Sampling Methods In Approximate Query Answering Systems

Gautam Das
Department of Computer Science and Engineering
The University of Texas at Arlington
Box 19015
416 Yates St.
Room 300, Nedderman Hall
Arlington, TX 76019-0015

Telephone: 817.272.7595
Fax: 817.272.3784
email: gdas@cse.uta.edu

Sampling Methods In Approximate Query Answering Systems

Gautam Das

Department of Computer Science and Engineering

The University of Texas at Arlington

INTRODUCTION

In recent years, advances in data collection and management technologies have led to a proliferation of very large databases. These large data repositories are typically created in the hope that through analysis, such as data mining and decision support, they will yield new insights into the data and the real-world processes that created it. In practice, however, while the collection and storage of massive data sets has become relatively straightforward, effective data analysis has proven more difficult to achieve. One reason that data analysis successes have proven elusive is that most analysis queries, by their nature, require aggregation or summarization of large portions of the data being analyzed. For multi-gigabyte data repositories, this means that processing even a single analysis query involves accessing enormous amounts of data, leading to prohibitively expensive running

times. This severely limits the feasibility of many types of analysis applications, especially those that depend on timeliness or interactivity.

While keeping query response times short is very important in many data mining and decision support applications, exactness in query results is frequently less important. In many cases, "ballpark estimates" are adequate to provide the desired insights about the data, at least in preliminary phases of analysis. For example, knowing the marginal data distributions for each attribute up to 10% error will often be enough to identify top-selling products in a sales database or to determine the best attribute to use at the root of a decision tree.

For example, consider the following SQL query:

```
SELECT State, COUNT (*) as ItemCount
FROM SalesData
WHERE   ProductName = 'Lawn Mower'
GROUP BY State
ORDER BY ItemCount DESC
```

This query seeks to compute the total number of a particular item sold in a sales database, grouped by state. Instead of a time-consuming process that produces completely accurate answers, in some circumstances it may be suitable to produce ball-park estimates, e.g. counts to the nearest thousands.

The acceptability of inexact query answers coupled with the necessity for fast query response times has led researchers to investigate *approximate query answering techniques* (AQA) that sacrifice accuracy to improve running time, typically through some sort of lossy data compression. The general rubric in which most approximate query processing systems operate is as follows: first, during the "pre-processing phase", some auxiliary data structures, or data *synopses*, are built

over the database; then, during the "runtime phase", queries are issued to the system and approximate query answers are quickly returned using the data synopses built during the pre-processing phase. The quality of an approximate query processing system is often determined by how accurately the synopsis represents the original data distribution, how practical it is to modify existing database systems to incorporate approximate query answering, and whether error estimates can be returned in addition to ballpark estimates.

BACKGROUND

Figure 1 describes a general architecture for most AQA systems. There are two components in the architecture: (1) a component for building the synopses from database relations, and (2) a component that rewrites an incoming query to use the synopses to answer the query approximately and report the answer with an estimate of the error in the answer.

The different approximate query answering systems that have been proposed differ in various ways: in the types of synopses proposed, whether the synopses building component is executed in a preprocessing phase or whether it executes at runtime, the ability of the AQA system to also provide error guarantees in addition to the approximate answers, and finally (from a practical point of view, perhaps the most important), the amount of changes necessary to query processing engines of commercial database management systems to incorporate approximate query answering.

The types of synopses developed for AQA systems can be divided into two broad groups: *sampling* based approaches and *non-sampling* based approaches. In sampling based approaches, a small random sample of the rows of the original

database table is prepared, and queries are directed against this small sample table. The non-sampling based approaches encompass a wide variety of techniques, e.g., sophisticated data structures such as *wavelets* (Chakrabarti, Garofalakis, Rastogi & Shim (2001), Matias, Vitter & Wang (1998)) and *histograms* (Ioannidis & Poosala (1999)) have been proposed as useful tools for AQA.

