Join Synopses for Approximate Query Answering

Swarup Acharya Phillip B. Gibbons Viswanath Poosala Sridhar Ramaswamy

Information Sciences Research Center
Bell Laboratories
600 Mountain Avenue
Murray Hill NJ 07974

November 5, 1998

Abstract

In large data warehousing environments, it is often advantageous to provide fast, approximate answers to complex aggregate queries based on statistical summaries of the full data. In this paper, we demonstrate the difficulty of providing good approximate answers for join-queries using only statistics (in particular, samples) from the base relations. We propose *join synopses* (join samples) as an effective solution for this problem and show how precomputing just *one join synopsis* for each relation suffices to significantly improve the quality of approximate answers for arbitrary queries with foreign key joins. We present optimal strategies for allocating the available space among the various join synopses when the query work load is known and identify heuristics for the common case when the work load is not known. We also present efficient algorithms for incrementally maintaining join synopses in the presence of updates to the base relations. One of our key contributions is a detailed analysis of the error bounds obtained for approximate answers that demonstrates the trade-offs in various methods, as well as the advantages in certain scenarios of a new subsampling method we propose. Our extensive set of experiments on the TPC-D benchmark database show the effectiveness of join synopses and various other techniques proposed in this paper.

1 Introduction

Traditional query processing has focused solely on providing exact answers to queries, in a manner that seeks to minimize response time and maximize throughput. However, in large data recording and warehousing environments, providing an exact answer to a complex query can take minutes, or even hours, due to the amount of computation and disk I/O required.

There are a number of scenarios in which an exact answer may not be required, and a user may prefer a fast, approximate answer. For example, during some drill-down query sequences in ad-hoc data mining, initial queries in the sequence are used solely to determine what the interesting queries are [HHW97]. An approximate answer can also provide feedback on how well-posed a query is. Moreover, it can provide a tentative answer to a query when the base data is unavailable. Another example is when the query requests numerical answers, and the full precision of the exact answer is not needed, e.g., a total, average, or percentage for which only the first few digits of precision are of interest (such as the leading few digits of a total in the millions, or the nearest percentile of a percentage). Finally, note that techniques for fast approximate answers can also be used in a more traditional role within the query optimizer to estimate plan costs; such an application demands very fast response times but not exact answers.

Motivated by the above reasoning, we study the issue of providing approximate answers to queries in this paper. Our work is tailored to data warehousing environments. In particular, we focus on providing *approximate answers to* *aggregate queries on joins of relations.*[1] Our goal is provide an estimated response in orders of magnitude less time than the time to compute an exact answer, by avoiding or minimizing the number of accesses to the base data.

While there has been a flurry of recent work in approximate query answering (e.g., [VL93, BDF$^+$97, GMP97a, GMP97b, HHW97, GM98a]), only the work by Hellerstein *et al* [HHW97] has looked at the problem of approximate join aggregates. We consider this to be an important problem since most non-trivial queries, especially on data warehousing schemas, involve joining two or more tables. For example, 13 of the 17 queries in the TPC-D benchmark involve queries on joins.

We show, both theoretically and empirically, that schemes for providing approximate join aggregates that rely on using random samples of base relations alone suffer from serious disadvantages. (This is discussed in detail in Sections 2 and 3.) Instead, we propose the use of *precomputed samples of a small set of distinguished joins* —we refer to these precomputed samples as join synopses—in order to compute approximate join aggregates. Our key contribution is to show that for database schemas with only foreign key joins— these schemas are the ones typically used in data warehousing—*it is possible to provide good quality approximate join aggregates using a very small number of join synopses.*[2] An important issue arising out of the use of several sets of statistics is the careful allocation of a limited amount of space among them. When a query workload characterization is available, we show how to design an optimal allocation for join synopses that minimizes the overall error in the approximate answers computed. We discuss heuristic allocation strategies that work well when the workload is not known. We also show how join synopses can be maintained in the presence of updates, and show that the overhead that they impose is very minimal.

An important issue in computing approximate answers is that of providing confidence bounds for the answers. Such bounds give the user valuable feedback on how reliable an answer is. In addition to discussing how traditional methods for providing confidence bounds (for example, based on Hoeffding bounds or the Central Limit Theorem [Haa97]) apply to join synopses, we propose a novel empirical technique for computing confidence bounds based on extracting subsamples from samples.

The contributions of this paper are as follows:

- We propose join synopses as a technique for computing approximate join aggregates. Our solution is applicable to data warehousing environments, which have schemas that involve only foreign key joins.
- We present an optimal allocation strategy for join synopses when the query workload is known. We propose allocation heuristics for the case where the query workload is unknown.
- We discuss the issue of generating confidence bounds for approximate join aggregates and propose a new empirical technique for computing error bounds.
- We present an efficient maintenance algorithm for join synopses.
- We present the results of a detailed experimental study on the performance of the techniques we propose. Using the TPC-D benchmark, we show the advantages of join synopses over samples of base relations in computing approximate join aggregates with good confidence bounds. We also show that join synopses can be maintained efficiently and with minimal overheads.

[1]We abbreviate this phrase to "approximate join aggregates."

[2]It is, of course, not possible to provide good approximate answers to queries of arbitrary low selectivity with a fixed amount of storage. Our goal is to show that for a given amount of space, join synopses work better than schemes that use only samples of base relations.

The research in this paper was conducted as part of the Aqua project [GMP97a] at Bell Labs. The goal of the Aqua project is to develop an approximate query answering engine. The engine has been designed to run on top of any commercial DBMS. The engine uses the DBMS to store synopses of the original data and provides approximate answers via query rewriting. This approximate query answering engine, which we are in the process of implementing, is based partly on the techniques in this paper.

The rest of this paper is organized as follows. Section 2 discusses related work in the area of approximate query processing. Section 3 discusses why samples of base samples are inadequate for computing approximate join aggregates. Section 4 presents join synopses as a solution to this problem when the database schema has only foreign key joins. Section 5 discusses allocation strategies for join synopses. Section 6 discusses the issue of confidence bounds for approximate aggregates and presents a novel empirical solution for computing confidence bounds. Section 7 describes an algorithm for maintaining join synopses under insertions and deletions to the database. This is followed by our experimental study of the different techniques in Section 8. We conclude in Section 9.

