Adaptively Approximate Techniques in Distributed Architectures

Barbara Catania and Giovanna Guerrini

University of Genoa, Italy
{barbara.catania,giovanna.guerrini}@unige.it

Abstract. The wealth of information generated by users interacting with the network and its applications is often under-utilized due to complications in accessing heterogeneous and dynamic data and in retrieving relevant information from sources having possibly unknown formats and structures. Processing complex requests on such information sources is, thus, costly, though not guaranteeing user satisfaction. In such environments, requests are often relaxed and query processing is forced to be adaptive and approximate, either to cope with limited processing resources (QoS-oriented techniques), possibly at the price of sacrificing result quality, or to cope with limited data knowledge and data heterogeneity (QoD-oriented techniques), with the aim of improving the quality of results. While both kinds of approximation techniques have been proposed, most adaptive solutions are QoS-oriented. Additionally, techniques which apply a QoD-oriented approximation in a QoD-oriented adaptive way (called ASAP - Approximate Search with Adaptive Processing - techniques), though demonstrated potentially useful in getting the right compromise between precise and approximate computations, have been largely neglected. In this paper, we first motivate the problem and provide a taxonomy for classifying approximate and adaptive techniques according to the dimensions pointed out above. Then, we show, through some concrete examples, the benefits of using ASAP techniques in two different contexts.

Keywords: query processing, approximate technique, adaptive technique, Quality of Data, Quality of Service.

1 Introduction

The last decade has seen the raise of new applications and novel processing environments characterized by high heterogeneity, limited data knowledge, extremely high variability and unpredictability of data characteristics and dynamic processing conditions. All such characteristics are shared by most data management applications under distributed architectures, including data integration applications, web services, data streams, P2P systems, and hosting.

Query processing in such new application contexts is characterized by two main features: *adaptivity*, in order to adapt the processing to dynamic conditions that prevent the selection of a single optimal execution strategy, and *approximation*, in order to cope with data heterogeneity, limited data knowledge

G.F. Italiano et al. (Eds.): SOFSEM 2015, LNCS 8939, pp. 65–77, 2015.

during query specification, and limited resource availability, which make precise answers impossible to compute or unsatisfactory from a user point of view.

As discussed in [9], approximate and adaptive techniques can be classified into two main groups. When they are targeted to improve the quality of result, either in terms of completeness or in terms of accuracy, we refer to the techniques as *Quality of Data (QoD)-oriented* techniques. By contrast, when they are used in order to cope with limited or constrained resource availability during processing, we refer to the techniques as *Quality of Service (QoS)-oriented* techniques. For example, in order to maximize completeness or accuracy, QoD parameters have to be taken into account for adapting or approximating query specification and processing. Often, both QoD and QoS parameters are taken into account, in order to provide the best trade-off between resource usage and data quality.

While both QoS-oriented and QoD-oriented approximate techniques have been proposed, most adaptive solutions are QoS-oriented. QoS-oriented approximation is often applied in an adaptive way, that is, when targeted at achieving a QoS goal, e.g., related to load, throughput, or memory, approximation is applied adapting to runtime conditions, possibly ensuring that certain QoD constraints are met or, less frequently, with a QoD-oriented goal (that is, minimizing the introduced inaccuracy). By contrast, very few approaches apply QoD-oriented approximation techniques in an adaptive way.

In [8,9], we claimed instead that QoD-oriented adaptive approaches for QoD-oriented approximation techniques, called *ASAP (Approximate Search with Adaptive Processing) techniques* in the following, may help in getting the right compromise between precise and approximate computations. More generally, we claimed that ASAP can be defined as a new framework under which QoD-oriented approximation techniques, which may adaptively change, at run-time, the degree of the applied approximation, can be defined. For example, this can be achieved by providing execution plans which interleave both precise and approximate evaluations in the most efficient way, dynamically taking decisions concerning when, how, and how much to approximate, with the goal of improving the quality of result with efficiency guarantees. Unfortunately, as far as we know, no general solution has been proposed so far for the problem described above. Some preliminary work has been presented in [19], where the use of adaptive techniques for combining exact (fast) and approximate (accurate) joins when performing dynamic integration has been proposed.