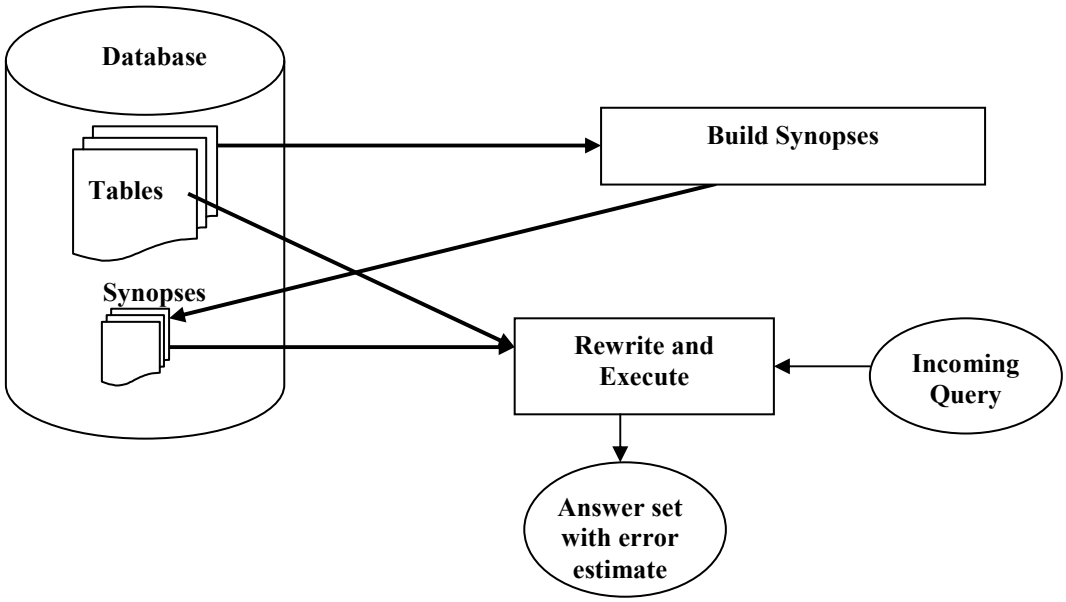

Figure 1: Architecture for Approximate Query Answering

Work in non-sampling based AQA techniques is of great theoretical interest, but their practical impact is often limited by the extensive modifications to query processors and query optimizers that are often needed to make use of these technologies. On the other hand, sampling-based systems have the advantage that they can be implemented as a thin layer of middleware which re-writes queries to run against sample tables stored as ordinary relations in a standard, off-the-shelf database server.

Partly for these reasons, sampling-based systems have in recent years been the most heavily studied type of AQA system. In the rest of this chapter our focus is on presenting an overview of the latest developments in sampling-based AQA techniques.

MAIN THRUST OF THE CHAPTER

In the following section, we summarize the various sampling-based AQA technologies that have been proposed in recent years by the research community. The focus of this chapter in on approximately answering standard SQL queries on relational databases; other exciting work done on approximate query processing in other scenarios such as streaming and time series data is beyond the scope of this chapter.

We assume a standard data warehouse schema, consisting of a few fact tables containing the measure columns, connected to several dimension tables via foreign key relationships. Furthermore, we assume that our queries are aggregation queries with SUM, COUNT and GROUP BY operators, either over a single fact table, or over a fact table joined to several dimension tables.

Uniform Random Sampling

The essential idea is that a small pre-computed uniform random sample of rows S of the database R often well-represents the entire database. For a fast approximate answer at runtime, one has to simply execute the query on S and "scale" the result. Thus, if S is a 1% sample of the database, the scaling factor is 100. The main advantages of uniform random sampling are simplicity, and efficiency of pre-

processing. However, there are several critical disadvantages which have not allowed this approach to be considered seriously for AQA systems.

One disadvantage is the well-known statistical problem of *large data variance*. For example, suppose we wish to estimate the average salaries of a particular corporation. Uniform random sampling does badly if the salary distribution is highly skewed.

The other disadvantage is specific to database systems, and is the *low selectivity* problem. For example, suppose a query wishes to find the average salary of a small department of a large corporation. If we only had a uniform random sample of the entire database, then it is quite likely that this small department may not be adequately represented, leading to large errors in the estimated average salary.

