a way that the codes themselves permit checking containment between two paths. A concept that deviates from the ideas presented is that of using a tree representation that permits sharing repeated subtrees. A good exponent is [5], which permits running a large subset of XPath directly over this compressed representation. The structure can be navigated almost transparently, and path matching operations can be sped up by factoring out the work done on repeated substructures.

Succinct Encodings for Labeled Trees

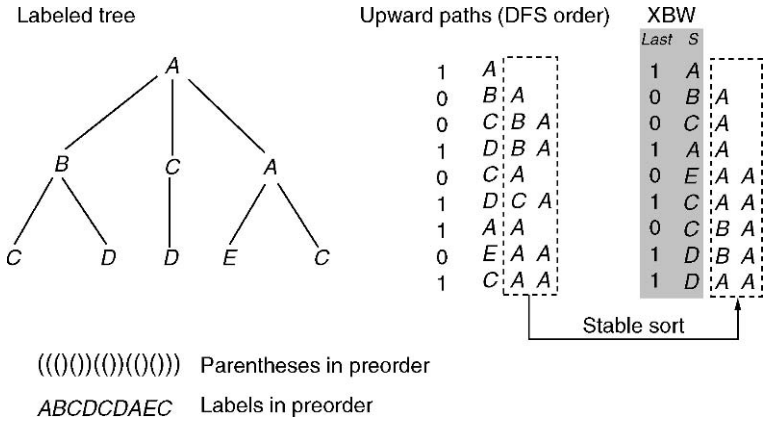
Succinct representations of labeled trees are an algorithmic development that finds applications in navigating semi-structured text in compressed form. In its simplest form, a general labeled tree of n nodes can be represented using a sequence P of $2n$ balanced parentheses and a sequence L of n labels (which correspond to tag names and will be regarded as atomic for simplicity).

This is obtained by traversing the tree in preorder (that is, first the current node and then recursively each of its children). As the tree is traversed, an opening parenthesis is added to P each time one goes down to a child, and a closing parenthesis when going up back to the parent. In L, the labels are added in preorder.

Figure 1 (left) shows an example representation of a labeled tree as a sequence of parentheses and labels in preorder. It is not hard to rebuild the tree from this representation. However, what is really challenging is to navigate the tree directly in this representation (where a node is represented by the position of its opening parenthesis). For this sake, only sublinear extra space on top of the plain representation is needed [9].

An essential operation to achieve efficient navigation in compressed form is the *rank* operation on bitmaps: $rank(P, i)$ is the number of 1s (here representing opening parentheses) in $P[1, i]$. One immediate application of *rank* is to obtain the label of a given node i, as $L[rank(P, i)]$. For example, consider the second child of the root in Fig. 1. It is represented by the opening parenthesis at position 8 in the sequence. Its label is therefore $L[rank(\text{“((()())(())(()()))”}, 8)] = L[5] = \text{“}C\text{”}$. Another application of *rank*, is to compute the depth of a node i. This is the number of opening minus closing parenthesis in $P[1, i]$, that is, $rank(P, i) - (i - rank(P, i)) = 2 \cdot rank(P, i) - i$. For example, the depth of the second child of the root is $2 \cdot rank(\text{“((()())(())(()()))”}, 8) - 8 = 2 \cdot 5 - 8 = 2$.

It is not possible to fully explain, in a short entry, the constant-time solutions to *rank* and other more complex operations needed to navigate the compressed tree. To have a flavor of how those solutions operate, consider the case of supporting *rank* in constant time and $o(n)$ extra bits. Absolute *rank* values are stored at every sth position of P (for some parameter s), and also relative (to the beginning of the last absolute sample) *rank* values at every bth position of P (for some parameter $b < s$). Note that each relative *rank* value



Managing Compressed Structured Text. Figure 1. An example labeled tree and two compression techniques. On the left (*bottom*), as a parentheses plus labels sequence in preorder. On the right, its *xbw* transform. The *dashed boxes* highlight the upward paths. The *grayed box* is the *xbw* transform of the tree.

needs only log s bits to be stored. Then, two table accesses (absolute plus relative *rank*) give the partial *rank* answer up to the b-bit chunk where position i belongs. To complete the query in constant time, a universal table is precomputed, which gives the number of 1-bits in *every possible* chunk of b bits. Some bit masking and a final access to this table suffice to count the remaining 1s within the chunk. By properly choosing s and b one achieves $o(n)$ extra bits overall, and still answers $rank(P, i)$ with three table accesses.

Other basic queries can be computed with similar mechanisms (coarse sampling to store absolute values, finer sampling to store relative values, and universal tables to process short chunks). For example, one can compute $select(P, i)$, the position of the ith opening parenthesis in P, so as to find the tree node corresponding to the ith label in L. Other essential operations for the navigation are $close(i)$, the position of the parenthesis that closes i (that is, the next parenthesis with the same depth of i); and $enclose(i)$, the lowest parenthesis that contains i (that is, the preceding parenthesis with depth smaller than that of i).

With these two operations one can navigate the tree as follows. The next sibling of i is $close(i) + 1$ (unless it is a closing parenthesis, in which case i is the last child of its parent). The first child of i is $i + 1$ unless $P[i + 1]$ is a closing parenthesis, in which case i is a leaf and hence has no children. The parent of i is $enclose(i)$. The size of the subtree rooted at i is $(close(i) - i + 1)/2$. For example, consider the first child of the root in Fig. 1, such that $i = 2$. It finishes at $close(i) = 7$. Its next sibling is $close(i) + 1 = 8$, the node of the previous examples. Its first child is $i + 1 = 3$, the leftmost tree leaf. Its parent is $enclose(i) = 1$, the root. The size of its subtree is $(close(i) - i + 1)/2 = (7 - 2 + 1)/2 = 3$.

In order to enrich the navigation using the labels, sequence $L[1, n]$ is also processed for symbol *rank* and *select* operations, where $rank_c(L, i)$ is the number of occurrences of c in $L[1, i]$ and $select_c(L, j)$ is the position of the jth occurrence of c in L. For example, the following procedure finds all the descendants of node i which are labeled c: (*i*) Find the position $j = rank(P, i)$ of node i in the sequence of labels. (*ii*) Compute $k = rank_c(L, j - 1)$, the number of occurrences of c prior to j. (*iii*) Find the positions $p_r = select_c(L, k + r)$ of c from j onwards, for successive r values until $select(P, p_r) > close(i)$, that is, until the answers are not anymore descendants of i. For example, consider again the first child of the root in Fig. 1, where $i = 2$ and $close(i) = 7$, and find its descendants labeled "D". The first step is to compute $j = rank(P, 2) = 2$, the position of its label in L. Now, $k = rank_{\text{“}D\text{”}}(L, 1) = 0$ tells that there are zero occurrences of "D" before $L[2]$. Now the next occurrences of "D" in L are found as $select_{\text{“}D\text{”}}(L, 1) = 4$, $select_{\text{“}D\text{”}}(L, 2) = 6$, ... The first such occurrence is mapped to the tree node $select(P, 4) = 5$ (the second tree leaf), which is within the subtree of i because $i \leq 5 \leq close(i)$. The second occurrence of "D" is already outside the tree because $select(P, 6) = 9$ exceeds $close(i) = 7$.

Many other powerful navigational operations can be supported, although a more sophisticated parentheses representation and much more technical developments are necessary. A good example can be seen in [4]. Empirical results have been given for the basic preorder parentheses represenation [9].

Integrating Indexing and Compression

In recent work [7,8], by means of introducing a so-called *xbw transform*, indexing and compression are made part of a single integrated process, so that the compressed data represents at the same time the structured text and an index built on it. This is is a very original idea which is likely to have practical impact in the next years.

A brief description of the transform follows. Imagine one takes all the upward paths in the labeled tree. There is one such path per tree node: given a node, its path starts from its parent and finishes in the root. Regard the upward paths as a sequence of labels, and assume for simplicity that labels are atomic symbols that can be sorted. If the tree has n nodes, the resulting n sequences of labels are collected in depth-first order and then stably sorted in lexicographical order. Finally, one forms a sequence with the nodes that originated each of the upward paths, once they are sorted. The *xbw transform* of a labeled tree is the sequence obtained plus a bitmap telling which of those nodes are the last child of their parent.

For example, take the tree of Fig. 1. The upward path from the root is the empty string. The upward path from the three root children are all "*A*", and so on. The list of all the upward paths found in a depth-first traversal is shown in the middle of the figure, and on the right one can see the paths after a stable lexicographical sorting. Now, collecting the nodes that originated those paths in order one gets the labels in the grayed area, S = "*ABCAECCDD*" (the other element is the bitmap marking the last children of their parents, *last* = "100101011"). Sequence S is essentially a permutation of the tree labels. As the sorting is stable, all the nodes originating the same path (e.g., "*A*") stay in depth-first order. In particular, sibling nodes are contiguous.

It turns out that it is possible to compute the *xbw* transform in linear time and space, and moreover to recover the original tree from these two sequences in linear time and space. Furthermore, it is possible to efficiently navigate the tree in *xbw*-transformed form, with operations such as moving to the parent of the current node, ith child, next sibling, ith child labeled X, and so on. Those operations also build on the *rank* and *select* operations described. For example, the third child of the root is represented by the upward path "*AA*". Its position after the *xbw* sorting is $i = 4$, which acts as the identifier for the node in this representation, note $S[4]$ = "*A*" is its label. The process to find its children is as follows: (*i*) Find how many "*A*"s are there before in S, $j = rank_{\text{"}A\text{"}}(S, i - 1) = 1$. (*ii*) Find the beginning of the range of the upward paths starting with $S[i]$ = "*A*" in the dashed box, $k = 2$ (this is precomputed in a table storing such value for each different label). (*iii*) Find the number of 1s in *last* before that range, $l = rank(last, k - 1) = 1$. (*iv*). Find the area corresponding to the children of i, $select(last, l + j) + 1 = 5$ to $select(last, l + j + 1) = 6$.

The key operation that makes the *xbw* transform unique compared to other tree representations its its path searching ability: It can identify all the nodes in the tree that descend from a given path sequence in time proportional to the length of the sequence and independent of the collection size (the nodes can then be retrieved one by one). That is, the *xbw*-transformed sequence acts not only as a navigable representation of the tree but also as a powerful index to carry out some path searching operations very efficiently.

Apart from saving all the pointer information, the *xbw*-transformed sequence groups together the node labels that descend from the same paths. Therefore, if root-to-node paths are good predictors of the contents of nodes (this is the property that most sophisticated techniques like *XMLPPM* exploit), the transformed sequence will contain long regions with similar contents. Those are easily compressible by block-wise encoding methods.

In [8] they showed how to apply this conceptual method to real XML data, mixing location path operations with queries on the text content. They present a practical implementation and empirical results showing that it is competitive with the best XML compressors, which do not offer simultaneous indexing capabilities.

Key Applications

Any application managing semi-structured text, particularly if it has to transmit it over slow channels or operate within limited fast memory, even if there is an unlimited supply of slower memory, benefits from these techniques.

Future Directions

Several problems remain open. A fundamental one is the definition of an adequate notion of entropy for semi-structured data, that is, a compressibility limit. While there is reasonable consensus on the entropy of plain text without structure (by taking it as a sequence in general), there is no agreement even on how to

measure the entropy of a tree, which is a key part of the entropy of semi-structured text. The fact that this issue is open implies that it is hard to determine how good, in absolute terms, is a compression scheme.

The future of the area is likely to be in manipulating XML in compressed form, and in this aspect the *xbw* transform is a promising direction. Yet, the area is far from offering a competitive and complete path search engine over compressed XML, for example. Similarly, the state of the art in permitting manipulating compressed XML, for example updating a semi-structured text collection, is very preliminary (see [11] for a recent, still theoretical, work on dynamizing the *xbw* transform).

Experimental Results

Experiments can be found in the papers cited. In particular, for the *xbw* transform, see [8].

URL to Code

Several public XML compressors are available, for example

XMill (http://sourceforge.net/projects/xmill),
XMLPPM (http://sourceforge.net/projects/xmlppm),
SCMPPM (http://www.infor.uva.es/~jadiego/download.php),
and
XGrind (http://cvs.sourceforge.net/viewcvs.py/xmill/xmill/XGrind).

Cross-references

- ▶ Compression
- ▶ Semi-Structured Data
- ▶ XML
- ▶ XPath/XQuery

Recommended Reading

1. Adiego J., Navarro G., and de la Fuente P. Using Structural Contexts to Compress Semistructured Text Collections. Inf. Proc. & Man., 43:769–790, 2007.
2. Baeza-Yates R. and Navarro G. Integrating contents and structure in text retrieval. ACM SIGMOD Rec., 25(1):67–79, 1996.
3. Barbay J., Golynski A., Munro I., and Rao S. Adaptive searching in succinctly encoded binary relations and tree-structured documents. In Proc. 17th Annual Symp. on Combinatorial Pattern Matching, 2006, pp. 24–35.
4. Bell T., Cleary J., and Witten I. Text Compression. Prentice Hall, Englewood Cliffs, NJ, 1990.
5. Buneman, P., Grohe, M., and Koch, C. Path Queries on Compressed XML. In Proc. 29th Very Large Databases Conference, 2003, pp. 141–152.
6. Cheney J. Compressing XML with multiplexed hierarchical PPM models. In Proc. 11th IEEE Data Compression Conf., 2001, pp. 163–172.
7. Ferragina P., Luccio F., Manzini G., and Muthukrishnan S. Structuring labeled trees for optimal succinctness, and beyond. In Proc. 46th Annu. Symp. on Foundations of Computer Science, 2005, pp. 184–196.
8. Ferragina P., Luccio F., Manzini G., and Muthukrishnan S. Compressing and searching XML data via two zips. In Proc. 15th Int. World Wide Web Conf., 2006.
9. Geary R., Rahman N., Raman R., and Raman V. A simple optimal representation of balanced parentheses. Theoretical Computer Science, 368(3):231–246, 2006.
10. Girardot M. and Sundaresan N. Millau: An encoding format for efficient representation and exchange of XML documents over the WWW. In Proc. 9th Int. World Wide Web Conference, 2000, pp. 747–765.
11. Gupta A., Hon W.K., Shah R., and Vitter J. A framework for dynamizing succinct data structures. In Proc. 34th Int. Colloquium on Automata, Languages, and Programming, 2007, pp. 521–532.
12. Levene M. and Wood P. XML structure compression. In Proc. 2nd Int. Workshop on Web Dynamics, 2002.
13. Liefke H. and Suciu D. XMill: an efficient compressor for XML data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 153–164.
14. Min J.K., Park M.J., and Chung C.W. XPress: a querieable compression for XML data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 122–133.
15. Tolani P. and Haritsa J. XGRIND: A query-friendly XML compressor. In Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 225–234.

Mandatory Access Control

Bhavani Thuraisingham
The University of Texas at Dallas, Richardson, TX, USA

Synonyms

Multilevel security

Definition

As stated in [1], "*in computer security, 'mandatory access control (MAC)' refers to a kind of access control defined by the National Computer Security Center's Trusted Computer System Evaluation Criteria (TCSEC) as a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such*

sensitivity." With operating systems, the subjects are processes and objects are files. The goal is to ensure that when a subject accesses a file, no unauthorized information is leaked.

Key Point

MAC Models: MAC models were developed initially for secure operating systems mainly in the 1970s and early 1980s, and started with the Bell and La Padula security model. This model has two properties: the simple security property and the *-property (pronounced the star property). The simple security property states that a subject has read access to an object if the subject's security level dominated the level of the object. The *-property states that a subject has write access to an object if the subject's security level is dominated by that of the object [2]. Since then, variations of this model as well as a popular model called the noninterference model [3] have been proposed. The noninterference model is essentially about higher-level processes not interfering with lower level processes. Note that with the Bell and La Padula model, a higher level process can covertly send information to a lower level process by manipulating the file locks, even though there can be no write down due to the star property. The noninterference model prevents such covert communication.

MAC for Database Systems: While Database Management Systems (DBMS) must deal with many of the same security concerns as operating systems (identification and authentication, access control, auditing), there are characteristics of DBMSs that introduce additional security challenges. For example, objects in DBMSs tend to be of varying sizes and can be of fine granularity such as relations, attributes and elements. This contrasts with operating systems where the granularity tends to be coarse such as files or segments. Because of the fine granularity in database systems the objects on which MAC is performed may differ. In operating systems MAC is usually performed on the same object such as a file whereas in DBMSs it could be on relations and attributes. The simple security and * property are both applicable for database systems. However many of the database systems have modified, the *-property to read as follows: *A subject has write access to an object if the subject's level is that of the object*. This means a subject can modify relations at its level. Various commercial secure DBMS products have emerged. These products have been evaluated using the Trusted Database Interpretation which interprets the TCSEC for database systems.

MAC for Networks: For applications in defense and intelligence multilevel secure networks are essential. The idea here is for the network protocols such as a TCP/IP (Transmission Control Protocol/Internet Protocol) protocols operate at multiple security levels. The Bell and La Padula model has been extended for networks. Furthermore, the commercial multilevel networks have been evaluated using the Trusted Network Interpretation that interprets the TCSEC for networks.

Cross-references

► Multilevel Secure Database Management System

Recommended Reading

1. http://en.wikipedia.org/wiki/Mandatory_access_control
2. Bell D. and LaPadula L. "Secure Computer Systems: Mathematical Foundations and Model," M74–244. The MITRE Corporation, Bedford, MA, 1973.
3. Goguen J. and Meseguer J. Security policies and security models. In Proc. IEEE Symp. on Security and Privacy, 1982, pp. 11–20.

MANET Databases

Yan Luo, Ouri Wolfson
University of Illinois at Chicago, Chicago, IL, USA

Synonyms

Mobile ad hoc network databases

Definition

A mobile ad hoc network (MANET) database is a database that is stored in the peers of a MANET. The network is composed by a finite set of mobile peers that communicate with each other via short range wireless protocols, such as IEEE 802.11, Bluetooth, Zigbee, or Ultra Wide Band (UWB). These protocols provide broadband (typically tens of Mbps) but short-range (typically 10–100 m) wireless communication. On each mobile peer there is a local database that stores and manages a collection of data items, or reports. A report is a set of values sensed or entered by the user at a particular time, or otherwise obtained by a mobile peer. Often a report describes a physical resource such as an available parking slot. All the local databases maintained by the mobile peers form the

MANET database. The peers communicate reports and queries to neighbors directly, and the reports and queries propagate by transitive multi-hop transmissions. Figure 1 below illustrates the definition.

MANET databases enable matchmaking or resource discovery services in many application domains, including social networks, transportation, mobile electronic commerce, emergency response, and homeland security.

Communication is often restricted by bandwidth and power constraints on the mobile peers. Furthermore, reports need to be stored and later forwarded, thus memory constraints on the mobile devices constitute a problem as well. Thus, careful and efficient utilization of scarce peer resources (specifically bandwidth, power, and memory) are an important challenge for MANET databases.

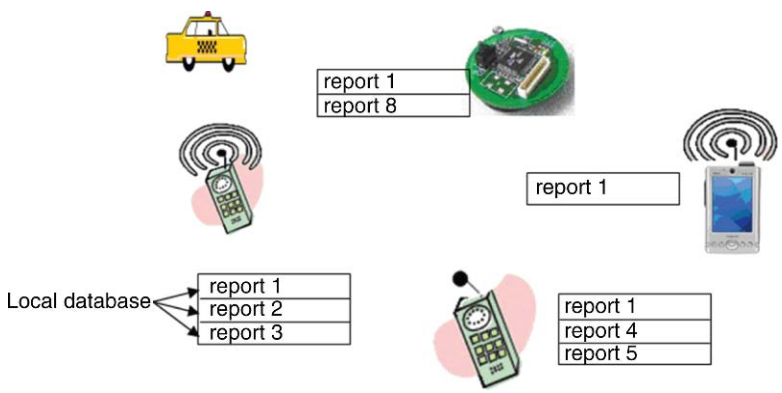
Historical Background

Consider mobile users that search for local resources. Assuming that the information about the existence and location of such a resource resides on a server, a communication infrastructure is necessary to access the server. Such an infrastructure may not be available in military/combat situations, disaster recovery, in a commercial flight, etc. Even if the infrastructure and a server are both available, a user may not be willing to pay the dollar-cost that is usually involved in accessing the server through the cellular infrastructure. Furthermore, cellular bandwidth is limited (e.g., 130 character text messages). In other words, a client-server approach may have accessibility problems.

Currently, Google and local.com provide static local information (e.g., the location of a restaurant, pharmacy, etc.), but not dynamic information such as the location of a taxi cab, a nearby person of interest, or an available parking slot. These dynamic resources are temporary in nature, and thus require timely, real-time update rates. Such rates are unlikely to be provided for the country or the world by a centralized server farm, e.g., Google. Thus, dynamic local resources may require local servers, each dedicated to a limited geographic area. However, for many areas such a local server may not exist due to lack of a profitable business model, and if it exists it may be unavailable (such servers are unlikely to have the reliability of global sites such as Google). Furthermore, the data on the server may be unavailable due to propagation delays (think of sudden-brake information that needs to be propagated to a server and from there to the trailing vehicles), or due to device limitations (e.g., a cab customer's cell-phone may have Bluetooth but not internet access to update the server), or due to the fact that updates from mobile devices may involve a communication cost that nobody is willing to pay, or due to the fact that the local server (e.g., of Starbucks) may accept only updates from certain users or certain applications but not others. In short, a client-(local)-server may have both accessibility and availability problems.

Thus, a MANET database can substitute or augment the client-(local)-server approach. Communication in the MANET is free since it uses the unlicensed spectrum, and larger in bandwidth than the cellular infrastructure, thus can provide media rich information, such as maps, menus, and even video. A mobile user may search the MANET database only, or combine it with a client-server search.

Currently, there are quite a few experimental projects in MANET databases. These can be roughly classified into pedestrians and vehicular projects. Vehicular



MANET Databases. Figure 1. A MANET database.

projects deal with high mobility and high communication topology change-rates, whereas pedestrians projects have a strong concern with power issues. The following are several active experimental MANET database projects for pedestrians and vehicles:

Pedestrians Projects

- *7DS* – Columbia University
  - http://www.cs.unc.edu/~maria/7ds/
  - Focuses on accessing web pages in environments where only some peers have access to the fixed infrastructure.
- *iClouds* – Darmstadt University
  - http://iclouds.tk.informatik.tu-darmstadt.de/
  - Focuses on the provision of incentives to brokers (intermediaries) to participate in MANET databases.
- *MoGATU* – University of Maryland, Baltimore County
  - http://mogatu.umbc.edu/
  - Focuses on the processing of complex data management operations, such as joins, in a collaborative fashion.
- *PeopleNet* – National University of Singapore
  - http://www.ece.nus.edu.sg/research/projects/abstract.asp?Prj=101
  - Proposes the concept of information Bazaars, each of which specializes in a particular type of information; reports and queries are propagated to the appropriate bazaar by the fixed infrastructure.
- *MoB* – University of Wisconsin and Cambridge University
  - http://www.cs.wisc.edu/~suman/projects/agora/
  - Focuses on incentives and the sharing among peers of virtual information resources such as bandwidth.
- *Mobi-Dik* – University of Illinois at Chicago
  - http://www.cs.uic.edu/~wolfson/html/p2p.html
  - Focuses on information representing physical resources, and proposes stateless algorithms for query processing, with particular concerns for power, bandwidth, and memory constraints.

Vehicular Projects

- *CarTALK 2000* – A European project
  - http://www.cartalk2000.net/
  - Develops a co-operative driver assistance system based upon inter-vehicle communication and MANET databases via self-organizing vehicular ad hoc networks.
- *FleetNet* – Internet on the Road Project
  - http://www.ccrle.nec.de/Projects/fleetnet.htm
  - Develops a wireless multi-hop ad hoc network for intervehicle communication to improve the driver's and passenger's safety and comfort. A data dissemination method called "contention-based forwarding" (CBF) is proposed in which the next hop in the forwarding process is selected through a distributed contention mechanism based on the current positions of neighbors.
- *VII* – Vehicle Infrastructure Integration, a US DOT project
  - http://www.its.dot.gov/vii/
  - The objective of the project is to deploy advanced vehicle-to-vehicle and vehicle-to-infrastructure communications that could keep vehicles from leaving the road and enhance their safe movement through intersections.
- *Grassroots, Trafficview* – Rutgers University

TrafficInfo – University of Illinois at Chicago

- http://paul.rutgers.edu/~gsamir/dataspace/grassroots.html
- http://discolab.rutgers.edu/traffic/veh_apps.htm
- http://cts.cs.uic.edu/
- These projects develop an environment in which each vehicle contributes a small piece of traffic information (its current speed and location) to the network, using the P2P paradigm, and each vehicle aggregates the pieces into a useful picture of the local traffic.

Foundations

There are two main paradigms for answering queries in MANET databases, one is report pulling and the other one is report pushing.

Report pulling means that a mobile peer issues a query which is flooded in the whole network, and the answer-reports will be pulled from the mobile peers that have them (see e.g., [2]). Report pulling is widely used in resource discovery, such as route discovery in mobile ad hoc networks and file discovery by query flooding in wired P2P networks like Gnutella. Flooding in a wireless network is in fact relatively efficient as compared to wired networks because of the wireless broadcast advantage, but there are also disadvantages which will be explained below.

Another possible approach for data dissemination is report pushing. Report pushing is the dual problem of report pulling; reports are flooded, and consumed by peers whose query is answered by received reports. So far there exist mechanisms to broadcast information in the complete network, or in a specific geographic area (geocast), apart from to any one specific mobile node (unicast/mobile ad hoc routing) or any one arbitrary node (anycast). Report pushing paradigm can be further divided into stateful methods and stateless methods. Most stateful methods are topology-based, i.e., they impose a structure of links in the network, and maintain states of data dissemination. PStree [4], which organizes the peers as a tree, is an example of topology based methods.

Another group of stateful methods is cluster- or hierarchy-based method, such as [14], in which moving peers are grouped into some clusters or hierarchies and the cluster heads are randomly selected. Reports are disseminated through the network in a cluster or hierarchy manner, which means that reports are first disseminated to every cluster head, and each cluster head then broadcasts the reports to the member peers in its group. Although cluster- or hierarchy-based methods can minimize the energy dissipation in moving peers, these methods will fail or cost more energy in highly mobile environments since they have to maintain a hierarchy structure and frequently reselect cluster heads.

Another stateful paradigm consists of location-based methods (see [9]). In location-based methods, each moving peer knows the location of itself and its neighbors through some localization techniques, such as GPS or Atomic Multilateration (see [9]).

The simplest location-based data dissemination is Greedy Forwarding, in which each moving peer transmits a report to a neighbor that is closer to the destination than itself. However, Greedy Forwarding can fail in some cases, such as when a report is stuck in local minima, which means that the report stays in a mobile peer whose neighbors are all further from the destination. Therefore, some recovery strategies are proposed, such as GPSR (Greedy Perimeter Stateless Routing [6]). Other location-based methods, such as GAF (Geographic Adaptive Fidelity [17]) and GEAR (Geographical and Energy Aware Routing [18]), take advantage of knowledge about both location and energy to disseminate information and resources more efficiently.

In stateless methods, the most basic and simplest one is flooding-based method, such as [11]. In flooding-based methods, mobile peers simply propagate received reports to all neighboring mobile peers until the destination or maximum hop is reached. Each report is propagated as soon as it is received. Flooding-based methods have many advantages, such as no state maintenance, no route discovery, and easy deployment. However, they inherently cannot overcome several problems, such as implosion, overlap, and resource blindness. Implosion refers to the waste of resources taking place when a node forwards a message to a neighbor although the latter may have already received it from another source. Overlap occurs when two nodes read the same report, and thus push into the network the same information. Resource blindness denotes the inability of the protocol to adapt the node's behavior to its current availability of resources, mainly power [12]. Therefore, other stateless methods are proposed, such as gossiping-based methods and negotiation-based methods.

Gossiping-based methods, such as [3], improve flooding by transmitting received reports to a subset of randomly selected neighbors; another option is to have some neighbors simply drop the report. For example, the neighbors that are not themselves interested in the report drop it. The advantages of gossiping-based methods include reducing the implosion and lowering the system overhead. However, dissemination, and thus performance, is reduced compared to pure flooding.

Negotiation-based methods solve the implosion and overlap problem by transmitting first the id's of reports; the reports themselves are transmitted only when requested (see [7]). Thus, some extra data transmission is involved, which costs more memory, bandwidth, and energy. In addition, in negotiation-based methods, moving peers have to generate meta-data or a signature for every report so that negotiation can be carried out, which will increase the system overhead and decrease the efficiency.

Another important stateless paradigm for data dissemination in MANET databases is store-and-forward. In contrast to flooding, store-and-forward does not propagate reports as soon as they are received; rather they are stored and rebroadcast later. This obviously introduces storage and bandwidth problems, if too many reports need to be saved and rebroadcast at the

same time. To address these, methods such as [5] rank all the reports in a peer's database in terms of their relevance (or expected utility), and then the reports are communicated and saved in the order of their relevance. Or, the reports requested and communicated are the ones with the relevance above a certain threshold. The notion of relevance quantifies the importance or the expected utility of a report to a peer at a particular time and at a particular location. Other store-and-forward methods include PeopleNet [10] and 7DS [13].

In summary, the paradigms for data dissemination in MANET databases are summarized in Fig. 2 below.

Key Applications

MANET databases provide mobile users a search engine for transient and highly dynamic information in a local geospatial environment. MANET databases employ a unified model for both the cellular infrastructure and the mobile ad hoc environments. When the infrastructure is available, it can be augmented by the MANET database approach.

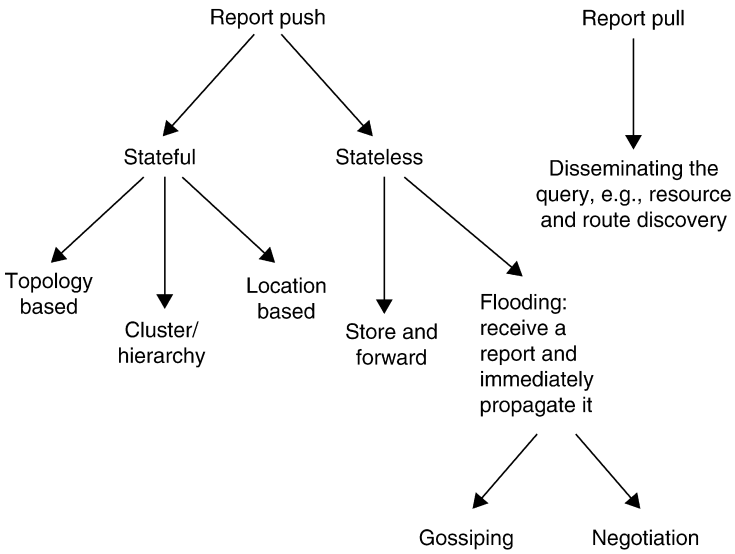
Consider a MANET database platform, i.e., a set of software services for data management in a MANET environment; it is similar to a regular Database Management System, but geared to mobile P2P interactions. Such a platform will enable quick building of matchmaking or resource discovery services in many application domains, including social networks, emergency response and homeland security, the military, airport applications, mobile e-commerce, and transportation.

Social Networks

In a large professional, political, or social gathering, MANET databases are useful to automatically facilitate a face-to-face meeting based on matching profiles. For example, in a professional gathering, MANET databases enable attendees to specify queries (interest profiles) and resource descriptions (expertise) to facilitate face-to-face meetings, when mutual interest is detected. Thus, the individual's profile that is stored in MANET databases will serve as a "wearable web-site." Similarly, MANET databases can facilitate face-to-face meetings for singles matchmaking.

Emergency Response, Homeland Security, and the Military

MANET databases offer the capability to extend decision-making and coordination capability. Consider workers in disaster areas, soldiers and military personnel operating in environments where the wireless fixed infrastructure is significantly degraded or nonexistent. As mobile users involved in an emergency response naturally cluster around the location of interest, a self-forming, high-bandwidth network that allows database search without the need of potentially



MANET Databases. Figure 2. Query answering methods in MANET databases.

compromised infrastructure could be of great benefit. For instance, the search could specify a picture of a wanted person.

Airport Applications

A potential opportunity that will benefit both the consumer and the airport operations is the dissemination and querying of real-time information regarding flight changes, delays, queue length, parking information, special security alerts and procedures, and baggage information. This can augment the present audio announcements that often cannot be heard in nearby restaurants, stores, or restrooms, and augment the limited number of displays.

Mobile E-commerce

Consider short-range wireless broadcast and mobile P2P dissemination of a merchant's sale and inventory information. It will enable a customer (whose cell phone is query-capable) who enters a mall to locate a desired product at the best price. When a significant percentage of people have mobile devices that can query retail data, merchants will be motivated to provide inventory/sale/coupons information electronically to nearby potential customers. The information will be disseminated and queried in a P2P fashion (in, say, a mall or airport) by the MANET database.

Transportation Safety and Efficiency

MANET databases can improve safety and mobility by enabling travelers to cooperate intelligently and automatically. A vehicle will be able to automatically and transitively communicate to trailing vehicles its "slow speed" message when it encounters an accident, congestion, or dangerous road surface conditions. This will allow other drivers to make decisions such as finding alternative roads. Also, early warning messages may allow a following vehicle to anticipate sudden braking, or a malfunctioning brake light, and thus prevent pile-ups in some situations. Similarly, other resource information, such as ridesharing opportunities, transfer protection (transfer bus requested to wait for passengers), will be propagated transitively, improving the efficiency of the transportation system.

Future Directions

Further work is necessary on data models for mobile P2P search applications. Work on sensor databases (e.g., Tinydb [8]) addresses data-models and languages for sensors, but considers query processing in an environment of static peers (see e.g., POS [1]). Cartel [5] addresses the translation of these abstractions to an environment in which cars transfer collected data to a central database via fixed access points. Work on MANET protocols deals mainly with routing and multicasting. In this landscape there is a gap, namely general query-processing in MANET's; such processing needs to be cognizant of many issues related to peer-mobility. For example, existing mobile P2P query processing methods deal with simple queries, e.g., selections; each query is satisfied by one or more reports. However, in many application classes one may be interested in more sophisticated queries, e.g., aggregation. For instance, in mobile electronic commerce a user may be interested in the minimum gas price within the next 30 miles on the highway. Processing of such P2P queries may present interesting optimization opportunities.

After information about a mobile resource is found, localization is often critical for finding the physical resource. However, existing (self)-localization techniques are insufficient. For example, GPS is not available indoors and the accuracy of GPS is not reliable. Thus, furthering the state of the art on localization is important for mobile P2P search.

As discussed above, MANET databases do not guarantee answer completeness. In this sense, the integration with an available infrastructure such as the internet or a cellular network may improve performance significantly. This integration has two aspects. First, using the communication infrastructure in order to process queries more efficiently; and second, using data on the fixed network in order to provide better and more answers to a query. The seamless integration of MANET databases and infrastructure databases introduces important research challenges.

Other important research directions include: incentives for broker participation in query processing (see [16]), and transactions/atomicity/recovery issues in databases distributed over mobile peers (virtual currency must be transferred from one peer to another in an atomic fashion, otherwise may be lost).

Of course, work on efficient resource utilization in mobile peers, and coping with sparse networks and dynamic topologies is still very important for mobile P2P search.

Cross-references

- ▶ Mobile Ad hoc Network Databases
- ▶ Peer-to-peer Database
- ▶ Peer-to-peer Network

Recommended Reading

1. Cox L., Castro M., and Rowstron A. POS: Practical Order Statistics for wireless sensor networks. In Proc. 23rd Int. Conf. on Distributed Computing Systems, 2006, pp. 52.
2. Das S.M., Pucha H., and Hu Y.C. Ekta: An efficient DHT substrate for distributed applications in Mobile Ad hoc Networks. In Proc. Sixth IEEE Workshop on Mobile Computing Systems and Applications, 2004, pp. 163–173.
3. Datta A., Quarteroni S., and Aberer K. Autonomous gossiping: a self-organizing epidemic algorithm for selective information dissemination in wireless Mobile Ad-Hoc Networks. In Proc. Int. Conf. Semantics of a Networked World, 2004, pp. 126–143.
4. Huang Y. and Molina H.G. Publish/subscribe in a mobile environment. In Proc. 2nd ACM Int. Workshop on Data Eng. for Wireless and Mobile Access, 2001, pp. 27–34
5. Hull B. et al. CarTel: a distributed mobile sensor computing system. In Proc. 4th Int. Conf. on Embedded Networked Sensor Systems, 2006, pp. 125–138.
6. Karp B. and Kung H.T. GPSR: Greedy Perimeter Stateless Routing for wireless sensor networks. In Proc. 6th Annual Int. Conf. on Mobile Computing and Networking, 2000, pp. 243–254.
7. Kulik J., Heinzelman W., and Balakrishnan H. Negotiation-based protocols for disseminating information in wireless sensor networks. Wireless Netw., 8:169–185, 2002.
8. Madden S.R., Franklin M.J., Hellerstein J.M., and Hong W. Tiny DB: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst., 30(1):122–173, 2005.
9. Mauve M., Widmer A., and Hartenstein H. A survey on position-based routing in Mobile Ad-Hoc Networks. IEEE Netw., 15(6):30–39, 2001.
10. Motani M., Srinivasan V., and Nuggehalli P. PeopleNet: engineering a wireless virtual social network. In Proc. 11th Annual Int. Conf. on Mobile Computing and Networking, 2005, pp. 243–257.
11. Oliveira R., Bernardo L., and Pinto P. Flooding techniques for resource discovery on high mobility MANETs. In Proc. Int. Workshop on Wireless Ad-hoc Networks, 2005.
12. Papadopoulos A.A., and McCann J.A. Towards the design of an energy-efficient, location-aware routing protocol for mobile, Ad-hoc Sensor Networks. In Proc. of the 15th Int. Workshop on Database and Expert Systems Applications, 2004, pp. 705–709.
13. Papadopouli M. and Schulzrinne H. Design and implementation of a P2P Data dissemination and prefetching tool for mobile users. In Proc. 1st New York Metro Area Networking Workshop, IBM TJ Watson Research Center. Hawthorne, NY, 2001.
14. Visvanathan A., Youn J.H., and Deogun J. Hierarchical Data Dissemination Scheme for Large Scale Sensor Networks. In Proc. IEEE Int. Conf. on Communications, 2005, pp. 3030–3036.
15. Wolfson O., Xu B., Yin H.B., and Cao H. Search-and-discover in Mobile P2P Network Databases. In Proc. 23rd Int. Conf. on Distributed Computing Systems, 2006, pp. 65.
16. Xu B., Wolfson O., and Rishe N. Benefit and pricing of spatio-temporal information in Mobile Peer-to-Peer Networks. In Proc. 39th Annual Hawaii Conf. on System Sciences, vol. 9, 2006, pp. 2236.
17. Xu Y., Heidemann J., and Estrin D. Geography-informed energy conservation for Ad hoc Routing. In Proc. 7th Annual Int. Conf. on Mobile Computing and Networking, 2001, pp. 70–84.
18. Yu Y., Govindan R., and Estrin D. Geographical and Energy Aware Routing: a Recursive Data Dissemination Protocol for Wireless Sensor Networks. Technical Report UCLA/CSD-TR-01-0023, UCLA, May 2001.

Manmachine Interaction (Obsolete)

- ▶ Human-Computer Interaction

Many Sorted Algebra

- ▶ Data Types in Scientific Data Management

MAP

Steven M. Beitzel[1], Eric C. Jensen[2], Ophir Frieder[3]
[1]Telcordia Technologies, Piscataway, NJ, USA
[2]Twitter, Inc., San Fransisco, CA, USA
[3]Georgetown University, Washington, DC, USA

Synonyms

Mean average precision

Definition

The Mean Average Precision (MAP) is the arithmetic mean of the average precision values for an information retrieval system over a set of n query topics. It can be expressed as follows:

$$MAP = \frac{1}{n}\sum_{n} AP_n$$

where AP represents the Average Precision value for a given topic from the evaluation set of n topics.

Key Points

The Mean Average Precision evaluation metric has long been used as the de facto "gold standard" for information retrieval system evaluation at the NIST Text Retrieval Conference (TREC) [1]. Many TREC tracks over the years have evaluated run submissions using the *trec_eval* program, which calculates Mean Average Precision, along with several other evaluation metrics. Much of the published research in the information retrieval field over the last 25 years relies on observed difference in MAP to draw conclusions about the effectiveness of a studied technique or system relative to a baseline.

Recently, the explosive growth of the World Wide Web and the corresponding difficulty of creating test collections that are representative, robust, and of appropriate scale has created new challenges for the research community. One such challenge is how to best evaluate systems in cases of incomplete relevance information. It has been shown that ranking systems by their MAP scores when relevance information is incomplete does not correlate highly with their rankings with complete judgments. This is a key weakness of MAP as a metric. In response to this problem, new metrics (such as BPref, for example) have been proposed that attempt to compensate for often incomplete relevance information [2].

Cross-references

- ▶ Average Precision
- ▶ BPref
- ▶ Chart
- ▶ Effectiveness Involving Multiple Queries
- ▶ Geometric Mean Average Precision

Recommended Reading

1. National Institute of Standards and Technology. TREC-2004 common evaluation measures. Available online at: http://trec.nist.gov/pubs/trec14/appendices/CE.MEASURES05.pdf (retrieved on August 27, 2007).
2. Sakai T. Alternatives to BPref. In Proc. 33rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2007, pp. 71–78.

Map Algebra

- ▶ Spatial Operations and Map Operations

Map Matching

Christian S. Jensen, Nerius Tradišauskas
[1]Aalborg University, Aalborg, Denmark

Synonyms

Position snapping

Definition

Map matching denotes a procedure that assigns geographical objects to locations on a digital map. The most typical geographical objects are point positions obtained from a positioning system, often a GPS receiver. In typical uses, the GPS positions derive from a receiver located in a vehicle or other moving object traveling in a road network, and the digital map models the embedding into geographical space of the roads by means of polylines that approximate the center lines of the roads. The GPS positions generally do not intersect with the polylines, due to inaccuracies. The aim of map matching is then to place the GPS positions at their "right" locations on the polylines in the map.

Map matching is useful for a number of purposes. Map matching is used when a navigation system displays the vehicle's location on a map. In many applications, information such as speed limits are assigned to the representations of roads in a digital map–map matching offers a means of relating such information to moving objects. Map matching may also be used for the representation of a route of a vehicle by means of the (sub-) polylines in the digital map.

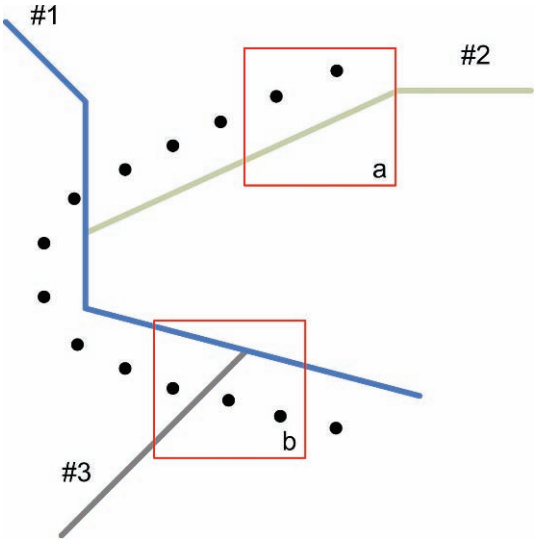
Two general types of map matching exist, namely on-line map matching and off-line map matching. With on-line map matching, the map location of an object's current position needs to be determined in real time. Only past, but not future, positions are available. Vehicle navigation systems exemplify this type of map matching. In off-line map matching, a static data set of positions is given, meaning that all future positions are available when map matching a position. Thus better map matching may result when compared to on-line map matching. For example, off-line map matching may be used for billing in pay-per-use scenarios (insurance, road pricing).

Historical Background

One of the earliest map matching algorithms found in the literature dates back to 1971 and is due to R.L.

French (see the overview in reference [1]). A 1996 paper by Berstein and Kornhauser [2] offers a brief introduction to the map matching problem and its variations.

The scientific literature contains a range of papers that address different aspects of the map matching problem. White et al. [3] study techniques that pay special attention to intersection areas, where map matching can be particularly challenging. Taylor et al. [4] propose a map matching technique that uses differential corrections and height, which leads to improved performance. Quddus et al. [5] provide a summary of different on-line and off-line map matching algorithms and describe advantages and disadvantages of these. Quddus et al. [6] have most recently proposed a map matching algorithm that utilizes techniques from fuzzy logic. This technique shows improved accuracy of polyline identification and the positioning on polylines. A complex off-line map matching algorithm was recently developed by Bratkatsoulas et al. [7] that uses the Fréchet distance to map match GPS position samples recorded only every 30 seconds. (While GPS receivers typically output a position every second, it may be that only some of these are saved for use in subsequent off-line map matching).



Map Matching. Figure 1. Map and vehicle location example.

Foundations

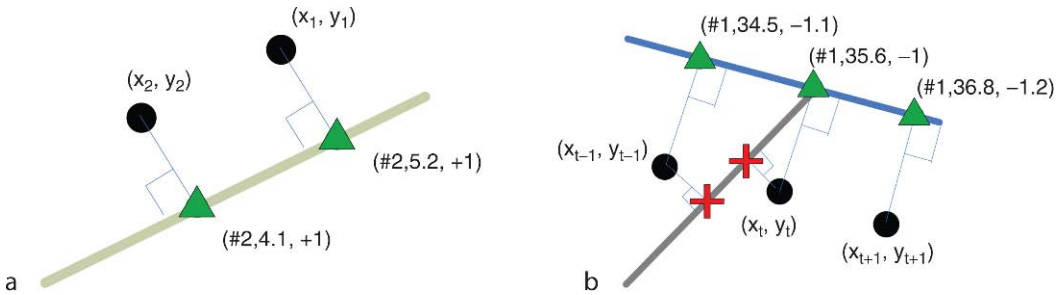
Basics. The most common use of map matching occurs in transportation where the GPS positions obtained from a GPS receiver in a vehicle are map matched to a digital representation of a road network. An example of GPS positions from a vehicle (dots) and a digital road network are shown in Fig. 1. The vehicle's trip started on road #2 and continued along road #1. In Fig. 2a the start of the trip is enlarged. The dots represent the vehicle's GPS locations, and the triangles represent the corresponding positions map matched onto the road network. The road network locations are typically expressed by using linear referencing, which is a standard means of indicating such locations. With linear referencing, a tuple (#2,5.2,+1) captures the road that the vehicle is driving on (#2), a position on that road measured as a distance from the beginning of the road (5.2 distance units). The third element captures a perpendicular distance from the road location given by the first two elements. In the tuple, the displacement is one distance unit to the left. Because a GPS position is typically mapped to the closest location in the road digital network (i.e., a perpendicular projection is used), the third element is capable of capturing the GPS position that was map matched to the road network location. Linear referencing is supported by, e.g., the Oracle DBMS [6].

In Fig. 2b, a place where map matching is challenging is enlarged. The crosses represent two instance of wrong map matching to the nearest road, and the triangles represent the correct map matching.

Categorization of Map Matching Algorithms. Map matching can be divided into on-line and off-line map matching. On-line map matching occurs in real-time. Here, the map matching algorithm tries to identify the network location of a GPS position every time a new position is received. The algorithm has available the current position as well as information about the map matching of previous positions.

This contrasts off-line map matching, which occurs after a trip is over and all the positions from the start to the end are known. Off-line map matching is more accurate than on-line map matching, as more information (i.e., future positions) is available. An off-line algorithm does not provide a map matching result until the entire trip has been map matched.

On-line map matching is mostly used in vehicle navigation, tracking, and other applications that need the most recent network location of a vehicle. Off-line map matching is mostly used to determine as accurately as possible which route a vehicle was driving. Off-line map matching may also be used in scenarios where



Map Matching. Figure 2. Map matching example.

GPS data are received in batches from content providers and where the purpose is to build speed maps that capture the expected travel speeds for different road segments and time intervals. Network locations of vehicles are essential in road load analysis, road pricing, and similar applications.

Map matching algorithms can also be divided into point-to-point, point-to-polyline, and polyline-to-polyline approaches. In point-to-point map matching, a point out of a point set is identified as a match for the given position. In point-to-polyline matching, a polyline in a polyline set is identified, and a point on the polyline is identified that represents the given point as a polyline-set location. In the typical scenario, the polyline set represents a road network, and the position is a GPS position. Finally, in polyline-to-polyline map matching, polylines are identified from the polylines in a road network that best match a given polyline that is usually constructed from point positions.

Map Matching Principles. This section follows the explanation of the basic principles of map matching by considering in some detail the commonly used point-to-polyline map matching for the on-line case.

Map matching algorithms often consist of two overall steps (some algorithms skip the first of these two steps or include an extra step).

In the start-up step, the polyline in the digital road network on which the vehicle is initially located is found. Specifically, map matching is done for the first GPS position received. The correct outcome of this step is very important, as the map matching algorithms use the connections between roads, or the road network topology, to determine the ensuing polylines of the vehicle's movement path. Therefore, algorithms often perform special operations to determine a reliable match for the first GPS position.

In the steady-state step, the subsequent polylines in the digital road network are identified to form the route along which the vehicle is moving. This subsequent map matching follows a standard pattern:

1. Extract the relevant information from the record received from the GPS receiver.
2. Select candidate polylines from the digital road network.
3. Use algorithm-specific heuristics to determine the most suitable polyline among the candidate polylines.
4. Determine the vehicle position on the selected polyline.

First, information such as latitude, longitude, speed, and heading is extracted from the record obtained from the positioning unit and is converted into an appropriate format (a unified coordinate and metric system consistent with the digital road network).

Second, the candidate polylines are selected. Usually the polylines that are within a certain threshold distance of the GPS position are selected. An alternative approach is to select the n polylines nearest to the GPS position. The polylines found make up the candidate polyline set.

Third, specific heuristics are used to select the best polyline among the candidates. A common approach is to assign weights to the candidate polylines according to different criteria. The polyline with the highest sum of weights is then chosen. Some algorithms reduce the set of candidate polylines prior to assigning weights. For example, polylines that are perpendicular to the vehicle's heading may be disregarded, as may polylines that are not connected with the polyline currently being considered.

Fourth, the vehicle position on the selected polyline is found. The usual approach is to select the location on the polyline that is closest to the vehicle's position. This is a point-to-polyline projection. The projection of the position may be an end point of a line segment

in the polyline, or it may be in the interior of a line segment. Quddus et al. [5] propose a more sophisticated approach that uses both the distance traveled since the last map matching and the "raw" projection of the GPS position onto the polyline.

Dead Reckoning. In certain regions, the signals emitted by the GPS satellites may be very weak due to obstacles. This in turn degrades the accuracies of the positions produced by GPS receivers in those regions, and in some cases no GPS position may be produced. In such cases, both on-line and off-line map matching algorithms may utilize dead reckoning to estimate the movement of a vehicle in the road network. The use of dead reckoning is particularly attractive when the average speed of the vehicle is known (preferably every second) and when the road on which the vehicle is driving neither splits nor has intersections. This occurs when the road is inside a tunnel with no exits.

Heuristics for the Selection of a Polyline. Different algorithms use different heuristics and weights when attempting to identify the best polyline among the candidate polylines. Each candidate polyline is assigned a weight for each criterion considered, and the polyline with the highest sum of weights is then selected. Common weighting criteria include the following:

- Weight for the proximity of a polyline. A polyline is assigned a weight according to its proximity to the GPS position being map matched. It is natural to assume that the closer a polyline is to the position, the better a candidate the polyline is.
- Weight for the continuity of a polyline. This weight is assigned to each polyline for being a continuation of the previously map matched polyline. This weight represents the reasoning that vehicles tend to drive on the same road most of the time.
- Weight for direction similarity. Polylines whose bearing is similar to the vehicle's heading are assigned higher values.
- Weight for topology. Higher weights are added to polylines that are connected to the polyline currently being map matched to.

Algorithms may also include weights for speed limit changes, shortest distance, road category, one-way streets, etc. Off-line map matching algorithms may use fewer, but more robust weights that are not suitable for the on-line map matching algorithms.

Execution Time Constraints and Accuracy. In on-line map matching, the algorithms must keep up with the GPS device that usually emits one position per second. With current on-board computing units, it is usually not a problem for on-line map matching algorithms map match a GPS position within 1 second. For off-line map matching, there are no real-time constraints.

Key Applications

Map matching is essential for applications that rely on the positioning of a user within a road network.

This occurs in vehicle navigation where the user's position is to be displayed so that it coincides with the road network. When GPS data is used for the construction of speed maps, map matching is used. Such speed maps may be used for travel time prediction, route construction, and capacity planning. In metered services such as insurance and road pricing, map matching is also used.

Further, map matching is used in location-based services where the content to be retrieved is positioned within the road network. Services that offer network-context awareness utilize map matching. These including current network location context awareness [9], and route context awareness [10].

Tracking also plays a key role in intelligent speed adaptation [11] where drivers are alerted when they exceed the current speed limit. The speed limits are attached to the digital road network, so map matching is needed to identify the current speed limit. Finally, map matching is important for a range of other application within intelligent transportation systems.

Future Directions

In the near future, digital road networks will have accurate lane information embedded, and positioning technologies will be accurate enough to enable lane-level positioning. This will necessitate the extension of map matching techniques to function at the lane level.

Cross-references

- ▶ Compression of Mobile Location Data
- ▶ Location Prediction
- ▶ Location-Based Services
- ▶ Mobile Database
- ▶ Mobile Sensor Network Data Management
- ▶ Road Networks

- ▶ Spatial Network Databases
- ▶ Spatial Operations and Map Operations
- ▶ Spatio-Temporal Trajectories
- ▶ Spatiotemporal Interpolation Algorithms

Recommended Reading

1. Bernstein D. and Kornhauser A. An introduction to map matching for personal navigation assistants. New Jersey TIDE Center, 1996. http://www.njtide.org/reports/mapmatchintro.pdf.
2. Brakatsoulas S., Pfoser D., Salas R., and Wenk C. On map-matching vehicle tracking data. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 853–864.
3. Brilingaitė A. and Jensen C.S. Enabling routes of road network constrained movements as mobile service context. Geoinformatica, 11(1):55–102, 2007.
4. Civilis A., Jensen C.S., and Pakalnis S. Techniques for efficient road-network-based tracking of moving objects. IEEE Trans. Knowl. Data Eng., 17(5):698–712, 2005.
5. French R.L. Historical overview of automobile navigation technology. In Proc. 36th IEEE Vehicular Technology Conf., 1986, pp. 350–358.
6. Oracle Corporation. Oracle Spatial and Oracle Locator. http://www.oracle.com/technology/products/spatial/index.html.
7. Quddus M., Ochieng W., Zhao L., and Noland R. A general map matching algorithm for transport telematics applications. GPS Solut. J., 7(3):157–167, 2003.
8. Quddus M.A., Noland R.B., and Ochieng W.Y. A high accuracy fuzzy logic based map matching algorithm for road transport. J. Intell. Transport. Syst., 10(3), 2006, pp. 103–115.
9. Taylor G., Blewitt G., Steup D., Corbett S., and Car A. Road reduction filtering for GPS-GIS navigation. Transactions in GIS, 5(3):193–207, 2001.
10. Tradisauskas N., Juhl J., Lahrmann H., and Jensen C.S. Map matching for intelligent speed adaptation. In Proc. 6th European Congress on Intelligent Transport Systems and Services, 2007.
11. White C.E., Bernstein D., and Kornhauser A.L. Some map matching algorithms for personal navigation assistants. Transport. Res. C, 8:91–108, 2000.

Mapping

- ▶ Mediation
- ▶ Schema Mapping

Mapping Composition

- ▶ Schema Mapping Composition

Mapping Engines

- ▶ Digital Rights Management

Markup Language

Ethan V. Munson
University of Wisconsin-Milwaukee, Milwaukee, WI, USA

Definition

A markup language is specification language that annotates content through the insertion of *marks* into the content itself. Markup languages differ from programming languages in that they treat data, rather than commands or declarations, as the primary element in the language.

Key Points

Markup languages were initially developed for text document formatting systems, though they are not limited to text. In fact, the term *markup* was taken directly from the jargon of the publishing business, where editors and typographers would "mark up" draft documents to indicate corrections or printing effects. Markup languages are generally quite declarative and have little, if any, computational semantics. The marks inserted into the content are often called "tags" because that term is used by XML.

Cross-references

- ▶ Document
- ▶ Document Representations (Inclusive Native and Relational)

MashUp

Alex Wun
University of Toronto, Toronto, ON, Canada

Definition

A MashUp is a web application that combines data from multiple sources, creating a new hybrid web application with functionality unavailable in the original individual applications that sourced the data.

Key Points

An emerging trend in web applications is to provide public APIs for accessing data that has traditionally been used only internally by those applications. The main purpose of providing access to traditionally private web application data is to encourage user-driven development. In other words, consumers are expected to take that public data and build custom applications for other consumers – thereby adding value to the original data sources. MashUps are web applications that take advantage of these publicly accessible data sources by correlating the data obtained from different sources and deriving some novel functionality. A simple and common example is correlating a data source that has location information (such as wireless hotspot locations) with cartographic data (from Google or Yahoo maps for example) to produce a graphical map of wireless hotspots.

MashUps are conceptually related to portals, which also collect data from multiple sources for presentation. However, portals perform server-side aggregation whereas MashUps can also perform this aggregation on the client-side (i.e., correlation can occur in the scripts of a web page). Additionally, portals present data collected from disparate sources together but without interaction between data sets. In contrast, MashUps focus heavily on merging disparate data sets into one unified representation. For example, a news portal would simply present a set of interesting articles gathered from various sources on a single page while a news MashUp would correlate textual news with related images and multimedia as well as automatically linking related articles in a single view.

While similar to MashUps, service composition is a more generic concept that focuses on orchestrating web service calls as part of some higher level application logic. The coordination of web service calls in a service composition are often more process-centric rather than data-centric. For example, a flight-booking composite service would query the flight reservation services of different airlines to book a flight based on customer requirements, while a flight-booking MashUp would gather flight data from various airlines and present the data to customers in a single unified view – likely correlated with other useful data such as weather.

There is currently no standardization of technologies or tools used to develop MashUps. In fact, many industry leaders such as Google, Microsoft, and Yahoo are pushing their own MashUp development tools. In particular, many of these tools are targeting non-programmers in hopes of expanding the base of users capable of contributing to and developing MashUps.

Cross-references

► AJAX
► Service
► Web 2.0/3.0

Massive Array of Idle Disks

Kazuo Goda
The University of Tokyo, Tokyo, Japan

Synonyms

MAID

Definition

The term Massive Array of Idle Disks refers to an energy-efficient disk array which has the capability of changing its disk drives into a low-power mode when the disk drives are not busy. Disk drives of the array may be controlled individually or in a group. The term Massive Array of Idle Disks is often abbreviated to MAID. A MAID disk array may have additional functions such as data migration/replication and access prediction to improve the energy saving.

Key Points

The basic idea of MAID is to save energy by exploiting storage access locality. That is, some disk drives which are installed in a disk array are frequently accessed whereas the others are rarely busy. MAID tries to spin down or power off such "low-temperature" disk drives to decrease the total energy consumption. The original papers [1,2] which introduced MAID in 2002 studied different design choices and configurations. MAID disk arrays are mainly used for archival storage (replacing conventional tape libraries) or near-line storage (which falls between online storage and archival storage).

Cross-references

► Storage Power Management

Recommended Reading

1. Colarelli D., and Grunwald D. Massive Arrays of Idle Disks for Storage Archives. In Proc. 2002 ACM/IEEE conf. on Supercomputing, 2002, pp. 1–11.
2. Colarelli D., Grunwald D., and Neufeld M. The Case for Massive Arrays of Idle Disks (MAID). In Proc. 1st USENIX Conf. on File and Storage Technologies, Work-in Progress Reports, 2002.
3. Storage Network Industry Association. The Dictionary of Storage Networking Terminology. Also available at http://www.snia.org/.

Matching

► Similarity and Ranking Operations

Materialized Query Tables

► Physical Database Design for Relational Databases

Materialized View Maintenance

► View Maintenance

Materialized View Redefinition

► View Adaptation

Materialized Views

► Physical Database Design for Relational Databases

Matrix

► Table

Matrix Masking

Stephen E. Fienberg, Jiashun Jin
Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Adding noise; Data perturbation; Recodings; Sampling; Synthetic data

Definition

Matrix Masking refers to a class of statistical disclosure limitation (SDL) methods used to protect confidentiality of statistical data, transforming an $n \times p$ (cases by variables) data matrix Z through pre- and post-multiplication and the possible addition of noise.

Key Points

Duncan and Pearson [3] and many others subsequently categorize the methodology used for SDL in terms of transformations of an $n \times p$ (cases by variables) data matrix Z of the form

$$Z \rightarrow AZB + C, \tag{1}$$

where A is a matrix that operates on the n cases, B is a matrix that operates on the p variables, and C is a matrix that adds perturbations or noise.

Matrix masking includes a wide variety of standard approaches to SDL: (i) adding noise, i.e., the C in matrix masking transformation of equation [1]; (ii) releasing a subset of observations (delete rows from Z), i.e., sampling; (iii) cell suppression for cross-classifications; (iv) including simulated data (add rows to Z); (v) releasing a subset of variables (delete columns from Z); (vi) switching selected column values for pairs of rows (data swapping). Even when one has applied a mask to a data set, the possibilities of both identity and attribute disclosure remain, although the risks may be substantially diminished.

Cross-references

► Individually Identifiable Data
► Inference Control in Statistical Databases
► Privacy
► Randomization Methods to Ensure Data Privacy
► Statistical Disclosure Limitation for Data Access

Recommended Reading

1. Doyle P., Lane J.I., Theeuwes J.J.M., and Zayatz L. (eds.). Confidentiality, Disclosure and Data Access: Theory and Practical Application for Statistical Agencies. Elsevier, New York, 2001.

2. Duncan G.T., Jabine T.B., and De Wolf V.A. (eds.). Private Lives and Public Policies. Report of the Committee on National Statistics' Panel on Confidentiality and Data Access. National Academy Press, WA, USA, 1993.
3. Duncan G.T. and Pearson R.B. Enhancing access to microdata while protecting confidentiality: prospects for the future (with discussion). Stat. Sci., 6:219–239, 1991.
4. Federal Committee on Statistical Methodology. Report on statistical disclosure limitation methodology. Statistical Policy Working Paper 22. U.S. Office of Management and Budget, WA, USA, 1994.

Maximal Itemset Mining

► Max-Pattern Mining

Max-Pattern Mining

Guimei Liu
National University of Singapore, Singapore, Singapore

Synonyms

Maximal itemset mining

Definition

Let $I = \{i_1, i_2..., i_n\}$ be a set of items and $D = \{t_1, t_2..., t_N\}$ be a transaction database, where $t_i (i \in [1,N])$ is a transaction and $t_i \subseteq I$. Every subset of I is called an *itemset*. If an itemset contains k items, then it is called a k-itemset. The support of an itemset X in D is defined as the percentage of transactions in D containing X, that is, $sup(X) = |\{t|t \in D \wedge X \subseteq t\}|/|D|$. If the support of an itemset exceeds a user-specified minimum support threshold, then the itemset is called a *frequent itemset* or a *frequent pattern*. If an itemset is frequent but none of its supersets is frequent, then the itemset is called a *maximal pattern*. The task of maximal pattern mining is given a minimum support threshold, to enumerate all the maximal patterns from a given transaction database.

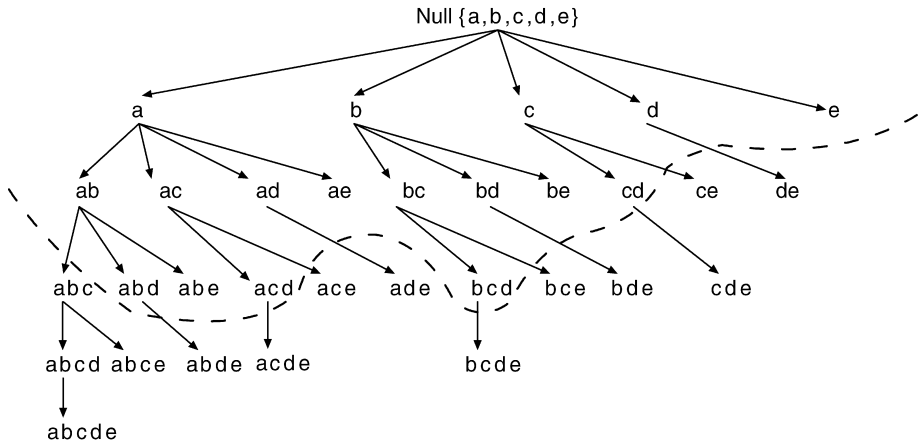
The concept of maximal patterns can be and has already been extended to more complex patterns, such as sequential patterns, frequent subtrees and frequent subgraphs. For each type of pattern, a pattern is maximal if it satisfies the given constraints but none of its super-patterns satisfies the given constraints.

Historical Background

If a k-itemset is frequent, then all of its subsets are frequent and the number of them is $2^k - 1$. Datasets collected from some domains can be very dense and contain very long patterns. Any algorithm which produces the complete set of frequent itemsets suffers from generating numerous short patterns on these datasets, and most of the short patterns may be useless. Some researchers have noticed the long pattern problem and suggested mining only maximal frequent patterns [1,3,8]. The set of maximal patterns provides a concise view of the frequent patterns, and it can be orders of magnitude smaller than the complete set of frequent patterns. The complete set of frequent patterns can be derived from the set of maximal patterns, but the support information is lost.

A different concept called *closed itemset* or closed pattern has also been proposed to reduce result size. A pattern is closed if none of its supersets has the same support as it does. Closed patterns retain the support information of frequent patterns. The complete set of frequent patterns can be derived from the set of frequent closed patterns without information loss. A maximal pattern must be a closed pattern, but not vice versa.

Given a set of items I, the search space of the *frequent itemset* mining problem is the power set of I, and it can be represented as a set-enumeration tree given a specific order of I [9]. Figure 1 shows the search space tree for $I = \{a, b, c, d, e\}$, and the items are sorted lexicographically. Every node in the search space tree represents an itemset. For every itemset X in the tree, only the items after the last item of X can be appended to X to form a longer itemset. These items are called *candidate extensions* of X, denoted as $cand\_ext(X)$. For example, items d and e are candidate extensions of ac, while item b is not a candidate extension of ac because b is before c in lexicographic order. Mining maximal patterns can be viewed as finding a border through the search space tree such that all the nodes below the border are infrequent and all the nodes above the border are frequent. As shown in Fig. 1, the dotted line represents the border. All the nodes above the dotted line are frequent and all the nodes below the dotted line are infrequent. Among all the nodes above the border, only leaf nodes can be maximal, but not all the leaf nodes are maximal; every internal node has at least one frequent child (superset) thus cannot be maximal. The goal of maximal pattern mining is to find the border by counting support for as less as



Max-Pattern Mining. Figure 1. Search space tree for $I = \{a, b, c, d, e\}$.

possible itemsets. Most, if not all, existing maximal pattern mining algorithms try to find some frequent long patterns first, and then use these long patterns to prune non-maximal patterns.

The first attempt at mining maximal patterns is made by Gunopoulos et al. [6], and an algorithm called *dualize and advance* is proposed. The algorithm is based on the observation that given a set of maximal patterns, any other maximal pattern not in the set must contain at least one common item with the complement of every maximal pattern in the set, where the complement of an itemset X is defined as $I - X$. The algorithm works as follows. It first uses a greedy search to generate some maximal patterns, denoted as $\mathcal{H}$, and then finds the minimal patterns that contain at least one common item with the complement of every maximal pattern in $\mathcal{H}$. Here a pattern is minimal if none of its subsets satisfies the condition. These minimal patterns are called minimal transversals of $\mathcal{H}$. If all the minimal transversals of H are infrequent, it means that all the maximal patterns are all in $\mathcal{H}$ already. Otherwise, there exists some minimal transversal X of $\mathcal{H}$ such that X is frequent, the algorithm then finds a maximal superset of X, denoted as Y, and Y must be a maximal pattern. The algorithm puts Y in $\mathcal{H}$, and then generates the minimal transversals of the updated $\mathcal{H}$. This process is repeated until no frequent minimal transversals of $\mathcal{H}$ exists. The upper bound for the time complexity of this algorithm is sub-exponential to the output size.

Pincer-search [7] combines the bottom-up and top-down search strategy, and approaches the border from both directions. The bottom-up search is similar to the A priori algorithm [2], and the top-down search is implemented by maintaining a set called maximum-frequent-candidate-set (MFCS). MFCS is a minimum cardinality set of itemsets such that the union of all the subsets of its elements contains all the frequent itemsets that have been discovered so far but does not contain any itemsets that have been determined to be infrequent. Pincer-search uses MFCS to prune those candidate itemsets that have a frequent superset in MFCS to reduce the database scan times and the support counting cost.

Both the dualize and advance algorithm and the Pincer-search algorithm maintain the set of candidate maximal patterns during the mining process and use them to prune short non-maximal patterns. The main difference between the two algorithms is that Pincer-Search considers large sets that may be frequent first, and then shrinks them to find the real maximal patterns, while the dualize and advance algorithm starts from some seed maximal patterns and uses them to find other maximal patterns. Both algorithms can prune non-maximal patterns effectively, but maintaining and manipulating the set of candidate maximal patterns can be very costly.

Zaki et al. propose two maximal pattern mining algorithms MaxClique and MaxEclat [13]. Both algorithms rely on a preprocessing step to cluster itemsets, and then use a hybrid bottom-up and top-down approach to find maximal patterns from each cluster with a vertical data representation. The two algorithms differ in how the itemsets are clustered. The purpose of the clustering step is to find some potential maximal patterns, and the two algorithms use these potential

maximal patterns to restrict the search space. However, the cost of the clustering step can be very high.

Max-Miner [3] is the first successful and practical algorithm for mining maximal patterns. It uses the bottom-up search strategy to traverse the search space as the A priori algorithm [2], but it always attempts to look ahead in order to quickly identify long patterns. By identifying a long pattern first, Max-Miner can prune all its subsets from consideration. Two pruning techniques are proposed in the Max-Miner algorithm: pruning based on superset frequency and the dynamic reordering technique. These two pruning techniques are very effective in removing non-maximal patterns, and have been adopted by all later maximal pattern mining algorithms.

1. *Superset frequency pruning*. This technique is also called the lookahead technique. It is based on the observation that the itemsets in the sub search space tree rooted at X are subsets of $X \cup cand\_ext(X)$. Therefore, if $X \cup cand\_ext(X)$ is frequent, then none of the itemsets in the subtree rooted at X can be maximal and the whole branch can be pruned. There are two ways to check whether $X \cup cand\_ext(X)$ is frequent. One way is to check whether $X \cup cand\_ext(X)$ is a subset of some maximal pattern that has already been discovered. The other way is to look at the support of $X \cup cand\_ext(X)$ in the database, which can be done when counting the support of X's immediate supersets.
2. *Dynamic item reordering*. At every node X, Max-Miner sorts the items in $cand\_ext(X)$ in ascending frequency order. The candidate extensions of an item include all the items that are after it in the ascending frequency order. Let i_1 and i_2 be two items in $cand\_ext(X)$. Item i_1 is before i_2 in the ordering if $sup(X \cup \{i_1\})$ is smaller than $sup(X \cup \{i_2\})$, and item i_2 is a candidate extension of $X \cup \{i_1\}$. The motivation behind dynamic item reordering is to increase the effectiveness of the superset frequency pruning technique. The superset frequency pruning can be applied when $X \cup cand\_ext(X)$ is frequent. It is therefore desirable to make many Xs satisfy this condition. A good heuristic for accomplishing this is to force the most frequent items to be the candidate extensions of all other items because items with high frequency are more likely to be part of long frequent itemsets.

Besides the above two pruning techniques, Max-Miner also uses a technique that can often determine when a new candidate itemset is frequent before accessing the database. The idea is to use information gathered during previous database passes to compute a good lower-bound on the number of transactions that contain the itemset.

The Max-Miner algorithm uses the breadth-first search order to explore the search space, which makes it not very efficient on dense datasets. DepthProject [1], MAFIA [4], GenMax [5] and AFOPT-Max [8] use the depth-first search strategy to traverse the search space. The depth-first search strategy is capable of finding long patterns first, which makes the superset frequency pruning technique more effective. These algorithms differ mainly in their support counting technique. Both DepthProject and MAFIA assume that the dataset fits in the main memory. At any point in the search, DepthProject maintains the projected transaction sets for some of the nodes on the path from the root to the node currently being processed, where a projected transaction of a node contains only the candidate extensions of the node. It is possible that a projected transaction is empty, in this case, the projected transaction is discarded. Since the projected database is substantially smaller than the original database both in terms of the number of transactions and the number of items, the process of finding the support counts is speeded up substantially. DepthProject also uses a bucketing technique to speed up the support counting. MAFIA and GenMax use the vertical mining technique, that is, each itemset is associated with a tid (transaction id) bitmap, and support counting is performed by tid bitmap join. MAFIA uses another pruning technique called parent equivalence pruning (PEP), which is essentially to remove frequent non-closed itemsets. GenMax[5] proposes a progressive focusing technique to improve the efficiency of superset searching. AFOPT-Max uses a prefix-tree structure to store projected transactions, which can make the lookahead pruning technique be performed more efficiently.

Foundations

In practice, mining maximal patterns is much cheaper than mining the complete set of frequent itemsets. However, the worst-case time complexity of maximal pattern mining is the same as mining all frequent patterns. Yang [11] studies the complexity of the

maximal pattern mining problem and proves that the problem of counting the number of distinct maximal frequent itemsets in a transaction database, given an arbitrary support threshold, is #P-complete, thereby providing strong theoretical evidence that the problem of mining maximal frequent itemsets is NP-hard.

The concept of maximal patterns has been extended to other similar data mining problems dealing with complex data structures. Yang et al. [12] devise an algorithm that combines statistical sampling and a technique called border collapsing to discover long sequential patterns with sufficiently high confidence in a noisy environment. Xiao et al. [10] propose an algorithm to mine maximal frequent subtrees from a database of unordered labeled trees.

Key Applications

Maximal pattern mining is applicable to dense domains where extracting all frequent patterns is not feasible. It can also be used as a preprocessing step to improve the efficiency of frequent pattern mining and to decide appropriate thresholds for frequent pattern mining and association rule mining.

Experimental Results

Each introduced method has an accompanying experimental evaluation in the corresponding reference. A comprehensive comparison of different algorithms can be found at http://fimi.cs.helsinki.fi/experiments/.

Data Sets

A collection of datasets commonly used for experiments can be found at http://fimi.cs.helsinki.fi/data/.

URL to Code

http://fimi.cs.helsinki.fi/src/.

Cross-references

► Closed Itemset Mining and Non-Redundant Association Rule Mining

► Frequent Itemsets and Association Rules

Recommended Reading

1. Agarwal R.C., Aggarwal C.C., and Prasad V.V.V. Depth first generation of long patterns. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 108–118.
2. Agrawal R. and Srikant R. Fast algorithms for mining association rules in large databases. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 487–499.
3. Bayardo R.J. Jr. Efficiently mining long patterns from databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 85–93.
4. Burdick D., Calimlim M., and Gehrke J. Mafia: A maximal frequent itemset algorithm for transactional databases. In Proc. 17th Int. Conf. on Data Engineering, 2001, pp. 443–452.
5. Gouda K. and Zaki M.J. GenMax: Efficiently mining maximal frequent itemsets. In Proc. 2001 IEEE Int. Conf. on Data Mining, 2001, pp. 163–170.
6. Gunopulos D., Mannila H., and Saluja S. Discovering all most specific sentences by randomized algorithms. In Proc. 6th Int. Conf. on Database Theory, 1997, pp. 215–229.
7. Lin D.I., and Kedem Z.M. Pincer search: A new algorithm for discovering the maximum frequent set. In Advances in Database Technology, Proc. 1st Int. Conf. on Extending Database Technology, 1998, pp. 105–119.
8. Liu G., Lu H., Lou W., Xu Y., and Yu J.X. Efficient mining of frequent patterns using ascending frequency ordered prefix-tree. Data Mining Knowledge Discovery, 9(3):249–274, 2004.
9. Rymon R. Search through systematic set enumeration. In Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning, 1992, pp. 268–275.
10. Xiao Y., Yao J.F., Li Z., and Dunham M.H. Efficient data mining for maximal frequent subtrees. In Proc. 2003 IEEE Int. Conf. on Data Mining, 2003, pp. 379–386.
11. Yang G. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2004, pp. 344–353.
12. Yang J., Wang W., Yu P.S., and Han J. Mining long sequential patterns in a noisy environment. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2002, pp. 406–417.
13. Zaki M.J., Parthasarathy S., Ogihara M., and Li W. New algorithms for fast discovery of association rules. In Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, 1997, pp. 283–286.

Maybe Answer

► Possible Answers

MDIS

► Meta Data Interchange Specification

MDR

► Metadata Registry, ISO/IEC 11179

MDS

► Multidimensional Scaling

Mean Average Precision

► MAP

Mean Reciprocal Rank

Nick Craswell
Microsoft Research Cambridge, Cambridge, UK

Synonyms

MRR; Mean Reciprocal Rank of the First Relevant Document; MRR1

Definition

The Reciprocal Rank (RR) information retrieval measure calculates the reciprocal of the rank at which the first relevant document was retrieved. RR is 1 if a relevant document was retrieved at rank 1, if not it is 0.5 if a relevant document was retrieved at rank 2 and so on. When averaged across queries, the measure is called the Mean Reciprocal Rank (MRR).

Key Points

Mean Reciprocal Rank is associated with a user model where the user only wishes to see one relevant document. Assuming that the user will look down the ranking until a relevant document is found, and that document is at rank n, then the precision of the set they view is 1/n, which is also the reciprocal rank measure. For this reason, MRR is equivalent to Mean Average Precision in cases where each query has precisely one relevant document. MRR is not a shallow measure, in that its value changes whenever the required document is moved, although the change is much larger when moving from rank 1 to rank 2 (change is 0.5) compared to moving from rank 100 to 1,000 (change of 0.009).

MRR is an appropriate measure for known item search, where the user is trying to find a document that he either has seen before or knows to exist. This is called *navigational search* in the case of web search. In a case where there are multiple copies of the required document, or otherwise a set of relevant documents that are substitutes, MRR can still be applied based on the first copy.

Cross-references

► Mean Average Precision
► Precision-Oriented Effectiveness Measures

Mean Reciprocal Rank of the First Relevant Document

► MRR (Mean Reciprocal Rank)

Measure

Torben Bach Pedersen
Aalborg University, Aalborg, Denmark

Synonyms

Numerical fact

Definition

A *measure* is a numerical property of a multidimensional *cube*, e.g., sales price, coupled with an aggregation formula, e.g., SUM. It captures numerical information to be used for aggregate computations.

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: DollarSales and ItemSales, capturing the sales price and the number of items sold, respectively. ItemSales can be viewed as a function: ItemSales: $Dom(P) \times Dom(T) \times Dom(S) \mapsto \mathbb{N}_0$ that given a certain combination of dimension values returns the total number of items sold for that combination. If a dimension value corresponds to a higher level in the dimension *hierarchy*, e.g., a product group or even *all* products, the result is an aggregation of several lower-level measure values.

In a multidimensional database, measures generally represent the properties of the chosen facts that the users want to study, e.g., with the purpose of optimizing them. Measures then take on different values for

different combinations of dimension values. The property and formula are chosen such that the value of a measure is meaningful for all combinations of aggregation levels. The formula is defined in the metadata and thus not stored redundantly with the data. Although most multidimensional data models have measures, some do not. In these, dimension values are also used for computations, thus obviating the need for measures, but at the expense of some user-friendliness [2].

It is important to distinguish three classes of measures, namely *additive*, *semi-additive*, and *non-additive* measures, as these behave quite differently in computations. Additive measure values can be combined meaningfully along any dimension. For example, it makes sense to add the total sales over Product, Store, and Time, as this causes no overlap among the real-world phenomena that caused the individual values. Semi-additive measure values cannot be combined along one or more of the dimensions, most often the Time dimension. Semi-additive measures generally occur for so-called "snapshot" facts. For example, it does not make sense to sum inventory levels across time, as the same inventory item, e.g., a specific product item, may be counted several times, but it is meaningful to sum inventory levels across products and stores. Non-additive measure values cannot be combined along any dimension, usually because of the chosen formula. For example, this occurs when averages for lower-level values cannot be combined into averages for higher-level values. The additivity of measures is related to the so-called "type" of the measure (Flow, Stock or Value-Per-Unit).

Cross-references

- ▶ Cube
- ▶ Dimension
- ▶ Hierarchy
- ▶ Multidimensional Modeling
- ▶ On-Line Analytical Processing
- ▶ Summarizability

Recommended Reading

1. Kimball R., Reeves L., Ross M., and Thornthwaite W. The Data Warehouse Lifecycle Toolkit. Wiley Computer Publishing, New York, 1998.
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.

Media Recovery

▶ Crash Recovery

Media Semantics

▶ Computational Media Aesthetics

Median

▶ Quantiles on Streams

Mediation

Cesare Pautasso
University of Lugano, Lugano, Switzerland

Synonyms

Transformation; Adaptation; Bridging; Mapping

Definition

Mediation is the process of reconciling differences to reach an agreement between different parties. In databases, the goal of mediation is to compute a common view over multiple, distinct, and heterogeneous sources of data. In software architecture, a component plays the role of mediator if it achieves interoperability by decoupling heterogeneous component having mismatching interfaces. Protocol mediation enables the exchange of information between autonomous endpoints that use incompatible communication protocols.

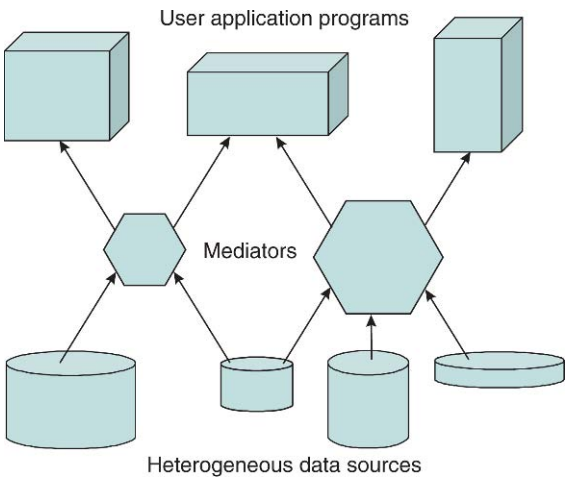
Mediation middleware helps applications deal with heterogeneity. By hiding the multiplicity and the complexity of the underlying systems, it transforms a one-to-many interaction (the application communicating with multiple data sources) into a simpler, one-to-one interaction (the application communicates with the mediator) and shifts the complexity of handling the communication with multiple, heterogeneous parties into a reusable component: the mediator.

Historical Background

In the context of information systems, the concept of mediation has been introduced by Gio Wiederhold in 1992 as the organizing principle for the interoperation of heterogeneous software components and data sources [8]. Mediation is performed by *a layer of intelligent middleware services in information systems, linking data resources and application programs* [9]. Programs need to consume information of multiple data resources and mediators offer them a solution to deal with representation and abstraction problems and exchange objects across multiple, heterogeneous information sources [5]. In the original vision, mediators were seen as independently developed, reusable software modules exploiting expert knowledge about the data to create aggregated information for application programs found at a higher level of abstraction. As shown in Fig. 1, queries from applications are translated by mediators into sub-queries sent out to the different data sources. The various sub-answers are then collected, integrated and returned by the mediator as a single answer to the application.

Foundations

Mediation techniques help to deal with the integration of heterogeneous and incompatible systems. A good understanding of the nature of the problem of dealing with heterogeneity helps to determine when and how mediation should be applied [6]. Conflicts requiring mediation may be found at the level of specific data elements (or messages) to be exchanged or at the level of schema (or interface metadata) definitions. Data values may be stored with multiple representations (syntactic mismatches) or the same representation may be interpreted in different ways (semantic mismatches). Data units (e.g., centimeters versus inches) are also a significant source of conflicts and potential bugs if they are not correctly accounted for when performing data fusion. Identification mismatches are due to the use of locally unique key identifiers that need to be reconciled when tuples from different databases are joined. At the schema level, conflicts involve mismatches in the naming of corresponding elements ("Customer" versus "Client" table); conflicts in the structure ("Street, Number, Zip, City" versus "Address") and granularity ("Purchase Order" versus "Order Line Item") of corresponding data types.



Mediation. Figure 1. Mediation architecture.

It is important to address such mismatches and conflicts at both data and schema levels. To do so, typical mediation middleware tools provide support for operations such as:

1. Selection, filtering, and aggregation of data
2. Data type, encoding and format conversion
3. Join, comparison, and fusion of data originating from multiple data sources
4. Resolution of conflicts between inconsistent sources
5. Multiplexing and demultiplexing over different channels
6. Lookup-based data translation
7. View materialization and caching of intermediate results

Mediator components can be developed using both imperative and declarative programming languages. Automatic schema matching techniques and algorithms that can be applied to support the development of mediators [7].