2 Related Work

Statistical techniques have been applied in databases for more than two decades now, but primarily inside a query optimizer for selectivity estimation [SAC$^+$79]. However, the application of statistical techniques to approximate query answering has started receiving attention only very recently. Below, we describe the work on approximate query answering and the work on general statistical techniques applied in databases.

Approximate query answering: Hellerstein *et al* [HHW97] proposed a framework for approximate answers of aggregation queries called *online aggregation*, in which the base data is scanned in random order at query time and the approximate answer is continuously updated as the scan proceeds. Unlike Aqua, this work involves accessing original data at query time, thus being more costly, but at the same time, this approach provides an option to get the fully accurate answer gradually and it is not affected by database updates. However, the problems with join queries discussed in this paper also apply to online aggregation – basically, a large fraction of the data needs to be processed before the errors become tolerable. Other systems support limited on-line aggregation features; e.g., the Red Brick system supports running COUNT, AVG, and SUM (see [HHW97]). Since the scan order used to produce these aggregations is not random, the accuracy can be quite poor. There have been several recent works on "fast-first" query processing, whose goal is to quickly provide a few tuples of the actual query answer [BM96, CK98, AZ96], but the focus there is not on obtaining statistically representative approximate answers. In the APPROXIMATE query processor, developed by Vrbsky and Liu [VL93], an approximate answer to a set-valued query is any superset of the exact answer that is a subset of the cartesian product. The query processor uses various class hierarchies to iteratively fetch blocks relevant to the answer, producing tuples certain to be in the answer while narrowing the possible classes that contain the answer. Clearly, this work is quite different from the statistical approach taken by us and by Hellerstein *et al.* Finally, Matias *et al* [MVN93, MVY94, MSY96] proposed and studied *approximate data structures*, for providing faster (approximate) answers to data structure queries. For example, an approximate priority queue returns a fast, approximate min in response to an extract-min query.

Statistical techniques: There has been considerable amount of work in developing statistical techniques to solve selectivity estimation and more recently, for data reduction in large data warehouses. The three major classes of techniques used are *sampling* (e.g., [HÖT88, LNS90, HNS94, LN95, HNSS95, GGMS96]), *histograms* (e.g., [Koo80, PIHS96, Poo97]), and parametric modeling (e.g., [CR94]). A survey of various statistical techniques is given in the paper by Barbará *et al* [BDF+97]. Gibbons and Matias present a framework for studying synopsis data structures for massive data sets [GM98c] and introduced two sampling-based synopses, *concise samples and counting samples*, that can be used to obtain larger samples for the same space and to improve approximate query answers for hot list queries [GM98a]. Maintenance algorithms exist for samples [OR92, GMP97b, GM98a] and histograms [GMP97b]. However, these maintenance techniques are applicable only to "base" statistics and not to the join synopses presented in this paper.

3 The Problem with Joins

As pointed out in the introduction, an approximate query answering system has two key requirements — providing an accurate estimate of the actual answer and providing tight bounds on the confidence of the estimate. Unfortunately, both the accuracy of the estimate and the spread of confidence bounds are strongly dependent on the size of the sample used to derive them. Moreover, unless some statistical properties can be guaranteed on this sample, the bounds are usually very pessimistic. In this section, we will motivate why using a straightforward sampling approach can produce a poor estimation quality when approximating aggregates on multi-way joins.

A natural set of synopses for an approximate query engine would include uniform random samples of each base relation (i.e., relations in the database). We refer to these as *base samples*. The use of base samples to estimate the output of a multi-way join, however, can produce a poor quality approximation. This is for the following two reasons:

1. **Statistical guarantee:** In general, the join of two uniform random base samples is *not a uniform random sample of the output of the join*. In most cases, the non-uniformity introduced by the join significantly degrades the accuracy of the answer and the confidence bounds.

2. **Join output size:** The join of two random samples typically has very few tuples, even when the actual join selectivity is fairly high. This can lead to both inaccurate answers and very poor confidence bounds since they critically depend on the query result size.

Consider the first problem. In order for the join of the base samples to be a uniform random sample of the actual join, the probability of any two joined tuples to be in the former should be the same as their probability in the latter. (This is a necessary, but not sufficient, condition.) We will use a simple 2-way join counter example to show this is not always the case.

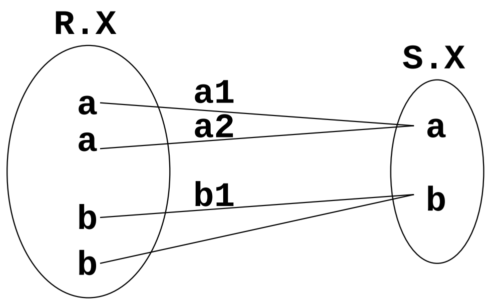


Figure 1: Join of samples is *not* a sample of joins

Consider the (equality) join of two relations R and S on an attribute X. The distribution of X values in the two relations are given in Figure 1. The edges connect joining tuples. Consider joining base samples from R and S. Assume that each tuple is selected for a base sample with probability $1/r$. From Figure 1, we see that $a1$ and $a2$ are in the join if and only if both a tuples are selected from R and the one a tuple is selected from S. This occurs with probability $1/r^3$, since there are three tuples that must be selected. On the other hand, $a1$ and $b1$ are in the join if and only if the *four* tuples incident to these edges are selected. This occurs with probability only $1/r^4$. This contrasts with the fact that in a uniform random sample of the actual join, the probability that both $a1$ and $a2$ are selected *equals* the probability that both $a1$ and $b1$ are selected. In general, for any pair of relations joining on an attribute X, any X value that occurs in each relation, and occurs more than once in at least one of the relations, introduces a bias such that the join of the base samples is *not* a uniform random sample of the output of the join.

We now highlight the second problem of small output sizes. Consider two relations, A and B, and base samples comprising of 1% of each relation. The size of the foreign key join between A and B is equal to the size of A. However the expected size of the join of the base samples is .01% of the size of A, since for each tuple in A, there is only one tuple in B that joins with it, and that tuple is in the sample for B with only a 1% probability. In general, consider a k-way foreign key join and k base samples each comprising $1/r$ of the tuples in their respective base relations. Then the expected size of the join of the base samples is $1/r^k$ of the size of the actual join. In fact the best known confidence interval bounds for approximate join aggregates based on base samples are quite pessimistic [Haa97].