To mitigate these problems, much research has been attempted using so-called *biased sampling* techniques, where a non-uniform random sample is pre-computed such that parts of the database deemed "more important" than the rest are better represented in the sample. We discuss such techniques later in the chapter.

Online Aggregation

Hellerstein, Haas & Wang (1997) describe techniques for *online aggregation* in which approximate answers for queries are produced during early stages of query processing and gradually refined until all the data has been processed. This framework is extended in Raman & Hellerstein (2002) to have the query processor gives precedence to tuples that contribute to "higher-priority" parts of the query result, where priority is defined using a user-specified function. The online aggregation approach has some compelling advantages, e.g., it does not require pre-processing, and it allows progressive refinement of approximate answers at runtime

until the user is satisfied or the exact answer is supplied, and it can provide confidence intervals that indicate the uncertainty present in the answer.

However, there are two important systems considerations that represent practical obstacles to the integration of online aggregation into conventional database systems. First, stored relations are frequently clustered by some attribute, so accessing tuples in a random order as required for online aggregation requires (slow) random disk accesses. Second, online aggregation necessitates significant changes to the query processor of the database system. This is impractical, as it is desirable for an AQA system to leverage today's commercial query processing systems with minimal changes to the greatest degree possible.

We next consider several biased-sampling AQA methods that are based on pre-computing the samples. Towards the end, we also discuss a method that attempts to strike a balance between online and pre-computed sampling.

Icicles

Recognizing the low selectivity problem, designing a biased sample that is based on known *workload* information was attempted in Ganti, Lee & Ramakrishnan (2000). In this paper, the assumption was that a workload of queries, i.e., a log of all recent queries executing against the database, is a good predictor of the queries that are yet to execute on the database in the future. Thus, for example, if a query requests for the average salary of a small department in a large corporation, it is assumed that such (or similar) queries will repeat in the future. A heuristic pre-computation procedure called *Icicles* was developed in which tuples that have been accessed by many queries in the workload were assigned greater probabilities of being selected into the sample.

While this was an interesting idea based on biased sampling that leverages workload information, a disadvantage was that it focuses only on the low selectivity problem, and furthermore the suggested solution is rather heuristical.

Outlier Indexing

The first paper that attempted to address the problem of large data variance was by Chaudhuri, Das, Datar, Motwani & Narasayya (2001). It proposes a technique called *Outlier Indexing* for improving sampling-based approximations for aggregate queries when the attribute being aggregated has a skewed distribution.

The basic idea is that *outliers* of the data (i.e. the records that contribute to high variance in the aggregate column) are collected into a separate index, while the remaining data is sampled using a biased sampling technique. Queries are answered by running them against both the outlier index as well as the biased sample, and an estimated answer is composed out of both results. A disadvantage of this approach was that the primary emphasis was on the data variance problem, and while the authors did propose a hybrid solution for both the data variance as well as the low selectivity problem, the proposed solution was heuristical and therefore, suboptimal.

Congressional Sampling

The AQUA project at Bell Labs (Acharya, Gibbons & Poosala (1999)) developed a sampling-based system for approximate query answering. Techniques used in AQUA included *congressional sampling* (Acharya, Gibbons & Poosala (2000)), which is targeted towards answering a class of common and useful analysis queries: group by queries with aggregation. Their approach stratifies the database by considering the set of queries involving all possible combinations of grouping columns, and produces a weighted sample that balances the approximation errors of

these queries. However, their approach is still ad-hoc in the sense that even though they try to reduce the error, their scheme does not minimize the error for any of the well-known error metrics.

Join Synopses

The AQUA project at Bell Labs also developed the *join synopses* technique (Acharya, Gibbons, Poosala & Ramaswamy (1999)), which allow approximate answers to be provided for certain types of join queries, in particular foreign-key joins. The technique involved pre-computing the join of samples of fact tables with dimension tables so that at runtime queries only need to be executed against single (widened) sample tables. This is an alternate to the approach of only pre-computing samples of fact tables and having to join these sample tables with dimension tables at runtime.