Mediation can be provided according to two styles:

- Standard-Based Mediation. The mediator transforms each of the incompatible interfaces to comply with a standardized interface that features the least common denominator between all representations. In order for two parties to communicate, this style requires performing two back-to-back transformations (to the standard representation and from the standard representation), thus introduces a larger performance overhead. However, in terms of development cost and maintainability of the mediator, providing support for a new interface

only requires introducing one additional transformation in the mediator (assuming that the new interface can be funneled through the standard one).

- Point-to-Point Mediation. The mediator directly maps each pair of incompatible interfaces. This way, the performance overhead is minimized as only one transformation is required as data is exchanged between two parties. Also, there is no need to define a standardized representation, which may be a difficult task in some cases. However, this style should be applied to mediate between a limited (and fixed) set of interfaces only. The complexity of maintaining this kind of mediator does not scale: adding support for a new interface requires to introduce *n* additional mappings in the mediator (one for each existing interface).

Key Applications

Mediation plays a prominent role in applications that require integrating data and functionality originally provided by separate systems (especially in database federation and data integration [10]). For example, a mediator performing currency conversion makes it possible to compare prices of products on sale in different markets. Likewise, mediation is needed to build a country-wide census database out of local databases maintained by different cities, if these have been created independently based on different data models and schemas.

In the context of Service-Oriented Architectures, the role of mediator is associated with the communication bus (or Enterprise Service Bus). Through the bus, services exchange messages without being aware that messages may be transformed while in transit to reconcile differences between service interfaces. Such transformations may be based on context and semantics metadata [4]. Mediation is one of the main features of the Web Services Modeling Framework [1], which applies it to address heterogeneity going beyond traditional data mappings. Also mediators for Business Logic (to compensate between different message ordering constraints found in the public processes of service) and Message Exchange Protocols (for translation between different transport protocols, e.g., to provide reliable and secure message delivery) are introduced in the framework.

In object-oriented design, the *mediator behavioral pattern* has been applied to decouple multiple collaborating objects [3, p. 273]. Instead of having the objects depend on each other and partitioning the interaction logic among them, a mediator object is introduced. It contains and centralizes some potentially complex interaction logic. All objects only refer to the mediator and interact through it. This pattern has also been named *mapper* in the context of enterprise application architecture [2]. Similar to the other applications of mediation, also in this case the interfaces and the overall interactions are simplified thanks to the mediator. However, the price of employing an additional layer of indirection between the elements of a system must be paid.

Cross-references

- ▶ Enterprise Service Bus
- ▶ Event Transformation
- ▶ Schema Mapping
- ▶ View Adaptation

Recommended Reading

1. Fensel D. and Bussler C. The web service modeling framework WSMF. Electron. Comm. Res. Appl., 1(1):113–137, 2002.
2. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, Reading, MA, November 2002.
3. Gamma E., Helm R., Johnson R., and Vlissides J. Design Patterns: Elements of Reusable Software. Addison-Wesley, Reading, MA, 1995.
4. Mrissa M., Ghedira C., Benslimane D., Maamar Z., Rosenberg F., and Dustdar S. A context-based mediation approach to compose semantic web services. ACM Trans. Internet Technol., 8(1):4, 2007.
5. Papakonstantinou Y., Garcia-Molina H., and Widom J. Object exchange across heterogeneous information sources. In Proc. 11th Int. Conf. on Data Engineering, 1995, pp. 251–260.
6. Park J. and Ram S. Information systems interoperability: What lies beneath? ACM Trans. Inf. Syst., 22(4):595–632, 2004.
7. Rahm E. and Bernstein P.A. A survey of approaches to automatic schema matching. Int. VLDB J., 10(4):334–350, December 2001.
8. Wiederhold G. Mediators in the architecture of future information systems. Computer, 25(4):38–49, 1992.
9. Wiederhold G. and Genesereth M.R. The conceptual basis for mediation services. IEEE Expert, 12(5):38–47, 1997.
10. Ziegler P. Data Integration Project World-Wide, 2008. http://www.ifi.unizh.ch/~pziegler/IntegrationProjects.html.

Mediation and Adaptation

- ▶ Database Middleware

Medical Genetics

► Implications of Genomics for Clinical Informatics

MEDLINE/ PubMed

► Biomedical Scientific Textual Data Types and Processing

Membership Query

Mirella M. Moro
The Federal University of Rio Grande do Sol, Porte Alegre, Brazil

Synonyms

Equality query; Equality selection

Definition

Consider a relation R whose schema contains some attribute A taking values over a domain D. A *membership query* retrieves all tuples in R with $A = x$ ($x \in D$).

Key Points

A membership query effectively checks membership in a set (relation). As such, it can be implemented using either a hash-based index (built on the attribute(s) involved in the query) or a B+-tree. If a hashing scheme is used, each indexed value is placed on an appropriate hash bucket. Then all records that satisfy $A = x$ are located on the bucket responsible for value x. If A is a numeric attribute (and can thus be indexed using an order-preserving access method like a B+-tree) the membership query is a special case of a range query where the range interval *[low, high]* is reduced to a single value ($low = high = x$).

Cross-references

► Access Methods
► B+-Tree
► Hashing

Memory Consistency

► Consistency Models For Replicated Data

Memory Hierarchy

Stefan Manegold
CWI, Amsterdam, The Netherlands

Synonyms

Hierarchical memory system

Definition

A Hierarchical Memory System – or Memory Hierarchy for short – is an economical solution to provide computer programs with (virtually) unlimited fast memory, taking advantage of locality and cost-performance of memory technology. Computer storage and memory hardware – from disk drives to DRAM main memory to SRAM CPU caches – shares the limitation that as they became faster, they become more expensive (per capacity), and thus smaller. Consequently, memory hierarchies are organized into several levels, starting from huge and inexpensive but slow disk systems to DRAM main memory and SRAM CPU caches (both off and on chip) to registers in the CPU core. Each level closer to the CPU is faster but smaller than the next level one step down in the hierarchy. Memory hierarchies exploit the principle of locality, i.e., the property that computer programs do not access all their code and data uniformly, but rather focus on referencing only small fractions for given periods of time. Consequently, during each period of time, only the fraction currently referenced – also called hot-set, locality-set, or working-set – needs to be present in the fastest memory level, while the remaining data and code can stay in slower levels. In general, all data in one level is also found in all (slower but larger) memory levels below it.

Historical Background

A three-level memory hierarchy, consisting of (i) the CPU's registers, (ii) DRAM main-memory as primary storage and (iii) secondary storage, has been in use since the introduction of drums and then disk drives as secondary storage. In this memory hierarchy, the decision of which data are loaded into a higher level at what time (and written back in case it is modified) is completely under software control. Application programs determine when to load data from secondary to primary memory, and compilers determine when to load data from main memory into CPU registers.

With the introduction of virtual memory around 1960 [10] application software – and in particular their programmers – are provided with uniformly addressable memory larger than physical memory. The operating system takes care of loading the references portion of data into main memory, automating page transfers, and hence relieving programmers from this task. In the late 1960s, the discovery of the locality principle [7] led to the invention of working sets [6] that exploit locality properties to predict data references, and enabled the design of page replacement algorithms that make virtual memory work efficiently and reliably in multiprogramming environments. Since then, virtual memory has become an inherent feature of operating systems. In contrast to many other application programs, a database management system usually does not rely on the operating system's generic virtual memory management, only. Instead, it implements its own buffer pool, exploiting domain specific knowledge to implement application-specific replacement algorithms.

Until the 1980s, the three-level memory hierarchy has been the state-of-the art, mainly because main memory is considered fast enough to serve the CPU – or better, the CPU were "slow enough." Since then, a continuously growing performance gap develops between CPU and main memory. With the chip integration technology following Moore's Law [12], i.e., doubling the number of transistors per chip area roughly every 1.5 years, the performance has grown exponentially, due to increasing clock-speeds, increasing inherent parallelism, or both. Advanced manufacturing techniques has grown the capacity and – thanks to even wider and faster system buses – their data transfer bandwidth of main memory similarly. Memory access latency, however, has lagged behind, demonstrating at most a slight linear improvement.

To bridge this performance gap, in the 1980's hardware designers started to extend the memory hierarchy by adding small (and expensive) but fast SRAM cache memories between the CPU and main memory. Initially, a single cache level is added, either located on the system board between CPU and main memory, or integrated on the CPU chip. With advancing integration and manufacturing techniques, more levels are added. Nowadays, two to three cache levels integrated on the CPU chip represent the most common configuration.

The main difference between cache memories and the original three levels of the memory hierarchy is that their contents are completely controlled by hardware logic. Relying on the locality principle, carefully turned replacement algorithms (usually variations of LRU), decide when data are loaded into or evicted from which cache level. While ensuring transparent use and easy portability, this approach leaves programs virtually without means to explicitly control the content of CPU caches. In modern systems, software prefetching commands have been introduced to provide programs with limited control to (pre-)load data into caches without actually accessing it.

The scientific computing and algorithm communities – both usually focusing on compute-intensive tasks on memory-based data sets – quickly realized that they have to make the algorithms and data structures *cache-conscious* to exploit the performance potentials of ever faster CPU and CPU caches effectively and efficiently.

The database world initially ignored the new hardware developments, assuming that optimizing disk access (I/O) is still the key to high performance execution of data-intensive tasks on large disk-based data sets. First proposals of cache-conscious database algorithms occurred in the mid 1990s [14]. It was not until the end of the 20th century that the database community realized that memory access has become a severe bottleneck also for database query processing performance, taking up to 90% of the execution time [4,5]. In the last decade, the development of hardware-aware database technology from system architectures over data structures to query processing algorithms has become a very active and recognized research area [1–3,13].

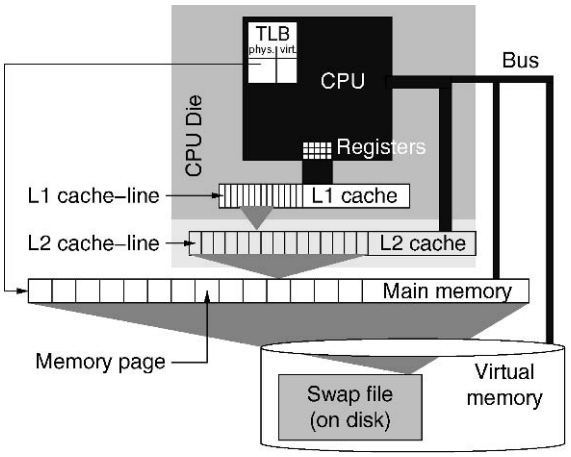
Foundations

Memory- and Cache-Architectures

Modern computer architectures have a *hierarchical memory system*, as depicted in Fig. 1. The main memory on the system board consists of *DRAM* (*Dynamic Random Access Memory*) chips. While CPU speeds are increasing rapidly, DRAM access latency has hardly progressed over time. To narrow the exponentially growing performance gap between CPU speed and memory latency (cf., Fig. 2), *cache memories* have been introduced, consisting of fast but expensive *SRAM* (*Static Random Access Memory*) chips. SRAM cells are usually made-up of six transistors per memory bit, and hence, they consume a rather large area on the

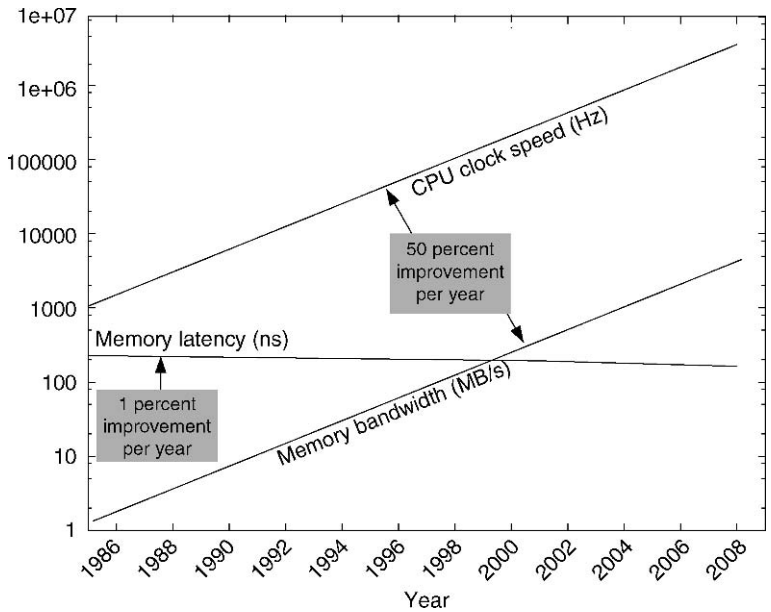
chips. DRAM cells require a single transistor and a small capacitor to store a single bit. Thus, DRAMs can store much more data than SRAMs of equal (physical) size. But due to some current leakage, the capacitor in DRAMs get discharged over time, and have to be recharged (*refreshed*) periodically to keep their information. These refreshes slow down access.

The fundamental principle of all cache architectures is "*reference locality*," i.e., the assumption that at any time the CPU, thus the program, repeatedly accesses only a limited amount of data (i.e., memory) that fits in the cache. Only the first access is "slow," as the data have to be loaded from main memory. This is called a *compulsory cache miss* (see below). Subsequent accesses (to the same data or memory addresses) are then "fast", as the data are then available in the cache. This is called a *cache hit*. The fraction of memory accesses that can be fulfilled from the cache is called *cache hit rate*; analogously, the fraction of memory accesses that cannot be fulfilled from the cache is called *cache miss rate*.



Memory Hierarchy. Figure 1. Hierarchical memory architecture.

Cache memories are often organized in *multiple cascading levels* between the main memory and the CPU. As they become faster, but smaller, the closer they are to the CPU. Originally, there was one level (typically 64 KB to 512 KB) of cache memory located on the system board. As the chip manufacturing processes improved, a small cache of about 4 KB to 16 KB was integrated on the CPU's die itself, allowing much faster access. The on-board cache is typically not replaced by the on-chip cache, but rather both make up a cache hierarchy, with the one on chip called *first level* (*L1*) cache and the one on board called *second level* (*L2*) cache. Over time, the L2 cache has also been integrated on the CPU's die (e.g., with Intel's Pentium III "Coppermine," or AMD's Athlon "Thunderbird"). On PC systems, the on-board cache has since disappeared, keeping two cache levels. On other platforms, e.g., workstations based on Compaq's (formerly DEC's)



Memory Hierarchy. Figure 2. Trends in CPU and DRAM speed.

Alpha CPU, the on-board cache is kept as *third level* (*L3*) cache, next to the two levels on the die. Most recent multi-core CPUs usually have a private L1 cache of 16 KB to 64 KB per core. The L2 cache is also integrated on the CPU's die. L2 configuration vary between one private L2 per core to one single L2 that is shared among all cores. On Intel's Core 2 Quad, for instance, each of the two L2 caches is shared by two of the four cores. Typical L2 sizes are in the order of 1 MB to 8 MB.

To simplify presentation, the remainder of this entry assumes a typical system with two cache levels (L1 and L2). However, the discussion can easily be generalized to an arbitrary number of cascading cache levels in a straightforward way.

In practice, caches memories do not only cache the data used by an application, but also the program itself, more accurately, the instructions that are currently being executed. With respect to caching, there is one major difference between data and program. Usually, a program must not be modified while it is running, i.e., the caches may be read-only. Data, however, requires caches that also allow modification of the cached data. Therefore, almost all systems nowadays implement two separate L1 caches, a read-only one for instructions and a read-write one for data. The L2 cache, however, is usually a single "unified" read-write cache used for both instructions and data.

Caches are characterized by three major parameters: Capacity (C), Line Size (Z), and Associativity (A):

Capacity (C) A cache's capacity defines its total size in bytes. Typical cache sizes range from 8 KB to 8 MB.

Line Size (Z) Caches are organized in *cache lines*, which represent the smallest unit of transfer between adjacent cache levels. Whenever a cache miss occurs, a complete cache line (i.e., multiple consecutive words) is loaded from the next cache level or from main memory, transferring all bits in the cache line in parallel over a wide bus. This exploits spatial locality, increasing the chances of cache hits for future references to data that are "close to" the reference that caused a cache miss. Typical cache line sizes range from 16 bytes to 256 bytes. Dividing the cache capacity by the cache line size yields the *number of available cache lines* in the cache: $\# = C/Z$.

Associativity (A) To which cache line the memory is loaded depends on the memory address and on the cache's *associativity*. An *A-way set associative* cache allows loading a line in A different positions. If $A > 1$, some *cache replacement* policy chooses one from among the A candidates. *Least Recently Used* (*LRU*) is the most common replacement algorithm. In case $A = 1$, the cache is called *directly-mapped*. This organization causes the least (virtually no) overhead in determining the cache line candidate. However, it also offers the least flexibility and may cause a lot of *conflict misses* (see below). The other extreme case is a *fully associative* cache. Here, each memory address can be loaded to any line in the cache ($A = \#$). This avoids conflict misses, and only *capacity misses* (see below) occur as the cache capacity is exceeded. However, determining the cache line candidate in this strategy causes a relatively high overhead that increases with the cache size. Hence, it is feasible for only smaller caches. Current PCs and workstations typically implement two-way to eight-way set associative caches.

With multiple cache levels, two types are distinguished: inclusive and exclusive caches. With *inclusive caches*, all data stored in L1 is also stored in L2. As data are loaded from memory, they are stored in all cache levels. Whenever a cache line needs to be replaced in L1 (because a mapping conflict occurs or as the capacity is exceeded), its original content can simply be discarded as another copy of that data still remains in the (usually larger) L2. The new content is then loaded from where it is found (either L2 or main memory). The total capacity of an inclusive cache hierarchy is hence determined by the largest level. With *exclusive caches*, all cached data are stored in exactly one cache level. As data are loaded from memory, they get stored only in the L1 cache. When a cache lines needs to be replaced in L1, its original content is first written back to L2. If the new content is then found in L2, it is moved from L2 to L1, otherwise, it is copied from main memory to L1. Compared to inclusive cache hierarchies, exclusive cache hierarchies virtually extend the cache size, as the total capacity becomes the sum of all levels. However, the "swap" of cache lines between adjacent cache levels in case of a cache miss also causes more "traffic" on the bus and hence increases the cache miss latency.

Cache misses can be classified into the following disjoint types [9]:

Compulsory The very first reference to a cache line always causes a cache miss, which is hence classified as a compulsory miss, i.e., an unavoidable miss (even) in an infinite cache. The number of compulsory misses obviously depends only on the data volume and the cache line size.

Capacity A reference that misses in a fully associative cache is classified as a capacity miss because the finite sized cache is unable to hold all the referenced data. Capacity misses can be minimized by increasing the temporal and spatial locality of references in the algorithm. Increasing cache size also reduces the capacity misses because it captures more locality.

Conflict A reference that hits in a fully associative cache but misses in an *A*-way set associative cache is classified as a conflict miss. This is because even though the cache is large enough to hold all the recently accessed data, its associativity constraints force some of the required data out of the cache prematurely. For instance, alternately accessing just two memory addresses that "happen to be" mapped to the same cache line will cause a conflict cache miss with each access. Conflict misses are the hardest to remove because they occur due to address conflicts in the data structure layout and are specific to a cache size and associativity. Data structures would, in general, have to be remapped to minimize conflicting addresses. Increasing the associativity of a cache will decrease the conflict misses.

Coherence Only in case of multi-processor or multi-core systems with private per processor/core high-level caches but shared lower-level caches and/or main memory, the following can occur. If two or more cores access the same data item, it will be loaded in each private cache. If one core than modifies this data item in its private cache, the other copies are invalidated and cannot server futures references. Instead, to ensure cache coherence, a cache miss occurs, and the modified data item has to be loaded from the cache that holds the most up-to-date copy.

Memory Access Costs

In general, memory access costs are characterized by the following three aspects:

Latency Latency is the time span that passes after issuing a data access request until the requested data is available in the CPU. In hierarchical memory systems, the latency increases with the distance from the CPU. Accessing data that are already available in the L1 cache causes *L1 access latency* (λ_{L1}), which is typically rather small (one or two CPU cycles). In case the requested data are not found in L1, an *L1 miss* occurs, additionally delaying the data access by *L2 access latency* (λ_{L2}) for accessing the L2 cache. Analogously, if the data are not yet available in L2, an *L2 miss* occurs, further delaying the access by *memory access latency* (λ_{Mem}) to finally load the data from main memory. Hence, the total latency to access data that are in neither cache is $\lambda_{Mem} + \lambda_{L2} + \lambda_{L1}$. As L1 accesses cannot be avoided, L1 access latency is often assumed to be included in the pure CPU costs, leaving only memory access latency and L2 access latency as explicit memory access costs. As mentioned above, all current hardware actually transfers multiple consecutive words, i.e., a complete cache line, during this time.

When a CPU requests data from a certain memory address, modern DRAM chips supply not only the requested data, but also the data from subsequent addresses. The data are then available without additional address request. This feature is called *Extended Data Output* (EDO). Anticipating sequential memory access, EDO reduces the effective latency. Hence, two types of latency for memory access need to be distinguished. *Sequential access latency* (λ^s) occurs with sequential memory access, exploiting the EDO feature. With random memory access, EDO does not speed up memory access. Thus, *random access latency* (λ^r) is usually higher than sequential access latency.

Bandwidth Bandwidth is a metric for the data volume (in megabytes) that can be transferred between CPU and main memory per second. Bandwidth usually decreases with the distance from the CPU, i.e., between L1 and L2 more data can be transferred per time than between L2 and main memory. The different bandwidths are referred to as *L2 access bandwidth* (β_{L2}) and *memory access bandwidth* (β_{Mem}), respectively. In conventional hardware, the memory bandwidth used to be simply the cache line size divided by the memory latency. Modern multiprocessor systems typically provide excess bandwidth capacity $\beta' \geq \beta$. To exploit this, caches need to be *non-blocking*, i.e., they need to allow more than one outstanding memory load at a time, and the CPU has to be able to issue subsequent load requests while waiting for the first one(s) to be

resolved. Further, the access pattern needs to be sequential, in order to exploit the EDO feature as described above.

Indicating its dependency on sequential access, the excess bandwidth is referred to as *sequential access bandwidth* ($\beta^s = \beta'$). The respective *sequential access latency* is defined as $\lambda^s = Z/\beta^s$. For *random access latency* as described above, the respective *random access bandwidth* is defined as $\beta^r = Z/\lambda^r$.

On some architectures, there is a difference between read and write bandwidth, but this difference tends to be small.

Address Translation For data access, logical virtual memory addresses used by application code have to be translated to physical page addresses in the main memory of the computer. In modern CPUs, a *Translation Lookaside Buffer* (*TLB*) is used as a cache for physical page addresses, holding the translation for the most recently used pages (typically 64). If a logical address is found in the TLB, the translation has no additional costs. Otherwise, a *TLB miss* occurs. The more pages an application uses (which also depends on the often configurable size of the memory pages), the higher the probability of TLB misses.

The actual *TLB miss latency* (l_{TLB}) depends on whether a system handles a TLB miss in hardware or in software. With software-handled TLB, TLB miss latency can be up to an order of magnitude larger than with hardware-handled TLB. Hardware-handled TLB fetches the translation from a fixed memory structure that is just filled by the operating system. Software-handled TLB leaves the translation method entirely to the operating system, but requires trapping to a routine in the operating system kernel on each TLB miss. Depending on the implementation and hardware architecture, TLB misses can therefore be more costly than a main-memory access. Moreover, as address translation often requires accessing some memory structure, this can in turn trigger additional memory cache misses.

TLBs can be treated similar to memory caches, using the memory page size as their cache line size, and calculating their (virtual) capacity as *number_of _entries* × *page_size*. TLBs are usually fully associative. Like caches, TLBs can be organized in multiple cascading levels.

For TLBs, there is no difference between sequential and random access latency. Further, bandwidth is irrelevant for TLBs, because a TLB miss does not cause any data transfer.

Unified Hardware Model

Summarizing the above discussion, one can describe a computer's memory hardware as a cascading hierarchy of N levels of caches (including TLBs) [11]. An index $i \in \{1,...,N\}$ added to the parameters described above identifies to the respective value of a specific level. The relation between access latency and access bandwidth then becomes $\lambda_{i+1} = Z_i/\beta_{i+1}$. Exploiting the dualism that an access to level $i + 1$ is caused a miss on level i allows some simplification of the notation. Introducing the *miss latency* $l_i = \lambda_{i+1}$ and the respective *miss bandwidth* $b_i = \beta_{i+1}$ yields $l_i = Z_i/b_i$. Each cache level is characterized by the parameters given in Table 1. Costs for L1 cache accesses are assumed to be included in the CPU costs, i.e., λ_1 and β_1 are not used and hence undefined.

Manegold developed a system independent C program called *Calibrator* to measure these parameters on any computer hardware.

Key Applications

In the last decade, the database community has done much research on modifying existing and developing new database technology (system architecture, data

Memory Hierarchy. Table 1. Characteristic parameters per cache level ($i \in \{1,..., N\}$)[2]

Description	Unit	Symbol
Cache name (level)	–	L_i
Cache capacity	[bytes]	C_i
Cache block size	[bytes]	Z_i
Number of cache lines	–	$\#_i = C_i/Z_i$
Cache associativity	–	A_i
Sequential access		
Access bandwidth	[bytes/ns]	β^S_{i+1}
Access latency	[ns]	$\lambda^s{}_{i+1} = Z_i/\beta^S_{i+1}$
Miss latency	[ns]	$l_i{}^s = \lambda^S_{i+1}$
Miss bandwidth	[bytes/ns]	$b_i{}^s = \beta^S_{i+1}$
Random access		
Access latency	[ns]	$\lambda^r{}_{i+1}$
Access bandwidth	[bytes/ns]	$\beta^r{}_{i+1} = Z_i/\lambda^r_{i+1}$
Miss bandwidth	[bytes/ns]	$b_i{}^r = \beta^r_{i+1}$
Miss latency	[ns]	$l_i{}^r = \lambda^r_{i+1}$

structures, query processing algorithms) to exploit the characteristics of the extended memory hierarchy efficiently and effectively, improving query evaluation performance up to orders of magnitude.

URL to Code

Manegold's cache-memory and TLB calibration tool *Calibrator* is available at http://homepages.cwi.nl/~manegold/Calibrator/calibrator.shtml

Cross-references

- ▶ Buffer Management
- ▶ Buffer Manager
- ▶ Buffer Pool
- ▶ Cache-Conscious Query Processing
- ▶ Locality
- ▶ Main Memory
- ▶ Main Memory DBMS
- ▶ Processor Cache
- ▶ Secondary Memory

Recommended Reading

1. Ailamaki A., Boncz P.A., and Manegold S. (eds.). Proc. Workshop on Data Management on New Hardware, 2005.
2. Ailamaki A., Boncz P.A., and Manegold S. (eds.). Proc. Workshop on Data Management on New Hardware, 2006.
3. Ailamaki A. and Luo Q. (eds.) Proc. Workshop on Data Management on New Hardware, 2007.
4. Ailamaki A.G., 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.
5. 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.
6. Denning P.J. The working set model for program behaviour. Commun. ACM, 11(5):323–333, 1968.
7. Denning P.J. The locality principle. Commun. ACM, 48 (7):19–24, 2005.
8. Hennessy J.L. and Patterson D.A. Computer Architecture – A Quantitative Approach, 3rd edn. Morgan Kaufmann, San Mateo, CA, USA, 2003.
9. Hill M.D. and Smith A.J. Evaluating associativity in CPU caches. IEEE Trans. Comput., 38(12):1612–1630, December 1989.
10. Kilburn T., Edwards D.B.C., Lanigan M.I., and Sumner F.H. One-level storage system. IRE Trans. Electronic Comput., 2(11):223–235, April 1962.
11. Manegold S. Understanding, Modeling, and Improving Main-Memory Database Performance. PhD thesis, Universiteit van Amsterdam, Amsterdam, The Netherlands, December 2002.
12. Moore G.E. Cramming more components onto integrated circuits. Electronics, 38(8):114–117, April 1965.
13. Ross K. and Luo Q. (eds.). In Proc. Workshop on Data Management on New Hardware, 2007.
14. Shatdal A., Kant C., and Naughton J. Cache conscious algorithms for relational query processing. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 510–512.

Memory Locality

Stefan Manegold
CWI, Amsterdam, The Netherlands

Synonyms

Principle of locality; Locality principle; Locality of reference

Definition

Locality refers to the phenomenon that computer programs — or computational processes in general — do not access all of their data items uniformly and independently, but rather in a clustered and/or dependent/correlated manner. Some data items are accessed more often than others, repeated accesses to the same data item occur in bursts, and related items are usually accessed together, concurrently or within a short time interval.

There are two types of locality:

1. *Temporal locality* means that accesses to the same data item are grouped in time, i.e., multiple accesses to the same data item occur in rather short time intervals compared to rather long time periods where the same data item is not accessed. Hence, temporal locality is the concept that a data item that is referenced by a program at one point in time will be referenced again sometime in the near future.
2. *Spatial locality* means that data items that are stored physically close to each other tend to be accessed together. Hence spatial locality is the concept that likelihood of referencing a data item by a program is higher if a data item near it has been referenced recently.

Locality belongs to the most fundamental principles of computer science.

Key Point

The discovery of the locality principle dates back to the 1960s. Denning's pioneering work on working-set memory management exploits the locality principle to avoid thrashing of virtual memory systems high levels of multiprogramming [2,3]. Today, this is the

key to making virtual memory systems work reliably and efficiently.

In particular, with modern hierarchical memory architectures, exploiting and increasing locality in algorithms and data structures is the key to achieving high performance.

Increased temporal locality ensures that once a data item is referenced, and hence loaded into a fast high-level memory (e.g., cache), all subsequent references occur in short succession while the data item is still available in the cache. Ideally, this data item will not be required, again, once it is evicted from the fast high-level memory.

Data transfer between adjacent levels of hierarchical memory systems does not happen per byte but with larger granularities, e.g., pages of multiple KB or even MB at a time between disk and main memory, cache lines of tens to hundreds of bytes between main memory and CPU cache. *Increased spatial locality* ensures that all data bytes/items that are loaded with each transfer are indeed useful for the program.

Database system architecture exploits and increases locality in numerous ways. Key examples for increased temporal locality are, for instance partitioned join algorithms, where iterating over small partition of the outer relation increases temporal locality of repeated access to the inner relation [5,4]. In fact, smaller partitions of the inner relation also increased spatial locality. Examples of spatial locality range from clustered indices over tuned page layouts such as PAX [1] to the decision between column-stores and row-stores to optimal support of column-major (OLAP) of row-major (OLTP) workloads.

Cross-references

- ▶ Buffer Management
- ▶ Buffer Manager
- ▶ Buffer Pool
- ▶ Cache-Conscious Query Processing
- ▶ Main Memory
- ▶ Main Memory DBMS
- ▶ Memory Hierarchy
- ▶ Processor Cache
- ▶ Secondary Memory

Recommended Reading

1. Ailamaki A.G., 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.
2. Denning P.J. The working set model for program behaviour. Commun. ACM, 11(5):323–333, 1968.
3. Denning P.J. The locality principle. Commun. ACM, 48(7):19–24, 2005.
4. Manegold S., Boncz P.A., and Kersten M.L. Optimizing main-memory join on modern hardware. IEEE Trans. Knowl. Data Eng., 14(4):709–730, July 2002.
5. Shatdal A., Kant C., and Naughton J. Cache conscious algorithms for relational query processing. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 510–512.

Merge Join

▶ Sort-Merge Join

Merge-purge

▶ Deduplication in Data Cleaning
▶ Record Matching

Merkle Hash Trees

▶ Merkle Trees

Merkle Trees

Barbara Carminati
University of Insubria, Varese, Italy

Synonyms

Merkle hash trees; Hash trees; Authentication trees

Definition

Merkle trees are data structures devised to authenticate, with a unique signature, a set of messages, by at the same time making an intended verifier able to verify authenticity of a single message without the disclosure of the other messages. In particular, given a set of messages $M = \{m_1,...,m_n\}$, the Merkle tree created with them is a binary tree whose leaves contain

the hash value of each message m in M, whereas internal nodes contain the concatenation of the hash values corresponding to its children.

Key Point

A Merkle tree is a data structure introduced by Merkle in 1979 [1] to improve the Lamport-Diffie one-time signature scheme [2]. In this digital signature scheme, keys can be used to sign, at most, one message. This implies that for each signed message a new public key has to be generated and published. As consequence, Lamport-Diffie one-time digital signature scheme requires publishing a large amount of data. To overcome this drawback, in [1] Merkle proposed a tree-structure, called *authentication tree*, with the aim of authenticating a large number of public keys to be used in one-time signature scheme.

In general, Merkle trees can be exploited to authenticate with a unique signature a set of messages by, at the same, time making an intended verifier able to authenticate a single message without the disclosure of the other messages. Given a set of messages $M = \{m_1,...,m_n\}$, the corresponding Merkle tree is computed by means of the following bottom-up recursive computation: at the beginning, for each different message $m \in M$, a different leaf containing the hash value of m is inserted into the tree; then, for each internal node, the value associated with it is equal to $h(h_l||h_r)$, where $h_l||h_r$ denotes the concatenation of the hash values corresponding to the left and right children nodes, and $h()$ is an hash function. The root node of the resulting binary hash tree is the digest of all the messages, and thus it can be digitally signed by using a standard signature technique. The main benefit of this method is that a user is able to validate the signature by having a subset of messages, providing him/her with a set of additional hash values corresponding to missing messages. Indeed, these additional hash values, together with the provided set of original messages, make a user able to locally re-build the binary tree and, therefore, to validate the signature generated on its root. Consider, for instance, the following set of messages $M = \{m_1, m_2, m_3, m_4\}$. The Merkle tree created with them is a complete binary tree with height of two. More precisely, according to the recursive computation, the root value of the Merkle tree is equal to $h(hr_l||hr_r)$, where hr_l is its left children with value $h(h(m_1)||h(m_2))$, whereas hr_r is its right children with value $h(h(m_3)||h(m_4))$. Assume, now, that a user receives only messages m_1 and m_2. To make him/her able to validate the signature, he/she must be provided also with hash value hr_r. Indeed, by having m_1 and m_2 messages, the user is able to calculate hr_l. Then, using hr_r and hr_l he/she can compute the hash value of the root, and thus verify the signature.

Cross-references

- ▶ Digital Signatures
- ▶ Secure Data Outsourcing

Recommended Reading

1. Merkle R. Secrecy, authentication, and public key systems. Electrical Engineering, PhD Thesis, Stanford University, 1979.
2. Lamport L. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, 1979.

Message Authentication Codes

Marina Blanton
University of Notre Dame, Notre Dame, IN, USA

M

Synonyms

MAC; Message integrity codes

Definition

A *message authentication code* (MAC) is a short fixed-length value which is used to authenticate a message. A MAC algorithm can be viewed as a hash function that takes as input two functionally distinct values: a secret key and a message. The output of a MAC algorithm is a short string computed in such a way that it is infeasible to produce the same output on the message without the knowledge of the key. Thus, the MAC value protects both the *integrity* and *authenticity* of a message by allowing the entity in possession of the secret key to detect any changes to the message content.

Key Points

While MAC functions can be viewed as keyed cryptographic hash functions, they have specific security requirements for authentication purposes. More precisely, an attacker who does not have access to the secret key and has not seen the MAC value for a specific message before should not be able to compute that value. MAC functions use symmetric techniques (i.e., the same key is used to create and verify a MAC) and

thus are different from digital signatures where the signing and verification keys differ. Practical MAC algorithms can be constructed from cryptographic hash functions (for example, HMAC) or from block ciphers (for example, CBC-MAC and others).

Cross-references

- ▶ Authentication
- ▶ Digital Signatures
- ▶ Hash Functions
- ▶ Symmetric Encryption

Recommended Reading

1. Krawczyk H., Bellare M., and Canetti R. HMAC: Keyed-hashing for message authentication, RFC 2104. Internet Engineering Task Force (IETF), 1997.
2. Stallings W. Cryptography and Network Security: Principles and Practices (4th edn.). Pearson-Prentice Hall, Upper Saddle River, NJ, 2006.

Message Integrity Codes

- ▶ Message Authentication Codes (MAC)

Message Queuing Systems

Sara Bouchenak[1], Noël de Palma[2]
[1]University of Grenoble I — INRIA, Grenoble, France
[2]INPG — INRIA, Grenoble, France

Synonyms

Message-oriented middleware (MOM); Message-oriented systems; Messaging systems; Queuing systems

Definition

A message is an information sent by a sender process to a receiver process. A message queue is a mechanism that allows a sender process and a receiver process to exchange messages. The sender posts a message in the queue, and the receiver retrieves the message from the queue. A message queuing system provides a means to build distributed systems, where distributed processes communicate through messages exchanged via queues.

Key Points

A message queuing system provides several facilities, such as creating messages, creating queues, initializing sender and receiver processes, and providing a means to send and receive messages.

First of all, a message queuing system provides a facility to build a message and fill it with data. Properties may be associated with a message, such as the message size, the message expiration time and the message priority.

A message queuing system also provides facilities to create a queue and, optionally, to associate parameters with a queue, such as the queue length (i.e., the maximum number of messages a queue may hold), the queue topics (i.e., the types of messages the queues may contain), etc.

The senders and receivers of messages may communicate in a synchronous way or in an asynchronous way. With a synchronous communication protocol, a receiver waits for a message from a sender, i.e., it blocks until the message arrives. Whereas with an asynchronous communication protocol, the receiver continues executing and is notified of the reception of a message when this one arrives.

Furthermore, the destination of a message may be specified either explicitly or implicitly. In the explicit mode, the sender specifies the queue to which the message is sent. While in the implicit mode, the sender specifies a topic to which a message is sent, and the message queuing system is responsible of automatically finding the queues that correspond to that topic before sending the message to these queues.

Several message queuing systems are proposed, some are proprietary and others are open source. Oracle proposes Advanced Queuing for Oracle databases [3], Skype has Skytools PgQ for PostgreSQL databases [5], IBM provides WebSphere MQ, Microsoft has MSMQ [1], and Sun Microsystems defines Java Message Service (JMS) as a specification of a Java standard for message queuing systems [6]. Open source message queuing systems include ActiveMQ [7], JBoss Messaging [2], and JORAM [4].

Cross-reference

- ▶ Adaptive Middleware for Message Queuing Systems

Recommended Reading

1. IBM. WebSphere MQ, 2008. http://www-306.ibm.com/software/integration/wmq/.
2. JBoss. JBoss Messaging, 2008. http://labs.jboss.com/jbossmessaging/.
3. Oracle. Oracle9i Application Developer's Guide – Advanced Queuing, 2008. http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96587/toc.htm.
4. ScalAgent. JORAM: Java Open Reliable Asynchronous Messaging, 2008. http://joram.objectweb.org/.
5. Skype. SkyTools PgQ, 2008. https://developer.skype.com/SkypeGarage/DbProjects/SkyTools.
6. Sun Microsystems. Java Message Service (JMS), 2008. http://java.sun.com/products/jms/.
7. The Apache Software Foundation. Apache ActiveMQ, 2008. http://activemq.apache.org/.

Message-Oriented Middleware (MOM)

► Message Queuing Systems
► Publish/Subscribe Over Streams

Message-oriented Systems

► Message Queuing Systems

Messaging Engines

► Interface Engines in Healthcare

Messaging Systems

► Message Queuing Systems

Meta Data Base

► Meta Data Repository

Metadata Interchange Specification

Wei Tang
Teradata Corporation, El Segundo, CA, USA

Synonyms

MDIS

Definition

Metadata Interchange Specification (MDIS) is a standard proposed by the Metadata Coalition (MDC) for defining metadata.

Key Points

Metadata Coalition (MDC) was an organization of database and data warehouse venders founded in October 1995. Its aim was to define a tactical set of standard specifications for the access and interchange of meta-data between software tools.

In July 1996, Metadata Interchange Specification (MDIS) 1.0 was officially ratified by MDC.

The MDIS Version 1.0 specification represents Coalition member input and recommendations collected and synthesized by the Coalition's technical subcommittee which included representatives from Business Objects, ETI, IBM, Platinum Technology, Price Waterhouse, Prism Solutions, R&O and SAS Institute. The latest version of MDIS is 1.1, which was published in August 1997.

The Metadata Interchange Specification draws a distinction between:

- The Application Metamodel – the tables, etc., used to "hold" the metadata for schemas, etc., for a particular application; for example, the set of tables used to store metadata in Composer may differ significantly from those used by the Bachman Data Analyst.
- The Metadata Metamodel – the set of objects that the MDIS can be used to describe. These represent the information that is common (i.e., represented) by one or more classes of tools, such as data discovery tools, data extraction tools, replication tools, user query tools, database servers, etc. The metadata metamodel should be:
  - Independent of any application metamodel.
  - Character-based so as to be hardware/platform-independent.

– Fully qualified so that the definition of each object is uniquely identified.

There are two basic aspects of the specification:

1. Those that pertain to the semantics and syntax used to represent the metadata to be exchanged. These items are those that are typically found in a specifications document.
2. Those that pertain to some framework in which the specification will be used. This second set of items is two file-based semaphores that are used by the specification's import and export functions to help the user of the specification control consistency.

MDIS consists of a metamodel, which defines the syntax and semantics of the metadata to be exchanged, as well as the specification of a framework for supporting an actual MDIS implementation. The MDIS Metamodel is a hierarchically structured, semantic database model that's defined by a tag language. The metamodel consists of a number of generic, semantic constructs, such as Element, Record, View, Dimension, Level, and Subschema, plus a Relationship entity that can be used in the specification of associations between arbitrary source and target constructs. The MDIS metamodel may be extended through the use of named properties that are understood to be tool-specific and not defined within MDIS. Interchange is accomplished via an ASCII file representation of an instance of this metamodel. Although support for an API is mentioned in the specification, no API definition is provided.

The MDIS Access Framework specifies several fairly general mechanisms that support the interchange of metamodel instances. The Tool and Configuration Profiles define semaphores that ensure consistent, bidirectional metadata exchange between tools. The MDIS Profile defines a number of system parameters (environment variables) that would be necessary in the definition of an MDIS deployment. Finally, Import and Export functions are exposed by the framework as the primary file interchange mechanisms for use by tools.

MDIS 1.1 was planned to be incorporated with Microsoft's Open Information Model (OIM) when Microsoft joined the MDC in December 1998. MDC decided later that MDIS be superseded by OIM. In 2000, the Metadata Coalition merged with the Object Management Group (OMG). OMG has worked on integrating OIM into its Common Warehouse Model (CWM) in order to provide a single standard for modeling meta-data in data warehouses. MDC, MDIS, and OIM are no longer in existence today (as independent entities).

Cross-references

- ▶ Common Warehouse Metamodel (CWM)
- ▶ Metadata Coalition (MDC)
- ▶ Open Information Model (OIM)

Recommended Reading

1. Metadata Interchange Specification (MDIS) Version 1.1. Available at: http://www.eda.org/rassp/documents/atl/MDIS-11.pdf

Meta Data Management System

- ▶ Meta Data Repository

Meta Data Manager

- ▶ Meta Data Repository

Meta Data Registry

- ▶ Meta Data Repository

Meta Data Repository

Christoph Quix
RWTH Aachen University, Aachen, Germany

Synonyms

Meta data base; Meta data manager; Meta data management system; Meta data registry

Definition

A meta data repository (MDR) is a component which manages meta data. In the context of database systems, one example of meta data is information about the database schema, i.e., a description of the data. In addition, MDRs can manage information about the processes which create, use, or update the data, the hardware components that host these processes or

the database system, or other (human) resources which make use of the data [6]. As meta data is also data, meta data repositories offer the same functionality for meta data as database management systems (DBMS) for data, e.g., queries, updates, transactions, access control.

Moreover, as meta data is semantically rich data, MDRs often employ object-oriented data models as the basis for the definition of meta data. MDRs should also offer predefined meta models for different types of meta data, so that the user is able to store meta data directly, without defining a meta model in advance.

Another task for a meta data repository is the integration of meta data from various sources into a comprehensive meta data model.

Historical Background

The first components that managed meta data in the context of database systems were dictionary systems which were integrated into the database management systems (DBMS) [12]. These dictionaries were already part of early DBMS products, such as IDMS or IBM IMS. For example, the integrated data dictionary (IDD) of IDMS was a separate database inside the IDMS which was used to maintain meta data of products in the IDMS family [11]. It could be extended also to maintain other types of meta data.

The relational database systems developed in the 1980s also had integrated dictionary systems (also known as system catalogs) to maintain definitions about tables, views, columns, etc. These dictionaries were mainly used by the DBMS itself, but they could be also queried by users and other applications to retrieve information about the contents of a database.

In the 1980s, ANSI started to develop a standard for Information Resource Dictionary Systems (IRDS) which was later adopted by ISO [6]. The standard defined the content, structure, and functionality of an IRDS. The main requirements stated by the IRDS standard are the availability of data modeling facilities, extensibility (i.e., the possibility to add new data types), and the provision of standard DBMS functionality such as query and reporting facilities, integrity and constraint management, and access control.

With the growing need for integrated information systems, stand-alone meta data repository systems became more popular in the 1990s. In contrast to the integrated repositories in DBMS products, a stand-alone MDR was able to manage meta data from different systems. The requirement for meta data integration was especially important for data warehouses, where data that was managed by independent, heterogeneous systems should be integrated into a common data store with a uniformed data model. The availability of meta data of the data sources was a prerequisite for data integration. In this context, some companies tried to build enterprise wide meta data repositories which were supposed to manage all meta data that is available in the enterprise. Such an ambitious goal was hard to achieve, and often, the return-of-investment of such a system was not as high as expected [5]. Therefore, the MDR market was significantly reduced at the end of the 1990s.

Since 2000, two trends for meta data repositories gained importance: community-focused repositories and federated repositories [5]. Communicty-focused repositories are employed in communities (within a company), which share a common interest, such as data warehousing or enterprise application integration. In these communities, the main problem of interoperability of meta data tools could be solved by dedicated bridging technologies, because of the limited scope of the meta data. With the rising importance of service oriented architecture (SOA), MDR needed again to address a broader scope of meta data. Therefore, federated solutions for MDRs are considered to be a solution for the meta data integration problem. In a federated MDR, there are still several MDRs for different communities but federated queries across several MDRs are enabled [5].

Foundations

Requirements

Requirements for MDRs were stated in the IRDS standard [6], in [2], and in [1]:

1. *Dynamic extensibility.* The MDR should provide easy functionalities for the extension of the built-in data models.
2. *Management of objects and relationships.* Objects and relationships between objects should be managed by the MDR.
3. *Notification.* An operation on a specific object in the MDR might trigger other operations on the same or different objects. Therefore, the MDR must be able to notify applications which are interested in certain events. In addition, the invocation

of methods inside the MDR (based on other events) should be also possible.
4. *Version management*. Versioning of meta data is required to track the evolution of a meta data object. It is also important to know which versions of two objects were active at a specific time. It might be also necessary to maintain relationships between older and newer versions of an object.
5. *Configuration management*. A configuration is a set of meta data objects which belong together in respect of content, e.g., they all describe the state of one component. The MDR should be able to manage a configuration as one group. Configurations can be also versioned.
6. *Integrity constraints*. The MDR must provide a language for the definition of integrity constraints on meta data, and enforce the compliance of the meta data with these constraints.
7. *Query and reporting functionality*. To retrieve meta data from the repository, the MDR needs to offer a query language. In addition, user-configurable reports should be also supported.
8. *User access*. If the MDR can be accessed directly by end-users or administrators, the MDR needs to support: a browsing facility for meta data, so that users can navigate through the metadata; an access control, so that users see or update only meta data which they are allowed to; a sophisticated user interface if the users are also allowed to update the meta data, so that the integrity of the MDR is maintained.
9. *Interoperability*. To enable interoperability with other tools and repositories, the MDR should support standards for meta data exchange (such as XMI) and offer an API (application program interface).

Architecture

There are several MDRs already available (see "Systems" section below), each having its own unique architecture. However, by abstracting from these concrete architectures, several components which are common for all MDRs can be identified:

1. *Repository*. The repository component is the internal data store of the MDR and therefore the core of the MDR. As MDRs have to provide similar functionality for meta data as DBMS for data, the repository is often implemented on top of a DBMS.
2. *Meta data manager*. The meta data manager acts as the controller of the repository. As all accesses to the repository should go through the meta data manager, it provides an interface for external applications. Using this interface, applications can store, update, and query meta data.
3. *Models*. A MDR needs to come with already predefined meta models (or information models) which can be directly employed by the users of the MDR to store meta data. If the user has to define its own meta models, the effort to get the MDR running might be too high for the application. Nevertheless, it should be possible to extend the existing models for the specific requirements of the applications that use the MDR.
4. *User interface*. As described above, a MDR can be also accessed by users, which are either end-users using the data or processes described in the MDR, or administrators controlling the system of which the MDR is a part. The user interface can consist of a query facility, a meta data browser, an administrator interface, and an interface to update the meta data.

As mentioned above, a current trend for MDRs is the idea of a federated MDR. This changes the standard architecture described before: the repository component in a federated MDR is not one single data store, the meta data can be distributed across several independent and heterogeneous components. In a federated MDR architecture, the meta data could be stored in files, databases, or managed by specific applications. This increases the complexity of the meta data manager significantly, as meta data queries have to be transformed into queries of the individual systems holding the meta data.

Systems

There are several MDRs available in the market. They can be classified as stand-alone MDRs, repositories integrated into larger software platforms, open source systems and research prototypes.

The market for stand-alone MDRs is changing frequently as companies specialized on MDRs are being acquired by other companies. The current products for separate meta data management solutions are, for example, *ASG Rochade*, *Adaptive Metadata Manager*, and *Advantage Repository*. These products came mainly from the data management area

(especially used as MDRs in data warehouse systems), but are now also addressing other areas such as enterprise application integration and service-oriented architectures. Other systems, such as *Logidex* from *LogicLibrary* or *BEA AquaLogic Registry Repository*, have been originally developed as meta data systems for service-oriented architectures.

As mentioned above, large software companies are also addressing the meta data challenges in their software or technology platforms. For example, *IBM* has an integrated MDR in their information integration framework.

There are also a few open source systems which can be used as MDR. Two examples are *Repository in a Box* and *XMDR*. In the research community, *ConceptBase* [9] is a MDR which has been used in several research projects. ConceptBase provides a very flexible data model which can be used for any kind of meta data structure.

Key Applications

There are various application areas for MDRs, basically in all areas in which the management of meta data is necessary. The most important applications for MDRs are situations in which meta data from different sources has to be integrated in one repository. This goes usually beyond the capabilities of builtin MDRs, i.e., repositories which are integrated with other software components.

Data integration in general is an application area in which MDRs play a central role. If data has to be integrated from heterogeneous systems, the description of this data is required to enable the integration. Data warehouse systems [8] are an example for an architecture of integrated data management in which the role of MDRs has been defined explicitely.

In the context of data warehouse systems, also the problem of data quality has been discussed [10,7]. Meta data is often the basis for data quality measurements, e.g., meta data describes the provenance, the age, the semantics of data. Therefore, MDRs are important components for data quality projects.

A MDR can also be used as a resource for structured documentation about IT systems. In addition to the "usual" meta data artefacts such as models and mappings for data integration, also a documentation of the employed systems and their architecture in an organization is useful.

As discussed before, service-oriented architecture (SOA) are becoming an important concept for software development. As a system based on a SOA is a distributed and often heterogeneous system, the management of meta data in a SOA is also important. Therefore, a MDR is often also a component in a SOA.

Another application area for MDRs might the management of meta data on the web, such as RDF or OWL ontologies. However, the web is build on the idea of decentralized data management which is in conflict with the concept of a central, integrated repository for all kind of meta data. Nevertheless, MDRs can be useful to manage the meta data at a specific site, e.g., the ontologies which are offered by that site and their mappings to other ontologies.

Future Directions

The integration of meta data will remain to be a challenge, however the meta data will be integrated, either materialized in a repository, federated with a virtual integration system, or some combination of these. With the rising importance of web applications, service-oriented architectures and similar concepts, it can be expected that more loosely coupled MDRs with a federated integration become more successful. Existing or upcoming meta data standards, such as such as CWM (common warehouse metamodel) and XMI (XML metadata interchange), might simplify the task, but the integration of meta data will remain a problem. Another trend is the integration of MDR into larger software platforms as it is done (or planned) by the major software vendors.

Meta data integration and new architectures for MDRs are also interesting questions for the research community: How can such systems be built, that enable the integrated querying of various meta data sources? Which lessons can be applied for meta data that have already been learned at the data level? Other research questions for MDRs and meta data management are addressed in model management [3,4] which investigates formal methods for working with data models. The challenge for MDRs here is to provide generic structures for the representation of models and mappings.

Cross-references

- ▶ Data Warehouse Metadata
- ▶ Meta Data Registry
- ▶ Meta Model
- ▶ Meta Object Facility

Recommended Reading

1. Bauer A. and Günzel H. (eds.) Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung. dpunkt-Verlag, Heidelberg, 2001.
2. Bernstein P.A. Repositories and Object Oriented Databases. ACM SIGMOD Rec., 27(1):88–96, 1998.
3. Bernstein P.A., Halevy A.Y., and Pottinger R. A Vision for Management of Complex Models. ACM SIGMOD Rec., 29(4):55–63, 2000.
4. Bernstein P.A. and Melnik S. Model Management 2.0: Manipulating Richer Mappings. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 1-12.
5. Blechar M., IT Metadata Repository Magic Quadrant Update 2002. Gartner, Inc., 2002.
6. ISO/IEC Information technology – Information Resource Dictionary System (IRDS) Framework. International Standard ISO/IEC 10027:1990, DIN Deutsches Institut für Normung, e.V., 1990.
7. Jarke M., Lenzerini M., Vassiliou Y., and Vassiliadis P. (eds.) Fundamentals of Data Warehouses. Springer-Verlag, 2000.
8. Jarke M. and Vassiliou Y. Foundations of Data Warehouse Quality - a Review of the DWQ Project. In Proc. 2nd Int. Conf. Information Quality, 1997, pp. 299–313.
9. Jeusfeld M.A., Jarke M., Nissen H.W., and Staudt M. ConceptBase – Managing Conceptual Models about Information Systems. In Handbook on Architectures of Information Systems, P. Bernus, K. Mertins, and G. Schmidt (eds.). Springer-Verlag, 1998, pp. 265–285.
10. Tayi G.K. and Ballou D.P. Examining Data Quality. Commun. ACM, 41(2):54–57, 1998.
11. Wikipedia – The Free Encyclopedia IDMS (Integrated Database Management System). Article in the encyclopedia, 2008, URL http://en.wikipedia.org/wiki/IDMS.
12. Wikipedia – The Free Encyclopedia. Metadata. Article in the encyclopedia, 2008, URL http://en.wikipedia.org/wiki/Metadata.

Meta Model

► Metamodel

Meta Object Facility

Wei Tang
Teradata Corporation, El Segundo, CA, USA

Synonyms

MOF

Definition

The Meta Object Facility (MOF) is an OMG metamodeling and metadata repository standard. It is an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner. MOF-based standards are in use for integrating tools, applications and data [1].

Key Points

MOF was developed as a response to a request for proposal (RFP), issued by the OMG Analysis and Design Task Force, for Metadata repository facility (http://www.omg.org/cgi-bin/doc?cf/96-05-02). The purpose of the facility was to support the creation, manipulation, and interchange of meta models.

MOF provides a metadata management framework, and a set of metadata services to enable the development and interoperability of model and metadata driven systems. The MOF metadata framework is typically depicted as a four-layer architecture as shown in Table 1:

The MOF specification has three core parts:

1. The specification of the MOF Model
   a. The MOF's built-in meta-metamodel, the "abstract language" for defining MOF metamodels
2. The MOF IDL Mapping
   a. A standard set of templates that map an MOF metamodel onto a corresponding set of CORBA IDL interfaces
3. The MOF's interfaces
   a. The set of IDL interfaces for the CORBA objects that represent an MOF metamodel

The OMG adopted the MOF version 1.0 in November 1997. The most recent revision of MOF, 2.0, was adopted in January 2006 and based on the following OMG specifications:

- MOF 1.4 Specification – MOF 2.0 is a major revision of the MOF 1.4 Specification. MOF 2.0 addresses issues deferred to MOF 2.0 by the MOF 1.4 RTF.

Meta Object Facility. Table 1. OMG's Metadata Architecture

Meta-level	MOF terms	Examples
M3	Meta-metamodel	The "MOF Model"
M2	Metamodel, meta-metadata	UML Metamodel, CWM Metamodel
M1	Model, metadata	UML models, CWM metadata
M0	Object, data	Modeled systems, Warehouse data

- UML 2.0 Infrastructure Convenience Document: ptc/04-10-14 – MOF 2.0 reuses a subset of the UML 2.0 Infrastructure Library packages.
- MOF 2.0 XMI Convenience document: ptc/04-06-11 – Defines the XML mapping requirements for MOF 2.0 and UML 2.0.

The MOF 2 Model is made up of two main packages, Essential MOF (EMOF) and Complete MOF (CMOF).

1. The EMOF Model merges the Basic package from UML2 and merges the Reflection, Identifiers, and Extension capability packages to provide services for discovering, manipulating, identifying, and extending metadata.
2. The CMOF Model is the metamodel used to specify other metamodels such as UML2. It is built from EMOF and the Core:Constructs of UML 2. The Model package does not define any classes of its own. Rather, it merges packages with its extensions that together define basic metamodeling capabilities.

Examples of metadata driven systems that use MOF include modeling and development tools, data warehouse systems, metadata repositories etc. A number of technologies standardized by OMG, including UML, MOF, CWM, SPEM, XMI, and various UML profiles, use MOF and MOF derived technologies (specifically XMI and more recently JMI which are mappings of MOF to XML and Java respectively) for metadata-driven interchange and metadata manipulation. MOF mappings from MOF to W3C XML and XSD are specified in the XMI (ISO/IEC 19503) specification. Mappings from MOF to Java are in the JMI (Java Metadata Interchange) specification defined by the Java Community Process.

Note that MOF 2.0 is closely related to UML 2.0. MOF 2.0 specification integrates and reuses the complementary UML 2.0 Infrastructure submission to provide a more consistent modeling and metadata framework for OMG's Model Driven Architecture. UML 2.0 provides the modeling framework and notation, MOF 2.0 provides the metadata management framework and metadata services.

MOF was also incorporated into an ISO/IEC (the International Organization for Standardization/the International Electrotechnical Commission) standard (19502:2005) in November 2005. The standard defines a metamodel (defined using the MOF), a set of interfaces (defined using ODP IDL – ITU-T Recommendation X.920 (1997) | ISO/IEC 14750:1999), that can be used to define and manipulate a set of interoperable metamodels and their corresponding models (including the Unified Modeling Language metamodel – ISO/IEC 19501:2005, the MOF meta-metamodel, as well as future standard technologies that will be specified using metamodels). It also defines the mapping from MOF to ODP IDL (ITU rec X920|ISO 14750).

In conclusion, the MOF provides the infrastructure for implementing design and reuse repositories, application development tool frameworks, etc. The MOF specifies precise mapping rules that enable the CORBA interfaces for metamodels to be generated automatically, thus encouraging consistency in manipulating metadata in all phases of the distributed application development cycle.

Cross-references

- ▶ Meta Object Facility
- ▶ Metadata
- ▶ Metamodel
- ▶ Model-Driven Architecture
- ▶ Unified Modeling Language
- ▶ XMI

Recommended Reading

1. Common warehouse metamodel (CWM). Available at http://www.omg.org/technology/documents/formal/cwm.htm (accessed on September 22, 2008).
2. ISO/IEC standard 19502:2005 (Information Technology – Meta Object Facility). Available at http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32621
3. MOF Query/Views/Transformations. Available at http://www.omg.org/spec/QVT/ (current version 1.0)
4. MOF 2.0 versioning and development lifecycle. Available at http://www.omg.org/technology/documents/formal/MOF_version.htm
5. OMG's meta object facility. Available at http://www.omg.org/mof/ (current version 2.0)

Metadata

Manfred A. Jeusfeld
Tilburg University, Tilburg, The Netherlands

Definition

Metadata is data linked to some data item, i.e., metadata is data about data. The metadata of a data item

specifies how the data item was created, in which context it can be used, how its was transformed, or how it can be interpreted or processed. The earliest use of metadata are bibliographic records about books, such as the author of the book. In principle, any type of data item can have metadata attached to it. The type of the data item itself can be metadata and determines which other metadata fields may be attached to the data item.

Key Points

The purpose of metadata is to provide contextual information for a data item. It may be used hy humans to determine the usability of a data item. Likewise, computer programs can read the metadata in order to guide the processing of a data item. Metadata can be included in the data item (e.g., the date and location of a photography) or it may be stored apart of the data item. In the latter case, the data item requires being identifiable. In databases, metadata fields can be represented next to data fields, virtually blurring the distinction between metadata and data.

Metadata is mostly used in domains where the structure of the data item is rather complex. Metadata typically has a simple structure such as name/value pairs. Applications domains are word processing, multi-media processing, system design, data warehouses, data quality management, and others. The common characteristic of these domains is the presence of many data items of the same type, which need to be managed according to some criteria. Metadata allows providing the necessary information to check these criteria. In the semantic web, metadata can be represented in RDF and related formalisms such as the Dublin Core.

There is no limitation on the size of metadata attached to data items. It can be that the size of metadata exceeds the size of the data item itself. For example, the complete change history of a document is metadata of the document.

The schema of a database can be interpreted as metadata about the database. It specifies the type of the data items stored in the database. Likewise, a metamodel can be interpreted as metadata about schemas (or models).

Cross-references

- ▶ Database Schema
- ▶ Dublin Core
- ▶ Metamodel
- ▶ RDF

Recommended Reading

1. Duval E., Hodgins W., Sutton S.A., and Weibel S. Metadata principles and practicalities. D-Lib Magazine, 8(4), April 2002.

Metadata Encoding and Transmission Standard

- ▶ LOC Mets

Metadata Registry, ISO/IEC 11179

Raymond K. Pon[1], David J. Buttler[2]
[1]University of California, Los Angeles, Los Angeles, CA, USA
[2]Lawrence Livermore National Laboratory, Livermore, CA, USA

Synonyms

Metadata repository; MDR

Definition

ISO/IEC-11179 [10] is an international standard that documents the standardization and registration of metadata to make data understandable and shareable. This standardization and registration allows for easier locating, retrieving, and transmitting data from disparate databases. The standard defines the how metadata are conceptually modeled and how they are shared among parties, but does not define how data is physically represented as bits and bytes. The standard consists of six parts. Part 1 [5] provides a high-level overview of the standard and defines the basic element of a metadata registry – a data element. Part 2 [7] defines the procedures for registering classification schemes and classifying administered items in a metadata registry (MDR). Part 3 [4] specifies the structure of an MDR. Part 4 [6] specifies requirements and recommendations for constructing definitions for data and metadata. Part 5 [8] defines how administered items are named and identified. Part 6 [9] defines

how administered items are registered and assigned an identifier.

Historical Background

The first edition of the standard was published by the Technical Committee ISO/IEC JTC1, Information Technology Subcommittee 32, Data Management and Interchange, starting in 1994 and completed in 2000. The second edition was started in 2004 and was completed in 2005. The second edition cancels and replaces the first edition of the standard.

Foundations

Metadata is data that describes other data. A metadata registry is a database of metadata. The database allows for the registration of metadata, which enables the identification, provenance tracking, and quality monitoring of metadata. Identification is accomplished by assigning a unique identifier to each object registered in the registry. Provenance details the source of the metadata and the object described. Monitoring quality ensures that the metadata accomplishes its designed task. An MDR also manages the semantics of data, so that data can be re-used and interchanged. An MDR is organized so that application designers can determine whether a suitable object described in the MDR already exists so that it may be reused instead of developing a new object.

Part 1: Framework

Part 1 introduces the building blocks of the MDR standard: data elements, value domains, data element concepts, conceptual domains, and classification schemes. An MDR is organized as a collection of concepts, which are mental constructs created by a unique combination of characteristics. A concept system is a set of concepts with relations among them. One such concept system that classifies objects is a classification scheme. A classification scheme is organized with some specified structure and is designed for assigning objects to concepts defined within it.

The basic construct in a metadata registry is the data element. A data element consists of a data element concept and a representation. A data element concept (DEC) is a concept that can be represented as a data element described independently of any particular representation. The representation of a data element consists of a value domain, a data-type, units of measure, and a representation class. A data element concept may consist of an object class, which is a set of abstractions in the real world that can be identified with explicit boundaries, and a property, which is a characteristic common to all members of an object class. A value domain is a set of permissible values. Each value domain is a member of the extension of a concept known as the conceptual domain. A conceptual domain is a set of value meanings, which are the associated meanings to values.

An MDR contains metadata describing data constructs. Registering a metadata item makes it a registry item. If the registry item is subject to administration, it is called an administered item. An ISO/IEC 11179 MDR consists of two levels: the conceptual level and the representational level. The conceptual level contains the classes for the data element concept and conceptual domain. The representational level contains classes for data element and value domain.

Part 2: Classification

Part 2 provides a conceptual model for managing concept systems used as classification schemes. Associating an object with a concept from a classification scheme provides additional understanding of the object, comparative information across similar objects, an understanding of an object within the context of a subject matter field, and the ability to identify differences of meaning between similar objects.

Classification schemes are registered in an MDR by recording their attributes, such as those regarding its designation, definition, classification scheme, administration record, reference document, submission, stewardship, registration authority, and registrar.

Part 2 also defines the mechanism for classifying an administered item, which is the assignment of a concept to an object. Objects can also be linked together by relationships linking concepts in the concept system.

Part 3: Registry Metamodel and Basic Attributes

Part 3 describes the basic attributes that are required to describe metadata items and the structure for a metadata registry. The standard uses a metamodel to describe the structure of an MDR. A metamodel is a model that describes other models. The registry metamodel is specified as a conceptual data model, which describes how relevant information is structured in the

real world, and is expressed in the Unified Modeling Language [13].

The registry model is divided into six regions:

- **The administration and identification region:** supports the administrative aspects of administered items in the MDR. This region manages the identification and registration of items submitted to the registry, organizations that have submitted and/or are responsible for items in the registry, supporting documentation, and relationships among administered items. An administered item can be a classification scheme, a conceptual domain, context for an administered item, a data element, a data element concept, an object class, a property, a representation class, and a value domain. An administered item is associated with an administration record, which records administrative information about the administered item in the registry.
- **The naming and definition region:** manages the names and definitions of administered items and the contexts for names. Each administered item is named and defined within one or more contexts. A context defines the scope within which the data has meaning, such as a business domain, a subject area, an information system, a data model, or standards document.
- **The classification region:** manages the registration and administration of classification schemes and their constituent classification scheme items. It is also used to classify administered items.
- **The data element concepts region:** maintains information on the concepts upon which the data elements are developed, primarily focusing on semantics.
- **The conceptual and value domain region:** administrates the conceptual domains and value domains.
- **The data element region:** administrates data elements, which provide the formal representations for some information (e.g., a fact, observation, etc.) about an object. Data elements are reusable and shareable representations of data element concepts.

Part 4: Formulation of Data Definitions

Part 4 specifies the requirements and recommendations for constructing data and metadata definitions. A data definition must be stated in the singular. It also must be a descriptive phrase, containing only commonly understood abbreviations, that state what the concept is (as opposed to what the concept is not). A data definition must also be expressed without embedding definitions of other data. The standard also recommends that a data definition should be concise, precise, and unambiguous when stating the essential meaning of the concept. Additionally, a data definition should be self-contained and be expressed without embedding rationale, functional usage, or procedural information, circular reasoning. Terminology and consistent logical structure for related definitions should also be used.

Part 5: Naming and Identification Principles

Part 5 defines the naming and identification of the data element concept, the conceptual domain, data element, and value domain. Each administered item has a unique data identifier within the register of a Registration Authority (RA), which is the organization responsible for an MDR. The international registration data identifier (IRDI) uniquely identifies an administered item globally and consists of a registration authority identifier (RAI), data identifier (DI), and version identifier (VI).

Each administered item has at least one name within a registry of an RA. Each name for an administered item is specified within a context. A naming convention can be used for formulating names. A naming convention may address the scope of the naming convention and the authority that establishes the name. A naming convention may additionally address semantic, syntactic, lexical, and uniqueness rules. Semantic rules govern the existence of the source and content of the terms in a name. Syntactic rules govern the required term order. Lexical rules govern term lists, name length, character set, and language. Uniqueness rules determine whether or not names must be unique.

Part 6: Registration

Part 6 specifies how administered items are registered and assigned an IRDI. Metadata in the MDR is also associated with a registration status, which is a designation of the level of registration or quality of the administered item. There are two types of status categories: lifecycle and documentation. The lifecycle registration status categories address the development and progression of the metadata and the preferences of usage of the administered item. The documentation registration status categories are used when there is no

further development in the quality of metadata or use of the administered item.

Each RA establishes its own procedures for the necessary activities of its MDR. Some activities include the submission, progression, harmonization, modification, retirement, and administration of administered items.

Key Applications

The standardization that ISO/IEC 11179 provides enables for the easy sharing of data. For example, many organizations exchange data between computer systems using data integration technologies. In data warehousing schemes, completed transactions must be regularly transferred to separate data warehouses. Exchanges of data can be accomplished more easily if data is defined precisely so that automatic methods can be employed. By having a repository of metadata that describes data, application designers can reuse and share data between computer systems, making the sharing of data easier. ISO/IEC 11179 also simplifies data manipulation by software by enabling the manipulation of data based on characteristics described by the metadata in the registry. This also allows for the development of a data representation model for CASE tools and repositories [3].

There are several organizations that have developed MDRs that comply with ISO/IEC 11179, such as the Australian Institute of Health and Welfare [1], the US Department of Justice [14], the US Environmental Protection Agency [15], the Minnesota Department of Education [11], and the Minnesota Department of Revenue [12]. Currently, there is also an MDR available developed by Data Foundations [2].

Cross-references

► Metadata

Recommended Reading

1. Australian Institute of Health and Welfare. Metadata Online Registry (METeOR). http://meteor.aihw.gov.au/, 2007.
2. Data Foundations. Metadata Registry. http://www.datafoundations.com/solutions/data_registries.shtml, 2007.
3. ISO/IEC JTC1 SC32. Part 1: Framework for the specification and standardization of data elements. Information Technology – Metadata registries (MDR), 1st edn., 1999.
4. ISO/IEC JTC1 SC32. Part 3: Registry metamodel and basic attributes. Information Technology – Metadata registries (MDR), 2nd edn., 2003.
5. ISO/IEC JTC1 SC32. Part 1: Framework. Information Technology – Metadata registries (MDR), 2nd edn., 2004.
6. ISO/IEC JTC1 SC32. Part 4: Formulation of data definitions. Information Technology – Metadata registries (MDR), 2nd edn., 2004.
7. ISO/IEC JTC1 SC32. Part 2: Classification. Information Technology – Metadata registries (MDR), 2nd edn., 2005.
8. ISO/IEC JTC1 SC32. Part 5: Naming and identification principles. Information Technology – Metadata registries (MDR), 2nd edn., 2005.
9. ISO/IEC JTC1 SC32. Part 6: Registration. Information Technology – Metadata registries (MDR), 2nd edn., 2005.
10. ISO/IEC JTC1 SC32. ISO/IEC 11179, Information Technology – Metadata registries (MDR), 2007.
11. Minnesota Department of Education. Metadata Registry (K-12 Data). 2007. http://education.state.mn.us/mde-dd.
12. Minnesota Department of Revenue. Property Taxation (Real Estate Transactions). 2007. http://proptax.mdor.state.mn.us/mdr.
13. Object Management Group. Unified Modeling Language. 2007. http://www.uml.org/.
14. US Department of Justice. Global Justice XML Data Model (GJXDM). 2007. http://justicexml.gtri.gatech.edu/.
15. US Environmental Protection Agency. Environmental Health Registry. 2007. http://www.epa.gov/edr/.

Metadata Repository

► Data Dictionary
► Metadata Registry, ISO/IEC 11179

Meta-Knowledge

► Multimedia Metadata

Metamodel

Manfred A. Jeusfeld
Tilburg University, Tilburg, The Netherlands

Synonyms

Meta model

Definition

A metamodel is a model that consists of statements about models. Hence, a metamodel is also a model but its universe of discourse is a set of models, namely those models that are of interest to the creator of

the metamodel. In the context of information systems, a metamodel contains statements about the constructs used in models about information systems. The statements in a metamodel can define the constructs or can express true and desired properties of the constructs. Like models are abstractions of some reality, metamodels are abstractions of models. The continuation of the abstraction leads to meta metamodels, being models of metamodels containing statements about metamodels. Metamodeling is the activity of designing metamodels (and metametamodels). Metamodeling is applied to design new modeling languages and to extend existing modeling languages.

A second sense of the term metamodel is the specification of the generation of mathematical models, in particular sets of mathematical equations that describe some reality.

Historical Background

One of the earliest metamodels is the definition of the binary data model of Abrial [1]. Abrial distinguished three abstraction levels: the data level of a database, the schema of the database (model), and the category level (metamodel).

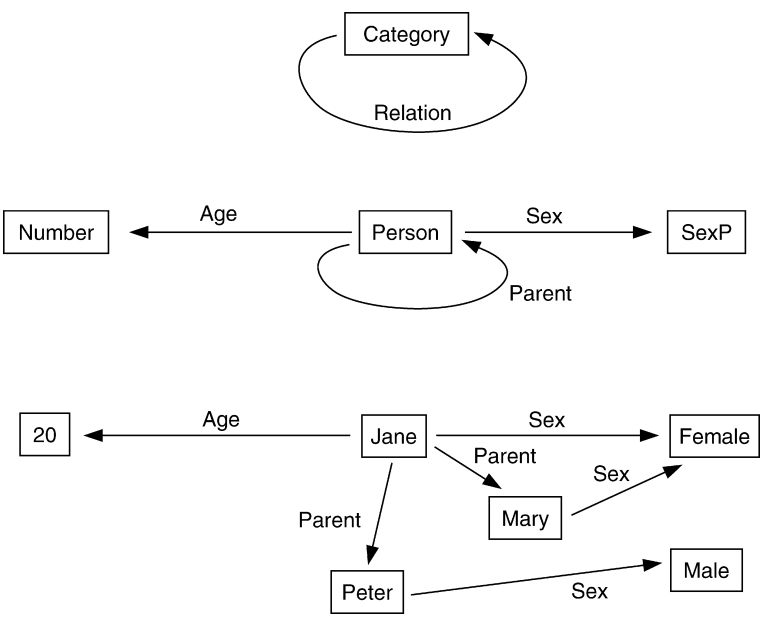
The metamodel defining the binary data model consists of the construct `Category` and the construct `relation`. Abrial interpreted the abstraction between the levels as classification. For example, `Jane` is classified to `Person` and `Person` is classified to `Category`. A similar classification holds for the relations.

In the 1980s, the use of metamodels became so widespread that an ISO standard, the Information Resource Dictionary Standard [2], was defined. It extended Abrial's view by a fourth level, i.e., by metametamodels. In the late 1990s, the Object Management Group (OMG) [5] consolidated and standardized the terminology of metamodels. They distinguished the levels M_0 (information), M_1 (model), M_2 (metamodel), and M_3 (metametamodel). The M_3 level is under control of OMG. It defined four basic constructs (classes, associations, data types, and packages). The M_2 level is used to define modeling languages such as UML, IDL, and so forth.

Besides the standardization efforts, there were several metamodeling languages developed from the 1990s onwards that mostly adopted the four-level approach. Examples is the Telos language and the GOPPR language of MetaEdit+ [9].

Foundations

Meta models in computer science and related domains are mainly used to facilitate conceptual modeling, to define constructs of the conceptual modeling languages, to specify constraints on the use of constructs, and to encode the similarities of different models (and metamodels). As conceptual modeling is about representing concepts, an element of a metamodel is



Metamodel. Figure 1. Abrial's definition of the binary data model.

also a concept say *meta concept*, being interpreted by all entities that are defined or constrained by the meta concept. For example, the meta concept `EntityType` is a construct of the Entity-Relationship Diagramming language. It is interpreted by all possible entity types of all possible entity relationship diagrams. Essentially, `EntityType` is the name of a set that has all possible entity types as elements. A problem with this set-view is that it immediately introduces sets of sets (=concepts in metamodels) and sets of sets of sets (concepts of metametamodels). To overcome this complexity, metamodels were originally only investigated as level-pairs: (metametamodel vs. metamodel), (metamodel vs. model), (model vs. data). First a metametamodel is developed. Then, a metamodel or several metamodels are expressed as instances of the metametamodel, then models are expressed in terms of the metamodels. The lowest level (M_0 in MOF) is typically not expressed in conceptual modeling since it is about data or actual activities of some application domain. The pair-wise approach allows to keep the set-oriented semantics or other forms of semantics specification relying on distinguishing a concept from its instances.

The set-oriented semantics is mirrored by a logical interpretation, in which concepts are represented by unary predicates and relations and attributes are represented by binary predicates. For example, `EntityType (Employee)` is the fact expressing that `Employee` (model level) is classified to `EntityType` (metamodel level). A fact `Employee(Jane)` would then express that `Jane` (data level) is an instance of `Employee`. If one restricts to just two consecutive levels, predicate symbols can be distinguished from constant symbols. In other words, the underlying logic is a first order logic. Scaling the semantics to more than two levels would move the logic to higher order.

One can avoid higher order semantics by introducing a binary predicate `In(x,c)` where `x` is some concept of some modeling level M_i and `c` is a concept of the next higher modeling level M_{i+1}. This framework allows to represent the facts `In(Employee,EntityType)` and `In(Jane,EntityType)` without leaving first order logic.

The specific choice of the underlying semantics for metamodels determines to which degree a metamodel can express the intended meaning of the concepts included in a metamodel. In UML, the semantics of the UML constructs are defined in a metamodel using OCL (object constraint language [6]). Current metamodeling tools dominantly use cardinality constraints as means to constrain the semantics of metamodel concepts. Constraints exceeding cardinalities have to be expressed in OCL or script languages.

The second sense of metamodels, the generation of mathematical equations to describe some reality, is for example used by Bailey and Basili [3] to develop a formal framework for understanding real world phenomena in the domain of software engineering.

Key Applications

Meta models became a popular technique at the end of the 1990s. The current specification of UML is supporting metamodeling in order to extend the capabilities of the language and to adapt it to specific modeling domains. Tools supporting metamodeling are among others MetaEdit+ [9], ConceptBase [4] and Aris [7]. The MetaEdit+ tool claims to accelerate system development by orders of magnitude since the concepts of a metamodel can be linked to parameterized program code.

Future Directions

An open problem of metamodels is their utility. If a metamodel is represented as a UML class diagram, then it does list the allowed constructs but it does not explain how to use it in a meaningful way, i.e., to represent models in terms of the metamodel. Conceptual modeling textbooks motivate constructs by examples and discuss scenarios in which certain constructs are usable. This pragmatic level is neglected by metamodels.

Meta models should be seen as part of the larger model-driven architecture framework. That framework (also defined by OMG) is based on the assumption that system development is essentially a series of model transformations. The design of system development methods is then the combination of metamodeling and the specification of suitable model transformations.

The relationship between metamodels (or metametamodel) with ontologies is not yet well understood. Ontologies rely on two levels of abstraction: the concepts defined in the ontology and the real world objects being the interpretations of the concepts. Apparently, an ontology makes no difference between a model level concept like `Employee` and a metamodel level concept like `EntityType`. See also [8] for a discussion.

Cross-references

► Telos

Recommended Reading

1. Abrial J.R. Data semantics. In Database Management. In Proc. IFIP Working Conf. on Database Management, 1974, pp. 1–60.
2. American National Standard Institute. American National Standard X3.138-1988, Information Resource Dictionary System (IRDS). American National Standard Institute, 1989.
3. Bailey J.W. and Basili V.R. A Meta-model for software development resource expenditures. In Proc. 5th Int. Conf. on Software Eng., 1981, pp. 107–116.
4. Jeusfeld M.A., Jarke M., Nissen H.W., and Staudt M. Managing conceptual models about information systems. In Handbook on Architectures of Information Systems, 2nd edn., P. Bernus, K. Mertins, G. Schmidt (eds.). Springer, Berlin Heidelberg New York, 2006, pp. 273–294.
5. Object Management Group. Meta Object Facility (MOF) Specification, Version 1.4. April 2002. Available at: http://www.omg.org/technology/documents/formal/mof.htm.
6. Object Management Group. Object Constraint Language, OMG Available Specification Version 2.0. May 2006. Available at: http://www.omg.org/cgi-bin/doc?formal/2006-05-01.
7. Scheer A.-W. and Schneider K. ARIS – Architecture of integrated information systems. In Handbook on Architectures of Information Systems, 2nd edn., P. Bernus, K. Mertins, G. Schmidt (eds.).Springer, Berlin Heidelberg New York, 2006, pp. 605–623.
8. Terrasse M.-N., Savonnet M., Leclercq E., Grison T., and Becker G. Do we need metamodels and ontologies for engineering platforms? In Proc. 2006 Int. Workshop on Global Integrated Model Management, 2006, pp. 21–28.
9. Tolvanen J.-P. MetaEdit+: integrated modeling and metamodeling environment for domain-specific languages. In Proc. 21st ACM SIGPLAN Conf. on Object-Oriented Programming Systems, Languages & Applications, 2006, pp. 690–691.

Metaphor

► Visual Metaphor

Metasearch Engines

Weiyi Meng
State University of New York at Binghamton, Binghamton, NY, USA

Synonyms

Federated search engine

Definition

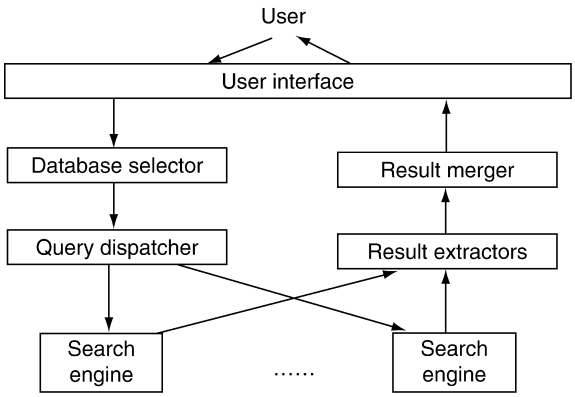
Metasearch is to utilize multiple other search systems (called *component search systems*) to perform simultaneous search. A metasearch engine is a search system that enables metasearch. To perform a basic metasearch, a user query is sent to multiple existing search engines by the metasearch engine; when the search results returned from the search engines are received by the metasearch engine, they are merged into a single ranked list and the merged list is presented to the user. Key issues include how to pass user queries to other search engines, how to identify correct search results from the result pages returned from search engines, and how to merge the results from different search sources. More sophisticated metasearch engines also perform *search engine selection* (also referred to as *database selection*), i.e., identify the search engines that are most appropriate for a query and send the query to only these search engines. To identify appropriate search engines to use for a query requires estimating the usefulness of each search engine with respect to the query based on some usefulness measure.

Historical Background

The earliest Web-based metasearch engine is probably the MetaCrawler system [12] that became operational in June 1995. (The MetaCrawler's website (www.metacrawler.com) says the system was first developed in 1994.) Motivations for metasearch include (i) increased search coverage because a metasearch engine effectively combines the coverage of all component search engines, (ii) improved convenience for users because a metasearch engine allows users to get information from multiple sources with one query submission and the metasearch engine hides the differences in query formats of different search engines from the users, and (iii) better retrieval effectiveness because the result merging component can naturally incorporate the voting mechanism, i.e., results that are highly ranked by multiple search engines are more likely to be relevant than those that are returned by only one of them. Over the last thirteen years, many metasearch engines have been developed and deployed on the Web. Most of them are built on top of a small number of popular general-purpose search engines but there are also metasearch engines that are connected to more specialized search engines (e.g., medical/health search engines) and some are connected to over one thousand search engines.

Even the earliest metasearch engines tackled the issues of search result extraction and result merging. Result merging is one of the most fundamental components in metasearch, and as a result, it has received a lot of attention in the metasearch and distributed information retrieval (DIR) communities and a wide range of solutions has been proposed to achieve effective result merging. Since different search engines may index a different set of web pages and some search engines are better than others for queries in different subject areas, it is important to identify the appropriate search engines for each user query. The importance of search engine selection was realized early in metasearch research and many approaches have been proposed to address this issue. A survey on some earlier result merging and search engine selection techniques can be found in [11].

Most metasearch engines are built on top of other search engines without explicit cooperation from these search engines. As a result, creating these metasearch engines requires a connection program and an extraction program (wrapper) for each component search engine. The former is needed to pass the query from the metasearch engine to the search engine and receive search results returned from the search engine, and the latter is used to extract the search result records from the result pages returned from the search engine. While the programs may not be difficult to produce by an experienced programmer, maintaining their validity can be a serious problem as they can become obsolete when the used search engines change their connection parameters and/or result display format. In addition, for applications that need to connect to hundreds or thousands of search engines, it can be very expensive and time-consuming to produce and maintain these programs. As a result, in recent years, automatic wrapper generation techniques have received much attention. Figure 1 shows a basic architecture of a typical metasearch engine.



Metasearch Engines. Figure 1. Metasearch engine component architecture.