Thus, it is in general impossible to produce good quality approximate answers using samples on the base relations alone, a fact that we further demonstrate in our experiments. Since nearly all queries in the warehousing context involve complex queries with large number of (foreign-key) joins, it is critical to solve this problem. In the next section we provide a solution for this problem and discuss its ramifications.

4 Join Synopses

In this section we present a practical and effective solution for producing approximate join aggregates of good quality. At a high level, we propose to precompute samples of join results, making quality answers possible even on complex joins. A naive way to precompute such samples is to execute all possible join queries of interest and collect samples of their results. However, this is not feasible since it is too expensive to compute and maintain. Our main contribution is to show that by computing samples of the results of a *small set of distinguished joins*, we can obtain random samples of all possible joins in the schema. We refer to samples of these distinguished joins as join synopses. Our technique works for the star and snowflake schemas (more precisely, acyclic schemas with only foreign key joins[3]) typically found in data warehousing [Sch97].

In order to develop this solution, we model the database schema by a graph whose nodes correspond to relations and whose edges correspond to every possible 2-way foreign key join for the schema. The key result we prove is that there is a one-one correspondence between a tuple in a relation r and a tuple in the output of *any* foreign key join involving r and the relations corresponding to one or more of its descendants in the graph. This provides us with the technical tool for join synopses: a sample S_r of a relation r can be used to produce another relation $\mathcal{J}(S_r)$—called

[3]A 2-way join $r_1 \bowtie r_2$, $r_1 \neq r_2$, is a *foreign key join* if the join attribute is a foreign key in r_1 (i.e., a key in r_2). For $k \geq 3$, a k-way join is a *k-way foreign key join* if there is an ordering $r_1, r_2, \ldots, r_k$ of the relations being joined that satisfies the following property: for $i = 2, 3, \ldots, k$, $s_{i-1} \bowtie r_i$ is a 2-way foreign key join, where s_{i-1} is the relation obtained by joining $r_1, r_2, \ldots, r_{i-1}$.

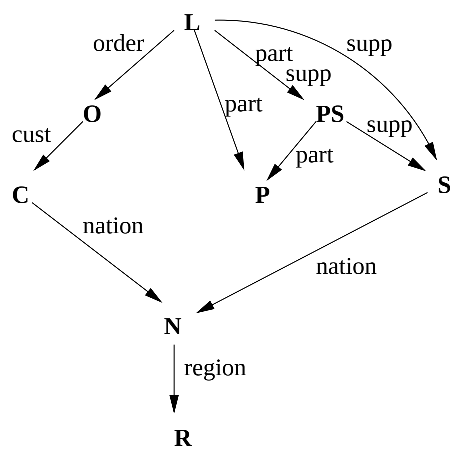


Figure 2: Directed graph for the TPC-D schema.

a join synopsis of r—that can be used to provide random samples of *any join involving r and one or more of its descendants*.

We now move to the technical development of the results. Consider a directed acyclic graph, G, with a vertex for each base relation and a directed edge from a vertex u to a vertex $v \neq u$ if there are one or more attributes in u's relation that constitute a foreign key for v's relation. The edge is labeled with the foreign key. An example is given in Figure 2 for the TPC-D benchmark. We show two key lemmas about the properties of this graph.

Lemma 4.1 *The subgraph of G on the k nodes in any k-way foreign key join must be a connected subgraph with a single root node.*

Proof. Consider an ordering $r_1, \ldots, r_k$ on the relations that satisfies the k-way foreign key join property given above. The proof is by induction, with the base case of a single node r_1. Let $1 < i \leq k$ and $s_{i-1} = r_1 \bowtie \cdots \bowtie r_{i-1}$. Assume that the subgraph G_{i-1} on the $i-1$ nodes in s_{i-1} is connected with a single root node r_1. Since $s_{i-1} \bowtie r_i$ is a 2-way foreign key join, the join attribute must be a key in r_i. Thus there is an edge directed from some node in G_{i-1} to r_i, implying that $G_i = G_{i-1} \cup r_i$ is a connected subgraph of G. Hence there is a directed path in G from r_1 to r_i. Since G is acyclic, $r_i \neq r_1$, so r_1, which by the inductive assumption is the only root node in G_{i-1}, is the only root node of G_i. The lemma follows by induction. ∎

We denote the relation corresponding to the root node as the *source* relation for the k-way foreign key join.

Lemma 4.2 *There is a 1-1 correspondence between tuples in a relation r_1 and tuples in any k-way foreign key join with source relation r_1.*

Proof. By the definition of a join, for each tuple τ in the output of a join, there exists a tuple τ' in r_1 such that τ projected on the attributes in r_1 is τ'. Conversely, we claim that for each tuple τ' in r_1 there is exactly one tuple τ in the k-way foreign key join. The claim is shown by induction. Consider an ordering $r_1, \ldots, r_k$ on the relations that satisfies the k-way foreign key join property given above. The claim trivially holds for the base case of a single relation r_1. Let $1 < i \leq k$ and $s_{i-1} = r_1 \bowtie \cdots \bowtie r_{i-1}$. Assume inductively that for each tuple τ' in r_1 there is exactly one tuple τ in s_{i-1}. Since $s_{i-1} \bowtie r_i$ is a 2-way foreign key join, the join attribute must be a key in r_i. Thus there is at most one tuple in r_i joining with each tuple in s_{i-1}, and furthermore, due to foreign key integrity constraints, there is at least one such tuple. Hence, for each tuple τ' in r_1 there is exactly one tuple τ in $s_i = s_{i-1} \bowtie r_i$. The claim, and hence the lemma, follows by induction. ■

From Lemma 4.1, we have that each node can be the source relation only for k-way foreign key joins involving its descendants in G. For each relation r, there is some *maximum foreign key join* (i.e., having the largest number of relations) with r as the source relation. For example, in Figure 2, $C \bowtie N \bowtie R$ is the maximum foreign key join with source relation C, and $L \bowtie O \bowtie C \bowtie N1 \bowtie R1 \bowtie PS \bowtie P \bowtie S \bowtie N2 \bowtie R2$ is the maximum foreign key join with source relation L.