We mention that the problem of sampling over joins that are not foreign-key joins is a difficult problem, and under certain conditions, is essentially not possible (Chaudhuri, Motwani & Narasayya (1999)). Thus, approximate query answering does not extend to queries that involve non-foreign key joins.

Stratified Sampling (STRAT)

The paper by Chaudhuri, Das & Narasayya (2001) sought to overcome many of the limitations of the previous works on pre-computed sampling for approximate query answering, and proposed a method called STRAT for approximate query answering.

Unlike most previous sampling-based studies that used ad-hoc randomization methods, the authors here formulated the problem of pre-computing a sample as an optimization problem, whose goal is to minimize the error for the given workload. They also introduced a generalized model of the workload ("lifted workload") that

makes it possible to tune the selection of the sample so that approximate query processing using the sample is effective not only for workloads that are exactly identical to the given workload, but also for workloads that are "similar" to the given workload (i.e., queries that select regions of the data that overlap significantly with the data accessed by the queries in the given workload) – a more realistic scenario. The degree of similarity can be specified as part of the user/database administrator preference. They formulate selection of the sample for such a lifted workload as a *stratified sampling* task with the goal to minimize error in estimation of aggregates. The benefits of this systematic approach are demonstrated by theoretical results (where it is shown to subsume much of the previous work on pre-computed sampling methods for AQA), as well as experimental results on synthetic data as well as real enterprise data-warehouses.

Dynamic Sample Selection

A sampling technique that attempts to strike a middle ground between pre-computed and online sampling is *dynamic sample selection* (Babcock, Chaudhuri & Das (2003)).

The requirement for fast answers during the runtime phase means that scanning a large amount of data to answer a query is not possible, or else the running time would be unacceptably large. Thus, most sampling-based approximate query answering schemes have restricted themselves to building only a small sample of the data. However, because relatively large running times and space usage during the pre-processing phase are generally acceptable as long as the time and space consumed are not exorbitant, nothing prevents us from *scanning* or *storing* significantly larger amounts of data during pre-processing than we are able to access at runtime. Of course, because we are only able to access a small amount of

stored data at runtime, there is no gain to be had from building large auxiliary data structures unless they are accompanied by some indexing technique that allows us to decide, for a given query, which (small) portion of the data structures should be accessed to produce the most accurate approximate query answer.

In Babcock, Chaudhuri & Das (2003), the authors describe a general system architecture for approximate query processing that is based on the dynamic sample selection technique. The basic idea is to construct during the pre-processing phase a random sample containing a large number of differently biased sub-samples, and then, for each query that arrives during the runtime phase, to select an appropriate small subset from the sample that can be used to give a highly accurate approximate answer to the query. The philosophy behind dynamic sample selection is to accept greater disk usage for summary structures than other sampling based AQA methods in order to increase accuracy in query responses while holding query response time constant (or alternatively, to reduce query response time while holding accuracy constant). The belief is that for many AQA applications, response time and accuracy are more important considerations than disk usage.

FUTURE TRENDS

In one sense, AQA systems are not new. These methods have been used internally used by query optimizers of database systems for selectivity estimation for a long time. However, approximate query answering has not yet been externalized to the end user by major vendors, though sampling operators are appearing in commercial database management systems. Research prototypes exist in the industry, e.g., AQP from Microsoft Research and the AQUA system from Bell Labs.

From a research potential viewpoint, approximate query answering promises to be a very fertile area with several deep and unresolved problems. Currently there is a big gap between the development of algorithms and their adaptability in real systems. This gap needs to be addressed before AQA techniques can be embraced by the industry. Secondly, the research has to broaden beyond the narrow confines of aggregation queries over single table databases or multi-tables involving only foreign-key joins. It is important to investigate how to return approximations to set-valued results, AQA over multi-table databases with more general types of SQL queries, AQA over data streams, as well investigations into the practicality of other non-sampling based approaches to approximate query answering. As data repositories get larger and larger, effective data analysis will prove increasingly harder to accomplish.

CONCLUSIONS

In this chapter we discussed the problem of approximate query answering in database systems, especially in decision support applications. We described various approaches takes to design approximate query answering systems, especially focusing on sampling based approaches. We believe that approximate query answering is an extremely important problem for the future, and much work needs to be done before practical systems can be built that leverage the substantial theoretical developments already accomplished in the field.