Foundations

Result Merging

Result merging is to combine the search results returned from multiple search engines into a single ranked list. Early search engines often associated a numerical matching score (similarity score) to each retrieved search result and the result merging algorithms at that time were designed to "normalize" the scores returned from different search engines into values within a common range with the goal to make them more comparable. Normalized scores will then be used to re-rank all the search results. When matching scores are not available, the ranks of the search results from component search engines can be aggregated using voting-based techniques (e.g., Borda Count [1]). Score normalization and rank aggregation may also take into consideration the estimated usefulness of each selected search engine with respect to the query, which is obtained during the search engine selection step. For example, the normalized score of a result can be weighted by the usefulness score of the search engine that returned the result. This increases the chance for the results from more useful search engines to be ranked higher.

Another result merging technique is to download all returned documents from their local servers and compute their matching scores using a common similarity function employed by the metasearch engine. The results will then be ranked based on these scores. For example, the Inquirus metasearch engine employs this approach [7]. The advantage of this approach is that it provides a uniform way to compute ranking scores so the resulted ranking makes more sense. Its main drawback is the longer response time due to the delay caused by downloading the documents and analyzing them on the fly. Most modern search engines display the title of each retrieved result together with a short summary called *snippet*. The title and snippet of a result can often provide good clues on whether or not the result is relevant to a query. As a result, result merging algorithms that rely on titles and snippets

have been proposed recently (e.g., [9]). When titles and snippets are used to perform the merging, a matching score of each result with the query can be computed based on several factors such as the number of unique query terms that appear in the title/snippet and the proximity of the query terms in the title/snippet.

It is possible that the same result is retrieved from multiple search engines. Such results are more likely to be relevant to the query based on the observation that different ranking algorithms tend to retrieve the same set of relevant results but different sets of irrelevant results [8]. To help rank these results higher in the merged list, the ranking scores of these results from different search engines can be added up to produce the final score for the result. The search results are then ranked in descending order of the final scores.

Search Engine Selection

To enable search engine selection, some information that can represent the contents of the documents of each component search engine needs to be collected first. Such information for a search engine is called the *representative* of the search engine. The representatives of all search engines used by the metasearch engine are collected in advance and are stored with the metasearch engine. During search engine selection for a given query, search engines are ranked based on how well their representatives match with the query. Different search engine selection techniques have been proposed and they often use different types of representatives. A simple representative of a search engine may contain only a few selected key words or a short description. This type of representative is usually produced manually by someone familiar with the contents of the search engine but it can also be automatically generated. As this type of representatives provides only a general description of the contents of search engines, the accuracy of using such representatives for search engine selection is usually low. More elaborate representatives consist of detailed statistical information for each term in each search engine. In [14], the *document frequency* of each term in each search engine is used to compute the *cue validity variance* of each query term, which measures the skew of the distribution of the query term across all component search engines, to help rank search engines for each query. In [3], the *document frequency* and *collection frequency* of each term (the latter is the number of component search engines that contain the term) are used to represent each search engine. In [10], the *adjusted maximum normalized weight* of each term across all documents in a search engine is used to represent a search engine. For a given term t and a search engine S, the adjusted maximum normalized weight of t is computed as follows: compute the normalized weight of t in every document (i.e., the term frequency weight of t divided by the length of the document) in S, find the maximum value among these weights, and multiply this maximum weight by the global *idf* weight of t across all component search engines. In [13], the notion of optimal search engine ranking is proposed based on the objective of retrieving the m most similar (relevant) documents with respect to a given query Q from across all component search engines: n search engines are said to be optimally ranked with order $[S_1, S_2,...,S_n]$ if for any integer m, an integer k can be found such that the m most similar documents are contained in $[S_1,...,S_k]$ and each of these k search engines contain at least one of the m most similar documents. It is shown in [13] that a necessary and sufficient condition for the component search engines to be optimally ranked is to order the search engines in descending order of the similarity of the most similar document with respect to Q in each search engine. Different techniques have been proposed to estimate the similarity of the most similar document with respect to a given query and a given search engine [10,13]. Since it is impractical to find out all the terms that appear in some pages in a search engine, an approximate vocabulary of terms for a search engine can be used. Such an approximate vocabulary can be obtained from pages retrieved from the search engine using probe queries [2].

There are also techniques that create search engine representatives by learning from the search results of past queries. Essentially such type of representatives is the knowledge indicating the past performance of a search engine with respect to different queries. In the Savvy Search metasearch engine [4], for each component search engine S, a weight is maintained for every term that has appeared in previous queries. After each query Q is evaluated, the weight of each term in the representative that appears in Q is increased or decreased depending on whether or not S returns useful results. Over time, if a term for S has a large positive (negative) weight, then S is considered to have responded well (poorly) to the term in the past. For a new query received by the metasearch engine, the weights of the query terms in the representatives of

different search engines are aggregated to rank the search engines. In the ProFusion metasearch engine [5], training queries are used to find out how well each search engine responds to queries in different categories. The knowledge learned about each search engine from training queries is used to select search engines for each user query and the knowledge is continuously updated based on the user's reaction to the search result, i.e., whether or not a particular retrieved result is clicked by the user.

Automatic Search Engine Connection

The search interfaces of most search engines are implemented using the HTML *form* tag with a query textbox. In most cases, the form tag of a search engine contains all information needed to make the connection to the search engine, i.e., sending queries and receiving search results, via a program. Such information includes the name and the location of the program (i.e., the search engine server) that evaluates user queries, the network connection method (i.e., the HTTP request method, usually GET or POST), and the name associated with the query textbox that is used to save the query string. The form tag of each search engine interface is usually pre-processed to extract the information needed for program connection and the extracted information is saved at the metasearch engine. The existence of Javascript in the form tag usually makes extracting the connection information more difficult. After the metasearch engine receives a query and a particular search engine, among possibly other search engines, is selected to evaluate this query, the query is assigned to the name of the query textbox of the search engine and sent to the server of the search engine using the HTTP request method supported by the search engine. After the query is evaluated by the search engine, one or more result pages containing the search results are returned to the metasearch engine for further processing.

Automatic Search Result Extraction

A result page returned by a search engine is a dynamically generated HTML page. In addition to the search result records for a query, a result page usually also contains some unwanted information/links such as advertisements and sponsored links. It is important to correctly extract the search result records on each result page. A typical search result record corresponds to a retrieved document and it usually contains the URL and the title of the page as well as a short summary (snippet) of the document. Since different search engines produce result pages in different format, a separate result extraction program (also called *extraction wrapper*) needs to be generated for each search engine. Automatic wrapper generation for search engines has received a lot of attention in recent years and different techniques have been proposed. Most of them analyze the source HTML files of the result pages as text strings or tag trees (DOM trees) to find the repeating patterns of the search record records. A survey that contains some of the earlier extraction techniques can be found in [6]. Some more recent works also utilize certain visual information on result pages to help identify result patterns (e.g., [15]).

Key Applications

The main application of metasearch is to support search. It can be an effective mechanism to search both surface web and deep web data sources. By providing a common search interface over multiple search engines, metasearch eliminates users' burden to search multiple sources separately. When a metasearch engine employs certain special component search engines, it can support interesting special applications. For example, for a large organization with many branches (e.g., a university system may have many campuses), if each branch has its own search engine, then a metasearch engine connecting to all branch search engines becomes an organization-wide search engine. As another example, if a metasearch engine is created over multiple e-commerce search engines selling the same type of product, then a comparison-shopping system can be created. Of course, for comparison-shopping applications, a different type of result merging is needed, such as listing different search results that correspond to the same product in non-descending order of the prices.

Future Directions

Component search engines employed by a metasearch engine may change their connection parameters and result display format anytime. These changes can make the affected search engines un-usable in the metasearch engine unless the corresponding connection programs and result extraction wrappers are changed accordingly. How to monitor the changes of search engines and make the corresponding changes in the metasearch engine automatically and timely is an area that needs

urgent attention from metasearch engine researchers and developers.

Most of today's metasearch engines employ only a small number of general-purpose search engines. Building large-scale metasearch engines using numerous specialized search engines is another area that deserves more attention. The current largest metasearch engine is a news metasearch engine called AllInOneNews (www.allinonenews.com). This metasearch engine currently connects to about 1,800 news search engines. Challenges arising from building very large-scale metasearch engines include automatic generation and maintenance of high quality search engine representatives needed for efficient and effective search engine selection, and highly automated techniques to add search engines into metasearch engines and to adapt to changes of search engines.

Cross-references

- ▶ Deep-Web Search
- ▶ Document Length Normalization
- ▶ Hidden-Web Search
- ▶ Information Extraction
- ▶ Information Retrieval
- ▶ Inverse Document Frequency
- ▶ Query Routing
- ▶ Result Integration
- ▶ Snippet
- ▶ Term Weighting
- ▶ Web Data Extraction
- ▶ Web Data Extraction System
- ▶ Web Information Retrieval Models
- ▶ Web Search Engines
- ▶ Wrapper
- ▶ Wrapper Generation
- ▶ Wrapper Induction
- ▶ Wrapper Maintenance
- ▶ Wrapper Stability

Recommended Reading

1. Aslam J. and Montague M. Models for metasearch. In Proc. 24th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2001, pp. 276–284.
2. Callan J., Connell M., and Du A. Automatic discovery of language models for text databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 479–490.
3. Callan J., Lu Z., and Croft W.B. Searching distributed collections with inference networks. In Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1995, pp. 21–28.
4. Dreilinger D. and Howe A. Experiences with selecting search engines using metasearch. ACM Trans. Inf. Syst., 15 (3):195–222, 1997.
5. Fan Y. and Gauch S. Adaptive agents for information gathering from multiple, distributed information sources. In Proc. AAAI Symp. on Intelligent Agents in Cyberspace, 1999, pp. 40–46.
6. Laender A.A., Ribeiro-Neto B., da Silva A., and Teixeira J. A brief survey of web data extraction tools. ACM SIGMOD Rec., 31 (2):84–93, 2002.
7. Lawrence S. and Lee Giles C. Inquirus, the NECi meta search engine. In Proc. 7th Int. World Wide Web Conference, 1998, pp. 95–105.
8. Lee J-H. Combining multiple evidence from different properties of weighting schemes. In Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1995, pp. 180–188.
9. Lu Y., Meng W., Shu L., Yu C., and Liu K. Evaluation of result merging strategies for metasearch engines. In Proc. 6th Int. Conf. on Web Information Systems Eng., 2005, pp. 53–66.
10. Meng W., Wu Z., Yu C., and Li Z. A highly scalable and effective method for metasearch. ACM Trans. Information Syst., 19(3):310–335, 2001.
11. Meng W., Yu C., and Liu K. Building efficient and effective metasearch engines. ACM Comput. Surv., 34(1):48–89, 2002.
12. Selberg E. and Etzioni O. The MetaCrawler architecture for resource aggregation on the web. IEEE Expert, 12(1):11–14, 1997.
13. Yu C., Liu K., Meng W., Wu Z., and Rishe N. A methodology to retrieve text documents from multiple databases. IEEE Trans. Knowledge and Data Eng., 14(6):1347–1361, 2002.
14. Yuwono B. and Lee D. Server ranking for distributed text resource systems on the internet. In Proc. 5th Int. Conf. on Database Systems for Advanced Applications, 1997, pp. 391–400.
15. Zhao H., Meng W., Wu Z., Raghavan V., and Yu C. Fully automatic wrapper generation for search engines. In Proc. 14th Int. World Wide Web Conf., 2005, pp. 66–75.

Metric Space

Pavel Zezula, Michal Batko, Vlastislav Dohnal
Masaryk University, Brno, Czech Republic

Synonyms

Distance space

Definition

In mathematics, a metric space is a pair $M = (D, d)$, where D is a domain of objects (or objects' *keys* or *indexed descriptors*) and d is a total (distance) function. The properties of the function $d : D \times D \mapsto R$,

sometimes called the metric space postulates, are typically characterized as:

(p1)	$\forall x, y \in D, d(x, y) \geq 0$	non-negativity,
(p2)	$\forall x, y \in D, d(x, y) = d(y, x)$	symmetry,
(p3)	$\forall x \in D, d(x, x) = 0$	reflexivity,
(p4)	$\forall x, y \in D, x \neq y \Rightarrow d(x, y) > 0$	positiveness,
(p5)	$\forall x, y, z \in D, d(x, z) \leq d(x, y) + d(y, z)$	triangle inequality.

Key Points

Modifying or even abandoning some of the metric function properties leads to interesting concepts that can better suit the reality in many situations. A *pseudo-metric* function does not satisfy the positiveness property (p4), i.e., there can be pairs of different objects that have zero distance. However, these functions can be transformed to the standard metric by regarding any pair of objects with zero distance as a single object. If the symmetry property (p2) does not hold, the function is called a *quasi-metric*. For example, a car-driving distance in a city where one-way streets exist is a quasi-metric. The following equation allows to transform a quasi-metric into a standard metric: $d_{sym}(x, y) = d_{asym}(x, y) + d_{asym}(y, x)$. By tightening the triangle inequality property (p5) to $\forall x, y, z \in D, d(x, z) \leq \max\{d(x, y), d(y, z)\}$, an *ultra-metric* also called *super-metric* is obtained. The geometric characterization of the ultra-metric requires every triangle to have at least two sides of equal length, i.e., to be isosceles. A metric space M is bounded if there exists a number r, such that $d(x, y) \leq r$ for any $x, y \in D$. More details about metric functions can be found in [3].

Cross-references

- ▶ Closest-Pair Query
- ▶ Indexing Metric Spaces
- ▶ Information Retrieval
- ▶ Nearest Neighbor Query
- ▶ Spatial Indexing Techniques

Recommended Reading

1. Burago D., Burago Y.D., and Ivanov S. A Course in Metric Geometry. American Mathematical Society, Providence, Rhode Island, USA, 2001.
2. Bryant V. Metric Spaces: Iteration and Application. Cambridge University Press, New York, USA, 1985.
3. Zezula P., Amato G., Dohnal V., and Batko M. Similarity Search: The Metric Space Approach, Springer-Verlag, Berlin, 2006.

Microdata

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms

Individual data

Definition

A *microdata* file V with s respondents and t attributes is an $s \times t$ matrix where V_{ij} is the value of attribute j for respondent i. Attributes can be numerical (e.g., age, salary) or categorical (e.g., gender, job).

Key Points

The attributes in a microdata set can be classified in four categories which are not necessarily disjoint [1,2]:

1. *Identifiers*. These are attributes that *unambiguously* identify the respondent. Examples are the passport number, social security number, name-surname, etc.
2. *Quasi-identifiers or key attributes*. These are attributes which identify the respondent with some degree of ambiguity. (Nonetheless, a combination of quasi-identifiers may provide unambiguous identification.) Examples are address, gender, age, telephone number, etc.
3. *Confidential outcome attributes*. These are attributes which contain sensitive information on the respondent. Examples are salary, religion, political affiliation, health condition, etc.
4. *Non-confidential outcome attributes*. Those attributes which do not fall in any of the categories above.

Cross-references

- ▶ k-Anonymity
- ▶ Data Rank/Swapping
- ▶ Inference Control in Statistical Databases
- ▶ Microdata Rounding
- ▶ Noise Addition
- ▶ Non-Perturbative Masking Methods
- ▶ PRAM
- ▶ SDC Score
- ▶ Tabular Data

Recommended Reading

1. Dalenius T. The invasion of privacy problem and statistics production: an overview. Statistik Tidskrift, 12:213–225, 1974.
2. Samarati P. Protecting respondents' identities in microdata release. IEEE Trans. on Knowl. and Data Eng., 13(6):1010–1027, 2001.

Microaggregation

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Tarragona, Catalonia

Definition

Microaggregation is a family of masking methods for statistical disclosure control of numerical microdata (although variants for categorical data exist). The rationale behind microaggregation is that confidentiality rules in use allow publication of microdata sets if records correspond to groups of k or more individuals, where no individual dominates (i.e., contributes too much to) the group and k is a threshold value. Strict application of such confidentiality rules leads to replacing individual values with values computed on small aggregates (microaggregates) prior to publication. This is the basic principle of microaggregation.

To obtain microaggregates in a microdata set with n records, these are combined to form g groups of size at least k. For each attribute, the average value over each group is computed and is used to replace each of the original averaged values. Groups are formed using a criterion of maximal similarity. Once the procedure has been completed, the resulting (modified) records can be published.

The optimal k-partition (from the information loss point of view) is defined to be the one that maximizes within-group homogeneity. The higher the within-group homogeneity, the lower the information loss, since microaggregation replaces values in a group by the group centroid. The sum of squares criterion is common to measure homogeneity in clustering. The within-groups sum of squares SSE is defined as

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)'(x_{ij} - \bar{x}_i)$$

The lower SSE, the higher the within-group homogeneity. Thus, in terms of sums of squares, the optimal k-partition is the one that minimizes SSE.

Key Points

For a microdata set consisting of p attributes, these can be microaggregated together or partitioned into several groups of attributes. Also the way to form groups may vary. Several taxonomies are possible to classify the microaggregation algorithms in the literature: (i) fixed group size vs. variable group size; (ii) exact optimal (only for the univariate case vs. heuristic microaggregation; (iii) continuous vs. categorical microaggregation.

To illustrate, a heuristic algorithm called MDAV (maximum distance to average vector, by Domingo-Ferrer, Mateo-Sanz and Torra) is next given for multivariate fixed group size microaggregation on unprojected continuous data. MDAV has been implemented in the μ-Argus package:

1. Compute the average record $\bar{x}$ of all records in the dataset. Consider the most distant record x_r to the average record $\bar{x}$ (using the squared Euclidean distance).
2. Find the most distant record x_s from the record x_r considered in the previous step.
3. Form two groups around x_r and x_s, respectively. One group contains x_r and the $k-1$ records closest to x_r. The other group contains x_s and the $k-1$ records closest to x_s.
4. If there are at least $3k$ records which do not belong to any of the two groups formed in Step 3, go to Step 1 taking as new dataset the previous dataset minus the groups formed in the last instance of Step 3.
5. If there are between $3k-1$ and $2k$ records which do not belong to any of the two groups formed in Step 3: (i) compute the average record $\bar{x}$ of the remaining records; (ii) find the most distant record x_r from $\bar{x}$; (iii) form a group containing x_r and the $k-1$ records closest to x_r; (iv) form another group containing the rest of records. Exit the Algorithm.
6. If there are less than $2k$ records which do not belong to the groups formed in Step 3, form a new group with those records and exit the Algorithm.

The above algorithm can be applied independently to each group of attributes resulting from partitioning the set of attributes in the dataset. Microaggregation can be used to achieve k-anonymity.

Cross-references

▶ Inference Control in Statistical Databases
▶ k-Anonymity

► Microdata
► SDC Score

Recommended Reading

1. Domingo-Ferrer J. and Mateo-Sanz J. M. Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. Knowl. Data Eng., 14(1):189–201, 2002.
2. Domingo-Ferrer J., Sebé F., and Solanas A. A polynomial-time approximation to optimal multivariate microaggregation. Comput. Math. Appl. 55(4):714–732, 2008.
3. Domingo-Ferrer J. and Torra V. Ordinal, continuous and heterogenerous *k*-anonymity through microaggregation. Data Mining Knowl. Dis., 11(2):195–212, 2005.
4. Hundepool A., Van de Wetering A., Ramaswamy R., Franconi L., Capobianchi A., DeWolf P.-P., Domingo-Ferrer J., Torra V., Brand R., and Giessing S. μ-ARGUS Version 4.0 Software and User's Manual. Statistics Netherlands, Voorburg NL, May 2005. http://neon.vb.cbs.nl/casc.

Microbenchmark

Denilson Barbosa[1], Ioana Manolescu[2], Jeffrey Xu Yu[3]
[1]University of Alberta, Edmonton, AB, Canada
[2]INRIA Saday, Orsay, Cedex, France
[3]The Chinese University of Hong Kong, Hong Kong, China

Definition

A micro-benchmark is an experimental tool that studies a given aspect (e.g., performance, resource consumption) of XML processing tool. The studied aspect is called the target of the micro-benchmark. A micro-benchmark includes a parametric measure and guidelines, explaining which data and/or operation parameters may impact the target, and suggesting value ranges for these parameters.

Key Points

Micro-benchmarks help capture the behavior of an XML processing system on a given operation, as a result of varying one given parameter. In other words, the goal of a micro-benchmark is to study the *precise* effect of a given system feature or aspect *in isolation*.

Micro-benchmarks were first introduced for object-oriented databases [2]. An XML benchmark sharing some micro-benchmark features is the Michigan benchmark [3]. The MemBeR project [1], developed jointly by researchers at INRIA Futurs, the University of Amsterdam, and University of Antwerpen provides a comprehensive repository of micro-benchmarks for XML.

Unlike application benchmarks, micro-benchmarks do not directly help determining which XML processing system is most appropriate for a given task. Rather, they are helpful in assessing particular modules, algorithms and techniques present inside an XML processing tool. Micro-benchmarks are therefore typically very useful to system developers.

Cross-references

► Application Benchmark
► XML Benchmarks

Recommended Reading

1. Afanasiev L., Manolescu I., and Michiels P. MemBeR: a micro-benchmark repository for XQuery. In Proc. Database and XML Technologies, 3rd Int. XML Database Symp., 2005, pp. 144–161.
2. Carey M.J., DeWitt D.J., and Naughton J.F. The OO7 Benchmark. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 12–21.
3. Runapongsa K., Patel J.M., Jagadish H.V., Chen Y., and Al-Khalifa S. The Michigan benchmark: towards XML query performance diagnostics. Inf. Syst., 31(2):73–97, 2006.

Microdata Rounding

Josep Domingo-Ferrer
Universitat Rovira i Virgili, Tarragona, Catalonia

Synonyms

Rounding

Definition

Microdata rounding is a family of masking methods for statistical disclosure control of numerical microdata; a similar principle can be used to protect tabular data. Rounding replaces original values of attributes with rounded values. For a given attribute X_i, rounded values are chosen among a set of rounding points defining a *rounding set* (often the multiples of a given base value).

Key Points

In a multivariate original dataset, rounding is usually performed one attribute at a time (*univariate* rounding); however, multivariate rounding is also possible [1,2]. The operating principle of rounding makes it suitable for continuous data.

Cross-references

- ▶ Inference Control in Statistical Databases
- ▶ Microdata
- ▶ SDC Score

Recommended Reading

1. Cox L.H. and Kim J.J. Effects of rounding on the quality and confidentiality of statistical data. In J. Domingo-Ferrer and L. Franconi (eds.). Privacy in Statistical Databases-, LNCS, vol. 4302, 2006, pp. 48–56.
2. Willenborg L. and DeWaal T. Elements of Statistical Disclosure Control. Springer-Verlag, New York, 2001.

Middleware Support for Database Replication and Caching

Emmanuel Cecchet
EPFL, Lausanne, Switzerland

Definition

Database replication is a technique that aims at providing higher availability and performance than a single RDBMS. A database replication middleware implements a number of replication algorithms on top of existing RDBMS. Features provided by the replication middleware include load balancing, caching, and fault tolerance.

Historical Background

Database replication is a well-known mechanism for performance scaling and availability of databases across a wide range of requirements. Limitations of 2-phase commit and synchronous replication have been pointed out early on by Gray et al. [7]. Since then, research on middleware-based replication addresses these issues and tries to provide solutions for better performance and availability while maintaining consistency guarantees for applications.

Foundations

Database replication is a wide area of research that encompasses multiple architectures and possible designs. This entry does not address in-core database replication, where the replication algorithms are implemented inside the database engine. Instead, it focuses on middleware-based replication, where the replication logic is implemented in a set of middleware components, outside the database engine.

Shared Disk Versus Shared Nothing

Two main architecture designs can be chosen for database replication. *Shared disk* replication is mostly used by in-core implementations, where replicas share the data storage, such as, a SAN (Storage Area Network). Middleware-based replication usually uses a *shared nothing* architecture, where each replica has its own local storage. This allows disk IOs to be distributed among replicas and prevents the storage from being a Single Point of Failure (SPOF).

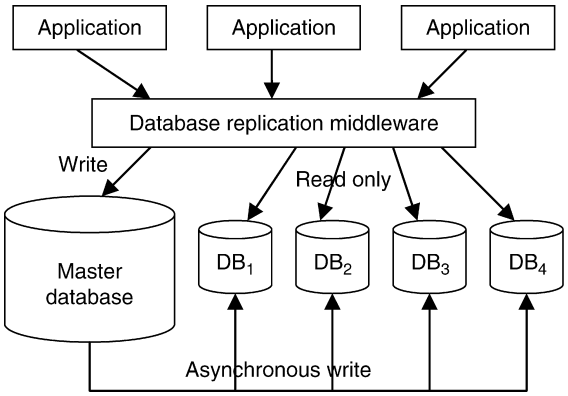
Master/Slave Versus Multi-Master

Database replication is often used as a way to scale up performance. Such efforts are typically targeted at increasing read performance or at increasing write performance; increasing both simultaneously is difficult.

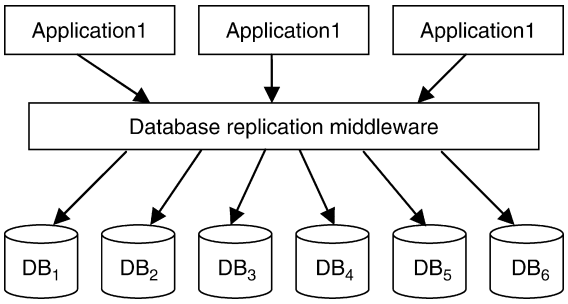
Master-slave replication, depicted in Fig. 1, is popular because it improves read performance. This setup is frequently used in e-commerce applications, with slave databases dedicated to product catalog browsing, while the master performs all catalog updates.

In this scenario, read-only content is accessed on the slave nodes and updates are sent to the master. If the application can tolerate loose consistency, any data can be read at any time from the slaves given a freshness guarantee. As long as the master node can handle all updates, the system can scale linearly simply by adding slave nodes.

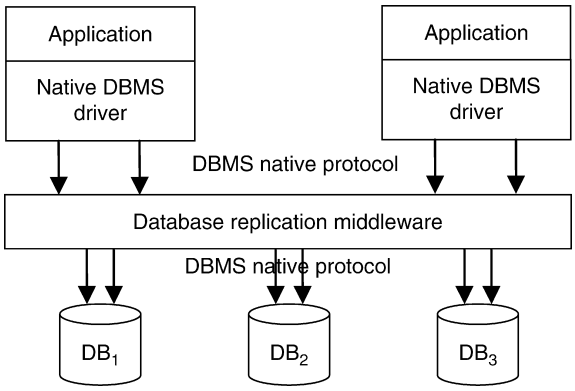
Multi-master replication, as shown on Fig. 2, allows each replica that owns a full copy of the database to serve both read and write requests. The replicated system then behaves as a centralized database which theoretically does not require any application modifications. However, replicas need to synchronize to agree on a serializable execution order of transactions so that each replica executes update transactions in the same order. Also, concurrent transactions might



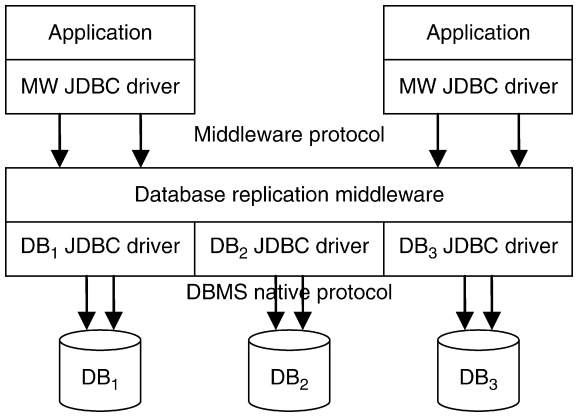
Middleware Support for Database Replication and Caching. Figure 1. Master/slave scale-out scenario.



Middleware Support for Database Replication and Caching. Figure 2. Multi-Master database replication.



Middleware Support for Database Replication and Caching. Figure 3. Query interception at the DBMS protocol level.



Middleware Support for Database Replication and Caching. Figure 4. Query interception in JDBC-based replication.

conflict leading to aborts and limiting the system scalability [7]. Even though real applications generally avoid conflicting transactions, significant efforts are spent to optimize for this problem in middleware replication research. However, the volume of update transactions remains the limiting performance factor for such systems.

Middleware Design

The replication middleware has to intercept client requests to process them and run them through the replication algorithm. A first technique consists of keeping existing database drivers and to intercept queries at the database protocol level as show on Fig. 3. This has the significant advantage to not have to re-implement database drivers but the database protocol specification must be available, which is not always the case for commercial databases. Moreover, this design requires multi-protocol implementations and bridges to support heterogeneous clusters.

Another technique provides the application with a replacement driver that can communicate with the replication middleware and that is API compatible with the original driver so that application changes are not required. Figure 4 shows an example of a middleware that intercepts queries at the JDBC level. The client application uses the middleware JDBC driver, and the middleware uses the database native JDBC driver to access the replicas. The middleware driver typically adds functionality such as load balancing and failover that is usually not present in standalone database drivers. This is a popular approach introduced by C-JDBC [3] (now Sequoia [9]) and used in other prototypes like Tashkent [5] and Ganymed [8].

Concurrency Control

In replicated database systems, each replica runs Snapshot Isolation (SI) as its local concurrency control and the replicated system provides Generalized Snapshot Isolation (GSI) to the clients.

Snapshot isolation (SI) is a multi-version database concurrency control algorithm for centralized databases. In snapshot isolation, when a transaction begins it receives a logical copy, called snapshot, of the database for the duration of the transaction. This snapshot is the most recent version of the committed state of the database. Once assigned, the snapshot is unaffected by (i.e., isolated from) concurrently running transactions. When an update transaction commits, it produces a new version of the database.

Many database vendors use SI, e.g., PostgreSQL, Oracle and Microsoft SQL Server. SI is weaker than serializability but in practice many applications run serializably under SI including the widely used database benchmarks TPC-C and TPC-W. SI has attractive performance properties. Most notably, read-only transactions do not block or abort-they do not need read-locks, and they do not cause update transactions to block or abort.

Generalized Snapshot Isolation (GSI) extends SI to replicated databases such that the performance properties of SI in a centralized setting are maintained in a replicated setting. In addition, workloads that are serializable under SI are also serializable under GSI.

Informally, a replica using GSI works as follows. When a transaction starts, the replica assigns its latest snapshot to the transaction. All read and write operations of a transaction, e.g., the SELECT, UPDATE, INSERT and DELETE SQL statements, are executed locally on the replica. At commit, the replica extracts the transaction writeset. If the writeset is empty (i.e., it is a read-only transaction), the transaction commits immediately. Otherwise, certification is performed to detect write-write conflicts among update transactions in the system. If no conflict is found, then the transaction commits, otherwise it aborts.

Certification results in total order on the commits of update transactions. Since committing an update transaction creates a new version (snapshot) of the database, the total order defines the sequence of snapshots the database goes through. Therefore, processing update transactions proceeds as follows: When a replica receives update transaction T, it executes T against a snapshot. At commit, the certification service receives the writeset of T and the version of the assigned snapshot. If certification is successful, the replica applies writesets of concurrent update transactions that committed before T in the order determined during certification and then commits T. Certification is a stateful service because it maintains recent committed writesets and their versions.

Statement Replication Versus Transaction Replication

Multi-master replication can be implemented either by multicasting every update statement (statement replication) or by capturing transaction writesets (the set of data W updated by a transaction T, such that applying W onto a replica is equivalent to executing T on it) and propagating them after certification (transaction replication). Both approaches have different performance/availability tradeoffs.

Statement-based replication requires that the execution of an update statement produces the same result on each replica. However, many SQL statements may produce different results on every replica if they are not processed before execution. This requires macros related to timing or random numbers to be preprocessed for a deterministic execution cluster-wide. Moreover, stored procedures or user-defined functions must have a deterministic behavior to prevent replicas from diverging in content. The advantage of statement-based replication is that it can replicate any kind of SQL statement including DDL (Data Definition Language) queries that alters the database schema or requests that modify non-persistent objects such as environment variables, sequences or temporary tables.

Transaction replication relies on writeset extraction that is usually implemented using triggers. This requires declaring additional triggers on every database table. This can become complex if the database already uses triggers, materialized views or temporary tables. Writeset extraction does not capture changes such as auto-incremented keys, sequence values or environment variable updates. Queries altering such database structures change the replica they execute on and can result in cluster divergence. Moreover, most of these data structures cannot be rolled back (for instance, an auto-incremented key or sequence number incremented in a transaction is not decremented at rollback time).

With statement replication, all replicas execute write transactions simultaneously in the same serializable order, whereas transaction replication executes update transaction at only one replica and propagates

the transaction writeset to other replicas only after certification at commit time. Therefore, transaction replication usually offers better performance over statement replication as long as the writeset extraction and certification mechanisms are efficient. Statement replication offers a better infrastructure for failover during a transaction since each replica has a copy of every transactional context. With transaction replication, the failure of the replica executing the transaction will systematically abort the transaction and force the transaction to retry.

High Availability

High availability is often synonymous with little downtime. Such downtime can be either planned or unplanned, depending on whether it occurs under the control of the administrator or not. Planned downtime is incurred during most software and hardware maintenance operations, while unplanned downtime can strike at any time and is caused by foreseeable and unforeseeable failures (hardware failures, software bugs, human error, etc.).

A system's availability is the ratio of its uptime to total time. In practice, it is computed as the ratio between the expected time of continuous operation between failures to total time, or

$$Availability = \frac{MTTF}{MTTF + MTTR} \Rightarrow Unavailability = \frac{MTTR}{MTTF + MTTR} \approx \frac{MTTR}{MTTF}$$

where MTTF is Mean Time To Failure and MTTR is Mean Time To Repair. Since MTTF >> MTTR, one can approximate unavailability (ratio of downtime to total time) as MTTR/MTTF.

The goal of replication together with failover/failback is to reduce MTTR, and thus reduce unavailability. Failover is the ability for users of a database node to be switched over to another database node containing a replica of the data whenever the node they were connected to has failed. Failback happens when the original replica comes back from its failure and users are re-allocated to that replica.

A replicated database built for availability must eliminate any single point of failure (SPOF). This means that the middleware components (load balancer, certifier...) must also be replicated. Group communication libraries are used to synchronize the state of the different components. Total order is usually required by replication protocols to ensure a serializable execution order.

Several database drivers or connection pools offer automatic reconnection when a failure is detected. This technique only offers session failover but not failover of the transactional context. Sequoia [9] (the continuation of the C-JDBC project) provides transparent failover without losing transactional context. Failover code is available in the middleware to handle a database failure and additional code is available in the middleware driver running in the application to handle a middleware failure. A fully transparent failover requires consistently replicated state kept at all components, and is more easily achieved using statement-based rather than transaction-based replication. In the latter case, the transaction is only played at a single replica; if the replica fails, the entire transaction has to be replayed at another replica, which cannot succeed without the cooperation of the application.

Load Balancing

Load balancing aims at dispatching user requests or transactions to the replica that can provide consistent data with the lowest latency. Load balancer design is tightly coupled with the replication strategy implemented. Static strategies such as round-robin or even weighted-round-robin are usually not well adapted to the dynamic nature of transactional workloads. Algorithms taking into account replica resource usage such as LPRF (Least Pending Request First) perform much better. With additional information on transaction working set, it is also possible to optimize load balancing to improve in-memory request execution such as MALB (Memory-Aware Load Balancing) used in Tashkent+ [6]. More information on load balancing can be found in [1].

Caching

To reduce request execution time, the middleware can provide multiple caches. C-JDBC [3] provides three different caches. The parsing cache stores the results of query parsing so that a query that is executed several times is parsed only once. The metadata cache records all ResultSet metadata such as column names and types associated with a query result.

These caches work with query skeletons found in PreparedStatements used by application servers. A query skeleton is a query where all variable fields are replaced with question marks and filled at runtime

with a specific API. An example of a query skeleton is "SELECT * FROM t WHERE x=?". In this example, a parsing or metadata cache hit will occur for any value of x.

The query result cache is used to store the ResultSet associated with each query. The query result cache reduces the request response time as well as the load on the database replicas. By default, the cache provides strong consistency. In other words, C-JDBC invalidates cache entries that may contain stale data as a result of an update query. Cache consistency may be relaxed using user-defined rules. The results of queries that can accept stale data can be kept in the cache for a time specified by a staleness limit, even though subsequent update queries may have rendered the cached entry inconsistent.

Different cache invalidation granularities are available ranging from database-wide invalidation to table-based or column-based invalidation. An extra optimization concerns queries that select a unique row based on a primary key. These queries are often issued by application servers using JDO (Java Data Objects) or EJB (Enterprise Java Beans) technologies. These entries are never invalidated on inserts since a newly inserted row will always have a different primary key value and therefore will not affect this kind of cache entries. Moreover, update or delete operations on these entries can be easily performed in the cache.

Key Applications

Middleware-based database replication is currently used in many e-Commerce production environments that require both high availability and performance scalability. Many open problems remain on the integration of databases with replication middleware, failure detection and transparent failover/failback, autonomic management and software upgrades.

Future Directions

Current research trends explore autonomic behavior for replicated databases [4] to automate all management operations such as provisioning, tuning, failure repair and recovery. Heterogeneous clustering is also used in the context of satellites databases [8] to scale legacy databases with open source databases. Partial replication is studied in conjunction with WAN (Wide Area Network) replication for global applications spanning over multiple datacenters distributed on different continents. A summary of the remaining gaps between the theory and practice of middleware-based replication can be found in [2].

URL to Code

The Sequoia source code is available from http://sequoia.continuent.org

Cross-references

- ▶ Autonomous Replication
- ▶ Caching and Replication
- ▶ Consistency Models for Replicated Data
- ▶ 1-Copy-Serializability
- ▶ Data Broadcasting
- ▶ Data Partitioning
- ▶ Data Replication
- ▶ Database Clusters
- ▶ Database Middleware
- ▶ Distributed Database Design
- ▶ Distributed Database Systems
- ▶ Distributed DBMS
- ▶ Distributed Deadlock Management
- ▶ Distributed Query Processing
- ▶ Distributed Recovery
- ▶ Extraction
- ▶ Inter-Query Parallelism
- ▶ Middleware Support for Precise Failure Semantics
- ▶ Partial Replication
- ▶ Replica Control
- ▶ Replica Freshness
- ▶ Replicated Database Concurrency Control
- ▶ Replication
- ▶ Replication Based on Group Communication
- ▶ Replication for High Availability
- ▶ Replication for Scalability
- ▶ Replication in Multi-Tier Architectures Regular
- ▶ Shared-Disk Architecture
- ▶ Shared-Nothing Architecture
- ▶ Snapshot Isolation
- ▶ Strong Consistency Models for Replicated Data
- ▶ Transactional Middleware
- ▶ Transformation and Loading
- ▶ Weak Consistency Models for Replicated Data

Recommended Reading

1. Amza C., Cox A., and Zwaenepoel W. A comparative evaluation of transparent scaling techniques for dynamic content servers. In Proc. 21st Int. Conf. on Data Engineering, 2005, pp. 230–241.

2. Cecchet E., Candea G., and Ailamaki A. Middleware-based database replication: the gaps between theory and practice. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2008, pp. 739–752.
3. Cecchet E., Marguerite J., and Zwaenepoel W. C-JDBC: flexible database clustering middleware. In Proc. USENIX Annual Technical Conf., 2004.
4. Chen J., Soundararajan G., and Amza C. Autonomic provisioning of backend databases in dynamic content web servers. In Proc. IEEE Int. Conf. Autonomic Computing, 2006, pp. 231–242.
5. Elnikety S., Dropsho S., and Pedone F. Tashkent: uniting durability with transaction ordering for high-performance scalable database replication. In Proc. 1st ACM SIGOPS/EuroSys European Conf. on Comp. Syst., 2006, pp. 117–130.
6. Elnikety S., Dropsho S., and Zwaenepoel W. Tashkent+: memory-aware load balancing and update filtering in replicated databases. In Proc. 2nd ACM SIGOPS/EuroSys European Conf. on Comp. Syst., 2007, pp. 399–412.
7. Gray J.N., Helland P., O'Neil P., and Shasha D. The dangers of replication and a solution. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 173–182.
8. Plattner C., Alonso G., Özsu M.T. Extending DBMSs with satellite databases. VLDB J., 17(4):657–682, 2008.
9. Sequoia project. Available at: http://sequoia.continuent.org

Middleware Support for Precise Failure Semantics

Vivien Quéma
CNRS, INRIA, Saint-Ismier Cedex, France

Definition

Providing support for precise failure semantics requires defining an appropriate correctness criterion for replicated action execution of a replication algorithm. Such a correctness criterion allows formally verifying that a sequence of actions is executed correctly. In the context of replication, a sequence of actions executed is correctly if their side-effect appears to have happened exactly-once.

Historical Background

Reasoning about the behavior of concurrent programs has been an active research area during the past decades. Of particular interest in this area are the works on linearizability, a consistency criterion for concurrent objects [4], and on serializability, a consistency criterion for concurrent transactions [5]. These two criteria facilitate certain kinds of formal reasoning by transforming assertions about complex concurrent behavior into assertions about simpler sequential behavior. Moreover, these consistency criteria are local properties: the correctness of individual objects or services is used to reason about system-level correctness. Recently, Frølund and Guerraoui introduced x-ability [2] (exactly-once ability), a correctness criterion for replicated services. X-ability is independent of particular replication algorithms. Although they facilitate reasoning in similar ways, there are fundamental differences between x-ability on one hand and serializability [5] and linearizability [4] on the other. X-ability has safety as well as liveness aspects to it whereas serializability and linearizability are safety conditions only. X-ability is a theory of distribution and partial failures where serializability and linearizability are theories of concurrency. X-ability does not specify correctness for concurrent invocations of a replicated service. More precisely, x-ability states constraints about the concurrency among replicas in the context of a given request (intra-request concurrency), but ignores the concurrency that originates from different requests (inter-request concurrency). This entry gives a precise description of the x-ability theory.

Foundations

Frølund and Guerraoui proposed x-ability [2] (exactly-once ability), a correctness criterion for replicated services. X-ability is independent of particular replication algorithms. The main idea behind x-ability is to consider a replicated service correct if it provides the illusion of a single, fault-tolerant entity. More precisely, an x-able service must satisfy a contract with its clients as well as a contract with third-party entities. In terms of clients, a service must provide idempotent, non-blocking request processing. Moreover, it must deliver replies that are consistent with its invocation history. The side-effect of a service, on third-party entities, must obey exactly-once semantics. X-ability is a local property: replicated services can be specified and implemented independently, and later composed in the implementation of more complex replicated services.

To model side-effects, x-ability is based on the notion of action execution. Actions are executed correctly (i.e., are *x-able*) if their side-effect *appears* to have happened *exactly-once*. The side-effect of actions can be the modification of a shared state or the invocation of another (replicated or non-replicated) service. X-ability represents the execution of actions as event

histories and defines the notion of "appears to have happened exactly-once" in terms of history equivalence: an event history h is x-able if it is equivalent to a history h' obtained under failure-free conditions. Being defined relative to failure-free executions, x-ability encompasses both safety and liveness. It is a safety property because it states that certain partial histories must not occur. It is also a liveness property since it enforces guarantees about what must occur. History equivalence is defined relatively to the execution of two particular kinds of actions, namely *idempotent* and *undoable* actions:

- The side-effect of a history with n incarnations of an idempotent action is equivalent to a history with a single incarnation. For example, writing a particular value to a data object is an idempotent action.
- An undoable action is similar to a transaction: its side-effect can be cancelled up to a certain point (the commit point), after which the side-effect is permanent. Thus, the side-effect of a history with a cancelled action is equivalent to the side-effect of a history with no action at all.

System Model

The replicated service is implemented by a set of replicas. The functionality of the service is captured by a *state machine*. Each replica has its own copy of the state machine. A state machine exports a number of *actions*. An action takes an input value and produces an output value. In addition, an action may modify the internal state of its state machine and it may communicate with external entities. A client can invoke a replica's state machine by sending a request to the replica. A request contains the name of an action and an input value for the action. When it receives a request, a replica invokes its state machine based on the values in the request. If no failures occur, the replica returns the action's output value to the client. The execution of an action may fail or the replica executing the action may fail. If the action fails, it returns an exception (or error) value as the execution result. Formally speaking, action names are modeled as elements of a set `Action` (referred to using the letter a). The set `Value` contains the input and output values associated with actions. Furthermore, two sets, `Request` and `Result`, are defined as follows: `Request`$=$(`Action` $\times$ `Value`) and `Result`$=$`Value`. This signifies that a request is simply a pair that contains an action name and an input value (noted "(a, v)" for a request with action name a and value v).

The actions performed by state machines are represented by *events*. More precisely, the x-ability theory considers two kinds of events: start events to represent the invocation of a state-machine action by a process, and completion events to represent the successful completion of a state-machine action: a process receives a non-exception value back from the state machine. The causal and temporal relationship between action execution and event observation is subject to the following axioms: (i) an action's start event cannot be observed unless the action is invoked, (ii) an action's completion event cannot be observed before its start event, and (iii) if an action returns successfully, then its start and completion events have been observed. Events are modeled as elements of the set `Event`. Events are structured values with the following structure: $e ::= S(a, iv) \mid C(a, ov)$. The event $S(a, iv)$ captures the start of executing the action a with iv as argument. The event $C(a, ov)$ captures the completion of executing the action a, and ov is the output value produced by the action.

A sequence of events form a *history*. The notion of a sequence captures the total order in which events are observed. Histories are modeled as elements of the set `History`. Histories are structured values as defined by the following syntax: $h ::= \Lambda \mid e_1 ... e_n \mid h_1 \bullet ... \bullet h_n$. The symbol Λ denotes the empty history – a history with no events. The history $e_1 ... e_n$ contains the events e_1 through e_n. The history $h_1 \bullet ... \bullet h_n$ is the concatenation of histories h_1 through h_n. The semantics of concatenating histories is to concatenate the corresponding event sequences. An action a *appears* with input value iv in a history h (noted $(a, iv) \in h$) if h contains a start event produced by the execution of a on iv.

To capture structural properties of histories, the x-ability theory defines the notion of history *patterns*. Formally speaking, patterns are elements of the set `Pattern` (referred to using the letter p). The abstract syntax for patterns is depicted in Fig. 1. A simple pattern sp matches single-action histories. The pattern

$$sp ::= [a, iv, ov] \mid ?[a, iv, ov]$$
$$p ::= sp \mid sp_1 \|_h sp_2$$

Middleware Support for Precise Failure Semantics.
Figure 1. Abstract syntax for history patterns.

$[a, iv, ov]$ matches a history that contains the events from a failure-free execution of an action a. The value iv is the input to a and ov is the output from a. The pattern $?[a, iv, ov]$ matches a history in which a may have failed. A matching history may be the empty history, it may contain a start event only, or it may contain both the start and completion event of a. The pattern $sp_1 \parallel h\ sp_2$ matches a history h' that contains an interleaving of three sub-histories h_1, h_2, and h, where h_1 matches sp_1, h_2 matches sp_2, and h is an arbitrary history. The interleaving is constrained as follows: the first event in h_1 must also be the first event in h' and the last event in h_2 must also be the last event in h'.

X-ability defines *pattern matching* as a relation $\triangleright$ between elements of the set `History` and elements of the set `Pattern`. In other words, $\triangleright$ is a subset of `History` $\times$ `Pattern` (the set of all pairs from `History` and `Pattern`). Pattern matching rules are shown in Fig.2. A history that matches a simple pattern contains at most two events. X-ability defines two operators on such histories: `first` and `second` (see Fig.3). The `first` operator returns the first element in a history, if any, and Λ otherwise. The `second` operator returns the second element in a history of length two, the only element in a history of length one, and the empty history otherwise.

X-Able Histories

To be fault-tolerant, a replicated service must be prepared to invoke the same action multiple times until the action executes successfully. To provide replication transparency, the service must have exactly-once semantics relative to its environment – the service must maintain the illusion that the action was executed once only. In short, an x-able history is a history that maintains the illusion of exactly-once but possibly contains multiple incarnations of the same action. The rest of this section describes how the x-ability theory defines the notion of x-able history.

The x-ability theory defines a *history reduction* relation, $\Rightarrow$, on histories as follows: if $h \Rightarrow h'$, then the execution that produced h has the same side-effect as an execution that produced h'. Essentially, a history is x-able if it can be reduced, under $\Rightarrow$, to a history that could arise from a system that does not fail. Two particular types of actions are considered: `idempotent` and `undoable`. The corresponding sets are called Idempotent and Undoable. The set

(1) $\triangleright \subseteq (\texttt{History} \times \texttt{Pattern})$

(2) $S(a, iv)C(a, ov) \triangleright [a, iv, ov]$

(3) $\Lambda \triangleright ?[a, iv, ov]$

(4) $S(a, iv) \triangleright ?[a, iv, ov]$

(5) $S(a, iv)C(a, ov) \triangleright ?[a, iv, ov]$

(6) $$\frac{h_1 \triangleright sp_1 \quad h_2 \triangleright sp_2}{(h_1 \bullet h \bullet h_2) \triangleright (sp_1 \|_h sp_2)}$$

(7) $$\frac{h_1 \triangleright sp_1 \quad h_2 \triangleright sp_2}{(\text{first}(h_1) \bullet h_3 \bullet \text{second}(h_1) \bullet h_4 \bullet \text{first}(h_2) \bullet h_5 \bullet \text{second}(h_2)) \triangleright (sp_1 \|_{h_3 \bullet h_4 \bullet h_5} sp_2)}$$

(8) $$\frac{h_1 \triangleright sp_1 \quad h_2 \triangleright sp_2}{(\text{first}(h_1) \bullet h_3 \bullet \text{first}(h_2) \bullet h_4 \bullet \text{second}(h_1) \bullet h_5 \bullet \text{second}(h_2)) \triangleright (sp_1 \|_{h_3 \bullet h_4 \bullet h_5} sp_2)}$$

Middleware Support for Precise Failure Semantics. Figure 2. Pattern matching rules.

(9) $\text{first}(\Lambda) = \Lambda$ $\quad$ $\text{first}(e_1 e_2) = e_1$

(10) $\text{first}(e) = e$ $\quad$ $\text{second}(\Lambda) = \Lambda$

(11) $\text{second}(e) = e$ $\quad$ $\text{second}(e_1 e_2) = e_2$

Middleware Support for Precise Failure Semantics. Figure 3. The definition of first and second.

Idempotent contains the names of idempotent actions. The notation a^i indicates that the action a is idempotent. The set Undoable contains names of undoable actions. The notation a^u indicates that an action a is undoable. An undoable action, a^u, has two associated actions: a cancellation action, a^{-1}, and a commit action, a^c. The commit and cancellation actions for an action a^u take the same arguments as a^u, and they return the value nil. Moreover, cancellation and commit actions are idempotent.

Figure 4 defines the $\Rightarrow$ operator in terms of idempotent and undoable actions. The first inference rule (13) defines $\Rightarrow$ as a transitive relation. The second rule (14) captures the semantics of idempotent actions. If a history contains a successfully executed idempotent action a^i, then the events from a previous attempt to execute a^i can be removed. The third rule (15) is concerned with cancellation of undoable actions. Intuitively, if an undoable action is successfully cancelled, then its side-effect can be removed. The fourth rule (16) states that commit actions are idempotent.

The x-ability theory defines a *failure-free history* as a history that could have been produced by a failure-free execution of a single state-machine action. To define the notion of failure-free history, the x-ability theory relies on the definition of a function, called eventsof, which returns the failure-free history associated with an action and its values.

$$\texttt{eventsof}(a^u,\ iv,\ ov) = S(a^u, iv)C(a^u, ov)\ S(a^c,\ iv)C(a^c, \text{nil}) \quad (17)$$

$$\texttt{eventsof}(a^i) = S(a^i, iv)C(a^i, ov) \quad (18)$$

Due to non-determinism, there are multiple failure-free histories which are possible for a given action a and a given input value iv. The set of all possible histories, $\texttt{FailureFree}_{(a,iv)}$, is defined as follows:

$$\texttt{FailureFree}_{(a,iv)} = \{h \in \texttt{History} \mid \exists\, ov \in \texttt{Result} : \texttt{h} = \texttt{eventsof}(a, iv, ov)\} \quad (19)$$

An *x-able history* is defined as a history that can be reduced to a failure-free history. Formally speaking, an x-able history is one that satisfies the predicate x-able on histories:

$$\texttt{X-4able}_{(a,iv)}(h) = \begin{cases} \texttt{true} & \text{if } \exists h' \in \texttt{FailureFree}_{(a,iv)} : h \Rightarrow h' \\ \texttt{false} & \text{otherwise} \end{cases} \quad (20)$$

This definition of x-able histories applies to single-action histories, that is, a history that arises from a particular request. This reflects the fact that x-ability only specifies correctness relative to distribution and failures, it does not specify correctness for the concurrent processing of multiple requests from different clients.

Client-Service Consistency

The x-ability theory formalizes the relationship between clients and services. More precisely, the reply value given to a client in response to a request must be the value returned from the server-side state machine when the service processes the request. Moreover, the service is not allowed to invent requests. The server-side history is used to define the constraints for requests and replies. This history contains a request value as part of start events and reply values as part of

(12) $$\Rightarrow \subseteq (\texttt{History} \times \texttt{History})$$

(13) $$\frac{h_1 \Rightarrow h_2 \quad h_2 \Rightarrow h_3}{h_1 \Rightarrow h_3}$$

(14) $$\frac{h \triangleright (?[a^i, iv, ov] \parallel_{h'} [a^i, iv, ov])}{h_1 \bullet h \bullet h_2 \Rightarrow h_1 \bullet h' \bullet (S(a^i, iv)C(a^i, ov)) \bullet h_2}$$

(15) $$\frac{h \triangleright (?[a^u, iv, ov] \parallel_{h'} [a^{-1}, iv, \text{nil}]) \quad (a^u, iv) \notin h_1 \quad (a^c, iv) \notin h'}{h_1 \bullet h \bullet h_2 \Rightarrow h_1 \bullet h' \bullet h_2}$$

(16) $$\frac{h \triangleright (?[a^c, iv, \text{nil}] \parallel_{h'} [a^c, iv, \text{nil}]) \quad (a^u, iv) \notin h'}{h_1 \bullet h \bullet h_2 \Rightarrow h_1 \bullet h' \bullet (S(a^c, iv)C(a^c, \text{nil}) \bullet h_2}$$

Middleware Support for Precise Failure Semantics. Figure 4. Definition of history reduction.

completion events. The x-ability theory introduces the notion of *history signature*, which captures the client-side information (request and result) that is legal relative to a given server-side history. Because of non-determinism and server-side retry, a history can have multiple signatures. The set of signatures is defined by the following inference rules:

$$\frac{h \Rightarrow S(a^u, iv)C(a^u, ov)S(a^c, iv)C(a^c, \text{nil})}{(a, iv, ov) \in \texttt{signature}(h)} \quad (21)$$

$$\frac{h \Rightarrow S(a^i, iv)C(a^i, ov)}{(a, iv, ov) \in \texttt{signature}(h)} \quad (22)$$

If a client submits a sequence of requests, one after the other, later requests should be processed in the context of earlier requests. To prevent a service from forgetting the effect of previous requests, the x-ability theory assumes the existence of a set `PossibleReply` that contains the possible reply values for a given request. To capture the history-sensitive nature of the set of possible replies, PossibleReply is defined in the context of a request sequence $R_1 ... R_n$. The interpretation of `PossibleReply` in the context of a sequence is the set of possible replies to request R_n after the state machine has executed the requests $R_1 ... R_{n-1}$ one after the other. Thus, the set is written as follows: $\texttt{PossibleReply}_{(R_1 ... R_n)}$.

X-Able Services

The x-ability theory provides a formal specification of replication that is independent of a particular replication protocol. Formally speaking, a replicated service consists of a server-side state machine *S* and a client-side action *submit*. The state machine captures the functionality of the service. It is executed by a set of server processes $s_1 ... s_n$ that each have a copy of *S*. The action *submit* can be used by any process *p* to invoke the service. The action takes a value in the domain `Request` and, when executed, produces a value in the domain `Result`. Correctness is specified relative to a single client *C*. Thus, the considered system consists of the processes $s_1 ... s_n$ and *C* only. The client submits one request at a time, and the service is x-able if the following conditions hold:

- R1. The action *submit* is idempotent.
- R2. The client *C* will eventually be able to execute *submit* successfully.
- R3. If the client submits a request (a, iv), then the server-side history for (a, iv) is either empty or it satisfies $\texttt{x-able}_{(a,iv)}$.
- R4. If the client receives a reply *ov* in response to a request (a, iv), and if the server-side history for executing this request is *h*, then $(a, iv, ov) \in \texttt{signature}(h)$.
- R5. If the client successfully submits a sequence of requests, $R_1 ... R_n$, and receives the reply R' in response to R_n, then R' is in $\texttt{PossibleReply}_{(R_1 ... R_n)}$.

The first two requirements (R1 and R2) are concerned with the contract between a service and its clients. Clients use the action *submit* to invoke the service. Because *submit* is idempotent, clients can repeatedly invoke the service without concern for duplicating side-effects. The second requirement (R2) is a liveness property. The action *submit* is not allowed to fail an infinite number of times. The requirement also makes a service non-blocking in the sense that *submit* is guaranteed to eventually return a value. The third requirement (R3) deals with the server-side side-effect of executing a request. The resulting server-side history must be x-able, that is, it must be equivalent (under history reduction) to a failure-free history. The fourth requirement (R4) forces an algorithm to preserve consistency between the client-side view (request and reply) and the server-side view (the side-effect). This requirement, prevents the *submit* action from inventing reply values. It also prevents the service from inventing request values. The fifth requirement (R5) forces the service to correctly maintain *S*'s state, if any. The server-side history must be equivalent to a failure-free execution of the sequence $R_1 ... R_n$. But since R_1 may result in a transformation of *S*'s state, the actions executed for R_2 may depend on this state transformation. So, a replication algorithm must ensure that the state resulting from R_1 is used as a context for executing R_2. The replication algorithm cannot assume that R_1 did not update the state of *S*, or that the state update is immaterial to the processing of R_2.

Key Applications

A key application of the x-ability theory is the design transactions protocols for three-tier applications. Such applications encompass three layers: human users interact with front-end clients (e.g., browsers), middle-tier application servers (e.g., Web servers) contain

the business logic of the application, and perform transactions against back-end databases. Three-tier applications usually rely on replication and transaction-processing techniques. It has been defined in [1,3] the notion of the Exactly-Once Transaction (e-Transaction) abstraction: an abstraction that encompasses both safety and liveness properties in three-tier environments and ensures end-to-end reliability.

Recommended Reading

1. Frølund S. and Guerraoui R. Implementing e-transactions with asynchronous replication. IEEE Trans. Parallel Distrib. Syst., 12 (2):133–146, 2001.
2. Frølund S. and Guerraoui R. X-ability: a theory of replication. Distrib. Comput., 14(4):231–249, 2001.
3. Frølund S. and Guerraoui R. e-Transactions: end-to-end reliability for three-tier architectures. IEEE Trans. Software Eng., 28(4):378–395, 2002.
4. Herlihy M. and Wing J.M. Linearizability: a correctness condition for concurrent objects. ACM Trans. Program. Lang. Syst., 12(3):463–492, 1990.
5. Papadimitriou C.H. The serializability of concurrent database updates. J. ACM, 26(4):631–653, 1979.

Mini

► Snippet

Minimal-change Integrity Maintenance

► Constraint-Driven Database Repair

Mining of Chemical Data

Xifeng Yan
IBM T. J. Watson Research Center, Hawthorne, NY, USA

Definition

Given a set of chemical compounds, chemical data mining is to characterize the compounds present in the data set and apply a variety of mining methods to discover relationships between the compounds and their biological and chemical activities.

Historical Background

In 1969, Hansch [6] introduced quantitative structure-activity relationship (QSAR) analysis which attempts to correlate physicochemical or structural properties of compounds with biological and chemical activities. These physicochemical and structural properties are determined empirically or by computational methods. QSAR prefers vectorial mappings of compounds, which are usually coded by existing physicochemical and structural fingerprints. Dehaspe et al. [3] applied inductive logic programming to predict chemical carcinogenicity by mining frequent substructures in chemical datasets, which identifies new structural fingerprints so that QSAR could build comprehensive analytical models.

Foundations

Chemical compounds are unstructured data with no explicit vector representation. For chemical compounds, various similarity measures are defined, which could be classified into three categories: (i) physicochemical property-based, e.g., toxicity and weight; (ii) structure-based; and (iii) feature-based. The structure-based similarity measure directly compares the topology of two chemical compounds, e.g., maximum common subgraph, graph edit distance, and graph kernel. As for the feature-based similarity measure, each graph is represented as a feature vector, $\mathbf{x} = [x_1, x_2,...,x_n]$, where x_i is the value of feature f_i. A feature could be physicochemical or structural. The similarity between two graphs is measured by the similarity between their feature vectors. Bunke and Shearer [1] used maximum common subgraph to measure structure similarity. Given two graphs G and G', if P is the maximum common subgraph of G and G', then the structure similarity between G and G' is defined by

$$\frac{2|E(P)|}{|E(G)| + |E(G')|},$$

where $E(G)$ is the edge set of G.

Kashima et al. [7] introduced marginalized kernels between labeled graphs,

$$K(G, G') = \sum_{h} \sum_{h'} K_z(z, z') p(h|G) p(h'|G'),$$

where $z = [G,h]$ and $K_z(z,z')$ is the joint kernel over z. The hidden variable h is a path generated by random walks and the joint kernel K_z is defined as a kernel between these paths. Other sophisticated graph kernels are also available. For example, Fröhlich et al. [5]

proposed optimal assignment kernels for attributed molecular graphs, which compute an optimal assignment from the atoms of one molecule to those of another one, including local structures and neighborhood information.

The structure-based similarity measure can serve general chemical data mining such as chemical structure classification and clustering. The implicit definition of feature space makes it hard to interpret, and hard to adapt to many powerful data management and analytical tools such as R-tree and support vector machine. An alternative approach is to mine the most interesting features from chemical data directly, such as patterns that are discriminative between compounds with different chemical activities. Figure 1 depicts the pipeline of this approach built on features discovered by a mining process.

The feature-based mining framework includes three steps: (i) mine patterns/features from chemical data, (ii) select discriminative or significant features, and (iii) perform advanced mining. The first step is the process of finding and extracting useful features from raw datasets. One kind of features used in data mining is frequent substructures, the common structures that occur in many compounds. Formally, given a graph dataset $D = \{G_1,G_2,...,G_n\}$ and a minimum frequency threshold θ, frequent substructures are subgraphs that are contained by at least $\theta|D|$ graphs in D. A set of graph pattern mining algorithms are available for mining frequent substructures, including SUBDUE, Warmr, AGM, gSpan, FSG, MoFa/MoSS, FFSM, Gaston, and so on. Generally, for mining graph patterns measured by an objective function F, there are two related mining tasks: (i) enumeration task, find all of subgraphs g such that $F(g)$ is no less than a threshold; and (ii) optimization task, find a subgraph g^* such that

$$g^* = \text{argmax}_g F(g).$$

The enumeration task might encounter the exponential number of patterns as the traditional frequent substructure mining does. To resolve this issue, one may rank patterns according to their objective score and select patterns with the highest value. The feature-based mining framework finds many key applications in chemical data mining including, but not limited to, chemical graph search, classification and clustering.

Chemical graph search aims to find graphs that contain a specific query structure. It is inefficient to scan the whole database and check each graph. Yan et al. [9] applied the feature-based mining framework to support fast search using frequent substructures selected by the following criterion. Let substructures $f_1,f_2,...,f_n$ be selected features. Given a new substructure x, the selectivity power of x can be measured by

$$1 - Pr(x|f_{\varphi_1},...,f_{\varphi_m}), f_{\varphi_i} \subseteq x, 1 \le \varphi_i \le n.$$

which shows the absence probability of x given the presence of $f_{\varphi_1},...,f_{\varphi_m}$ in a graph. When the selectivity is high, substructure x is a good candidate to index.

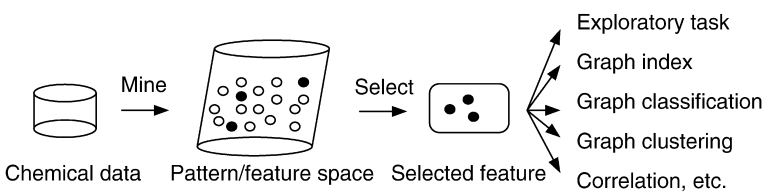
In addition to molecule search, chemical data classification and clustering could also benefit from graph patterns. A typical setting of molecule classification is to induce a mapping $h(\mathbf{g}) : \mathcal{G} \to \{\pm 1\}$ from the training samples $D = \{\mathbf{g}_i,y_i\}_{i=1}^n$, where $\mathbf{g}_i \in \mathcal{G}$ is a labeled graph and $y_i \in \{\pm 1\}$ is the class label. Feature-based classification models were proposed in [8,4]. In these models, graphs are first transformed to vectors using discriminative substructures, which are then processed by standard classification methods.

Key Applications

Chemical Structure Search
Chemical Classification
Chemical Clustering
Quantitative Structure-Activity Relationship Analysis

Experimental Results

PubChem (see dataset URL) provides information on the biological activities of small molecules, containing



Mining of Chemical Data. Figure 1. Feature-based mining framework.

the bioassay records for anti-cancer screen tests with different cancer cell lines. Each dataset belongs to a certain type of cancer screen with the outcome active or inactive. These bioassays are experimented to identify the chemical compounds that display the desired and reproducible behavior against cancers. Chemical compound classification is to computationally predict the activity of untested compounds given the bioassay data. This process can replace or supplement the physical assay techniques. Furthermore, it could also identify substructures that are critical to specific biological or chemical activities.

The following experiment demonstrates the effectiveness of graph kernel method [7,5] (optimal assignment kernel, OA) and pattern-based classification [8,4] (PA), both of which show good accuracy. From the PubChem screen tests, 11 bioassay datasets are displayed. Since the active class is very rare (around 5%) in these datasets, 500 active compounds and 2,000 inactive compounds are randomly sampled from each dataset for performance evaluation. The classification accuracy is evaluated with 5-fold cross validation. For both methods, the same implementation of support vector machine, LIBSVM [2], with parameter C selected from $[2^{-5},2^{5}]$, is used. Table 1 shows AUC by OA and PA. The area under the ROC curve (AUC) is a measure of the model accuracy, in the range of [0,1]. A perfect model will have an area of one. As shown in Table 1, PA achieves comparable results with OA. A detailed examination shows that PA is able to discover substructures that determine the activity of compounds, without domain knowledge.

Mining of Chemical Data. Table 1. Chemical compound classification

Dataset	OA	PA
MCF-7: Breast	0.68 ± 0.12	0.67 ± 0.10
MOLT-4: Leukemia	0.65 ± 0.06	0.66 ± 0.06
NCI-H23: Non-Small Cell Lung	0.79 ± 0.08	0.76 ± 0.09
OVCAR-8: Ovarian	0.67 ± 0.04	0.72 ± 0.06
P388: Leukemia	0.79 ± 0.07	0.82 ± 0.04
PC-3: Prostate	0.66 ± 0.09	0.69 ± 0.09
SF-295: Central Nerv Sys	0.75 ± 0.11	0.72 ± 0.12
SN12C: Renal	0.75 ± 0.08	0.75 ± 0.06
SW-620: Colon	0.70 ± 0.02	0.74 ± 0.06
UACC257: Melanoma	0.65 ± 0.05	0.64 ± 0.05
Yeast: Yeast anticancer	0.64 ± 0.04	0.71 ± 0.05
Average	0.70 ± 0.07	0.72 ± 0.07

Data Sets

PubChem provides bioassay records for anti-cancer screen tests with different cancer cell lines, available at http://pubchem.ncbi.nlm.nih.gov

Cross-references

- ▶ Frequent Graph Patterns
- ▶ Graph Classification
- ▶ Graph-based Clustering
- ▶ Graph Database
- ▶ Graph Database Mining
- ▶ Graph Kernel
- ▶ Graph Search

Recommended Reading

1. Bunke H. and Shearer K. A graph distance metric based on the maximal common subgraph. Pattern Recogn. Lett., 19:255–259, 1998.
2. Chang C.-C. and Lin C.-J. LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
3. Dehaspe L., Toivonen H., and King R. Finding frequent substructures in chemical compounds. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, 1998, pp. 30–36.
4. Deshpande M., Kuramochi M., Wale N., and Karypis G. Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans. Knowl. Data Eng., 17:1036–1050, 2005.
5. Fröhlich H., Wegner J., Sieker F., and Zell A. Optimal assignment kernels for attributed molecular graphs. In Proc. 22nd Int. Conf. on Machine Learning, 2005, pp. 225–232.
6. Hansch C. A quantitative approach to biochemical structure-activity relationships. Acct. Chem. Res., 2:232–239, 1969.
7. Kashima H., Tsuda K., and Inokuchi A. Marginalized kernels between labeled graphs. In Proc. 20th Int. Conf. on Machine Learning, 2003, pp. 321–328.
8. Kramer S., Raedt L., and Helma C. Molecular feature mining in HIV data. In Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2001, pp. 136–143.
9. Yan X., Yu P.S., and Han J. Graph indexing: A frequent structure-based approach. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2004, pp. 335–346.

Mixed Evidence

- ▶ Contextualization

Mixed-Media

► Multimedia Metadata

MM Indexing

► Multimedia Data Indexing

MMDBMS

► Main Memory DBMS

Mobile Ad hoc Network Databases

► MANET Databases

Mobile Database

Ouri Wolfson
University of Illinois at Chicago, Chicago, IL, USA

Definition

A mobile database is a database that resides on a mobile device such as a PDA, a smart phone, or a laptop. Such devices are often limited in resources such as memory, computing power, and battery power.

Key Points

Due to device limitations, a mobile database is often much smaller than its counterpart residing on servers and mainframes. A mobile database is managed by a Database Management System (DBMS). Again, due to resource constraints, such a system often has limited functionality compared to a full blown database management system. For example, mobile databases are single user systems, and therefore a concurrency control mechanism is not required. Other DBMS components such as query processing and recovery may also be limited.

Queries to the mobile database are usually posed by the user of the mobile device. Updates of the database may originate from the user, or from a central server, or directly from other mobile devices. Updates from the server are communicated wirelessly. Such communication takes place either via a point-to-point connection between the mobile device (the client) and the server, or via broadcasting by the server [Acharya S., Franklin M., and Zdonik S, 1995, Dissemination-based Data Delivery Using Broadcast Disks, Personal Communications]. Direct updates from other mobile devices may use short-range wireless communication protocols such as Bluetooth or Wifi [1].

Cross-references

► Mobile Ad hoc Network Databases

Recommended Reading

1. Cao H., Wolfson O., Xu B., and Yin H., "MOBI-DIC: MOBIle DIscovery of loCal Resources in Peer-to-Peer Wireless Network". Bull Comput Soc Tech Committee Data Eng, 28(3):11–18, 2005 (Special Issue on Database Issues for Location Data Management).

Mobile Interfaces

Giuseppe Santucci
University of Rome, Roma, Italy

Synonyms

Handhelds interfaces; Navigation system interfaces

Definition

Mobile interfaces are interfaces specifically designed for little portable electronic devices (handhelds), like cellular phones, personal digital assistants (PDAs), and pagers. Mobile interfaces provide means to execute complex activities on highly constrained devices, characterized by small screens, low computing power, and limited input/output (I/O) capabilities. While this term is mainly used to refer to interfaces for classical Web applications, like email and Web browsing it should be intended in a broader sense, encompassing all Web and non-Web applications that run on little mobile equipments (e.g., GPS navigation systems).

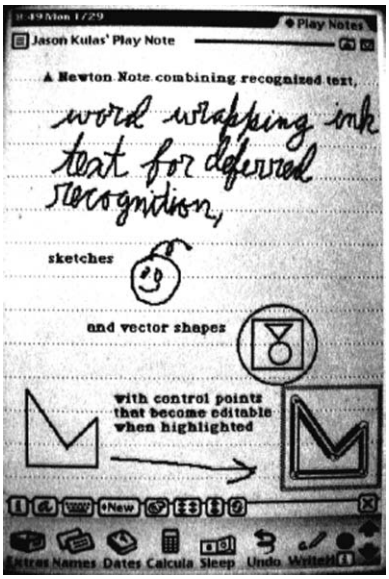
Historical Background

Mobile interfaces came up with the proliferation of cellular phones through the 1980s. At that time, the notion of mobile interface was very primitive and the offered services were very simple: handling lists of contacts, usually limited to <name, phone-number>

pairs, starting/answering a phone call, writing short text, setting phone preferences (volume, light, ring tone, etc.). Even in this restricted scenario, the vendors had to deal with challenging issues, posed by the very limited early cellular phones capabilities, in terms of screen dimension and resolution, limited computing power, and I/O capabilities. No standards were around and different, unrelated solutions were adopted. The most used interaction strategy was to mimic the typical hierarchical computer menus with very poor results: the user was (and still is) forced to access functions and services through a series of boring menus. The main reason of failure is that while PCs can present to the user a complete list of all possible choices mobile phones can show a small number of options at time (usually one), forcing the user to remember the paths to the commands. That results in a very large number of key presses, errors, and mental overwhelming: many of the advantages of the conventional menus are lost [12].

For about ten years the unique available portable devices were mainly cellular phones; from 1993 to 1998 Apple Computer (now Apple Inc.) marketed the "Apple Newton," the first line of PDAs, personal digital assistants, a term introduced on January 7, 1992 by John Sculley at the Consumer Electronics Show in Las Vegas, Nevada, referring to the Apple Newton. The Apple's official name for the PDA was "MessagePad" and it was based on the ARM 610 RISC processor using a dedicated operating system (Newton OS, an allusion to Isaac Newton's apple); however, the word Newton was popularly used to refer to the device and its software. The new device was characterized by a wider touchable monochromatic screen (366 × 240) with retro illumination (only for the top version). The increased computational power and the larger screen allowed for very innovative interaction techniques (for a portable device): icons, simulated touchable QWERTY keyboard, and handwriting recognition (called Calligrapher). Moreover, the user was allowed to turn the screen horizontally ("landscape") as well as vertically ("portrait") preserving the handwriting recognition functionality (see Fig. 1).

Even the data management was quite innovative: programs were able to convert and share data (e.g., the calendar was able to refer to names in the address book or to convert a note into an appointment). Finally all the devices were equipped with a built in infrared and an expansion port for connecting to a modem or Ethernet. Disregarding the color absence and the limited device connectivity, the Palm Newton functionalities were quite similar to the ones of modern PDAs!



Mobile Interfaces. Figure 1. A screenshot from Apple Newton.

In spite of the innovative interface and capabilities, the product was not a successful one. The main reasons were the poor handwriting recognition accuracy, the high price, and the non comfortable size (it did not fit in a regular pocket). Still, the pioneering Apple ideas were captured by other vendors and many similar devices were around in the next years. Among them, the Palm series with the PalmOS operative system and the Graffiti handwriting recognition system conquered (in 1999) about 80% of the world market, mostly for the effective handwriting system and for the availability of a huge amount of third part software. On February 2000 Palm marketed the PalmIIIc, the first color PDA. Concerning the interface it is worth noting the different Graffiti philosophy: instead of allowing the user to write in his/her own style it forced the user to learn a set of pen strokes for each character (see Fig. 2). This narrowed the possibility for erroneous input, although memorization of the stroke patterns did increase the learning curve for the user. Some studies demonstrated that even if the error rate with Graffiti was higher than using the virtual keyboard (19.3 vs. 4.1%), users prefer Graffiti, because its usage is more natural.

The graffiti® alphabet

Letter	Strokes	Letter	Strokes
A		N	
B		O	
C		P	
D		Q	
E		R	
F		S	
G		T	
H		U	
I		V	
J		W	
K		X	
L		Y	
M		Z	

Mobile Interfaces. Figure 2. The PalmOS Graffiti.

As soon as the hardware made it possible a deeper integration between PDA and cellular phones started: phones were equipped with larger screen and complex applications (smartphones), while PDA included phone hardware. Moreover, the so called pager devices, mainly intended for on the way email, came up. In 2004, smartphones were outselling PDAs for the first time. Nowadays, the difference between these three categories in term of capabilities and interface is very little. The main consequence of this integration is that all these devices allow for Internet access (through GPRS or WI-FI) and it is very likely that in the next future there will be more people accessing the Internet via mobile devices (phones, PDA, etc) than via conventional PCs. According to this issue many researchers are now dealing with the problem of designing friendly interface for accessing Internet through portable devices.

Foundations

The research effort concerning mobile interfaces followed two main paths: designing interface for a specific application running on a specific device (e.g., the agenda interface of a specific cellular phone) or designing web based applications that can be deployed on multiple devices (e.g., a web browser working on different PDAs and cellular phones).

Mobile Interface for Specific Devices

The problem of dealing with little screen is not new. Much literature from the 1980s and early 1990s, written even before the Web, deals with user interfaces on small screens. At that time the focus was on first generation cash dispensing ATMs, electronic typewriters and photocopiers. All of these systems could display only a limited number of text lines to communicate with the user. Research looked into the impact of reduced screen size on comprehension [3], reading rate [4], and interaction [12]. This research is still valid and has new relevance to mobile Web devices [8].

The availability of graphic display and the diffusion of navigation system raised the issue of interacting with graphics and maps [6]. Moreover, the increasing usage of such systems while driving a car posed several new issues concerning distracting factors and security [1].

Multi Target Applications

In this case, research deals with techniques able to support the design of applications that run on several devices that present different interaction capabilities (e.g., small/large screens, keyboard/keypad input, etc.). In order to address this issue, researchers have adopted two main approaches:

1. Designing applications in order to let them run on different platforms
2. Adapting existing systems in order to render them usable on platform that were not originally included as service provider (e.g., standard web sites)

In the first approach the solution is offered at design time, that is, the designer explicitly knows that the system will run on multiple devices. Some research exists that follows this approach and the key issues consist in giving models, methods, and tools to support the designer in the creation of multi-device applications and in offering techniques and algorithms to generate the final interface at run time for the specific

device utilized. In [10], a tool for support is presented together with a well defined development life cycle: the designer starts from abstract tasks definition and upon it designs the interface in abstract. The final interfaces are statically generated at design time by following suggestions offered by the system and further refined in order to accommodate specific details. Puerta et al. in [5] propose a similar framework, defining a model to design user interfaces in abstract as well, but their work envisions also adaptive techniques to produce the final user interface. A central mediator agent is responsible for translating the abstract specification of the user interface, according to a description of device's characteristics, into the final interface. Fundamental questions arising from this kind of research are what kind of models should be employed to specify abstract interaction elements and what strategies/techniques can permit an effective translation of abstract models into real interfaces.

The second approach is the one that attempts to bring existing systems, typically web sites, to small screen devices by means of some sort of adaptation/filtering. Among them, the WAP forum (www.wapforum.org) defined a protocol that enables Web-like services on little, portable devices. The interface uses a card metaphor and the designers claim that it is highly appropriate and usable. However, Nielson [11] raised doubts about its effectiveness. A similar proposal is i-mode, developed by the dominant Japanese carrier NTT DoCoMo, and widely used in Japan. Moreover, the World Wide Web Consortium (W3C) is also developing a framework that will enable Web content to be accessed on a diverse range of devices [13]. This framework will allow a document to exist in multiple variants, each variant specifying the type of support needed by the browser to display its contents. Device capabilities and user preferences will also be captured. The information about documents, devices and users will be used to automatically adapt a Web page to best suit the device and user.

Key Applications

Users of mobile interfaces are growing at high speed. According to IDC's Worldwide Quarterly Mobile Phone Tracker (www.idc.com), worldwide mobile phone shipments rose 19.1% year over year and increased sequentially 8.8% in 2005 to reach 208.3 million units. While unconnected PDA's are a declining segment, the emerging navigator market is still increasing about 10 million cars guided by navigation systems, 43 million dedicated portable navigators, and 16 millions smart phones based navigators forecasted for 2010 (www.strategyanalytics.net).

These figures make clear the importance of mobile interfaces in the coming years.

Future Directions

There are some emerging issues that will likely affect the behavior and structure of mobile interfaces:

- Search versus menu browsing. Several proposals relies on the idea that, in order to start an action, it is better to search it (i.e., to write down the command name) instead of browsing boring menus [9,7].
- Multimodal interaction. When the hardware allows it, mobile interfaces can exploit multimodal input/output (e.g., voice commands, likely together with the aforementioned search strategy).
- Context and user modeling. Some proposal describe interfaces that can adapt themselves according to the user preferences and the context [2], e.g., an interface for browsing tourist information presents a user with list of vegetarian restaurants at walking distance, according to the user location, discovered through the integrated GPS device, knowing that the user is vegetarian and that s/he is currently walking.

These research issues could greatly improve the mobile interfaces in the next years.

Cross-references

- ▶ Multimodal Interfaces
- ▶ Visual Interfaces
- ▶ WIMP Interfaces

Recommended Reading

1. Commission of the European Communities. Commission recommendation of 22 December 2006 on safe and efficient in-vehicle information and communication systems: Update of the European Statement of Principles on Human Machine Interface, 2006.
2. Coutaz J., Crowley J., Dobson S., and Garlan D. Context is key. Commn. ACM, 48(3):49–53, 2005.
3. Dillon A., Richardson J., and McKnight. The effect of display size and text splitting on reading lengthy text from the screen, Behav. Inf. Technol., 9(3):215–227, 1990.
4. Duchnicky R.L. and Kolers P.A. Readability of text scrolled on visual display terminals as a function of window size. Hum. Factors, 25(6):683–692, 1983.
5. Eisenstein J., Vanderdonckt J., and Puerta A. Applying model-based techniques to the development of UIs for mobile

computers. In Proc. Sixth Int. Conf. on Intelligent User Interfaces, 2001, pp. 69–76.
6. Frey P.R., Rouse W.B., and Garris R.D. Big graphics and little screens: designing graphical displays for maintenance tasks. IEEE Trans. Syst. Man Cybernetics, 22(1): 10–20, 1992.
7. Graf S., Spiessl W., Schmidt A., Winter A., and Rigoll G. In-car interaction using search based user interfaces. In Proc. 26th Annual SIGCHI Conf. on Human factors in Computing Systems. 2008, pp. 1685–1688.
8. Jones M., Marsden G., Mohd-Nasir N., Boone K., and Buchanan G. Improving web interaction on small displays. In Proc. W8 Conf., Toronto, 1999 Also reprinted in Int. J. Comput. Telecommn. Netw., 31(11–16):1129–1137, 1999.
9. Marsden G., Gillary P., Jones M. and Thimbleby H. Successful user interface design from efficient computer algorithms. In Proc. ACM CHI 2000 Conf. on Human Factors in Computing Systems, 2000, pp. 181–182.
10. Mori G., Paternò F., and Santoro C. Tool support for designing nomadic applications. In Proc. 2003 Int. Conf. on Intelligent User Interfaces, 2003, pp. 141–148.
11. Nielson J. Graceful degradation of scalable internet services. Available online at:http://www.useit.com/alertbox/991031.html, 1999.
12. Swierenga S.J. Menuing and scrolling as alternative information access techniques for computer systems: interfacing with the user. In Proc. 34th Annual Meeting of Human Factors Society, 1990, pp. 356–359.
13. W3C Mobile Activity Statement. Available online at: http://www.w3.org/Mobile/Activity.

Mobile Map Services

► Location-Based Services (LBS)

Mobile Sensor Network Data Management

Demetrios Zeinalipour-Yazti[1]
Panos K. Chrysanthis[2]
[1]University of Cyprus, Nicosia, Cyprus
[2]University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

MSN data management; Mobile wireless sensor network data management

Definition

Mobile Sensor Network (MSN) Data Management refers to a collection of centralized and distributed algorithms, architectures and systems to handle (store, process and analyze) the immense amount of spatio-temporal data that is cooperatively generated by collections of sensing devices that move in space over time.

Formally, given a set of n homogenous or heterogeneous mobile sensors $\{s_1, s_2,...,s_n\}$ that are capable of acquiring m physical attributes $\{a_1, a_2,...,a_m\}$ from their environment at every discrete time instance t (i.e., datahas a temporal dimension), an implicit or explicit mechanism that enables each s_i $(i \leq n)$ to move in some multi-dimensional Euclidean space (i.e., data has one or more spatial dimensions), MSN Data Management provides the foundation to handle spatio-temporal data in the form $(s_i, t, x, [y, z,]a_1[,...,a_m])$, where x, y, z defines three possible spatial dimensions and the bracket expression "[ ]" denotes the optional arguments in the tuple definition. In a more general perspective, MSN Data Management deals with algorithms, architectures and systems for in-network and out-of-network query processing, access methods, storage, data modeling, data warehousing, data movement and data mining.

Historical Background

The improvements in hardware design along with the wide availability of economically viable embedded sensor systems have enabled scientists to acquire environmental conditions at extremely high resolutions. Early approaches to monitor the physical world were primarily composed of passive sensing devices, such as those utilized in wired weather monitoring infrastructures, that could transmit their readings to more powerful processing units for storage and analysis. The evolution of passive sensing devices has been succeeded by the development of *Stationary Wireless Sensor Networks (Stationary WSNs)*. These are composed of many tiny computers, often no bigger than a coin or a credit card, that feature a low frequency processor, some flash memory for storage, a radio for short-range wireless communication, on-chip sensors and an energy source such as AA batteries or solar panels. Applications of stationary WSNs have emerged in many domains ranging from environmental monitoring [15] to seismic and structural monitoring as well as industry manufacturing.