Definition 4.1 Join synopses: *For each node u in G, corresponding to a relation r_1, define $\mathcal{J}(u)$ to be the output of the maximum foreign key join $r_1 \bowtie r_2 \bowtie \cdots \bowtie r_\kappa$ with source r_1. (If u has no descendants in G, then $\kappa = 1$ and $\mathcal{J}(u) = r_1$.) Let S_u be a uniform random sample of r_1. Define a* join synopsis, *$\mathcal{J}(S_u)$, to be the output of $S_u \bowtie r_2 \bowtie \cdots \bowtie r_\kappa$. The* join synopses *of a schema consists of $\mathcal{J}(S_u)$ for all u in G.* ■

To emphasize the sampling nature of join synopses, we will sometimes refer to them as *join samples*.

For example, in the TPC-D schema, the join synopsis for R is simply a sample of R whereas for C it is the join of N, R, and a sample of C. Next, we show that the join synopsis of a relation can be used to obtain a uniform random sample for a large set of queries!

Theorem 4.3 *Let $r_1 \bowtie \cdots \bowtie r_k$, $k \geq 2$, be an arbitrary k-way foreign key join, with source relation r_1. Let u be the node in G corresponding to r_1, and let S_u be a uniform random sample of r_1. Let A be the set of attributes in $r_1, \ldots, r_k$. Then, the following are true:*

- *$\mathcal{J}(S_u)$ is a uniform random sample of $\mathcal{J}(u)$, with $|S_u|$ tuples. (From Lemma 4.2.)*
- *$r_1 \bowtie \cdots \bowtie r_k = \pi_A \mathcal{J}(u)$, i.e., the projection of $\mathcal{J}(u)$ on the attributes in $r_1, \ldots, r_k$. (Trivially true from the definition of $\mathcal{J}(u)$ given in the above definition.)*
- *$\pi_A \mathcal{J}(S_u)$ is a uniform random sample of $r_1 \bowtie \cdots \bowtie r_k$ $(= \pi_A \mathcal{J}(u))$, with $|S_u|$ tuples. (Follows from the above two statements.)*

Thus we can extract from our synopsis a uniform random sample of the output of any k-way foreign key join, $k \geq 2$. For example, the join synopsis on L in the TPC-D schema can be used to obtain a sample of any join involving L (which is true for most queries in the benchmark). The next lemma shows that a single join synopsis can be used for a large number of *distinct joins*, especially for the star-like schemas common in data warehouses. Here, two joins are *distinct* if they do not join the same set of relations.

Lemma 4.4 *From a single join synopsis for a node whose maximum foreign key join has κ relations, we can extract a uniform random sample of the output of between $\kappa - 1$ and $2^{\kappa-1} - 1$ distinct foreign key joins.*

Proof. The former case arises if all the descendants of the node form a line in G. The latter case arises if the node is the root of a star of all its descendants, as in a star schema. ■

Note that since Lemma 4.2 fails to apply in general for any relation other than the source relation, the joining tuples in any relation r other than the source relation will not in general be a uniform random sample of r. Thus distinct join synopses are needed for each node/relation.

A limitation of our solution of maintaining join synopses is that for worst case schemas, the size of the maximum foreign key join can be exponential in the number of relations in the schema:

Lemma 4.5 *There exists foreign key schema with t relations such that the maximum foreign key join has $4 \cdot 2^{(t-1)/3} - 3$ relations.*

Proof. Consider a "coat hanger" H_i with root r_i. H_{i+1} has root r_{i+1} with two children l and r each of which join to r_i. It is easy to verify that the coat hanger H_i has $3i + 1$ nodes. Consider t relations which are the nodes of $H_{(t-1)/3}$ with edges between them depicting the foreign key relationships. Then it is easy to verify that the maximum foreign key join has $4 \cdot 2^{(t-1)/3} - 3$ relations. ■

However, we can expect such pathological schemas to be extremely rare in real-life situations. This is because a schema like the one above implies the presence of an exponential number of meaningful joins in a schema, which is not something that we see in real-life.

Since tuples in join synopses are the results of multi-way joins, a possible concern is that they will be too large because they have many columns. To reduce the columns stored for tuples in join synopses, we can eliminate redundant columns (for example, join columns) and only store columns of interest. Small relations can be stored in their entirety, rather than as part of join synopses. To further reduce the space required for join synopses, we can renormalize the tuples in a join synopsis into its constituent relations and remove duplicates. To the extent that foreign keys are many-to-one, this will reduce the space, although the key will then be replicated. Of course, with renormalization, when a tuple in S_u is deleted, one has to delete any joining tuples in the constituent relations as well. This can be done either immediately or in a lazy fashion in a batch. The following lemma, which we state without proof, places a bound on the size of a renormalized join synopsis.

Lemma 4.6 *For any node u whose maximum foreign key join is a κ-way join, the number of tuples in its renormalized join synopsis $\mathcal{J}(S_u)$ is at most $\kappa|S_u|$.*

As an example, consider the TPC-D schema in Figure 2. In the TPC-D benchmark database, the relations N and R, corresponding to Nation and Region, have 25 and 5 tuples in them, respectively. Therefore, we can store them in their entirety without considering any samples for them. We can therefore remove them from the graph. We are left with the nodes $L, PS, O, C, P,$ and S. The number of relations in the *maximum foreign key join* corresponding to each of these nodes (denoted by the letter κ above) is $6, 3, 2, 1, 1,$ and 1 for L, PS, O, C, P and S respectively. Let us now make two simplifying assumptions: (1) the size of the tuples in each base relation is the same; and (2) the number of tuples, n, allocated to each of the join synopses is the same. By Lemma 4.6, the total number of tuples in the synopsis is at most $|N| + |R| + \sum_u \kappa_u |S_u| = 14n + 30$. Thus we can obtain, *for every possible join* in the TPC-D schema, a uniform random sample of 1% of *each join result*, from a collection of join synopses that in total use less than 15% of the space needed for the original database! Note also that we can further reduce the size of the join synopses by taking advantage of the fact that many foreign keys are many-to-one.

To summarize, we have shown that it is possible to create compact join synopses of a schema with foreign key joins such that we can obtain a random samples of any join in the schema. In the next section, we present a detailed analysis of deciding the size of the join synopses taking into account tuple size, query frequency, etc.

5 Allocation

In this section, we present optimal strategies for allocating the available space among the various join synopses when certain properties of the query work load are known and identify heuristics for the common case when such properties are not known.

