Sampling Methods in Approximate Query Answering Systems

Gautam Das
The University of Texas at Arlington, USA

INTRODUCTION

In recent years, advances in data collection and management technologies have led to a proliferation of very large databases. These large data repositories typically are 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 them. In practice, however, while the collection and storage of massive datasets 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 often will 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= 'LawnMower'
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 ballpark estimates (e.g., counts to the nearest thousands).

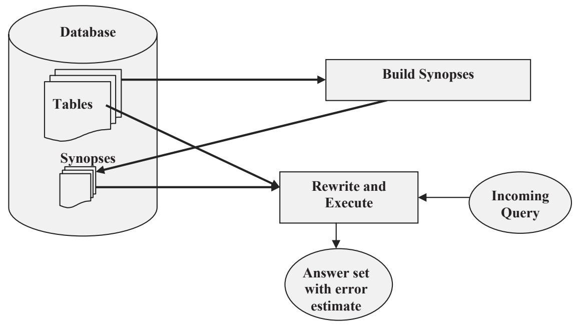
The acceptability of inexact query answers, coupled with the necessity for fast query response times, has led researchers to investigate approximate query answering (AQA) techniques that sacrifice accuracy to improve running time, typically through some sort of lossy data compression. The general rubric in which most approximate query processing systems operate is as follows: first, during the preprocessing 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 quickly are returned, using the data synopses built during the preprocessing phase. The quality of an approximate query processing system often is 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 in order 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

Figure 1. Architecture for approximate query answering



the AQA system also to provide error guarantees in addition to the approximate answers; and, finally (from a practical point of view and 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 et al., 2001; Matias, Vitter & Wang, 1998] and histograms [Ioannidis & Poosala, 1999]) have been proposed as useful tools for AQA.

Work in non-sampling-based AQA techniques is of great theoretical interest, but its practical impact often is limited by the extensive modifications to query processors and query optimizers that often are 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 that rewrites 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 article, our focus is on presenting an overview of the latest developments in sampling-based AQA techniques.

MAIN THRUST

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 article is 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 article.

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 precomputed uniform random sample of rows S of the database R often represents the entire database well. For a fast approximate answer at runtime, one simply has to 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 preprocessing. However, there are several critical disadvantages that 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 precomputed, such that parts of the database deemed more important