Our group is currently interested in investigating ASAP approaches in different scenarios, taking care of problems related to data heterogeneity and limited data knowledge in different potentially distributed architectures. In this paper, after summarizing existing approximate and adaptive approaches with respect to the type of quality they are targeted to, we consider two different instantiations of the concept of ASAP technique. The first one, that we call *ASAP in the small*, concerns the definition of a specific ASAP technique for a given application context, namely advanced architectures with a limited degree of distribution, like data stream management systems. The second one, that we call *ASAP in the large*, concerns a vision related to environments characterized by a higher

degree of distribution and data heterogeneity. The paper is then concluded by some final considerations and discussion about on-going work on this subject.

2 A Taxonomy for Approximate and Adaptive Techniques

A query processing technique is said to be *adaptive* if the way in which a query is executed may change on the basis of the feedbacks obtained from the environment during evaluation. Adaptive techniques can be characterized by the following components: (i) *subject*, i.e., the elements in the processing affected by the adaptation (e.g., the query execution plan, or the assignment of load to processors); (ii) *target*, i.e., what the technique attempts at adapting to, that is, the properties monitored and the feedbacks collected during evaluation (e.g., data characteristics, arrival rates, network condition, processors load); (iii) *goal* or *aim*, i.e., the parameter(s) appearing in the objective function, that is, what the technique attempts to maximize/minimize (e.g., result data quality, time, throughput, energy consumption).

A query is said to be *relaxed* if its result is either stretched or shrunk when the results of the original query are too few or too many. Preference-based queries, such as top-k or skyline queries, can be considered relaxed queries as well: they can be thought as a shrinking with respect to the overall set of possible results, since they reduce the cardinality of the ranked result set, or as a stretching approach with respect to the set of optimal results. A processing technique is said to be *approximate* if it does not produce the exact result but an approximate one, possibly with some guarantees on the "distance" of the generated solution from the exact one.

Approximate queries or techniques can be characterized by the following components: (i) *subject*, representing the query processing task or the data to which approximation is applied (e.g., query specification, through rewriting or preferences, or processing algorithm); (ii) *target*, representing the information used for the approximation (e.g., ranking function, set of relevant attributes, similarity function, pruning condition, used summary); (iii) *goal* or *aim*, i.e., the parameter(s) appearing in the object function of the technique, that is, what it attempts to maximize/minimize (e.g., result data quality, time, throughput).

As pointed out in the introduction, depending on their aim, approximate and adaptive techniques can be classified into *Quality of Data (QoD)-oriented* techniques, when they are finalized at improving the quality of result, either in terms of completeness or in terms of accuracy, and *Quality of Service (QoS)-oriented* techniques, when they are used in order to cope with limited or constrained resource availability during query processing.

While both QoS-oriented and QoD-oriented approximate techniques have been proposed, most adaptive solutions are QoS-oriented. In the following, each group of proposals is discussed, pointing out the main considered subjects.

QoD-Oriented Approximate Techniques. They provide approximate answers in situations where precise results are not always satisfactory for the user.

High data heterogeneity and limited user knowledge about such data, indeed, may cause precise evaluation produce empty/few answers or too many answers. A solution consists in modifying traditional queries, such as selections and joins, by relaxing their definition or by approximating their evaluation, in order to improve result quality, in terms of completeness and relevance with respect to the original query. Query rewritings like those presented in [6,22,29] and preference-based queries, such as top-k and skyline [7,16,25] are examples of QoD-oriented approximation techniques which relax the original query definition with the goal of returning a more satisfactory answer set. A third group of QoD-oriented approximate techniques concerns processing algorithms for executing a traditional query (e.g., a join) by using ad hoc query processing algorithms which automatically apply the minimum amount of relaxation based on the available data, in order to return a non-empty result similar to the user request. Most QoD-oriented ApQP techniques concern the join operator [18] and face approximate match issues for strings [14] or numeric values [26]. In defining QoD-oriented techniques, QoS guarantees have to be provided, in order to cope with the available resources in the most efficient way.