The transfer of information in such networks is conducted without electrical conductors (i.e., wires) using technologies such as radio frequency (RF), infrared light, acoustic energy and others, as the mobility aspect inherently hinders the deployment of any

technology that physically connects nodes with wires. Since communication is the most energy demanding factor in such networks, data management researchers have primarily focused on the development of energy-conscious algorithms and techniques.

In particular, declarative approaches such as TinyDB [9] and Cougar [16] perform a combination of in-network aggregation and filtering in order to reduce the energy consumption while conveying data to the *querying node (sink)*. Additionally, approaches such as TiNA [13] and MINT Views [17] take into account intelligent in-network data reduction techniques to further reduce the consumption of energy. *Data Centric Routing* approaches, such as directed diffusion [8], establish low-latency paths between the sink and the sensors in order to reduce the cost of communication. *Data Centric Storage* [14] schemes organize data with the same attribute (e.g., humidity readings) on the same node in the network in order to offer efficient location and retrieval of sensor data.

The evolution of stationary WSNs in conjunction with the advances made by the distributed robotics and low power embedded systems communities have led to a new class of *Mobile (and Wireless) Sensor Networks (MSNs)* that can be utilized for land [3,5,10], ocean exploration [11], air monitoring [1], automobile applications [7,6], Habitant Monitoring [12] and a wide range of other scenarios. MSNs have a similar architecture to their stationary counterparts, thus are governed by the same energy and processing limitations, but are supplemented with implicit or explicit mechanisms that enable these devices to move in space (e.g., motor or sea/air current) over time. Additionally, MSN devices might derive their coordinates through absolute (e.g., dedicated Geographic Positioning System hardware) or relative means (e.g., *localization techniques*, which enable sensing devices to derive their coordinates using the signal strength, time difference of arrival or angle of arrival). There are several classes of MSNs which can coarsely be structured into the following classes: (i) highly mobile, which contains scenarios in which devices move at high velocities such as cars, human with cell phones, airplanes, and others; (ii) mostly static, which contains scenarios in which devices move at low velocities such as monitoring sensors in a shop floor with moving robots; and (iii) hybrid, which contains both classes such as an airplane that has sensors installed on inside and outside.

Foundations

The unique characteristics of MSNs create novel data management opportunities and challenges that have not been addressed in other contexts including those of mobile databases and stationary WSNs. In order to realize the advantages of such networks, researchers have to re-examine existing data management and data processing approaches in order to consider sensor and user mobility; develop new approaches that consider the impact of mobility and capture its trade-offs. Finally, MSN data management researchers are challenged with structuring these networks as huge distributed databases whose edges consist of numerous "receptors" (e.g., RFID readers or sensor networks) and internal nodes form a pyramid scheme for (in-network) aggregation and (pipelined) data stream processing.

There are numerous advantages of MSNs over their stationary counterparts. In particular, MSNs offer: (i) *dynamic network coverage*, by reaching areas that have not been adequately sampled; (ii) *data routing repair*, by replacing failed routing nodes and by calibrating the operation of the network; (iii) *data muling*, by collecting and disseminating data/readings from stationary nodes out of range; (iv) *staged data stream processing*, by conducting in-network processing of continuous and ad-hoc queries; and (v) *user access points*, by enabling connection to handheld and other mobile devices that are out of range from the communication infrastructure.

These advantages enable a wide range of new applications whose data management requirements go beyond those of stationary WSNs. In particular, MSN system software is required to handle: (i) *the past*, by recording and providing access to history data; (ii) *the present*, by providing access to current readings of sensor data; (iii) *the future*, by generating predictions; (iv) *distributed spatio-temporal data*, by providing new means of distributed data storage, indexing and querying of spatio-temporal data repositories; (v) *data uncertainty*, by providing new means of handling real world signals that are inherently uncertain; (vi) *self-configurability*, by withstanding "harsh" real-life environments; and (vii) *data and service mash-ups*, by enabling other innovative applications that build on top of existing data and services.

In light of the above characteristics, the most predominant data management challenges that have prevailed in the context of MSNs include:

In-Network Storage: The absence of a stationary network structure in MSNs makes continuous data acquisition to some sink point a non-intuitive task (e.g., mobile nodes might be out of communication range from the sink). In particular, the absence of an always accessible sink mandates that acquisition has to be succeeded by in-network storage of the acquired events so that these events can later be retrieved by the user. Mobile devices usually utilize flash memory as opposed to magnetic disks, which are not shock-resistant and thus are not appropriate for a mobile setting. Consequently, a major challenge in MSNs is to extend local storage structures and access methods in order to provide efficient access to the data stored on the local flash media of a sensor device while traditional database research has mainly focused on issues related to magnetic disks.

Flexible and Expressive Query Types: In a traditional database management system, there is a single correct answer to a given query on a given database instance. When querying MSNs the situation is notably different as there are many more degrees of freedom and the underlying querying engine needs to be guided regarding which alternative execution strategy is the right one, typically on the basis of target answer quality and resource availability. In this context, there are additional relevant parameters that include: (i) *Resolution:* physical sensor data can be observed at multiple resolutions along space and time dimensions; (ii) *Confidence:* more often than not, correctness of query results can be specified only in probabilistic terms due to the inherent uncertainty in the sensor hardware and the modeling process; (iii) *Alternative models:* in some cases, several alternative models apply to a single scenario. Each alternative typically represents a different point in the efficiency (resource consumption) and effectiveness (result quality) spectrum, thereby allowing a tradeoff between these two metrics on the basis of application-level expectations. The prime challenge is to define new declarative query languages that make use of these new parameters while allowing a highly flexible and optimizable implementation. Additionally, approximate query processing with controlled result accuracy becomes vital for dynamic mobile environments with varying node velocities, changing data traffic patterns, information redundancy, uncertainty, and inevitable flexible load shedding techniques. Finally, in order to have an efficient and optimized implementation of query types, MSNs will need to consider cross-layer optimization since all layers of the data stack are involved in query execution.

Efficient Query Routing Trees: Query routing and resolution in stationary WSNs is typically founded on some type of query routing tree that provides each sensor with a path over which answers can be transmitted to the sink. In a MSN, such a query routing tree can neither be constructed in an efficient manner nor be maintained efficiently as the network topology is transient. The dynamic nature of the underlying physical network tremendously complicates the interchange of information between nodes during the resolution of a query. In particular, it is known that sensing devices tend to power-down their transceiver (transmitter-receiver) during periods of inactivity in order to conserve energy [2]. While stationary WSNs define transceiver scheduling approaches, such as those defined in TAG [9], Cougar [16] and MicroPulse [1], in order to enable accurate transceiver allocation schemes, such approaches are not suitable for mobile settings in which a sensor is not aware of its designated parent node in the query tree hierarchy. Consequently, nodes are not able to agree on rendezvous time-points on which data interchange can occur.

Purpose-Driven Data Reduction: The amount of data generated from MSNs can be overwhelming. Consequently, a main challenge is to provide data reduction techniques which will be tuned to the semantics of the target application. Furthermore, data reduction must take into account the entire spectrum of uses, ranging from real-time to off-line, supporting both snapshot and continuous queries that take advantage of designated optimization opportunities (e.g., multi-query) especially targeted for mobile environments. Finally, it must also consider the inherently dynamic aspects of these environments and the possibility of in-network data reduction (e.g., in-network aggregation).

Perimeter Construction and Swarm-Like Behavior: In many types of MSNs, new events are more prevalent at the periphery of the network (e.g., water detection and contamination detection) rather than uniformly throughout the network (which is more typically for applications like fire detection). This creates the necessity to construct the perimeter of a MSN in an online and distributed manner. Additionally, many types of MSNs are expected to feature a swarm-like behavior (The term *Swarm (or Flock)* refers to a group of objects that exhibit a polarized, non-colliding and aggregate

motion.). For instance, consider a MSN design that consists of several rovers that are deployed as a swarm in order to detect events of interest (e.g., the presence of water) [18]. The swarm might collaboratively collect spatio-temporal events of interest and store them in the swarm until an operator requests them. In order to increase the availability of the detected answers, in the presence of unpredictable failures, individual rovers can perform replication of detected events to neighboring nodes. That creates challenges in data aggregation, data fusion and data storage that have not yet been addressed.

Enforcement of Security, Privacy and Trust: Frequent node migrations and disconnections in MSNs, as well as resource constraints raise severe concerns with respect to security, privacy and trust. Additionally, the cost of traditional secure data dissemination approaches (e.g., using encryption) may be prohibitively high in volatile mobile environments. As such, research on encryption-free data dissemination strategies becomes very relevant here. This includes strategies to deliver separate and under-defined data shares, secure multiparty computation and advanced information recovery techniques.

Context-Awareness and Self-Everything: Providing a useful level of situational awareness in an unobtrusive way is crucial to the success of any application utilizing MSNs as this can be used to improve functionality by including preferences from the users but can also be used to improve performance (e.g., better network routing decisions if the exact topology is known). Note that context is often obvious in stationary WSN deployments (i.e., a specific sensor is always in the same location) but in the context of a MSN additional data management measures need to be taken into account in order to enable this parameter. Additionally, it is crucial for them to be "plug-and-play" and self-everything (i.e., self-configurable and self-adaptive) as application deployment of sensors in the field is famously hard, even without the mobility aspect which is introducing additional challenges. Finally, a crucial parameter is that of being adaptive both in how to deal with the system issues (i.e., how to adapt from failures in network connectivity) and also with user-interface/application issues (i.e., how to adapt the application when the context changes).

Key Applications

MSN Data Management algorithms, architectures and systems will play a significant role in the development of future applications in a wide range of disciplines including the following:

Environmental and Habitant Monitoring: A large class of MSN applications have already emerged in the context of environmental and habitant monitoring systems. Consider an ocean monitoring environment that consists of *n* independent surface drifters floating on the sea surface and equipped with either acoustic or radio communication capabilities. The operator of such a MSN might seek to answer queries of the type: "*Has the MSN identified an area of contamination and where exactly?*". The MSN architecture circumvents the peculiarities of individual sensors, is less prone to failures and is potentially much cheaper. Similar applications have also emerged with MSNs of car robots, such as CotsBots [3], Robomotes [5] or Millibots [10], and MSNs of Unmanned Aerial Vehicles (UAVs), such as SensorFlock [1], in which devices can fly autonomously based on complex interactions with their peers. One final challenging application in this class is that of detecting a phenomenon that itself is mobile, for example a brush fire which is being carried around by high winds.

Intelligent Transportation Systems: Sensing systems have been utilized over the years in order to better manage traffic with the ultimate goal of reducing accidents and minimizing the time and the energy (gasoline) wasted while staying idle in traffic. Since cars are already equipped with a wide range of sensors, the generated information can be shared in a vehicle-to-vehicle network. For example the ABS system can detect when the road is slippery or when the driver is hitting the brakes thus this information can be broadcasted to the surrounding cars but also to the many cars back and forth, as needed, in order to make sure that everybody can safely stop with current weather conditions and car speeds.

Medical Applications: This class includes applications that monitor humans in order to improve living conditions and in order to define early warning systems that identify when human life is at risk. For instance, Nike+ is an example for monitoring the health of a group of runners that have simple sensing devices embedded in their running shoes. Such an application would require embedded storage and retrieval techniques in order to administer the local amounts of data. Applications in support of the elderly and those needing constant supervision (e.g., due to chronic diseases like diabetes, allergies, etc.) are another example in which MSN data management

techniques will play an important role. Wellness applications could also be envisioned, where a health "dose" of exercise is administered according to ones needs and capabilities. Another area are systems to protect soldiers on the battlefield. SPARTNET has recently developed wearable physiological sensor systems that collect, organize and interpret data on the health status of soldiers in order to improve situational and medical awareness during field trainings. Such systems could be augmented with functionality of detecting and reporting threats that are either derived from individual signals (e.g., when a soldiers personal health monitor shows erratic life-signals) and from correlated signals that are derived from multiple sensors/soldiers (e.g., by recognizing when a small group of soldiers is deviating away from the expected formation). Finally, disaster and emergency management are another prime area where MSN data management techniques will play a major impact.

Location-Based Services and the Sensor Web: The last group of challenging motivating applications is that of real-time location-based services, for example a service that can report whether there are any available parking spaces or a service that can keep track of buses moving and report how delayed a certain bus is. Many of these services become more powerful with the integration of data from the *Sensor Web* (i.e., live sensor data) with the *Web* (i.e., static content available online) and the *Deep-web* (i.e., data that is stored in a database, but are accessible through a web page or a web service).

Cross-references

- ▶ Mobile and Ubiquitous Data Management
- ▶ Sensor Networks
- ▶ Spatial Network Databases
- ▶ Stream Data Analysis

Recommended Reading

1. Allred J., Hasan A.B., Panichsakul S., Pisano B., Gray P., Huang J-H., Han R., Lawrence D., and Mohseni K. SensorFlock: an airborne wireless sensor network of micro-air vehicles. In Proc. 5th Int. Conf. on Embedded Networked Sensor Systems, 2007, pp. 117–129.
2. Andreou P., Zeinalipour-Yazti D., Chrysanthis P.K., and Samaras G. Workload-aware optimization of query routing trees in wireless sensor networks In Proc. 9th Int. Conf. on Mobile Data Management, 2008, pp. 189–196.
3. Bergbreiter S. and Pister K.S.J. CotsBots: an off-the-shelf platform for distributed robotics. In Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2003, pp. 1632–1637.
4. Chintalapudi K. and Govindan R. Localized Edge Detection in Sensor Fields. Ad-hoc Networks, 1(2–3):273–291, 2003.
5. Dantu K., Rahimi M.H., Shah H., Babel S., Dhariwal A., and Sukhatme G.S. Robomote: enabling mobility in sensor networks. In Proc. 4th Int. Symp. on Information Processing in Sensor Networks, 2005, pp.
6. Eriksson J., Girod L., Hull B., Newton R., Madden S., and Balakrishnan H. The Pothole Patrol: using a mobile sensor network for road surface monitoring. In Proc. 6th Int. Conf. Mobile Systems, Applications and Services, 2008, pp. 29–39.
7. Hull B., Bychkovsky V., Chen K., Goraczko M., Miu A., Shih E., Zhang Y., Balakrishnan H., and Madden S. CarTel: a distributed mobile sensor computing system. In Proc. 4th Int. Conf. on Embedded Networked Sensor Systems, 2006, pp. 125–138.
8. Intanagonwiwat C., Govindan R., and Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In Proc. 6th Annual Int. Conf. on Mobile Computing and Networking, 2000, pp. 56–67.
9. Madden S.R., Franklin M.J., Hellerstein J.M., and Hong W. The design of an acquisitional query processor for sensor networks. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003.
10. Navarro-Serment L.E., Grabowski R., Paredis C.J.J., and Khosla P.K. Millibots: the development of a framework and algorithms for a distributed heterogeneous robot team. IEEE Robot. Autom. Mag., 9(4), December 2002.
11. Nittel S., Trigoni N., Ferentinos K., Neville F., Nural A., and Pettigrew N. A drift-tolerant model for data management in ocean sensor networks. In Proc. 6th ACM Int. Workshop on Data Eng. for Wireless and Mobile Access, 2007, pp. 49–58.
12. Sadler C., Zhang P., Martonosi M., and Lyon S. Hardware design experiences in ZebraNet. In Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems, 2004, pp. 227–238.
13. Sharaf M., Beaver J., Labrinidis A., and Chrysantrhis P.K. Balancing energy efficiency and quality of aggregate data in sensor networks. VLDB J., 13(4):384–403, 2004.
14. Shenker S., Ratnasamy S., Karp B., Govindan R., and Estrin D. Data-centric storage in sensornets. SIGCOMM Comput. Commun. Rev., 33(1):137–142, 2003.
15. Szewczyk R., Mainwaring A., Polastre J., Anderson J., and Culler D. An analysis of a large scale habitat monitoring application. In Proc. 2nd Int. Conf. on Embedded Networked Sensor Systems, 2004, pp. 214–226.
16. Yao Y. and Gehrke J.E. The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Rec., 32(3):9–18, 2002.
17. Zeinalipour-Yazti D., Andreou P., Chrysanthis P., and Samaras G. MINT views: materialized in-network top-k views in sensor networks. In Proc. Int. Conf. on Mobile Data Management, 2007, pp. 182–189.
18. Zeinalipour-Yazti D., Andreou P., Chrysanthis P., and Samaras G. SenseSwarm: a perimeter-based data acquisition framework for mobile sensor networks. In Proc. VLDB Workshop on Data Management for Sensor Networks, 2007, pp. 13–18.

Mobile Wireless Sensor Network Data Management

▶ Mobile Sensor Network Data Management

Model Management

Christoph Quix
RWTH Aachen University, Aachen, Germany

Definition

Model management comprises technologies and mechanisms to support the integration, transformation, evolution, and matching of models. It aims at supporting metadata-intensive applications such as database design, data integration, and data warehousing. To achieve this goal, a model management system has to provide definitions for *models* (i.e., schemas represented in some metamodel), *mappings* (i.e., relationships between different models), and *operators* (i.e., operations that manipulate models and mappings). Model management has become more and more important, since the interoperability and/or integration of heterogeneous information systems is a frequent requirement of organizations. Some important operations in model management are *Merge* (integration of two models), *Match* (creating a mapping between two models), and *ModelGen* (transforming a model given in one modeling language into a corresponding model in a different modeling language).

The current understanding of model management has been defined in [4] and focuses mainly on (but is not limited to) the management of *data* models. Most of the problems mentioned before have been already addressed separately and for specific applications. The goal now is to build a model management system (MMS) which unifies the previous approaches by providing a set of *generic* structures representing models and mappings, and the definition of *generic* operations on these structures. Such a system could then be used by an application to solve model management tasks.

Historical Background

Model management, as it is understood today, has been defined by Bernstein et al. [4]. In the 1980s, the term "model management" was used in the context of decision support systems, but this refered mainly to mathematical models. Dolk [7] stated first the requirement for a theory for models similar to the relational database theory. Such a theory should include formal definitions of models and operations on models and could be used as a basis for the implementation of a model management system. This work was based on a draft of the *Information Resource Dictionary System* (IRDS) standard (ISO/IEC 10027:1990) which was accepted in 1990. The IRDS standard clarified the terminology of modeling systems and defined a framework structure for such systems as a four-level hierarchy: at the lowest level reside data instances which are described by a model (or schema) on the next higher level. This model is expressed in some modeling language (or metamodel) which is located at the third level. The highest level contains a metametamodel which can be used to define metamodels.

The new definition of model management in 2000 [4] integrated the research efforts of several previously loosely coupled areas. Therefore, research in model management did not start from scratch, rather it could build already on many results such as schema integration [3], or model transformation [1]. The main contribution of [4] was the definition of operations (such as Match, Merge, Compose) which a model management system should offer. Furthermore, it was required that models and mappings are considered as first class objects and that operations should address them as a whole and not only one model element at a time.

Since the vision of model management has been stated, research diverted into the areas of schema matching [14], model transformation [1], generic metamodels [10], schema integration [3], and the definition and composition of mappings [8]. Each of these areas will be summarized briefly in the next section.

Recently, research on model management has been summarized in [5]. It was also emphasized that more expressive mapping languages are required than proposed in the original vision of model management. Model management systems should also include a component in which the mappings can be executed.

Foundations

Schema Matching

Schema matching is the task of identifying a set of correspondences (also called a morphism or a mapping) between schema elements. Many aspects have to be considered during the process of matching, such as data values, element names, constraint information, structure information, domain knowledge, cardinality relationships, and so on. All this information is useful in understanding the semantics of a schema, but it can be a very time consuming problem to collect this

information. Therefore, automatic methods are required for schema matching.

A multitude of methods have been proposed for schema matching [14] using different types of information to identify similar elements. The following categories of schema matchers are frequently used:

- Element-Level Matchers take only the information of one schema element separately into account. These can either use linguistic information (name of the element) or constraint information (data type, key constraints).
- Structure-Level Matchers use graph matching approaches to measure the similarity of the structures implied the schema.
- Instance-Level Matchers use also data instances to match schema elements. If the instance sets of two elements are similar, or have a similar value distribution, this might indicate a similarity of the schema elements.
- Machine-Learning Matchers use either instance data or previously identified matches as training data for a machine-learning system. Based on this training data, the system should then detect similar matches in new schema matching problems.

It has been agreed that no single method can solve the schema matching problem in general. Therefore, matching frameworks have been developed which are able to combine multiple individual matching methods to achieve a better result.

Model Transformation

Model transformation (also called *ModelGen*) is the task of transforming a given model M in some particular modeling language into a corresponding model M' in some other modeling language. A classical example for such a transformation is the transformation of an entity-relationship model into a relational database schema. In general, the transformation cannot guarantee that all semantic information of M is still present in the resulting model M' as the target modeling language might have limited expressivity.

Whereas transformation of models between different modeling languages is a frequent task, it has up to now mainly been adressed in specialized settings which map from one particular metamodel to another fixed metamodel. Recent approaches for generic model transformations are based on generic model representations and use a rule-based system to transform a generic representation of the original model into a different model in the generic metamodel. As a side effect, these approaches generate also instance-level mappings which are able to transform data conforming to the original model into data of the generated model.

Another important application area of model transformation is MDA (Model Driven Architecture). The MDA concept is based on the transformation of abstract, conceptual models into more concrete implementation-oriented models.

Generic Metamodel

A generic representation of models is a prerequisite for building a model management system. Without a generic representation, model operations would have to be implemented for each modeling language that should be supported by the system. Especially for the task of model transformation, a generic representation of models is advantageous as the necessary transformations have just to be implemented for the generic representation.

Such a generic representation is called a *generic metamodel*. A generic metamodel should be able to represent models originally represented in different metamodels (or modeling languages) in a generic way without loosing detailed information about the semantics of the model.

First implementations of model management systems used rather simple graph representations of models, e.g., Rondo [13]. Although the graph-based approach might allow an efficient implementation of operations which do not rely on a detailed representation of the models (such as schema matching), it is more difficult to implement more complex operations (such as model transformation or schema integration).

Schema integration approaches used rather abstract generic metamodels. More detailed generic metamodels have been used for model transformation. In [1], the authors describe a metamodel consisting of "superclasses" of the modeling constructs in the native metamodels. The transition between this internal representation and a native metamodel is described as a set of patterns. This induces the concept of a *supermodel* which is the union of patterns defined for any supported native metamodel.

A detailed generic metamodel called *GeRoMe* (Generic Role based Metamodel) is proposed in [10].

GeRoMe employs the *role based* modeling approach in which an object is regarded as playing roles in collaborations with other objects. This allows to describe the properties of model elements as accurately as possible while using only metaclasses and roles from a relatively small set. Therefore, *GeRoMe* provides a generic, yet detailed representation of data models originally represented in different metamodels.

Schema Integration

In model management, the *Merge* operator addresses the problem of schema integration, i.e., generating a merged model given two input models and a mapping between them. The merged model should contain all the information contained in the input models and the mapping. In this context, a mapping is not just a simple set of correspondences between model elements; it might have itself a complex structure and is therefore often regarded also as a *mapping model*. A mapping model is necessary because the models to be merged also have complex structures, which usually do not correspond to each other; the mapping model then acts as a "bridge" to connect these heterogeneous structures.

These structural heterogeneities are one class of conflicts which have to be solved in schema integration. Other types of conflicts are semantic conflicts (model elements describe overlapping sets of objects), descriptive conflicts (the same elements are described by different sets of properties; this includes also name conflicts), and heterogeneity conflicts (models are described in different modeling languages) [15]. The resolution of these conflicts is the main problem in schema integration.

The problem of schema integration has been addressed for various metamodels, such as variants of the ER metamodel [15] or generic metamodels.

Mappings

Depending on the application area, such as data translation, query translation or model merging, schema mappings come in different flavors. One can distinguish between correspondences, also called morphisms, extensional and intensional mappings. Correspondences usually do not have a formal semantics but only state informally that the respective model elements are similar. Morphisms are often – as the result of a schema matching operation [14] – the starting point for specifying more formal mappings. Intensional mappings are usually used for schema integration (see above) and are based on the *possible* instances of a schema. In contrast to intensional mappings, extensional mappings refer to the actual instances of a schema.

Extensional mappings are defined as local-as-view (LAV), global-as-view (GAV), source-to-target tuple generating dependencies (s-t tgds), [12], second order tuple generating dependencies (SO tgds) [8], or similar formalisms. Each of these classes has certain advantages and disadvantages when it comes to properties such as composability, invertibility or execution of the mappings.

Composition is an important sub-problem when dealing with extensional mappings. In general, the problem of composing mappings has the following definition: given a mapping M_{12} from model S_1 to model S_2, and a mapping M_{23} from model S_2 to model S_3, derive a mapping M_{13} from model S_1 to model S_3 that is equivalent to the successive application of M_{12} and M_{23} [8].

Fagin et al. [8] explored the properties of the composition of schema mappings specified by a finite set of s-t tgds. They proved that the language of s-t tgds is not closed under composition. To ameliorate the problem, they introduced the class of SO tgds which are closed under composition.

Model Management Systems

In the recent years, several prototype systems related to model management have been developed. Many of them focus only on some particular aspects of model management whereas only a few try to address a broader range of model management operators. The systems Rondo [13] and GeRoMeSuite [11] aim at providing a complete set of model management operators and are not restricted to particular modeling languages. Clio [9] focuses especially on mappings between XML and relational databases, the generation, and composition of these mappings. COMA++ [2] provides schema matching functionality.

Key Applications

Model management can be applied in scenarios, in which the management of complex data models is necessary. For example, data warehouses (DWs) are one application area for model management systems [6]. For integrating a new data source into the DW, a *Match* operator could be used to identify the similarities between the source schema and the DW schema. If the DW schema has not yet been created, a *Merge*

operator can be used to generate it from several source schemas. The composition of mappings can be used after a source schema S has been evolved to a new version S': if a mapping from S' to S is known (either given by the schema evolution operation, or computed by a *Match* operator), one can compose this mapping with the original mapping between the source schema S and the integrated schema T to get a mapping from the new version of the source schema S' to the integrated schema T.

Such applications of model management operators are also possible for other integrated information systems. For example, web services or e-business systems often have to take a message in XML format and store it in some relational data store for further processing. To implement this data transformation, a mapping between the XML schema and the schema of the relational database has to be defined. Such tasks are already supported by commercial products (e.g., Microsoft BizTalk, Altova MapForce). These products already use some model management operators (e.g., simple *Match* operators to simplify the task of mapping definition), but could further benefit from more powerful techniques for mapping generation. Clio started as a research prototype for mapping generation between XML and relational schemas, results have now been integrated into the IBM Information Integration platform.

Another application area for model management is the design and development of software applications. Model transformation is an inherent problem in software development, models have to be transformed into new metamodels, and then interoperability between the original model and the transformed model has to be implemented. A classical example is the transformation of an object-oriented data model of an application to a relational database schema, for which data access objects later have to be implemented to enable the synchronization between the data in the database and in the application. Such tasks are supported by frameworks (e.g., ADO.NET Entity Framework or the Entity Beans in Java 2 Enterprise Edition).

Future Directions

Mappings will become more and more important for MMS in the future [5]. The initial vision of model management, which considered mappings as rather simple correspondences attached with some complex expressions which contain the semantics of the mapping, was too limited. Mappings are in practice very complex and therefore, a rich mapping language is required in a MMS. The MMS must also be able to reason about the mappings. Furthermore, the MMS should not only enable the definition of a mapping, but support also its application; for example, using it for the integration of two schemas or for data transformation between two data stores. In addition, complex processes involving mappings (such as ETL process in data warehouses) have also to be considered.

Cross-references

- ▶ Data Integration
- ▶ Meta Data Repository
- ▶ Meta Model
- ▶ Schema Matching
- ▶ Schema Mapping
- ▶ Schema Mapping Composition

Recommended Reading

1. Atzeni P. and Torlone R. Management of Multiple Models in an Extensible Database Design Tool. In Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology, 1996, pp. 79–95.
2. Aumueller D., Do H.H., Massmann S., and Rahm E. Schema and ontology matching with COMA++. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 906–908.
3. Batini C., Lenzerini M., and Navathe S.B. A comparative analysis of methodologies for database schema integration. ACM Comput. Surv., 18(4):323–364, 1986.
4. Bernstein P.A., Halevy A.Y., and Pottinger R. A Vision for Management of Complex Models. ACM SIGMOD Rec., 29(4):55–63, 2000.
5. Bernstein P.A. and Melnik S. Model Management 2.0: Manipulating Richer Mappings. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 1–12.
6. Bernstein P.A. and Rahm E. Data Warehousing Scenarios for Model Management. In Proc. 19th Int. Conf. on Conceptual Modeling, 2000, pp. 1–15.
7. D.R. Dolk Model Management and Structured Modeling: The Role of an Information Resource Dictionary System. Commun. ACM, 31(6), 1988.
8. Fagin R., Kolaitis P.G., Popa L., and Tan W.C. Composing schema mappings: Second-order dependencies to the rescue. ACM Trans. Database Syst., 30(4):994–1055, 2005.
9. Hernández M.A., Miller R.J., and Haas L.M. Clio: A Semi-Automatic Tool for Schema Mapping. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 607.
10. Kensche D., Quix C., Chatti M.A., and Jarke M. GeRoMe: A Generic Role Based Metamodel for Model Management. J. Data Semant., VIII:82–117, 2007.
11. Kensche D., Quix C., Li X., and Li Y. GeRoMeSuite: A System for Holistic Generic Model Management. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 1322–1325.

12. Lenzerini M. Data Integration: A Theoretical Perspective. In Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2002, pp. 233–246.
13. Melnik S., Rahm E., and Bernstein P.A. Rondo: A Programming Platform for Generic Model Management. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 193–204.
14. Rahm E. and Bernstein P.A. A Survey of Approaches to Automatic Schema Matching. VLDB J., 10(4):334–350, 2001.
15. Spaccapietra S. and Parent C. View Integration: A Step Forward in Solving Structural Conflicts. IEEE Trans. Knowl. Data Eng., 6(2):258–274, 1994.

Model-based Querying in Sensor Networks

Amol Deshpande[1], Carlos Guestrin[2], Sam Madden[3]
[1]University of Maryland, College Park, MD, USA
[2]Carnegie Mellon University, Pittsburgh, PA, USA
[3]Massachussetts Institute of Technology, Cambridge, MA, USA

Synonyms

Approximate querying; Model-driven data acquisition

Definition

The data generated by sensor networks or other distributed measurement infrastructures is typically incomplete, imprecise, and often erroneous, such that it is not an accurate representation of physical reality. To map raw sensor readings onto physical reality, a mathematical description, a *model*, of the underlying system or process is required to complement the sensor data. Models can help provide more robust interpretations of sensor readings: by accounting for spatial or temporal biases in the observed data, by identifying sensors that are providing faulty data, by extrapolating the values of missing sensor data, or by inferring hidden variables that may not be directly observable. Models also offer a principled approach to predict future states of a system. Finally, since models incorporate spatio-temporal correlations in the environment (which tend to be very strong in many monitoring applications), they lead to significantly more energy-efficient query execution – by exploiting such attribute correlations, it is often possible to use a small set of observations to provide approximations of the values of a large number of attributes.

Model-based querying over a sensor network consists of two components: (i) identifying and/or building a model for a given sensor network, and (ii) executing declarative queries against a sensor network that has been augmented with such a model (these steps may happen serially or concurrently). The queries may be on future or hidden states of the system, and are posed in a declarative SQL-like language. Since the cost of acquiring sensor readings from the sensor nodes is the dominant cost in these scenarios, the optimization goal typically is to minimize the total data acquisition cost.

Historical Background

Statistical and probabilistic models have been a mainstay in the scientific and engineering communities for a long time, and are commonly used for a variety of reasons, from simple pre-processing tasks for removing noise (e.g., using Kalman Filters) to complex analysis tasks for prediction purposes (e.g., to predict weather or traffic flow). Standard books on machine learning and statistics should be consulted for more details (e.g., Cowell [3], Russell and Norvig [14]).

The first work to combine models, declarative SQL-like queries and live data acquisition in sensor networks was the BBQ System [7,6]. The authors proposed a general architecture for model-based querying, and posed the optimization problem of selecting the best sensor readings to acquire to satisfy a user query (which can be seen as a generalization of the *value of information problem* [14]). The authors proposed several algorithms for solving this optimization problem; they also evaluated the approach on a several real-world sensor network datasets, and demonstrated that model-based querying can provide high-fidelity representation of the real phenomena and leads to significant performance gains versus traditional data acquisition techniques. Several works since then have considerably expanded upon the basic idea, including development of sophisticated algorithms for data acquisition [12,13], more complex query types [15], and integration into a relational database system [8,11].

The querying aspect of this problem has many similarities to the problem of approximate query processing in database systems, which often uses model-like *synopses*. For example, the AQUA project [1] proposed a number of sampling-based synopses that can provide approximate answers to a variety of queries using a fraction of the total data in a database. As with BBQ, such answers typically include tight

bounds on the correctness of answers. AQUA, however, is designed to work in an environment where it is possible to generate an independent random sample of data (something that is quite tricky to do in sensor networks, as losses are correlated and communicating random samples may require the participation of a large part of the network). AQUA also does not exploit correlations, which means that it lacks the *predictive* power of representations based on probabilistic models. Deshpande et al. [4] and Getoor et al. [9] proposed exploiting data correlations through use of graphical modeling techniques for approximate query processing, but, unlike BBQ, neither provide any guarantees on the answers returned. Furthermore, the optimization goal of approximate query processing is typically not to minimize the data acquisition cost, rather it is minimizing the size of the synopsis, while maintaining reasonable accuracy.

Foundations

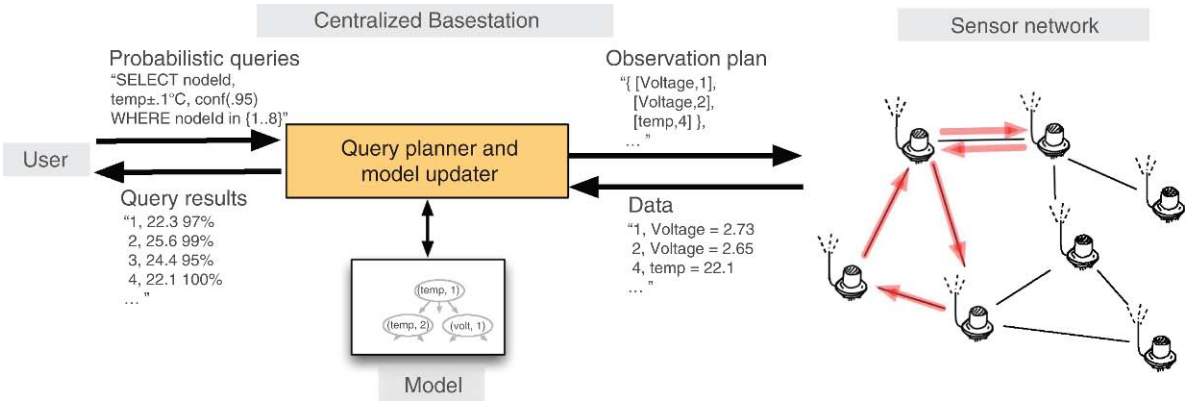
Figure 1 shows the most common architecture of a model-based querying system (adapted from the architecture of the BBQ system). The model itself is located at a centralized, Internet-connected basestation, which also interacts with the user. The user may issue either continuous or ad hoc queries against the sensor network, using a declarative SQL-like language. The key module in this architecture is the *query planner and model updater*, which is in charge of maintaining the model and answering the user queries (possibly by acquiring more data from the underlying sensor network). The following sections elaborate on the various components of such a system.

Model

A model is essentially a simplified representation of the underlying system or the process, and describes how various attributes of the system interact with each other, and how they evolve over time. Hence, the exact form of the model is heavily dependent on the system being modeled, and an astounding range of models have been developed over the years for different environments. For ease of exposition, the rest of this entry focuses on a dynamic model similar to the one used in the BBQ system.

Let $X_1,...,X_n$ denote the (n) attributes of interest in the sensor network. Further, let X_i^t denote the value of X_i at time t (assuming that time is discrete). At any time t, a subset of these attributes may be observed and communicated to the basestation; here, the observations at time t are denoted by $\mathbf{o}^t$ (note that hidden variables can never be observed). The attributes typically correspond to the properties being monitored by the sensor nodes (e.g., *temperature* on sensor number 5, *voltage* on sensor number 8). However, more generally, they may be *hidden variables* that are of interest, but cannot be directly observed. For example, it may be useful to model and query a hidden boolean variable that denotes whether a sensor is faulty [11] – the value of this variable can be inferred using the model and the actual observations from the sensor.

The model encodes the spatial and temporal relationships between these attributes of interest. At any time t, the model provides us with a posterior *probability density function* (pdf), $p(X_1^t,..., X_n^t|\mathbf{o}^{1...(t-1)})$, assigning a probability for each possible assignment to the attributes at time t given the observations



Model-based Querying in Sensor Networks. Figure 1. Architecture of a model-based querying system (adapted from BBQ [7,6]).

made so far. Such a joint distribution can capture all the spatial correlations between the attributes; more compact representations like Bayesian networks can be used instead as well.

To model the temporal correlations, it is common to make a *Markov* assumption; given the values of *all* attributes at time t, one assumes that the values of the attributes at time $t + 1$ are independent of those for any time earlier than t. This assumption leads to a simple model for a dynamic system where the dynamics are summarized by a conditional density called the *transition model*, $p(X_1^{t+1},...,X_n^{t+1}|X_1^t,...,X_n^t)$. Using a transition model, one can compute $p(X_1^{t+1},...,X_n^{t+1}|\mathbf{o}^{1...t})$ from $p(X_1^t,...,X_n^t|\mathbf{o}^{1...t})$ using standard probabilistic procedures. Different transition models may be used for different time periods (e.g., hour of day, day of week, season, etc.) to model the differences in the way the attributes evolve at different times.

Learning the model. Typically in probabilistic modeling, a *class* of models is chosen (usually with input from a *domain expert*), and learning techniques are then used to pick the best model in the class. Model parameters are typically learned from training data, but can also be directly inferred if the behavior of the underlying physical process is well-understood. In BBQ, the model was learned from historical data, which consisted of readings from all of the monitored attributes over some period of time.

Updating the model. Given the formulation above, model updates are fairly straightforward. When a new set of observations arrives (say $\mathbf{o}^t$), it can be incorporated into the model by conditioning on the observations to compute new distributions (i.e., by computing $p(X_1^t,...,X_n^t|\mathbf{o}^{1...t})$ from $p(X_1^t,...,X_n^t|\mathbf{o}^{1...(t-1)})$. Similarly, as time advances, the transition model is used to compute the new distribution for time $t + 1$ (i.e., $p(X_1^{t+1},...,X_n^{t+1}|\mathbf{o}^{1...t})$ is computed from $p(X_1^t,...,X_n^t|\mathbf{o}^{1...t})$ and $p(X_1^{t+1},...,X_n^{t+1}|X_1^t...X_n^t)$).

Query Planning and Execution

User queries are typically posed in a declarative SQL-like language that may be augmented with constructs that allow users to specify the approximation that the user is willing to tolerate, and the desired confidence in the answer. For example, the user may ask the system to report the temperature readings at all sensors within $\pm$ 0.5, with confidence 95%. For many applications (e.g., building temperature control), such approximate answers may be more than sufficient. Tolerance for such approximations along with the correlations encoded by the model can lead to significant energy savings in answering such queries.

Answering queries probabilistically based on a pdf over the query attributes is conceptually straightforward; to illustrate this process, consider two types of queries (here, assume that all queries are posed over attributes $X_1^t,...,X_n^t$, and the corresponding pdf is given by $P(X_1^t,...,X_n^t)$):

Value query. A value query [6] computes an approximation of the values of the attributes to within $\pm\ \varepsilon$ of the true value, with confidence at least $1 - \delta$. Answering such a query involves computing the expected value of each of the attribute, μ_i^t, using standard probability theory. These μ_i^t's will be the reported values. The pdf can then be used again to compute the probability that X_i^t is within ε from the mean, $P(X_i^t \in [\mu_i^t - \varepsilon, \mu_i^t + \varepsilon])$. If all of these probabilities meet or exceed user specified confidence threshold, then the requested readings can be directly reported as the means μ_i^t. If the model's confidence is too low, additional readings must be acquired before answering the query (see below).

Max query. Consider an *entity* version of this query [2] where the user wants to know the identity of the sensor reporting the maximum value. A naive approach to answering this query is to compute, for each sensor, the probability that its value is the maximum. If the maximum of these probabilities is above $1 - \delta$, then an answer can be returned immediately; otherwise, more readings must be acquired. Although conceptually simple, computing the probability that a given sensor is reporting the maximum value is non-trivial, and requires complex integration that may be computationally infeasible [15].

If the model is not able to provide sufficient confidence in the answer, the system must acquire more readings from the sensor network, to bring the model's confidence up to the user specified threshold. Suppose the system observes a set of attributes $\mathcal{O} \subset \{X_1,...,X_n\}$. After incorporating these observations into the model and recomputing the answer, typically the confidence in the (new) answer will be higher (this is not always true). The new confidence will typically depend on the actual observed values. Let $R(\mathcal{O})$ denote the *expected* confidence in the answer after observing

$\mathcal{O}$. Then, the optimization problem of deciding which attributes to observe can be stated as follows:

$$\begin{aligned} &\text{minimize}_{\mathcal{O}\subseteq\{1,\ldots,n\}} \quad C(\mathcal{O}), \\ &\text{such that} \qquad\qquad\quad R(\mathcal{O}) \geq 1-\delta. \end{aligned} \tag{1}$$

where $C(\mathcal{O})$ denotes the *data acquisition cost* of observing the values of attributes in $\mathcal{O}$.

This optimization problem combines three problems that are known to be intractable, making it very hard to solve it in general:

1. Answering queries using a pdf: as mentioned above, this can involve complex numerical integration even for simple queries such as max.
2. hoosing the minimum set of sensor readings to acquire to satisfy the query (this is similar to the classic *value of information problem* [14,12,15,10]).
3. Finding the optimal way to collect a required set of sensor readings from the sensor network that minimizes the total communication cost. Meliou et al. [13] present several approximation algorithms for this NP-Hard problem.

Example

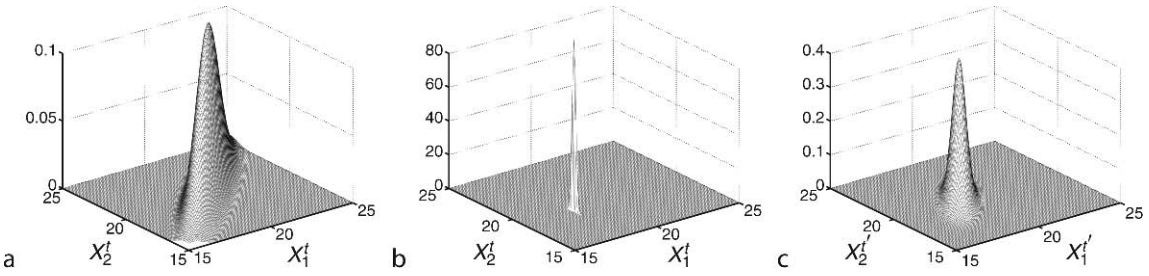
Figure 2 illustrates the query answering process using a simple example, where the model takes the form of time-varying *bivariate Gaussian (normal)* distribution over two attributes, X_1 and X_2. This was the basic model used in the BBQ system. A bivariate Gaussian is the natural extension of the familiar unidimensional normal probability density function (pdf), known as the "bell curve". Just as with its one-dimensional counterpart, a bivariate Gaussian can be expressed as a function of two parameters: a length-2 vector of means, μ, and a 2×2 matrix of covariances, Σ. Figure 2a shows a three-dimensional rendering of a Gaussian over the two attributes at time t, X_1^t and X_2^t the z axis represents the *joint density* that $X_2^t = x$ and $X_1^t = y$).

Now, consider a *value query* over this model, posed at time t, which asks for the values of X_1^t and X_2^t, within $\pm\epsilon$, with confidence $1-\delta$. The reported values in this case would be the means (μ), and the confidence can be computed easily using Σ (details can be found in [6]). Considering the high initial covariance, it is unlikely that the Gaussian in Figure 2a can achieve the required confidence. Suppose the system decides to observe X_1^t. Figure 2b shows the result of incorporating this observation into the model. Note that not only does the spread of X_1^t reduces to near zero, because of the high correlation between X_1^t and X_2^t, the variance of X_2^t also reduces dramatically, allowing the system to answer the query with required confidence.

Then, after some time has passed, the belief about the values of X_1 and X_2 (at time $t' > t$) will be "spread out", again providing a high-variance Gaussian over two attributes, although both the mean and variance may have shifted from their initial values, as shown in Fig. 2c.

Key Applications

Model-based querying systems like BBQ, that exploit statistical modeling techniques and optimize the utilization of a network of resource constrained devices could have significant impact in a number of application domains, ranging from control and automation in buildings [16] to highway traffic monitoring. Integrating a model into the data acquisition process can significantly improve data quality and reduce data



Model-based Querying in Sensor Networks. Figure 2. Example of Gaussians: (a) 3D plot of a 2D Gaussian with high covariance; (b) the resulting Gaussian after a particular value of X_1^t has been observed (because of measurement noise, there might still be some uncertainty about the true value of X_1^t); (c) the uncertainty about X_1 and X_2 increases as time advances to t'.

uncertainty. The ability to query over missing, future or hidden states of the system will prove essential in many applications where sensor failures are common (e.g., highway traffic monitoring) or where direct observation of the variables of interest is not feasible. Finally, model-based querying has the potential to significantly reduce the cost of data acquisition, and thus can improve the life of a resource-constrained measurement infrastructure (e.g., battery-powered wireless sensor networks) manyfold.

Future Directions

Model-based querying is a new and exciting research area with many open challenges that are bound to become more important with the increasingly widespread use of models for managing sensor data. The two most important challenges are dealing with a wide variety of models that may be used in practice, and designing algorithms for query processing and data acquisition; these are discussed briefly below:

Model selection and training. The choice of model affects many aspects of model-based querying, most importantly the accuracy of the answers and the confidence bounds that can be provided with them. The problem of selecting the right model class has been widely studied [3,14] but can be difficult in some applications. Furthermore, developing a new system for each different model is not feasible. Ideally, using a new model should involve little to no effort on the part of user. Given a large variety of models that may be applicable in various different scenarios, this may turn out to be a tremendous challenge.

Algorithms for query answering and data acquisition. Irrespective of the model selected, when and how to acquire data in response to a user query raises many hard research challenges. As discussed above, this problem combines three very hard problems, and designing general-purpose algorithms that can work across the spectrum of different possible model remains an open problem.

See Deshpande et al. [5] for a more elaborate discussion of the challenges in model-based querying.

Cross-references

- ▶ Ad-Hoc Queries in Sensor Networks
- ▶ Approximate Query Processing
- ▶ Continuous Queries in Sensor Networks
- ▶ Data Acquisition and Dissemination in Sensor Networks
- ▶ Data Uncertainty Management in Sensor Networks
- ▶ Models

Recommended Reading

1. Acharya S., Gibbons P.B., Poosala V., and Ramaswamy S. Join synopses for approximate query answering. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 275–286.
2. Cheng R., Kalashnikov D.V., and Prabhakar S. Evaluating probabilistic queries over imprecise data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 551–562.
3. Cowell R., Dawid P., Lauritzen S., and Spiegelhalter D. Probabilistic Networks and Expert Systems. Spinger, New York, 1999.
4. Deshpande A., Garofalakis M., and Rastogi R. Independence is good: dependency-based histogram synopses for high-dimensional data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 199–210.
5. Deshpande A., Guestrin C., and Madden S. Using Probabilistic Models for Data Management in Acquisitional Environments. In Proc. 2nd Biennial Conf. on Innovative Data Systems Research, 2005, pp. 317–328.
6. Deshpande A., Guestrin C., Madden S., Hellerstein J., and Hong W. Model-Driven Approximate Querying in Sensor Networks. VLDB J., 14(4):417–443, 2005.
7. Deshpande A., Guestrin C., Madden S., Hellerstein J.M., 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. Deshpande A. and Madden S. MauveDB: supporting model-based user views in database systems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 73–84.
9. Getoor L., Taskar B., and Koller D. Selectivity estimation using probabilistic models. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 461–472.
10. Goel A., Guha S., and Munagala K. Asking the right questions: model-driven optimization using probes. In Proc. 25th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2006, pp. 203–212.
11. 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.
12. Krause A., Guestrin C., Gupta A., and Kleinberg J. Near-optimal sensor placements: maximizing information while minimizing communication cost. In Proc. 5th Int. Symp. Inf. Proc. in Sensor Networks, 2006, pp. 2–10.
13. Meliou A., Chu D., Hellerstein J., Guestrin C., and Hong W. Data gathering tours in sensor networks. In Proc. 5th Int. Symp. Inf. Proc. in Sensor Networks, 2006, pp. 43–50.
14. Russell S. and Norvig P. Artificial Intelligence: A Modern Approach. Prentice Hall, 1994.
15. Silberstein A., Braynard R., Ellis C., Munagala K., and Yang J. A Sampling-Based approach to Optimizing Top-k Queries in Sensor networks. In Proc. 22nd Int. Conf. on Data Engineering, 2006, p. 68.
16. Singhvi V., Krause A., Guestrin C., Garrett Jr J., and Matthews H. Intelligent light control using sensor networks. In Proc. 3rd Int. Conf. on Embedded Networked Sensor Systems, 2005, pp. 218–229.

Model-driven Data Acquisition

► Model-Based Querying in Sensor Networks

Module

► Snippet

MOF

► Meta Object Facility

Molecular Interaction Graphs

► Biological Networks

Moment

► Chronon
► Time Instant

Monitoring

► Auditing and Forensic Analysis

Monitoring of Real-Time Logic Expressions

► Event Detection

Monotone Constraints

Carson Kai-Sang Leung
University of Manitoba, Winnipeg, MB, Canada

Synonyms

Monotonic constraints

Definition

A constraint C is *monotone* if and only if for all itemsets S and S':

$$\text{if } S \supseteq S' \text{ and } S \text{ violates } C, \text{ then } S' \text{ violates } C.$$

Key Points

Monotone constraints [1–3] possess the following property. If an itemset S violates a monotone constraint C, then any of its subsets also violates C. Equivalently, all supersets of an itemset satisfying a monotone constraint C also satisfy C (i.e., C is upward closed). By exploiting this property, monotone constraints can be used for reducing computation in frequent itemset mining with constraints. As frequent itemset mining with constraints aims to find frequent itemsets that satisfy the constraints, if an itemset S satisfies a monotone constraint C, no further constraint checking needs to be applied to any superset of S because all supersets of S are guaranteed to satisfy C. Examples of monotone constraints include $min(S.Price) \leq \$30$, which expresses that the minimum price of all items in an itemset S is at most \$30. Note that, if the minimum price of all items in S is at most \$30, adding more items to S would not increase its minimum price (i.e., supersets of S would also satisfy such a monotone constraint).

Cross-references

► Frequent Itemset Mining with Constraints

Recommended Reading

1. Brin S., Motwani R., and Silverstein C. Beyond market baskets: generalizing association rules to correlations. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 265–276.
2. Grahne G., Lakshmanan L.V.S., and Wang X. Efficient mining of constrained correlated sets. In Proc. 16th Int. Conf. on Data Engineering, 2000, pp. 512–521.
3. 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.

Monotonic Constraints

► Monotone Constraints

Monotonicity Property

► Apriori Property and Breadth-First Search Algorithms

Motion Graphics

► Dynamic Graphics

Moving Object

Ralf Hartmut Güting
University of Hagen, Hagen, Germany

Synonyms

Time dependent geometry

Definition

A moving object is essentially a time dependent geometry. Moving objects are the entities represented and queried in moving objects databases.

Key Points

The term emphasizes the fact that geometries may change continuously (whereas earlier work on spatio-temporal databases allowed only discrete changes, e.g., of land parcels). One can distinguish between moving objects for which only the time dependent position is of interest and those for which also shape and extent are relevant and may change over time. The first can be characterized as *moving points*, the second as *moving regions*. For example, moving points could represent people, vehicles (such as cars, trucks, ships or air planes), or animals. Moving regions could be hurricanes, forest fires, spread of epidemic diseases etc. Moving point data may be captured by GPS devices or RFID tags; moving region data may result from processing sequences of satellite images, for example. Moving points and moving regions can be made available as data types in suitable type systems; such a design can be found in [1]. Such an environment may have further "moving" data types (e.g., *moving lines*).

Cross-references

► Moving Objects Databases and Tracking

Recommended Reading

1. Güting R.H., Böhlen M.H., Erwig M., Jensen C.S., Lorentzos N.A., Schneider M., and Vazirgiannis M. A foundation for representing and querying moving objects in databases. ACM Trans. Database Syst., 25:1–42, 2000.

Moving Object Trajectories

► Spatio-Temporal Trajectories

Moving Objects Databases and Tracking

Ralf Hartmut Güting
University of Hagen, Hagen, Germany

Synonyms

Spatio-temporal databases; trajectory databases

Definition

Moving objects database systems provide concepts in their data model and data structures in the implementation to represent moving objects, i.e., continuously changing geometries. Two important abstractions are *moving point*, representing an entity for which only the time dependent position is of interest, and *moving region*, representing an entity for which also the time dependent shape and extent is relevant. Examples of moving points are cars, trucks, air planes, ships, mobile phone users, RFID equipped goods, or polar bears; examples of moving regions are forest fires, deforestation of the Amazon rain forest, oil spills in the sea, armies, epidemic diseases, hurricanes, and so forth.

There are two flavors of such databases. The first represents information about a set of currently moving objects. Basically one is interested in efficiently maintaining their location information and asking queries about the current and expected near future positions and relationships between objects. In this case, no information about histories of movement is kept. This is sometimes also called a *tracking database*.

The second represents complete histories of movements. The goal in the design of query languages for moving objects is to be able to ask any kind of questions about such movements, perform analyses, derive

information, in a way as simple and elegant as possible. The underlying system must support efficient execution of such analyses. This view is associated with the term *moving objects database*, sometimes also called *trajectory database*.

Historical Background

The field of moving objects databases came into being in the late 1990s mainly by two parallel developments. First, a model was developed [11,16,17] that allows one to keep track in a database of a set of time dependent locations, e.g., to represent vehicles. The authors observed that one should store in a database not the locations directly, which would require high update rates, but rather a motion vector, representing an object's expected position over time. An update to the database is needed only when the deviation between the expected position and the real position exceeds some threshold. At the same time this concept introduces an inherent, but bounded uncertainty about an object's real location. The model was formalized introducing the concept of a *dynamic attribute*. This is an attribute of a normal data type which changes implicitly over time. This implies that results of queries over such attributes also change implicitly over time. A related query language FTL (future temporal logic) was introduced that allows one to specify time dependent relationships between expected positions of moving objects.

Second, the European project CHOROCHRONOS set out to integrate concepts from spatial and temporal databases. In this case, one represents in a database time-dependent geometries of various kinds such as points, lines, or regions. Earlier work on spatio-temporal databases had generally admitted only discrete changes. This restriction was dropped and continuously changing geometries were considered. A model was developed based on the idea of *spatio-temporal data types* to represent histories of continuously changing geometries [2,4–6]. The model offers data types such as *moving point* or *moving region* together with a comprehensive set of operations. For example, there are operations to compute the projection of a moving point into the plane, yielding a *line* value, or to compute the distance between a moving point and a moving region, returning a time dependent real number, or *moving real*, for short. Such data types can be embedded into a DBMS data model as attribute types and can be implemented as an extension package.

A second approach to data modeling for moving object histories was pursued in CHOROCHRONOS. Here the constraint model was applied to the representation of moving objects [10] and a prototype called Dedale was implemented. Constraint databases can represent geometries in n-dimensional spaces; since moving objects exist in 3D (2D + time) or 4D (3D + time) spaces, they can be handled by this approach. Several researchers outside CHOROCHRONOS also contributed to the development of constraint-based models for moving objects.

Foundations

Modeling and Querying Current Movement (Tracking)

Consider first moving objects databases for current and near future movement, or tracking databases. Sets of moving entities might be taxi-cabs in a city, trucks of a logistics company, or military vehicles in a military application. Possible queries might be:

- Retrieve the three free cabs closest to Cottle Road 52 (a passenger request position).
- Which trucks are within 10 km of truck T70 (which needs assistance)?
- Retrieve the friendly helicopters that will arrive in the valley within the next 15 min and then stay in the valley for at least 10 min.

Statically, the positions of a fleet of taxi-cabs, for example, could be easily represented in a relation

```
taxi-cabs(id: int, pos: point)
```

Unfortunately this representation needs frequent updates to keep the deviation between real position and position in the database small. This is not feasible for large sets of moving objects.

The MOST (moving objects spatio-temporal) data model [11,16], discussed in this section, stores instead of absolute positions a motion vector which represents a position as a linear function of time. This defines an expected position for a moving object. The distance between the expected position and the real position is called the *deviation*. Furthermore, a *distance threshold* is introduced and a kind of contract between a moving object and the database server managing its position is assumed. The contract requires that the moving object observes the deviation and sends an update to the server when it exceeds the threshold. Hence the threshold establishes a bound on the *uncertainty* about an object's real position.

A fundamental new concept in the MOST model is that of a *dynamic attribute*. Each attribute of an object class is classified to be either static or dynamic. A dynamic attribute is of a standard data type (e.g., *int*, *real*) within the DBMS conceptual model, but changes its value automatically over time. This means that queries involving such attributes also have time dependent results, even if time is not mentioned in the query and no updates to the database occur.

The MOST model assumes that time advances in discrete steps, so-called clock ticks. Hence time can be represented by integer values. For a data type to be eligible for use in a dynamic attribute, it is necessary that the type has a value 0 and an addition operation. This holds for numeric types but can be extended to types like *point*. A dynamic attribute A of type T is then internally represented by three subattributes $A.value$, $A.updatetime$, and $A.function$, where $A.value$ is of type T, $A.updatetime$ is a time value, and $A.function$ is a function f: $int \rightarrow T$ such that at time $t = 0$, $f(t) = 0$. The semantics of this representation is called the value of A at time t and defined as

$$value(A, t) = A.value + A.function(t - A.updatetime)$$
$$for\ t \geq A.updatetime$$

When attribute A is mentioned in a query, its dynamic value $value(A, t)$ is meant.

With dynamic attributes, for each clock tick one obtains a new state of the database, even without explicit updates. Such a sequence of states is called a database history. With each explicit update, all subsequent states change so that one obtains a new database history. One can now define different types of queries:

- An *instantaneous query* issued at a time t_0 is evaluated once on the database history starting at time t_0.
- A *continuous query* issued at time t_0 is (conceptually) reevaluated for each clock tick. Hence it is evaluated once on the database history starting at time t_0, then on the history starting at t_1, then on... t_2, and so forth.

Of course, reevaluating a continuous query on each clock tick is not feasible. Instead, the evaluation algorithm for such queries is executed only once and returns a time dependent result, in the form of a set of tuples with associated time stamps. A reevaluation is only necessary when explicit updates occur.

The query language associated with the MOST model is called FTL (future temporal logic). Here are a few example queries formulated in FTL.

1. Which trucks are within 10 km of truck T70?
```
RETRIEVE t
FROM trucks t, trucks s
WHERE s.id = 'T70' ∧ dist(s, t) <= 10
```
Here nothing special happens, yet, the result is time dependent.
2. Retrieve the helicopters that will arrive in the valley within the next 15 min and then stay in the valley for at least 10 min.
```
RETRIEVE h
FROM helicopters h
WHERE eventually_within_15
  (inside(h, Valley) ∧
  always_for_10 (inside(h, Valley)))
```
Here *Valley* is a polygon object.

The general form of a query in FTL is
```
RETRIEVE <target-list> FROM <object
classes> WHERE <FTL-formula>
```
FTL formulas may contain special time dependent constructs, in particular:

- If f and g are formulas, then f **until** g and **nexttime** f are formulas

Informally, the meaning is that for a given database state s, f **until** g holds if there exists a future state s' on the database history such that g holds in state s' and for all states from s up to s', f holds. Similarly, **nexttime** f holds in state s_{i+1} if f holds in state s_i. Based on such temporal operators one can define bounded temporal operators like **eventually_within_c** g or **always_for_c** g as they occur in the second example query.

Modeling and Querying History of Movement

Now consider the problem of representing complete histories of movement in a database. The scope is also extended from point objects to more complex geometrical shapes.

The idea of the approach [4] presented in the following is to introduce spatio-temporal data types that encapsulate time dependent geometries with suitable operations. For moving objects, point and region appear

to be most relevant, leading to data types *moving point* and *moving region*, respectively. The *moving point* type (*mpoint* for short) can represent entities such as vehicles, people, or animals moving around whereas the *moving region* type (*mregion*) can represent hurricanes, forest fires, armies, or flocks of animals, for example. Geometrically, values of spatio-temporal data types are embedded into a 3D space (2D + time) if objects move in the 2D plane, or in a 4D space if movement in the 3D space is modeled. Hence, a moving point and a moving region can be visualized as shown in Fig. 1

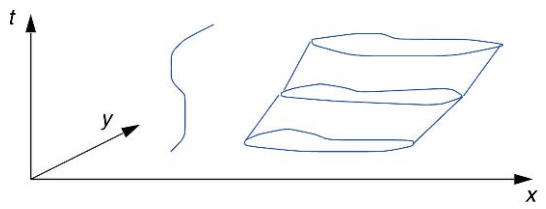
Data types may be embedded in the role of attribute types into a DBMS data model. For example, in a relational setting, there may be relations to represent the movements of air planes or storms:

```
flight (id: string, from: string, to:
  string, route: mpoint)
weather (id: string, kind: string,
  area: mregion)
```

The data types include suitable operations such as:

intersection: *mpoint* × *mregion* → *mpoint*
trajectory: *mpoint* → *line*
deftime: *mpoint* → *periods*
length: *line* → *real*

One discovers quickly that in addition to the main types of interest, *mpoint* and *mregion*, related spatial and temporal as well as other time-dependent types are needed. The operations above have the following meaning: **Intersection** returns the part of a moving point whenever it lies inside a moving region, which is a moving point (*mpoint*) again. **Trajectory** projects a moving point into the plane, yielding a *line* value (a curve in the 2D space). **Deftime** returns the set of time intervals when a moving point is defined, of a data type called *periods*. **Length** returns the length of a *line* value.



Moving Objects Databases and Tracking. Figure 1. A moving point and a moving region.

Given such operations, one may formulate queries:

"Find all flights from Düsseldorf that are longer than 5,000 km."

```
select id
from flights
where from = 'DUS' and length
  (trajectory (route)) > 5000
```

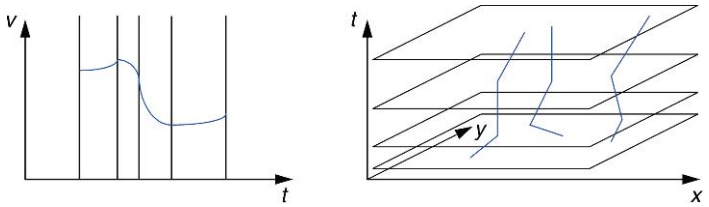
"At what times was flight BA488 within the snow storm with id S16?"

```
select deftime(intersection
  (f.route, w.area))
from flights as f, weather as w
where f.id = 'BA488' and w.id = 'S16'
```

Reference [4] develops the basic idea, discusses the distinction between *abstract models* (using infinite sets, and describing e.g., a moving region as a function from time into region values) and *discrete models* (selecting a suitable finite representation, e.g., describing a moving region as a polyhedron in the 3D space), and clarifies several fundamental questions related to this approach. A system of related data types and operations for moving objects is carefully defined in [6], emphasizing genericity, closure, and consistency. The semantics of these types is defined at the abstract level.

Implementation is based on the discrete model proposed in [5] using algorithms for the operations studied in [2]. The discrete model uses the so-called *sliced representation* as illustrated in Fig. 2. A temporal function value is represented as a time-ordered sequence of *units* where each unit has an associated time interval, and time intervals of different units are disjoint. Each unit is capable of representing a piece of the moving object by a "simple" function. Simple functions are linear functions for moving points or regions, and quadratic polynomials (or square roots of such) for moving reals, for example.

Within a database system, an extension module (data blade, cartridge, extender, etc.) can be provided offering implementations of such types and operations. The sliced representation is basically stored in an array of units. (It is a bit more complicated in case of variable size units as for a moving region, for example.) Because values of moving object types can be large and complex, the DBMS must provide suitable storage techniques for managing large objects. A large



Moving Objects Databases and Tracking. Figure 2. Sliced representation of a *moving*(*real*) and a *moving*(*points*) value.

part of this design has been implemented prototypically in the SECONDO extensible DBMS [1] which is available for download (see URL below).

Related Issues

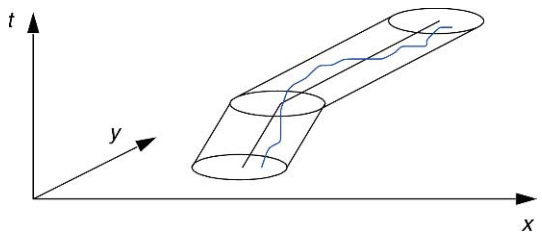
In this short closing section some issues related to moving objects databases are briefly discussed.

Uncertainty Locations of moving objects are most often captured using GPS devices at certain instants of time. This introduces an inherent uncertainty already for the sampled positions (due to some inaccuracy of the GPS device) and in particular for the periods of time between measurements [9]. Bounded uncertainty is also introduced due to a contract between location server and moving object, as discussed above. The MOST model includes concepts to deal with this uncertainty in querying [16,17]. For history of movement, one can consider uncertain trajectories based on an uncertainty threshold, resulting in a shape of a kind of slanted cylinder (Fig. 3). It is only known that the real position is somewhere inside this volume. Based on this model, Trajcevski et al. [15] have defined a set of predicates between a trajectory and a region in space taking uncertainty and aggregation over time into account.

Movement in Networks Whereas the basic case is free movement in the Euclidean plane, it is obvious that vehicles usually move on transport networks. There is a branch of research on network-constrained movement (e.g., [7,13]); there is also work on indexing network based movements. For network based movement, captured GPS positions have to be mapped to the transportation network; this is called map matching.

Spatio-Temporal Indexing A lot of research exists on indexing movement, both for expected near future movement and for history movement.

Query Processing for Continuous/Location Based Queries Continuous queries for moving objects have been studied in depth, for example, maintaining the result of nearest neighbor or range queries both for moving query and moving data objects (e.g., [12,14], see continuous monitoring of spatial queries).



Moving Objects Databases and Tracking. Figure 3. Geometry of an uncertain trajectory.

Spatiotemporal Aggregation and Selectivity Estimation Another subfield of research in moving objects databases considers the problem of computing precisely or estimating the numbers of moving objects within certain areas in space and time – hence, of computing aggregates. For example, various index structures have been proposed to compute efficiently such aggregates. This is also related to the problem of performing selectivity estimation for spatio-temporal query processing.

Key Applications

Databases for querying current and near future movement like the MOST model described, are the foundation for location-based services. Service providers can keep track of the positions of mobile users and notify them of upcoming service offers even some time ahead. For example, gas stations, hotels, shopping centres, sightseeing spots, or hospitals in case of an emergency might be interesting services for car travelers.

Several applications need to keep track of the current positions of a large collection of moving objects, for example, logistics companies, parcel delivery services, taxi fleet management, public transport systems, air traffic control. Marine mammals or other animals

are traced in biological applications. Obviously, the military is also interested in keeping track of fighting units in battlefield management.

Database systems for querying history of movement are needed for more complex analyses of recorded movements. For example, in air traffic control one may go back in time to any particular instant or period to analyze dangerous situations or even accidents. Logistics companies may analyze the paths taken by their delivery vehicles to determine whether optimizations are possible. Public transport systems in a city may be analyzed to understand reachability of any place in the city at different periods of the day. Movements of animals may be analyzed in biological studies. Historical modeling may represent movements of people or tribes and actually animate and query such movements over the centuries.

The data model of such systems offers not only moving point entities but also moving regions. Hence also developments of areas on the surface of the earth may be modeled and analyzed like the deforestation of the Amazon rain forest, the Ozone hole, development of forest fires or oil spills over time, and so forth.

Future Directions

Recent research in databases has often addressed specific query types like continuous range queries or nearest neighbor queries, and then focused on designing efficient algorithms for them. An integration of the many specific query types into complete language designs as presented in this entry is still lacking. Uncertainty may be treated more completely also in the approaches for querying history of movement. A seemless query language for querying past, present, and near future would also be desirable.

A text book covering the topics presented in this article in more detail is [8].

Experimental Results

Running times for queries of the BerlinMOD benchmark (see below), evaluated in the SECONDO system, can be found in [3].

Data Sets

A collection of links to data sets with real spatio-temporal data, partially assembled within the CHOROCHRONOS project mentioned above can be found at http://dke.cti.gr/people/pfoser/data.html

Recently a benchmark data set is available, the so-called BerlinMOD benchmark [3]. It is based on a simulation of the movements of 2,000 people's vehicles in the city of Berlin, observed over 1 month (at scale factor 1). The benchmark contains a number of test queries. Test data are generated by the SECONDO system. The benchmark can be found and the relevant resources downloaded at

http://dna.fernuni-hagen.de/secondo/BerlinMOD/Berlin MOD.html

See also real and synthetic test data sets; this entry should include links to further test data generators.

URL to Code

SECONDO as a prototypical moving objects database system (for histories of movement, or trajectories) is available for download at

http://dna.fernuni-hagen.de/Secondo.html/

Cross-references

- ▶ Constraint Query Languages
- ▶ Continuous Monitoring of Spatial Queries
- ▶ Indexing Historical Spatio-Temporal Data
- ▶ Indexing of the Current and Near-Future Positions of Moving Objects
- ▶ Location-based Services
- ▶ Map Matching
- ▶ Real and Synthetic Test Datasets
- ▶ Spatial and Spatio-temporal Data Models and Languages
- ▶ Spatio-Temporal Data Mining
- ▶ Spatio-temporal Data Types
- ▶ Spatio-temporal Data Warehouses
- ▶ Spatio-temporal Trajectories

Recommended Reading

1. Almeida V.T., Güting R.H., and Behr T. Querying moving objects in SECONDO. In Proc. 7th Int. Conf. on Mobile Data Management, 2006, pp. 47–51.
2. Cotelo Lema J.A., Forlizzi L., Güting R.H., Nardelli E., and Schneider M. Algorithms for moving object databases. The Comput. J., 46(6):680–712, 2003.
3. Düntgen C., Behr T., and Güting R.H. BerlinMOD: a benchmark for moving object databases. Informatik-Report 340, Fernuniversität Hagen, 2007. Available at: http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.pdf
4. Erwig M., Güting R.H., Schneider M., and Vazirgiannis M. Spatio-temporal data types: an approach to modeling and querying moving objects in databases. GeoInformatica, (3):265–291, 1999.

5. Forlizzi L., Güting R.H., Nardelli E., and Schneider M. A data model and data structures for moving objects databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 319–330.
6. Güting R.H., Böhlen M.H., Erwig M., Jensen C.S., Lorentzos N.A., Schneider M., and Vazirgiannis M. A foundation for representing and querying moving objects in databases. ACM Trans. Database Syst., 25:1–42, 2000.
7. Güting R.H., de Almeida V.T., and Ding Z. Modeling and querying moving objects in networks. VLDB J., 15(2):165–190, 2006.
8. Güting R.H., and Schneider M. Moving Objects Databases. Morgan Kaufmann Publishers, San Fransisco, CA, USA, 2005.
9. Pfoser D., and Jensen C.S. Capturing the uncertainty of moving-object representations. In Proc. 6th Int. Symp. Advances in Spatial Databases, 1999, pp. 111–131.
10. Rigaux P., Scholl M., Segoufin L., and Grumbach S. Building a constraint-based spatial database system: model, languages, and implementation. Inf. Syst., 28(6):563–595, 2003.
11. Sistla A.P., Wolfson O., Chamberlain S., and Dao S. Modeling and querying moving objects. In Proc. 13th Int. Conf. on Data Engineering, 1997, 422–432.
12. Song Z., and Roussopoulos N. K-nearest neighbor search for moving query point. In Proc. 7th Int. Symp. Advances in Spatial and Temporal Databases, 2001, pp. 79–96.
13. Speicys L., Jensen C.S., and Kligys A. Computational data modeling for network-constrained moving objects. In Proc. 11th ACM Int. Symp. on Advances in Geographic Inf. Syst., 2003, pp. 118–125.
14. Tao Y., and Papadias D. Spatial queries in dynamic environments. ACM Trans. Database Syst., 28(2):101–139, 2003.
15. Trajcevski G., Wolfson O., Hinrichs K., and Chamberlain S. Managing uncertainty in moving objects databases. ACM Trans. Database Syst., 29(3):463–507, 2004.
16. Wolfson O., Chamberlain S., Dao S., Jiang L., and Mendez G. Cost and imprecision in modeling the position of moving objects. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 588–596.
17. Wolfson O., Sistla A.P., Chamberlain S., and Yesha Y. Updating and querying databases that track mobile units. Distrib. Parallel Databases, 7:257–387, 1999.

Moving Objects Interpolation

► Spatiotemporal Interpolation Algorithms

Moving Span

► Variable Time Span

MRR

► MRR (Mean Reciprocal Rank)

MRR1

► MRR (Mean Reciprocal Rank)

MSN Data Management

► Mobile Sensor Network Data Management

Multi-Database

► Distributed Architecture

Multidatabases

► Distributed Database Systems

Multidimensional Clustering

► Physical Database Design for Relational Databases

Multidimensional Data Formats

Amarnath Gupta
University of California San Diego, La Jolla, CA, USA

Definition

The term "multidimensional data" is used in two different ways in data management. In the first sense, it refers to data aggregates created by different groupings of relational data for on-line analytical processing. In the second sense, the term refers to data that can be described as arrays over heterogeneous data types together with metadata to describe them.

Example: HDF (Hierarchical Data Format) and NetCDF (network Common Data Form) are well known multidimensional data formats used in scientific applications.

Key Points

The goal of a multidimensional data format is to enable random access to very large, very complex, heterogeneous data, such that the data is self describing, sharable, compact, extendible, and archivable. For example, a composite of 900 files from a seismic simulation has been organized in HDF5 format to create a terabyte-sized dataset. One can mix tables, images, small metadata, streams of data from instruments, and structured grids all in the same HDF file. While multidimensional file formats are very flexible, they present the challenge of storing such large datasets and providing concurrent, random access to any part of the data required by user queries. Design of novel index structures over these formats is an area of active research.

Cross-references

► Bitmap-based Index Structures
► Query Evaluation Techniques for Multidimensional Data
► Storage of Large Scale Multidimensional Data

Recommended Reading

1. Home page of the HDF group. Available at: http://hdf.ncsa.uiuc.edu/
2. Home page of the NetCDF group. Available at: http://www.unidata.ucar.edu/software/netcdf/
3. Wu K., Otoo E.J., and Shoshani A. "An efficient compression scheme for bitmap indices". Technical Report LBNL-49626, Lawrence Berkeley National Laboratory, Berkeley, CA, 2002.

Multidimensional Database Management System

► Storage of Large Scale Multidimensional Data

Multi-dimensional Mapping

► Space Filling Curves
► Space-Filling Curves for Query Processing

Multidimensional Modeling

Torben Bach Pedersen
Aalborg University, Aalborg, Denmark

Synonyms

Dimensional modeling; Star schema modeling

Definition

Multidimensional modeling is the process of modeling the data in a universe of discourse using the modeling constructs provided by a multidimensional data model. Briefly, multidimensional models categorize data as being either *facts* with associated numerical *measures*, or as being *dimensions* that characterize the facts and are mostly textual. For example, in a retail business, *products* are sold to *customers* at certain *times* in certain *amounts* and at certain *prices*. A typical fact would be a *purchase*. Typical measures would be the amount and price of the purchase. Typical dimensions would be the location of the purchase, the type of product being purchased, and the time of the purchase. Queries then aggregate measure values over ranges of dimension values to produce results such as the total sales per month and product type.

Historical Background

Multidimensional databases do not have their origin in database technology, but stem from multidimensional matrix algebra, which has been used for (manual) data analyses since the late nineteenth century. During the late 1960s, two companies, IRI and Comshare, independently began the development of systems that later turned into multidimensional database systems. The IRI Express tool became very popular in the marketing analysis area in the late 1970s and early 1980s; it later turned into a market-leading OLAP tool and was acquired by Oracle. Concurrently, the Comshare system developed into System W, which was heavily used for financial planning, analysis, and reporting during the 1980s.

A concurrent development started in the early 1980s in the area of so-called *statistical data management* which focused on modeling and managing statistical data [1], initially within social science contexts such as census data. Many important concepts of multidimensional modeling such as *summarizability* (ensuring correct aggregate query results for complex

data) have their roots in this area. An overview is found in [14].

In 1991, Arbor was formed with the specific purpose of creating "a multiuser, multidimensional database server," which resulted in the Essbase system. Arbor, now Hyperion, later licensed a basic version of Essbase to IBM for integration into DB2. It was Arbor and Codd who in 1993 coined the term OLAP [2].

Another significant development in the early 1990s was the advent of large *data warehouses* [6] for storing and analyzing massive amounts of enterprise data. Data warehouses are typically based on relational *star schemas* or *snowflake schemas*, an approach to implementing multidimensional databases using relational database technology. The 1996 version of [6] popularized the use of star schema modeling for data warehouses.

From the mid 1990s and beyond, the introduction of the "data cube" operator [4] sparked a considerable research interest in the field of modeling multidimensional databases for use in data warehouses and *on-line analytical processing* (OLAP).

In 1998, Microsoft shipped its MS OLAP Server, the first multidimensional system aimed at the mass market. This has lead to the current situation where multidimensional systems are increasingly becoming commodity products that are shipped at no extra cost together with leading relational database systems.

A more in-depth coverage of the history of multidimensional databases is available in the literature [16]. Surveys of multidimensional data models can also be found in the literature [12,17].

Foundations

First, an overview of the concept of a multidimensional *cube* is given, then dimensions, facts, and measures are covered in turn.

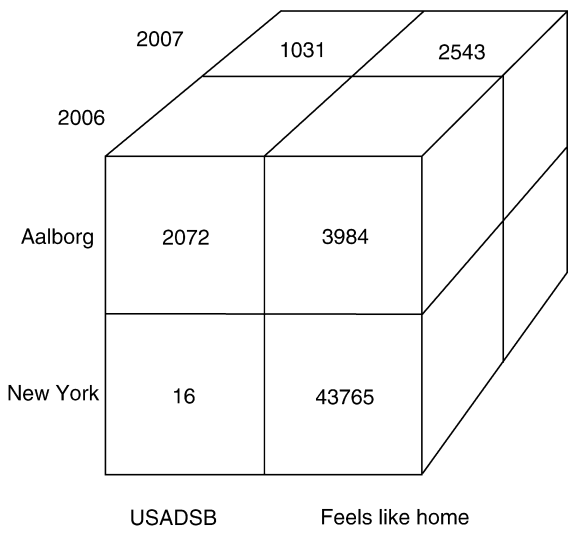
Data Cubes

Data cubes provide true multidimensionality. They generalize spreadsheets to any number of dimensions. 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*.

A dimensional cube for, e.g., CD sales can be obtained by including additional dimensions apart from just the album and the city where the album was sold. The most pertinent example of an additional dimension is a time dimension, but it is also possible to include other dimensions, e.g., an artist dimension that describes the artists associated with albums. In a cube, the combinations of a dimension value from each dimension define the *cells* of the cube. The actual sales counts are stored in the corresponding cells.

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 [6,16]. 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."

Figure 1 illustrates a three-dimensional cube based on the number of CD sales of two particular albums in Aalborg, Denmark, and New York, USA, for 2006 and 2007. The cube then contains sales counts for two cities, two albums, and two years. 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.



Multidimensional Modeling. Figure 1. Sales data cube.

A non-empty cell is called a *fact*. The example has a fact for each combination of time, album, and city where at least one sale was made. A fact has associated with it a number of *measures*. These are numerical values that "live" within the cells. In our case, there is only one measure, the sales count.

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, if the user wants to view just sales by City and Time, she aggregates over the entire dimension that characterizes the sales by Album for each combination of City and Time.

An important goal of multidimensional modeling is to "provide as much context as possible for the facts" [6]. The concept of *dimension* is the central means of providing this context. One consequence of this is a different view on *data redundancy* than in relational databases. In multidimensional databases, controlled redundancy is generally considered appropriate, as long as it considerably increases the information value of the data. One reason to allow redundancy is that multidimensional data is often *derived* from other data sources, e.g., data from a transactional relational system, rather than being "born" as multidimensional data, meaning that updates can more easily be handled [6]. However, there is usually no redundancy in the facts, only in the dimensions.

Having introduced the cube, its principal elements, dimensions, facts, and measures, are now described in more detail.

Dimensions

The notion of a dimension is an essential and distinguishing concept for multidimensional databases. Dimensions are used for two purposes: the *selection* of data and the *grouping* of data at a desired level of detail.

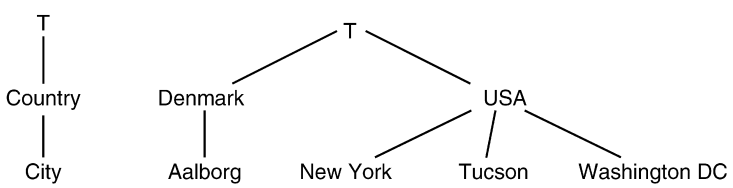
A dimension is organized into a containment-like *hierarchy* composed of a number of *levels*, each of which represents a level of detail that is of interest to the analyses to be performed. The instances of the dimension are typically called *dimension values*. Each such value belongs to a particular level.

In some cases, it is advantageous for a dimension to have *multiple hierarchies* defined on it. For example, a Time dimension may have hierarchies for both *Fiscal Year* and *Calendar Year* defined on it. Multiple hierarchies share one or more common lowest level(s), e.g., Day and Month, and then group these into multiple levels higher up, e.g., Fiscal Quarter and Calendar Quarter to allow for easy reference to several ways of grouping. Most multidimensional models allow multiple hierarchies. A dimension hierarchy is defined in the metadata of the cube, or the metadata of the multidimensional database, if dimensions can be shared.

In Fig. 2, the schema and instances of a sample *Location* dimension capturing the cities where CDs are sold are shown. The Location dimension has three levels, the City level being the lowest. City level values are grouped into *Country* level values, i.e., countries. For example, Aalborg is in Denmark. The $\top$ ("top") level represents *all* of the dimension, i.e., every dimension value is part of the $\top$ ("top") value.

In some multidimensional models, a level may have associated with it a number of *level properties* that are used to hold simple, non-hierarchical information. For example, the duration of an album can be a level property in the Album level of the Music dimension. This information could also be captured using an extra Duration dimension. Using the level property has the effect of not increasing the dimensionality of the cube.

Unlike the linear spaces used in matrix algebra, there is typically no ordering and/or distance metric on the dimension values in multidimensional models. Rather, the only ordering is the containment of



Multidimensional Modeling. Figure 2. Schema and instance for the location dimension.

lower-level values in higher-level values. However, for some dimensions, e.g., the Time dimension, an ordering of the dimension values is available and is used for calculating cumulative information such as "total sales in year to date."

Most models require dimension hierarchies to form *balanced trees*. This means that the dimension hierarchy must have uniform height everywhere, e.g., all departments, even small ones, must be subdivided into project groups. Additionally, direct links between dimension values can only go between immediate parent-child levels, and not jump two or more levels. For example, all cities are first grouped into states and then into countries, cities cannot be grouped directly under countries (as is the case in Denmark which has no states). Finally, each non-top value has precisely one parent, e.g., a product must belong to exactly one product group. Below, the relaxation of these constraints is discussed.

Facts

Facts are the objects that represent the *subject* of the desired analyses, i.e., the interesting "thing," or event or process, that is to be analyzed to better understand its behavior.

In most multidimensional data models, the facts are *implicitly* defined by their combination of dimension values. If a non-empty cell exists for a particular combination, a fact exists; otherwise, no fact exists. (Some other models treat facts as first-class objects with a separate identity [12].) Next, most multidimensional models require that each fact be mapped to precisely one dimension value at the lowest level in each dimension. Other models relax this requirement [12].

A fact has a certain *granularity*, determined by the levels from which its combination of dimension values are drawn. For example, the fact granularity in our example cube is "Year by Album by City." Granularities consisting of higher-level or lower-level dimension levels than a given granularity, e.g., "Year by Album Genre by City" or "Day by Album by City" for our example, are said to be *coarser* or *finer* than the given granularity, respectively.