5.1 Optimal strategies

We consider the following high-level characterization of a set, S, of queries with selects, aggregates, group bys and foreign key joins. For each relation, R_i, we determine the fraction, f_i, of the queries in S for which R_i is either the source relation in a foreign key join or the sole relation in a query without joins. For example, for the 17 queries in the TPC-D benchmark, L is the source or sole relation for 14 queries and PS is the source or sole relation for 3 queries, and hence the fraction f_i equals $14/17$ for L, equals $3/17$ for PS, and equals zero for all other relations.

We seek to select join synopsis (join sample) sizes so as to minimize the average relative error over a collection of aggregate queries, based on this characterization of the set of queries. This can be done analytically by minimizing the average relative error bounds (i.e., confidence intervals) over the collection. Although this seems to imply that the optimal sample size allocation is specific to the type of error bounds used, we will show that a large class of error bounds share a common property that we will exploit for this purpose. Namely, we observe that the error bounds for COUNT, SUM, and AVG based on the commonly-used Hoeffding bounds and/or Chebychev bounds, including the new approaches discussed in Section 6, *all* share the property that the error bounds are inversely proportional to $\sqrt{n}$, where n is the number of tuples in the (join) sample. (Details on these bounds are discussed in Section 6.)

Thus the average relative error bound over the queries is proportional to

$$\sum_i \frac{f_i}{\sqrt{n_i}}, \tag{1}$$

where n_i is the number of tuples allocated to the join sample for source relation R_i.

Our goal is to select the n_i so as to minimize Equation 1 for a given bound, N, on the total memory allotted for join synopses. For each source relation R_i, let s_i be the size of a single join synopsis tuple for i. Then we require $\sum_i n_i s_i \leq N$. We show that the optimal allocation selects n_i to be proportional to $(f_i/s_i)^{2/3}$:

Theorem 5.1 *Given N, and f_i and s_i for all relations R_i, taking*

$$n_i = N' \cdot \left(\frac{f_i}{s_i}\right)^{2/3}$$

where $N' = N/(\sum_j {f_j}^{2/3} {s_j}^{1/3})$, minimizes Equation 1 subject to $\sum_i n_i s_i \leq N$.

The proof appears in the Appendix.

Note that the above analysis has ignored predicate selectivities. We observe that the relative error bounds for COUNT, SUM, and AVG based on the commonly-used Hoeffding bounds and/or Chebychev bounds, including our new approaches, are either proportional to $1/\sqrt{qn}$ or proportional to $1/q\sqrt{n}$, where q is the selectivity. In the absence of a characterization of the query work load in terms of predicate selectivities, we assume that the selectivities are independent of the relations. (Incorporating a selectivity characterization can readily be done, although the analysis is more detailed.) Under this assumption, our analysis above holds good for any mix of selectivities.

Finally, note that the sample sizes can be adapted to a changing query load by maintaining the frequencies f_i, and reallocating among the join samples as the frequencies change.

5.2 Heuristic strategies

We next consider three strategies for allocating join synopses that can be used in the absence of query work load information. These can be used as starting points for the adaptive procedure proposed above.

- *EqJoin* divides up the space allotted, N, equally amongst the relations. Each relation devotes all its allocated space to join synopses. (For relations with no descendants in the schema, this equates to a sample of the base relation.)
- *CubeJoin* divides up the space amongst the relations in proportion to the cube root of their join synopsis tuple sizes. Each relation devotes all its allocated space to join synopses.
- *PropJoin* divides up the space amongst the relations in proportion to their join synopsis tuple sizes. Each relation devotes all its allocated space to join synopses, and hence each join synopsis has the same number of tuples.

Thus for *EqJoin*, *CubeJoin*, and *PropJoin*, the number of tuples for a join synopsis with tuple size s_i is inversely proportional to s_i, $s_i^{2/3}$, and 1, respectively. When the error bounds are inversely proportional to $\sqrt{n}$, *CubeJoin* minimizes the average relative error bounds when all frequencies f_i are assumed to be equal (Theorem 5.1), and *PropJoin* minimizes the *maximum* error bound when all frequencies f_i are nonzero.

These allocation strategies using join samples can be compared against similar strategies that use only base samples:

- *EqBase* is like *EqJoin* on base samples, i.e., it devotes all its allocated space to samples of the base relations.
- *CubeBase* is like *CubeJoin* on base samples.
- *PropBase* is like *PropJoin* on base samples.

The experimental results in Section 8 quantify the advantage of the join samples strategies over the base samples strategies for representative queries.

6 Accuracy Measures

In this section, we present a detailed analysis of the error bounds obtained for approximate answers that demonstrates the trade-offs in various methods, as well as the advantages in certain scenarios of a new subsampling method we propose. We consider both traditional error bounds (e.g., confidence intervals based on Hoeffding bounds) as well as empirical bounds arising from this subsampling process. Our analysis shows that the subsampling process can noticeably improve the confidence bounds.

An important advantage of using join synopses is that queries with foreign key joins can be treated as queries without joins (single-table queries). Known confidence bounds for single-table queries are much faster to compute

Table 1: Traditional estimates and bounds.

Aggregate	Estimate e	Method	Bound on t	Guarantee
AVG	$\frac{1}{n}\sum_{i=1}^{n} v_i$	CLT	$z_p\hat{\sigma}$	no
		Hoeffding	$(\text{MAX} - \text{MIN})\sqrt{\frac{1}{2n}\ln\frac{2}{1-p}}$	yes
		Chebychev (known σ)	$\frac{\sigma}{\sqrt{n(1-p)}}$	yes
		Chebychev (estimated σ)	$\frac{\hat{\sigma}}{\sqrt{n(1-p)}}$	no
		Chebychev (conservative)	$\frac{\text{MAX}-\text{MIN}}{2\sqrt{n(1-p)}}$	yes
SUM	$\frac{m}{n}\sum_{i=1}^{n} v_i$	same as AVG, but the bound is multiplied by m		
COUNT	m	trivial	0	yes

and much more accurate than the confidence bounds for multi-table queries (see, e.g., [Haa96]).[4] Thus in this section we consider only single-table queries.

This section summarizes our analysis. For simplicity, only the results for queries with no predicates are presented. The full details, including the analysis with predicates, can be found in [GM98b].