QoS-Oriented Approximate Techniques. They provide approximate answers, with accuracy guarantees, to computationally expensive operations also in environments characterized by limited or unavailable resources, where a precise result can be obtained only at the price of a unacceptably high response time, communication overhead, occupied space, or it cannot be obtained at all. QoS-oriented techniques have been mainly defined for queries to be executed over either huge amount of data (as in data warehousing systems and in stream-based management systems) or complex data (like spatial data) or because corresponding to very expensive computations (as multi-way joins). Concerning the subject, four main distinct aspects have been considered: query rewriting, e.g., those presented in the data stream context [23]; data reduction, where data themselves are approximated with the aim of reducing or simplifying the dataset over which queries have to be executed [13,27], including load shedding [4,28]; processing algorithms, which modify traditional and non approximate processing techniques in order to generate an approximate result in an efficient way, with respect to the available resources [1,3].

QoS-Oriented Adaptive Techniques. In adaptive query processing, the way in which a query is executed is changed on the basis of the feedbacks obtained from the environment during evaluation. The classical *plan-first execute-next* approach to query processing is replaced either by giving away the notion of query plan at all, as in routing based approaches, where each single data item can participate to the production of the final result taking its own way (*route*) through operators composing the query, or by a *dynamic optimization* process, in which queries are on-the-fly re-optimized through a two-steps solution. Some approaches (e.g., [4,5,24,28]) introduce approximation, thus they have an impact

on QoD, and some others [15,20] process approximate operators (i.e., top-k), but the aim of the adaptation is QoS.

ASAP Techniques. Some of the techniques discussed above exhibit an adaptive behavior and, at the same time, introduce some approximation. Specifically, some approaches to QoS-oriented adaptive processing of QoD-oriented approximate queries have been proposed, but target and goal of the adaptation is QoS, namely processing efficiency [15, 20]. Additionally, adaptive approaches have been proposed for some QoS-oriented approximation techniques (namely, load shedding and data summarization) [4, 5, 24, 28] but only few of them take QoD-information into account as aim [4]. Constraints on data are exploited in [5] to improve QoS (specifically, to reduce memory overhead). However, a QoD-oriented adaptation target is rarely considered, with the exception of [19].

Thus, QoD-oriented adaptive approaches for QoD-oriented approximation techniques (i.e., ASAP techniques) have been so far neglected. However, we claim that such techniques could be very relevant for various data management applications in different application contexts. In the following sections, we will present two examples of ASAP techniques showing their potential.

3 ASAP in the Small

The first example of ASAP technique we present refers to a, potentially distributed, architecture for data stream management. In this context, similarly to [19], ASAP techniques may help in defining adaptive techniques which combine exact (fast) and approximate (accurate) relaxed queries over dynamic (stream) data. In the following, we point out which data and which requests we are going to consider; we then present the targeted problem and we introduce an ASAP technique as a possible solution to the identified problem.

Data. A data stream is a continuous, unbounded, and potentially infinite sequence of data, e.g., tuples. In a data stream, each item is associated with a timestamp, either assigned by the source dataset or by the Data Stream Management System (DSMS), at arrival time. Queries over data streams can be either one-time, if they are evaluated once on a given subset of data, or continuous, if they are continuously evaluated as soon as new data arrive.

Request. According to the STREAM DSMS [2], continuous queries can be evaluated over data streams and time-varying relations. Continuous queries are evaluated, according to the relational semantics, at each time instant on the relation states and on the subsets of the data streams available at that instant. Window operators are applied on data streams in order to compute, at each time instant, a subset of the data items arrived so far in the stream.