It is commonplace to distinguish among three kinds of facts: *event* facts, *state* facts, and *cumulative snapshot* facts [6]. Event facts (at least at the finest granularity) typically model *events in the real world*, meaning that a unique instance, e.g., a particular sale of a given (particular physical instance of a) product in a given store at a given time, of the overall real-world process that is captured, e.g., sales for a supermarket chain, is represented by one fact. Examples of event facts include sales, clicks on web pages, and movement of goods in and out of (real) warehouses (flow).

A snapshot fact models the *state* of a given process at a given point in time. Typical examples of snapshot facts include the inventory levels in stores and warehouses, and the number of users using a web site. For snapshot facts, the same physical object, e.g., a specific physical instance of a can of beans on a shelf, with which the captured real-world process, e.g., inventory management, is concerned, may be "measured" at several time points, meaning that data related to that particular physical object will occur in several facts at different time points. This is unlike event facts, where a specific physical object such as a particular instance of a can of beans can only be sold once, and will thus only occur in one fact.

Cumulative snapshot facts are used to handle information about *a process up to a certain point in time*. For example, the total sales in the year to date may be considered as a fact. Then the total sales up to and including the current month this year can be easily compared to the figure for the corresponding month last year.

Often, all three types of facts can be found in a given data warehouse, as they support complementary classes of analyses. Indeed, the same base data, e.g., the movement of goods in a (real) warehouse, may often find its way into three cubes of different types, e.g., warehouse flow, warehouse inventory, and warehouse flow in year-to-date.

Measures

A *measure* has two components: a *numerical property* of a fact, e.g., the sales price or profit, and a *formula* (most often a simple aggregation function such as SUM) that can be used to combine several measure values into one. In a multidimensional database, measures generally represent the properties of the chosen facts that the users want to study, e.g., with the purpose of optimizing them.

Measures then take on different values for different combinations of dimension values. The property and formula are chosen such that the value of a measure is meaningful for all combinations of aggregation levels. The formula is defined in the metadata and thus not replicated as in the spreadsheet example. Most multidimensional data models provide the built-in

concept of measures, but a few models do not. In these models, dimension values are used for computations instead [12].

It is important to distinguish among three classes of measures, namely *additive*, *semi-additive*, and *non-additive* measures, as these behave quite differently in computations.

Additive measure values can be summed meaningfully along any dimension. For example, it makes sense to add the total sales over Album, Location, and Time, as this causes no overlap among the real-world phenomena that caused the individual values. Additive measures occur for any kind of fact.

Semi-additive measure values cannot be summed along one or more of the dimensions, most often the Time dimension. Semi-additive measures generally occur when the fact is of type snapshot or cumulative snapshot. For example, it does not make sense to sum inventory levels across time, as the same inventory item, e.g., a specific physical instance of an album, may be counted several times, but it is meaningful to sum inventory levels across albums and stores.

Non-additive measure values cannot be summed along any dimension, usually because of the chosen formula. For example, this occurs when averages for lower-level values cannot be summed into averages for higher-level values. Non-additive measures can occur for any kind of fact.

The Modeling Process

Now, the process to be carried out when doing multidimensional modeling is covered. One difference from "ordinary" data modeling is that the multidimensional modeler should not try to include all the available data and all their relationships in the model, but only those parts which are essential "drivers" of the business. Another difference is that redundancy may be ok (in a few, well-chosen places) if introducing redundancy makes the model more intuitive for the user. For example, time-related information may be stored in both a Calendar time dimension and a Fiscal Year time dimension, or specific customer info may be present both in a person-oriented Customer dimension or a group-oriented Demographics dimension.

Kimball [6,5] advocates a four-step process when doing multidimensional modeling.

1. Choose the business process(es) to model
2. Choose the grain of the business process
3. Choose the dimensions
4. Choose the measures

Step 1 refers to the facts that not all business processes may be equally important for the business. For example, in a supermarket, there are business processes for *sales* and *purchases*, but the sales process is probably the one with the largest potential for increasing profits, and should thus be prioritized. Step 2 says that data should be captured at the right grain, or granularity, compared to the analysis needs. For example, "individual sales items" may be captured, or perhaps (slightly aggregated) "total sales per product per store per day" may be precise enough, enabling performance and storage gains. Step 3 then goes on to refine the schema of each part of the grain into a complete dimension with levels and attributes. For the example above, a Store, a Product, and a Time dimension are specified. Finally, Step 4 chooses the numerical measures to capture for each combination of dimension values, for example dollar sales, unit sales, dollar cost, profit, etc.

When doing multidimensional modeling "in the large" for many types of data (many cubes) and several user groups, the most important task is to ensure that analysis results are comparable across cubes, i.e., that the cubes are somehow "compatible." This is ensured by (as far as possible) picking dimensions and measures from a set of common so-called "conformed" dimensions and measures [6,5] rather than "re-definining" the same concept, e.g., product, each time it occurs in a new context. New cubes can then be put onto the common "DW bus" [5] and used together. This sounds easier than it is, since it often requires quite a struggle with different parts of an organisation to define for example a common Product dimension that can be used by everyone.

Complex Multidimensional Modeling

Multidimensional data modeling is not always as simple as described above. A complexity that is almost always present is that of handling *change* in the dimension values. Kimball [6,5] calls this the problem of *slowly changing dimensions*. For example, customer addresses, product category names, and the way products are categorized may change over time. This must be handled to ensure correct results both for current and historical data. Kimball advises three types of slowly changing dimensions: Type 1 (overwrite previous value with current value), Type 2 (keep versions of

dimension rows), and Type 3 (keep previous and current value in different columns). Finally, the concept of *minidimensions* [6] advocates the separation of relatively static information (customer name, etc.) and dynamic information (income, number of kids, etc.) into separate dimensions.

The traditional multidimensional data models and implementation techniques assume that the data being modeled is quite regular. Specifically, it is typically assumed that all facts map (directly) to dimension values at the lowest levels of the dimensions and only to one value in each dimension. Further, it is assumed that the dimension hierarchies are simply balanced trees. In many cases, this is adequate to support the desired applications satisfactorily. However, situations occur where these assumptions fail.

In such situations, the support offered by "standard" multidimensional models and systems is inadequate, and more advanced concepts and techniques are called for. Now, the impact of irregular hierarchies on the performance enhancing technique known as partial, or practical, pre-computation, is reviewed.

Complex multidimensional data are problematic as they are not summarizable. Intuitively, data is *summarizable* if the results of higher-level aggregates can be derived from the results of lower-level aggregates. Without summarizability, users will either get wrong query results, if they base them on lower-level results, or the system cannot use pre-computed lower-level results to compute higher-level results. When it is no longer possible to pre-compute, store, and subsequently reuse lower-level results for the computation of higher-level results, aggregates must instead be calculated directly from base data, which leads to considerable increases in computational costs.

It has been shown that summarizability requires that aggregate functions be distributive and that the ordering of dimension values be *strict*, *onto*, and *covering* [12, 7]. Informally, a dimension hierarchy is *strict* if no dimension value has more than one (direct) parent, *onto* if the hierarchy is balanced, and *covering* if no containment path skips a level. Intuitively, this means that dimension hierarchies must be balanced trees. If this is not the case, some lower-level values will be either double-counted or not counted when reusing intermediate query results.

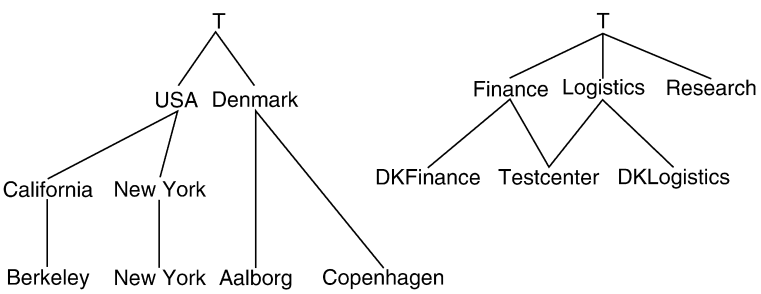
Figure 3 contains two dimension hierarchies: a Location hierarchy including a State level, and the hierarchy for the Organization dimension for some company. The hierarchy to the left is *non-covering*, as Denmark has no states. If aggregates at the State level are pre-computed, there will be no values for Aalborg and Copenhagen, meaning that facts mapped to these cities will not be counted when computing country totals.

To the right in figure 3, the hierarchy is non-onto because the Research department has no further subdivision. If aggregates are materialized at the lowest level, facts mapping directly to the Research department will not be counted. The hierarchy is also non-strict as the TestCenter is shared between Finance and Logistics. If aggregates are materialized at the middle level, data for TestCenter will be counted twice, for both Finance and Logistics, which is, in fact, what is desired at this level. However, this means that data will be double-counted if these aggregates are then combined into the grand total.

Several design solutions exist that aims to solve the problems associated with irregular hierarchies by altering the dimension schemas or hierarchies [8,11].

Key Applications

Multidimensional data models have three important application areas within data analysis. First, multidimensional models are used in *data warehousing*. Briefly, a data warehouse is a large repository of



Multidimensional Modeling. Figure 3. Irregular dimensions.

integrated data obtained from several sources in an enterprise for the specific purpose of data analysis. Typically, this data is modeled as being multidimensional, as this offers good support for data analyses.

Second, multidimensional models lie at the core of *on-line analytical processing* (OLAP) systems. Such systems provide fast answers to queries that aggregate large amounts of so-called detail data to find overall trends, and they present the results in a multidimensional fashion. Consequently, a multidimensional data organization has proven to be particularly well suited for OLAP. The widely acknowledged "OLAP Report" company [15] provides an "acid test" for OLAP by defining OLAP as "fast analysis of shared multidimensional information" (FASMI). In this definition, "Fast" refers to the expectation of response times that are within a few seconds, "Analysis" refers to the need for easy-to-use support for business logic and statistical analyses, "Shared" suggests a need for security mechanisms and concurrency control for multiple users, "Multidimensional" refers to the expectation that a data model with hierarchical dimensions is used, and "Information" suggests that the system must be able to manage all the required data and derived information.

Third, multidimensional data are increasingly becoming the basis for *data mining*, where the aim is to (semi-) automatically discover unknown knowledge in large databases. Indeed, it turns out that multidimensionally organized data are also particularly well suited for the queries posed by data mining tools.

Future Directions

A pressing need for multidimensional modeling is the aspect of standardization, i.e., agreeing on a common data model, a graphical notation for it, and support by tools. Also, better integration between ordinary "operational modeling" and multidimensional modeling is needed. Another future research line is the modeling of important system aspects such as security, quality, requirements, evolution, and interoperability [13]. This will be extended to also cover the modeling of business intelligence applications such as data mining, patterns, *extraction-transformation-loading* (ETL), *what-if analysis*, and business process modeling [13]. Finally, an important line of research will cover the modeling of more (complex) types of data, including integrating multidimensional data with text data, semi-structured/XML/web data and spatial/spatio-temporal/mobile data [9].

Cross-references

- ▶ Business Intelligence
- ▶ Cube
- ▶ Data Warehouse
- ▶ Data Warehouse Maintenance, Evolution and Versioning
- ▶ Data Warehousing Systems: Foundations and Architectures
- ▶ Dimension
- ▶ Hierarchy
- ▶ Measure
- ▶ On-Line Analytical Processing
- ▶ Statistical Data Management
- ▶ Summarizability
- ▶ What-If Analysis

Recommended Reading

1. Chan P. and Shoshani A. SUBJECT: A directory driven system for organizing and accessing large statistical databases. In Proc. 9th Int. Conf. on Very Data Bases, 1983, pp. 553–563.
2. Codd E.F. Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate. E.F. Codd and Assoc., 1993.
3. Dyreson C.E., Pedersen T.B., and Jensen C.S. Incomplete information in multidimensional databases, M. Rafanelli (ed.). Multidimensional Databases: Problems and Solutions. Idea Group Publishing, 2003.
4. Gray J., Chaudhuri S., Bosworth A., Layman A., Venkatrao M., Reichart D., Pellow F., Pirahesh H. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining Knowl. Dis., 1(1):29–54, 1997.
5. Kimball R. et al. The Data Warehouse Lifecycle Toolkit. Wiley, New York, 1998.
6. Kimball R. and Ross M. The Data Warehouse Toolkit, 2nd ed. Wiley, New York, 2002.
7. Lenz H. and Shoshani A. Summarizability in OLAP and statistical data bases. In Proc. 9th Int. Conf. on Scientific and Statistical Database Management, 1997, pp. 39–48.
8. Niemi T., Nummenmaa J., and Thanisch P. Logical multidimensional database design for ragged and unbalanced aggregation. In Proc. 3rd Int. Workshop on Design and Man of Data Warehouses, CEUR Workshop Proc. 39, 2001, Paper 7.
9. Pedersen T.B. Warehousing the world: a few remaining challenges. In Proc. ACM 10th Int. Workshop on Data Warehousing and OLAP, 2007, pp. 101–102.
10. Pedersen T.B. and Jensen C.S. Multidimensional database technology. IEEE Comput., 34(12):40–46, 2001.
11. Pedersen T.B., Jensen C.S., and Dyreson C.E. Extending practical pre-aggregation in on-line analytical processing. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 663–674.
12. 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.
13. Rizzi S., Abello A., Lechtenbrger J., and Trujillo J. Research in data warehouse modeling and design: dead or alive? In Proc.

ACM 9th Int. Workshop on Data Warehousing and OLAP, 2006, pp. 3–10.
14. Shoshani A. OLAP and statistical databases: similarities and differences. In Proc. 16th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1997, pp. 185–196.
15. The OLAP Report web page. http://www.olapreport.com. Current as of November 22, 2007.
16. Thomsen E. OLAP Solutions: Building Multidimensional Information Systems. Wiley, New York, 1997.
17. Vassiliadis P. and Sellis T.K. A survey of logical models for OLAP databases. ACM SIGMOD Rec., 28(4):64–69, 1999.

Multidimensional Scaling

Heng Tao Shen
The University of Queensland, Brisbane, QLD, Australia

Synonyms
MDS

Definition
Multidimensional scaling (MDS) is a mathematical dimension reduction technique that best preserves the inter-point distances by analyzing gram matrix. Given any two points p_i and p_j in a dataset P, MSD aims to minimize the following objective function:

$$Error = \sum [d(p_i, p_j) - d'(p_i, p_j)]^2$$

Where $d(p_i, p_j)$ and $d'(p_i, p_j)$represent the distance between points p_i and p_j in original space and the lower dimensional subspace respectively.

Key Points
Multidimensional scaling (MDS) is a set of related statistical techniques often used in data visualisation and analysis for exploring similarities or dissimilarities in data. An MDS algorithm starts with a matrix of point-point (dis)similarities, then assigns a location of each point in a low-dimensional space. The points are arranged in this subspace so that the distances between pairs of points have their original distance maximally retained. MDS is a generic term that includes many different specific types. These types can be classified according to whether the data are qualitative (called nonmetric MDS) or quantitative (metric MDS). The number of (dis)similarity matrices and the nature of the MDS model can also classify MDS types. This classification yields classical MDS (one matrix, unweighted model), replicated MDS (several matrices, unweighted model), and weighted MDS (several matrices, weighted model) [1,2].

MDS applications include scientific visualisation and data mining in fields such as cognitive science, information science, psychophysics, psychometrics, marketing and ecology [2].

Cross-references
- ▶ Dimensionality Reduction
- ▶ Discrete Fourier Transform
- ▶ Discrete Wavelet Transform and Wavelet Synopses
- ▶ Independent Component Analysis
- ▶ Isometric Feature Mapping
- ▶ Latent Semantic Indexing
- ▶ Locality-Preserving Mapping
- ▶ Locally Linear Embedding (Lle) Laplacian Eigenmaps
- ▶ Principal Component Analysis
- ▶ Semantic Subspace Projection

Recommended Reading
1. Cox M.F. and Cox M.A.A. Multidimensional Scaling. Chapman and Hall, 2001.
2. Young F.W. and Hamer R.M. Multidimensional Scaling: History, Theory and Applications. Erlbaum, New York, 1987.

Multidimensional Visualization

- ▶ Parallel Coordinates

Multi-Granularity Modeling

- ▶ Multiple Representation Modeling

Multi-Layered Architecture

- ▶ Multi-Tier Architecture

Multi-Level Recovery and the ARIES Algorithm

Gerhard Weikum
Max-Planck Institute for Informatics, Saarbrücken, Germany

Definition
In contrast to basic database recovery with page-level logging and redo/undo passes, *multi-level recovery* is

needed whenever the database system uses fine-grained concurrency control, such as index-key locking or operation-based "semantic" conflict testing, or when log records describe composite operations that are not guaranteed to be atomic by single page writes (as a consequence of concurrency control or for other reasons). Advanced methods perform logging and recovery at multiple levels like pages and data objects (records, index entries, etc.). Page-level recovery is needed to ensure the atomicity and applicability of higher-level operations, and also for efficient redo. Higher-level recovery is needed to perform correct undo for composite operations of loser transactions. In addition, logged actions at all levels must be testable at recovery-time, by embedding extra information in database pages, typically using log sequence numbers (LSNs), and appropriate logging of recovery steps. A highly optimized instantiation of these principles is the *ARIES algorithm* (Algorithms for Recovery and Isolation Exploiting Semantics) [8,7], the de facto standard solution for industrial-strength database systems. Its salient features are: very fast, potentially parallelizable or selective, redo for high availability; use of LSNs and compensation log records (CLRs) for tracking recovery progress; full-fledged support for crash and media recovery; suitability for all kinds of semantic concurrency control methods.

Historical Background

Crash recovery for composite operations on database records and indexes has first been addressed by [3], but that solution required heavy-weight check pointing (based on shadow storage) and was fairly inefficient. Some commercial engines developed various techniques to overcome these problems while supporting fine-grained concurrency control, but there is very little public literature on such system internals [1,2,4]. The ARIES algorithm was the first comprehensive solution [8] and has become the state-of-the-art method for industrial-strength recovery [7]. In parallel to and independently of the ARIES papers, research on multi-level recovery developed general principles and a systematic framework [9,10,6]. The textbook [11] discusses both the general framework and the ARIES algorithm in great detail. A correctness proof for the ARIES algorithm is given in [5].

Foundations

Basic database recovery methods log page modifications during normal operation. During the restart after a system crash (i.e., soft failure of the server with all disks intact), a three-phase recovery procedure is performed with an analysis pass, a redo pass, and an undo pass over the log file. This is appropriate when the database system uses page locking or some other page-granularity concurrency control. Such locking guarantees that all updates of loser transactions (i.e., uncommitted transactions that require undo) that are in conflict with updates of winner transactions (i.e., committed transactions that may require redo) follow the winners' updates in the log file. However, with fine-grained locking or some form of semantic concurrency control (e.g., exploiting commutative operations on hot-spot objects), this invariant does no longer hold and thus necessitates more advanced recovery algorithms.

As an example, consider the following execution history of two transactions t1 and t2 that insert various database records, with time proceeding from left to right and c2 denoting a successful commit of t2:

$$insert1(x) \quad insert1(y) \quad crash \quad insert2(z) \quad c2$$

The insertions may appear to be single operations, but they are not atomic from the system viewpoint. In fact, each of them may require multiple page writes to maintain indexes and other storage structures. For example, the following history of page writes may result from the above execution:

$$w1(p)\ w1(q)\ w1(s)\ w1(r)\ w1(p)\ w1(q)\ crash$$
$$w2(p)\ w2(s)\ c2$$

Here, p would be a data page into which all three records x, y, z are inserted, and q and s may be leaf pages of the same B + -tree index. For simplicity, the history does not show any read accesses like accesses for descending the tree. It may happen that the index update on behalf of record x triggers a leaf split of page q with a newly allocated page s and a corresponding update to the parent page r. The subsequent record operations may then access the new pages or the old page q, depending on where their corresponding keys now reside. This is a very normal situation for a database system, and it is perfectly admissible from a concurrency control viewpoint, because there are no (high-level) conflicts between the three insert operations. The system may have to use additional short-term locks or latches on pages to implement a multi-level concurrency control method, but this is very normal as well.

If the standard page-level recovery were applied to this situation, it would first redo (if necessary) all

page writes of winner transaction t2 and then undo all page writes of loser transaction t1, using before-images of pages or byte-range-oriented log records. This would lead to two kinds of problems:

- Redoing the page write w2(s) may be logically flawed and lead to an inconsistent index state if the effects of the preceding leaf split are not properly reflected in page s. But it is indeed possible that the loser updates w1(q) and w1(s) were not written to disk before the crash, and no redo would be performed for them.
- If all index updates were fully recovered and correctly captured in the database by the time the undo recovery for t1 takes place, undoing the write w1(s) of page s would restore the page as of the time before t1 started, thus accidentally – and incorrectly – eliminating also the index update of the winner transaction t2.

If, on the other hand, the logging and recovery procedures were changed so that only record- and index-level operations are captured and redone or undone, one would run into a third problem:

- If the system crashed in the middle of a high-level operation, say in between the w1(q) and the w1(s) steps, the database may be left in an inconsistent state with partial effects of an operation. Such a database would not be recoverable as all logged operations could face a state that they cannot properly interpret.

These problems show the need for multi-level recovery; the solution must meet the following requirements:

- *Operation atomicity:* High-level operations that comprise multiple page writes must be guaranteed to appear atomic (i.e., have an all-or-nothing impact on the database).
- *High-level undo:* Operations of loser transactions must, in full generality, be undone by means of inverse operations at the same level of abstraction. For example, the insertion of an index key must be undone by performing a delete operation on that key, not by restoring the before-images of the underlying pages (and neither by corresponding byte-range modifications).
- *Testable operations:* Before invoking a high-level operation for undoing or redoing the effect of a prior operation, it must be tested whether the effects of the prior operation are indeed present (as they may have been lost by the crash or already undone/redone in a recovery procedure that was interrupted by another crash). This testability is crucial for handling non-idempotent operations.
- *Efficient redo:* As the restart time and thus the unavailability of the system is usually dominated by the redo pass, it is crucial that the redo actions are performed as efficiently as possible. This strongly suggests performing redo in terms of page writes rather than re-executing high-level operations.

Multi-level recovery methods address these requirements in the following way:

- For *proper undo*, both page-level writes and higher-level operations are logged. The page-level log records guarantee that high-level operations can always be made to appear atomic. The high-level log records guarantee that undo can be performed by means of inverse operations. During the undo of a high-level operation, page-level logging is again enabled. This way, the first two requirements are satisfied.
- For *testable operations*, when the recovery procedure undoes a (high-level) operation, both the resulting page writes and a marker for the inverse operation itself are logged. The latter kind of log record is referred to as a compensation log record (CLR). In addition, the standard technique of maintaining log sequence numbers (LSNs) in the headers of modified pages, as a form of virtual time stamping, is used to be able to compare a log record to the state of a page and decide whether the logged action should be undone/redone or disregarded.
- For *efficient redo*, although redoing high-level operations may add to the repertoire of recovery actions, it is much more desirable to perform all redo steps in terms of page writes. This can leverage all kinds of acceleration techniques that have been developed for more conventional, page-level recovery like asynchronous check pointing and dirty-pages bookkeeping, smart scheduling of page reads from the database disk, parallelized per-page redo, and selective redo for pages with very high availability demands.
- Further considerations on the redo pass lead to the *repeating-history principle* [8]: rather than aiming to redo only winner updates or at least as few loser updates as possible, it is much simpler to redo all

logged page writes regardless of their transaction status, thus effectively reconstructing the database as of the time of the crash.

All these principles together result in the following algorithmic template for multi-level recovery:

- Analysis pass for determining loser transactions.
- Redo pass by repeating history in terms of logged page writes.
- Undo pass for loser transactions, with page-level undo for incomplete high-level operations and high-level undo for complete (and possibly just redone) high-level operations. Logging at both levels is enabled during the undo pass, thus creating new log records: page-write log records during the operation's execution, and undo information for the entire operation at the very end, thus also marking the completion of the operation.

For all steps, *idempotence* is ensured by two means: for page writes the standard comparison of page-header LSN versus log-record LSN is performed; for high-level operations, only undo idempotence is a potential issue, and this is guaranteed by the fact that the preceding redo pass always repeats history so that all completely repeated operations need subsequently be undone by definition.

The *ARIES algorithm* is an integrated and highly optimized instantiation of these principles, with various additional features. Its recovery procedure performs three passes over the log: analysis, repeating-history redo, and undo. The analysis pass mostly follows standard recovery methods; the redo pass has been discussed above; the undo pass uses additional techniques based on the use of *compensation log records (CLRs)*. The following undo-relevant log records are produced by ARIES:

- During normal operation, page writes are logged in a way that they can be redone or undone (whichever is needed later), and each high-level operation is logged for undo purposes following all page-write log records that were produced during the operation's execution. The high-level log records have a backward pointer that points to the preceding high-level action of the same transaction, thus allowing the recovery manager to skip the operation's logged page writes.
- During the undo pass, when undoing a page write, a CLR is written with a backward pointer to the log record that precedes the undo page write within the same transaction. When undoing a high-level operation, normal page-write log records are written during the execution of the inverse operation, and a CLR for the entire high-level operation is written at the end. That CLR again has a backward pointer to its preceding high-level action, skipping its own page writes.

With these preparations, the undo procedure itself is rather straightforward. For each loser transaction, it locates the most recent log record and then follows the backward chain of log records. Whenever a CLR is encountered, this tells the recovery manager that the undo of the corresponding action is already completed (either already during normal operation or by the preceding redo pass) and the log record should thus be disregarded. Page-write log records are relevant for incomplete high-level operations; otherwise high-level log records determine the undo logic.

This undo procedure of the ARIES algorithm has a number of great benefits:

- It handles high-level undo in a correct and efficient way, thus allowing *fine-grained and semantic concurrency control*.
- It handles *nested rollbacks* in a correct and efficient way. These are situations where a transaction rollback is interrupted by a crash and later considered for undo or when the undo pass after a server crash is interrupted by a second crash. In all these situations, it is guaranteed that the amount of recovery work stays bounded, regardless of how many "nested" crashes might occur during recovery. This is important for high availability.
- For *media recovery*, restoring the database after disk failures, an analogous but even more severe situation arises. As media recovery always starts with a backup copy of the database and then repeats the history of a potentially very long archive log, rollbacks or undo steps for (soft) system crashes that happened long ago would interfere with log truncation and become performance showstoppers with pre-ARIES recovery methods. The way ARIES generates redo log records for undo actions and CLRs for progress tracking, media recovery is as fast as possible, which is crucial for availability.

For the example scenario given above, ARIES would create the following log records during normal

operation, denoted in the form *LSN:action*. Note that log records 5 and 8 will only be used for undo purposes (if necessary). Further note that the example happens to show page-level log records for the second insert operation of t1 but no high-level log record. This may occur because of the crash happening before the high-level log record was flushed to the log disk.

1 : w1(p) 2 : w1(q) 3 : w1(s) 4 : w1(r)

5 : insert1(x) 6 : w2(p) 7 : w2(s) 8 : insert2(z)

9 : w1(p) 10 : w1(q) 11 : c2

During recovery, the redo pass processes log records 1, 2, 3, 4, 6, 7, 9, and 10. Subsequently the undo pass processes log records 10, 9, and 5 (in this – chronologically reverse – order). As it does so, it will create the following new log records, with CLRs denoted in the form *LSN:action→UndoNextLSN* with UndoNextLSN being the LSN of the log record to which the CLR has a backward pointer.

12 : w1(q) → 9 13 : w1(p) → 5 14 : w1(s)

15 : w1(p) 16 : delete1(x) → 0

If the system crashed again immediately after the completion of the delete1(x) undo step, the redo pass would repeat the page writes with LSNs 12, 13, 14, and 15 (in addition to all writes with LSNs 1 through 11 that may need redo again). This means that all effects of t1 have been properly removed. The subsequent undo pass would then encounter the CLR 16, but its backward pointer immediately tells the recovery manager that it can skip all log records of transaction t1 as t1 had already been completely undone before the second crash.

If the system crashed again after the action with LSN 14 (a page write issued on behalf of the high-level undo of insert1(x)), the redo pass would repeat the page writes with LSNs 12, 13, and 14, thus effectively removing all effects of insert1(y) but only some partial effects of insert1(x). The subsequent undo pass would start with LSN 14, undo it and create a new CLR, and then encounter LSN 13, which points to LSN 5 which in turn is the next logged action to undo.

In general, ARIES can be implemented with very low overhead, and it is compatible with other optimizations in the storage engine of a database system: flexible free space management, flexible buffer management, acceleration techniques for the redo pass, and many more. For *high availability*, the redo pass of both crash and media recovery can be parallelized or performed selectively for most important page sets; media recovery efficiently works also with fuzzy backups without ever quiescing the system. For *index management*, extensions of ARIES have been developed that optimize the locking, logging, and recovery of index keys in B + -trees. Finally, there are also extensions of ARIES for the special requirements of *shared-disk clusters* with automated fail-over procedures and very high availability.

Future Directions

The ARIES algorithm is a mature and comprehensive solution that can be readily adopted for most data management systems. A salient property of the multi-level recovery framework is that it can be generalized to arbitrary kinds of composite operations (with deeper and flexible nestings). All the ARIES techniques for efficient repeating-history redo are directly applicable, and the undo procedures need to be extended to handle a conceptual stack of undo log records and corresponding CLRs – with the stack actually being embedded in the linear log. This generalization is of potential interest for modern applications like composite Web services or enterprise-level middleware with integrated recovery.

Cross-references

- ▶ Atomicity
- ▶ Logging
- ▶ Persistence
- ▶ System Recovery
- ▶ Transaction

Recommended Reading

1. Borr A.J. Robustness to crash in a distributed database: a non shared-memory multi-processor approach. In Proc. 10th Int. Conf. on Very Large Data Bases, 1984, pp. 445–453.
2. Crus R.A. Data recovery in IBM database 2. IBM Syst. J., 23(2):178–188, 1984.
3. Gray J., McJones P.R., Blasgen M.W., Lindsay B.G., Lorie R.A., Price T.G., Putzolu G.R., and Traiger I.L. The recovery manager of the system R database manager. ACM Comput. Surv., 13(2):223–243, 1981.
4. Gray J. and Reuter A. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA, 1993.
5. Kuo D. Model and verification of a data manager based on ARIES. ACM Trans. Database Syst., 21(4):427–479, 1996.

6. Lomet D.B. MLR: a recovery method for multi-level systems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1992, pp. 185–194.
7. Mohan C. Repeating history beyond ARIES. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 1–17.
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. Moss J.E.B., Griffeth N.D., and Graham M.H. Abstraction in recovery management. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1986, pp. 72–83.
10. Weikum G., Hasse C., Brössler P., and Muth P. Multi-level recovery. In Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1990, pp. 109–123.
11. Weikum G. and Vossen G. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann, San Francisco, CA, 2001.

Multilevel Secure Database Management System

Bhavani Thuraisingham
University of Texas at Dallas, Richardson, TX, USA

Synonyms

Secure database systems; Trusted database systems

Definition

Many of the developments in the 1980s and 1990s in database security were on multi-level secure database management systems (MLS/DBMS). These systems were also called trusted database management systems (TDBMS). In a MLS/DBMS, users are cleared at different clearance levels such as Unclassified, Confidential, Secret and TopSecret. Data is assigned different sensitivity levels such as Unclassified, Confidential, Secret, and TopSecret. It is generally assumed that these security levels form a partially ordered lattice. For example, Unclassified $<$ Confidential $<$ Secret $<$ TopSecret. Partial ordering comes from having different compartments. For example, Secret Compartment A may be incomparable to Secret Compartment B.

Historical Background

MLS/DBMSs have evolved from the developments in multilevel secure operating systems such as MULTICS and SCOMP (see for example [4]) and the developments in database systems. Few developments were reported in the late 1970s on MLS/DBMSs. However, during this time there were many developments in discretionary security, such as access control for System R and INGRES as well as many efforts on statistical database security. Then there was a major initiative by the Air Force and a summer study was convened. This summer study marks a significant milestone in the development of MLS/DBMSs [2].

The early developments in MLS/DBMSs influenced the Air Force Summer Study a great deal. Notable among these efforts are the Hinke-Schaefer approach to operating system providing mandatory security, the Ph.D. Thesis of Deborah Downs at UCLA (University of California at Los Angeles), the IP Sharp Model developed in Canada and the Naval Surveillance Model developed at the MITRE Corporation. The Hinke Schaefer approach [3] essentially developed a way to host MLS/DBMSs on top of the MULTICS MLS operating system. The system was based on the relational system and the idea was to partition the relation based on attributes and store the attributes in different files at different levels. The operating system would then control access to the files. The early efforts showed a lot of promise to designing and developing MLS/DBMSs. As a result, the Air Force started a major initiative, which resulted in the summer study of 1982

Since the summer study, several efforts were reported throughout the 1980s. Many of the efforts were based on the relational data model. At the end of that decade, the National Computer Security Center started a major effort to interpret the Trusted Computer Systems Evaluation Criteria for database systems [7]. This interpretation was called the Trusted Database Interpretation [8]. In the 1990s research focused on non-relational systems including MLS object database systems and deductive database systems. Work was also carried out on multilevel secure distributed database systems. Challenging research problems such as multilevel data models, inference problem and secure transaction processing were being investigated. Several commercial products began to emerge. Since the late 1990s, while the interest in MLS/DBMSs began to decline a little, efforts are still under way to examine multilevel security for emerging data management technologies. A detailed discussion of many of the developments with significant references are given in [5].

Foundations

Many of the developments were based on the relational model. The early systems were based on the Integrity

Lock approach developed at the MITRE Corporation. Two prototypes were designed and developed. One used the MISTRESS relational database system and the other used the INGRES relational database system. Around 1985 TRW designed and developed a MLS/DBMS called ASD and this system was designed to be hosted on ASOS (the Army Secure Operating System). The approaches were based on the Trusted Subject based architecture. Later on TRW developed some extensions to ASD and the system was called ASD Views where access was granted on views (GARV88). Two of the notable systems designed in the late 1980s were the SeaView system by SRI International and LOCK Data Views system by Honeywell. These two efforts were funded by the then Rome Air Development Center and the goal was to focus on the longer term approaches proposed by the Summer Study. Both efforts influenced the commercial developments a great deal. Three other efforts worth mentioning are the SINTRA system developed by the Naval Research Laboratory, the SWORD system developed by the then Defense Research Agency and funded by the Ministry of Defense in the United Kingdom and the SDDBMS effort by Unisys. The SINTRA system was based on the distributed architecture proposed by the Air Force Summer Study. The SWORD system proposed some alternatives to the SeaView and LOCK Data Views data models. While the initial planning for these systems began in the late 1980s, the designs were actually developed in the early 1990s. The SDDBMS effort was funded by the Air Force Rome Laboratory and investigated both the partitioned and replicated approaches to designing an MLS/DBMS.

Around l987, the Rome Air Development Center (now known as Air Force Research Laboratory in Rome) funded an effort to design an MLS/DBMS based on the Entity Relationship (ER) model. The ER model was initially developed in 1976 by Peter Chen and since then it has been used extensively to model applications. The goal of the security effort carried out by Gajnak and his colleagues was to explore security properties for the ER model as well as to explore the use of secure ER models to design DBMSs. The effort produced MLS ER models that have since been used to model secure applications. Furthermore variations of this model have been used to explore the inference problem by Burns, Thuraisingham and Smith. However, there does not appear to have been any efforts undertaken on designing MLS/DBMSs based on the ER model. In summary, the ER approach has contributed extensively toward designing MLS applications.

During the late 1980s, efforts began on designing MLS/DBMSs based on object models. Notable among these efforts is the one by Keefe, Tsai and Thuraisingham who designed the SODA model by Keefe and his colleagues. Later Thuraisingham designed the SORION and SO2 models. These models extended models such as ORION and O2 with security properties. Around 1990 Millen and Lunt produced an object model for secure knowledge base systems. Jajodia and Kogan developed a message-passing model in 1990. Finally MITRE designed a model called UFOS. Designs of MLS/DBMSs were also produced based on the various models. The designs essentially followed the designs proposed for MLS/DBMSs based on the relational model. However with the object model, one had to secure complex objects as well as handle secure method execution. While research progressed on designing MLS/DBMSs based on objects, there were also efforts on using object models for designing secure applications. Notable efforts were those by Sell and Thuraisingham. Today with the development of UML (Unified Modeling Language) there are efforts to design secure applications based on UML.

Around 1989 work began at MITRE on the design and development of multilevel secure distributed database systems (MLS/DDBMS). Prototypes connecting MLS/DBMSs at different sites were also developed. Work was then directed toward designing and developing MLS heterogeneous distributed database systems. These efforts focused on connecting multiple MLS/DBMSs, which are heterogeneous in nature. Research was also carried out on MLS federated databases by Thuraisingham and Rubinovitz.

In the late 1970 and throughout the 1980s there were many efforts on designing and developing logic-based database systems. These systems were called deductive databases. While investigating the inference problem, multilevel secure deductive database systems were designed. These systems were based on a logic called NTML (Non monotonic Typed Multilevel Logic) designed by Thuraisingham at MITRE. NTML essentially provides the reasoning capability across security levels, which are non-monotonic in nature. Essentially, it incorporates constructs to reason about the applications at different security levels. A Prolog language based on NTML, which is called NTML-Prolog, was also designed. Both reasoning with the Closed World

Assumption as well as with the Open World Assumption were investigated. Due to the fact that there was limited success with logic programming and the Japanese Fifth Generation Project, deductive systems are being used only for a few applications. If such applications are to be multilevel secure, then systems such as those based on NTML will be needed. Nevertheless there is use for NTML on handling problems such as the inference problem. Note that presently the integration of NTML-like logic with descriptive logics for secure semantic webs is being explored.

Researchers have identified several hard problems. The most notable hard problem is the Inference problem. Inference problem is the process of posing queries and deducing sensitive information form the legitimate responses received. Many efforts have been discussed in the literature to handle the inference problem First of all, Thuraisingham proved that the general inference problem was unsolvable [6] and this effort was stated by Dr. John Campbell of the National Security Agency as one of the significant developments in database security in [1]. Then Thuraisingham explored the use of security constraints and conceptual structures to handle various types of inferences. Note that the aggregation problem is a special case of the inference problem where collections of data elements are sensitive while the individual data elements are Unclassified. Another hard problem is secure transaction processing. Many efforts have been reported on reducing covert channels when processing transactions in MLS/DBMSs including the work of Jajodia, Bertino and Atluri among others. A third challenging problem is developing a multilevel secure relational data model. Various proposals have been developed including those by Jajodia and Sandhu, the Sea View model by Denning and her colleagues and the LOCK Data Views model by Honeywell. SWORD developed by Wiseman also proposed its own model. The problem is due to the fact that different users have different views of the same element. If multiple values are used to represent the same entity then the integrity of databases is violated. However, if what is called polyinstantiation is not enforced, then there is a potential for signaling channels. This is still an open problem.

Key Applications

The department of defense was the major funding agency for multilevel secure database management systems. The applications are mainly in the defense and intelligence area. However many of the concepts can be used to design systems that have multiple labels, privacy levels or roles. Therefore these systems can also be used to a limited extent for non-defense applications including healthcare and financial applications.

Future Directions

As technologies emerge, one can examine multilevel security issues for these emerging technologies. For example, as object database systems emerged in the 1980s, multilevel security for object databases began to be explored. Today there are many new technologies including data warehousing, e-commerce systems, multimedia systems, real-time systems and the web and digital libraries. Only a limited number of efforts have been reported on investigating multilevel security for the emerging data management systems. This is partly due to the fact the even for relational systems, there are hard problems to solve with respect to multilevel security. As the system becomes more complex, developing high assurance multilevel systems becomes an enormous challenge. For example, how can one develop usable multilevel secure systems say for digital libraries and e-commerce systems? How can one get acceptable performance? How does one verify huge systems such as the World Wide Web? At present, there is still a lot to do with respect to discretionary security for such emerging systems. As progress is made with assurance technologies and if there is a need for multilevel security for such emerging technologies, then research initiatives will commence for these areas.

Cross-references

- ▶ Database Security
- ▶ Inference Problem
- ▶ Mandatory Access Control
- ▶ Role Based Access Control

Recommended Reading

1. Campbell J. A year of progress in database security. In Proc. National Computer Security Conf., 1990.
2. Committee on Multilevel Data Management Security, Air Force Studies Board. Multilevel Data Management Security. National Academy Press, Washington, DC, 1983.
3. Hinke T. and Schaefer M. Secure data management system. System Development Corp., Technical Report RADC-TR-75-266, November 1975.
4. IEEE Computer Magazine, Volume 16, #7, 1983.
5. Thuraisingham B. Database and Applications Security: Integrating Data Management and Information Security. CRC Press, Boca Raton, FL, 2005.

6. Thuraisingham B. Recursion theoretic properties of the inference problem. Presented at the IEEE Computer Security Foundations Workshop, Franconia, NH, June 1990 (also available as MITRE technical Paper MTP291, June 1990).
7. Trusted Computer Systems Evaluation Criteria, National Computer Security Center, MD, 1985.
8. Trusted Database Interpretation. National Computer Security Center, MD, 1991.

Multilevel Security

► Mandatory Access Control

Multilevel Transactions and Object-Model Transactions

Gerhard Weikum
Max-Planck Institute for Informatics, Saarbrücken, Germany

Synonyms

Layered transactions; Open nested transactions

Definition

Multilevel transactions are a variant of nested transactions where nodes in a transaction tree correspond to executions of operations at particular levels of abstraction in a layered system architecture. The edges in a tree represent the implementation of an operation by a sequence (or partial ordering) or operations at the next lower level. An example instantiation of this model are transactions with record and index-key accesses as high-level operations which are in turn implemented by reads and writes of database pages as low-level operations. The model allows reasoning about the correctness of concurrent executions at different levels, aiming for serializability at the top level: equivalence to a sequential execution of the transaction roots. This way, semantic properties of operations, like different forms of commutativity, can be exploited for higher concurrency, and correctness proofs for the corresponding protocols can be derived. Likewise, multilevel transactions provide a framework for structuring recovery methods and reasoning about their correctness.

Multilevel transactions have wide applications outside of database engines as well. For example, transactional properties can be provided by middleware application servers, layered on top of a database system. A generalization of this approach is the notion of object-model transactions, also known as open nested transactions (trees where the nodes correspond to arbitrary method invocations). In contrast to multilevel transactions, there is no layering constraint anymore, and arbitrary caller-callee relations among objects of abstract data types can be expressed. This provides a model for reasoning about transactional guarantees in composite web services, and for structuring the design and run-time architecture of web-service-based applications.

Historical Background

Object-model transactions have been around for thirty years, going back to the work of Bjork and Davies on "spheres of control" [2]. The first work that made these concepts explicit and gave formal definitions is by Beeri et al. [1]. Parallel work on the important special case of multilevel transactions has been done by Moss et al. [7] and Weikum et al. [13,11]. The textbook [14] gives a detailed account of both conceptual and practical aspects. A broader perspective of extended transaction models is given by [9]. More recently, the concept of object-model transactions has received considerable attention also for long-running workflows (e.g., [10,15]) and transactional memory (e.g., [8]).

On the system side, multilevel transaction protocols have been employed for concurrency control and recovery in various products and prototypes [3,5,6,12]. Typically, the layered structure of the protocols is only implicit; a suite of additional smart implementation techniques is used for integrated, highly efficient code. An example for concurrency control is transaction-duration locking for index-manager operations combined with operation-duration latching. Examples for recovery are the ARIES family of algorithms by Mohan et al. [6] and the MLR algorithm by Lomet [5].

Foundations

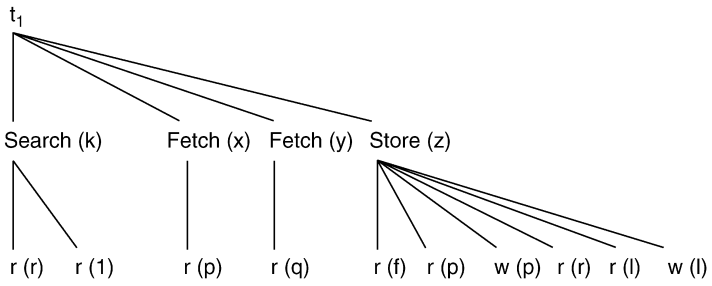
Multilevel and object-model transactions are best understood in an object model where operations are invoked on arbitrary objects. This allows exploiting "semantic" properties of the invoked operations for the sake of improved performance. This model also captures situations where an operation on an object invokes other operations on the same or other objects. Often the implementation of an object and its

operations requires calling operations of some lower-level types of objects.

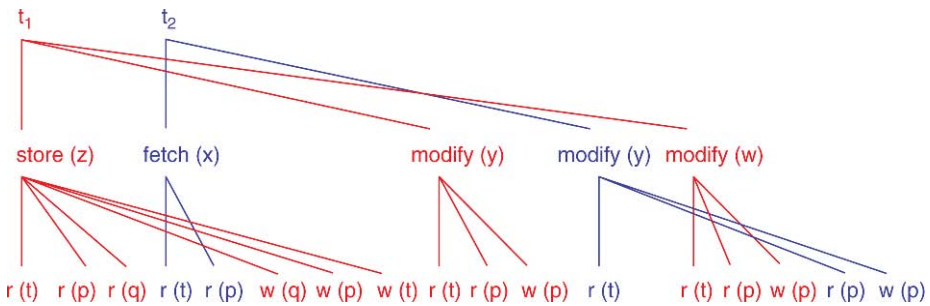
For example, operations at the access layer of a database system, such as index searches, need to invoke page oriented operations at the storage layer underneath. Similar invocation hierarchies may exist among a collection of business objects that are made available as abstract data type (ADT) instances within a data server or an application server, e.g., a "shopping cart" or a "bank account" object type along with operations like deposit, withdraw, get_balance, get_history, compute_interests, etc. The following figure depicts an example of a transaction execution against an object model scenario that refers to the internal layers of a database system.

The figure shows a transaction, labeled t1, which performs, during its execution, (i) an SQL Select command to retrieve all records from a database that satisfy a certain attribute-value condition, and, after inspecting the result set, (ii) an SQL command to insert a record for a new record with this attribute value. Since SQL commands are translated into query execution plans already at compile time, the operations invoked at run time refer to an internal level of index and record accesses. The Select command is executed by first issuing a Search operation with some key k on an index that returns the RIDs (i.e., addresses) of the result records. Next, these records, referred to as x and y in the figure, are fetched by dereferencing their RIDs. The Search operation in turn invokes operations at the underlying storage layer: read and write operations on pages. First the root of a B+ tree is read, labeled as page r in the figure, which points to a leaf page, labeled l, that contains the relevant RID list for key k. The subsequent Fetch operations to access the two result records x and y by their RIDs, require only one page access each to pages p and q, respectively. Finally, the SQL Insert command is executed as a Store operation, storing the new record and also maintaining the index. This involves first reading a metadata page, labeled f that holds free space information in order to find a page p with sufficient empty space. Then that page is read and subsequently written after the new record z has been placed in the page. Finally, the RID of the new record z is added to the RID list of the key k in the index. This requires reading the B+ tree root page r, reading the proper leaf page l, and finally writing page l after the addition of the new RID.

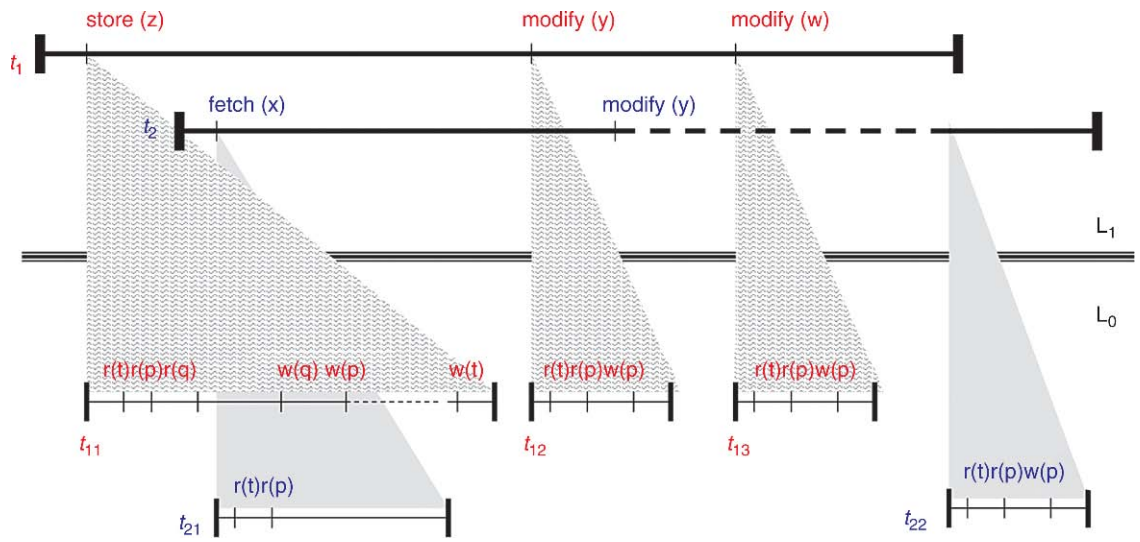
This entire execution is represented in a graphical, compact form, by connecting the calling operation and the called operation with an arc when operations invoke other operations. As a convention, the caller is



Multilevel Transactions and Object-Model Transactions. Figure 1. Example of a multilevel transaction.



Multilevel Transactions and Object-Model Transactions. Figure 2. Example of a multilevel schedule.



Multilevel Transactions and Object-Model Transactions. Figure 3. Concurrent execution of multilevel schedules with a multilevel locking protocol.

always placed closer to the top of the picture than the callee. Furthermore, the order in which operations are invoked is represented by placing them in "chronological" order from left to right, which suffices for illustration purposes.

More formally, a ***multilevel transaction tree*** is defined as a partially ordered labeled tree with node labels being operation invocations and the leaf nodes denoting elementary (i.e., indivisible) read and write operations. Moreover, the tree must be perfectly balanced with all leaves having the same distance from the root; this constraint is dropped for the more general case of ***object-model transactions (open nested transactions)***. Finally, a constraint is imposed on conflicting leaf nodes to be totally ordered so that no concurrent write-write or read-write pair of operations is possible on the same elementary object; for all other cases partial orders are allowed. It is important to note that transaction trees model executions and not programs. Thus, node labels are method names along with concrete input parameter values (and possibly even output parameter values if these are exploited in the reasoning about concurrency); edges denote the dynamic calling structure and not a static hierarchy.

A ***concurrent execution of several transaction trees***, referred to as a multilevel schedule, is essentially an interleaved forest of the individual transaction trees. This is illustrated in the figure below, for two transactions with record-level and page-level operations, in the same spirit as the previous example but with some simplifications. Like before, the ordering of operations is indicated by drawing the leaf nodes in their execution order from left to right (assuming total ordering of leaves for simplicity). As the caller-callee relationship in transaction trees is captured by vertical or diagonal arcs, the crossing of such arcs indicates that two (non-leaf) operations are concurrent. In the figure the two transactions t1 and t2 are concurrent, and also the store and fetch operations execute concurrently and the same holds for the last two modify operations.

To reason about the correctness of such interleavings, it is first necessary to define the ordering of non-leaf operations: node o1 precedes o2 in the execution, $o1 < o2$, if all leaf-level descendants of o1 precede all leaf-level descendants of o2. The forest of labeled trees and this execution order $<$ define a ***multilevel schedule***, more generally, a schedule of transaction trees.

Following the standard argumentation about serializability for conventional, "flat" transactions, the goal for a correct schedule is to show that the execution is equivalent to a sequential one based on a notion of ***conflicting versus non-conflicting operations***. Usually, ***commutativity*** properties of operations are the basis for defining conflict relations. In the example, this suggests that store and fetch operations on different objects as well as pairs of modify operations on different objects are non-conflicting (even if their implementations write the same page). Thus, their

observed execution order could be changed, by swapping adjacent operations, without changing the overall effect of the schedule. But applying this principle to, for example, the store and fetch operations in the above schedule does not work because these two nodes are composite operations (i.e., non-leaf nodes) and executed concurrently among themselves. To disentangle the concurrency between these operations, one needs to reason about the execution ordering of their children, and in an actual system, one would need a ***lower-level concurrency control*** mechanism that treats the two operations as ***subtransactions***. The goal of this disentangling, the counterpart to serial schedules in conventional concurrency control theory, are ***isolated subtrees*** for the two operations. A subtree rooted at node o is isolated if there is a total ordering among all its leaf-level descendants and o either precedes or follows all other operations o' that are not among its descendants ($o < o'$ or $o' < o$). Once a subtree is isolated, the fact that its root is a composite operation is no longer important, and it is possible, for reasoning about equivalent executions, to ***reduce an isolated subtree*** to its root alone. This argument abstracts from the lower-level executions, as they are now (shown to be equivalent to) sequential.

Putting everything together, the above considerations lead to three rules for transforming a multilevel schedule into equivalent and abstracted executions, ideally leading to a sequential execution of the transaction roots:

- *Commutativity rule:* The order of two ordered leaf operations p and q with, say, the order $p < q$, can be reversed provided that
  - both are isolated, adjacent in that there is no other operation r with $p < r < q$, and commutative,
  - the operations belong to different transactions, and
  - the operations p and q do not have ancestors, say p' and q', respectively, which are non-commutative and totally ordered (in the order $p' < q'$).
- *Ordering rule:* Two unordered leaf operations p and q can be (arbitrarily) ordered, i.e., assuming either $p < q$ or $q < p$, if they are commutative.
- *Tree pruning rule:* An isolated subtree can be pruned and replaced by its root.

A schedule that, by applying the above rules, can be transformed into a sequential execution of the transaction roots is called ***tree-reducible*** or ***multilevel serializable***. Note that this notion of multilevel serializability is much more liberal than the conventional notion of read-write-oriented serializability. The example schedule shown above is not serializable at the leaf level of read and write operations (i.e., if one ignored the level of search, fetch, store, and modify operations and simply connected all leaves directly to the roots), but these seemingly non-serializable effects on the low-level storage structures are irrelevant as long as they are properly handled within the scope of their parent operations and new transactions access the data through the higher-level operations like search, fetch, store, and modify.

The example schedule depicted above is multilevel serializable. It can be reduced as follows. First the two reads of the fetch(x) operation are commuted with their left-hand neighbors so that fetch(x) completely precedes store(z); analogously the r(t) step of modify (y) is commuted with its right-hand neighbors, the children of modify(w), so that modify(w) completely precedes modify(y). This establishes a serial order of the record-level operations, all of them now being isolated subtrees. This enables the application of the pruning rule to remove all page-level operations. Next, the fetch(x) operation of t2 is commuted with t1's store(z), modify(y), and modify(w) operations all the way to the right, producing an order where all of t1's operations precede all of t2's operations. This turns t1 and t2 into isolated subtrees. Finally, pruning the operations of t1 and t2 produces the sequential order of the transaction roots: $t1 < t2$.

The transformation rules do not directly lead to an efficient concurrency control protocol. Rather their purpose is to prove the correctness of protocols. But for the case of a layered system, the way the example was handled points towards a practically viable protocol. The key is to consider pairs of adjacent levels and apply the transformation rules in a bottom-up manner. So first, the commutativity and ordering rules are used to establish a sequential execution of the parent nodes of the leaf-level nodes, then these isolated parents are reduced. Then, with the lowest level removed, this procedure is iterated through the levels until the roots of the entire transaction trees are isolated. This proof strategy can be directly turned into a protocol by enforcing conventional order-preserving

conflict-serializability (OPCSR) for each pair of adjacent levels. Any protocol for OPCSR can be used, and even different protocols for different level pairs are possible. The most widely used protocol, two-phase locking, is often a natural choice, and then forms the following ***multilevel locking protocol***:

- *Lock acquisition rule:* When an operation f(x) is issued, an f-mode lock on x needs to be acquired before the operation can start its execution.
- *Lock release rule:* Once a lock originally acquired by an operation f(x) with parent o (an operation at the next higher level) is released, no other descendant of o is allowed to acquire any locks.
- *Subtransaction rule:* At the termination of an operation o, all locks that have been acquired for descendants of o are released, thus treating o as a committed subtransaction. Note that the o-lock for o itself is still kept – until the parent of o terminates. The releasing of lower-level locks at the end of a subtransaction is the origin of the name "open nested transaction".

A possible execution of the example schedule under this multilevel locking protocol is shown in the figure below (with levels L1 and L0 referring to the record and page layer in a database engine, and t_{ij} denoting the jth subtransaction of transaction t_i).

The example shows that, despite many page-level conflicts, high concurrency is possible by exploiting the finer granularity and richer semantics of record-level operations. These benefits are even more pronounced for index-key operations. For this case, highly optimized special-purpose protocols like ARIES Key-Value Locking have been developed. One important optimization for both record and index operations is that the subtransactions may use light-weight latching instead of full-fledged locks.

Another use case with wide applicability are operations on counters, such as increment and decrement or conditional variants on lower-bounded or upper-bounded counters. Such objects and operations are common in reservation systems, inventory control, financial trading, and so on. The relaxed (but not universal) commutativity properties of the operations can be leveraged for very high concurrency even if operations access the same object. Again, special implementation techniques like escrow locking have been developed for these settings. When counter operations have a composite nature, e.g., by automatically triggering updates on other objects, then the special commutativity techniques need to be embedded in a multilevel transaction framework.

A complication that arises from all these high-concurrency settings is that undo recovery (for transaction abort and to wipe out effects of incomplete transactions after a crash) can no longer be implemented merely by restoring prior page versions. Instead, adequately implemented forms of inverse operations need to be executed. Together with the composite nature of operations, this necessitates a form of multilevel recovery.

Most of the outlined principles and algorithms apply to the general case of object-model transactions as well. However, the absence of a layering does incur some extra difficulties, which are beyond the scope of this entry. The algorithms for fully general object-model transactions are explained in detail in the textbook [14].

Future Directions

Multilevel transactions have originally been developed in the database system context, but their usage and potential benefits are by no means limited to database management. So not surprisingly, object-model transactions and related concepts are being explored in the operating systems and programming languages community. Recent trends include, for example, enhancing the Java language with a notion of atomic blocks that can be defined for methods of arbitrary classes. This could largely simplify the management of concurrent threads with shared objects, and potentially also the handling of failures and other exceptions. The runtime environment could be based on an extended form of software transactional memory [8].

Another important trend is to enhance composite web services with transactional properties. Again, object-model transactions is a particularly intriguing paradigm because of its flexibility in allowing application-specific methods for providing atomicity, isolation, and persistence. Adapting and extending transactional concepts for web services and combining them with other aspects of service-oriented computing is the subject of ongoing research [15].

Cross-references

- ▶ Atomicity
- ▶ Concurrency Control
- ▶ Escrow Transactions
- ▶ Key Value Locking

- ▶ Locking
- ▶ Multi-Level Recovery and the ARIES Algorithm
- ▶ Nested Transaction Models
- ▶ System Recovery
- ▶ Transaction
- ▶ Transaction Management

Recommended Reading

1. Beeri C., Bernstein P.A., and Goodman N. A model for concurrency in nested transactions systems. J. ACM, 36(2):230–269, 1989.
2. Davies C.T. and Davies C.T. Jr. Data processing spheres of control. IBM Syst. J., 17(2):179–198, 1978.
3. Gray J. and Reuter A. Transaction processing: concepts and techniques. Morgan Kaufmann, Los Altos, CA, 1993.
4. Greenfield P., Fekete A., Jang J., Kuo D., and Nepal S. Isolation support for service-based applications: A position paper. In Proc. 3rd Biennial Conf. on Innovative Data Systems Research, 2007, pp. 314–323.
5. Lomet D.B. MLR: a recovery method for multi-level systems. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1992, pp. 185–194.
6. 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.
7. Moss J.E.B., Griffeth N.D., and Graham M.H. Abstraction in recovery management. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1986, pp. 72–83.
8. Ni Y., Menon V., Adl-Tabatabai A.-R., Hosking A.L., Hudson R.L., Moss J.E.B., Saha B., and Shpeisman T. Open nesting in software transactional memory. In Proc. 12th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, 2007, pp. 68–78.
9. Ramamritham K. and Chrysanthis P.K. A taxonomy of correctness criteria in database applications. VLDB J., 5(1):85–97, 1996.
10. Schuldt H., Alonso G., Beeri C., and Schek H.-J. Atomicity and isolation for transactional processes. ACM Trans. Database Syst., 27(1):63–116, 2002.
11. Weikum G. Principles and realization strategies of multilevel transaction management. ACM Trans. Database Syst., 16(1):132–180, 1991.
12. Weikum G. and Hasse C. Multi-level transaction management for complex objects: implementation, performance, parallelism. VLDB J., 2(4):407–453, 1993.
13. Weikum G. and Schek H.-J. Architectural Issues of Transaction Management in Multi-Layered Systems. In Proc. 10th Int. Conf. on Very Large Data Bases, 1984, pp. 454–465.
14. Weikum G. and Vossen G. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. Morgan Kaufmann, Los Altos, CA, 2001.
15. Zimmermann O., Grundler J., Tai S., and Leymann F. Architectural Decisions and Patterns for Transactional Workflows in SOA. In Proc. 5th Int. Conf. Service-Oriented Computing, Germany, 2007, pp. 81–93

Multi-Level Visualization

- ▶ Visualizing Hierarchical Data

Multilingual Information Retrieval

- ▶ Cross-Language Mining and Retrieval

Multi-Master System

- ▶ Optimistic Replication and Resolution

Multimedia

- ▶ Image
- ▶ Image Representation
- ▶ Video

Multimedia Content Enrichment

- ▶ Automatic Image Annotation

Multimedia Data

Ramesh Jain
University of California-Irvine, Irvine, CA, USA

Synonyms

Multimodal data

Definition

Multimedia in principle means data of more than one medium. It usually refers to data representing multiple types of medium to capture information and experiences related to objects and events. Commonly used forms of data are numbers, alphanumeric, text, images, audio, and video. In common usage, people refer a data set as multimedia only when time-dependent data such as audio and video are involved.

Historical Background

In early stages of computing, the major applications were scientific computations. In these applications, computers dealt with numbers and were programmed to carry out a sequence of calculations to solve a scientific problem. As people realized power of computing, new applications started emerging. Alphanumeric data was the next type of data to be used in different applications. In early days, these applications were mostly related to businesses. These applications were mostly to store large volumes of data to find desired information from this data set. These applications were the motivation of the development of current database technology.

Text is a special case of alphanumeric data. In text, there is a large string of alphanumeric data that humans associate with written language. Text has been the basis of written human communication and has become one of the most common data form. Most of the information and communication among humans takes place in text.

Next data type to start appearing on computers was images. Images started in many applications where they needed to be analyzed as well as in applications where computers were used to create and display images. Image processing and computer vision emerged as fields dealing with image analysis and understanding while computer graphics emerged as a field dealing with creation and display of images. Images were initially represented in two ways: a list of lines (called vectors) and a 2-dimensional array of intensity values. The second method has now become the most common method of representing images. Images represent a more complex data type because people perceive not the data, which is really a large collection of intensity values, but what the data represents. In computer generated images, the semantics of the pixels is determined and is known at the creation time. In all other images, the semantics must be determined. Computer vision researchers have been developing tools to automatically determine this semantics and have made progress. However, segmenting an image to determine objects in it has been a difficult problem and in general remains an unsolved problem.

Audio data represents variation of a signal over time. Signal processing deals with many types of signals, but due to its closeness to human perception, audio became an important signal type. Unlike regular numerical, alphanumerical, and image data, audio is time-varying or time dependent data. Like images, the numbers have semantics only when they are rendered, in this case using a speaker, to a human. Both images and audio are a collection of numbers that have strong semantics associated with them. This semantics can be associated only by segmenting the data and identifying each segment. Video is next in this sequence of semantic richness. Video is a time-dependent sequence of images synchronized with audio. This means that it brings with it enormous volume of data and richness of semantics.

In late 1980s, people started using the term multimedia to denote combination of text, audio, and video. This gained popularity because the technology had advanced enough to combine these media to articulate thoughts, messages, and stories using appropriate combination of these components and present them easily on computers, save them on CDs, and transmit and receive them using compression/decompression and streaming technologies. By the year 2000, multimedia had become a common data form on computers and Internet.

Foundations

Multimedia data is fundamentally different than the data traditional databases normally manage. Some fundamental differences in multimedia data are discussed here by considering several aspects.

Types and Semantics

The data in early generation databases was either a number or a string of characters. Each data item usually represented value of an attribute. These attribute had clear and explicit semantics in the applications that used the data.

Multimedia data may be considered to be composed of numbers or strings. So an image may be viewed as a two dimensional array of integers. In multimedia applications, however, the semantics is not defined and used at the level of such basic types as in traditional applications. An image is usually considered an image that contains certain objects that are characterized by regions in the image. The relationships among these regions should also be captured. Depending on the context and an application, the semantics associated with an image may change and may need to be represented differently. Similarly an audio file may be viewed as a collection of phonemes rather than just integer values at a time instant representing sound energy. Video is a synchronized combination of audio and images. But if a video is considered

just a combination of separate sound energy and images, then the semantics of video is lost. The semantics of video is due to synchronized combination of its components rather than individual elements.

Multimedia types cannot be considered simply by considering its atomic components. One must consider whole data. The data types and the semantics of multimedia data are the result of the "multi" and are not present in single (mono) medium that may be part of the whole data.

Gestalt philosophy is in action in multimedia data: the whole is bigger than the sum of its parts.

Sequence and Order

Many components of multimedia data are measurements using some sensors. These sensors measure some attribute of physical world. These measurements represent the attribute at a point in space at a particular time. The semantics of the data is intimately tied to the space and time underlying the data. The data is usually organized in the time sequence as it is acquired over some predefined spatial ordering of its acquisition using multiple sensors covering the space of interest.

Multimedia data could be archived data or live data. Archived data is the one that was acquired and stored and hence comes from a server. Live data is presented as it is being acquired. Live data is increasingly being used in many applications.

Size

Multimedia data is voluminous. Audio, Images, and video are much larger in size than alphanumeric data and text. Usually the size of traditional data can be measured in bytes to Kilobytes. Images usually, even the regular amateur photographs run into Megabytes and video easily runs into Gigabytes to Terabytes. Due to the size of the multimedia data, it is usually stored and transmitted in compressed form. For analysis and use of the data, it must be usually decompressed.

Meta data plays a significant role in the analysis of multimedia data and is commonly stored as part of the dataset. Metadata can be of two types: about context or about content. Contextual metadata is about the situation of the real world and the parameters of devices used in acquiring the data. Content related metadata is obtained either thru analysis of the data or by human annotation or interpretation of data.

Many different standards have evolved for compression of multimedia data and association and storage of metadata. Usually these standards related to the medium and are developed by international standards body. Some commonly used standards are JPEG for images, MP3 for audio, and MPEG for video.

Accessing Multimedia Data

Each multimedia data is usually large and represents measurements acquired using a sensor over space and time. Even an image is acquired at a location at a particular time and also contains measurements performed in space using an array of pixel. Each pixel represents measurements related to a particular point in three-dimensional space. Each image or audio video is usually represented as a separate file. This file may contain raw measurements in original form or in compressed form and may also contain associated metadata such as in EXIF data for photos acquired using digital cameras.

Multimedia data representing a measurement is usually represented as one file. In databases such data is usually represented as a pointer to the file, as a BLOB, or the name of the file.

In most current applications, multimedia data is accessed based on the metadata. All queries are formed based on metadata and then the correct file is retrieved and presented. The granularity at which multimedia data is accessed is at the level of file. Text search became so useful when it was applied to documents by analyzing and indexing all areas of a document. This content analysis and indexing based on content within a file will be very useful in multimedia data also. Research in content analysis of multimedia documents for content-based retrieval is an active research area currently.

Presentation

Multimedia data must be presented to a user by sending it to appropriate devices. Audio must be sent to speakers and images and video should also be displayed using special display programs. Displaying raw data in a file is not useful to users. In most cases, before displaying the data, it must be decompressed.

Considering large files and copyright issues, many times multimedia data is not transferred to users for storage, users are allowed to see or listen it only once each time a display request is made. Such playback of data is commonly called streaming of data and is commonly used with video. In streaming, the data from server is sent to a client only for displaying it once. This is also used in the context of live data also.

Key Applications

Computing at one time was mostly numeric, then it became alphanumeric. Now it is multimedia. Almost all applications in computing now deal with multimedia data. In a sense, the term multimedia was a good term to use in the last decade, but now it is a redundant term. In early days of computing there were two types of computing: analog and digital. Slowly all computing became digital. Now no body normally uses the term digital computing because all computing is digital. In the same way all computing ranging from scientific to entertainment will use multimedia data and hence the term multimedia data or multimedia computing will shed "multimedia" and simply become data and computing.

Future Directions

Multimedia data has already become ubiquitous. With the increasing popularity of mobile phones with camera, digital cameras, and falling prices of sensors of different kinds multimedia data is becoming as widespread as alphanumeric data. Considering the current trend and human dependence on sensory data, it is likely that soon multimedia data will become more common than the traditional alphanumeric data. In terms of volume, multimedia data already may be far ahead of alphanumeric data.

Most of the current techniques which deal with multimedia data have two major limitations: first they mostly rely on metadata for access and they treat each type, such as images, audio, and video, as a separate type and hence create silos. What is required is dealing with all data, alphanumeric as well as different types of multimedia, as the data related to some physical objects or situations. This unified approach will treat all data in a unified manner and will not distinguish between media. Each media will be considered only as a source helping understand an object or a situation.

Cross-references

► Multimedia Databases

Recommended Reading

1. Jain R. Experiential computing. Commn. ACM, 46(7):48–55, 2003.
2. Kankanhalli M.S., Wang J., and Jain R. Experiential sampling in multimedia systems. IEEE Trans. Multimed., 8(5):937–946, 2006.
3. Rowe L. and Jain R. ACM SIGMM retreat report on future directions in multimedia research. ACM Trans. Multimedia Comp., Comm., and Appl., 1(1):3–13, 2005.
4. Steinmetz R. and Nahrstedt K. Multimedia Fundamentals: Media Coding and Content Processing (IMSC Press Multimedia series). Prentice Hall, 2002.

Multimedia Data Buffering

Jeffrey Xu Yu
Chinese University of Hong Kong, Hong Kong, China

Definition

Multimedia data are large in size and reside on disks. When users retrieve large multimedia data, in-memory buffers are used to reduce the number of disk I/Os, since memory is significantly faster than disk. The problem to be studied is to efficiently make use of buffers in the multimedia system to reduce the number of I/Os in order to get a better performance when multiple users are retrieving multiple multimedia data simultaneously. Existing works on multimedia data buffering focus on either the replacement algorithms to lower the number of cache misses or the buffer sharing algorithms when many simultaneous clients reference the same data item in memory.

Historical Background

Early works on multimedia data buffering focus on replacement algorithms to reduce the number of cache misses. Although in the traditional database systems, a number of different buffer replacement algorithms, such as the least recently used (LRU) and most recently used (MRU) algorithms are used to approximate the performance behavior of the optimal buffer replacement algorithm [1,2,6,8,15]. They do not reduce disk I/O significantly when they are used in a multimedia database system. Many new buffer replacement algorithms are proposed to save as much of the reserved disk bandwidth for continuous media data as possible. In [5], the effects of various buffer replacement algorithms on the number of glitches experienced by clients are studied. In [9], the authors introduce two buffer replacement algorithms, namely, the basic replacement algorithm (BASIC) and the distance-based replacement algorithm (DISTANCE), for multimedia database systems, which have a much

better performance in comparison with LRU and MRU schema.

In terms of buffer sharing, a simple buffer replacement strategy may miss some opportunities to share memory buffers [14]. A straightforward use of LRU or LRU-k [6,8] is shown to be inadequate [7]. Bridging [3,4,10–12] as a new technique is studied to facilitate data sharing in memory, but it can degrade the system performance. In [13], the authors observe that an uncontrolled buffer sharing scheme may reduce system performance, and introduce the Controlled Buffer Sharing (CBS), which can trade memory for disk bandwidth in order to minimize cost per stream.

Foundations

Buffer Replacement

Assume that each buffer in the buffer space of a system is of the same size and is tagged as either free or used. All the free buffers are kept in a free buffer pool. In order to meet the rate requirement for clients, the system must pre-fetch the required data block from disks into the buffer space, so that the required piece of data is already in the buffer space before being read. In each service cycle, the system first moves the buffers containing data blocks that were already consumed in the last service cycle to the free buffer pool, then determines which data block need to be pre-fetched from disk to the buffer next. If the block is not in the buffer space, then it allocates buffers, from the set of free buffers for the block and issues disk I/O to retrieve the needed data block from disk into the allocated buffers. The algorithm to decide which of the buffers should be allocated is referred to as the buffer replacement algorithm. Several general replacement algorithms are listed below, which are widely used in database management systems. (i) LRU: when a buffer is to be allocated, the buffer containing the block that is used least recently is selected. (ii) MRU: when a buffer is to be allocated, the buffer containing the block that is used most recently is selected. (iii) Optimal: when a buffer is to be allocated, the buffer containing the block that will not be referenced for the longest period of time is selected. Since arrival, pause, resume and jump time when playing an object are unknown in advance, the optimal algorithm can only be implemented for simulation studies.

For the multimedia database systems, the commonly used LRU and MRU algorithms may not reduce disk I/O significantly. Two buffer replacement algorithms are proposed. They are the basic replacement algorithm (BASIC) and the distance-based replacement algorithm (DISTANCE).

The BASIC Buffer Replacement Algorithm The main idea behind the BASIC buffer replacement algorithm [9] is as follows. It is possible to estimate the duration by assuming each client will remain its consumption rate for a long period, even though it is difficult to decide which block will not be referenced for the longest period of time. It assumes that clients continue to consume data at the specified rate they are accessing the blocks. When there is a new request to allocate a buffer, the BASIC algorithm selects the buffer containing the block that will not be accessed for the longest period of time. If there are several such buffers, the algorithm will select the block with the highest offset-rate ratio (the ratio of offset/rate) to be replaced. The BASIC algorithm may reduce the miss ratio to nearly optimal, but it requires to sort clients and free buffers in the increasing order of their offset, which make the overhead of the BASIC algorithm very high. The DISTANCE algorithm is proposed to handle the overhead.

The DISTANCE Buffer Replacement Algorithm The main idea behind the DISTANCE buffer replacement algorithm is based on distance between clients [9]. Suppose that there are clients, $c_1,c_2,...$, accessing the same media data, M. Assume that each client, c_i, is accessing the M at a certain position of M, denoted as $p_i(M)$, and the data block on disk starting from $p_i(M)$ is kept in a buffer, B_i. Let all clients that are accessing the same media data M be sorted in order, $c_1,c_2,....$ Here, c_i is accessing M ahead of c_j if $i < j$, or in other words, $p_i(M) > p_j(M)$, because c_i has already accessed $p_j(M)$ and is now accessing $p_i(M)$. The distance between c_i and its next c_{i+1} is denoted as $dist_i$ which is equal to $p_i(M) - p_{i+1}(M)$. Note that the distance d_i is a value associated with the client c_i. Suppose all clients $c_1,c_2,...$, are accessing their blocks in the buffers in the current cycle. They all need to move ahead and access the next data blocks. The question becomes which buffer they are accessing in the current cycle needs to be freed if the buffer is full. In brief, the buffers consumed by a client, c_i, will be kept longer if the next client, c_{i+1}, will need them shortly (small distance $dist_i$). The buffers consumed by a client, c_j, will be freed earlier if the next client, c_{j+1}, does not need to

access the data block $p_j(M)$, that c_j has just accessed, shortly (large distance $dist_j$). The DISTANCE algorithm frees buffers consumed by clients in the previous cycle in the decreasing order of clients' $dist_i$. In other words, when a new buffer needs to be allocated and there are no free buffers, a buffer consumed by a client, which will not be accessed by its next client shortly, based on the distance between clients, will be selected as a victim to be freed.

The DISTANCE algorithm can be implemented by dynamically maintaining a client list which is ordered in the decreasing order of clients' $dist_i$. The overhead is lower than the BASIC algorithm.

Table 1 shows the comparison of overhead and cache misses of different buffer replacement algorithms, n_B is the number of buffers used.

Buffer Sharing

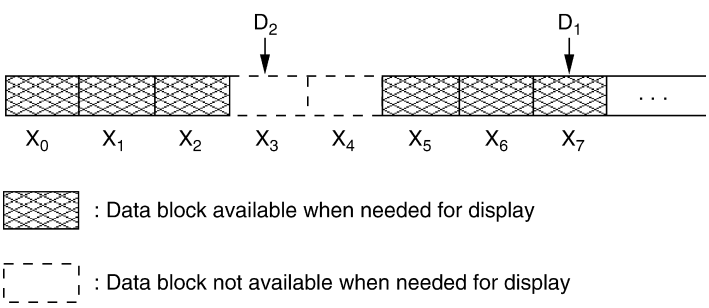
Consider buffer sharing, where cached data can be shared among all the clients. A naive approach is to use LRU or LRU-k, which is shown to be inadequate to efficiently share data. An example is given in Fig. 1. There are two displays, D_1 and D_2, and both reference different blocks of the same clip. With LRU as a global buffer pool replacement policy, the blocks accessed by D_1 may be discarded before D_2 needs to access.

Bridging [3,4,10–12] as a technique is to form a bridge between the data blocks staged by two different clients referencing the same clip, which enables them to share memory and use one disk stream. As shown in Fig. 2, two displays D_1 and D_2 are supported using a single disk stream. The distance between D_1 and D_2 is 5. With the bridging technique, it holds the intermediate data pages between D_1 and D_2 in the buffer pool, and does not swap these pages out from the buffer pool. However, as analyzed in [11,12], a potential problem is that a simple bridging may possibly exhaust the available buffer space, which will have great impacts on the system performance.

A Controlled Buffer Sharing (CBS) technique is proposed in [13], which increases disk bandwidth using memory in order to achieve two objectives, namely, minimization of cost per simultaneous stream, and balancing memory and disk utilization. The latter considers that unlimited memory consumption may in fact degrade the system performance. The framework of CBS is shown in Fig. 3. The framework consists of three components: a configuration planner, a system generator, and a buffer management technique. The configuration planner determines the amount of required buffer and disk bandwidth in support of a prespecified performance objective. The system generator simply acts as a multiplier. The first two components

Multimedia Data Buffering. Table 1. Overhead and cache misses of different buffer replacement algorithms (Table 1 in [9])

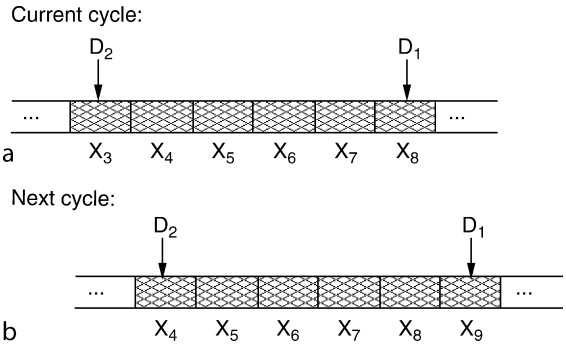
n_B	LRU	MRU	BASIC	DISTANCE	# of refs
300	13:48 s/670,080	13:44 s/668,974	33:30 s/638,974	11:89 s/641,274	670,080
600	13:32 s/670,080	13:18 s/665,748	1:12min/595,416	10:96 s/599,214	670,080
1,200	16:80 s/665,934	16:27 s/657,634	3:41min/549,570	12:28 s/554,640	670,080
2,400	13:22 s/654,240	12:32 s/642,914	5:31min/480,068	8:64 s/481,364	670,080



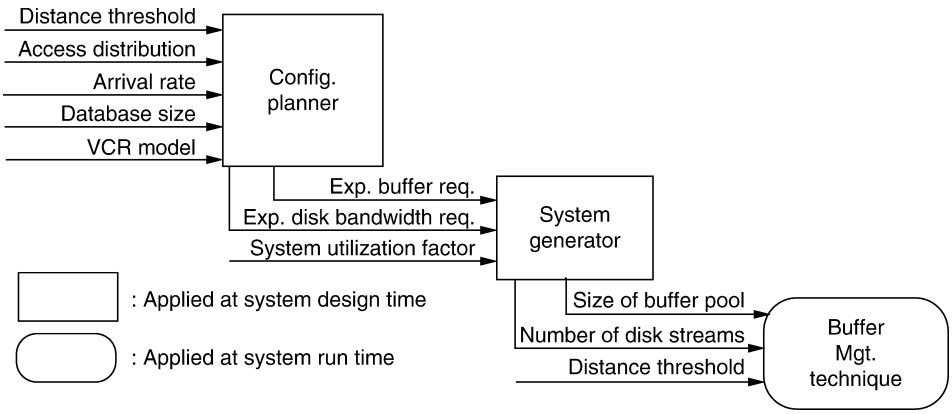
Multimedia Data Buffering. Figure 1. Two displays may compete for buffer frames with LRU (Fig. 1 in [13]).

are applied off-line to determine the system size. The buffer management technique controls the memory consumption at run time.

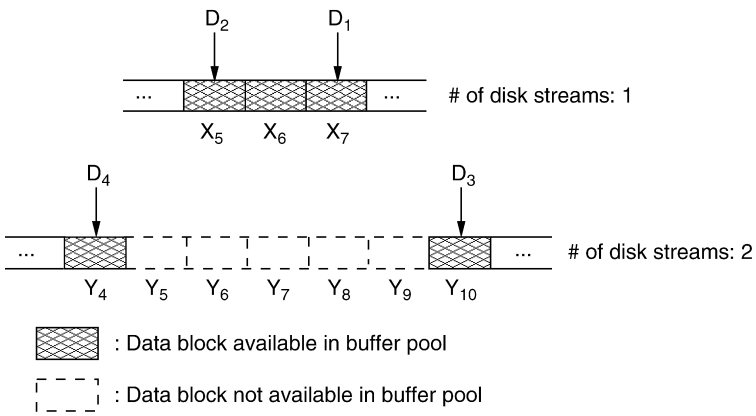
In the CBS framework, a distance threshold, d_t, is used to capture the cost of memory and disk bandwidth and control the number of pinned buffer blocks between two adjacent displays that access the same clip. As shown in Fig. 4, suppose $d_t = 5$, D_1 and D_2 can share one disk stream because their distance is below the specified threshold, while D_3 and D_4 cannot share one disk stream because their distance exceeds the threshold.



Multimedia Data Buffering. Figure 2. Bridging (Fig. 2 in [13]).



Multimedia Data Buffering. Figure 3. The CBS Scheme (Fig. 4 in [13]).



Multimedia Data Buffering. Figure 4. The effectiveness of distance threshold ($d_t = 5$) (Fig 5 in [13]).

Key Applications

Buffering is widely used in retrieving and playing multimedia data, especially for network continuous media applications, where multiple users may need to display multiple medias simultaneously.

Cross-references

- ▶ Buffer Management
- ▶ Buffer Manager
- ▶ Continuous Multimedia Data Retrieval
- ▶ I/O Model of Computation
- ▶ Multimedia Data Buffering
- ▶ Multimedia Data Storage
- ▶ Multimedia Resource Scheduling

Recommended Reading

1. Chew K.M., Reddy J., Romer T.H., and Silberschatz A. Kernel support for recoverable-persistent virtual memory. In Proc. USENIX MACH III Symposium, 1993, pp. 215–234.
2. Chou H.T. and DeWitt D.J. An evaluation of buffer management strategies for relational database systems. In Proc. 11th Int. Conf. on Very Large Data Bases, 1985, pp. 127–141.
3. Dan A., Dias D.M., Mukherjee R., Sitaram D., and Tewari R. Buffering and caching in large-scale video servers. In Digest of Papers - COMPCON, 1995, pp. 217–224.
4. Dan A. and Sitaram D. Buffer management policy for an on-demand video server. IBM Research Report RC 19347.
5. Freedman C.S. and DeWitt D.J. The SPIFFI scalable video-on-demand system. ACM SIGMOD Rec., 24(2):352–363, 1995.
6. Lee D., Choi J., Kim J.H., Noh S.H., Min S.L., Cho Y., and Kim C.S. On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (IFU) policies. SIGMETRICS Perform. Eval. Rev., 27(1):134–143, 1999.
7. Martin C. Demand paging for video-on-demand servers. In Proc. Int. Conf. on Multimedia Computing and Systems, 1995, pp. 264–272.
8. O'Neil E.J., O'Neil P.E., and Weikum G. The LRU-K page replacement algorithm for database disk buffering. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 297–306.
9. Özden B., Rastogi R., and Silberschatz A. Multimedia Information Storage and Management, chap. 7: Buffer Replacement Algorithms for Multimedia Storage Systems. Kluwer Academic, 1996.
10. Rotem D. and Zhao J.L. Buffer management for video database systems. In Proc. 11th Int. Conf. on Data Engineering, 1995, pp. 439–448.
11. Shi W. and Ghandeharizadeh S. Buffer sharing in video-on-demand servers. SIGMETRICS Perform. Eval. Rev., 25(2):13–20, 1997.
12. Shi W. and Ghandeharizadeh S. Trading memory for disk bandwidth in video-on-demand servers. In Proc. 1998 ACM Symp. on Applied Computing, 1998, pp. 505–512.
13. Shi W. and Ghandeharizadeh S. Controlled Buffer Sharing in Continuous Media Servers. Multimedia Tools Appl., 23(2):131–159, 2004.
14. Christodoulakis S., Ailamaki N., Fragonikolakis Y., and Koveos L. Leonidas K. An object oriented architecture for multimedia information systems. Data Eng., 14(3):4–15, 1991.
15. Stonebraker M. Operating system support for database management. Readings in database systems (3rd ed.), Morgan Kaufmann, San Francisco, CA, USA, pp. 83–89, 1998.