REFERENCES

Acharya S., Gibbons P. B. & Poosala V. (1999). Aqua: A Fast Decision Support System Using Approximate Query Answers. *Proceedings of International Conference on Very Large Databases*, 754-755.

Acharya S., Gibbons P. B. & Poosala V. (2000). Congressional Samples for Approximate Answering of Group-By Queries. *Proceedings of Special Interest Group on Management of Data*, 487-498.

Acharya S., Gibbons P. B., Poosala V. & Ramaswamy S. (1999). Join Synopses for Approximate Query Answering. *Proceedings of Special Interest Group on Management of Data*, 275-286.

Chaudhuri S., Das G. & Narasayya V. (2001). A Robust, Optimization-Based Approach for Approximate Answering of Aggregate Queries. *Proceedings of Special Interest Group on Management of Data*, 295-306.

Babcock B., Chaudhuri S. & Das G. (2003). Dynamic Sample Selection for Approximate Query Processing. *Proceedings of Special Interest Group on Management of Data*, 589-550.

Chaudhuri S., Das G., Datar M., Motwani R. & Narasayya V. (2001). Overcoming Limitations of Sampling for Aggregation Queries. *Proceedings of International Conference on Data Engineering*, 534-542.

Chaudhuri S., Motwani R. & Narasayya V. (1999). On Random Sampling over Joins. *Proceedings of Special Interest Group on Management of Data*, 263-274.

Chakrabarti K., Garofalakis M. N., Rastogi R. & Shim K. (2001). Approximate Query Processing Using Wavelets. *Proceedings of International Conference on Very Large Databases*, 111-122.

Ganti V., Lee M. & Ramakrishnan R. (2000). ICICLES: Self-Tuning Samples for Approximate Query Answering. *Proceedings of International Conference on Very Large Databases*, 176-187.

Hellerstein J. M., Haas P. J. & Wang H. (1997). Online Aggregation. *Proceedings of Special Interest Group on Management of Data*, 171-182.

Ioannidis Y. E. & Poosala V. (1999). Histogram-Based Approximation of Set-Valued Query-Answers. *Proceedings of International Conference on Very Large Databases*, 174-185.

Matias Y., Vitter J. S. & Wang M. (1998). Wavelet-Based Histograms for Selectivity Estimation. *Proceedings of Special Interest Group on Management of Data*, 448-459.

Raman V. & Hellerstein J. M. (2002). Partial Results for Online Query Processing. *Proceedings of Special Interest Group on Management of Data*, 275-286.

TERMS AND DEFINITIONS

Decision Support Systems: Typically business applications that analyze large amounts of data in warehouses, often for the purpose of strategic decision making.

Aggregation Queries: Common queries executed by decision support systems that aggregate and group large amounts of data, where aggregation operators are typically SUM, COUNT, AVG, etc.

Uniform Sampling: A random sample of k tuples of a database where each subset of k tuples is equally likely to be the sample.

Histograms: Typically used for representing 1-dimensional data, though multi-dimensional histograms are being researched in the database field. A histogram is a division of the domain of a 1-dimensional ordered attribute into buckets, where each bucket is represented by a contiguous interval along the domain, along with the count of the number of tuples contained withing this interval and other statistics.

Biased Sampling: A random sample of k tuples of a database where the probability of a tuple belonging to the sample varies across tuples.

Stratified Sampling: A specific procedure for biased sampling, where the database is partitioned into different strata, and each stratum is uniformly sampled at different sampling rates. Tuples that are "more important" for aggregation purposes, such as outliers, are put into strata that are then sampled at a higher rate.

Standard Error: The standard deviation of the sampling distribution of a statistic. In the case of approximate query answering, it measures the expected value of the error in the approximation of aggregation queries.

Workload: The log of all queries that execute on a database system. Workloads are often used by database administrators as well as by automated systems (such as AQA systems) to tune various parameters of database systems for optimal performance, such as indexes and physical design, and in the case of AQA, the set of sample tables.