The basic idea of a skyline-based approach to query relaxation of selection and join operations is to use a relaxing distance function d (usually, a numeric function) to quantify the distance of each tuple (pair of tuples) from the specified conditions. The relaxed version of the query provides a non-empty answer while being 'close',

according to funcion d, to the original query formulated by the user [17]. Unfortunately, skyline queries for data streams are blocking operators, that require the usage of a specific window-based operator in order to compute, at each instant of time, the finite subset of data from which the best itemset (i.e., the skyline set) is computed. A skyline set is computed in terms of a dominance relation with respect to a given set of attributes, by returning those items that are not dominated by any other item.[1] An example of a relaxed operator for the data stream context, based on a skyline operator, has been proposed in [11], where the concept of relaxation skyline (r-skyline), first introduced in [17] for stored data, has been extended to deal with data streams and queries composed of selection and window-based join operations.

Example 1. Consider an application of habitat monitoring and assume that sensors have been located inside nests, returning several properties of the place around the nest, including light. Assume the user is interested in detecting the nests under a light above a certain threshold. Suppose also this monitoring should last for a long period, thus a continuous selection query is issued. Suppose the query is submitted during daytime, a given light threshold and a given humidity level is chosen, leading to the following query:

```
SELECT idNest, light, humidity
FROM SensorNest [RANGE 2 min]
WHERE light >= 50 and humidity <= 60
```

In the previous query, `[RANGE 2 Minutes]` is a window operator that, when applied on stream `SensorNest`, at each time instant returns the set of tuples arrived in the last 2 minutes which satisfy the specified conditions.

Table 1 reports a portion of data stream `SensorNest`. For each tuple, the arrival time τ, expressed in minutes, is shown. By assuming that the previous query is executed in a precise way, at each instant of time a non-empty result is returned, thus facing a few answer problem. On the other hand, if we interpreted the query as a r-skyline query with respect to both the selection conditions $C_1 \equiv$ `SensorNest.humidity` $<=$ `60` and $C_2 \equiv$ `SensorNest.light` $>=$ `50`, the computation returns different itemsets. As an example, at time 4, tuples s_4 and s_3 belong to the window and we get the following distance values, just computing differences between attribute values and query constants: d(s_3, C_1) = 3, d(s_3, C_2) = 5; d(s_4, C_1) = 5, d(s_4, C_2) = 10. According to the classical notion of dominance, and assuming to prefer lower values, it follows that s_3 dominates s_4 and s_3 is returned as result at time 4. ◊

The Targeted Problem. Precise queries in a data stream management context guarantee a very efficient execution for selection operations, since they are not blocking operations, i.e., they do not require window-based operators for

[1] Given a set of points, each corresponding to a list of values for the relevant attributes, a point A dominates a point B if it is better in at least one dimension and equal to or better than B in all the others, with respect to some ordering [7].

Table 1. SensorNest data stream

tuple	idNest	humidity %	light %	τ
s_1	0001	49	45	1
s_2	0002	47	49	2
s_3	0001	63	45	3
s_4	0003	65	40	4
s_5	0004	66	76	5
s_6	0005	70	70	6
s_7	0006	70	66	7
...	...	...	...	...

their computation, which, on the other hand, is mandatory for join. At the same time, they guarantee maximal accuracy by definition. However, users may not be acquainted of the actual data arriving in streaming and, as a consequence, they may issue queries that, for specific instant of times, return an empty result set, thus potentially decreasing user satisfaction. With reference to the previous example, assuming that 50 is a suitable value for daytime, during light hours, probably, a non empty answer is computed. However, at sunset, the light is getting low and few (or no) data may be returned as answer. Two scenarios may arise to increase user satisfaction: (i) this is exactly what the user wants and no modification to the query has to be specified; (ii) the user may anyway want some results to be returned, the closest to the specified conditions. In the second case, the system should modify the query in order to provide a non-empty result with accuracy guarantees. This behavior can be obtained, for example, by changing selection conditions, through user interaction, similarly to what has been done in [22] for stored data. Unfortunately, this approach is suitable neither for a data stream management context since the query is usually specified once and continuously executed over arriving data nor for a more general distributed environment, where the limited user knowledge about data often make this approach unfeasible. A different approach would be that of executing an r-skyline query, even for selection, thus always obtaining the best result and avoiding the empty/few answer problem, at the price of a costly window-based computation.