Multimedia Data Indexing

Paolo Ciaccia
University of Bologna, Bologna, Italy

Synonyms

MM indexing

Definition

Multimedia (MM) data indexing refers to the problem of preprocessing a database of MM objects so that they can be efficiently searched for on the basis of their content. Due to the nature of MM data, indexing solutions are needed to efficiently support *similarity queries*, where the similarity of two objects is usually defined by some expert of the domain and can vary depending on the specific application. Peculiar features of MM indexing are the intrinsic high-dimensional nature of the data to be organized, and the complexity of similarity criteria that are used to compare objects. Both aspects are therefore to be considered for designing efficient indexing solutions.

Historical Background

Earlier approaches to the problem of MM data indexing date back to the beginning of 1990s, when it became apparent the need of efficiently supporting queries on large collections of non-standard data types, such as images and time series. Representing the content of such data is typically done by automatically extracting some low-level features (e.g., the color distribution of a still image), so that the problem of finding objects similar to a given reference one is transformed into the one of looking for similar features. Although, at that time, many solutions from the pattern recognition field were available for this problem, they were mainly concerned with the *effectiveness* issue (which features to consider and how to compare them), thus almost disregarding *efficiency* aspects.

The issue of making similarity query processing scalable to large databases was first considered in

systems like QBIC [6] for the indexing of color images and by more focused approaches such as the one described by Jagadish in [8] for indexing shapes. Not surprisingly, these solution adopted index methods available at that time that had been developed for the case of low-dimensional spatial databases, such as R-trees and Grid files. The peculiarity of MM data then originated a flourishing brand new stream of research, which resulted in many indexes explicitly addressing the problems of high-dimensional features and complex similarity criteria.

Foundations

Figure 1 illustrates the typical scenario to be dealt with for indexing multimedia data. The first step, *feature extraction*, is concerned with the problem of highlighting those relevant features, f_i, of an object o_i on which content-based search wants to be performed. In the figure, this is the shape of the image subject (a cheetah). The second step, *feature approximation*, is optional and aims to obtain a more compact representation, af_i, of f_i that can be inserted into a suitable index structure (third step). It has to be remarked that, while feature extraction is needed to define which are the relevant aspects of objects on which the search has to focus on, feature approximation is mainly motivated by feasibility and efficiency reasons. This is because it might not be possible to directly index non-approximate features and/or indexing approximate features might result in a better performance of the search algorithms.

Consider a collection $O = \{o_1, o_2, ..., o_n\}$ of MM objects with corresponding features $F = \{f_1, f_2, ...f_n\}$ and approximate features $AF = \{af_1, af_2, ...af_n\}$. In order to compare features, a *distance function* d is typically set up, where $d(f_i, f_j)$ measures how dissimilar are the feature values of objects o_i and o_j. Given a reference object q (the *query point*), a range query with radius $\in$, also called an $\in$-similarity query, will return all the objects $o_i \in O$ such that $d(f_i, f(q)) \leq \in$, whereas a k-nearest neighbor query (k-NN) will return the k objects in O whose features are closest to those of q.

A simple yet remarkable result due to Agrawal, Faloutsos, and Swami [1], and now popularly known as the lower-bounding lemma, provides the basis for exactly solving queries by means of an index that organizes approximate features:

The lower-bounding lemma. Let I be an index that organizes the set of approximate features $AF = \{af_1, af_2, ...af_n\}$ and that compares such features using an *approximate distance* d_{appr}. If, for any pair of objects, it is $d_{appr}(af_i, af_j) \leq d(f_i, f_j)$, then the result of a range query obtained from I is guaranteed to contain the exact result, i.e., no false dismissals are present.

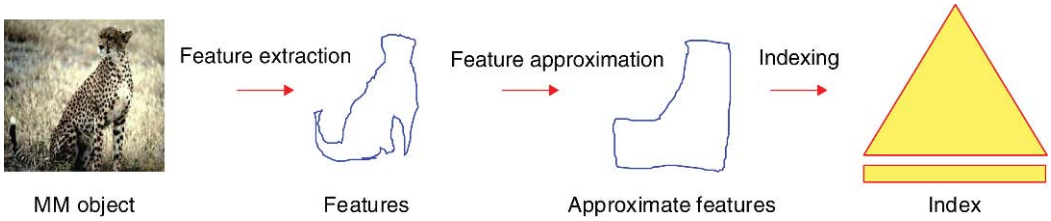
The result easily follows from the observation that, since d_{appr} lower bounds d by hypothesis, $d(f_i, f(q)) \leq \in$ implies $d_{appr}(af_i, af(q)) \leq \in$. The lower-bounding lemma guarantees that querying the index with a search radius equal to $\in$ will return a result set that contains all the objects whose non-approximate features satisfy the query constraint.

Filter & Refine

When indexing is based on an approximate distance, a two-step *filter & refine* process is therefore needed, in which the role of the index is to filter out many irrelevant objects. The so-resulting candidate objects then need to be verified by using the actual distance d. The lower-bounding lemma is also the key for solving k-NN queries using a multi-step query processing approach.

The effectiveness of the filter & refine approach depends on two contrasting requirements:

1. The approximate distance function d_{appr} should be a tight approximation of d, in order to minimize the number of false hits, i.e., those objects that do not satisfy the query constraint yet the index is not able to discard them. These are exactly those objects o_i for which both $d_{appr}(af_i, af(q)) \leq \in$ and $d(f_i, f(q)) > \in$ hold.



Multimedia Data Indexing. Figure 1. The multimedia data indexing scenario.

2. At the same time, d_{appr} should be relatively cheap to compute as compared to d, in order to avoid wasting much time in the filter phase.

The literature on MM indexing abounds of examples showing how to derive effective approximations for complex distance functions. For instance, the QBIC system compares color images using a quadratic form distance function, $d_A^2(f_i,f_j) = (f_i - f_j)A(f_i - f_j)^T = \sum_{k=1}^{D} \sum_{l=1}^{D} a_{k,l}(f_{i,k} - f_{j,k})(f_{i,l} - f_{j,l})$, where $A = (a_{k,l})$ is a color-to-color similarity matrix and features are color histograms. Evaluating d_A has complexity $O(D^2)$, which becomes too costly even for moderately large values of D, the number of bins in the color histograms. In [6] it is demonstrated that using as approximate features the average RGB color of an image, which is a three-dimensional vector, and comparing average colors using the Euclidean distance, i.e., $d_{avg}^2(af_i,af_j) = \sum_{k\in\{R,G,B\}}(af_{i,k} - af_{j,k})^2$, leads to derive that $d_{avg}^2 \leq d_A^2/\lambda_1$, where λ_1 is the smallest eigenvalue of matrix A. Then, the lower-bounding lemma guarantees that querying an index built on average colors with a range query of radius $\epsilon/\sqrt{\lambda_1}$ will not lead to any false dismissal.

As another relevant example, consider the problem of comparing feature vectors that represent time-varying signals. A distance function more robust than the Euclidean one to misalignments on the time domain is the dynamic time warping (DTW) distance. However, evaluating DTW has a complexity $O(D^2)$, which is untenable for long time series. In [9] Keogh introduces an effective lower-bounding function for the DTW distance. In essence, the idea is to construct an *envelope*, $Env(q)$, around the query time series q, after that an Euclidean-like distance between $Env(q)$ and a stored sequence f_i can be computed in $O(D)$ time.

The Need for Approximate Features

As anticipated, there are several reasons for which approximate features might have to be considered. First, in many relevant cases the features of a MM object are represented through a high-dimensional vector, $f_i = (f_{i,1}, f_{i,2},\ldots,f_{i,D})$, with D of the order of the hundreds or even thousands. At such high dimensions it is known that the performance of multidimensional indexes rapidly deteriorates, becoming either comparable to, or even worse than, that of a sequential scan. This phenomenon, known as the dimensionality curse, inhibits any approach based on a direct indexing of feature values. Besides ad hoc solutions, such as those above described, one might consider using some dimensionality reduction technique that projects feature vectors onto a (much) lower D'-dimensional space, $D' \ll D$, and then indexing the so-obtained D'-dimensional feature vectors. The effectiveness of such techniques however is highly variable, being dependent on the actual data distribution.

Another practical reason that could motivate the use of approximate features is the mismatch between the type of the features and the one natively supported by the index. As a simple example, consider an index implementation that only manages entries of an arbitrary, but fixed, size, and that objects to be indexed are regions of pixels described by their boundaries. Clearly, boundary descriptions have different sizes, depending on the shape of the region. In this case a possible solution would be to use a *conservative approximation* of boundaries, like minimum bounding rectangles.

Finally, it might also be the case that, although in principle actual feature values could be stored in the index, the distance function to be used on them cannot be supported by the index organization. A remarkable example is the DTW distance: since DTW is *not* a true metric, in that it does not satisfy the triangle inequality, no multidimensional index can directly process queries with such a distance function.

Metric Indexing

When features are not vectors and/or the distance function is not the Euclidean distance or some other (possibly weighted) L_p norm, coordinate-based spatial indexes cannot be used. There are many cases in which this situation shows up. For instance, in region-based image retrieval (RBIR), each image is first automatically segmented into a set of homogeneous regions, each of them being represented by a vector of low-level features (usually encoding color and texture information). Thus, each f_i is a *set of vectors* and as such cannot be indexed by a spatial index. As a further example, *graphs* representing, say, spatially located objects with their relationships cannot be directly supported by a coordinate-based index. In cases like these, one could consider using a metric index, such as the M-tree [5]. A metric index just requires the distance function d used to compare feature values to be a metric, i.e., a positive and symmetric function that also satisfies the triangle inequality: $d(f_i,f_j) \leq d(f_i,f_k) + d(f_k,f_j) \quad \forall f_i,f_j,f_k$. Although there is nowadays a

large number of metric indexes available [14], as demonstrated in [3] all of them are based on the common principle of organizing the indexed features into a set of equivalence classes and then discarding some of these classes by exploiting the triangle inequality. For instance, in the case of the M-tree each class corresponds to the set of feature values stored into a same leaf of the tree. Triangle inequality can also be applied to save some distance computations while searching the index, which turns out to be particularly relevant in the case of computationally demanding distance functions (a common case with MM data). This was first shown for the M-tree, in which distances between each feature value and its parent in the index tree are precomputed and stored in the tree. The idea is quite general and effective, an obvious tradeoff existing between the amount of extra information stored in the tree and the benefit this has on pruning the search space. Along this direction, Skopal and Hoksza propose the M*-tree [12], a variant of the M-tree in which each entry in a node also includes its NN in that node, i.e., the *NN-graph* of the features in each node is maintained.

A common objection to metric indexes is that they are bound to use only a specific distance function, namely the one with which the index is built. Along the direction of increasing flexibility, Ciaccia and Patella [4] introduce the QIC-M-tree, which is an extension of the M-tree able to support queries with any distance function d_Q from the same "family" of the distance d_I used to build the tree. On the condition that there exists a *scaling factor* $S_{d_I \to d_Q}$ such that $d_I(f_i, f_j) \leq S_{d_I \to d_Q} d_Q(f_i, f_j)$ holds (i.e., d_I lower bounds d_Q up to a constant factor), the lower-bounding lemma applies, and the index can answer queries based on d_Q. A similar idea allows the QIC-M-tree to use also a "cheap" approximate distance d_C as a filter before computing the "costly" d_I and d_Q functions.

Ad Hoc Solutions

The availability of general purpose metric indexes does not rule out the possibility of deriving better, more specialized solutions for the problem at hand. For instance, the STRG-Index [10] is a specialized structure for indexing spatio-temporal graphs arising from the modelling of video sequences. Consider a video segment with N frames. Each frame is first segmented into a set of homogeneous color regions, each of which becomes a node in the region adjacency graph (RAG) of that frame, with edges connecting spatially adjacent regions. Node attributes (such as size, color, and location) are then defined, and the same is done for edges (in which case attributes such as the distance and the orientation between the centroids of connected regions can be used). Since a node representing a region can span multiple frames, nodes in consecutive RAGs can be connected to represent temporal aspects. The resulting graph is called spatio-temporal region graph (STRG). The STRG is then decomposed into a set of object graphs (OGs) and background graphs (BGs), and clusters of OGs are obtained for the purpose of indexing. Since the distance function used for comparing OGs (the so-called extended graph edit distance (EGED)) is a metric, any metric index could be used. The ad hoc STRG-Index proposed in [10] is a three-level metric tree, where the root node contains entries for the BGs, the intermediate level stores clusters of OGs, and individual OGs are inserted into the leaf level.

Extensions of available indexes might be also required as a consequence of feature approximation. An example is found in [13], where the problem of providing rotation-invariant retrieval of shapes under the Euclidean (L_2) distance is considered. After converting a shape boundary into a time series $f_i = (f_{i,1}, f_{i,2},..., f_{i,D})$ (this is quite a common way to represent shapes, see e.g., [2]), a discrete fourier transform (DFT) is applied to obtain a representation of f_i in the frequency domain. Due to Parseval's theorem, the DFT transformation preserves the Euclidean distance [1]. To obtain invariance to rotation, only the magnitude of DFT coefficient is retained. The so-resulting vectors F_i are then compressed by keeping only the k ($k \ll D$) coefficients with the highest magnitude (together with their position in the original vector) plus an error term $\in F_i$ given by the square root of the sum of the squares of dropped coefficients. This information allows a tight lower bound to be derived on the actual rotation-invariant Euclidean distance between f_i and a query shape q. For indexing, a variation of the VP-tree is introduced, which allows compressed features to be stored and searched.

Key Applications

Any application dealing with massive amounts of multimedia data requires effective indexing solutions for efficiently supporting similarity queries. This is further motivated by the complexity of distance functions that are of interest for multimedia data.

Future Directions

All the above indexing techniques and methods assume (at least) that the distance function is a metric. An interesting problem is to devise indexing methods for non-metric distance functions that do not rely on the lower-bounding lemma. The work of Skopal [11] on *semimetrics* appears to be a relevant step on this direction. In the same spirit, Goial, Lifshits and Schütse [7] study how to avoid turning the *similarity* search problem into a *distance*-based one, which in several cases might not yield a metric. Working directly with similarities is however more complex, since there is no analogue of the triangle inequality property for similarity values. Let $rank_y(x)$ be the rank of object x with respect to object y (i.e., x is the NN of y if $rank_y(x) = 1$). Then, [7] introduces the concept of *disorder constant DC*, the smallest value for which the *disorder inequality* $rank_y(x) \leq DC(rank_z(x) + rank_z(y))$ holds $\forall x, y, z$ in the given dataset, and describes algorithms for NN search based on this idea. Making this approach practical for large MM databases remains an open problem.

Cross-references

- ▶ Curse of Dimensionality
- ▶ Dimensionality Reduction
- ▶ High Dimensional Indexing
- ▶ Indexing and Similarity Search
- ▶ Indexing Metric Spaces
- ▶ Multimedia Data Querying
- ▶ Spatial Indexing Techniques

Recommended Reading

1. Agrawal R., Faloutsos C., and Swami A. Efficient similarity search in sequence databases. In Proc. 4th Int. Conf. on Foundations of Data Organizations and Algorithms, 1993, pp. 69–84.
2. Bartolini I., Ciaccia P., and Patella M. WARP: Accurate retrieval of shapes using phase of Fourier descriptors and time warping distance. IEEE Trans. Pattern Anal. Machine Intell., 27(1):142–147, 2005.
3. Chávez E., Navarro G., Baeza-Yates R., and Marroquín J.S. Proximity searching in metric spaces. ACM Comput. Surv., 33 (3):273–321, September 2001.
4. Ciaccia P. and Patella M. Searching in metric spaces with user-defined and approximate distances. ACM Trans. Database Syst., 27(4):398–437, December 2002.
5. Ciaccia P., Patella M., and Zezula P. M-tree: An efficient access method for similarity search in metric spaces. In Proc. 23th Int. Conf. on Very Large Data Bases, 2007, pp. 426–435.
6. Faloutsos C., Barber R., Flickner M., Hafner J., Niblack W., Petkovic D., and Equitz W. Efficient and effective querying by image content. J. Intell. Inf. Sys., 3(3/4):231–262, July 1994.
7. Goyal N., and Lifshits Y., and Schütse H. Disorder inequality: A combinatorial approach to nearest neighbor search. In Proc. 1st ACM Int. Conf. on Web Search and Data Mining, 2008, pp. 25–32.
8. Jagadish H.V. A retrieval technique for similar shapes. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 208–217.
9. Keogh E. Exact indexing of dynamic time warping. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 406–417.
10. Lee J., Oh J.H., and Hwang S. STRG-index: Spatio-temporal region graph indexing for large video databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 718–729.
11. Skopal T. On fast non-metric similarity search by metric access methods. In Advances in Database Technology, Proc. 10th Int. Conf. on Extending Database Technology, 2006, pp. 718–736.
12. Skopal T. and Hoksza D. Improving the performance of M-tree family by nearest-neighbor graphs. In Proc. 11th East European Conf. Advances in Databases and Information Systems, 2007, pp. 172–188.
13. Vlachos M., Vagena Z., Yu P.S., and Athitsos V. Rotation invariant indexing of shapes and line drawings. In Proc. ACM Int. Conf. on Information and Knowledge Management, 2005, pp. 131–138.
14. Zezula P., Amato G., Dohnal V., and Batko M. Similarity Search: The Metric Space Approach. Springer, Berlin Heildelberg, New York, 2005.

Multimedia Data Querying

K. Selcuk Candan[1], Maria Luisa Sapino[2]
[1]Arizona State University, Tempe, AZ, USA
[2]University of Turin, Turin, Italy

Definition

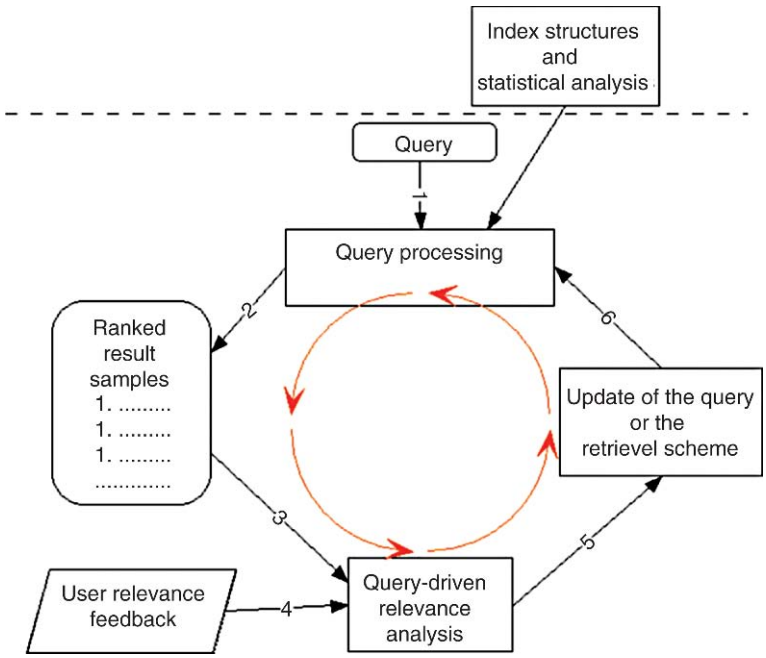
One common characteristic of multimedia systems is the *uncertainty or imprecision of the data*. The models that can capture the imprecise and statistical nature of multimedia data and query processing are fuzzy and probabilistic in nature. Therefore multimedia data query evaluation requires fuzzy and probabilistic data and query models as well as appropriate query processing mechanisms. Probabilistic models rely on the premise that the sources of imprecision in data and query processing are inherently statistical and thus they commit onto probabilistic evaluation. Fuzzy models are more flexible and allow various different semantics, each applicable under different system requirements to be selected for query evaluation.

Historical Background

Due to the possibly redundant ways to sense the environment, the alternative ways to process, filter, and fuse multimedia data, and the subjectivity involved in the interpretation of data and query results, multimedia data quality is inherently imprecise:

- *Feature extraction algorithms that form the basis of content-based multimedia data querying are generally imprecise.* For example, high error rate is encountered in motion capture data due to the multitude of environmental factors involved, including camera and object speed. Especially for video/ audio/ motion streams, data extracted through feature extraction modules are only *statistically* accurate and may be based on the frame rate or the position of the video camera related to the observed object.
- *It is rare that a multimedia querying system relies on exact object matching.* Instead, in many cases, multimedia databases leverage similarities between feature vectors to identify data objects that are similar to the query. In many cases, it is also necessary to account for s*emantic similarities* between associated annotations and *partial matches,* where objects in the result satisfy some of the requirements in the query, but fail to satisfy all query conditions.
- *Imprecision can also be due to the available index structures which are imperfect.* Due to the sheer size of the data, many systems rely on clustering and classification algorithms for pruning during query processing.
- *Query formulation methods are not able to capture user's subjective intention perfectly.* For example, in Query by Example (QBE), which features, feature value ranges, feature combinations, or which similarity notions are to be used for processing is left to the system to figure out through feature significance analysis, user preferences, relevance feedback (Fig. 1), and/or collaborative filtering techniques, which are largely statistical and probabilistic in nature.

In many multimedia querying systems, more than one of these reasons coexist and, consequently, the system must take them into consideration collectively. Figure 2 provides an example query (in an SQL-like syntax used by the SEMCOG system [10]) which brings together imprecise and exact predicates. Processing this query



Multimedia Data Querying. Figure 1. Multimedia query processing usually requires the semantic gap between what is stored in the database and how the user interprets the query and the data to be bridged through a relevance feedback cycle. This process itself is usually statistical in nature and, consequently, introduces probabilistic imprecision in the results.

```
select image P, imageobject object
where contains (P, object) and
        semantically_similar(P.semanticannotation, "travel") and
        visually_similar(object.imageproperties, "Fujimountain.jpg") and
        after(P.date, 2007).
```

Multimedia Data Querying. Figure 2. A sample multimedia query with imprecise and exact predicates.

requires assessment of different sources of imprecision and merging them into a single value. Traditional databases are not able to deal with imprecision since they are based on Boolean logic: predicates are treated as propositional functions, which return either *true* or *false*. A naive way to process queries is to transform imprecision into *true* or *false* by mapping values less than a cut-off to *false* and the remainder to *true*. With this naïve approach, partial results can be quickly refuted or validated based on their relationships to the cut-off. User provided cut-offs can also be leveraged for *filtering*, while maintaining the imprecision value of the results for further processing. In general, however, cut-off based early pruning leads to misses of relevant results. This leads to the need for data models and query evaluation mechanisms, which can take into account imprecision in the evaluation of the query criteria. In particular, the data and query models cannot be propositional in nature.

Foundations

Assessments of the degrees of imprecisions in multimedia data can take different forms. For example, if the data is generated through a sensor/operator with a quantifiable quality rate (for instance a function of the available sensor power), then a scalar-valued assessment of imprecision may be applicable. This is similar to the (so called type-1) fuzzy predicates, which (unlike propositional functions which return *true* or *false*) return a membership value to a fuzzy set. *In this simplest case, the quality assessment of a given object, o, is modeled as a value* $0 \leq qa(o) \leq 1$. A more general quality assessment model would take into account the uncertainties in the assessments themselves. These type of predicates, where sets have grades of membership that are themselves fuzzy, are referred to as type-2 fuzzy predicates. For example the assessment of a given object o can be modeled as a normal distribution of qualities, $qa(o) = No(qo, \xi o)$, where qo is the expected quality and ξo is the variance. Although the type-2 model can be more general and use different probability distributions, this specific model (using the normal distribution) is a generally applicable sampling-related imprecision as it relies on the well-known *central limit theorem*, which states that the average of the samples tends to be normally distributed, even when the distribution from which the average is computed is not normally distributed. Note that, in general, such complex statistical assessments of data precision can be hard to obtain. A compromise between the above two models represents the range of possible qualities of an object with a lower- and an upper-bound. In this case, *given an object o, its quality assessment,* $qa(o)$ *is modeled as a pair* $<qo_{low}, qo_{high}>$, *where* $0 \leq qo_{low} \leq qo_{high} \leq 1$.

Fuzzy data and query models for multimedia querying are based on the fuzzy set theory and fuzzy logic introduced by Zadeh in mid 1960s [14]. A fuzzy set, F, with domain D is defined using a membership function, F: $D \rightarrow [0,1]$. A fuzzy predicate, then, corresponds to a fuzzy set: instead of returning *true(1)* or *false(0)* values as in propositional functions, fuzzy predicates return the corresponding membership values (or scores). Fuzzy clauses combine fuzzy predicates and fuzzy logical operators into complex fuzzy statements. Like the predicates, the fuzzy clauses also have associated scores. The meaning of a fuzzy clause (i.e., the score it has, given the constituent predicate scores) depends on the semantics chosen for the fuzzy logical operators, *not* ($\neg$), *and* ($\wedge$), and *or* ($\vee$).

Table 1 shows popular *min* and *product* fuzzy semantics used in multimedia querying. These two semantics (along with some others) have the property that binary conjunction and disjunction operators are triangular-norms (t-norms) and triangular-conorms (t-conorms). Intuitively, t-norm functions reflect the (boundary, commutativity, monotonicity, and associativity) properties of the corresponding Boolean operations. Although the property of capturing Boolean semantics is desirable in many applications of fuzzy logic, for multimedia querying, this is not always the case [3]. For instance, the partial match

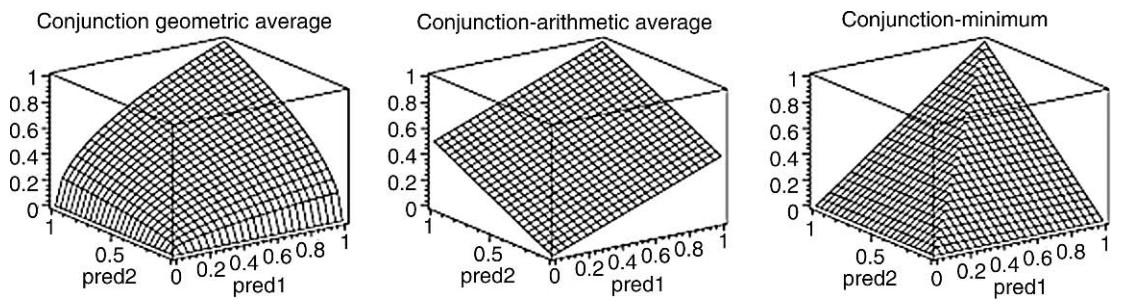
requirements invalidate the boundary conditions. Monotonicity can be too weak a condition for multimedia query processing. In many cases, according to real-world and artificial nearest-neighbor workloads, the highest-scoring predicates are interesting and the rest is not interesting. This implies that the *min* semantics, which gives the highest importance on the lowest scoring predicate, may not be suitable for real workloads. Other fuzzy semantics used in multimedia systems include arithmetic and geometric average semantics. Figure 3 visualizes the behavior of the fuzzy conjunction operator under different fuzzy semantics. It is well established that the only fuzzy semantics which preserves logical equivalence of statements (involving conjunction and disjunction) and is also monotonic is the *min* semantics. This, and the query processing efficiency it enables due to its simplicity, make it a popular choice despite its shortcomings.

Processing multimedia queries, like the one depicted in Fig. 2, under a fuzzy system requires extending query languages and query processors with fuzzy semantics. Many commercial database management systems include fuzzy extensions that are suitable for multimedia applications. Relational databases can be extended to capture fuzzy data in various different ways. In *tuple-level approaches,* the schema of each fuzzy relation is extended to include one or more attributes, each representing the degrees of imprecision of the tuples in the relation with respect to a different interpretation of the tuples. In these systems, the relational algebra operators (such as select, project, join, union, difference) are also extended to apply the selected fuzzy semantics to the tuple scores. In the *attribute-level approaches*, the degrees of uncertainty are associated individually to the attribute values. Especially when the imprecisions in the various attributes of a multimedia object are due to different reasons, attribute level approaches are more applicable due to their finer granularity. Furthermore, since each attribute can be treated as a fuzzy predicate on the multimedia object, query evaluation within these models can benefit more naturally from fuzzy logic evaluation schemes.

Processing these queries, on the other hand, requires significant extensions to the underlying database engines. For example, the underlying relational concepts, such as functional dependencies and normalization, need to be extended to cope with fuzziness in the stored data. In particular, in multimedia

Multimedia Data Querying. Table 1. Fuzzy *min* and *product* semantics for logic operators: $\mu_i(x)$ stands for the score of the predicate P_i on x

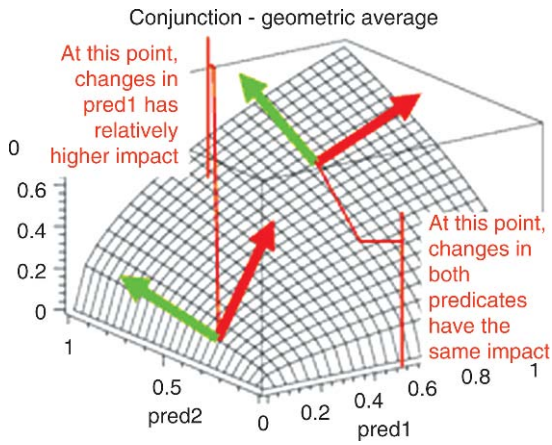
Min semantics	Product semantics
$\mu_{P_i \wedge P_j}(x) = min\{\mu_i(x), \mu_j(x)\}$	$\mu_{P_i \wedge P_j}(x) = \frac{\mu_i(x) \times \mu_j(x)}{max\{\mu_i(x), \mu_j(x), \alpha\}} \quad \alpha \in [0,1]$
$\mu_{P_i \wedge P_j}(x) = max\{\mu_i(x), \mu_j(x)\}$	$\mu_{P_i \vee P_j}(x) = \frac{\mu_i(x)+\mu_j(x)-\mu_i(x)\times\mu_j(x)-min\{\mu_i(x),\mu_j(x),1-\alpha\}}{max\{1-\mu_i(x),1-\mu_j(x),\alpha\}}$
$\mu_{\neg P_i}(x) = 1 - \mu_i(x)$	$\mu_{\neg P_i}(x) = 1 - \mu_i(x)$



Multimedia Data Querying. Figure 3. Visual representations of various binary fuzzy conjunction semantics: The horizontal axes correspond to the values between 0 and 1 for the two input conjuncts and the vertical axis represents the resulting scores according to the corresponding function.

databases, users are usually interested in a result set which is ranked according to a ranking criterion which is generally user dependent (Fig. 1). Adali et al. [1] introduces a similarity algebra which brings together relational operators and results of multiple similarity implementations in a uniform language. Other algebraic treatments of fuzzy multimedia queries, relying on finer granularity attribute-based models, include the *FNF*^{2} *algebra* [5]. When the requirement for exact matches is removed, the result space becomes significantly large, and thus, the query engine cannot rely on any processing scheme which would need to touch or enumerate all solutions. Consequently, query processing schemes would need to generate results as progressively (in decreasing order of relevance) as possible. Fagin [7] proposes ranked query evaluation algorithms, which assume that individual sources can progressively output sorted results and also enable random access. These algorithms also assume that the query has a monotone combined scoring function. Candan et al. [4] presents *approximate* ranked query processing techniques for cases where not all subqueries are able to return ordered results. In turn, Fagin et al. [8] recognizes that there may be cases where random accesses are impossible and presents algorithms under monotonicity assumption to enumerate top-k objects without accessing all the data. These augment *monotonicity* with an *upper bound* principle, which enables bounding of the maximum possible score of a partial result. Qi et al. [13] establishes an alternative, *sum-max monotonicity* property and shows how to leverage this for developing a self-punctuating, horizon-based ranked join (HR-Join) operator for cases when the more strict monotonicity property does not hold. Top-k querying can also be viewed as a k-constrained optimization problem, where the goal function includes both a Boolean constraint characterizing the data of interest and a quantifying function which acts as the numeric optimization target [15]. Adali et al. [2] and Li et al. [11] extend the relational algebra to support ranking as a first-class construct. Li et al. [11] also presents a pipelined and incremental execution model of ranking query plans.

A particular challenge in multimedia querying is that (as shown in Fig. 1) the underlying query processing scheme needs to adapt to the specific needs and preferences of individual users. Due to its flexibility, the fuzzy model enables various mechanisms of adaptation. First of all, if user's feedback focuses on a particular attribute in the query, the way the fuzzy score of the corresponding predicate is computed can change based on the feedback. Secondly, the semantics of the fuzzy logic operator can be adapted based on the feedback of the user. A third mechanism through which user's feedback can be taken into account is to enrich the merge function, used for merging the fuzzy scores, with weights that regulate the impacts of the individual predicates. Fagin proposed a generic weighting mechanism that can be used for any fuzzy merge function [7]. The mechanism ensures that (a) the result is a continuous function of the weights (as long as the original merge function is continuous), (b) sub-queries with zero weight can be dropped without affecting the rest of the query, and (c) if all weights are equal, then the result is equal to the original, not-weighted merge function. Candan and Li [3], on the other hand, argued that the relative importance of predicates in a merge function should be measured in terms of the overall impacts changes in the scores that the individual predicates would have on the overall score (Fig. 4). Consequently, the relative importance of predicates can vary based on the scores the individual predicates take and the corresponding partial derivatives. A more direct mechanism to capture the user feedback is to modify the partial derivatives of the scoring functions appropriately. While the generic scheme presented by Fagin [7] would satisfy this for some merge functions, such as the arithmetic average, it would fail to capture this requirement for others, such as the commonly used product semantics.



Multimedia Data Querying. Figure 4. The relative impact of the predicates in a scoring function can vary based on the scores of the individual predicates.

Unlike the fuzzy models, which can capture a large spectrum of application requirements, probabilistic approaches to data and query modeling are applicable only to those cases where the source of imprecision is of statistical nature. These cases include probabilistic noise in data collection, sampling (over time, space, or population members) during data capture or processing, randomized and probabilistic algorithms (such Markov chains and Bayesian networks) used in media processing and pattern detection, and probabilistic consideration of relevance feedback. Dalvi and Suciu [6], for example, associates a value between 0 and 1 to each tuple in a given relation: the value expresses to probability with which a given tuple belongs to the relation. By extending SQL and the underlying relational algebra with probabilistic semantics and a theory of belief, the authors provide a probabilistic semantics for query processing with uncertain matches.

A general simplifying assumption in many probabilistic models is that the individual attributes (and the corresponding predicates) are independent of each other: consequently, the probability of a conjunction can be computed as the product of the probabilities of the conjuncts; i.e., under these conditions, the probabilistic model corresponds to the fuzzy product semantics. However, the independence assumption does not always hold (in fact, it rarely holds). Lakshmanan et al. [9] presents a probabilistic relational data model, an algebra, and aggregate operators that capture various types of interdependencies, including independence, mutual exclusion, as well as positive, negative, and conditional correlation.

While the simplest probabilistic models associate a single value between 0 and 1 to each attribute or tuple, more complete models represent the score in the form of an interval of possible values or more generally in terms of a probability distribution describing the possible values for the attribute or the tuple. Consequently, these models are able to capture more realistic scenarios, where the imprecision in data collection and processing prevents the system to compute the exact *quality* of the individual media objects, but (based on the domain knowledge) can associate probability distributions to them. Note that relaxing the independence assumption or extending the model to capture non-singular probability distributions both necessitate changes in the underlying rank evaluation algorithms.

Other non-relational probabilistic models for multimedia querying includes Markov chains and Bayesian networks. A stochastic process is said to be Markovian if the conditional probability distribution of the future states depends only on the present. A Markov chain is a discrete-time stochastic process which is conditionally independent of the past states. A random walk on a graph, $G(V,E)$, is a Markov chain whose state at any time is described by a vertex of G and the transition probability is distributed equally among all outgoing edges. The transition probability distribution in the corresponding Markov model can be represented as a matrix, where the (i, j)'th element of this matrix, T_{ij}, describes the probability that, given that the current state is i, the process will be in state j in the next time unit; i.e., the n-step transition probabilities can be computed as the n'th power of the transition matrix. Markovian models are used heavily for linkage analysis in supporting queries over web, multimedia, and social network data with graphical representations.

A Bayesian network is another graphical probabilistic model used especially for representing probabilistic relationships between variables (e.g., objects, properties of the objects, or beliefs about the properties of the objects) [12]. In a Bayesian network nodes represent variables and edges between the nodes represent the relationships between the probability distributions of the corresponding variables. Consequently, once they are fully specified, Bayesian networks can be used for answering probabilistic queries given certain observations. However, in many cases, both the structure as well as the parameters of the network have to be learned through iterative and sampling-based heuristics, such as expectation maximization (EM), and Markov Chain Monte Carlo (MCMC) algorithms. Hidden Markov models (HMMs), where some of the states are hidden (i.e., unknown), but variables that depend on these states are observable, are Bayesian networks used commonly in many machine-learning based multimedia pattern recognition applications. This involves training (i.e., given a sequence of observations, learning the parameters of the underlying HMM) and pattern recognition (i.e., given the parameters of an HMM, finding the most likely sequence of states that would produce a given output).

Key Applications

Applications of multimedia querying include personal and public photo/media collections, personal information management systems, digital libraries, on-line and print advertisement, digital entertainment,

communications, long-distance collaborative systems, surveillance, security and alert detection, military, environmental monitoring, ambient and ubiquitous systems that provide real-time personalized services to humans, improved accessibility to blind and elderly, rehabilitation of patients through visual and haptic feedback, and interactive performing arts.

Future Directions

While most of the existing work in this area focused on content-based and object-based query processing, future directions in multimedia querying will involve understanding of how media objects affect users and how do they fit into users experiences in the real world. These require better understanding of underlying psychological and cognitive processes in human media processing. Ambient media-rich systems which collect and feed in diverse media from environmentally embedded sensors necessitate novel ways of continuous and distributed media processing and fusion schemes. Intelligent schemes to choose the right objects to process are needed to scale query processing workflows to the immense influx of real-time media data. In a similar manner, collaborative-filtering based query processing schemes that can help overcoming the semantic gap between media and users' experiences will help the multimedia databases scale to Internet-scale media indexing and querying.

Cross-references

- ▶ Fuzzy Models
- ▶ Probabilistic Databases
- ▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model
- ▶ Information Retrieval
- ▶ Multimedia Data
- ▶ Multimedia Databases
- ▶ Multimedia Data Indexing
- ▶ Multimedia Information Retrieval Model
- ▶ Multimedia Retrieval Evaluation
- ▶ Top-K Selection Queries on Multimedia Datasets

Recommended Reading

1. Adali S., Bonatti P.A., Sapino M.L., and Subrahmanian V.S. A multi-similarity algebra. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 402–413.
2. Adali S., Bufi C., and Sapino M.L. Ranked relations: query languages and query processing methods for multimedia. Multimed. Tools Appl., 24(3):197–214, 2004.
3. Candan K.S. and Li W.-S. On similarity measures for multimedia database applications. Knowl. Inf. Syst., 3(1):30–51, 2001.
4. Candan K.S., Li W.-S., and Priya M.L. Similarity-based ranking and query processing in multimedia databases. Data Knowl. Eng., 35(3):259–298, 2000.
5. Chianese A., Picariello A., Sansone L., and Sapino M.L. Managing uncertainties in image databases: a fuzzy approach. Multimed. Tools Appl., (23):237–252, 2004.
6. Dalvi N.N. and Suciu D. Efficient query evaluation on probabilistic databases. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 864–875.
7. Fagin R. Fuzzy queries in multimedia database systems. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 1–10.
8. Fagin R., Lotem A., and Naor M. Optimal aggregation algorithms for middleware. J. Comput. Syst. Sci., 66 (4):614–656, 2003.
9. Lakshmanan L.V., Leone N., Ross R., and Subrahmanian V.S. ProbView: a flexible probabilistic database system. ACM Trans. Database Syst., 22(3):419–469, 1997.
10. Li W.-S. and Candan K.S. SEMCOG: a hybrid object-based image and video database system and its modelling, language, and query processing. Theory & Practice of Object Syst., 5 (3):163–180, 1999.
11. Li C., Chang K.C.-C., Ilyas I.F., and Song S. RankSQL: Query algebra and optimization for relational top-k queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 131–142.
12. Pearl J. Bayesian networks: a model of self-activated memory for evidential reasoning. In Proc. 7th Conf. of the Cognitive Science Society, 1985, pp. 329–334.
13. Qi Y., Candan K.S., and Sapino M.L. Sum-Max monotonic ranked joins for evaluating top-K twig queries on weighted data graphs. In Proc. 33rd Int. Conf. on Very Large Data Bases, 2007, pp. 507–518.
14. Zadeh L.A. Fuzzy sets. Inf. Control, 8(3):338–353, 1965.
15. Zhang Z., Hwang S., Chang K.C., Wang M., Lang C.A., and Chang Y. Boolean + Ranking: querying a database by K-constrained optimization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 359–370.

Multimedia Data Storage

Jeffrey Xu Yu
Chinese University of Hong Kong, Hong Kong, China

Definition

Data storage management, as one of the important functions in database management systems, is to manage data on disk in an efficient way to support data retrieval and data update. Multimedia data storage management is to manage continuous media data (audio/video) on disk. The uniqueness of multimedia data storage management is, in a multiuser environment, how to arrange the data storage to support a continuous retrieval of large continuous media data

from disk to be displayed on screen, at a pre-specified rate, without any disruptions, which is also called hiccup-free display.

Historical Background

In a multimedia environment, the continuous media needs to be retrieved and displayed continuously. As magnetic disks are used as the mass storage device for multimedia data, zoning is one approach to increase the storage capacity of magnetic disks. Here, zones of a disk drive are different regions of the disk drive that usually have different transfer rates. A number of studies have investigated techniques to support a hiccup-free display of continuous media (video/audio) using magnetic disk drives with a single zone [1,2,9,10] in the early 1990s. These studies assume a fixed transfer rate for a disk drive. These techniques can be possibly adopted to design a multi-zone disk system, but such a multi-zone disk system is then forced to use the minimum transfer rate of the zones for the entire disk, in order to guarantee a continuous display of continuous media objects. Such an approach is called Min-Z-tfr.

In the late 1990s and early 2000s, many new techniques are proposed to deal with the storage of continuous media in multi-zone disks [4,5]. In [4], VARB and FIXB are proposed to place media objects on the multi-zone disks. VARB and FIXB techniques provide the average transfer rate of zones while ensuring a continuous display, compared with Min-Z-tfr, which is forced to use the minimum transfer rate of zones. VARB and FIXB increase the throughput of the system, while they also (i) increase startup latency, (ii) waste disk space, and (3) increase the amount of memory required to support more simultaneous displays. A configuration planner is proposed to decrease the drawbacks of VARB and FIXB, in order to meet the performance requirements of applications [4]. As VARB and FIXB [4] take account of a single media type only, RP, MTP, and MVP are proposed in [5] to support multiple media types with different bandwidth requirements. In [4,5], the discussions focus on multimedia data placement across disk drives to support continuous display requirement.

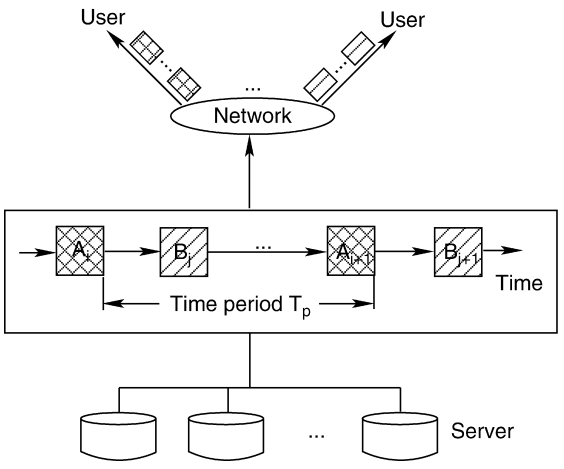
Foundations

Hiccup-Free Display

The size of continuous media, especially videos, can be very large. The transmission of data must be just-in-time. In other words, data must be retrieved from disks and transmitted to the display in a timely manner that prevents hiccups. A cycle-based data retrieval technique [6,12] is designed to provide continuous display for multiple users. As shown in Fig. 1, it is a cycle-based data retrieval technique to support hiccup-free display [5]. Consider a constant-bit-rate (CBR) media. In order to guarantee a continuous display of a continuous media A, the system needs to retrieve the block of A_i before its immediate previous block A_{i-1} completes its display. For each block A_i, there are two tasks, namely, block retrieval and display initialization. This process of block retrieval and display initialization repeats in a cyclic manner until all A blocks have been displayed. If the time to retrieve a block, termed *block retrieval time*, T_p (as indicated in Fig. 1), is smaller than or equal to the time period to display a block, then the whole display process will be hiccup free [5]. The time interval between the time a request of A arrives and the time the display of A starts is called *startup latency*.

FIXB and VARB

Modern disk drivers are produced with multiple zones to meet the demands for a higher storage capacity [11]. A zone is a contiguous collection of disk cylinders where the tracks in the cylinders are supposed to have the same storage capacity. The outer zones have a higher transfer rate in comparison with the inner zones. Two approaches, FIXB and VARB, are proposed in [4] to support a continuous display of continuous media using a single disk with multi-zones. Suppose that the disk consists of m zones, Z_1, Z_2,...,Z_m, and



Multimedia Data Storage. Figure 1. A hiccup free display (Fig. 3 in [5]).

the transfer rate of zone Z_i is R_i. Assume that each object X is partitioned into f blocks: $X_1, X_2,...,X_f$.

With FIXB (Fixed Block Size), the blocks of an object X are rendered equi-sized, i.e., $X_i = X_j$, for any i and j. The system assigns the blocks of X to the zones in a round-robin manner. FIXB is designed to support a predetermined number of simultaneous displays (N). The retrieval process of this system is to scan the disk in one direction, for example, starting with the outermost zone moving inward, visiting one zone at a time and multiplexing the bandwidth of that zone among N block reads. A sweep is a scan of the zones. The time to perform one such a sweep is denoted as T_{Scan}. The time of reading N blocks from zone Z_i, denoted $T_{MUX}(Z_i)$, is dependent on the transfer rate of zone Z_i. As the transfer rate of zones varies, the time to read blocks from different zones also varies. To support hiccup-free displays, the system uses buffers to compensate for the low transfer rates of innermost zones.

VARB makes $T_{MUX}(Z_i)$ to be identical for all zones, using variable block sizes. The size of a block, $B(Z_i)$, is a function of the transfer rate of the zone Z_i. This results in an identical transfer time for all the blocks, T_{disk}, i.e., $T_{disk} = \frac{B(Z_i)}{R_i} = \frac{B(Z_j)}{R_j}$, for any i and j. Like FIXB, VARB assigns the blocks of an object to the zones in a round-robin manner. Unlike FIXB, with VARB, the blocks of an object X have different sizes depending on which zones the blocks are assigned to. Also, like FIXB, VARB employs memory to compensate for the low bandwidth of innermost zones.

With FIXB, the blocks of an object is equi-sized, whereas VARB renders the blocks in different sizes, which depends on the transfer rate of its assigned zone. FIXB is easy to be implemented in compared with VARB. But VARB requires a lower amount of memory and incurs a lower latency as compared to FIXB.

RP, MTP, and MVP

Based on zoning, there are different data transfer rates (R_i) to retrieve data from a disk. When a server is required to support multiple media types with different bandwidth requirements, the block reading time varies widely depending on the block size and its assigned zone. Suppose that there are n different media types to be supported. The block size of a media type i object is determined by $B_i = T_p \times D_i$ where D_i is the bandwidth requirement of the media type i, and T_p is a fixed time period which is set to be the same for all the media types. The transfer time (service time) to retrieve a block of a media type i object in zone Z_j is $s_{i,j} = \frac{B_i}{R_j}$. Suppose that there are b blocks, the average service time is computed as

$$\bar{s} = \sum_{i=1}^{b} F_i \sum_{j=1}^{n} P_{i,B_j} \sum_{k=1}^{m} \frac{P_{i,Z_k} B_j}{R_k}$$

where F_i is the access frequency of block i, for $1 \le i \le b$, P_{i,B_j} is the probability that the size of block i is B_j, and P_{i,Z_k} is the probability that this block is assigned to zone Z_k. The variance of service time is:

$$\sigma_s^2 = \sum_{i=1}^{b} F_i \sum_{j=1}^{n} \sum_{k=1}^{m} P_{i,B_j} P_{i,Z_k} (s_{j,k} - \bar{s})^2$$

Three approaches are proposed in [5]: RP, MTP, and MVP. RP (Random Placement) assigns blocks to the zones in a random manner. MTP (Maximizing Throughput Placement) sorts blocks based on their size and frequency of access ($F_i \times B_i$). The blocks are assigned to the zones sequentially starting with the fastest zone, i.e., block i with the highest $F_i \times B_i$ value is assigned to the fastest zone. With MVP (Minimizing Variance Placement), a block of size B_i is placed on the zone Z_j (with R_j) which has the closest $\frac{B_i}{R_j}$ value to the average block reading time ($\bar{T}_B$):

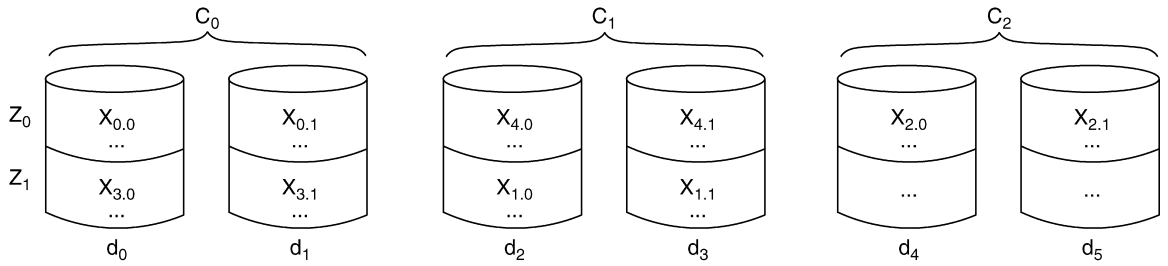
$$\bar{T}_B = \frac{\textit{average block size}}{\textit{average transfer rate}} = \frac{\frac{1}{n}\sum_{i=1}^{n} B_i}{\frac{1}{m}\sum_{i=1}^{m} R_i}$$

Performance studies in [5] demonstrate that both MTP and MVP are superior to RP. MVP outperforms MTP regarding the average service time and/or variance of service time. One advantage of MVP is that it is not sensitive to the access frequency of objects.

Data Placement across Disk Drivers

The bandwidth of a single disk is insufficient for the multimedia applications that strive to support thousands of simultaneous displays. One approach is to employ a multi-disk architecture. Assuming a system with D homogeneous disks, the data is striped across the disks in order to distribute the load of a display evenly across the disks [2,3,8].

The striping technique is as follows (Fig. 2). First, the disks are partitioned into k disk clusters where each cluster consists of d disks: $k = \lceil \frac{D}{d} \rceil$. An object X is partitioned into f blocks, $X_1, X_2,..., X_f$, and the blocks



Multimedia Data Storage. Figure 2. Three clusters with two logical zones per cluster (Fig. 12 in [4]).

of X are assigned to the k disk clusters in a round-robin manner, starting with an arbitrarily chosen disk cluster and zone, for example, zone Z_j in disk cluster C_i. In a disk cluster, each block of X, X_i, is declustered [7] into d fragments, $X_{i,j}$, where each fragment is assigned to a different disk in the disk cluster. As shown in Fig. 2, the X_0 block is assigned to the disk cluster C_0, and its two declustered fragments, $X_{0,0}$ and $X_{0,1}$ are assigned to the zone Z_0. Note that the fragments of a block need to be assigned to the same zone on the d disks in the disk cluster where the block is assigned to. In the retrieval of objects, one zone of all disks in a disk cluster is active per time period. To display object X of Fig. 2, it needs to access zone Z_0 in disk cluster C_0, when the disk cluster is idle, followed by accessing zone Z_1 in disk cluster C_1. This process repeats to retrieve/display all blocks of the object X.

Key Applications

Multimedia information systems have emerged as an essential component in many application domains ranging from library information systems to entertainment technology. The data storage management is the basis to support a continuous display of multimedia objects.

Cross-references

- ▶ Continuous Multimedia Data Retrieval
- ▶ Multimedia Data Buffering
- ▶ Multimedia Resource Scheduling
- ▶ Storage Access Model
- ▶ Storage Devices
- ▶ Storage Management
- ▶ Storage Manager
- ▶ Storage Resource Management

Recommended Reading

1. Anderson D.P. and Homsy G. A continuous media I/O server and its synchronization Mechanism. Computer, 24(10):51–57, 1991.
2. Berson S., Ghandeharizadeh S., Muntz R., and Ju X. Staggered striping in multimedia information systems. ACM SIGMOD Rec., 23(2):79–90, 1994.
3. Ghandeharizadeh S. and Kim S. Striping in Multi-disk Video Servers. In Proc. SPIE High-Density Data Recording and Retrieval Tech. Conf., 1995.
4. Ghandeharizadeh S., Kim S., Shahabi C., and Zimmermann R. Multimedia Information Storage and Management, chap. 2: Placement of Continuous Media in Multi-Zone Disks. Kluwer Academic, 1996.
5. Ghandeharizadeh S. and Kim S.H. Design of multi-user editing servers for continuous media. Multimedia Tools Appl., 11(1):101–127, 2000.
6. Ghandeharizadeh S., Kim S.H., Shi W., and Zimmermann R. On minimizing startup latency in scalable continuous media servers. In Proc. SPIE Conf. on Multimedia Computing and Networking, 1997.
7. Ghandeharizadeh S., Ramos L., Asad Z., and Qureshi W. Object placement in parallel hypermedia systems. In Proc. 17th Int. Conf. on Very Large Data Bases, 1991, pp. 243–254.
8. Ozden B., Rastogi R., and Silberschatz A. Disk striping in video server environments. In Proc. Int. Conf. on Multimedia Computing and Systems, 1996, pp. 580–589.
9. Rangan P.V. and Vin H.M. Efficient storage Techniques for Digital Continuous Multimedia. IEEE Trans. Knowl. Data Eng., 5(4):564–573, 1993.
10. Reddy A.L.N. and Wyllie J.C. I/O issues in a multimedia system. Computer, 27(3):69–74, 1994.
11. Ruemmler C. and Wilkes J. An introduction to disk drive modeling. Computer, 27(3):17–28, 1994.
12. Tewari R., Mukherjee R., Dias D.M., and Vin H.M. Design and performance tradeoffs in clustered video servers. In Proc. Int. Conf. on Multimedia Computing and Systems, 1996, pp. 144–150.

Multimedia Databases

Ramesh Jain
University of California-Irvine, Irvine, CA, USA

Synonyms

Multimodal databases

Definition

Multimedia Databases are databases that contain and allow key data management operations with multimedia data. Traditional databases contained alphanumeric data and managed it for various applications. Increasingly, applications now contain multimedia data that requires defining additional types and requires development of operations for storage, management, access, and presentation of multimedia data. Multimedia databases must increasingly deal with issues related to managing multimedia data as well as the traditional data. Commonly, databases that manage images, audio, and video in addition to metadata related to these and other alphanumeric information are called multimedia databases. When databases contain only one of the images, audio, or video, they are called image databases, audio databases, and video databases, respectively. Considering the current trend, it is likely that most databases will slowly become multimedia databases.

Historical Background

The first in multimedia databases to appear were image databases that started appearing in late 1980s. Researchers in early image databases were more concerned with using databases for maintaining results of image processing operations to analyze and understand image analysis systems. Remote sensing and medical imaging produced images that needed to be saved and analyzed to extract information for various applications. In most of these applications, an environment to save images and processing results of these images were required.

The idea of making images an integral component of databases first started appearing in early 1990s. Relational data model had become the most common data model to deal with structured data and was used to store images as binary large objects (BLOBs) in these databases. To deal with images as first class data objects in images, a multilayered data model was proposed. This model considered image objects, and domain objects and suggested storage of those along with changes in relationships among objects. Some interesting developments in early systems evolved along two independent directions. In one direction [3], a user was considered an integral part of the query environment and feedback from user resulted in continuous refinement of queries leading to finding images that were required. In the other approach, many low level features were computed and used for finding images using query by example approach. These two approaches adopted distinctly different directions, the first used domain knowledge and the second relied only on image features without any use of domain knowledge. The image features used commonly are different types and characteristics of color histograms and texture measures. These approaches are commonly called content-based retrieval, to differentiate them from metadata based retrieval. Commercially image database systems appeared in traditional database systems in mid 1990s. IBM used its QBIC technology in their DB2 database system and Oracle, Sybase, and Illustra used technology developed by a start-up company Virage. All search engines use image retrieval mostly based on the metadata that includes name of the file and text in the context of the image on a webpage.

Content based video retrieval result started in analyzing video into its constituent parts. At the lowest level is a frame, an individual image. Images are grouped into shots, shots into scenes, and scenes into episodes. All this data is extracted from video and stored in the database. Speech recognition techniques are used to prepare the transcript of the video and are also used in the database. Such systems found early use in TV program production and defense applications. Virage technology was used in these applications. Current search engines usually use metadata for searching video. Some specialized video search companies such as Blinkx (http://www.blinkx.com/) use predominantly text obtained using speech recognition or closed captions in television video. News videos have been one of the major application domains due to their applications as well as to good quality of audio available for these.

In audio databases, the signal is analyzed to detect characteristics that could be used in searching musical pieces that are similar to those. Such techniques, sometimes referred to as query by humming, were thought to be useful in finding music of interest.

Some effort has gone into analyzing CAD databases also for retrieving drawing and objects of interest.

Though some research has started in addressing multimedia data, rather than just one of the above multimedia types, most research is in either images, video, or audio. Text or metadata available in context of multimedia data is being considered in multimedia database research increasingly.

Foundations

The most fundamental difference in multimedia databases compared to the traditional databases is in the

rich semantics of the data. Multimedia data, in addition to being lot more voluminous, is very rich in semantics. Many queries that users are interested require understanding of the semantics of the data. The problem becomes more complex because the semantics of multimedia is dependent not only on the data, but also on the specific user, the context in which the query is asked, and other sets of data available in the system.

Early approaches to multimedia databases did not consider the nature of multimedia data and just stored multimedia data either as BLOBs or links to files containing the data. These systems could allow only limited operations on multimedia data – usually limited to display or rendering of the data. No other operations or queries could be performed on this data.

With increasing use of multimedia data and applications that require use of multimedia data in many different ways, the nature of multimedia databases started changing. Currently, multimedia databases are still in their early stages. Many different concepts and approaches are being tried. Some of the important emerging ideas that are being tried in multimedia databases are discussed below. This area is currently a very active one and is likely to receive increasing attention both from academic and industrial research community.

A multimedia database system is considered to have the following four clear modules that need to work together to provide the functionality desired from them.

Data Analysis and Feature Extraction

A MMDB contains multimedia data but just storing the data as a BLOB does not allow any queries related to the content of the data. To solve this problem, data is analyzed to extract features from the data. These features can then be used to derive the required semantics. The features extracted depend on the nature of the type of the data and domain of application for the MMDB. These features could range from low-level features that are very general and do not depend on the application domain such as, color histograms and texture features for images to high level features directly tied to application domain such as shape of the tumor.

A significant amount of research related to MMDB is in specific media related research communities such as audio processing or computer vision. There is strong interest in finding efficient and effective features to interpret multimedia data in general as well as in specific applications.

Domain Knowledge and Interpretation

Multimedia data interpretation requires use of domain knowledge. Moreover, the knowledge required for interpretation of this data is not only the traditional domain knowledge represented using ontologies and similar techniques well developed for interpretation of text; but also media dependent models that require sophisticated classification approaches. In audio and video events must be detected in the data and that requires processing time dependent features.

There is a new emerging perspective that multimedia data should be considered evidence for real world events captured using such data. This requires modeling events and representing knowledge about domain events. This knowledge is then used in interpretation of multimedia data not in silos but together. Some progress is being made in representation of events.

Interaction and User Interface

Interaction environments used in traditional databases and search engines are not satisfactory in many applications of MMDB. Using keywords or names of objects, some limited searches can be performed, but many applications require concepts and ideas that require both continuous interactions and successive refinement of queries in what is called emergent semantics environment. Query by example including query using sketches, humming, and some other non-textual approaches are being developed for some applications.

Presentation of results of queries also requires different techniques. Multimedia data is not very suited to list or record based presentations. Also, in many applications different media sources must be combined to create multimedia presentations with which a user can interact to refine and re-articulate their queries in the emergent semantics environment.

Storage, Matching, and Indexing

In most applications, the size of the multimedia data requires special attention. Commonly the video files can not be stored even using BLOBs. It is common to store file names of multimedia data and compute and store features from the data in the database. In most applications, for each file the number of features that should be stored becomes very large from hundreds to hundreds of thousands for each multimedia data item. These features are used in searching for correct results.

Unlike traditional databases, where records are searched based on exact matches, in MMDB search

requires similarity matching. It is very rare to find a result using exact matching. Search in MMDB usually becomes finding data that has maximum similarity based on features. The similarity techniques [] requires comparing the features in queried data with all the data in a MMDB for evaluating similarity.

Indexing in MMDB for similarity computation requires representing features in a way that can allow fast computation for potentially similar objects. Many high dimensional techniques have been developed for organizing this data. The dimensionality of data, sometimes called curse of dimensionality, poses interesting challenges in such organization.

Key Applications

Multimedia data is becoming ubiquitous. Ranging from photos to videos, multimedia data is becoming part of all applications. Most emerging applications now have some kind of multimedia data that must be considered integral part of the databases. Moreover, emerging applications in all applications areas, ranging from homeland security to healthcare contain rich multimedia data. Based on current trend, it is safe to assume that in very near future, much of the data managed in databases will be multimedia. Some particular application domains where multimedia is natural and will continue dominating are entertainment, news, healthcare, and homeland security.

Future Directions

From structured data to semi-structured data and then to unstructured data, databases are being challenged to deal with increasingly semantic-rich environment. MMDB offer the biggest challenge to databases in terms of bridging the semantic gap.

Increasingly, applications are talking about situation modeling using real life sensor data. These applications combine live sensory data with other information to project current situation and also predict near future for users to take appropriate actions. These databases will require sophisticated tools to manage streaming multimedia data. Research efforts in these areas have already started and are likely to accelerate significantly in the near future.

Cross-references

► Multimedia Data

Recommended Reading

1. Bach J., Paul S., and Jain R. An interactive image management system for face information retrieval. IEEE Trans. Knowl. Data. Eng., Special Section on Multimedia Information Systems., 5(4):619–628, 1993.
2. Gupta A., Weymouth T., and Jain R. Semantic Queries with Pictures, The VIMSYS Model. In Proc. 17th Int. Conf. on Very Large Data Bases, 1991, pp. 3–6.
3. Jain R. Out of the Box Data Engineering Events in Heterogeneous Data (Keynote talk). In Proc. 19th Int. Conf. on Data Engineering, 2003.
4. Jain R. Events and experiences in human centered computing. IEEE Comput, 41(2):42–50, 2008.
5. Katayama N. and Shin'ichi Satoh. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 369–380.
6. Lew M., Sebe N., Djerba C., and Jain R. Content-based multimedia information retrieval: state of the art and challenges. ACM Trans. Multimedia Comp., Comm., and Appl., 2(1):1–19, 2006.
7. Santini S., Gupta A., and Jain R. Emergent semantics through interaction in Image Databases. IEEE Trans. Knowl. Data. Eng.,13(3):337–351, 2001.
8. Santini S. and Jain R. Similarity Measures. IEEE Trans. Pattern. Anal. and Mach. Intell, 21:9, 1999.
9. Smeulders A., Worring M., Santini S., Gupta A., and Jain R. Image Databases at the end of the early years. IEEE Trans Pattern Anal Mach Intell, 23(1), 2001.

Multimedia Information Discovery

► Multimedia Information Retrieval Model

Multimedia Information Retrieval

► Semantic Modeling and Knowledge Representation for Multimedia Data

Multimedia Information Retrieval Model

Carlo Meghini, Fabrizio Sebastiani, Umberto Straccia
The Italian National Research Council, Pisa, Italy

Synonyms

Content-based retrieval; Semantic-based retrieval; Multimedia information discovery

Definition

Given a collection of multimedia documents, the goal of multimedia information retrieval (MIR) is to find the documents that are relevant to a user information need. A multimedia document is a complex information object, with components of different kinds, such as text, images, video and sound, all in digital form.

Historical Background

The vast body of knowledge nowadays labeled as MIR, is the product of several streams of research, which have arisen independently of each others and proceeded largely in an autonomous way, until the beginning of 2000, when the difficulty of the problem and the lack of effective results made it evident that success could be achieved only through integration of methods. These streams can be grouped into three main areas:

The first area is that of *information retrieval* (IR) proper. The notion of IR attracted significant scientific interest from the late 1950s in the context of textual document retrieval. Early characterizations of IR simply relied on an "objective" notion of topic-relatedness (of a document to a query). Later, the essentially subjective concept of relevance gained ground, and eventually became the cornerstone of IR. Nowadays, IR is synonymous with "determination of relevance" [9].

Around the beginning of the 1980s, the area of multimedia documents came into existence and demanded an IR functionality that no classical method was able to answer, due to the *medium mismatch problem* (in the image database field, this is often called the *medium clash problem*). This problem refers to the fact that when documents and queries are expressed in different media, matching is difficult, as there is an inherent intermediate mapping process that needs to reformulate the concepts expressed in the medium used for queries (e.g., text) in terms of the other medium (e.g., images). In response to this demand, a wide range of methods for achieving IR on multimedia documents has been produced, mostly based on techniques developed in the areas of *signal processing* and *pattern matching*, initially foreign to the IR field. These methods are nowadays known as *similarity-based* methods, due to the fact that they use as queries an object of the same kind of the sought ones (e.g., a piece of text or an image) [6]. Originally, the term *content-based* was used to denote these methods, where the content in question was not the content of the multimedia object under study (e.g., the image) but that of the file that hosts it.

The last area is that of *semantic information processing* (SIP) which has developed across the information system and the artificial intelligence communities starting in the 1960s. The basic goal of SIP was the definition of artificial languages that could represent relevant aspects of a reality of interest (whence the appellation *semantic*), and of suitable operations on the ensuing representations that could support knowledge-intensive activities. Since the inception of the field, SIP methods are rooted in first-order mathematical logic, which offers the philosophically well-understood and computationally well-studied notions of syntax, semantics and inference as bases on which to build. Nowadays, SIP techniques are mostly employed in the context of Knowledge Organization Systems. In MIR, SIP methods have been used to develop sophisticate representations of the contents (in the sense of "semantics") of multimedia documents, in order to support the retrieval of these documents based on a logical model. According to this model, user's information needs are predicates expressed in the same language as that used for documents representations, and a document is retrieved if its representation logically implies the query. A wide range of logical models for IR have been proposed, corresponding to different ways of capturing the uncertainty inherent in IR, of expressing document contents, of achieving efficiency and effectiveness of retrieval [4].

To a lesser extent, the database area has also contributed to MIR, by providing indexing techniques for fast access to large collections of documents. Initially, typical structures such as inverted files and B-trees were employed. When similarity-based retrieval methods started to appear, novel structures, such as R- or M-trees were developed in order to support efficient processing of range and k nearest neighbors queries [15].

Foundations

MIR is a scientific discipline, endowed with many different approaches, each stemming from a different branch of the MIR history. All these approaches can be understood as addressing the same problem through a different aspect of multimedia documents.

Documents can be broadly divided from a user perspective into two main categories: simple and complex.

A document is *simple* if it cannot be further decomposed into other documents. Images and pieces of text are typically simple documents. A simple document is an arrangement of symbols that carry information via meaning, thus concurring in forming what is called the *content* of the document. In the case of text, the symbols are words (or their semantically significant fractions, such as stems, prefixes or suffixes), whereas for images the symbols are colors and textures. Simple documents can thus be characterized as having two parallel dimensions: that of *form* (or *syntax*, or *symbol*) and that of *content* (or *semantics*, or *meaning*). The form of a simple document is dependent on the medium that carries the document. On the contrary, the meaning of a simple document is the set of states of affairs (or "worlds") in which it is true, and is therefore medium-independent. For instance, the meaning of a piece of text is the set of (spatio-temporally determined) states of affairs in which the assertions made are true, and the meaning of an image is the set of such states of affairs in which the scene portrayed in the image indeed occurs.

Complex documents (or simply documents) are structured sets of simple documents. This leads to the identification of *structure* as the third dimension of documents. Document structure is typically a binary relation, whose graph is a tree rooted at the document and having the component simple documents as leaves. More complex structures may exist, for instance those requiring an ordering between the children of the same parent (such as between the chapters of a book), or those having an arity greater than 2 (such as synchronization amongst different streams of an audio-visual document).

Finally, documents, whether simple or complex, exist as independent entities characterized by (meta-) attributes (often called *metadata* in the digital libraries literature), which describe the relevant properties of such entities. The set of such attributes is usually called the *profile* of a document, and constitutes the fourth and last document dimension.

Corresponding to the four dimensions of documents just introduced, there can be four categories of retrieval, each one being a projection of the general problem of MIR onto a specific dimension. In addition, it is possible, and in some cases desirable, to combine different kinds of retrieval within the same operation.

Retrieval based on document structure does not really lead to a genuine discovery, since the user must have already seen (or be otherwise aware of) the sought document(s) in order to be able to state a predicate on their structure. Retrieval based on document profile, from a purely logical point of view, is not different from content-based retrieval and in fact many metadata schema used for document description (notable, the Dublin Core Metadata Set) include attributes of both kinds.

Form-based Multimedia Information Retrieval

The retrieval of information based on form addresses the syntactic properties of documents. In particular, form-based retrieval methods automatically create the document representations to be used in retrieval by extracting low-level features from documents, such as the number of occurrences of a certain word in a text, or the energy level in a certain region of an image. The resulting representations are abstractions which retain that part of the information originally present in the document that is considered sufficient to characterize the document for retrieval purposes. User queries to form-based retrieval engines may be documents themselves (this is especially true in the non-textual case, as this allows overcoming the medium mismatch problem), from which the system builds abstractions analogous to those of documents. Document and query abstractions are then compared by an appropriate function, aiming at assessing their degree of similarity. A document ranking results from these comparisons, in which the documents with the highest scores occur first.

In the case of text, form-based retrieval includes most of the traditional IR methods, ranging from simple string matching (as used in popular Web search engines) to the classical *tf-idf* term weighting method, to the most sophisticated algorithms for similarity measurement. Some of these methods make use of information structures, such as thesauri, for increasing retrieval effectiveness. However, what makes them form-based retrieval methods is their relying on a form-based document representation. Two categories of queries addressing text can be distinguished:

1. *Full-text* queries, each consisting of a *text pattern*, which denotes, in a deterministic way, a set of texts; when used as a query, the text pattern is supposed to retrieve any text layout belonging to its denotation.
2. *Similarity* queries, each consisting of a text, and aimed at retrieving those text layouts which are similar to the given text.

In a full-text query, the text pattern can be specified in many different ways, e.g., by enumeration, via a regular expression, or via *ad hoc* operators specific to text structure such as proximity, positional and inclusion operators [9].

Queries referring to the form dimension of images are called *visual* queries, and can be partitioned as follows:

1. *Concrete visual queries:* These consist of full-fledged images that are submitted to the system as a way to indicate a request to retrieve "similar" images; the addressed aspect of similarity may concern color [2,7], texture [8,14], appearance [12] or combination thereof [13].
2. *Abstract visual queries:* These are artificially constructed image elements (hence, "abstractions" of image layouts) that address specific aspects of image similarity; they can be further categorized into:
   a. *Color queries:* specifications of color patches, used to indicate a request to retrieve those images in which a similar color patch occurs [6,7].
   b. *Shape queries:* specifications of one or more shapes (closed simple curves in the 2D space), used to indicate a request to retrieve those images in which the specified shapes occur as contours of significant objects [6,11].
   c. Combinations of the above [2].

Visual queries are processed by matching a vector of features extracted from the query image, with each of the homologous vectors extracted from the images candidate for retrieval. For concrete visual queries, the features are computed on the whole image, while for abstract visual queries only the features indicated in the query (such as shape or color) are represented in the vectors involved. For each of the above categories of visual queries, a number of different techniques have been proposed for performing image matching, depending on the features used to capture the aspect addressed by the category, or the method used to compute such features, or the function used to assess similarity.

Semantic Content-based Multimedia Information Retrieval

On the contrary, semantic-based retrieval methods rely on symbolic representations of the meaning of documents, that is descriptions formulated in some suitable knowledge representation language, spelling out the truth conditions of the involved document. Various languages have been employed to this end, ranging from net-based to logical. Description Logics [1], or their Semantic Web syntactic forms such as OWL, are contractions of the Predicate Calculus that are most suitable candidates for this role, thanks to their being focused on the representation of concepts and to their computational amenability. Typically, meaning representations are constructed manually, perhaps with the assistance of some automatic tool; as a consequence, their usage on collections of remarkable size (text collections can reach nowadays up to millions of documents) is not viable. The social networking on which Web 2.0 is based may overcome this problem, as groups of up to thousands of users may get involved in the collaborative indexing process (flicker).

While semantic-based methods explicitly apply when a connection in meaning between documents and queries is sought, the status of form-based methods is, in this sense, ambiguous. On one hand, these methods may be viewed as pattern recognition tools that assist an information seeker by providing associative access to a collection of signals. On the other hand, form-based methods may be viewed as an alternative way to approach the same problem addressed by semantic-based methods, that is deciding relevance, in the sense of connection in meaning, between documents and queries. This latter, much more ambitious view, can be justified only by relying on the assumption that there be a systematic correlation between "sameness" in low-level signal features and "sameness" in meaning. Establishing the systematic correlation between the expressions of a language and their meaning is precisely the goal of a *theory of meaning* (see, e.g., [5]), a subject of the philosophy of language that is still controversial, at least as far as the meaning of natural languages is concerned. So, pushed to its extreme

consequences, the ambitious view of form-based retrieval leads to viewing a MIR system *as an algorithmic simulation of a theory of meaning*, in force of the fact that the sameness assumption is relied upon in every circumstance, not just in the few, happy cases in which everybody's intuition would bet on its truth. At present, this assumption seems more warranted in the case of text than in the case of non-textual media, as the representations employed by form-based textual retrieval methods (i.e., vectors of weighted words) come much closer to a semantic representation than the feature vectors employed by similarity-based image retrieval methods. Irrespectively of the tenability of the sameness assumption, the identification of the alleged syntactic-semantic correlation is at the moment a remote possibility, so the weaker view of form-based retrieval seems the only reasonable option.

Mixed Multimedia Information Retrieval

Suppose a user of a digital library is interested in retrieving all documents produced after January 2007, containing a critical review on a successful representation of a Mozart's opera, and with a picture showing Kiri in a blueish dress. This need addresses all dimensions of a document: it addresses structure because it states conditions on several parts of the desired documents; it addresses profile because it places a restriction on the production date; it addresses form- (in particular color-) and semantic-based image retrieval on a specific region of the involved image (the region must be blue and represent the singer Kiri) as well as on the whole image (must be a scene of a Mozart's opera); it addresses from-based text retrieval by requiring that the document contains a piece of text of a certain type and content. This is an example of mixed MIR, allowing the combination of different types of MIR in the context of the same query [10].

Emerging standards in multimedia document representation (notably, the ISO standard MPEG21) address all of the dimensions of a document. Consequently, their query languages support more and more mixed MIR.

Key Applications

Nowadays, MIR finds its natural context in *digital libraries,* a novel generation of information systems [3], born in the middle of the 1990s as a result of the First Digital Library Initiative. Digital Libraries are large collections of multimedia documents which are made on-line available on global infrastructures for discovery and access. MIR is a core service of any DL, addressing the discovery of multimedia documents.

Cross-references

► Information Retrieval

Recommended Reading

1. Baader F., Calvanese D., McGuiness D., Nardi D., and Patel-Scheneider P. (eds.). The description logic handbook. Cambridge University Press, Cambridge, 2003.
2. Bach J.R., Fuller C., Gupta A., Hampapur A., Horowitz B., Humphrey R., Jain R., and Shu C.-F. The Virage image search engine: an open framework for image management. In Proc. 4th SPIE Conf. on Storage and Retrieval for Still Images and Video Databases, 1996, pp. 76–87.
3. Candela L., Castelli D., Pagano P., Thanos C. Ioannidis Y., Koutrika G., Ross S., Schek H.-J., and Schuldt H. Setting the foundations of digital libraries. The DELOS manifesto. D-Lib Magazine, 13(3/4), March/April 2007.
4. Crestani F., Lalmas M., and van Rijsbergen C.J. (eds.). Logic and uncertainty in information retrieval: advanced models for the representation and retrieval of information, The Kluwer International Series On Information Retrieval, vol. 4. Kluwer Academic, Boston, MA, October 1998.
5. Davidson D. Truth and meaning. In Inquiries into truth and interpretation. Clarendon, Oxford, UK, 1991, pp. 17–36.
6. Del Bimbo A. Visual Information Retrieval. Morgan Kaufmann, Los Altos, CA, 1999.
7. Faloutsos C., Barber R., Flickner M., Hafner J., and Niblack W. Efficient and effective querying by image content. J. Intell. Inform. Syst., 3:231–262, 1994.
8. Liu F. and Picard R.W. Periodicity, directionality, and randomness: Wold features for image modelling and retrieval. IEEE Trans. Pattern Analysis Machine Intell., 18(7):722–733, 1996.
9. Manning C.D., Raghavan P., and Schütze H. An Introduction to Information Retrieval. Cambridge University Press, Cambridge, 2007.
10. Meghini C., Sebastiani F., and Straccia U. A model of multimedia information retrieval. J. ACM, 48(5):909–970, 2001.
11. Petrakis E.G. and Faloutsos C. Similarity searching in medical image databases. IEEE Trans. Data Knowl. Eng., 9(3): 435–447, 1997.
12. Ravela S. and Manmatha R. Image retrieval by appearance. In Proc. 20th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1997, pp. 278–285.
13. Rui Y., Huang T.S., Ortega M., and Mehrotra S. Relevance feedback: a power tool for interactive content-based image retrieval. IEEE Trans. Circuits Syst. Video Tech., 8(5):644–655, September 1998.
14. Smith J.R. and Chang S.-F. Transform features for texture classification and discrimination in large image databases. In Proc. Int. Conf. Image Processing, 1994, pp. 407–411.
15. Zezula P., Amato G., Dohnal V., and Batko M. Similarity Search: The Metric Approach. Springer, Berlin, 2006.

Multimedia Metadata

Frank Nack
University of Amsterdam, Amsterdam,
The Netherlands

Synonyms

New media metadata; Hypermedia metadata; Meta-knowledge; Mixed-media

Definition

Multimedia is media that utilizes a combination of different content forms. In general, a multimedia asset includes a combination of at least two of the following: text, audio, still images, animation, video, and some sort of interactivity. There are two categories of multimedia content: linear and non-linear. Linear content progresses without any navigation control for the viewer, such as a cinema presentation. Non-linear content offers users interactivity to control progress. Examples are computer games or computer based training applications. Non-linear content is also known as hypermedia content.

Metadata is data about data of any sort in any media, describing an individual datum, content item, or a collection of data including multiple content items. In that way, metadata facilitates the understanding, characteristics, use and management of data.

Multimedia metadata is structured, encoded data that describes content and representation characteristics of information-bearing multimedia entities to facilitate the automatic or semiautomatic identification, discovery, assessment, and management of the described entities, as well as their generation, manipulation, and distribution.

Historical Background

The first appearance of the idea of mixed media was Vannevar Bush's "Memex" system ("memory extender"). In his article "As we may think" [1], published 1945 in the "The Atlantic Monthly," he proposed a device that is electronically linked to a library, able to display books and films from the library and automatically follow cross-references from one work to another. The Memex was the first simple and elegant approach towards multimedia information access. Further important steps toward the use of multimedia in digital systems in the 1960s were the introduction of "hyperlinks and hypermedia" by Ted Nelson [2], which allowed the generation of non-linear presentations, the development of the mouse by Douglas Engelbart [3], which supported the direct manipulation of objects on a computer screen, and the invention of the GUI, developed at Xerox Parc, which introduced media into computing.

In particular, it was the development of the web and digital consumer electronics that allowed computing to leave the realm of research or government organizations and enter society in large. The development of personal computers in the 1980s enabled desktop publishing, which further enhanced in the 1990s into adequate manipulation software for digital media, such as Adobe's Photoshop and Flash, or Appel's Final Cut Express. Yet, only the development of new distribution platforms in the 1990s, such as CD-ROM, DVD, and essentially the World Wide Web (Web) in combination with the rise of media technologies, such as the digital photo and video camera, or midi and MP3 player, stimulated the swift growth of digital media in mixed form. The growing amount of mixed media available to the public, and more recently provided by the public in form of user-generated content, requested for effective access mechanisms, which resulted in a steady development of content description technologies, mainly based on metadata.

Over the years, a number of metadata standards have been developed, which address various aspects of multimedia data, such as

- Low-level features: usually automatically extracted from the content.
- Semantic features: describing high-level concepts in form of key-word, ratings and links.
- Structure and identification: describing the spatial and temporal arrangement of one or more multimedia assets.
- Management: describing the information gathered during the life cycle of the multimedia asset, such as information about reuse, archiving, and rights management.

The major organizations who contributed to the development of standards are: the Dublin Core Metadata Initiative [4], the World Wide Web Consortium (W3C) [5], the Society for Motion Pictures and Television Engineering (SMPTE) [6], the Moving Picture Expert Group (MPEG) [7], the TV-Anytime Consortium [8], and the International Press Telecommunications

Council (IPTC) [9]. The common definition language between all theses languages is in one or the other way the Extensible Markup Language (XML) [10], defined by W3C.

The two major approaches towards open standards for multimedia metadata, with respect to content description and distribution for multimedia assets, were certainly provided by the W3C and MPEG, a working group of ISO/IEC charged with the development of video and audio encoding standards.

The W3C provided, besides the well known markup languages HTML, XHTML, CSS, SVG, in particular the Synchronized Multimedia Integration Language (SMIL). SMIL enables simple authoring of interactive audiovisual presentations, which integrate streaming audio and video with images, text or any other media type. SMIL [11] is an HTML-like language facilitating the authoring of a SMIL application by using a simple text-editor. SMIL 1.0 received recommendation status in 1998, SMIL 2.0 in 2005, and SMIL 3.0 is under development while this article is written.

Work in the Media Annotation Work has started in September 2008. This working group (http://www.w3.org/2008/01/media-annotations-wg.html) is chartered to provide a simple ontology to support cross-community data integration of information related to media objects on the Web, as well as an API to access the information. In addition there is the Media Fragments Working Group (http://www.w3.org/2008/WebVideo/Fragments/), which address temporal and spatial media fragments in the Web using Uniform Resource Identifiers (URI). Both groups should finish their work by June 2010.

MPEG's contribution to multimedia metadata are certainly the three standards MPEG-4 [12], MPEG-7 [13] and MPEG-21 [14].

With MPEG-4 the group entered the realm of media content, arisen due to the growing need for content manipulation and interaction, and expanded MPEG-1 to support video/audio "objects," 3D content, low bitrate encoding and support for Digital Rights Management. With respect to multimedia metadata MPEG4 also provides content authors with a textual syntax for the MPEG-4 Binary Format for Scenes (BIFS) to exchange their content with other authors, tools, or service providers. First, XMT is an XML-based abstraction of the object descriptor framework for BIFS animations. Moreover, it also respects existing practice for authoring content, such as SMIL, HTML, or Extensible 3D (X3D) by allowing the interchange of the format between a SMIL player, a Virtual Reality Modeling Language (VRML) player, and an MPEG player through using the relevant language representations such as XML Schema, MPEG-7 DDL, and VRML grammar. As such, the XMT serves as a unifying framework for representing multimedia content where otherwise fragmented technologies are integrated and the interoperability of the textual format between them is facilitated.

In the mid 1990s, the need for retrieving and manipulating digital media content form exploding digital libraries requested new ways of describing multimedia content on deeper semantic granularity, which resulted in MPEG-7, the multimedia content description standard. MPEG-7 provides a large set of descriptors and description schemata for video (part 3), audio (part 4) and multimedia content, including its presentation (part 5). All schemata are described in the Description Definition Language (DDL), which is modeled on XML-Schema, a schemata language developed by the WC3. Since MPEG-7 tries to establish the richest and most versatile set of audio-visual feature description structures by embracing standards such as SMPTE, or PTC, a 1:1 mapping to the text-oriented XML schema language could not be achieved (see [15] for a detailed description of existing DDL problems). However, the goal to be a highly interoperable standard among well-known industry standards and related standards of other domains – such as the area of digital libraries and ontologies using RDF or Dublin Core, was a useful exercise.

The beginning of the 21st century established an even faster exchange of multimedia data via the web, as higher bandwidth as well as access to high quality data became a commodity. The media businesses, such as the record or film industry, feared, due to peer-to-peer technology, for their markets and requested strict digital rights management enforcement. MPEG reacted with MPEG-21: MPEG describes this standard as a multimedia framework.