6.1 Traditional error bounds

Consider a data set with m items (m known). Let $x_1, \ldots, x_m \in [\text{MIN}, \text{MAX}]$ be the results of applying any arithmetic expression on the items in the set. Let $v_1, \ldots, v_n$ be a uniform random sample of the multiset $\{x_1, \ldots, x_m\}$. We wish to estimate the aggregate (AVG, SUM, and COUNT) on all m values based on this sample of n values.

Table 1 summarizes the traditional estimates and the bounds for AVG, SUM and COUNT with no predicates, where p is the desired confidence probability. Shown are upper bounds for t such that $\Pr(|e - \mu| \leq t) \geq p$, where μ is the precise result to an aggregate, and e is an estimate based on n samples. These bounds are expressed in terms of σ, the standard deviation of the x_i, and $\hat{\sigma}$, the square root of the sample variance, computed from the v_i. For the CLT bound, which is derived from the *Central Limit Theorem*, (denoted the *large sample* bound in [HHW97]), z_p is the quantile of the normal distribution such that for a standard normal random variable, X, $\Pr(X \in [-z_p, z_p]) = p$ (e.g., $z_{.95} = 1.96$ and $z_{.9} = 1.65$). Three versions of the Chebychev bound are provided: one where σ is known, one where σ is replaced by $\hat{\sigma}$ (similar to the CLT bound), and one where σ is replaced by its upper bound (similar to the Hoeffding bound). The last column indicates whether or not a bound is guaranteed with probability p or holds with probability p only under large sample assumptions [HHW97, Haa97].

Comparing the bounds in Table 1, we see that among the two bounds using $\hat{\sigma}$, the Chebychev (estimated σ) bound is better than the CLT bound whenever $n > 1/(z_p^2(1-p))$. Since n must be sufficiently large for either approximation to hold, the Chebychev bound is better unless the desired error probability is inversely proportional to n. Comparing the guaranteed bounds, it can be shown that regardless of n, the Chebychev bounds are better for $p < .76$, even when the Chebychev (conservative) bound is used. When no better bound is known for σ than this conservative bound ($(\text{MAX} - \text{MIN})/2$), then the Hoeffding bound is better for $p > .76$. However, if σ is known, then it can be shown that

[4]For queries with joins, at least one of which is not foreign key, one can use the (much weaker) multi-table formulas from [Haa96], applied to the join synopses representing the joins that are foreign key.

for

$$p_1 \approx 1 - \frac{\sigma^2}{(\text{MAX} - \text{MIN})^2 \ln \frac{\text{MAX}-\text{MIN}}{\sigma}},$$

the Chebychev (known σ) bound is better for $p < p_1$, and the Hoeffding bound is better for $p > p_1$.

6.2 Improved accuracy measures through subsampling

In this section, we consider the following estimation approach:

1. We partition the set of sample points into k subsets (subsamples), which we call "chunks", and for each chunk j, we compute an estimator, e_j, based on the sample points in the chunk.

2. We report an estimate and a bound based on the e_j.

We can apply any of the methods in Table 1 to obtain the chunk estimators, e_j, and the confidence bounds on the estimators. Since each chunk estimator is based on only a subsample, the confidence in a single chunk estimator is less than if it were based on the entire sample. However, since the subsamples do not overlap, each chunk estimator is an independent random estimator, and an overall estimate based on all the e_j can potentially result in a better estimate and smaller error bounds. We analyze and compare two possible choices for reporting an overall estimate e: taking the average of the e_j and taking the median of the e_j.

Previous work (see, e.g., [AMS96]) has shown that given a procedure yielding a single random estimator, one can often obtain an overall estimate with small error bounds and high confidence, as follows: Generate a set of independent estimators, partition them into subsets and take the average within each subset, and then take the median of these averages. The final error bounds are obtained by using Chebychev bounds to analyze the subset estimators and Chernoff bounds to analyze the extent to which taking the median boosts the confidence. In our context, each sample point v_i is a random estimator, the subsets are the chunks, and the overall estimate is the average or the median. The number of sample points is fixed and we seek the best estimate using the sample at hand.

Our contributions are as follows.

- Within the general chunking framework, we propose and explore a number of alternative procedures for reporting an estimate and an error bound based on the chunks, including varying the number of chunks.

- Whereas previous work on the median has been asymptotic in nature, we show the precise trade-offs for when the guaranteed bounds for the median improve upon the bounds with no chunking, and what the optimal number of chunks to use for confidence probabilities of practical interest. To do this, we do not apply the asymptotic Chernoff bounds, but instead the actual bounds.

- We show that for desired confidence probabilities above 96%, the best bounds are obtained by taking the median of a small number of chunks and applying Chebychev bounds. However, we show that chunking does not help either when the Hoeffding bound is used or when the Chebychev (conservative) bound is smaller than the Hoeffding bound.

- We propose and explore the use of the chunk estimators in generating *empirical* error bounds. Each chunk estimator can be viewed as an experiment run on independent samples of the actual data, and the user is presented with a summary of the results of these experiments.

Table 2: Comparison of the optimal number of chunks, for guaranteed bounds.

Aggregate	Desired confidence	Estimate e	Guaranteed bound on t
AVG	$0 < p \leq .961$	$\frac{1}{n}\sum_{i=1}^{n} v_i$	$\frac{\sigma}{\sqrt{n(1-p)}}$
	e.g., $p = .95$		$\frac{4.47\sigma}{\sqrt{n}}$
	$.961 \leq p \leq .984$	median of 3 chunk averages	$\frac{\sigma\sqrt{3}}{\sqrt{n(1-\rho_3)}}$
	e.g., $p = .98$		$\frac{5.98\sigma}{\sqrt{n}}$
	$.984 \leq p$	median of $k \geq 5$ chunk averages	$\frac{\sigma\sqrt{k}}{\sqrt{n(1-\rho_k)}}$
	e.g., $p = .99$	$k = 5$	$\frac{6.87\sigma}{\sqrt{n}}$
SUM	same as AVG, but the estimate and the bound are multiplied by m		

Trade-offs in guaranteed bounds with chunking: Consider the set-up defined in Section 6.1. Partition the n sample points into k equal-sized chunks and let e_j be the estimator for chunk j. For example, for the AVG aggregate, e_j is the sum of the sample points v_i in chunk j divided by n/k, the number of sample points in the chunk.