From the previous considerations, it follows that, as soon as we want to combine processing efficiency (a QoS parameter) with result accuracy (a QoD parameter), a trade-off arises: the definition of skyline-based relaxation techniques may help in solving the empty answer problem but the price to pay is the introduction of a window-based computation and therefore, in general, a decrease of performance.

The ASAP Proposal. In order to combine the benefits of both precise and relaxed queries, an ASAP approach can thus be considered. The idea is to rely on the usage of an adaptive processing approach, in order to switch from precise selection operations to skyline-based ones as soon as, based on some dynamically monitored QoD parameters, the system understands that this is needed for improving result quality. The same technique may then switch from a skyline-based

computation to a precise one to reduce result size as soon as a QoD parameter indicates that a precise computation can generate a result with QoD guarantees. In the resulting approach, the target of both adaptation and approximation is thus QoD-oriented, while the aim of both adaptation and approximation is both QoD and QoS since the techniques aim at achieving the best trade-off between a QoD parameter, namely result completeness, and a QoS parameter, namely response time, by recurring to (more expensive) skyline-based computations only when needed.

The idea is to model adaptive query processing as a finite state automa in which each state corresponds to one (possibly relaxed) query to be executed [12]. Transition from one state to another is performed during processing using heuristics with the aim of maximizing accuracy, defined according to specific user or system constraints and statistics, computed over already processed data. The overall ASAP framework relies on three main components: (i) *monitor*, which collects aggregate values, related to selectivity and precision of the query in execution; (ii) *assessor*, which determines whether some QoD conditions are satisfied; (iii) *responder*, which, based on assessor predicates, determines whether the query plan should be modified in a certain instant of time.

In order to give an idea of how the ASAP processing works, we suppose that the QoD-oriented user request corresponds to a precise continuous query annotated with specific QoD constraints, over parameters to be computed in a continuous way, like: σ_{avg}, average result cardinality (selectivity constraint); π_{max}, maximal distance of the returned tuples from the specified query conditions (precision constraint); μ, weight for selectivity and precision (trade-off constraint). We can then consider the very simple ASAP automata presented in Figure 1, containing just two states: one corresponding to a precise query and one corresponding to the r-skyline query obtained by relaxing all conditions appearing in the original request. More complex state machines can of course be provided by increasing the number of the considered states, i.e., the number of relaxed queries taken into account.

The computation then proceeds as follows:

- Statistics computed by the monitor may quantify how far the current result is from the selectivity and precision constraints associated with the original request. Based on such statistics, say σ for selectivity and π for precision, an accuracy measure has been provided, which, given an annotated precise query Q and a, either precise or relaxed, query Q measures how far is $Q^{'}$ result with respect to Q' result. A higher accuracy for $Q^{'}$ implies an higher user satisfaction in obtaining $Q^{'}$ result.
- The assessor can then determine whether, during the computation, some QoD conditions are satisfied. The following are examples of some relevant predicates: (i) sel^{+}/sel^{-}: too many/too less results are generated by the query at hand, with respect to initially specified selectivity constraints; (ii) $relax^{+}/relax^{-}$: the distance of the returned tuples is too high or the non returned tuples are quite close to the initial query and thus can be returned.