Both the W3C as well as MPEG provide open standards, which are able to describe adaptive content, represented in a single file that can be targeted to several platforms, such as mobile, broadband or the web. Both organizations compete with highly successful but rather proprietary industry standards, such as Adobe's Flash standard.

The latest trend on multimedia metadata also reflects the trend towards user-generated content, namely social tagging. A tag is a keyword or term

associated with or assigned to a piece of information (a picture, a map, a video clip etc), which enables keyword-based classification and search. The advantage of tagging is its ease of use. This approach, though highly popular (e.g., in YouTube [16] and Flickr [17]), carries serious problems. Typically there is no information about the semantics of a tag, no matter if it is a single tag or a bag of tags. Additionally, different people may use drastically different terms to describe the same concept. This lack of semantic distinction can lead to inappropriate connections.

Foundations

The essential models describing the internal structures of multimedia compositions and the related production processes are: the Dexter Hypertext Reference Model [18], the Amsterdam Hypermedia Model [19], HyTime [20], the reference model for intelligent multimedia presentation systems (IMMPSs) [21], MPEG-4, SMIL, and the model of Canonical processes of media production [22].

The basic five functionalities that every multimedia system needs to address are: media content, layout, timing, linking and adaptivity.

Media Content

Readers who are interested in the fundamentals of the single media elements (i.e., their low-level as well as high-level feature descriptions) are referred to the articles on audio metadata, image metadata and video metadata in this encyclopedia.

Layout

Layout deals with the arrangement and style treatment of media on a screen. This means that layout determines how a particular media item is presented at a point in time and how it is rendered when activated. A multimedia presentation layout describes "the look and feel" of a composite of all of its media components. Thus, layout adds the semantic organization that enables a viewer to quickly and efficiently absorb the multiple content streams in a multimedia presentation.

There are in general three approaches towards multimedia layout, namely

1. *Embedded Layout,* where all layout decisions are resolved at media creation time and then performed by the presentation. Here the control lies ultimately by the designer of the presentation and there is no control, besides the required rendering, at the side of the media player.
2. *Dynamic Layout*, where all layout is dynamically determined by the user's media player, depending on the multimedia document structure or the timeline of the presentation. Here, the actual visual design is mainly in the hand of the media player rather than the presentation's designer.
3. *Compositing Layout*, which decouples media content from media placement. Here, a presentation is understood as a composite of relatively autonomous objects, where each uses embedded or dynamic layout models, which are then positioned into a arrangement by a presentation designer.

The essential classes and their attributes that describe a layout are:

- A root and several region elements, which establish the primary connection between media objects elements and the layout structure.
- Basic layout classes, such as referencing (region name and ID), scaling (z-index fit), positioning (width, height, top, left, bottom, right), background (back ground color, show back ground), and audio (sound level).

Timing

Timing deals with issues on how elements in a multimedia presentation get scheduled. Moreover, once an element is active, it needs to be determined how long it will be scheduled. The aim in a multimedia presentation is to go beyond the timing concepts known from audio and video objects.

Media timing: Media objects in multimedia presentations can either be *discrete* (e.g., text, with no implicit duration) or *continuous* (e.g., a music object, with an explicit duration defined within its encoding).

Presentation timing: A reference list to one or more media objects, describing timing primitives that determine the start and end time relative to one another.

There are basically 4 different ways of defining the active period of an object:

1. *Implicit duration*, as defined when the object was created (e.g., length of a video in sec).
2. *Explicit duration*, which describes the actual duration of the media object in the application (e.g., the actual duration might be shorter or longer than the implicit duration).
3. *Active duration*, which allows repetitions or other temporal manipulations of a media object.

4. *Rendered duration*, which describes the persistence of a media object at the end of its active duration.

There are various ways to describe time in values, such as full clock value (e.g., 7:45:23.76, where the last two items present ms), partial clock values (any sort of short base notation), time count values (numbers with a additional type string, e.g., 10S for "10 s"), and time context values, which are represented in three parts: a date field (YYY:MM:DD), a time field, and a time zone field.

Usually, a type of synchronization is required, as media objects start in relation to the container they belong to. A child element in a parallel container starts relative to the start time of the parent, whereas child elements in a sequential container are started relative to the end of their predecessor.

Linking

Linking defines and activates a non-linear navigation structure within and across documents.

The simplest form of a link is a pointer. The pointer defines an address of a document (e.g., a URI) and, optionally, an offset within the document. The element that identifies that a link exists is called a source anchor. If the anchor points to anything other than the beginning of a document, this anchor is called a destination anchor. The typical elements in HTML for linking are the <a> element, to define the source anchor and link address, and the <area> </area> element, which is roughly the same, but it is applied only to a part of a media object. The basic linking attributes define the uri (href), the source and destination state (e.g., play or pause), the external or target state (e.g., true or false, the display environment) and the impact of link activation on the source and destination presentation (e.g., as non-negative percentage).

Both source and destination anchor need to express temporal moments, as their activation depends on the temporal behavior of the application. The three key temporal moments to be addressable are: the destination is already active, the destination is inactive, and the destination is inactive and the begin time is unresolved.

Finally, the link needs some attribute that describe geometry, as the linking into a region of a media item is possible. The core attributes are shape (values can be rectangle, circle or poly). The size and position of the anchor are defined via the origin of the coordinate space (the 0.0 point) and the resolution of the display device (support of rendering).

Adaptivity

The aim of multimedia presentations is usually to adapt them to the needs of the user, which might address either the runtime environment available to the user, or the personalized presentation wishes by the user.

There are four techniques to customize information in a presentation:

1. *Minimum set:* The multimedia presentation assumes a minimum set of performance, and device and user characteristics and the presentation document is designed based on this lowest denominator set (manageable solution but usually no compelling content)
2. *Multiple presentation set*: Each presentation represents a quality level, which the user selects on runtime (this one-size-fits-all approach is a dead end for the current trend towards portable and quality mix devices).
3. *Over-specified presentation*: All of the potential media items are available and it is the media player at the client side which makes a selection at run time (demands too much during the making phase).
4. *Control presentation*: The presentation contains pointers to all potential alternatives and only those used by the user would be sent from the server to the client (the advantage is that the presentation does not need to send copies of each of the various data streams across the network).

The essential elements a system would need to provide any of the above techniques are:

- Switch: which establishes a collection of alternatives for an interactive multimedia presentation;
- System control attributes, such as sys_language, sys_captions, sys_bitrate, sys_screensize, syt_cpu, etc.

Key Applications

Multimedia metadata is useful for the creation, manipulation, retrieval and distribution of mixed media sources within domains, such as

- The creative industries (e.g., fine arts, entertainment, commercial art, journalism, games, etc)

- The entertainment industries (e.g., special effects in movies and animations)
- Education (e.g., in computer based training courses and computer simulations, military or industrial training)
- Mathematical and scientific research (e.g., modeling and simulation)
- Medicine (e.g., virtual surgery or simulations of virus spread, etc)

Cross-references

► Audio Metadata
► Image Metadata
► Video Metadata

Recommended Reading

1. Bordegoni M., Faconti G., Maybury M.T., Rist T., Ruggieri S., Trahanias P., and Wilson M. A standard reference model for intelligent multimedia presentation systems (1997). Available at http://kazan.cnuce.cnr.it/papers/abstracts/9708.IJCAI97.Immps.html
2. Bush V. As we may think. Atl. Mon., 176(1):101–108, 1945.
3. Engelbart D. (1968). Available at http://sloan.stanford.edu/mousesite/1968Demo.html
4. Flickr. Available at http://www.flickr.com/
5. Gronbaek K. and Trigg R.H. Design issues for a dexter-based hypermedia system. Commun ACM., 37(2):41–49, 1994.
6. Hardman L., Bulterman D.C.A., and van Rossum G. The amsterdam hypermedia model: adding time and context to the dexter model. Commun. ACM., 37(2):5062, February 1994.
7. Hardman L., Obrenovic Z., Nack F., Kerherve B., and Piersol K. Canonical processes of semantically annotated media production. Multimedia Syst., 14(6):427–433, 2008.
8. Information processing – Hypermedia/Time-based structuring language (HyTime) – 2nd ed., ISO/IEC 10744:1997, WG8 PROJECT: JTC1.18.15.1. Available at http://www1.y12.doe.gov/capabilities/sgml/wg8/document/n1920/
9. MPEG-4: ISO/IEC JTC1/SC29/WG11 N4668 March 2002. Available at http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm
10. MPEG-21: ISO/IEC JTC1/SC29/WG11/N5231 Shanghai, October 2002. Available at http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm
11. MPEG-7: ISO/IEC JTC1/SC29/WG11N6828 Palma de Mallorca, October 2004. Available at http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm
12. Nack F., van Ossenbruggen J., and Hardman L. That obscure object of desire: multimedia metadata on the web (Part II). IEEE MultiMedia, 12(1):54–63, 2005.
13. Nelson T.H. A File Structure for the Complex, the Changing, and the Intermediate. In The New Media Reader, Noha Wardrip-Fruin & Nick Montfort. The MIT Press, Cambridge, MA, 2003, pp. 133–146.
14. SMIL. Available at http://www.w3.org/AudioVideo/
15. The dublin core metadata initiative. Available at http://www.dublincore.org/
16. The extensible markup language (XML). Available at http://www.w3.org/XML/
17. The international press telecommunications council [IPTC). Available at http://www.iptc.org/pages/index.php
18. The moving picture expert group (MPEG). Available at http://www.chiariglione.org/mpeg/
19. The society for motion pictures and television engineering (SMPTE). Available at http://www.smpte.org/home/
20. The TV-anytime consortium. Available at http://www.tv-anytime.org/
21. Youtube. Available at http://www.youtube.com/

Multimedia Presentation Databases

V. S. Subrahmanian, Maria Vanina Martinez, DR. Reforgiato
University of Maryland, College Park, MD, USA

Synonyms

Multimedia presentation databases

Definition

A multimedia presentation consists of a set of media objects (such as images, text objects, video clips, and audio streams) presented in accordance with various temporal constraints specifying when the object should be presented, and spatial constraints specifying where the object should be presented on a screen. Today, multimedia presentations range from the millions of PowerPoint presentations users have created the world over, to more sophisticated presentations authored using tools such as Macromedia Director. Multimedia presentation databases provide the mechanisms needed to store, access, index, and query such collections of multimedia presentations.

Historical Background

Multimedia presentations have been in existence since the 1980s, when PowerPoint emerged as a presentation paradigm and animated computer video games started gaining popularity. Both of these paradigms allowed a multimedia presentation author to specify a set of objects (collections of images, video, audio clips, and text objects) and then specify how these objects should be presented. These objects could be presented in accordance with some temporal constraints that describe when and in conjunction with which other

objects a given object should be presented. As an increasing number of authoring frameworks came into being, accompanied by an increasing need to create, collaborate on, and share presentations, the notion of multimedia presentations as a programming paradigm gradually came into existence.

Buchanan and Zellweger [6] were one of the first to recognize the need to treat multimedia presentations in a rigorous framework. They recognized that presentations consisted of a set of media objects, and they proposed presenting these objects in accordance with some very simple precedence constraints.

Later, Candan et al. [8,7] extended the framework of Buchanan and Zellweger by describing a multimedia presentation as a set of media objects together with a very rich, but polynomially computable set of spatial and temporal constraints defining their presentation. They also showed how to help a presentation author identify when their presentation specifications were inconsistent and to minimally modify the presentation constraints so that consistency was restored. Related work by their co-authors showed how to deliver these presentations across a network in the presence of spatial and temporal constraints.

Adali et al. [1] introduced a relational model of data to support interactive multimedia presentations and define a variant of the relational algebra that allows users to dynamically query and create new presentations using parts of existing ones, generalizing select, project, and join operations. Lee et al. [11] present a graph data model for the specification of multimedia presentations, together with two icon-based based graphical query languages for multimedia presentations and the GCalculus (Graph Calculus), a relational calculus-style language that formalizes the use of temporal operators for querying presentation graphs that takes the content of a presentation into account. [5,4] propose methods to present the answer of a query to a multimedia database as a multimedia presentation. [9] focuses on specializations and improvements of the above methods when querying databases consisting solely of PowerPoint information.

[10] treats multimedia presentations like a temporal database, and supports querying and reuse of parts of existing presentations. It considers algebraic operators such as insert, delete, and join, as well as user interface operations such as Fast Forward/Rewind, Skip, and links to other presentation databases, etc. [13] focuses on indexing and retrieving complex Flash movies. A generic framework called FLAME (FLash Access and Management Environment) based on a 3-layer structure is presented to address this problem. This framework mines and understands the contents of the movies to address the representation, indexing and retrieval of the expressive movie elements, including heterogeneous components, dynamic effects, and the way in which the user can interact with the movie.

Foundations

A multimedia presentation consists of a set O of media objects and a set of constraints on the presentation of objects in O. A media-object o is a file such as an image file, a video file, a text file, or an audio file. Each type of file is assumed to have an associated player. For example, a video file may have QuickTime or the Windows Media players as its associated player.

The constraints associated with o fall into two categories: temporal and spatial constaints.

On the temporal side, each object o in O has two associated variables, $st(o)$ and $et(o)$ denoting, respectively, the start time and end time at which media object o is presented using its associated players. The *temporal specification* associated with a presentation is a set of constraints of the form

$$x - y <= c$$

where x, y are variables of the form $st(o_i)$, $et(o_j)$ and c is some constant. For example, if o is a video file, and there exists the constraint $et(o') - st(o) = 0$, then this means that the video object o should start playing as soon as object o' finishes being played out.

Likewise, on the spatial side, each object o in O has four variables $llx(o)$, $lly(o)$, $urx(y)$, $ury(o)$ denoting the x coordinate of the lower left corner of object o, the y coordinate of the lower left corner of object o, and likewise for the upper right corner's x and y coordinates. [8] presents algorithms to check the consistency of spatial and temporal constraints, and to minimally modify the spatial/temporal constraints when they are inconsistent. In addition, each object o in O has an associated set of properties that can be stored (and queried) using any object oriented database management system.

A *multimedia presentation database* $\boldsymbol{M}$ consists of a set of multimedia objects (and their associated spatial

and temporal constraints). [1] defines methods to query such multimedia presentation databases.

Consider the simplest operation: selection. Suppose the query is "*Select all objects o in* ***M*** such that *C[o]* holds and such that $st(o) > 10$." In this case, the goal is to look at each multimedia presentation m in ***M***, and eliminate all objects o from m such that $st(o) < 10$. Also, it is necessary to eliminate all remaining objects such that *C[o]* does not hold. The objects that survive this elimination process must be presented in accordance with the original set of temporal and spatial constraints present in m. If m^* denotes the modification of m in this way, then $\sigma_C(\boldsymbol{M}) = \{m^* \mid m \text{ in } \boldsymbol{M}\}$.

In addition, [1] defines other operations that allow videos to be concatenated together using various chromatic composition operators (such as smoothing, fading, etc.), methods to perform joins across videos, and methods to execute other kinds of relational style operations.

Key Applications

There are numerous possible applications for multimedia presentation databases. A simple application is an engine to query PowerPoint presentations, of which millions exist in the world today. [9] proposes a PowerPoint database query algebra.

Another application is in the area of digital rights management. Suppose a major record company wants to identify all multimedia documents on the web that contain a clip of their copyrighted music. This corresponds to a select query on all multimedia documents on the web. The result would be a set of multimedia documents, some of which might infringe on the copyright holder's rights. The same might apply to online video on sources such as YouTube where it is not uncommon to find copyrighted material. The ability for an entertainment company to find gross violations of their copyright by searching through YouTube archives is critical.

Future Directions

As video games become ever more common, and as virtual worlds such as Second Life become increasingly popular with users, the ability to query games and *avatar* based systems will become increasingly important.

In addition, methods to index multimedia presentation databases are in their very infancy. Taking into account the graph based nature of presentation databases such as those in [1,11,2], it is important to note that methods to index graphs may have a role to play. However, multimedia presentations are more complex than labeled directed graphs because presentation constraints can potentially be satisfied in many different ways.

Cross-references

▶ Spatial Constraints
▶ Temporal Constraints

Recommended Reading

1. Adali S., Sapino M.L., and Subrahmanian V.S. A multimedia presentation algebra. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 121–132.
2. Adali S., Sapino M.L., and Subrahmanian V.S. Interactive multimedia presentation databases, I: algebra and query equivalences. Multimedia Syst., 8(3):212–230, 2000.
3. Bailey B., Konstan J.A., Cooley R., and Dejong M. Nsync – a toolkit for building interactive multimedia presentations. In Proc. 6th ACM Int. Conf. on Multimedia, 1998, pp. 257–266.
4. Baral C., Gonzalez G., and Nandigam A. SQL+D: extended display capabilities for multimedia database queries. In Proc. 6th ACM Int. Conf. on Multimedia, 1998, pp. 109–114.
5. Baral C., Gonzalez G., and Son T.C. Design and implementation of display specification for multimedia answers. In Proc. 14th Int. Conf. on Data Engineering, 1998, pp. 558–565.
6. Buchanan M.C. and Zellweger P. Automatically generating consistent schedules for multimedia documents. Multimedia Syst., 1(2):55–67, 1993.
7. Candan K., Lemar E., and Subrahmanian V.S. View management in multimedia databases. VLDB J., 9(2):131–153, 2000.
8. Candan K.S., Prabhakaran B., and Subrahmanian V.S. CHIMP: a framework for supporting distributed multimedia document authoring and presentation. In Proc. 4th ACM Int. Conf. on Multimedia, 1996, pp. 329–340.
9. Fayzullin M. and Subrahmanian V.S. An algebra for powerpoint sources. Multimedia Tools Appl. J., 24(3):273–301, 2004.
10. Jiao B. Multimedia presentation database system. In Proc. 8th ACM Int. Conf. on Multimedia, 2000, pp. 515–516.
11. Lee T., Sheng L., Bozkaya T., Balkir N.H., Özsoyoglu Z.M., and Özsoyoglu G. Querying multimedia presentations based on content. In Readings in Multimedia Computing and Networking, K. Jeffay, H. Zhang (eds.). Morgan Kaufmann Publishers, San Francisco, CA, USA, 2001, pp. 413–437.
12. Wirag S. Modeling of adaptable multimedia documents. In Proc. Interactive Distributed Multimedia Systems and Telecommunication Services, International Workshop, LNCS vol. 1309, 1997, pp. 420–429.
13. Yang J., Li Q., Wenyin L., and Zhuang Y. Content-based retrieval of FlashTM movies: research issues, generic framework, and future directions. Multimedia Tools Appl., 34(1):1–23, 2007.

Multimedia Resource Scheduling

Jeffrey Xu Yu
Chinese University of Hong Kong, Hong Kong, China

Definition

Multimedia information systems are different from the traditional information systems, where continuous media (audio/video) requests special storage and delivery requirements due to (i) the large transfer rate, (ii) the storage space required, and (iii) the real-time and continuous nature. Due to the special characteristic of continuous media, different types of scheduling are proposed, namely, the disk scheduling and stream scheduling. On one hand, the disk scheduling is to tackle both the large storage space and the corresponding large transfer rate requirements. On the other hand, the stream scheduling is to schedule requests from multiple clients, in order to minimize the delay in satisfying the requests. It attempts to support as many requests as possible, and at the same time, keep the real-time and continuous nature.

Historical Background

Continuous media adds additional requirements to the traditional information systems. In order to satisfy these new requirements, new scheduling algorithms are proposed.

Disk scheduling is designed to achieve the low service latency and high disk throughput requirements [14]. Continuous media, such as audio and video, adds additional real-time constraints to the disk scheduling problem. There are two categories of disk scheduling, namely, single disk scheduling and multiple disk scheduling. The disk scheduling algorithms, for continuous media, can possibly adapt one of the conventional disk scheduling strategies, which include round-robin, SCAN [14], and EDF [12]. For single disk scheduling for continuous media, there are SCAN-EDF [13], and sorting-set [8,16] scheduling strategies. For multiple disk scheduling for continuous media, multiple disks can be accessed in parallel. The increased parallelism is mainly used to support more streams. There are two categories of multiple disk scheduling strategies, which are stripe data across the disks and replicate data across the disks. In additional, there are three schema for accessing striped data, which are called, striped retrieval, split-striped retrieval [15], and cyclic retrieval [2,3].

Stream scheduling (or session scheduling) is designed to effectively allocate and share server and network resources. It aims at supporting as many clients as possible simultaneously, within the limited server and network resources, by assigning multiple client requests to a shared data stream. Three policies are proposed to effectively select the multimedia to be shared: FCFS, MQL, and FCFS-n [10]. In [6,7], sharing is studied when a client pauses/resumes while viewing a long video. A two-level hierarchical scheduling policy is proposed to deal with time-varying load, such as the load at peek time and off-peek time, regarding the preservation of resources for the future requests [1]. In the two-level hierarchical scheduling, at the higher level, it focuses on channel allocation with consideration of how many requests will come in near future; at the lower level, it focuses on selecting clients to be served with consideration of the clients waiting time.

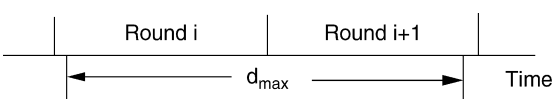
Foundations

There are mainly two types of resource scheduling in a multi-stream environment: (i) disk scheduling, and (ii) stream scheduling.

Disk Scheduling

In a multi-stream environment, multiple users are requesting to retrieve N continuous multimedia data streams in a similar time period. The system will serve each of the N data streams in rounds. In other words, in any i-th round, the system will serve all the requested data streams N by reading enough data for all the requests to be consumed until the next i+1-th round. Because there are differences between the transfer rate and the consumption rate, a scheduling strategy needs to deal with the differences between the transfer rate and the consumption rate.

Let r_c and r_t be the rate of display consumption of a data stream and the rate of data transfer from disk, respectively. And let p_{max} and d_{max} be the maximum time interval in any round and the maximum time interval between two consecutive reads for any data stream, respectively. Figure 1 illustrates the ideas on



Multimedia Resource Scheduling. Figure 1. Round length and delay between reads (Fig. 3 in [9]).

round-length and delay between two consecutive disk reads. In order to have enough data to be consumed by a display, at the consumption rate of r_c, in each round of the time interval at most p_{max} (to prevent starvation until the next round), the amount of data it needs is at least $r_c \cdot p_{max}$. The d_{max} shows the possible delay, called startup delay, to start consuming a data stream, which implies the time interval it needs to wait if it misses in the current round.

Consider the disk scheduling for multiple data streams on a single disk. The round-robin algorithm retrieves data for each data stream in a fixed order in every round. Due to the fixed order, the maximum startup delay (d_{max}) is similar to p_{max}. The SCAN algorithm [13] attempts to read more disk blocks while moving the disk head over the disk, and retrieve the requested blocks when the disk header passes over them [14]. The main advantages of the SCAN algorithm are: (i) it reduces the seek time to move the disk header from one location to another in order to read next disk block, and (ii) it maximizes the throughput of disk accesses. But, because the order of serving each data stream is not fixed, the startup delay for a given data stream can be larger up to $2p_{max}$. The SCAN-EDF algorithm [13] processes the data stream requests with the earliest deadline first, using the idea discussed in EDF [12] (Earliest-Deadline-First), and processes the requests, that be shared, using the SCAN algorithm. Note that the SCAN-EDF algorithm different from the other strategies, is not designed as a round-based algorithm. The sorting-set algorithm [8,16] is designed in a way where round-robin and SCAN are treated as special cases of it. In brief, in the sorting-set algorithm, each round is further divided into several time slots. Each time slot is conceptually considered as a sorting set (or simply set). If a round is divided into m time slots, there are m time slots, and therefore there are m sorting sets, namely, set_1, set_2,..., and set_m. All the sorting sets are served in a fixed order in each run. When there is a request to retrieve a multimedia data stream, the requested data stream is assigned to a sorting set. If the entire round is treated as a single time slot, the sorting-set algorithm behaves like the SCAN algorithm. If the requested data stream is assigned to a unique sorting set, the sorting-set algorithm behaves like the round-robin algorithm.

For multiple disks accessing in parallel, there are two main categories of multiple disks scheduling strategies, namely, stripe data across the disks and replicate data across the disks. There are also different schema to retrieve the striped data: striped retrieval, split-striped retrieval, and cyclic retrieval. In the striped retrieval, entire stripes are retrieved in parallel. The split-stripe retrieval [15] retrieves some consecutive units of an entire stripe rather than the entire stripe at one time. The cyclic retrieval [2,3] is designed to retrieve units of stripes for more than one data stream. The main idea behind it is to read a small portion of data for a data stream frequently. As comparison with the other striped approaches, cyclic retrieval does not need a large buffer space. Unlike the stripe based algorithms, data streams can be replicated across disks where each disk is treated individually and independently. If a data stream is requested frequently, it can be replicated in multiple disks.

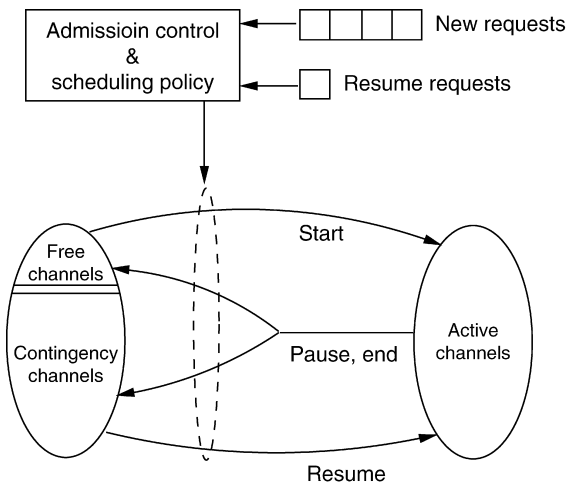
Stream Scheduling

Continuous data stream retrieval needs to be guaranteed by reserving sufficient resources. The resources are referred to as logical channels (or simply channels). Stream scheduling policies need to increase the server capacity, or in other words, to increase the number of continuous data stream requests to be served with the limited number of channels. The stream scheduling needs to consider several facts regarding the possibility of sharing. There are popular videos which are viewed by many clients most of the time during a certain time period. Several clients may view the same video but start viewing at different times in a short time interval. A client may pause and then resume when viewing a long multimedia (video) at any time. The time interval between such pause and resume is not known, which can be short or long. A client may also change to another video after the pause.

Data stream sharing serves multiple clients by a single I/O stream, which is also referred to as batching of requests [4,6]. A batching factor indicates how many clients can share a single I/O stream. In order to effectively support continuous multimedia requests, some requests need to be delayed, in order to be batched with other requests. There are three batching policies: FCFS, MQL, and FCFS-n [10]. With the FCFS (First Come First Served) policy, it queues all requests into a single queue. When there is a channel available, the FCFS policy selects the video, which is requested by the first client in the queue, to be served, in a first come first served fashion. If there are other requests in the queue that request the same video, they will be served

by sharing the I/O stream. The MQL (Maximum Queue Length) policy maintains a queue for a requested video. If there are n videos to be requested, there will be n queues. The MQL policy chooses the video with the maximum queue length to be served, when a channel becomes available. With the MQL policy, the videos with a small number of requests (short queue) may not be served. Therefore, unlike FCFS which is a fair policy, the MQL policy is seen as unfair to the videos that are not requested by many clients. The FCFS-n policy is similar to the FCFS policy, except that the n hottest videos are assigned to dedicated streams. A dedicated stream will be served in a batching window in turn, and the remaining videos, that are not assigned to a dedicated stream, are served following the FCFS policy. With the policies, the batching factor can be increased, but the amount of waiting time for a client to wait may be increased as well.

Sharing a data stream is affected by the fact that a client may pause. A new channel may need to be allocated when the client resumes. Assume that a client may resume shortly after the pause, the system may maintain some channels (contingency channels), which can improve resource utilization. The system needs to guarantee that the delay between the receipt of a resume request from a client and playback is small in order to assure client satisfaction [4,6]. Figure 2 illustrates the scheduling with contingency channels for VCR control operations (pause/resume) [6,7]. The admission control policy determines the acceptance of a new request.



Multimedia Resource Scheduling. Figure 2. Channel states under contingency policy (Fig. 2 in [5]).

The scheduling policy determines which request to be served on the available channel, in order to maximize certain performance objectives [6]. When there is a pause request from a client, the admission control and scheduling policy determines if the client is the only client viewing the multimedia stream. If it is, the channel is freed and returned to either the free channel pool or the contingency pool. A certain number of contingency channels are maintained in the system. When there is a resume request from a client, the scheduling policy checks if there is another request being served within a predetermined time window, in order to maximize the possibility of sharing. If there is another request being served but is beyond the predetermined time window, the scheduling policy tries to use another contingency channel where possible. Otherwise, it will request a free channel with higher priority when such a free channel becomes available.

The arrival rate of requests to multimedia system may vary with the time of a day [11], peek time and off-peek time. The scheduling policy also needs to address the issues of time-varying load. Note: the optimal policy at the current may not be the optimal when the load changes shortly. A two-level hierarchical scheduling policy is proposed to deal with such load fluctuations [1]. The high-level scheduler controls channels allocation, whereas the low-level scheduler controls the clients selection to be served. Three high level policies are proposed to control channel allocation rate [1]: on-demand allocation, forced-wait, and pure rate control. The on-demand allocation allocates channels to requests when there are channels available. Under this policy, the waiting time for new requests may be long, which may cause clients to change from one video to another to view at high possibility. The forced-wait policy, as the name implies, forces the first request to a data stream to wait for up to a certain time (minimum wait time). The minimum wait time is a critical parameter, and can be difficult to be selected in a dynamic environment where the load changes dynamically. The pure rate control policy allocates channels uniformly in fixed time intervals (called measurement intervals) during such a time interval only a certain number of channels are used [1].

Key Applications

In a multimedia environment, multimedia objects are retrieved from either a digital library, or a video database, or an audio database, and are delivered to a large

number of clients. The applications in such environments range from retrieving small multimedia objects (shopping, medias, education, etc.) to playback of large video objects (movie, entertainment, etc.).

Cross-references

- ► Continuous Multimedia Data Retrieval
- ► Multimedia Data Buffering
- ► Multimedia Data Storage
- ► Scheduler
- ► Scheduling Strategies Storage Resource Management

Recommended Reading

1. Almeroth K.C., Dan A., Sitaram D., and Tetzlaff W.H. Long Term Channel Allocation Strategies for Video Applications. IBM Research Report (RC 20249), 1995.
2. Berson S., Ghandeharizadeh S., Muntz R., and Ju X. Staggered striping in multimedia information systems. ACM SIGMOD Rec., 23(2):79–90, 1994.
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. Dan A., Shahabuddin P., Sitaram D., and Towsley D. Channel allocation under batching and VCR control in video-on-demand systems. J. Parallel Distrib. Comput., 30(2):168–179, 1995.
5. Dan A. and Sitaram D. Multimedia Information Storage and Management, chap. 11: Session Scheduling and Resource Sharing in Multimedia Systems. Kluwer Academic, 1996.
6. Dan A., Sitaram D., and Shahabuddin P. Scheduling policies for an on-demand video server with batching. In Proc. 2nd ACM Int. Conf. on Multimedia, 1994, pp. 15–23.
7. Dey-Sircar J.K., Salehi J.D., Kurose J.F., and Towsley D. Providing VCR capabilities in large-scale video servers. In Proc. 2nd ACM Int. Conf. on Multimedia, 1994, pp. 25–32.
8. Gemmell D.J. Multimedia network file servers: multi-channel delay sensitive data retrieval. In Proc. 1st ACM Int. Conf. on Multimedia, 1993, pp. 243–250.
9. Gemmell D.J. Multimedia Information Storage and Management, chap. 1: Disk Scheduling for Continuous Media. Kluwer Academic, 1996.
10. Ghose D. and Kim H.J. Scheduling video streams in video-on-demand systems: a survey. Multimedia Tools Appl., 11(2): 167–195, 2000.
11. Little T.D.C. and Venkatesh D. Prospects for interactive video-on-demand. IEEE Multimedia, 1(3):14–24, 1994.
12. Liu C.L. and Layland J.W. Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM, 20(1): 46–61, 1973.
13. Reddy A.L.N. and Wyllie J.C. I/O issues in a multimedia system. Computer, 27(3):69–74, 1994.
14. Teorey T.J. and Pinkerton T.B. A comparative analysis of disk scheduling policies. Commun. ACM, 15(3):177–184, 1972.
15. Tobagi F.A., Pang J., Baird R., and Gang M. Streaming RAID: a disk array management system for video files. In Proc. 1st ACM Int. Conf. on Multimedia, 1993, pp. 393–400.
16. 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.

Multimedia Retrieval Evaluation

Thijs Westerveld[1,2]
[1]Teezir Search Solutions, Ede, The Netherlands
[2]CWI, Amsterdam, The Netherlands

Definition

Multimedia Retrieval Evaluation is the activity of measuring the effectiveness of one or more multimedia search techniques. A common way of evaluating multimedia retrieval systems is by comparing them to each other in community wide benchmarks. In such benchmarks participants are invited to submit their retrieval results for a given set of topics, the relevance of the submitted items is checked, and effectiveness measures for each of the submissions are reported. Multimedia retrieval evaluation measures the effectiveness of multimedia retrieval systems or techniques by looking at how well the information need as described by a topic is satisfied by the results retrieved by the system or technique. Efficiency of the techniques is typically not taken into account, but may be studied separately.

Historical Background

Until the mid-1990s, no commonly used evaluation methodology existed for multimedia retrieval. An important reason for this is that the field has merely been a showcase for computer vision techniques. Many papers in the field 'proved' the technical merits and usefulness of their approaches to image processing by showing a few well-chosen, and well-performing examples. Since 1996, the problem of systematically evaluating multimedia retrieval techniques has gained more and more interest. In that year, the *Mira* (Multimedia Information Retrieval Applications) working group was formed [2]. The group, consisting of people from the fields of information retrieval, digital libraries, and library science studied user behavior and information needs in multimedia retrieval situations. Based on their findings, they developed performance measures. Around the same time, in the multimedia community, the discussion on proper evaluation started, and Narasimhalu et al. [8] proposed measures for evaluating content-based

information retrieval systems. These measures are based on comparing ranked lists of documents returned by a system to the perfect, or ideal, ranking. However, they do not specify how to obtain such a perfect ranking, nor do they propose a common test set. A year later, Smith [10] proposed to evaluate image retrieval using measures from the text retrieval community and in particular from *TREC*, the Text Retrieval Conference [12] for image retrieval evaluation. Again, no dataset was proposed. At the start of the twenty-first century, the evaluation problem gained more attention within the content-based image retrieval community, with the publication of three papers discussing benchmarking in visual retrieval [3,6,7]. These three papers call for a common test collection and evaluation methodology and a broader discussion on the topic. The *Benchathlon* network (http://www.benchathlon.net) was started to discuss the development of a benchmark for image retrieval. Then, in 2001, *TREC* started a video track [9] that evolved into the workshop now known as *TRECVID*.

Today, a variety of initiatives exists for evaluating the retrieval of different types of data in a variety of contexts, a list of these is provided below under data.

Foundations

Information retrieval is interactive. In web search, for example, queries are often changed or refined after an initial set of documents has been retrieved and inspected. In multimedia retrieval, where browsing is common, interactivity is perhaps even more important. Evaluation should take interactivity into account, and measure user satisfaction. The evaluation of a system as a whole in an interactive setting is often called an *operational test*. Such tests measure performance in a realistic situation. Designing such an operational test is difficult and expensive. Many users are needed to free the experiment of individual user effects, the experimental setup should not interfere with the user's natural behavior, and learning effects need to be minimized. Also, because there are many free variables, it is hard to attribute observations to particular causes. In contrast to these tests in fully operational environments, *laboratory tests* are defined as those tests in which possible sources of variability are controlled. Thus, laboratory tests can provide more specific information, even though they are further away from a realistic setting. Also, laboratory tests are cheaper to set up, because the interactive nature is ignored, and the user is removed from the loop. Laboratory tests measure the quality of the document ranking instead of user satisfaction. While some studies exist to evaluate multimedia retrieval systems in an operational setting by investigating user satisfaction, most approaches are studied in laboratory tests.

Current laboratory tests are based on the *Cranfield* paradigm [1]. In this paradigm, a test collection consists of a fixed set of documents, a fixed set of topics, and a fixed set of relevance judgements. Documents are the basic elements to retrieve, topics are descriptions of the information needs, and relevance judgements list the set of relevant documents for each topic.

The focus in laboratory tests is on comparative evaluation. Different approaches are tested, and their relative performance is measured. The process is as follows. Each approach produces a ranked lists of documents for each topic. The quality of the ranked lists is measured based on the positions of the relevant documents in the list. The results are averaged across all topics to obtain an overall quality measure.

For successful evaluation of retrieval techniques, two components are needed in addition to a test collection [4]: a measure that reflects the quality of the search and a statistical methodology for judging whether a measured difference between two techniques can be considered statistically significant. The measures used in multimedia retrieval evaluation are typically based on precision and recall, the fraction of retrieved documents that is relevant and the fraction of relevant documents that is retrieved. To measure recall, the complete set of relevant documents needs to be known. For larger collections this is impractical and a *pooling* method is used instead. With pooling, the assumption is that with a diverse set of techniques contributing runs to the evaluation, the probability that a relevant document is retrieved at a high rank by at least one of the approaches is high. A merged set of top ranked documents is assumed to contain most relevant documents and only this set of documents is manually judged for relevance. Documents that are not judged are assumed not relevant. In reality, some unjudged documents may certainly still be relevant, but it has been shown that this is of no influence to the comparative evaluation of search systems [11,13,14].

A number of aspects influence the reliability of evaluation results. First, a sufficiently large set of topics is needed. ?] suggest a minimum of 75. Second, the measures should be stable. This means it should not be

influenced too much by chance effects. Clearly, measures based on few observations are less stable than measures based on many observations. For example, precision at rank 1 (is the first retrieved document relevant) is not a very stable measure. Third, there needs to be a reasonable difference between two approaches before deciding one approach is better than the other. Sparck Jones [5] suggests a 5% difference is noticeable, and a difference greater than 10% is material. Statistical significance tests take all these aspects into account and are useful in deciding whether an observed difference between two approaches is meaningful or simply due to chance.

Key Applications

Multimedia retrieval evaluation helps to better understand what works and what does not in the area of multimedia retrieval. Many practitioners in the field benefit from the area. It gives them the opportunity to test their ideas in a principled manner and allows them to build upon approaches that are known to be successful. More and more, the papers published in renowned journals and conferences demonstrate the usefulness of their techniques by a thorough evaluation on a well-known test collection.

Data Sets

Creating a test collection for multimedia retrieval systems takes a lot of effort. Especially the generation of ground truth data for a sufficient number of topics is something that a small company or research institution cannot manage on its own. Many workshops exist that solve this problem by sharing resources in a collaborative effort to create valuable and re-usable test collections. This approach was first taken in the Text Retrieval Conferences (TREC) [12], but many others followed. Below the main collections and evaluation platforms in multimedia are listed.

The *Corel* document set is a collection of stock photographs, which is divided into subsets each relating to a specific theme (e.g., *tigers*, *sunsets*, or *English pub signs*). The collection is often used in an evaluation setting by using the classification into themes as ground truth. Given a query image, all images from the same theme –and only those– are assumed relevant. Evaluation results based on Corel are highly sensitive to the exact themes used in the evaluation [7]. In addition, Corel has a clear distinction between themes and an unusually high similarity within a theme because the photos in a theme often come from the same photographer or even the same location. This makes the collection more homogeneous than can be expected in a realistic setting.

TRECVID studies video retrieval. The data collections used have been dominated by broadcast news, but other raw and edited professional video footage is studied as well. TRECVID defines a number of tasks and provides test collections for each of them. In 2007, four main tasks existed:

Shot boundary detection: identify shot boundaries in the given video clips with their location and type (hard or soft transition).

High level feature extraction: For each of the predefined high-level features or concepts, detect the shots that contain the feature. Features that have been studied in the past include sky, road, face, vegetation, office and people marching.

Search: Given a textual description of an information need and/or one or more visual examples, find shots that satisfy this need.

Rushes summarization: Given a set of rushes, i.e., raw, unedited footage, provide a visual summary of this data that in a limited number of frames shows the key objects and events that are present in the footage.

As part of the Cross-Language Evaluation Forum (CLEF), *ImageCLEF* studies cross-language image retrieval. ImageCLEF concentrates on two main areas: retrieval of images from photographic collections and retrieval of images from medical collections.

The Initiative for the Evaluation of XML retrieval (INEX) aims to evaluate the effectiveness of XML retrieval systems. Within this initiative, the *INEX multimedia track* evaluates the retrieval of multimedia elements from a structured collection. The data collection used consists of wikipedia documents and the images contained in them. Both the retrieval of multimedia fragments (combinations of text and images) and the retrieval of images in isolation are studied.

ImagEVAL evaluates image processing technology for content-based image retrieval. The assessment focuses on features relating to what collection holders (from defence, industry and cultural sectors) expect in terms of how images may be used. The tasks include recognizing transformed images, combined textual and visual search and object detection. The collections are a heterogeneous mix of professional images including stock photography, museum archives and industrial images.

The Music Information Retrieval Evaluation Exchange (*MIREX*) evaluates subtasks of music information retrieval. The datasets used by MIREX are cd-quality audio originating from (internet) record labels that allow tracks of their artists to be published. The tasks include genre classification, melody extraction, onset detection, tempo extraction and key finding.

Cross-references

- ▶ Information Retrieval Evaluation Measures
- ▶ Multimedia Databases
- ▶ Multimedia Information Retrieval

Recommended Readings

1. Cleverdon C.W. The cranfield tests on index languagr devices. Aslib Proc., 1967, pp. 173–192.
2. Draper S.W., Dunlop M.D., Ruthven I., and van Rijsbergen C.J. (eds.). In Proc. Mira 99: Evaluating Interactive Information Retrieval. Electronic Workshops in Computing, 1999.
3. Gunther N.J. and Beretta G. A benchmark for image retrieval using distributed systems over the internet: BIRDS-I. Technical Report HPL-2000-162, HP Laboratories, 2000.
4. Hull D. Using statistical testing in the evaluation of retrieval experiments. In Proc. 16th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1993, pp. 329–338.
5. Jones K.S. Automatic indexing. J. Doc., 30:393–432, 1974.
6. Leung C.H.C. and Ho-Shing Ip H. Benchmarking for content-based visual information search. In Advances in Visual Information Systems, Fourth Int. Conf., 2000, pp. 442–456.
7. Müller H., Müller W., McG. Squire D., Marchand-Maillet S., and Pun T. Performance evaluation in content-based image retrieval: overview and proposals. Pattern Recogn. Lett. (Special Issue on Image and Video Indexing), 22(5):593–601, 2001.
8. Narasimhalu A.D., Kankanhalli M.S., and Wu J. Benchmarking multimedia databases. Multimedia Tools Appl., 4(3):333–356, 1997. ISSN 1380-7501.
9. Smeaton A.F., Over P., Costello C.J., de Vries A.P., Doermann D., Hauptmann A., Rorvig M.E., Smith J.R., and Wu L. The TREC-2001 video track: information retrieval on digital video information. In Research and Advanced Technology for Digital Libraries, Sixth European Conference, 2002, pp. 266–275.
10. Smith J.R. Image retrieval evaluation. In Proc. IEEE Workshop on Content-based Access of Image and Video Libraries, 1998, pp. 112–113.
11. Voorhees E.M. and Harman D.K. Overview of the eighth text retrieval conference (TREC-8). In Proc. The 8th Text Retrieval Conference, 2000.
12. Voorhees E.M. and Harman D.K. TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing). MIT, Cambridge, MA, 2005. ISBN 0262220733.
13. Westerveld T. Trecvid as a re-usable test-collection for video retrieval. In Proc. Multimedia Information Retrieval Workshop, 2005.
14. Zobel J. How reliable are the results of large-scale information retrieval experiments? In Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1998, pp. 307–314.

Multimodal Data

▶ Multimedia Data

Multimodal Databases

▶ Multimedia Databases

Multi-modal Information Retrieval

▶ Cross-Modal Multimedia Information Retrieval

Multimodal Interfaces

Monica Sebillo[1], Giuliana Vitiello[1], Maria De Marsico[2]
[1]University of Salerno, Salerno, Italy
[2]Sapienza University of Rome, Rome, Italy

Definition

Multimodal interfaces are characterized by the (possibly simultaneous) use of multiple human sensory modalities and can support combined input/output modes.

The term *multimodal* recurs across several domains. Since its first use in the field of interface design, its affinity and derivation from the terms "mode" and "modality" were discussed. According to Merriam-Webster, one of the meanings for *mode* is "a possible, customary, or preferred way of doing something," whereas *modality* can be "one of the main avenues of sensation (as vision)." In *multimodal interfaces*, the former influences the way information is conveyed, the latter refers to the exploited communication channel. Both express peculiar aspects of a multimodal system, which is expected to provide users with flexibility and natural interaction.

Early multimodal interfaces simply combined display, keyboard, and mouse with voice (speech

recognition/synthesis). Later, pen-based input, hand gestures, eye gaze, haptic input/output, head/body movements have been progressively used. As a result, modern multimodal interfaces aim at emulating the natural multi-sensorial forms of human-human dialogue, relying on the integration of advanced interaction modes. To support the described variety of modes, the system underlying such an interface must include hardware to acquire and render multimodal expressions and must exploit recognition-based technologies to interpret them, with response times compatible with user's interaction pace.

Historical Background

Bolt's "Put That There" demonstration system [2] can be considered as one of the earliest multimodal systems. The underlying technology allowed a joint use of voice and gesture by processing speech in parallel with manual pointing within a virtual graphical space, where users could be made easily aware of the available facility and its usage. Spoken input was semantically processed, while deictic terms in the speech were resolved by processing the spatial coordinates derived from pointing. Since then, different proposals of multimodal interfaces can be found in literature [5,4]. Many of them aim at integrating natural languages and direct manipulation in specific application domains, such as manufacturing environments and interactive maps.

The *CHI'90 Workshop on Multimedia and Multimodal Interface Design* represented a turning point [1]. The growing interest by the international community toward this research area induced the Program Committee to focus on both the integration of scientific knowledge about this discipline and the direction of future developments. Four main topics were identified for discussion, with respect to which interfaces could be designed and examined: structural principles for composition, media appropriateness, enabling technologies, and paradigms/metaphors/models/representations. Important issues emerged, from which some years later the first remarkable set of guidelines for the design of multimodal user interfaces stemmed [9].

Following the scientific attention to the multimodal paradigm, a variety of systems rapidly came out. Both hardware and software were enhanced to integrate parallel input streams, mostly acquired by speech and pen-based gestures. One of the first such prototypes was the QuickSet system developed in 1994 [3]. It was an agent-based, collaborative, multimodal-multimedia map system, allowing the user to issue commands using a combination of speech and pen input. For illustration purposes, in [8] Oviatt provides a comparison among five different speech and gesture systems, which represented the most exemplifying research-level systems developed until the late 1990s. In her evaluation, the author takes into account some characteristics, such as the type of signal fusion and the sizes of gesture and speech vocabularies, on which integration and interpretation functionalities were based.

More recently, new combinations of speech and other modalities, such as lip movements and gaze, have been exploited thanks to advanced input/output technologies, whose effective integration into flexible yet reliable interfaces requires the underlying system robustness and performance stability. As a result, multimodal interfaces have pervaded new application domains, ranging from virtual reality systems, meant to support expert users in decision making and simulation of scenarios, to training systems and to person identification/verification systems for security purposes. Such systems may be further distinguished on the basis of their input modes, which can be either intentionally exploited by users (active input mode) such as speech and gestures, or captured by the system without an explicit request by users (passive input mode), such as facial expressions.

Foundations

In order to clarify the different aspects involved in a multimodal interface, the complexity of multimodal interaction can be described in terms of the well-known *execution-evaluation cycle* defined by Don Norman in 1988 [6].

In the case of multimodal interactive cycles, Norman's model of interaction may be reformulated as follows:

1. Establishing the goal.
2. Forming the intention.
3. Specifying the multimodal action sequence in terms of human output modalities.
4. Executing the multimodal action.
5. Perceiving the system state in terms of human input modalities.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and the intentions.

- *Establishing the goal* is, as usual, the stage when the user determines what needs to be done in the given domain, in terms of a suitable task language.
- *Forming the intention*: at this stage the goal is translated into more precise user's intention, which will help the user determining the right sequence of actions that should be performed to achieve the goal
- *Specifying the multimodal action sequence.* The sequence of actions performed to accomplish the required task should be precisely stated at this stage. Here the complexity of multimodal interaction appears for the first time in the cycle. In fact, each multimodal action can be specified in terms of:
  1. Complementary human output sensory modalities (i.e., multiple utterances at once form the action) and/or
  2. Alternative human output sensory modalities (i.e., alternative, redundant utterances for the same action).

Examples of human output modalities include speaking, gesturing, gazing, touching, moving, facial expressions and some unintentional utterances such as blood pressure, temperature, heartbeat, excretion, etc. A user may, for instance, move an object in the interaction scene by speaking and pointing at (gesturing) the new object location (complementary modalities). Then, instead of gesturing (s)he may want to gaze at the new location on the interface where the object should be moved (alternative modality).

- *Executing each multimodal action.* At this stage, each human modality used to specify an action is translated into corresponding interaction modes. Thus, each action is executed through
  1. Complementary modes or
  2. Alternative modes

Text, speech, Braille, mimicking, eye/motion capture, haptics, bio-electrical sensoring are examples of modes used to translate human output modalities into the system input language.

When the execution of the whole sequence of multimodal actions is complete, the system reaches a new state and communicates it to the user again exploiting (possibly multiple) interaction modes, such as speech synthesis, display, haptic/tactile feedback, smell rendering and so on.

- *Perceiving the system state.* At this stage, the evaluation phase of the cycle begins. Depending on the combination of system output modes, the user may perceive the new state through multiple input sensory modalities, such as visual, auditory, tactile, and (in some revolutionary interfaces) even smelling and tasting.
- *Interpreting the system state.* Here the user is supposed to interpret the output of her/his sequence of actions to evaluate what has happened.
- *Evaluating the system state with respect to the goals and the intentions.* At the final stage, the user compares the new system state with her/his expectations, to evaluate if the initial goal has been effectively reached.

Of course a good mapping should be achieved between the execution and the evaluation phases in order to bridge what Norman calls the *gulf of execution* (i.e., the distance between user's specification of the action and the actions allowed by the system) and the *gulf of evaluation* (i.e., the distance between user's perception of the new state and her/his expectation). In multimodal interfaces the effective reduction of both gulfs crucially depends on the underlying interactive technology that must move as close as possible towards human-human forms of interaction and communication. Thus, on the execution side, human output modalities should be supported by suitable computer input devices, e.g., by mapping gaze, speech, touch, gesturing and smelling onto cameras, microphones, keyboard, haptic sensors and the most recent olfactory sensors, respectively. From the evaluation perspective, computer output and its perception by user should be also tightly linked. This requires the adoption of output devices, like display, audio and haptic/olfactory rendering devices, able to quickly and effectively reach the human input sensory system, so that the user sees, hears, touches and, in general, feels the new system state as it is communicated by the multimodal interface.

As an example, the successful execution of direct manipulation tasks within an immersive virtual reality environment may critically depend on the abolition of any latency time between the moment an action is performed, e.g., by means of datagloves, and the

moment the user recognizes the touch. In this way the user perceives gesturing as an act that can be directly realized at the interface. If this cannot be achieved, any usability benefits coming from the direct manipulation paradigm would be lost.

The ultimate advantage of multimodal interfaces is increased usability, in terms of both flexibility and robustness of the interaction when either redundant or complementary information is conveyed by modes. Higher flexibility is gained since multimodal interfaces can accommodate a wide range of users, tasks and environments for which each single mode may not be sufficient. Different types of information may be conveyed using the most appropriate or even less error prone modality, while alternation of different channels may prevent from fatigue in computer use intensive tasks. Redundancy of information through different communication channels is especially desirable when supporting accessibility, since users with different impairments may benefit from information and services otherwise difficult to obtain. As for the increased robustness of the interaction, the weaknesses of one modality may be offset by the strengths of another. More semantically rich input streams can support mutual disambiguation for the execution phase. As in human-human communication, the correct decoding of transmitted messages requires interpreting the mix of audio-video signals.

An important research theme in multimodal interface design is how to integrate and synchronize different modes, taking into account that synchrony of different "tracks" of interaction in different modes, does not imply their simultaneity. At present, each unimodal technique is developed separately, with noticeable advances produced by improvements in both software recognition-based techniques and hardware input/output technologies. However, as pointed out in [9], an effective integration of the involved modal technologies requires a deep understanding of the "natural" integration patterns that characterize people's combined use of different communication modes, as widely studied by psychologists and cognitive experts. The issue of integration may become even more complex when a multimodal interface is designed to support collaborative work, namely the work by multiple users who may interact through the interface using several input/output modes, either synchronously or asynchronously, and either locally or remotely.

Key Applications

Recent advances in technology have been urging IT researchers to investigate innovative multimodal interfaces and interaction paradigms, able to exploit the increased technological power. The common key goal is to reproduce in the best possible way the interaction through different channels, typical of a human-human dialogue. As an example, real "physical" manipulation might be simulated even when the latter is not possible due to logistical problems (e.g., remote operation) or to dangerous settings (e.g., radioactive materials and areas), or when it is convenient to just simulate a real operation (e.g., for training purposes). A more natural and familiar way of managing objects and situations is also expected to improve global user performances and increase applications effectiveness.

Among the most recent efforts, research on haptic equipment deserves a mention. Related advances trigger new potentialities and convey novel features towards many domains, especially industrial, medical, and biotechnological. In the industrial world, the goal of improving competitiveness has led to the experimentation of haptic interfaces in fields like automotive and aerospace engineering, and texture manufacturing. In the medical domain, education and research activities are being increasingly improved by the adoption of haptic environments for virtual surgery simulation. Several new challenges are arising in the field of Biology/Biotechnology, where the adoption of visual interfaces connected to haptic devices is recognized as a powerful and straightforward mean to handle nano-objects, such as cells, and the possibility of force feedback offered by certain haptic systems, is envisioned as a considerable improvement of operator's perception. Last but not least, several multimodal interfaces enhanced with haptic feedback have been conceived to address major societal needs, e.g., by visually impaired people or wheelchair users.

In the following, a brief list of some further application domains is presented, where multimodal interfaces are presently investigated.

Interaction in Mobile Environments

The problem that has to be solved in applications designed for mobile environments is that hands, which are the usual interaction mediator for human-computer communication with traditional input devices, must be devoted to different crucial activities, e.g., controlling a

steering wheel. In such situation, alternative modes should rather be exploited to interact with software applications such as a map browser. Moreover, user's visual attention must be focused on catching situations such as obstacles approach, so that relevant software events should be communicated for example through auditory signals, so as to relieve the user from continuously inspecting system state.

Geographic Information Systems

Multimodal interfaces are also being employed as a means to support decision makers in accessing and analyzing geospatial information in specific and critical scenarios, such as crisis management procedures and *what if* analyses. Some systems have been recently proposed, which rely on large screen displays and augmented reality tools for enhanced data visualization, as well as on collaborative advanced interfaces supporting speech and gesture recognition.

In these systems, multimodality becomes the way domain expert users can formulate appropriate requests to the underlying geographic information system and receive rapid responses, provided through different perspectives. Rapid feedback is in fact a crucial issue in situations when risk and vulnerability must be predicted as well as during exceptional events when recovery actions must be taken by users with complementary expertises.

Interaction in Adverse Settings

As discussed above, it is often necessary to substitute the human operator in adverse settings in a way that preserves both his/her health and the effectiveness of the interaction with environment objects. A much simpler case is when some communication channel might be hindered by disturbing conditions and the presence of other modes may provide possible missing information.

Multimodal Biometric Databases

Multimodal biometric databases are an example of tight integration of multiple input modes to achieve reliable person identification and verification. Fingerprints are the most well-known biometric method. More biometrics include hand conformation, iris scanning, features from face, ears or voice or handwriting. Despite noticeable progresses in biometrics research, no single bodily or behavioral trait satisfies acceptability, speed and reliability constraints of authentication in real applications. Single biometric systems are vulnerable to possible attacks, and may suffer from acquisition failures, or from the possible non-universality of the biometric feature, as in the case of deaf-mute subjects for voice recognition. The present trend is therefore towards multimodal systems, as flaws of an individual system can be compensated by the availability of a higher number of alternative biometries. Integration of single responses is a crucial point, especially when different reliability degrees can be assigned to them due to input quality or effectiveness of recognition algorithms.

Interaction in Impairment Conditions

Accessibility is a transversal issue relating to different application domains. Physical impairments call for flexible system interfaces allowing universal access to services and information. What should be affected when designing for accessibility is the structure of both input and output for each application function. Functions need parameters including both data and events triggered by user's actions. In both cases, it is necessary to adapt the format manageable by a user possibly bearing a specific disability to the one acceptable by the functions. Such adaptation could be provided by special pieces of software (wrappers). A different wrapper is needed for each different disability situation. They would be connected to suitable interfaces allowing the user to issue commands and data according to his/her ability, and translating them for function call. Information and data returned by the system undergoes a symmetrical translation. In other words, multimodal input/output should be dynamically provided. The contribution of (disabled) accessibility experts to the overall design of wrappers is essential to obtain significant results. They can suggest the best suited interaction mechanisms and the best input/output modes to use.

Future Directions

The future challenge for multimodal interfaces is the ability to better and better mimic human-like sensory perception. Such interfaces will be able to reliably interpret continuous input from more different visual, auditory, and tactile sources, chosen according to the target users' tasks. More advanced recognition of users' natural communication modalities will be supported, and more sophisticated models of multimodal interaction are expected to replace present bimodal systems. One of the problems to solve is to design and implement effective integration schemas among different modalities, based on available literature on human intersensory

perception and on natural human-human multimodal interaction patterns. More research is required on human inclination to multimodal communication with applications, depending on different target tasks, and about integration and synchronization characteristics of multimodal input/output in different contexts and situations.

Cross-references

- ▶ Geographic Information System
- ▶ Mobile and Ubiquitous Data Management
- ▶ Visual Interfaces

Recommended Reading

1. Blattner M.M. and Dannenberg R.B. CHI'90 Workshop on multimedia and multimodal interface design. SIGCHI Bull., 22(2):54–58, 1990.
2. Bolt R.A. Put that there: voice and gesture at the graphics interface. ACM Comput. Graph., 14(3):262–270, 1980.
3. Cohen P.R., Johnston M., McGee D.R., Oviatt S.L., Pittman J., Smith I., Chen L., and Clow J. QuickSet: multimodal interaction for distributed applications. In Proc. 5th ACM Int. Conf. on Multimedia, 1997, pp. 31–40.
4. European Telecommunications Standards Institute. Human Factors (HF); Multimodal interaction, communication and navigation guidelines ETSI EG 202 191 V1.1.1 (2003–08).
5. Jaimes A. and Sebe N. Multimodal human-computer interaction: a survey. Comput. Vis. Image Underst., 108:116–134, 2007.
6. Norman D. The design of everyday things. Doubleday, New York, 1988.
7. Oviatt S. Ten myths of multimodal interaction. Commun. ACM, 42(11):74–81, 1999.
8. Oviatt S. Multimodal interfaces. In The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, J. Jacko, A. Sears (eds.). Lawrence Erlbaum, NJ, 2003.
9. Reeves L.M., Lai J., Larson J.A., Oviatt S., Balaji T.S., Buisine S., Collings P., Cohen P., Kraal B., Martin J.C., McTear M., Raman T.V., Stanney K.M., Su H., and Wang Q.Y. Guidelines formultimodal user interface design. Commun. ACM, 47(1):57–59, 2004.
10. Yuen P.C., Tang Y.Y., and Wang P.S.P. (eds.). Multimodal Interface for Human-Machine Communication. World Scientific, NJ, 2002.

Multi-Pathing

Kaladhar Voruganti
Network Appliance, Sunnyvale, CA, USA

Definition

There can be multiple paths between a SCSI initiator and a SCSI target. Multiple paths between a host and a storage device are useful to provide more fault-tolerance as well as to improve system throughput. Multi-pathing software ensures that the same target volume is not seen as two separate LUNs by the host.

Key Points

The multiple paths can be configured in active-active or active-standby modes. In the active-active mode, both paths are actively transferring data. In the active-standby mode, the standby path does not actively transfer data. Some multi-pathing software allows for dynamic load balancing of traffic between the multiple paths. Typically, the storage controller vendor also provides the multi-pathing driver that runs on the host, and this software is usually limited to only operating with the vendor's storage devices. Software vendors are beginning to provide multi-pathing software that can interoperate with storage controllers from multiple storage vendors.

Cross-references

- ▶ Initiator
- ▶ LUN
- ▶ Target
- ▶ Volume

Multiple Classifier System

- ▶ Ensemble

Multiple Imputation

- ▶ Synthetic Microdata

Multiple Linked Plots

- ▶ Dynamic Graphics

Multiple Query Optimization

- ▶ Multi-Query Optimization

Multiple Representation Modeling

Christine Parent[1], Stefano Spaccapietra[2], Christelle Vangenot[2], Esteban Zimányi[3]
[1]University of Lausanne, Lausanne, Switzerland
[2]EPFL, Lausanne, Switzerland
[3]Free University of Brussels, Brussels, Belgium

Synonyms

Multi-scale; Multi-resolution; Multi-granularity modeling

Definition

Geodata management systems (i.e., GIS and DBMS) are said to support *multiple representations* if they have the capability to record and manage multiple representations of the same real-world phenomena. For example, the same building may have two representations, one with administrative data (e.g., owner and address) and a geometry of type point, and the other one with technical information (e.g., material and height) and a geometry of type surface. Multirepresentation is essential to make a data repository suitable for use by various applications that focus on the same real world of interest, while each application has a specific perception matching its goals. Different perceptions translate into different requirements determining what information is kept and how it is structured, characterized, and valued. A typically used case is map agencies that edit a series of national maps at various scales and on various themes.

Factors that concur in generating different representations include the intended use of data and the level of detail matching the applications concerns. The former rules the choice of data structures (which objects, relationships, and attributes are relevant) and of value domains (e.g., whether the temperatures are stored in Celsius or Fahrenheit). The latter rules data resolution from coarse to precise and impacts both the semantic and the spatial representation.

Multiple representation modeling is the activity of designing a data repository that consistently holds multiple representations for various perceptions of a given set of phenomena. It relies on a multirepresentation data model, i.e., a data model with constructs and rules to define and differentiate the various perceptions and for each perception the representations of the phenomena of the real world of interest.

Historical Background

Support for different requirements over the same data set has first been provided by using multiple data files, each one designed for a specific application. Aiming at consistency, databases looked instead for ways to gather data into a single repository, generating the need for multiple representations. First came facilities to define application-specific subschemas, as simple restrictions of the database schema. Later, more flexibility was achieved through the view mechanism. In object-oriented terms, views are virtual representations derived from existing data. They create either an alternative representation for existing objects (object preserving views) or new objects composed from existing objects (object generating views). Each view defines a single new representation. However, views do not provide multiple perceptions, i.e., there is no possibility to identify the collection of views that forms a consistent whole for an application.

Similarly, the concept of is-a link was borrowed from artificial intelligence to provide for various representations of the same object in a classification refinement hierarchy. However, the concept comes with a population inclusion constraint: The subtype population is included in the supertype population. This cannot cope with situations where the populations of two object types only overlap, i.e., the two populations may have specific objects not represented in the other population. Since then, the situation has not changed much. It is only in the last decade that the need for more flexible multirepresentation and explicit support of various perceptions has been stressed by researchers.

Multirepresentation research in spatial databases can be traced back to the 1989 NCGIA program. In the early 1980's, maps began to be stored in geographic databases, and that opened up many new research tracks. The specification and implementation of cartographic generalization was one of them. It relied on the idea that cartographic generalization would allow the automatic derivation of maps at different scales from a single geographic database. Realizing that this full automation was not possible increased the focus on multiple representation databases [7,9,15]. Under the pressure of delivering maps at different scales, the first researches were, in the early 1990's, focusing on multi-scale data structures (i.e., data structures allowing to retrieve geometry of real-world objects for any given scale) and

multiscale databases (i.e., databases in which the data for maps at different scales is stored and linked together). Later, the multiscale approach was refined and extended into the multirepresentation approach: scale indeed is not a concept relevant for databases and there is nowadays more to a geographical database than producing maps. The current scope of multirepresentation for GIS database comprises various techniques that can be used, within a single database, to automatically derive a coarser representation from another representation at finer resolution. These techniques aim at computing objects at coarser representations, for multiresolution analyses rather than for display. They include generic cartographic generalization operations (not driven by map display considerations) as well as operations performed on specific object types (e.g., selection of instances based on an ad-hoc predicate, aggregation of instances to create new objects). Some authors use the term model generalization to denote that the use of various techniques leads to the creation of a set of virtual databases for different resolution levels, and the mappings between their schemas (models in GIS terms). The automatic derivation rules (the mappings) allow update propagation from finer to coarser representations.

Currently, only a few simple multi-representation databases exist and are used to their expected potential.

Foundations

Early research on multi-representation in GIS was driven by cartographers' requirements. This explains why the ability to draw maps at different scales has been for long the targeted objective, popularizing the concept of *multi-scale databases*. However, many other spatial applications that need to perform spatial data analysis require storing and managing specific representations where objects may have various geometries (derived one from another or not) and also show varying thematic characteristics (e.g., have different attributes and different relationships). Thus, the research domain evolved from multi-scale databases to *multi-representation databases*. Equally important is the capability to provide each application with a consistent set of data that corresponds to its own perception of the real world of interest, in short its own database. Therefore, support of multirepresentation should be complemented with support of multiperception.

Multiscale Databases

Multiresolution databases are still referred to by the GIS community as multiscale databases, despite the fact that scale does not apply to data storing. Scale is a concept related to the drawing of maps on paper or on screen. It is the ratio between measures on a map and the corresponding measures in the real world. Scale only characterizes an intended use of data. Instead, the level of resolution of a spatial database determines what geometries are stored. It defines a threshold such that only geometries beyond the threshold are captured and stored.

In early work by Timpf, the different map representations of the same real-world entities are interconnected using a directed acyclic graph data structure. The graph allows users to navigate among maps at different scales by zoom-in and zoom-out operations. This work later developed into a more general Map Cube Model [14], supported by a theory on the structure of series of maps of the same region at different scales. Each map is described as a composition of four components: a set of lines representing transportation and hydrology networks, the set of areas, called containers, created by the lines of the trans-hydro network, areas that are a refinement of the container partition, and objects contained in these areas. The elements stored in each of the four components (e.g., streets, land-use areas, buildings) are then organized into aggregation and/or generalization hierarchies. This forms a graph for each component, where each level of the graph corresponds to a given scale.

Stell and Worboys have also proposed a solution to link a series of maps [12]. Their database organization is called a stratified map space. Each map gathers objects of a particular region that share the same semantic and spatial granularity. Maps are grouped by map spaces, i.e., sets of maps at the same granularity, describing various regions. The stratified map space is the set of all maps spaces organized according to a hierarchy based on different granularity levels. Transformation functions allow users to navigate in a stratified map space and propagate updates.

Multi-Representation Databases

Work on spatial multi-representation databases has followed two main tracks: either proposing new (conceptual) data models that include explicit description

of multi-representation, or proposing frameworks that organize a set of existing classic (i.e., without multi-representation) databases into a global multirepresentation repository.

Database Models for Multiple Representations Several data models with specific concepts for multiple-representation modeling have been proposed. They range from simple solutions allowing users to associate various geometries to the same objects to more sophisticated solutions. They are discussed here according to the requirements for multiple-representation modeling.

A model for multirepresentation should allow one to characterize the same objects using different sets of attributes, and attributes with different values and different domains. This flexibility is supported by the MADS model [8], where multiple representations of a given phenomenon may be organized according to two strategies. In the first one, the various spatial and semantic descriptions of the same real-world phenomenon are merged into a single database construct. Each element of a description is qualified by a tag (called stamp) whose value identifies the perceptions for which it is relevant. Object and relationship types can thus have various sets of attributes depending on the perception. Attributes may bear various cardinalities or value domains according to the perception stamp; they can also contain a value that is a function of the perception stamp.

The Vuel approach [2] also offers the possibility of associating various semantic and spatial descriptions to the same real-world entities. In addition, various graphical representations, useful for drawing maps at different scales, can be defined. The data model is a snowflake model for spatial data warehousing. The fact table is composed of a specific kind of tuples, called vuels. A vuel fact is a particular representation of a real-world entity. It has three components: a geometry, a graphical description, and a semantic description. The vuel representation may vary according to these three dimensions. Moreover, the semantic dimension is a fact table itself with four dimensions: the class, the attribute, the domain of value, and the value dimensions. This allows the creation of various semantic descriptions by combining the dimensions (different classes, different sets of attributes, attributes with various domains and various values). OMT-G [3], a UML-based model, supports the modeling of multiple representations of data through a specific kind of relationship called conceptual generalization relationship. This relationship allows the definition of various views of the same real-world entities as subclasses of a shared super-class. The superclass describes the thematic attributes that are common to all the representations and it has no spatial representation. Each subclass describes its own view by specifying its own thematic and spatial attributes. The subclasses inherit the common attributes from the superclass. A presentation diagram shows graphical representations that may be associated to a class and the operations to obtain them. MRSL [5], another UML-based model, supports multi-representation through the introduction of two concepts: representation objects (r-objects) and integration objects (i-objects). All r-objects corresponding to the same real-word phenomenon are linked by a monovalued or multivalued link to a single i-object, whose role is to ensure consistency among them. Each r-object specifies a specific set of attributes and values for the same real-world object.

Multiple representation modeling is not limited to associating multiple sets of attributes or values to one object. In particular, when changing the level of detail, objects may disappear, whereas others may be grouped. Thus, in addition, there is the need to put into correspondence one object with several objects or two different sets of objects.

In the second strategy of the MADS approach, the various descriptions of the same phenomena belong to separate object types. They can be linked by inter-representation links that are either traditional associations or multi-associations. Multi-associations are binary relationships that, contrarily to association relationships, do not link two objects but two groups of objects. A multi-association is needed whenever the real-world entities are not represented per se, but through two different decompositions; e.g., a decomposition of a road in segments according to the number of lanes, and one according to crossroads. The other modeling approaches only support correspondence links of kind association, and the supported cardinality of the link varies: in MRSL, a i-object can be linked to r-objects through 1:1 and 1:N links thus providing support for the 1:N and N:M correspondences. In Vuel, corresponding objects can also be linked through 1:1 or 1:N inter-representation links. However, there is no support for N:M inter-representation links. OMT-G does not support inter-representation associations between objects.

Not only do objects need multiple representations, but relationships also do. This is only supported by

MADS. In MADS, all characteristics of a relationship may have various representations: its semantics (e.g., topological, aggregation, or plain), its roles, and its cardinalities. For instance, a relationship type can be a topological adjacency relationship in one description and a near relationship in another one with a more precise resolution.

In the spatial context, as data from one representation may often result from the derivation of the same data represented at another resolution, the representations of the same real-world entity are not independent and one may expect to be able to state constraints between these representations. Consistency constraints in databases are maintained through the definition of integrity constraints. Some constraints, such as cardinalities, are embedded in the concepts of the model – in particular some constraints are inherent to the multiple-representation concepts – while other constraints need to be defined in the application. MRSL is the only model proposing specific multirepresentation constraints: three kinds of rules can be associated to an i-object and its linked r-objects: consistency rules, which can be object or value correspondences, matching rules, and restoration rules. Matching rules specify how to match objects representing the same entity. They can be attribute comparison, spatial match operations, or global identifiers. Restoration rules are used to restore consistency between an i-object and its r-objects when needed.

Finally, considering that a multirepresentation database contains several representations of the same real-world phenomena, it is important to associate metadata to the representations to identify the application(s) they are relevant for, but also in order to know which representations together form a consistent whole for the application. This important requirement is fulfilled by MADS through the concept of perception stamp. In MADS, a perception stamp is a vector of values (e.g., a viewpoint and a resolution) that identifies a particular perception, and all elements of the database (types, properties, instances) are stamped for defining for which perception they are relevant. In Vuel, the designer can define views that are compositions of vuels. Each view defines a particular perception, thus providing a functionality similar to perception stamps.

Architectures for Distributed Representations Instead of proposing new concepts allowing users to integrate multiple representations of the same real-world phenomena into a unique multirepresentation database, other proposals followed a less intrusive approach. Capitalizing on the fact that there already exist many spatial databases, these approaches create a multirepresentation framework out of a set of existing classic (i.e., describing a unique perception and resolution) databases. There are two main kinds of proposals: the first one focuses on the *definition of links between objects* in corresponding databases, the second one aims at building *federated database management systems*.

In the first category, the work of Kilpelainen [6] was one of the first proposals tackling multiple representations from a database point of view. It supports bidirectional links that allow one to propagate updates in both directions and perform reasoning processes in the form of generalization operators.

In federated spatial databases, users access a set of databases through a single integrated schema, which describes virtual multirepresentation objects. During query processing, multirepresentation objects are dynamically constructed by merging all the corresponding monorepresentation objects that exist in the various databases. There have been several proposals for spatial database integration [4]. Particularly interesting are those that build the integrated schema using multirepresentation concepts, e.g., [5], based on MRSL, and [11], based on MADS. Using MRSL, each r-object in the integrated schema holds an UML tag that identifies the corresponding source database. Using MADS, perception stamps can fulfill the same functionality.

Key Applications

Cartography

As they cannot automatically derive maps at different scales from a single detailed database, national map agencies have to create several databases, one per scale. For them, multirepresentation modeling is crucial for two main reasons:

1. To propagate updates [1]: The cost of updating can be lowered by entering updates only once in a database and propagating them, at least semi-automatically, to the other databases.
2. To enforce consistency [10]: Multi-representation databases play an important role in order to

enforce consistency between the same data described at different levels of details. In addition, integrating existing databases to create a multi-representation database allows one to detect inconsistencies between the databases.

Multi-Scale Analysis

Multirepresentation databases can benefit many applications that need to analyze data at different levels of details or defined for different viewpoints. For example, a fire monitoring application may need very detailed data on current fires (to direct the action of fire brigades as precisely as possible), only need medium-level resolution data for records of past fires, and use low-level resolution data for generic organization of fire management activities.

Other candidate applications are those relying on spatial data warehouses, using spatial OLAP and spatial data cubes to perform multi-dimensional analysis. An example is traffic accident monitoring applications, e.g., for analysis of the number of deadly accidents according to multilevel criteria (by road, region, department, or state). Multirepresentation storage of spatial data is needed in order to drill-up and drill-down the cube [2].

Future Directions

Work in progress explores the use of multirepresentation capabilities in support of modularization of knowledge repositories. In particular, the semantic web community is developing various approaches to turn huge ontologies that are being built in several knowledge domains into smaller sets of more manageable ontological modules. Existing approaches follow both the integrated direction (a single ontology is modularized) and the distributed direction (various existing ontologies are interconnected within a global knowledge sharing system). A forthcoming book on Ontology Modularization [13] is due for publication in 2008.

Cross-references

- ▶ Database Design
- ▶ Distributed Spatial Databases
- ▶ Field-Based Spatial Modeling
- ▶ Geographic Information System
- ▶ Multidimensional Modeling
- ▶ Semantic Modeling for Geographic Information Systems
- ▶ Spatial and Spatio-Temporal Data Models and Languages
- ▶ Spatial Data Types
- ▶ Topological Data Models
- ▶ Topological Relationships

Recommended Reading

1. Badard T. and Lemarié C. Propagating updates between geographic databases with different scales, chapter 10. In Innovations in GIS 7: GIS and GeoComputation, P. Atkinson, D. Martin (eds.). Taylor and Francis, London, UK, 2000, pp. 135–146.
2. Bédard Y. and Bernier E. Supporting multiple representations with spatial view management and the concept of VUEL. In Proc. Joint Workshop on Multi-Scale Representations of Spatial Data, 2002.
3. Borges K., Davis C.A., and Laender A. OMT-G: an object-oriented data model for geographic applications. GeoInformatica, 5(3):221–260, 2001.
4. Devogele T., Parent C., and Spaccapietra S. On spatial database integration. Int. J. Geogr. Inf. Syst., 12(4):335–352, 1998.
5. Friis-Christensen A., Jensen C.S., Nytun J.P., and Skogan D. A conceptual schema language for the management of multiple representations of geographic entities. Trans. GIS, 9(3):345–380, 2005.
6. Kilpelaïnen T. Maintenance of topographic data by multiple representations. In Proc. Annual Conference and Exposition of GIS/LIS, 1998, pp. 342–351.
7. Mustière S. and Van Smaalen J. Database requirements for generalisation and multiple representations. In Generalisation of Geographical Information: Cartographic Modelling and Applications, W.A. Mackaness, A. Ruas, T. Sarjakoski (eds.). Elsevier, Amsterdam, 2007.
8. Parent C., Spaccapietra S., and Zimányi E. Conceptual Modeling for Traditional and Spatio-temporal Applications. The MADS Approach. Springer, Berlin, 2006.
9. Sarjakoski L.T. Conceptual models of generalisation and multiple representation. In Generalisation of Geographical Information: Cartographic Modelling and Applications, W.A. Mackaness, A. Ruas, T. Sarjakoski (eds.). Elsevier, Amsterdam, 2007, pp. 11–36.
10. Sheeren D., Mustière S., and Zucker J.D. How to integrate heterogeneous spatial databases in a consistent way? In Proc. 8th East-European Conf. Advances in Databases and Information Systems, 2004, pp. 364–378.
11. Sotnykova A., Vangenot C., Cullot N., Bennacer N., and Aufaure M.-A. Semantic mappings in description logics for spatio-temporal database schema integration. Journal on Data Semantics III:143–167, 2005.
12. Stell J.G. and Worboys M.F. Stratified map spaces: a formal basis for multi-resolution spatial databases. In Proc. 8th Int. Symp. on Spatial Data Handling, 1998, pp. 180–189.
13. Stuckenschmidt H., Parent C., and Spaccapietra S. (Eds.). Modular Ontologies. Springer LNCS, 2009.
14. Timpf S. Map cube model: a model for multi-scale data. In Proc. 8th Int. Symp. on Spatial Data Handling, 1998, pp. 190–201.

15. Weibel R. and Dutton G. Generalizing spatial data and dealing with multiple representations. In Geographical Information Systems: Principles, Techniques, Management and Applications, vol. 1, 2nd edn., P. Longley, M.F. Goodchild, D.J. Maguire, D.W. Rhind (eds.). Wiley, 1999, pp. 125–155.

Multiplicity

► Statistical Disclosure Limitation For Data Access

Multiprocessor Data Placement

► Parallel Data Placement

Multiprocessor Database Management

► Parallel Database Management

Multiprocessor Query Processing

► Parallel Query Processing

Multi-Query Optimization

Prasan Roy[1], S. Sudarshan[2]
[1]Aster Data Systems, Inc., Redwood City, CA, USA
[2]Indian Institute of Technology, Bombay, India

Synonyms

Multiple query optimization; Global query optimization; Common subexpression elimination; Optimization of DAG-structured query evaluation plans

Definition

Multi-query optimization is the task of generating an optimal combined evaluation plan for a collection of multiple queries. Unlike traditional single-query optimization, multi-query optimization can exploit commonalities between queries, for example by computing common sub-expressions (i.e., subexpressions that are shared by multiple queries) once and reusing them, or by sharing scans of relations from disk.

Historical Background

Early work on multi-query optimization includes work by Sellis [11], Park and Segev [7] and Rosenthal and Chakravarthy [9]. Shim et al. [12] consider heuristics to reduce the cost of multi-query optimization. However, even with heuristics, these approaches are extremely expensive for situations where each query may have a large number of alternative evaluation plans.

Subramanian and Venkataraman [13] consider sharing only among the best plans of the query; this approach can be implemented as an efficient, post-optimization phase in existing systems, but does not guarantee optimality. In fact, Roy et al. [10] show that it can be significantly suboptimal. Rao and Ross [8] address the problem of sharing common computation across multiple invocations of a subquery, which is a special case of multi-query optimization,

Roy et al. [10] address the problem of extending top-down cost-based query optimizers to support multi-query optimization, and present greedy heuristics, as well as implementation optimizations. Their techniques were shown to be practical and to give good results. Dalvi et al. [1] explores the possibility of sharing intermediate results by pipelining, avoiding unnecessary materializations. Diwan et al. [2] consider issues of scheduling and caching in multi-query optimization. Zhou et al. [14] discuss the implementation of multi-query optimization on a commercial query optimizer.

In addition to the motivation of optimizing a collection (batch) of queries, multi-query optimization has also been applied to other settings. For example, Mistry et al. [6] consider the issue of multi-query optimization in the context of view maintenance, while Fan et al. [3] point out the importance of multi-query processing in optimizing XPath queries.

Foundations

Multi-query optimization is more expensive than independent optimization of multiple queries, since a globally optimal plan may involve subplans that are sub-optimal for the individual queries.

Consider a batch consisting of two queries $(A \bowtie B \bowtie C)$ and $(B \bowtie C \bowtie D)$. A traditional system would

evaluate each of these queries independently, using the individual best plans suggested by the query optimizer for each of these queries. Let these best plans be as shown in Fig. 1a. Suppose the base relations A, B, C and D each have a scan cost of 10 units (the actual unit of measure is not relevant to this example). Each of the joins have a cost of 100 units, giving a total evaluation cost of 460 units. On the other hand, in the plan shown in Fig. 1b, the common subexpression $(B \bowtie C)$ is first computed and materialized on the disk at a cost of 10. Then, it is scanned twice – the first time to join with A in order to compute $(A \bowtie B \bowtie C)$, and the second time to join it with D in order to compute $(B \bowtie C \bowtie D)$ – at a cost of 10 per scan. Each of these joins have a cost of 100 units. The total cost of this *consolidated* plan is thus 370 units, which is about 20% less than the cost of the traditional plan of Fig. 1a. Although the benefit here is small, it could be significantly more for batches containing more queries.

The expression $(B \bowtie C)$ that is common between the two queries $(A \bowtie B \bowtie C)$ and $(B \bowtie C \bowtie D)$ in the above example is a *common subexpression* (CSE). A relation used in multiple queries can be thought of as a special case of a common subexpression. Although there is no need to compute and store it, a scan of the relation from disk can be shared by multiple queries.

A plan for a single complex query can have common subexpressions within itself. Traditional optimizers ignore the possibility of exploiting such a common subexpression, but some of the techniques for multi-query optimization, such as [10] can exploit such common subexpressions.

Challenges

The job of a multi-query optimizer can be broken into two parts: (i) recognize possibilities of shared computation by identifying CSEs, and (ii) find a globally optimal evaluation plan exploiting the CSEs identified.

Identifying CSEs Each query can have a large number of alternative evaluation plans. Given a particular evaluation plan for each of a set of queries, it is straightforward to find common subexpressions amongst these plans. However, since the number of possible combinations of such plans is very large, enumerating them is not feasible.

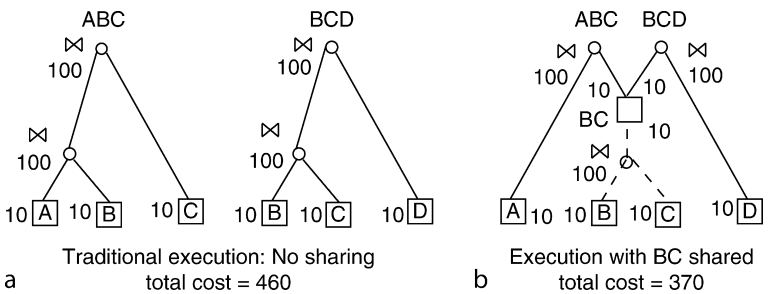
Subexpressions that could be shared between some plans for two or more queries can however be identified without enumerating all possible plan combinations. The number of such potentially common subexpressions is still very large, but smaller than the number of plan combinations.

Finding the Optimal Plan in Presence of CSEs Traditional query optimizers use dynamic programming algorithms to find the best plan for an input query. These dynamic programming algorithms are applicable because, in the absence of sharing of common subexpressions, each subplan of the overall best plan is also the best plan for the subexpression it computes.

In the presence of sharing, such a property does not hold – as shown in the example above, a globally optimal plan can consist of subplans that are not globally optimal – and therefore a straightforward dynamic programming approach does not work. The problem of finding an optimal combined plan in presence of CSEs is therefore a strictly harder problem than traditional query optimization.

Engineering an Efficient Multi-Query Optimizer

As mentioned earlier, a simple minded approach that iterates over all possible plans for each query and analyzes each combination of plans is very expensive, and infeasible for non-trivial queries. And conversely, a heuristic that only considers the individual best plan for each query does not work well, as mentioned earlier.



Multi-Query Optimization. Figure 1. Example illustrating benefits of sharing computation.

A more practical approach was presented in [10]. This approach efficiently finds the set of potentially common subexpressions for a set of queries, and then identifies the subset of CSEs to share, and the best resulting consolidated plan, using an iterative greedy heuristic.

Instead of enumerating the search space of possible plan combinations, the idea is to store all the plans across all the queries in a single compact data structure called the Logical Query DAG (LQDAG). The LQDAG is a refinement of the "memo" data structure used in transformational top-down optimizers, such as Volcano [4], to memorize the best plans of the intermediate results. (Such memorization, as done in top-down query optimizers such as Volcano, is equivalent to dynamic programming, as used in System R and other bottom-up query optimizers.)