We will apply the two-step estimation approach discussed above. While our desired confidence in the overall estimate is p, the confidence in a single chunk estimator can be less than p, since we plan to boost this confidence by taking the median. For any confidence ρ such that $1/2 < \rho \leq p$, we can use any of the approaches in Table 1 to determine an error bound t_j for each chunk estimator e_j. (An analysis and comparison of these approaches are given later in this section.) Let t be the maximum of the t_j, so that for each e_j, we have:

$$\Pr(|e_j - \mu| < t) \geq \rho. \tag{2}$$

We now apply the second step by taking the median[5], e^*, of the e_j's. Our goal is to have $\Pr(|e^* - \mu| < t) \geq p$.

Call an estimate, e, "good" if $|e - \mu| < t$, and "bad" otherwise. Since e^* is the median of the e_j, we know that e^* will be good if and only if at least half of the e_j are good. By Equation 2, each e_j is good with probability at least $\rho > 1/2$. Let k' be the number of good e_j, and let

$$q_k = \sum_{i=0}^{\lfloor \frac{k}{2} \rfloor} \binom{k}{i} \rho^i (1-\rho)^{k-i}. \tag{3}$$

Then the probability that e^* is bad is $\Pr\left(k' \leq \lfloor \frac{k}{2} \rfloor\right) \leq q_k$. So our goal is attained if $q_k = 1 - p$.

Recall that each chunk estimator is based on n/k sample points. Thus as the number of chunks increases, the quality of the chunk estimator decreases, so that, e.g., ρ decreases for a fixed t, and t increases for a fixed ρ. On the other hand, the failure probability q_k decreases for a fixed ρ. Thus the best choice for k depends on the relationship of ρ and t in Equation 2 as a function of k, and the desired confidence $p = 1 - q_k$.

In the remainder of this section, we highlight our results analyzing and comparing the effects of applying the various methods in Table 1, and determining the optimal number of chunks.

Table 2 summarizes our analysis on the use of Chebychev for Equation 2 in conjunction with various values for p, with and without chunking. This table shows, for various desired confidences p, the optimal choice for the number of

[5]If k is even, take either of the two medians. To reduce the bias, select with equal probability.

Table 3: Hoeffding versus Chebychev (conservative). Note that conservative chunk bounds are never helpful in conjunction with Hoeffding or Chebychev (conservative).

Aggregate	Desired confidence	Estimate e	Guaranteed bound on t
AVG	$0 < p \leq .76$	$\frac{1}{n}\sum_{i=1}^{n} v_i$	$\frac{\text{MAX}-\text{MIN}}{2\sqrt{n(1-p)}}$ by using Chebychev (conservative)
	$.76 \leq p \leq .99$	$\frac{1}{n}\sum_{i=1}^{n} v_i$	$(\text{MAX} - \text{MIN})\sqrt{\frac{1}{2n} \ln \frac{2}{1-p}}$ by using Hoeffding
SUM	same as AVG, but the estimate and the bound are multiplied by m		

chunks, k, and the bound that can be guaranteed. The bound is expressed in terms of ρ_k, which is the ρ in Equation 3 such that $p = 1 - q_k$. The bounds are shown for "Chebychev (known σ)". Alternatively, as in Table 1, we can obtain bounds for "Chebychev (estimated σ)" by plugging in $\hat{\sigma}$ for σ in Table 2, where $\hat{\sigma}$ is computed over all the sample points, not just those in one chunk. We can also obtain bounds for "Chebychev (conservative)" by plugging in $(\text{MAX} - \text{MIN})/2$ for σ.

Next we show that chunking does not help either when the Hoeffding bound is used or when the Chebychev (conservative) bound is used. First, we consider the use of Hoeffding for Equation 2 in conjunction with various values for p, with and without chunking.

Let $e_{(k)}$ be a chunk estimator when using $k \geq 1$ chunks. For a fixed n and $t > 0$, we have by Hoeffding that

$$\Pr(|e_{(k)} - \mu| \geq t) \leq 2e^{-2nt^2/k(\text{MAX}-\text{MIN})^2}.$$

Thus the probability that $e_{(k)}$ is bad is $2^{1-1/k}$ times the kth root of the probability that $e_{(1)}$, the estimate with no chunking, is bad. We show in the appendix that with Hoeffding bounds, no chunking is always better than chunking, by proving a more general result (Lemma A.1) on the ineffectiveness of chunking when analyzed using bounds whose dependency on k is the kth root of the no chunking bounds.

Although we indicated above that one can obtain bounds for Chebychev (conservative) by plugging in $(\text{MAX} - \text{MIN})/2$ for σ in Table 2, this bound is strictly worse than the Hoeffding bound for all probabilities at which chunking is useful for Chebychev (conservative). For example, for $.76 < p \leq .99$, the Hoeffding bound with no chunking is smaller than any Chebychev (conservative) bound obtained with or without chunking. Thus Table 3 summarizes what estimate and bound should be used, if the only bound for σ is $(\text{MAX} - \text{MIN})/2$.

Note that we have thus far considered only equal-sized chunks. Lemma A.2 in the appendix shows that for the techniques considered in this section, taking chunks of different sizes only increases the error bounds.

Using chunking for empirical error bounds: We next consider a novel use of the chunk estimators in generating empirical error bounds. The motivation is the following. Often, error bounds derived analytically are overly pessimistic: the estimated answer is closer to the exact answer more often than indicated by the analytical bound. Papers (such as this one) describing estimation techniques can report on multiple trials of an experiment on various data sets, in order to convince the reader that the estimate is more accurate than the analytical bounds show. However, this is not entirely satisfactory, as the data sets of interest to the reader/user may not exhibit the good behavior of the data sets used in the paper.

We propose chunking as a means to report on multiple experiments run on the *actual query and data*. Each subsample is its own experiment on the actual query and data, and there are various possibilities on how to report these results to the user. In Section 8, we study the effectiveness of reporting a CLT bound using the sample variance of the chunk estimators, or alternatively, reporting the minimum and maximum of the chunk estimators. Other alternatives include reporting various quantiles of the chunk estimators. The feedback to the user is intuitively of the form: k independent experiments were run for your query, all (or say, 90%) of which fell within the range $[x, y]$, with the average (or median) being e.