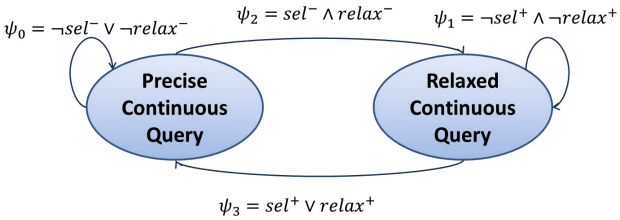

Fig. 1. An example of an ASAP automata

– Finally, the monitor component establishes whether a transition has to be performed, thus the query in execution has to be changed, in order to increase accuracy. This reasoning should rely on predicates computed by the assessor component and by the accuracy measure used to drive the process.

By varying the trade-off constraint, more emphasis can be given to either selection and precision, thus driving the process towards relaxed or precise computations.

4 ASAP in the Large

By increasing the complexity of the considered environment, moving towards highly distributed architectures, all concepts introduced in Section 3 become less clear. Indeed, while from one hand, user interactions with the network and its many applications generate a valuable amount of information, facts, and opinions with a great socio-economic potential, from the other hand, this huge wealth of information is currently being exploited much below its potential because of the difficulties in accessing data to retrieve relevant information. ASAP in the large can thus be interpreted as a step towards the realization of an entity-relationship search paradigm for uncontrolled and wide information domains, with an impact on qualitative and quantitative performance of systems for processing strongly interrelated and heterogeneous data in distributed dynamic environments. Under those new scenarios, data and requests can be characterized as follows.

Data Sources. Data from different sources are highly heterogeneous in terms of structure, semantic richness, and quality. They are often geo-referenced, time-variant, and dynamic. Information sources, which could be represented according to a graph-based data model, may contain: (i) strongly related and semantically complex but relatively static data (e.g., Linked Open Data); (ii) unstructured data, or data with a simple and defined structure; (iii) data dynamically generated by a multitude of diverse people (e.g., social networks, microblogs); (iv) highly dynamic data generated by public or private institutions linked to the territory (data streams).

Requests. Complex requests expressing relationships among the entities of user interest have to be represented, possibly relying on a graph-based query

language. Such requests are often vaguely specified, since users cannot reasonably know format and structure of data encoding the relevant information. For example, the user may ask for the nearest shops selling the book which her friend Luca likes or for the biography of the author of the painting she is watching.

The Targeted Problem. Processing complex requests on heterogeneous and dynamic information sources can be costly since it first requires a request interpretation, then processing has to be performed on available sources deemed relevant (to reduce processed data volumes), and finally results should be aggregated in a consistent answer and returned to the user. Additionally, the answer may not guarantee the user satisfaction since the request could have been incorrectly interpreted, processed on inaccurate, incomplete, unreliable data, or even it could have required a processing time inadequate to the urgency of the request. As pointed out in [21], one solution to these problems relies on user intervention. However, depending on the request urgency and the specific application scenario, such intervention may not always be possible. The problem thus arises to define approaches for providing approximate answers to shared and complex information needs, even vaguely and imprecisely specified, operating on the full spectrum of relevant content, overcoming the difficulties related to heterogeneity and dynamism while requiring a limited user involvement.

The ASAP Proposal. In [10], we claimed that the ASAP paradigm can be effectively used to tackle the problem described above. In order to limit user intervention, a specific type of QoD-oriented approximation has been proposed, namely *Wearable Queries* (WQs). WQs integrate explicit requests with profile (information provided by the user as well as induced by the system, e.g., user habits) and context (spatio-temporal coordinates of the request, its motivation, and its environment, e.g., in terms of potential interaction and urgency). The computed result should minimize the distance between the returned items and the specified context and user information. To this aim, WQs computation should take into account data specificities, with a particular reference to quality, geolocalization, and freshness of data sources and specific data items, in order to select relevant data sources and provide results at the appropriate level of detail.