Figure 2a shows a LQDAG for the query A ⋈ B ⋈ C; this LQDAG represents the three alternative plans for the query: (A ⋈ B) ⋈ C, A ⋈ (B ⋈ C) and B ⋈ (A ⋈ C). Each square node (equivalence node) in the LQDAG represents a distinct intermediate result, and each round node (operation node) below represents a distinct plan to compute the same from the underlying intermediate results. In general, a LQDAG can represent multiple queries in a consolidated manner, with a distinct root node for each distinct query. Figure 2b shows a consolidated LQDAG for the two example queries seen earlier, A ⋈ B ⋈ C, and B ⋈ C ⋈ D.

The CSEs for the given queries correspond to equivalence nodes in the LQDAG that are shared either within the same plan, or between plans for two distinct queries; [10] presents an efficient algorithm that identifies the set of all CSEs in a single bottom-up traversal of the LQDAG.

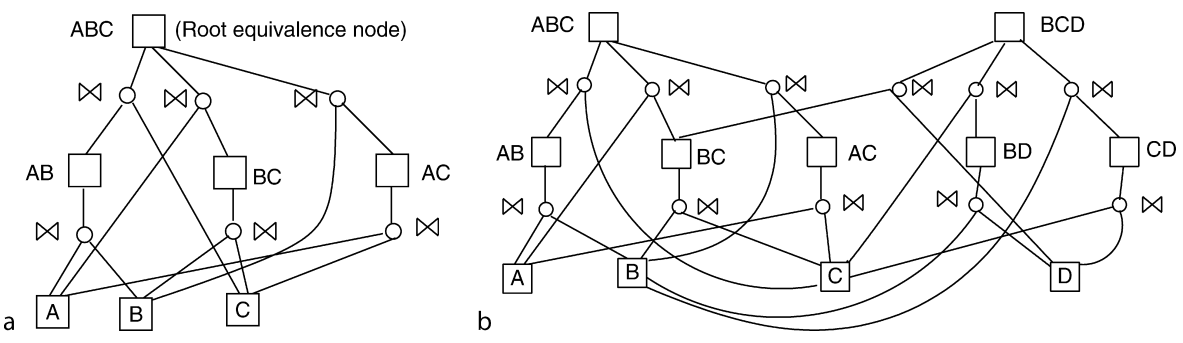
After the CSEs are identified, the next task is to find the best consolidated plan for the queries exploiting these CSEs. When the number of CSEs is large, an exhaustive search is not feasible; a natural approach is then to use a greedy heuristic that iteratively picks the CSE with the greatest benefit (i.e., whose use would result in the greatest decrease in the overall evaluation cost), terminating when no further decrease is possible. This algorithm requires that the benefit of each candidate CSE be recomputed in each iteration – this involves finding the best plan that uses the candidate CSE, in addition to the CSEs selected in earlier iterations.

With multiple such optimization calls in each iteration, a naive implementation of the greedy heuristic would be too expensive to be practical. [10] shows how to make this approach practical by (i) incorporating additional heuristics to significantly reduce the number of benefit computations, and (ii) showing how to efficiently perform a benefit computation by exploiting the LQDAG representation of the plan space. Additional insights on the task of seamlessly incorporating multi-query optimization into the Microsoft SQL-Server query optimizer are presented by Zhou et al. [14].

The above approach assumes that CSEs are materialized and read back from disk when required. Dalvi et al. [1] shows how to schedule queries such that results can be pipelined to multiple uses, even with a limited buffer space, thereby minimizing IO. Diwan et al. [2] addresses the issue of caching results in limited memory, and scheduling queries to minimize cache usage.

Key Applications

The idea of sharing computation among different queries to save on time and resources is ubiquitous.



Multi-Query Optimization. Figure 2. (a) LQDAG for A ⋈ B ⋈ C, and (b) Combined LQDAG for A ⋈ B ⋈ C and B ⋈ C ⋈ D.

As queries become increasingly expensive, the need for multi-query optimization to enable such savings is likely to increase as well. A few representative applications which motivate multi-query optimization are listed below.

- ***On-Line Analytic Processing (OLAP) and Reporting***: A typical OLAP and reporting workload consists of queries with a significant amount of overlap. This overlap can occur for several reasons. For instance, queries might overlap in the kind of analysis they perform, or in the subset of data they are interested in; different queries could compute different aggregates over a join of the same set of tables. Alternatively, the queries could be against a virtual view; these queries clearly overlap at least in the computation of the result of the virtual view. Or else, the queries could involve common table expressions (specified using the WITH clause) that could be used in multiple places in the query; parts or whole of such common table expressions could be transiently materialized and reused [14]. Finally, the queries could involve correlated nested subqueries – invariant parts of these nested subqueries could be computed once and shared across invocations of the subquery [8].
- ***Materialized View Maintenance***: Materialized views are supported by most major database systems today. Such materialized views must be updated when the underlying relations are updated. The maintenance plans for different views often share common computation. Mistry et al. [6] show how to exploit multi-query optimization to create an optimal view maintenance plan.
- ***XML Query Processing***: In systems that store XML data in relational databases, the XPATH queries containing regular path expressions translate into a sequence of queries with significant overlap. Such queries are likely to benefit significantly from multi-query optimization [3].
- ***Stream Query Processing***: Monitoring applications such as financial analysis and network intrusion detection often have to process multiple queries over a common stream of data. Such queries are likely to overlap significantly in the expressions they compute, and are likely to gain from multi-query optimization [5].

Cross-references

- ▶ Cost-based Query Optimization
- ▶ Query Optimization
- ▶ Transformational Query Optimization

Recommended Reading

1. Dalvi N.N., Sanghai S.K., Roy P., and Sudarshan S. Pipelining in multi-query optimization. J. Comput. Syst. Sci., 66(4):728–762, 2003.
2. Diwan A.A., Sudarshan S., and Thomas D. Scheduling and Caching in Multi-Query Optimization. In Proc. 13th Int. Conf. Management of Data, 2006.
3. Fan W., Yu J.X., Lu H., Lu J., and Rastogi R. Query translation from XPATH to SQL in the presence of recursive DTDs. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 337–348.
4. Graefe G. and McKenna W.J. The Volcano Optimizer Generator: Extensibility and Efficient Search. In Proc. 9th Int. Conf. on Data Engineering, 1993, pp. 209–218.
5. Krishnamurthy S., Wu C., and Franklin M. On-the-fly sharing for streamed aggregation. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2006, pp. 623–634.
6. Mistry H., Roy P., Sudarshan S., and Ramamritham K. Materialized view selection and maintenance using multi-query optimization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2001, pp. 307–318.
7. Park J. and Segev A. Using common subexpressions to optimize multiple queries. In Proc. 4th Int. Conf. on Data Engineering, 1988, pp. 311–319.
8. Rao J. and Ross K.A. Reusing invariants: a new strategy for correlated queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 37–48.
9. Rosenthal A. and Chakravarthy U.S. Anatomy of a modular multiple query optimizer. In Proc. 14th Int. Conf. on Very Large Data Bases, 1988, pp. 230–239.
10. Roy P., Seshadri S., Sudarshan S., and Bhobe S. Efficient and extensible algorithms for multi query optimization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 249–260.
11. Sellis T.K. Multiple query optimization. ACM Trans. Database Syst., 13(1):23–52, 1988.
12. Shim K., Sellis T., and Nau D. Improvements on a heuristic algorithm for multiple-query optimization. Data Knowl. Eng., 12:197–222, 1994.
13. Subramanian S.N. and Venkataraman S. Cost-based optimization of decision support queries using transient views. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 319–330.
14. Zhou J., Larson P.Å., Freytag J.C., and Lehner W. Efficient exploitation of similar subexpressions for query processing. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2007, pp. 533–544.

Multi-Resolution

- ▶ Multiple Representation Modeling

Multi-Resolution Terrain Modeling

Enrico Puppo
University of Genova, Genova, Italy

Synonyms

Level-of-detail (LOD) terrain modeling

Definition

Multi-resolution terrain models provide the capability of using different representations of terrain at different levels of accuracy and complexity, depending on specific application needs. The major motivation behind multi-resolution is improving performance in geometry processing and visualization. Given a terrain database, a multi-resolution model provides the mechanisms to answer queries that combine both spatial and resolution criteria. In the simplest case, one could ask for a representation of terrain on a given area and with a given accuracy in elevation. More sophisticated multi-resolution models support adaptive queries, also known as *selective refinement* queries, where resolution may vary smoothly on the extracted representation, according to some given criterion. For instance, one could ask for an accuracy of at least 10 m on a given range of elevations, and smoothly degrading to say 100 m out of that range; similarly, high resolution could be focused in the proximity of a lineal feature (e.g., a road, a river); in view-dependent visualization, it is useful to have maximal resolution close to the viewpoint, and degrade it smoothly according to distance from it; etc. Multi-resolution engines must be able to answer such queries in real time even on planetary size databases. For instance, in view-dependent visualization it may be necessary to change representation, hence answering a query, at each frame, i.e., 25–30 times per second.

Historical Background

The concept of multi-resolution has been known since the mid 1970s, with seminal work by J. Clark [3]. Since then, many different proposals appeared in the literature, both in the context of terrain modeling and, more generally, in CAD and computer graphics (see [13]).

The design of multi-resolution terrain models is inter-related with *terrain generalization*, i.e., the problem of taking a representation of a terrain and generating another, smaller representation of the same terrain at a lower accuracy. The ideal aim of terrain generalization is to achieve an optimal ratio between accuracy and size of representation. This problem has been shown to be NP-hard by Agarwal and Suri [1]. However, starting with seminal work by Fowler and Little in the late 1970s [8], many algorithms for terrain generalization have been proposed in the literature that achieve good results in practice.

Early multi-resolution models belonged to two general classes: discrete models and tree-like models. In a discrete model, a collection of alternative representations of the same terrain at different resolutions is stored. Tree-like models follow a hierarchical approach: a base model provides a coarse representation of terrain made of a small number of atomic cells; each such cell is refined by decomposition into smaller cells at the next level of resolution; refinement is repeated over several levels and the model in maintained in a tree-like data structure. A notable example is given by *restricted quadtrees*, introduced first by Von Herzen and Barr in 1987 [15] and widely developed later by several authors. This class of models is suited to efficiently manage data with a regular distribution.

Many models developed (starting in the mid 1990s) are based on Triangulated Irregular Networks (TINs). Such models support also the manipulation of irregularly distributed data and may all be seen as instances of a general framework introduced by Puppo in 1996 [14]. The basic elements of such a framework are local modification operations, which change the resolution of a representation locally, and the hierarchical organization of such modifications on a directed acyclic graph. Such models come in many variants, they all support selective refinement and may achieve the best ratios between level of accuracy and number of triangles used in a representation.

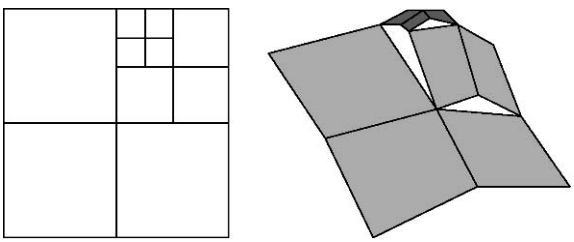
Foundations

The simplest way to perform terrain generalization and implicitly obtain a multi-resolution model from a database of regularly distributed data is based on sub-sampling. Given a grid of elevation data at high resolution, a coarser sub-grid is obtained by regularly sampling data along each axis with a fixed step. In this case, the term *resolution* is referred to the size of cells in the resulting grid, or, in other terms, to the size of the step used to sub-sample. This method maintains the regular structure of data, but it provides no control on the loss of accuracy and it is not adaptive. Thus, a large number of samples

may be used even to represent terrain in flat areas, while vertical error could easily exceed the allowed tolerance on areas that contain sudden variations of altitude.

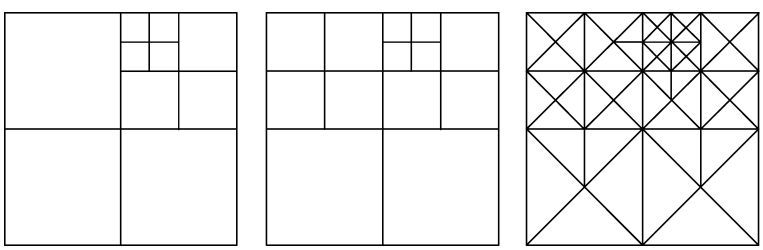
Better results can be obtained by building generalized representations in the form of TINs. Generalization algorithms, run with different thresholds on the same dataset at high resolution, provide a discrete model consisting of a collection of TIN representations at different accuracies. Being adaptive, such representations may contain small triangles in areas where terrain has large variations, and large triangles in relatively flat areas. In this case, the term *resolution* is more related to accuracy than to the size of atomic elements in the representation.

Discrete models support simple queries to extract a representation at fixed resolution. It is sufficient to select the layer corresponding to the desired resolution, and the region of interest within that layer. However, discrete models have several drawbacks: they usually provide only a small number of levels of detail; they cannot relate different representations of the same area at different resolutions; and they cannot combine data at different resolutions within a single representation. The *Delaunay pyramid* proposed by De Floriani in 1989 [5] is a TIN based discrete model in which vertical links between triangles that overlap at successive levels of detail are also maintained. In this sense, it comes midway between discrete and tree-like models. Compact data structures have also been proposed to maintain a Delaunay pyramid with many levels, and some authors have proposed variations of this model that can support selective refinement.

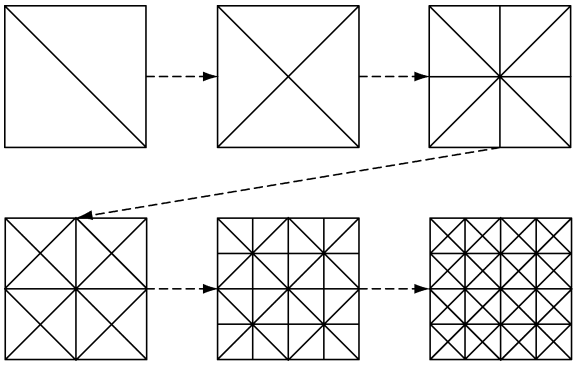


Multi-Resolution Terrain Modeling. Figure 1. A quadtree adaptive subdivision. The corresponding terrain surface has cracks.

In tree-like models, each node represents exactly the same portion of terrain covered by its children. Having a straightforward hierarchical structure, these models can also act as spatial indexes. Most successful models have been developed for regularly distributed data (for instance, the multi-resolution model adopted in *Google Earth* falls in this category). The simplest example consists of a quadtree structure built over a regular grid of data, which directly provides an adaptive version of the discrete model based on subsampling (see Fig. 1). Unfortunately, representations of adjacent portions of terrain from different levels of the quadtree cannot be combined seamlessly, as cracks would appear on the transition between quadrants from different levels (the resulting representation is said to be *non-conforming*). Restricted quadtrees solve this problem by triangulating the quadrants in a quadtree according to some predefined patterns that eliminate cracks, thus obtaining conforming representations (see Fig. 2). In practice, restricted quadtrees may support selective refinement and generate adaptive TINs made of right triangles and having their vertices at a subset of the data in the high resolution grid. Similar results are obtained by a hierarchical decomposition pattern based on triangle bisection (see Fig. 3), proposed in the mid 1990s by Lindstrom et al. [11] and later adopted in many variants by many authors. A square universe S is initially covered by two isosceles right triangles. The bisection rule subdivides a triangle into two similar triangles by splitting it at the midpoint v of its longest edge. A binary tree



Multi-Resolution Terrain Modeling. Figure 2. In a restricted quadtree, cracks can be eliminated by balancing the level of adjacent quadrants and triangulating quadrants with suitable patterns.



Multi-Resolution Terrain Modeling. Figure 3. Recursive triangle bisection generates a regular hierarchy based on the same triangles that appear in the restricted quadtree, but it exhibits a better flexibility.

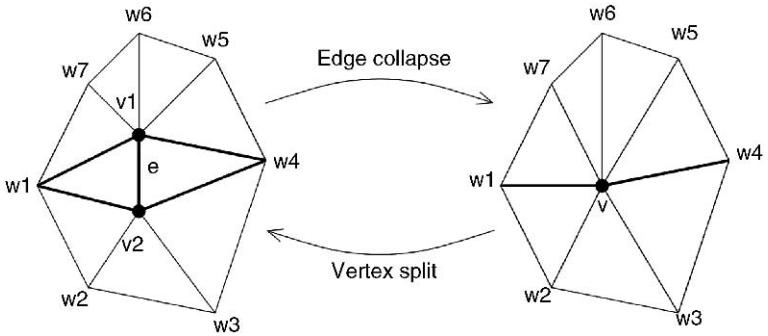
of right triangles is thus obtained. In order to extract conforming meshes, such tree must be traversed in a proper way. In practice, adjacent triangles that are split by introducing a given vertex v will have to be split together during selective refinement. This scheme can be easily generalized to a spherical domain starting, e.g., from an octahedron. Efficient algorithms and data structures have been developed for this scheme, which can support selective refinement efficiently even on planetary size databases. A great advantage of restricted quadtrees and triangle bisection schemes comes from the regular distribution of data. Very compact implicit data structures can be designed, which have a small overhead with respect to maintaining just the single resolution data at the highest available detail, and are also suitable for implementation in secondary memory. Most efficient data structures, though, assume that the collection of all data in the database forms a unique regular grid at a given (high) resolution. In most real cases, however, the database is rather a patchwork of partially overlapping grids at different resolutions. In 2003, Gerstner proposed a data structure for the scheme based on triangle bisection, which works also in the latter case [9].

Cignoni et al. in 2003 proposed a model oriented to terrain rendering, the *BDAM,* which combines triangle bisection with adaptive schemes based on TINs [2]. The triangle bisection scheme is used as a spatial index, to obtain a coarse decomposition of the domain. Standard algorithms traverse such an index during selective refinement, and triangular blocks are collected from the proper levels of the tree. A TIN consisting of possibly a large number of triangles is associated to each triangular block in the spatial index. In this way, the representation resulting from selective refinement is in fact given by the collection of TINs corresponding to triangular blocks extracted during traversal of the index. TINs can be maintained on efficient data structures, suitable to be used in combination with Graphics Processing Units (GPUs), which may greatly improve performance. With this mechanism, performances in terrain visualization are excellent even on huge (planetary size) databases.

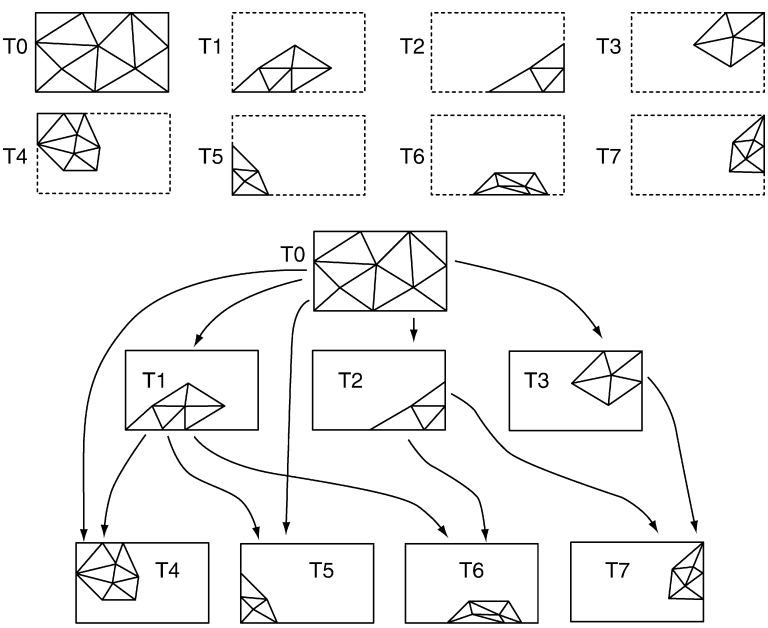
Other tree-like models have been developed based on TINs, which can work on arbitrary datasets. A triangle in a TIN may be refined by inserting a variable number of points either inside it or on its edges on the basis of an error-driven refinement criterion. Edges that survive across different levels of the hierarchy permit to combine surface patches from different levels of the tree, thus supporting selective refinement. Compared to models for regularly distributed data, these latter models can achieve a better ratio between size and accuracy, because no constraint is imposed on the vertex distribution, but need more complicated and expensive data structures to be stored.

Many other models developed (starting in the mid 1990s) have been oriented to irregularly distributed data and are all based on TINs and local modifications. A *local modification* is an operation that substitutes a small (local) portion of a TIN with another representation, formed by a different number of triangles. Modifications can be described either explicitly, by enumerating triangles that are eliminated and triangles that replace them, or, more often, implicitly through some mesh editing operations. The most famous and widely used editing operation is *edge collapse* (see Fig. 4), which is at the basis of the *Progressive Meshes (PM)* introduced by Hoppe in 1996 [10], and of many other models proposed in the literature. Edge collapse consists of collapsing an edge e of a TIN to a point. As a consequence, the two triangles incident at e will collapse to edges and disappear, thus the number of triangles, edges and vertices in the TIN will decrease by two, three, and one unit, respectively. Iterative edge collapse provides a powerful method for terrain generalization. The resulting sequence of collapses, together with their inverse operations called *vertex splits*, and the base mesh obtained from generalization, constitute a PM. Similar models can be built by using local modifications different from edge collapse, provided

that they can be reversed. This simple structure supports the efficient extraction of terrain representations at many different levels of detail, but do not directly support selective refinement. In 1996, Puppo took a more general approach in analyzing models based on local modifications [14]. He proved that the inherent dependency relation between local modifications in a sequence is in fact a partial order, which can be encoded in a directed acyclic graph having such modifications as nodes (see Fig. 5). By traversing such graph in a proper order, selective refinement can be performed efficiently. This general framework consisting of a partial order of local modifications is called a *Multi-Triangulation (MT)*. Among the many schemes that fit in the MT framework, the data structure proposed by El-Sana and Varshney in 1999 is excellent for compactness [7]. Their model is based on edge collapse and just a binary tree of the vertices introduced from collapse operations is maintained, which contains in fact just a subset of the links in the graph of



Multi-Resolution Terrain Modeling. Figure 4. *Edge collapse* is a local modification for iterative terrain generalization: edge *e* together with its endpoints v_1 and v_2 collapse to vertex *v*; triangles adjacent to *e* together with their other edges collapse to edges w_1v and w_4v, respectively. Edge collapse is inverted by a refinement operation called *vertex split*.



Multi-Resolution Terrain Modeling. Figure 5. A sequence of arbitrary local refinement modifications substitute portions of a mesh with other, more refined, groups of triangles. The corresponding Multi-Triangulation is described by a directed acyclic graph: a modification *M* depends on another modification *M′* if and only if *M* eliminates some triangle that was introduced by *M′*.

dependencies of the MT. A clever mechanism based on enumeration of nodes in the tree allows them to retrieve the correct dependencies among nodes and run selective refinement correctly and efficiently.

In the literature, also other kinds of multi-resolution models have been proposed, which follow a functional approach rather than a geometric one (see, e.g., [12]). The basic idea is that a function can be decomposed into a simpler part at low resolution, together with a collection of perturbations called wavelet coefficients which define its details at progressively finer levels of resolution. Wavelets have been widely used for multi-resolution representation and compression of signals and images, while their applications to terrain and surfaces is more recent. The discrete computation of wavelets requires a recursive subdivision of the domain into regular cells like equilateral triangles or squares. Therefore these methods are just suitable for regularly distributed data and resulting hierarchies correspond to either quaternary triangulations or quadtrees.

For a more detailed treatment of multi-resolution terrain modeling see, e.g., [13,4,6] and references therein.

Key Applications

Multi-resolution terrain modeling is essential to manage complexity in those applications that need to either analyze or visualize terrain data at different scales, such as planetary browsers, flight simulators, CAD tools for road design, and all intensive computational tasks related to terrain, such as drainage networks and visibility.

Cross-references

- ▶ Digital Elevation Models
- ▶ Discrete Wavelet Transform and Wavelet Synopses
- ▶ Geographic Information System
- ▶ Quadtrees (and Family)
- ▶ Simplicial Complex
- ▶ Triangulated Irregular Network

Recommended Reading

1. Agarwal P.K. and Suri S. Surface approximation and geometric partitions. In Proc. 5th Annual ACM -SIAM Symp. on Discrete Algorithms, 1994, pp. 24–33.
2. Cignoni P., Ganovelli F., Gobbetti E., Marton F., Ponchio F., and Scopigno R. Planet-sized batched dynamic adaptive meshes (P-BDAM). In Proc. IEEE Visualization, 2003, pp. 147–155.
3. Clark J.H. Hierarchical geometric models for visible surface algorithms. Commun. ACM, 19(10):547–554, 1976.
4. Danovaro E., De Floriani L., Magillo P., Puppo E., and Sobrero D. Level-of-detail for data analysis and exploration: A historical overview and some new perspectives. Comput. Graph., 30(3): 334–344, 2006.
5. De Floriani L. A pyramidal data structure for triangle-based surface description. IEEE Comp. Graph. Appl., 9(2):67–78, 1989.
6. De Floriani L., Magillo P., and Puppo E. Geometric structures and algorithms for geographical information systems. In Handbook of Computational Geometry, J.R. Sack and J. Urrita (eds.), Elsevier Science, Amsterdam, 1999.
7. El-Sana J. and Varshney A. Generalized view-dependent simplification. Comput. Graph. Forum, 18(3):C83–C94, 1999.
8. Fowler R.J. and Little J.J. Automatic extraction of irregular network digital terrain models. In Proc. 6th Annual Conf. Computer Graphics and Interactive Techniques, 1979, pp. 199–207.
9. Gerstner T. Multiresolution compression and visualization of global topographic data. Geoinformatica, 7(1):7–32, 2003.
10. Hoppe H. Progressive meshes. In Proc. 23rd Annual Conf. Computer Graphics and Interactive Techniques, 1996, pp. 99–108.
11. Lindstrom P., Koller D., Ribarsky W., Hodges L.F., Faust N., and Turner G.A. Real-time, continuous level of detail rendering of height fields. In Proc. 23rd Annual Conf. Computer Graphics and Interactive Techniques, 1996, pp. 109–118.
12. Lounsbery M., DeRose T.D., and Warren J. Multiresolution analysis for surfaces of arbitrary topological type. ACM Trans. Graph., 16(1):34–73, 1997.
13. Lübke D., Reddy M., Cohen J.D., Varshney A., Watson B., and Hübner R. Level Of Detail for 3D Graphics. Morgan Kaufmann, Los Altos, CA, 2002.
14. Puppo E. Variable resolution terrain surfaces. In Proc. 8th Canadian Conf. on Computational Geometry, 1996, pp. 202–210.
15. Von Herzen B. and Barr A.H. Accurate triangulations of deformed, intersecting surfaces. In Proc. 14th Annual Conf. Computer Graphics and Interactive Techniques, 1987, pp. 103–110.

Multi-scale

- ▶ Multiple Representation Modeling

Multiscale Views

- ▶ Distortion Techniques

Multiscale Interface

- ▶ Zooming Techniques

Multiset Semantics

▶ Bag Semantics

Multi-Step Query Processing

Peer Kröger, Matthias Renz
Ludwig Maximillian University of Munich, Munich, Germany

Synonyms

Filter/refinement query processing

Definition

A query on a database reports those objects which fulfill a given query predicate. A query processor has to evaluate the query predicate for each object in the database which is a candidate for the result set. Multi-step query processing (filter/refinement query processing) is a technique to speed up queries specifying query predicates that are complex and costly to evaluate. The idea is to save the costs of the evaluation of the complex query predicate by reducing the candidate set for which the query predicate has to be evaluated applying one or more filter steps. The aim of each filter step is to identify as many true hits (objects that truly fulfill the complex query predicate) and as many true drops (objects that truly do not fulfill the query predicate) as possible by applying a less costly query predicate. The remaining candidates that are not pruned as drops or reported as hits in one of the filter steps need to be tested in a refinement step where the exact (costly) query predicate is evaluated. Obviously, the less costly the filter predicates are and the smaller the number of candidates that need to be refined, the higher the performance gain of a multi-step query processing is over a single-step query processing. In addition, if any of the applied filter steps is able to report true hits, first results can be reported to the user significantly sooner by a multi-step query processor compared to a single-step query processor.

Historical Background

In many database applications the management of complex objects is required. For example, the parts of a geographical map such as streets, lakes, forests – or generally *regions* – are stored as polylines or polygons. Queries on these complex objects usually involve complex query predicates that are costly to evaluate. For example, in order to retrieve all regions of a map that intersect with a given query window it is required to test the intersection of the query window and the database polygons which is computationally rather expensive. In such situations, the evaluation of the query predicate (e.g., "intersects the query window") becomes the bottleneck of query processing. Index structures are designed for shrinking down the search space of tentative hits in order to scale well for very large databases. Principally, the aim of index structures is the same as that of the filter-steps in multi-step query processing. However, index structures are only applicable for the first filter step. The reason is that index structures are designed to organize the entire database and cannot be applied to a reduced set of candidates.

To cope with complex data objects and costly query predicates, the paradigm of multi-step query processing (filter/refinement query processing) has been defined originally for spatial queries such as point queries and region queries on databases of spatial objects [6,2]. This paradigm has been applied to similarity search in databases of complex objects performing general similarity queries such as distance range queries [1,3] and *k*-nearest neighbor (*k*NN) queries [4] using costly distance functions. The key idea is to apply one or more filter steps each using cheaper query predicates (e.g., cheaper distance functions), the so-called *filter predicates*, in order to identify as many objects as possible as true hits or true drops. For the remaining candidates, for which the query predicate cannot be decided using any of the filter steps, the exact (more costly) query predicate needs to be evaluated in a refinement step. To ensure correct results, the filter predicates are required to be based on *conservative approximations* of the exact objects. This ensures that if any object does not qualify for a filter predicate, it can also not qualify for the exact query predicate. For example, if the regions of a map are conservatively approximated by minimum bounding rectangles (MBRs) of the corresponding polygons, those regions whose corresponding MBRs do not intersect with the query window cannot intersect with the query window. This conservative property of the filter predicates enables discarding true drops. On the other hand, filter predicates that are based on progressive approximations of the exact objects can be used to identify true hits. For example, if the regions

of a map are progressively approximated by an incircle of the corresponding polygons, those regions whose corresponding incircle intersect with the query window do also intersect with the query window.

Foundations

Multi-step query processing is usually used in applications where the objects in the database are complex and the queries launched on objects rely on costly predicates that cannot be evaluated efficiently. In such applications, the evaluation of the query predicate becomes the bottleneck in query execution.

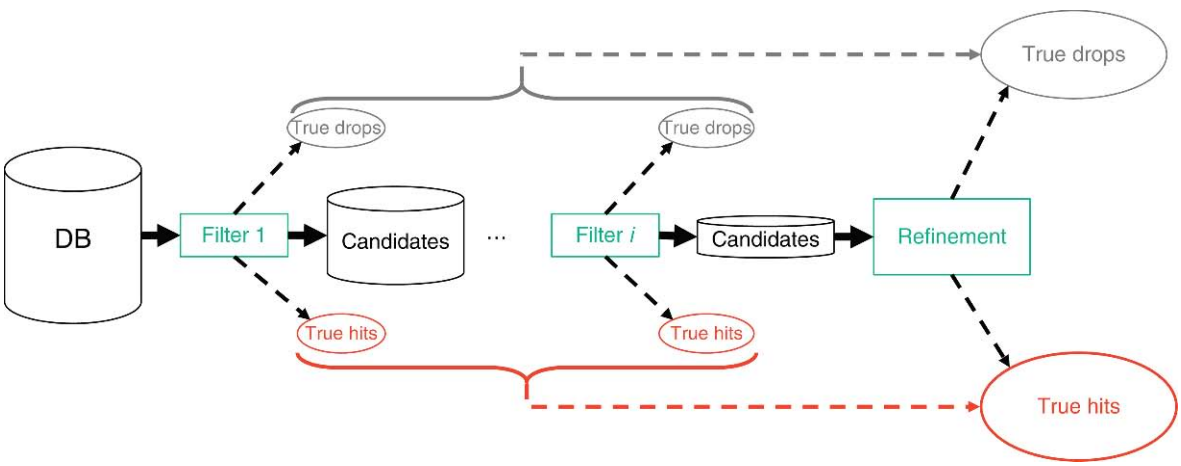
General Schema of Multi-Step Query Processing

Multi-step query processing is based on the following idea: design one or more filter predicates that can be evaluated much faster than the original query predicate and that can be used to shrink down the number of candidates for which it is unknown whether they qualify for the query predicate or not. The query processing starts with all database objects as candidates and applies the designed filters sequentially on the remaining candidates. Each filter ideally identifies true hits that can be added to the result set and true drops that can be pruned. The candidates that cannot be classified as true hits or true drops after all filter steps need to be refined by evaluating the (costly) original query predicate. This general schema is illustrated in Fig. 1. The order in which the single filter steps are applied usually depends on the cost of each filter step and on the *selectivity* of each filter step. The selectivity of a filter step determines the fraction of objects that are identified as true hit or true drop by the corresponding filter and do not need any further processing. In order to produce correct results, obviously, the filter steps must not produce false drops (i.e., drop objects that match the original query predicate according to a filter predicate) and false hits (i.e., report objects that do not match the original query predicate as hits according to a filter predicate).

In order to apply multi-step query processing, it is important to design appropriate filter predicates for the original query predicates of the given application. An appropriate filter predicate can usually be designed by designing a less complex representation that approximates the complex database objects. The evaluation of the query predicate on these less complex object approximations should be less costly than on the original object representation. In the following, special instances of multi-step query processing is discussed in more detail.

Example: Multi-Step Query Processing of Similarity Queries

Usually, similarity between objects is expressed by means of a pair-wise distance function *dist*. A high distance between two objects denotes low similarity of these objects whereas a low distance implies high similarity. For example, if the database objects are points (of any dimensionality), *dist* could be the Euclidean distance, i.e., the vicinity of the corresponding points in the Euclidean space. If the database objects are sequences, *dist* could be the Edit distance. If the database objects are spatial regions (e.g., of a map), *dist* could be the smallest Euclidean distance between the



Multi-Step Query Processing. Figure 1. General schema of multi-step query processing.

corresponding polygons. The two most important and general types of similarity queries are distance range (DR) queries and k-nearest neighbor (kNN) queries.

A distance range query is a general query type in non-standard database systems such as spatial DBS, temporal DBS, and multi-media DBS. Given a query object q, a distance function $dist(.,.)$, and a distance threshold ε, a distance range query returns all database objects o that have a distance less or equal than ε to q, i.e., $dist(q, o) \leq \varepsilon$. They can be efficiently supported using index structures or multi-step query processing.

According to the above definition the query predicate of DR queries is given as follows: all hits o must qualify the predicate $dist(q, o) \leq \varepsilon$, where q is the query object and ε is a distance threshold. The query predicate of kNN queries is quite similar to DR queries: all hits o must qualify the predicate $dist(q, o) \leq d(q, k)$, where q is the query object and $d(q, k)$ is the k-nearest neighbor distance. However, the big difference between DR queries and kNN queries is that the distance threshold ε is given in advance, whereas the value of $d(q, k)$ is usually not known at query time.

A filter predicate for identify true drops (conservative property) can be designed as follows. First, a less complex representations to conservatively approximate the exact objects should to be developed. Usually, this can only be implemented for spatial objects: the conservative approximation must completely contain the exact object, e.g., a minimum bounding box (MBR) is a conservative approximation of a polygon. A second step is essential: A (cheaper) distance function on the approximation must be designed that implements the *lower bounding property*. Let $dist(.,.)$ be the exact distance function on the exact database objects and $LB(.,.)$ the cheaper filter distance. The distance function $LB(.,.)$ lower bounds the exact distance $dist(.,.)$, if the following holds:

$$LB(x, y) \leq dist(x, y)$$

for all database objects x and y. Since the exact predicate of a similarity query usually determines the hits as those objects that have a distance less than a threshold ε to the query object q, all objects o with $\varepsilon < LB(q, o) \leq dist(q, o)$ can be excluded from the result set without further processing. In other words, the filter predicate is similar to the original query predicate, but uses LB instead of $dist$.

For example, if the database contains the regions of a map, and $dist(r_1, r_2)$ is the smallest Euclidean distance between the regions (polygons) r_1 and r_2, an appropriate filter can be designed as follows. The regions are approximated by MBRs and $LB(m_1, m_2)$ is defined as the smallest Euclidean distance between the MBRs m_1 and m_2) of r_1 and r_2, respectively. Obviously, evaluating LB on the MBRs is usually much less complex and costly than evaluating $dist$ on the polygons.

A filter predicate for identifying true hits can be designed analogously. First, a less complex representation to progressively approximate the exact objects should be developed. Again, this can usually be implemented only for spatial objects: the progressive approximation must be completely contained within the exact object, e.g., the maximal circle contained within a polygon (incircle) is a progressive approximation of that polygon. Again, a second step is essential: A (cheaper) distance function on the approximation must be designed that implements the upper bounding property. Again, let $dist(.,.)$ be the exact distance function on the exact database objects and $UB(.,.)$ the cheaper filter distance. The distance function $UB(.,.)$ upper bounds the exact distance $dist(.,.)$, if the following holds:

$$UB(x, y) \geq dist(x, y)$$

for all database objects x and y. All objects o with $\varepsilon > UB(q, o) \geq dist(q, o)$ can be added to the result set without further processing. In other words, the filter predicate is again similar to the original query predicate, but uses UB instead of $dist$.

Sometimes, an approximate representation of the database objects allows the definition of two distance functions, one lower bounding distance and one upper bounding distance. Filter predicates that do not use upper or lower distances cannot be applied to reduce the number of candidates.

Example: Algorithms for Multi-Step Query Processing of Similarity Queries

The algorithm for multi-step distance range queries is rather easy. Since the distance threshold ε is known in advance, in each filter step, true hits and/or true drops are identified as described above, depending on the property of the distance used in the filter predicate.

On the other hand, a multi-step solution for kNN queries is not trivial, because in order to determine the exact value of $d(q, k)$ that can be used to identify objects based on any filter predicates as true hits or true drops, at least k objects need to be refined.

Since the k nearest neighbors are not known in advance, the k objects that need to be refined to determine the exact value of $d(q, k)$ are not known. Obviously, this is a vicious circle.

The multi-step kNN query processing algorithm proposed in [4] tries to approximate $d(q, k)$ by refining any k objects and take the maximum value $d'(q, k)$ of these exact distances. then, a multi-step DR query with query object q and distance threshold $d'(q, k)$ is evaluated. The resulting (refined) objects are ranked in ascending exact distances to q and only the first k objects of this ranking are reported as final result.

In [7], the authors enhance this approach with an algorithm that minimizes the number of refinements. The basic assumption of this algorithm is that only a conservative filter is applied. In the case of only one filter step, the algorithm uses a ranking query in the filter step. Given a query object q and a distance function $dist(.,.)$, a ranking query returns a sequence of the database objects in a database D sorted by ascending distances to q. A ranking query is a general query type in non-standard database systems such as spatial DBS, temporal DBS, and multi-media DBS and can be efficiently supported using index structures. In the context of multi-step query processing in the filter step a ranking query returns a ranking of the database objects sorted in ascending filter distances to the query object q. Initially, the first k objects of the ranking are refined and an approximation $d'(q, k)$ of the true value of $d(q, k)$ is determined from these refined distances as above. Then, in each iteration, the next object from the ranking is fetched as long as the filter distance of the next object in the ranking is greater than the current approximation $d'(q, k)$. As long as this is not the case, the currently fetched object is refined and $d'(q, k)$ is updated. This algorithmic schema can easily be extended to applying multiple filter steps. It can be shown that – if only a conservative filter is implemented – this algorithm is optimal with regard to the number of refinements.

Finally, the algorithm in [5] further enhances the preceding algorithms that take only a conservative filter into account, by additionally using a progressive filter. The algorithm is similar to that in [7] but determines $d'(q, k)$ from the progressive filter as long as this is possible rather than from exact distances. As a consequence, the proposed algorithm reduces the number of refinements significantly. It can be shown that – if both a conservative filter and a progressive filter are implemented – this algorithm is optimal with regard to the number of refinements.

Key Applications

More and more applications suffer from the increasing complexity of the objects and of the functions required to evaluate query predicates on such objects, e.g., complex distance functions or spatial intersections. In the meantime, the efficient support of multi-step query processing is essential for many application areas such as molecular biology, medical imaging, CAD systems, and multimedia databases.

In this context, one of the most important application where multi-step query processing is essential for efficient query processing is similarity search in time series databases. Time series may be very large. Typical similarity queries in time series databases are distance range queries and k-nearest neighbor queries. Due to the curse of dimensionality, similarity queries cannot efficiently be supported by indexing the time-series based on the raw data. A common method to overcome this problem is to reduce the dimensionality of the object descriptions and use this lower-dimensional feature space to index the time series. Similarity queries are then performed using the paradigm of multi-step query processing. In the filter step, approximated similarity distances are computed based on the dimensionality reduced representations, while the refinement step applies similarity distance functions based on the raw time series data. Usually, the filter step is conservative, i.e., the filter distances lower bound the exact distances.

Another important application which requires multi-step query processing is the support of proximity queries in spatial networks like road networks where point objects located within the road network that is represented by a graph are queried. Usually, the objects are positions of buildings or individuals like persons or cars that can have a static location or may move within the network. Example queries could be "retrieve all cars within the road network having a smaller distance to the fast-food restaurant Pinky than 5.0 km" or "give me the three filling stations having the smallest distance to my actual position." Since the motion of the objects is restricted by the network, i.e., objects can only move along a path in the network graph, the distance between two objects is not measured using the Euclidean distance. Rather, the length of the shortest path between two objects is used as distance measure. For each

distance computation it is necessary to apply the Dijkstra algorithm which is too expensive to answer such proximity queries on large databases in real time. Therefore, distance approximations are needed, which can be computed more efficiently and can be used in the filter step of a multi-step query processing algorithm. The simplest road-network distance approximation that fulfills the lower bound criterion is the Euclidean distance. Another method to achieve suitable distance approximations is the pre-computation of distances based on certain landmarks (reference nodes). The distance approximation based on landmarks has the advantage that, in addition to the lower bounding distance approximation, it is possible to compute a distance approximation which fulfills the upper bounding property.

A further important application of multi-step query processing is the support of spatial queries in spatial databases, i.e., databases containing objects having a spatial extension. One of the most important query types in such databases is the point-in-polygon test. Given a database with two-dimensional polygon objects and a certain query point, retrieve all polygons that include the query point. Several filter steps can be applied for this problem to avoid unnecessary point-in-polygon-tests. For example, the polygons can be conservatively approximated by minimum bounding rectangles (MBRs). Obviously, MBRs that do not contain the query point can be discarded as true drops. On the other hand, progressive approximations of the polygons can be used to identify true hits.

Cross-references

- ▶ Closest-Pair Query
- ▶ High Dimensional Indexing
- ▶ Indexing Metric Spaces
- ▶ Nearest Neighbor Query
- ▶ Spatial Indexing Techniques
- ▶ Spatial Join
- ▶ Spatio-Temporal Data Mining

Recommended Reading

1. Agrawal R., Faloutsos C., and Swami A. Efficient similarity search in sequence databases. In Proc. 4th Int. Conf. on Foundations of Data Organization and Algorithms, 1993, pp. 69–80.
2. Brinkhoff T., Horn H., Kriegel H.-P., and Schneider R. A storage and access architecture for efficient query processing in spatial database systems. In Proc. 3rd Int. Symp. Advances in Spatial Databases, 1993, pp. 357–376.
3. Faloutsos C., Ranganathan M., and Manolopoulos Y. Fast subsequence matching in time series database. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1994, pp.419.
4. Korn F., Sidiropoulos N., Faloutsos C., Siegel E., and Protopapas Z. Fast nearest neighbor search in medical image databases. In Proc. 22th Int. Conf. on Very Large Data Bases, 1996, pp. 215–226.
5. Kriegel H.-P., Kröger P., Kunath P., and Renz M. Generalizing the optimality of multi-step k-nearest neighbor query processing. In Proc. 10th Int. Symp. Advances in Spatial and Temporal Databases, 2007, pp. 75–9.
6. Orenstein J. and Manola F. Probe spatial data modelling and query processing in an image database application. IEEE Trans. Softw. Eng., 14(5), 1988.
7. Seidl T. and Kriegel H.-P. Optimal multi-step k-nearest neighbor search. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 154–16.

Multi-Tier Architecture

Heiko Schuldt
University of Basel, Basel, Switzerland

Synonyms

n-tier architecture; Multi-layered architecture

Definition

A *Multi-tier Architecture* is a software architecture in which different software components, organized in tiers (layers), provide dedicated functionality. The most common occurrence of a multi-tier architecture is a three-tier system consisting of a data management tier (mostly encompassing one or several database servers), an application tier (business logic) and a client tier (interface functionality). Novel deployments come with additional tiers. Web information systems, for instance, encompass a dedicated tier (web tier) between client and application layer.

Conceptually, a multi-tier architecture results from a repeated application of the client/server paradigm. A component in one of the middle tiers is client to the next lower tier and at the same time acts as server to the next higher tier.

Historical Background

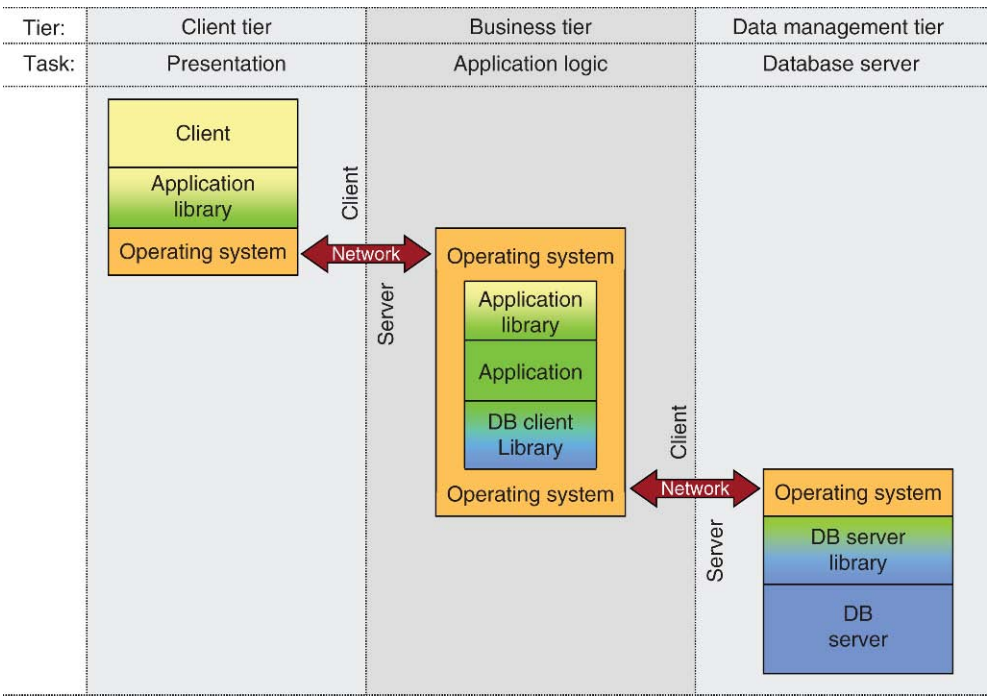
Early generation software systems have been built in a monolithic way. This means that all the different tasks for implementing a particular application and presenting the results to a user are provided by a single

dedicated software component. With the advent of client/server architectures in the 1980s, different tasks could be separated and possibly even be distributed across network boundaries. In a client/server architecture (two tier architecture), the client is responsible for presenting the application to the user while the server is in charge of data management. For the provision of business logic, two alternatives have emerged. First, in so-called fat client/thin server architectures, the client also provides business logic, in addition to presentation and user interfaces. This can be realized by using SQL against the underlying database server in the application program run by the client, either by embedding SQL into a higher programming language or by using the database server's call level interface (e.g., JDBC, ODBC). Second, in thin client/fat server architectures, the database server also provides business logic while the client solely focuses on presentation issues. Fat servers can be realized by using persistent stored modules or stored procedures inside the database server. In the case of evolving business logic, fat client architectures, although being the most common variant of client/server systems, impose quite some challenges when new client releases need to be distributed in large deployments. In addition, a fat client architecture usually comes along with a high network load since data is completely processed at the client side. Fat servers, in contrast, impose a single point of failure and a potential performance bottleneck.

Three-tier architectures thus are the next step in the evolution of client/server architectures where both client and database server are freed from providing business logic. This task is taken over by an application layer (business tier) between client and database server. In multi-tier architectures, additional tiers are introduced, such as for instance a web tier between client and application layer.

Foundations

Multi-tier systems follow an architectural paradigm that is based on separation of concerns. The architecture considers a vertical decomposition of functionality into a stack of dedicated software layers. Between each pair of consecutive layers, a client/server style of interaction is applied, i.e., the lower layer acts as server for the next higher layer (see Fig. 1). Typical tiers in a three-tier architecture are *data management*, *business* and *client* tier. Multi-tier architectures consider additional layers, such as a web tier which hosts servlet containers and a web server and which is located between client tier and application tier.



Multi-Tier Architecture. Figure 1. Structure of a three tier architecture.

In addition to vertical decomposition and distribution across tiers, in many cases multi-tier architectures also leverage horizontal distribution within tiers. For the data management tier, this means that several distributed database servers can be used. Most commonly, horizontal distribution is applied at the business tier, i.e., providing several application server instances [7].

The main benefit of multi-tier applications is that each tier can be deployed on different heterogeneous and distributed platforms. Load balancing within tiers, especially for the application tier, is supported by distributing requests across the different application server instances. This can be implemented by a dispatcher which accepts calls from the next higher layer and distributes them accordingly (this is done, for instance, in TP Monitors which allow to distribute requests among application processes at the middle tier in a three-tier architecture).

When multi-tier architectures are used in a business context, they have to support transactional interactions. Due to the inherent distribution of software components across layers and potentially even within layers, distributed transactions are needed. This is usually implemented by a two-phase commit protocol (2PC) [5] (depending on the application server and the middleware used, this can be done, for instance, via CORBA OTS, the Java Transaction Service JTS, etc.). While 2PC provides support for atomicity in distributed transactions, it does not take into account the layered architecture where transactions at one layer are implemented by using services and operations of the next lower layer. Multi-level transactions [11] take this structure into account. SAP ERP [4], for instance, applies multi-level transactions by jointly considering the application server and data management tier. Asynchronous interactions between components in a multi-tier architecture require a message-oriented middleware (MOM). In this case, transactional semantics can be supported by persistent queues and queued transactions [1].

In order to increase the performance of multi-tier systems and to improve response times, caching is used at the application tier. For this, different database technologies such as replication, materialized views, etc. can be applied outside the DBMS [6].

Key Applications

Due to the proliferation of both commercial and open source application servers, multi-tier architectures can be found in a very large variety of different domains. Applications include, but are not limited to, distributed information systems, Web information systems, e-Commerce, etc.

Experimental Results

The Transaction Processing Performance Council (TPC) has defined a benchmark, *TPC-App*, for evaluating the business tier and in particular the performance of application servers in a three- or multi-tier architecture [10]. It includes Web Service interactions, distributed transactions, and asynchronous interactions via message-oriented middleware (reliable messaging and persistent queues).

Cross-references

- ▶ Application Server
- ▶ Client/Server Architecture
- ▶ Database Middleware
- ▶ Distributed Transaction Management
- ▶ Java EE
- ▶ Message Queuing Systems
- ▶ Middleware Support for Database Replication and Caching
- ▶ Multilevel Transactions and Object-Model Transactions
- ▶ Replication in Multi-Tier Architectures
- ▶ Service Oriented Architecture
- ▶ Transactional Middleware
- ▶ Web Services
- ▶ Web Transactions

Recommended Reading

1. Bernstein P. and Newcomer E. Principles of Transaction Processing. Morgan Kaufmann, Los Altos, CA, 1997.
2. Birman K. Reliable Distributed Systems: Technologies, Web Services, and Applications. Springer, Berlin, 2005.
3. Britton C. IT Architectures and Middleware. Addison Wesley, Reading, MA, USA, 2001.
4. Buck-Emden R. and Galimow J. SAP R/3 System: A Client/Server Technology. Addison-Wesley, Reading, MA, USA, 1996.
5. Lindsay B., Selinger P., Galtieri C., Gray J., Lorie R., Price T., Putzolu F., and Wade B. Notes on Distributed Databases. IBM Research Report RJ2571, San Jose, CA, USA, 1979.
6. Mohan C. Tutorial: Caching Technologies for Web Applications. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001.
7. Mohan C. Tutorial: Application Servers and Associated Technologies. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002.
8. Myerson J. The Complete Book of Middleware. Auerbach, Philadelphia, PA, 2002.

9. Orfali R., Harkey D., and Edwards J. Client/Server Survival Guide. Wiley, 3rd edn., 1999.
10. Transaction Processing Performance Council.TPC-App. http://www.tpc.org/tpc_app/default.asp, 2008.
11. Weikum G. and Schek H.J. Concepts and Applications of Multilevel Transactions and Open Nested Transactions. In Database Transaction Models for Advanced Applications, K. Elmagarmid (ed.), Morgan Kaufmann, Los Altos, CA, 1992, pp. 515–553.
12. Weikum G. and Vossen G. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control. Morgan Kaufmann, Los Altos, CA, 2001.

Multivalued Dependency

Solmaz Kolahi
University of British Columbia, Vancouver, BC, Canada

Synonyms

MVD

Definition

A *multivalued dependency (MVD)* over a relation schema $R[U]$, is an expression of the form $X \twoheadrightarrow Y$, where X, $Y \subseteq U$. An instance I of $R[U]$ satisfies $X \twoheadrightarrow Y$, denoted by $I \models X \twoheadrightarrow Y$, if for every two tuples t_1,t_2 in I such that $t_1[X] = t_2[X]$, there is another tuple t_3 in I such that $t_3[X] = t_1[X] = t_2[X]$, $t_3[Y] = t_1[Y]$, and $t_3[Z] = t_2[Z]$, where $Z = U - XY$ (XY represents $X \cup Y$). In other words, for every value of X, the value of attributes in Y is independent of the value of attributes in Z. A multivalued dependency $X \twoheadrightarrow Y$ is a special case of a *join dependency* expressed as $\bowtie[XY,X(U - XY)]$, which specifies that the decomposition of any instance I satisfying $\bowtie [XY,X(U - XY)]$ into $\pi_{XY}(I)$ and $\pi_{X(U-XY)}(I)$ is lossless, i.e., $I = \pi_{XY}(I) \bowtie \pi_{X(U-XY)}(I)$.

Movies			
Title	*Director*	*Actor*	*Year*
Pulp Fiction	Quentin Tarantino	John Travolta	1994
Pulp Fiction	Quentin Tarantino	Samuel L. Jackson	1994
The Matrix	Andy Wachowski	Keanu Reeves	1999
The Matrix	Andy Wachowski	Laurence Fishburne	1999
The Matrix	Larry Wachowski	Keanu Reeves	1999
The Matrix	Larry Wachowski	Laurence Fishburne	1999

Key Points

Multivalued dependencies, like functional dependencies, can cause redundancy in relational databases. For instance, in the following table, each director of the movie *The Matrix* is recorded once per actor of the movie, and this is because the instance satisfies the MVD *title* $\twoheadrightarrow$ *director*.

Multivalued dependencies have been considered in the normalization techniques that try to improve the schema of a database by disallowing redundancies. The most common normal form that takes MVDs into account is the Fourth Normal Form (4NF). The implication problem for MVDs can be solved in polynomial time. That is, given a set Σ of MVDs, it is possible to check whether an MVD $X \twoheadrightarrow Y$ is logically implied by Σ, denoted by $\Sigma \models X \twoheadrightarrow Y$, in the time that is polynomial in the size of Σ and $X \twoheadrightarrow Y$. Multivalued dependencies are usually considered together with functional dependencies (FDs) in the normalization of relational data. There is a sound and complete set of rules (axioms) that can be used to infer new dependencies from a set of MVDs and FDs defined over a relation $R[U]$:

MVD0 (complementation): If $X \twoheadrightarrow Y$, then $X \twoheadrightarrow (U - X)$.
MVD1 (reflexivity): If $Y \subseteq X$, then $X \twoheadrightarrow Y$.
MVD2 (augmentation): If $X \twoheadrightarrow Y$, then $XZ \twoheadrightarrow YZ$.
MVD3 (transitivity): If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow (Z - Y)$.
FMVD1 (conversion): If $X \rightarrow Y$, then $X \twoheadrightarrow Y$.
FMVD2 (interaction): If $X \twoheadrightarrow Y$ and $XY \rightarrow Z$, then $X \rightarrow (Z - Y)$.

It is also known that the set {MVD0,...,MVD3} is an axiomatization for MVDs considered alone.

Cross-references

- ▶ Fourth Normal Form
- ▶ Functional Dependency
- ▶ Join
- ▶ Join Dependency
- ▶ Normal Forms and Normalization
- ▶ Projection

Recommended Reading

1. Abiteboul S., Hull R., and Vianu V. Foundations of Databases. Addison-Wesley, Reading, MA, USA, 1995.

Multivariate Data Visualization

- ▶ Dynamic Graphics

Multivariate Visualization Methods

Antony Unwin
Augsburg University, Augsburg, Germany

Synonyms

Graphical displays of many variables

Definition

Multivariate datasets contain much information. One- and two-dimensional displays can reveal some of this, but complex pieces of information need more sophisticated displays that visualize several dimensions of the data simultaneously. Usually several displays are needed.

Historical Background

Graphical displays have been used for presenting and analysing data for many years. Playfair [10] produced some fine work over 200 years ago. Minard prepared what Tufte has called "the finest graphic ever drawn" in the middle of the nineteenth century, showing Napoleon's advance on and retreat from Moscow, including information on the size of the army and the temperature at the time. Neugebaur introduced many innovative ideas in the 1920s and 1930s. Most of these graphics are primarily one- or two-dimensional. Techniques for displaying higher dimensional data have mainly been suggested more recently.

Foundations

There are two quite different aims of data display: analysis and presentation. Graphics aid analysts in understanding data and in determining structure. Graphics are good for identifying outliers, for picking out local patterns, and for recognizing global features. Graphics are also valuable for conveying that information to others. Wilkinson's book [13] defines a formal structure. Unwin et al. [12] discuss graphics for large datasets. Theus et al. [11] present interactive graphics for exploring data. The Handbook of Data Visualization [2] provides an overview of the current state of play.

For displaying multivariate data, indeed for displaying data in general, it is important to distinguish between different data types. Variables may be categorical, ordinal, continuous, temporal, spatial or logical (and other specialist types could be added as well). In data analysis the most common types are continuous and categorical. Ordinal may sometimes be treated as categorical (when there are only a few distinct values) and sometimes as continuous (when there are many).

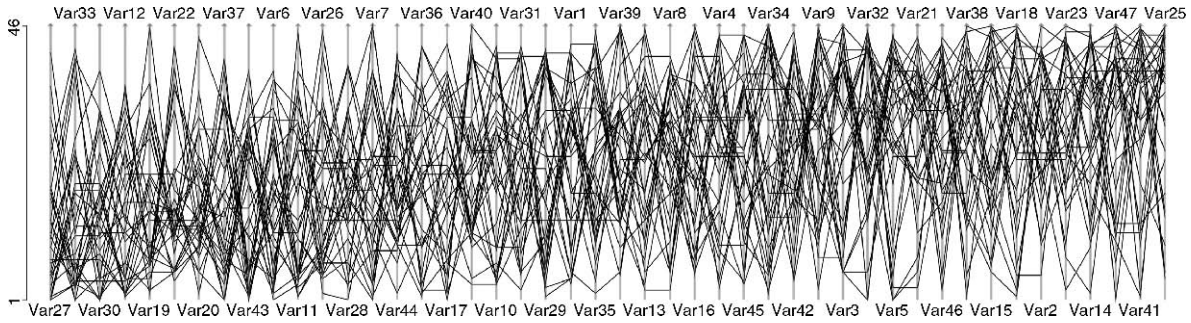
No printed graphic can display more than two dimensions fully at once. Multivariate graphics use projections, conditioning, and linking to capture higher dimensional information. Some displays can deal with very large numbers of cases (area displays such as mosaic plots), and some can potentially handle very many variables (parallel coordinate plots). Most displays are limited in both dimensions. One strategy is to use small multiples, multiple versions of the same graphic restricted to subsets of the data. Trellis plots [1] are the most important example of this approach. Whether many small displays are used or a range of large displays (which might be referred to as large multiples), more than one display will always be necessary to reveal the information in the data.

It is essential to bear in mind that to have enough evidence to confirm complex relationships lots of data are needed. Think of determining the effects of all the influences on car insurance premiums or of estimating the effects of factors in health studies (for instance breast cancer risk). Graphical methods must be able to deal with large datasets to be fully useful [12].

Multivariate Continuous Data

For multivariate continuous data the most popular graphic solution is parallel coordinate plots [8]. Further approaches include scatterplot matrices (sploms), showing all scatterplots of two variables at a time, and trellis plots, which display the data in subsets defined by conditioning variables. Glyphs, individual images for each case whose form depends on the separate variable values, can be an interesting possibility for smaller datasets (at most a few hundred cases). Matrix visualizations [2] are also interesting for smaller datasets and display each value by a color coding, with cases in the rows and variables in the columns. Including options for ordering both rows and columns is essential. Microarray data are often displayed in this way.

Other alternatives, known under the common heading of dimension reduction plots, display two-dimensional approximations of multivariate data, e.g., multi-dimensional scaling (MDS) and biplots. Dynamic methods include the grand tour and projection

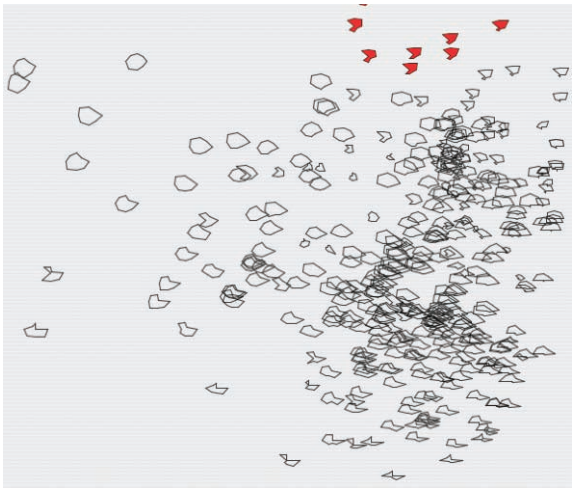


Multivariate Visualization Methods. Figure 1. A parallel coordinate plot of the ratings of 46 wines by 32 judges. The axes are common scaled and have been ordered by mean values.

pursuit [3], both of which work by moving smoothly through two dimensional projections of the data.

Figure 1 shows a parallel coordinate plot of 32 judges' rankings of 46 American and French Cabernet wines from a 1999 tasting. The axes (one for each wine) have been given a common scale and sorted by their mean ratings from left to right, so that the highest ranked wine is on the far left and the lowest on the far right. What is striking is the lack of agreement amongst the judges. While there is a discernible trend, it is obscured by the high variability of the ratings. Most wines were ranked best by at least one judge and worst by at least one other. So the main message of this plot is that while the league table of results and statistical tests (whether the ordering was significantly non-random) imply a consensus ranking of the wines, the data convey otherwise. Parallel coordinate plots, like all high-dimensional plots, require fine-tuning to reveal information. In this case, common scaling and sorting were important tools. As parallel coordinate plots are covered in another entry, they are not discussed in detail here. One interesting new variant is represented by textile plots [9] in which the axes are rescaled to make the individual lines linking cases as horizontal as possible.

In MDS [4] the attempt is made to find a low-dimensional (almost always two dimensional) approximation to high dimensional data by positioning points so that the distances between them in the low dimensional display are close to the distances between them in the original dimension. For a high number of dimensions this is unlikely to be effective, but it often produces interesting views. The MDS display depends on the criterion used to match the distances (e.g., emphasising the absolute differences or the relative differences). Since all possible pairs must be considered, it is not efficient for large datasets. MDS displays are not unique for two reasons: an optimal solution in terms of the criterion will not necessarily be found; any solution is rotation invariant. Figure 2 shows an MDS plot of five-dimensional data on 381 cars sold in Germany. Each case is represented by a circular-based glyph using these five dimensions and two additional ones. The selected group at the top of the display seem relatively well separated in this view. They are all midsize luxury cars.



Multivariate Visualization Methods. Figure 2. An MDS display based on five variables for cars sold in Germany. Each car is represented by a circular-based glyph.

Biplots were developed by Gabriel [5]. He pointed out that both cases and variables could be plotted on the same approximating low dimensional plot. The two axes are usually chosen to be the first two principal

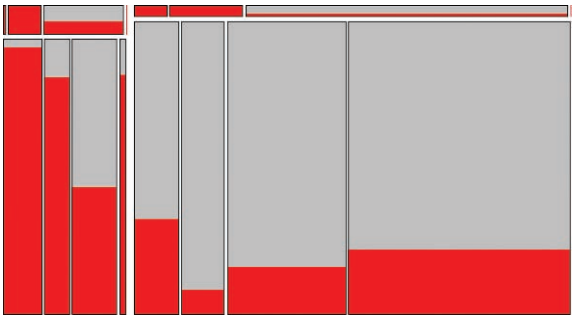
components. Lines representing variables which are well approximated appear longer than those which represent variables badly approximated and the angles between the lines reflect the correlations between the variables in the low dimensional hyperplane. More complex biplots are also possible [6]. Like MDS displays, biplots will not work well in general, neither for many cases nor many variables, but often the two-dimensional projections produced can offer insightful views of the data.

Multivariate Categorical Data

Continuous data can always be sensibly binned and compressed, while retaining the option of zooming in to reveal the full level of detail. This does not hold for categorical data, where it may not be possible to combine any of the individual categories with others. While displays of single categorical variables are simple, the number of combinations rises exponentially with the number of variables. One binary variable can be displayed in a barchart of two columns. Twenty binary variables would give rise to 2^{20} combinations, a little over a million, many of which are likely to be empty, even for extremely large datasets.

Classic mosaicplots were suggested by Hartigan [7] for displaying a small number of categorical variables in a multivariate way. Other variations (multiple barcharts, fluctuation diagrams, equal binsize plots, doubledecker plots) [12] are often more useful. All depend very much on a careful choice of the ordering of variables and on an informative choice of size and aspect ratio. The ordering of variables determines which comparisons can be made, while the aspect ratio influences how well the comparison can be made.

Figure 3 shows a mosaic plot of the Titanic data with the order of variables, gender, age, class. The block of four equally tall columns at the left of the display shows the numbers of adult women in each of the three passenger classes and the crew, with the proportion who survived highlighted. It is obvious that survival rates for adult women declined across the three passenger classes (the number of women in the crew was too small for any conclusion to be drawn). The next block of four columns relates to adult males and shows that the second class adult males had the lowest survival rate, a rather surprising result. The smaller bars at the top of the display refer to the children on board. The survival rates for males and females within classes can be compared approximately in this display, but

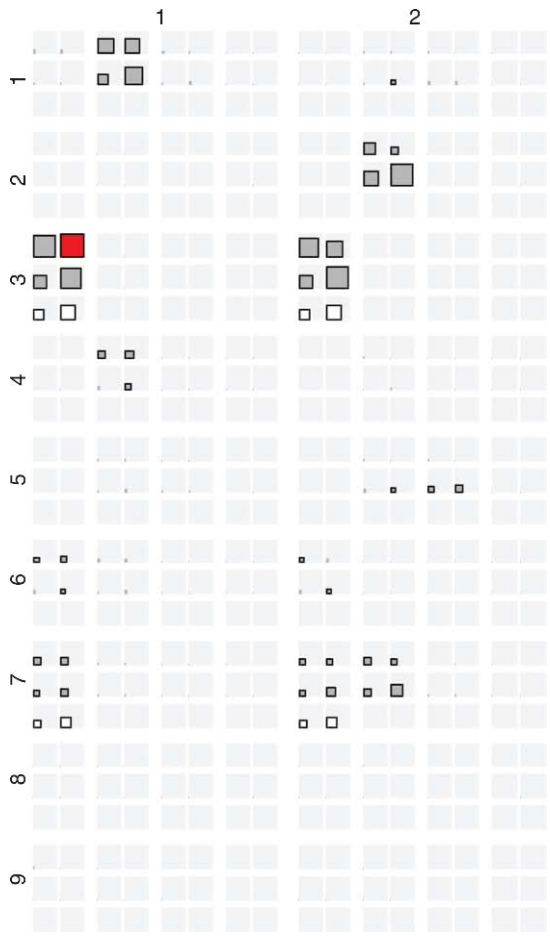


Multivariate Visualization Methods. Figure 3. A mosaicplot of the numbers who sailed on the Titanic with the survivors selected. Women are to the left and men to the right, adults are below and children above. Within these groups the classes are ordered first, second, third, crew.

clearly another display would be better for that, one using the variable ordering class, age, gender. Even in this dataset with only four variables, one plot is not enough.

The main idea underlying all mosaicplots is that each combination of variable values is displayed by a rectangle whose area is proportional to the number of cases with that combination. The layout of the combinations is key in determining the interpretation, which can be difficult at the best of times, and is eased by providing interactive tools to query and adjust the graphic. Multiple barcharts are for comparing distributions of subsets (and are therefore related to trellis plots). Fluctuation diagrams are best for larger numbers of combinations to identify which are most common. Equal binsize plots and doubledecker plots are for comparing highlighted proportions.

Figure 4 shows a fluctuation diagram of a dataset from the Pakistan Labour Force Survey. Five variables are considered (the numbers of categories, including missings, are in brackets): gender [2], relation to head of household [9], marital status [4], literacy [3], and urban/rural [2], making 432 possible combinations in all. Although there are just under 140,000 cases, many of the combinations are empty or rare (e.g., few women and few single men are heads of households). The biggest single combination (male, son in household, never married, literate, living in a rural area) is highlighted and includes 9,678 cases or 7% of the dataset. Using interactive querying and animating the construction of the plot, one variable at a time, aids interpretation considerably. No display of several categorical variables



Multivariate Visualization Methods. Figure 4. A fluctuation diagram of five variables from the Pakistani Labour Force Survey: gender, relation to head of household, marital status, literacy, urban/rural. The biggest single combination is highlighted.

at once can either be easy to grasp immediately or convey all the potentially available information.

Interactive Graphics and Multivariate Graphics

Although both parallel coordinate plots and mosaicplots can be used for static plots, they are much more effective when used interactively. Their necessarily complex nature (after all, they have to display multivariate structure) demands careful scrutiny to grasp the information in them to the full, and the gain of understanding can be considerably enhanced when these graphics are empowered with interactive tools.

Interactive graphics may also be used to gain insight into multivariate datasets using one- or two-dimensional displays. Multiple linked simple displays of the same dataset can be easier to interpret than complex multivariate plots.

Key Applications

Descriptive statistics and Exploratory Data Analysis.

Future Directions

Many other more or less esoteric multivariate visualizations have been proposed. None should be dismissed out of hand, every visualization is probably ideal for some particular dataset. Nevertheless any succesful graphic should satisfy a number of criteria: it should be based on a readily recognizable and interpretable concept; it should be flexible and capable of being made interactive; it should be able to handle more than just three or four dimensions.

Displaying and interpreting even four-dimensional data is tricky. Dominating features can usually be seen, more subtle effects cannot. In higher dimensions the difficulties become much greater. At the moment it is impossible to visualize large numbers of categorical variables and although several hundred continuous variables can readily be displayed in parallel coordinate plots, the chances of identifying important features are slim. Nevertheless, graphics displays are useful for checking results found analytically and this can be very valuable. Visualizing multivariate data is new and progress is to be expected.

Visualization is currently mainly used for presentation of data, rather than for exploration of data. A single graphic can only display a limited number of aspects of a multivariate dataset and many are needed to convey all information available. The development of multivariate graphics should consider the design of sets of graphics rather than more elaborate versions of single ones. More interactive tools will have be developed. Sorting, rescaling, and querying are just some of the basics required.

Visualization is an important component of data analysis. It provides a complementary approach to analytic modeling and is much more suited to carrying out exploratory data analysis. Results found by models should be checked with graphics and ideas generated with graphics should be investigated analytically. The tighter integration of analytic and graphical methods would be of great advantage.

Cross-references

► Data Visualization
► Dynamic Graphics

- ▶ Parallel Coordinates
- ▶ Parallel Coordinates Plot (PCP)
- ▶ Visual Data Mining
- ▶ Visualizing Categorical Data
- ▶ Visualizing Quantitative Data

Recommended Reading

1. Becker R., Cleveland W., and Shyu M.J. The Visual Design and Control of Trellis Display. J. Computational and Graphical Statistics, 5:123–155, 1996.
2. Chen C.H., Haerdle W., and Unwin A. Handbook of Data Visualization. Springer, Berlin, 2007.
3. Cook D. and Swayne D. Interactive and Dynamic Graphics for Data Analysis. Springer, New York, 2007.
4. Cox M. and Cox M. Multidimensional Scaling. Chapman and Hall, London, 2001.
5. Gabriel K. The biplot - graphic display of matrices with application to principal component analysis. Biometrika, 58:453–467, 1971.
6. Gower J. and Hand D. Biplots. Chapman & Hall, London, 1996.
7. Hartigan J.A. and Kleiner B. Mosaics for Contingency Tables. In Proc. 13th Symposium on the Interface, 1981, pp. 268–273.
8. Inselberg A. Parallel Coordinates. Springer, New York, 2008.
9. Kumasaka N. and Shibata R. High Dimesional Data Visualisation: the Textile Plot. Computational Statistics and Data Analysis, 52(7):3616–3644, 2008.
10. Playfair W. Playfair's Commercial and Political Atlas and Statistical Breviary. Cambridge University Press, London, 2005.
11. Theus M. and Urbanek S. Interactive Graphics for Data Analysis. CRC Press, London, 2008.
12. Unwin A.R., Theus M., and Hofmann H. Graphics of Large Datasets. Springer, New York, 2006.
13. Wilkinson L. The Grammar of Graphics. Springer, New York, 2nd edn., 2005.

Multi-Version Concurrency Control

▶ Multi-version Serializability and Concurrency Control

Multi-Version Concurrency Control Algorithms

▶ Multi-version Serializability and Concurrency Control

Multi-Version Database

▶ Supporting Transaction Time Databases

Multi-Version Databases

▶ Multi-version Serializability and Concurrency Control

Multi-version Serializability and Concurrency Control

Wojciech Cellary
Poznan University of Economics, Poznan, Poland

Synonyms

Multi-version databases; Multi-version concurrency control; Multi-version concurrency control algorithms

Definition

Given a multi-version database, where each data item is a sequence of its versions. The number of versions of a data item may be limited or not. If it is unlimited, then each update of a data item over the limit gives rise to its next version. If it is limited, than each update of a data item replaces its oldest version. In case of limited number of versions, a database is called a K-version database. In multi-version databases any read operation of a data item, subsequent to a write operation of this data item, may access any of its currently existing versions. Thus, a multi-version schedule of a transaction set differs form the ordinary, mono-version schedule by a mapping of the data item read operations into the data item version read operations. Multi-version serializability plays the same role for the multi-version databases, as serializability for the ordinary, mono-version ones. Multi-version serializability is used to prove correctness of a concurrent execution of a set of transactions, whose read and write operations interleave, and moreover, read operations may access one of many available versions of a data item.

Historical Background

Multi-version serializability problem was a hot research topic in mid eighties. First works were published by P.A. Bernstein and N. Goodman [2,3] in 1983. Research was continued by G. Lausen [7], next by S. Muro, T. Kameda, and T. Minoura [8]. The next group of researchers involved was composed of C.H. Papadimitriou, P.C. Kanellakis, and T. Hadzilacos [6,9]. There is a lot of work devoted to different variants of multi-version

concurrency control algorithms. A comprehensive background may be found in [5].

Foundations

Definition of a multiversion schedule. A *multiversion schedule mvs* of a set of transactions τ is a triple $mvs(\tau) = (T(\tau), h, <_{mvs})$, where (i) $T(\tau)$ is a the set of all database operations involved in the transactions of the set τ extended by the database operations of two hypothetical initial and final transactions and which respectively write the initial state of the database and read the final state of the database; (ii) h is a function which maps each read operation $r_{ij}(x) \in T(\tau)$ into a write operation $w_{kl}(x) \in T(\tau)$; and (iii) $<_{mvs} = \cup_i <_{T_i}$ is a partial order relation over $T(\tau)$ such that: if $T_{ij} <_{T_i} T_{ik}$ then $T_{ij} <_{mvs} T_{ik}$, and if $h(r_{ij}(x)) = w_{kl}(x)$ then $w_{kl}(x) <_{mvs} r_{ij}(x)$. Function h defined above maps a read operation of a data item into the write operation of a version of this data item – more precisely – into the write operation which creates the version of the data item read. Relation $<_{mvs}$ is defined by two conditions. The first one states that $<_{mvs}$ honors all orderings stipulated by transactions of the set τ. The second one states that a transaction cannot read a version of a data item until it has been created. A multi-version schedule is *serial* if no two transactions are executed concurrently, otherwise, it is *concurrent*.

Multiversion schedule equivalence. Two multiversion schedules are equivalent if they are *view* and *state equivalent*. Two multiversion schedules $mvs(\tau) = (T(\tau), h, <_{mvs})$ and $mvs'(\tau) = (T(\tau), h', <_{mvs'})$ of the set τ are *view equivalent* if and only if $h = h'$. If the transactions of two multi-version schedules $mvs(\tau)$ and $mvs'(\tau)$ receive an identical view of the database, i.e., if both multiversion schedules are view equivalent, then all the write operations issued by transactions in both schedules are the same. Two multiversion schedules $mvs(\tau) = (T(\tau), h, <_{mvs})$ and $mvs'(\tau) = (T(\tau), h', <_{mvs'})$ of the set of transactions τ are *final-state equivalent* if and only if for every initial state of the database and any computations performed by the transactions contained in τ the final states of the database reached as the result of schedules $mvs(\tau)$ and $mvs'(\tau)$ are identical.

Standard serial multiversion schedule. A serial multiversion schedule $mvs(\tau) = (T(\tau), h, <_{mvs})$ is *standard* if each read operation $r_{ij}(x) \in T(\tau)$ accesses the version of a data item x created by the last write operation $w_{kl}(x) \in T(\tau)$ preceding $r_{ij}(x)$. Since in a serial schedule, for every two transactions T_i and T_k, either all database operations of T_i precede all database operations of T_k or vice versa, the last write operation preceding a read operation is well defined. Note that a standard serial multi-version schedule in multi-version databases corresponds to a serial mono-version schedule in mono-version databases. From the consistency property of each transaction, i.e., from the assumption that each transaction separately preserves database consistency, it follows that a standard serial multi-version schedule must also preserve database consistency. On the basis of the above observation it is possible to define the multi-version serializability criterion [3].

Multi-version serializability criterion. A multi-version schedule $mvs(\tau)$ is *correct* if it is equivalent to any standard serial multi-version schedule of the set τ. Intuitively, the above criterion can be interpreted as follows. A concurrent schedule of a set of transactions in a multi-version database is correct if it is equivalent to a serial schedule of the transactions in which data replication over versions is transparent.

Key Applications

Multi-version serializability is used to prove correctness of concurrency control algorithms devoted to multiversion databases. As an example, consider a multi-version two-phase locking algorithm, called *WAB* [3,1,4], devoted to K-version databases. The concept of multi-version two-phase locking is broader than the concept of mono-version two-phase locking (cf. section on two-phase locking). An algorithm is a *multi-version two-phase locking algorithm* if it satisfies the following conditions:

1. There are two phases of transaction execution: the *locking phase* and the *unlocking phase*. During the locking phase a transaction must obtain all locks it requests. The moment when all locks are granted, which is equivalent to the end of the locking phase and the beginning of the unlocking phase, is called the *commit point* of a transaction. New versions of the data items prepared in the transaction's private workspace are written to the database during the unlocking phase.
2. The execution order of a set of transactions τ is determined by the order of transaction commit points.
3. The execution of any transaction $T \in (\tau)$ does not require locking data items that T does not access.

The concepts of locking and unlocking phases do not have exactly the same meaning as the similar notions used in the mono-version two-phase locking algorithm. In the *WAB* algorithm, the process of setting the so called "certify lock" is two-phase, but not as in the two-phase locking algorithm, the process of accessing data. In the *WAB* algorithm, each transaction initiated in the database and each version of a data item is *certified* or *uncertified*. When a transaction begins, it is uncertified. Similarly, each new version of a data item prepared in the transaction's workspace is uncertified. A *certify operation* is introduced, denoted by $c(w_{ij}(x))$, where $w_{ij}(x)$ is a T_i's write operation, and a new lock mode – the *certify lock* denoted by $CL(x)$. Certify locks are mutually incompatible. The algorithm requires that all certify and read operations of a data item x be $<_{mvs}$ related. Similarly, all certify operations must be $<_{mvs}$ related. The execution order of the certify operations determines a precedence relation $\ll_w$ defined on the set τ. The precedence relation $\ll_w$ specifies the order of transaction executions as follows: $T_i \ll_w T_k$ if and only if there exist such certify operations $c(w_{ij}(x))$ and $c(w_{kl}(x))$ that $c(w_{ij}(x)) \ll_{mvs} c(w_{kl}(x))$.

According to the *WAB* algorithm, any read operation $T_{ij}(x)$ concerns the last certified version of a data item x or any uncertified version of this data item. The version selected depends on a particular implementation of the *WAB* algorithm. Any write operation $w_{ij}(x)$ prepares a new version of a data item x in the workspace of transaction T_i (the version prepared is uncertified). At the end of transaction execution, the transaction and the new versions of the data items it prepared are being certified. The T_i's certification is a two-phase locking procedure. It consists of certify-locking all data items that the transaction T_i accessed to write. The T_i's certification is completed, when all certify locks are set and the following conditions are satisfied:

1. at the moment of T_i's certification, the versions of all data items read by T_i are certified;
2. for each data item x that T_i wrote, all transactions that read certified versions of x are certified.

To satisfy condition (ii), a *certify token* is allocated to each data item x to forbid reading certified versions of x other than the last one. On the other hand, all uncertified versions of x are allowed to be read. When the transaction T_i's certification is completed (the commit point), the procedure for certifying the versions of data items prepared by T_i is initiated. It was proved in [3] that the *WAB* algorithm is correct in the sense that any schedule produced by it is multi-version serializable. The main drawback of this algorithm is a possibility of a deadlock.

Future Directions

In database systems, multiple versions of data items are necessary to ensure transaction atomicity and to recover form crashes. The original idea of multiversion concurrency control based on multiversion serializability was to use those versions also to increase the degree of transaction concurrency, and as a result to improve database performance. However, such double use of versions decreases database reliability, because of the complexity of multiversion concurrency control. For practice, reliability of databases is of ultimate importance. This is why the concept of multiversion concurrency control was not well accepted in practice, except some implementations of two-version concurrency control concerning two values of each data item: before and after write operations. The concept of multiversion concurrency control may find attention in database systems applied in areas where ACID properties may be relaxed.

Cross-references

- ▶ Atomicity
- ▶ Concurrency Control – Traditional Approaches
- ▶ Replicated Database Concurrency Control
- ▶ Serializability
- ▶ Transaction
- ▶ Transaction Management
- ▶ Transaction Models–the Read/Write Approach

Recommended Reading

1. Bernstein P.A. and Goodman N. A sophisticate's introduction to distributed database concurrency control. In Proc. 8th Int. Conf. on Very Data Bases, 1982, pp. 62–76.
2. Bernstein P.A. and Goodman N. Concurrency Control and Recovery for Replicated Distributed Databases. Tech. Rep. TR-20/83, Harvard University, 1983.
3. Bernstein P.A. and Goodman N. Multiversion concurrency control – theory and algorithms. ACM Trans. Database Syst., 8(4):465–483, 1983.
4. Bernstein P.A., Hadzilacos V., and Goodman N. Concurrency Control and Recovery in Database Systems.Addison-Wesley, Reading, MA, 1987.
5. Cellary W., Gelenbe E., and Morzy T. Concurrency Control in Distributed Database Systems, Elsevier Science, North-Holland, 1988.

6. Hadzilacos T. and Papadimitriou C.H. Algorithmic aspects of multiversion concurrency control. J. Comput. Syst. Sci., 33(2):297–310, 1986.
7. Lausen G. Formal aspects of optimistic concurrency control in a multiple version database system. Inf. Syst., 8(4):291–301, 1983.
8. Muro S., Kameda T., and Minoura T. Multi-version concurrency control scheme for database system. J. Comput. Syst. Sci., 29:207–224, 1984.
9. Papadimitriou C.H. and Kanellakis P.C. On concurrency control by multiple versions. ACM Trans. Database Syst., 9(1):89–99, 1984.

Music Metadata

► Audio Metadata

Music Retrieval

► Query by Humming

MVD

► Multivalued Dependency