To enable search in a huge space of highly heterogeneous and poorly controlled sources, an adaptive pay-as-you-go approach, influenced by quality, dynamics, and specificities of the considered sources, needs to be adopted. The devised approach, which constitutes a completely new QoD-oriented adaptive approach applied to QoD-oriented approximate searches (thus, a new ASAP approach), generates and incrementally refines mappings between sources, according to the requirements induced by the submitted requests, thus avoiding the prohibitive costs of full integration. The role of the proposed QoD-adaptive approach is therefore twofold: the space of sources is incrementally adapted to the peculiarities of the submitted requests and, simultaneously, requests, specified as WQs, are processed by incrementally adapting them to the peculiarities of the space of sources and its evolution over time.

In order to maximize accuracy during the adaptive process, we explore a yet unexplored coordinate, namely metadata corresponding to synthetic representations of similar WQ executions repeated over time (called Profiled Wearable Query Patterns - PWQPs). Similar requests are very common in dynamic contexts, each time information needs are widespread among different users, because induced by an event, the interests of a community, or a place (e.g., during or after an exceptional event -environmental emergencies or flash mobbing initiatives-). The ability to take advantage of the experience gained by prior processing in new searches allows response times and interpretation errors to be limited, thus reducing the possibility of producing unsatisfactory answers.

For processing WQs, we envisage an mechanism that moves at each step, in the large space of possible approximate answers, towards the sources deemed capable of producing the best solutions with respect to profile and context of the request, quality and dynamism of the sources and knowledge gained from previous executions. The process is incremental, i.e., first it attempts to exploit PWQPs, then makes a coarse-grain selection of sources, and later it focuses approximation efforts on the description of the selected sources. The results are composed or reconciled through mappings, selected or generated on-the-fly. The envisioned solution couples this mechanism with a method for assessing the quality of each individual processing. This information, together with any explicit user feedback, is used for updates and refinements.

5 Conclusions

In this paper, after revising and classifying approximate and adaptive processing techniques with respect to the quality parameters they take into account, we introduced ASAP techniques and we showed that they can be successfully used in both specific and more general application contexts, characterized by an higher complexity of the environment and of the data sources at hand. We remark that ASAP is not a new concept, rather, it can be interpreted as a revision of existing processing approaches focusing on QoD parameters, which could be effectively and efficiently used in emerging contexts. Several issues are still open and require further investigation, especially under the "ASAP in the large" vision. In this context, we are currently investigating issues concerning data source characterization, for Linked and crowdsourced (social) data, as well as automatic acquisition of approximate geo-spatial contexts for crowdsourced (social) data.

References

1. Amato, G., Rabitti, F., Savino, P., Zezula, P.: Region Proximity in Metric Spaces and its Use for Approximate Similarity Search. ACM Trans. Inf. Syst. 21(2), 192–227 (2003)
2. Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Motwani, R., Nishizawa, I., Srivastava, U., Thomas, D., Varma, R., Widom, J.: STREAM: The Stanford Stream Data Manager. IEEE Data Eng. Bull. 26(1), 19–26 (2003)

3. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. J. ACM 45(6), 891–923 (1998)
4. Babcock, B., Datar, M., Motwani, R.: Load Shedding for Aggregation Queries over Data Streams. In: ICDE, pp. 350–361 (2004)
5. Babu, S., Srivastava, U., Widom, J.: Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams. ACM Trans. Database Syst. 29(3), 545–580 (2004)
6. Belussi, A., Boucelma, O., Catania, B., Lassoued, Y., Podestà, P.: Towards Similarity-Based Topological Query Languages. In: Grust, T., et al. (eds.) EDBT 2006. LNCS, vol. 4254, pp. 675–686. Springer, Heidelberg (2006)
7. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: ICDE, pp. 421–430 (2001)
8. Catania, B., Guerrini, G.: Towards Adaptively Approximated Search in Distributed Architectures. In: Vakali, A., Jain, L.C. (eds.) New Directions in Web Data Management 1. SCI, vol. 331, pp. 171–212. Springer, Heidelberg (2011)
9. Catania, B., Guerrini, G.: Approximate queries with adaptive processing. In: Catania, B., Jain, L.C. (eds.) Advanced Query Processing, Volume 1: Issues and Trends. ISRL, vol. 36, pp. 237–269. Springer, Heidelberg (2013)
10. Catania, B., Guerrini, G., Belussi, A., Mandreoli, F., Martoglia, R., Penzo, W.: Wearable queries: adapting common retrieval needs to data and users. In: 7th International Workshop on Ranking in Databases (co-located with VLDB 2013), DBRank 2013, Riva del Garda, Italy, August 30, p. 7. ACM (2013)
11. Catania, B., Guerrini, G., Pinto, M.T., Podestà, P.: Towards relaxed selection and join queries over data streams. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS 2012. LNCS, vol. 7503, pp. 125–138. Springer, Heidelberg (2012)
12. Catania, B., Guerrini, G., Pomerano, D.: An adaptive approach for processing relaxed continuous queries (in preparation)
13. Chaudhuri, S., Das, G., Narasayya, V.R.: Optimized Stratified Sampling for Approximate Query Processing. ACM Trans. Database Syst. 32(2), 9 (2007)
14. Chaudhuri, S., Ganti, V., Kaushik, R.: A Primitive Operator for Similarity Joins in Data Cleaning. In: ICDE, p. 5 (2006)
15. Ilyas, I.F., Aref, W.G., Elmagarmid, A.K., Elmongui, H.G., Shah, R., Vitter, J.S.: Adaptive Rank-aware Query Optimization in Relational Databases. ACM Trans. Database Syst. 31(4), 1257–1304 (2006)
16. Ilyas, I.F., Beskales, G., Soliman, M.A.: A Survey of Top-k Query Processing Techniques in Relational Database Systems. ACM Comput. Surv. 40(4) (2008)
17. Koudas, N., Li, C., Tung, A.K.H., Vernica, R.: Relaxing Join and Selection Queries. In: VLDB, pp. 199–210 (2006)
18. Koudas, N., Srivastava, D.: Approximate Joins: Concepts and Techniques. In: VLDB, p. 1363 (2005)
19. Lengu, R., Missier, P., Fernandes, A.A.A., Guerrini, G., Mesiti, M.: Time-completeness Trade-offs in Record Linkage using Adaptive Query Processing. In: EDBT, pp. 851–861 (2009)
20. Marian, A., Amer-Yahia, S., Koudas, N., Srivastava, D.: Adaptive Processing of Top-k Queries in XML. In: ICDE, pp. 162–173 (2005)
21. Mass, Y., Ramanath, M., Sagiv, Y., Weikum, G.: IQ: The Case for Iterative Querying for Knowledge. In: Proc. of CIDR 2011, pp. 38–44 (2011)
22. Mishra, C., Koudas, N.: Interactive Query Refinement. In: EDBT, pp. 862–873 (2009)

23. Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G.S., Olston, C., Rosenstein, J., Varma, R.: Query Processing, Approximation, and Resource Management in a Data Stream Management System. In: CIDR (2003)
24. Olston, C., Jiang, J., Widom, J.: Adaptive Filters for Continuous Queries over Distributed Data Streams. In: SIGMOD Conference, pp. 563–574 (2003)
25. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
26. Silva, Y.N., Aref, W.G., Ali, M.H.: The similarity join database operator. In: ICDE, pp. 892–903 (2010)
27. Spiegel, J., Polyzotis, N.: TuG Synopses for Approximate Query Answering. ACM Trans. Database Syst. 34(1) (2009)
28. Tatbul, N., Çetintemel, U., Zdonik, S.B., Cherniack, M., Stonebraker, M.: Load Shedding in a Data Stream Manager. In: VLDB, pp. 309–320 (2003)
29. Zhou, X., Gaugaz, J., Balke, W.-T., Nejdl, W.: Query Relaxation using Malleable Schemas. In: SIGMOD Conference, pp. 545–556 (2007)