6.3 Summary of this section

To summarize, an important advantage of using join synopses is that queries with foreign key joins can be treated as queries without joins (i.e., as single-table queries). There are several popular methods (see Table 1) for obtaining error bounds for approximate answers to (single-table) aggregation queries. We have presented a detailed analysis that demonstrates the *precise trade-offs* among these methods, as well as a method based on subsampling which we call "chunking". Within the general chunking framework, we proposed and explored a number of alternative procedures for reporting an estimate and an error bound based on the chunks. Our results (see Table 2) showed that for confidence probabilities above 96%, the best bounds are obtained by taking a small number of chunks and applying Chebychev (known σ) or Chebychev (estimated σ) to the chunk estimators and reporting the median of these estimators. For smaller probabilities, the best bounds are obtained by reporting an overall estimate ignoring the chunks (which is equivalent to taking an average of the chunk estimators[6]), and then either applying Hoeffding for *guaranteed* bounds (see Table 3), applying Chebychev (estimated σ) for *large sample* bounds, or using the chunk estimators for *empirical* bounds.

7 Maintenance of Join Synopses

In this section, we focus on the maintenance of join synopses when the underlying base relations are being updated. (We consider both insertions and deletions.) The techniques we propose are simple to implement and require only infrequent access to the base relations. Note that maintenance of samples of base relations in the present of updates was studied in [GMP97b]. We focus in this section on maintaining join synopses.

Our algorithm for maintaining a join synopsis $\mathcal{J}(S_u)$ for each u is as follows. Let p_u be the current probability for including a newly arriving tuple for relation u in the random sample S_u. (This probability is typically the ratio of the number of tuples in S_u to the number of tuples in u.) On an insert of a new tuple τ into a base relation corresponding to a node u in G, we do the following. Let $u \bowtie r_2 \bowtie \cdots \bowtie r_\kappa$ be the maximum foreign key join with source u. (1) We add τ to S_u with probability p_u. (2) If τ is added to S_u, we add to $\mathcal{J}(S_u)$ the tuple $\{\tau\} \bowtie r_2 \bowtie \cdots \bowtie r_\kappa$. This can be computed by performing at most $\kappa - 1$ look-ups to the base data, one each in $r_2, \ldots, r_\kappa$. (For any key already in $\mathcal{J}(S_u)$, the look-ups for it or any of its "descendants" are not needed.) (3) If τ is added to S_u and S_u exceeds its target size, then select uniformly at random a tuple τ' to evict from S_u. Remove the tuple in $\mathcal{J}(S_u)$ corresponding to τ'.

On a delete of a tuple τ from u, we first determine if τ is in S_u. If τ is in S_u, we delete it from S_u, and remove the tuple in $\mathcal{J}(S_u)$ corresponding to τ. As in [GMP97b], if the sample becomes too small due to many deletions to the sample, we repopulate the sample by rescanning relation u.

[6]In the case of predicates, it is equivalent to the *weighted* average of the chunk estimators, where each estimator is weighted by the number of tuples in the chunk that satisfy the predicate.

Note that this algorithm only performs look-ups to the base data with (small) probability p_u. Also, when a tuple is inserted into a base relation u, we never update join synopses for any ancestors of u. Such updates would be costly, since these operations would be performed for every insert and for each ancestor of u. Instead, we rely on the integrity constraints to avoid these costly updates.

Theorem 7.1 *The above algorithm properly maintains all S_u as uniform random samples of u and properly maintains all join synopses $\mathcal{J}(S_u)$.*

Proof. Due to the integrity constraints, for each edge from w to u, there is exactly one tuple in u joining with each tuple in w at all times. Thus any subsequent tuple inserted into u cannot join with any tuple already in w, and any tuple deleted from u cannot join with a tuple still in w. ■

Note that similar techniques work in general for functional dependencies in database schema [Ull88].

We assume that updates may be applied in a "batch" mode. In such environments, join synopses can be kept effectively up-to-date at all times without any concurrency bottleneck. In an online environment in which updates and queries intermix, an approximate answering system can not afford to maintain up-to-date synopses that require examining every tuple, such as the minimum and maximum value of an attribute, without creating a concurrency bottleneck. In such environments, maintenance is performed only periodically. Approximate answers depending on synopses that require examining every tuple would not take into account the most recent trends in the data (i.e., those occurring since maintenance was last performed), and hence the accuracy guarantees would be weakened. Note that the incremental maintenance algorithm described in this section can be used to compute a join synopsis from scratch in limited storage, in one scan of the base data followed by indexed look-ups on a small fraction of the keys, should such a recomputation be necessary.

8 Experimental Evaluation

In this section, we present the results of an experimental evaluation of the techniques proposed in this paper. Using data from the TPC-D benchmark, we show the effectiveness of our approach in providing highly accurate answers for approximate join aggregates.

We begin this section by describing the experimental testbed. We then present results from two classes of experiments—*accuracy* experiments and *maintenance* experiments. In the accuracy experiments, we compare the accuracy of techniques based on join synopses to that of techniques based on base samples. The two key parameters in this study are query selectivity and total space allocated to precomputed summaries (summary size). We first compare the techniques for a fixed selectivity and varying summary size and then compare the techniques for a fixed summary size and varying selectivities. We also study the performance of the different methods for generating confidence bounds. In the maintenance experiments, we study the cost of keeping the join synopses up to date in the presence of insertions/deletions to the underlying data. We show that join synopses can be maintained with very little overhead even when updates significantly change the characteristics of the underlying data. Finally, we address some implementation issues that arise in approximate query answering and briefly discuss the architecture of the Aqua system which was used to generate the results.

Table 4: Features of relations in the TPC-D benchmark.

Table Name	# of Columns	Cardinality	Table Name	# of Columns	Cardinality
`Customer`	8	$45K$	`Part`	9	$60K$
`Lineitem`	16	$1800K$	`Partsupplier`	5	$240K$
`Nation`	4	25	`Region`	3	5
`Order`	9	$450K$	`Supplier`	7	$3K$

```
select avg(l_extendedprice) from customer, order, lineitem, supplier, nation, region
    where c_custkey = o_custkey and o_orderkey = l_orderkey and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey and s_nationkey = n_nationkey and n_regionkey = r_regionkey
    and r_name = [region] and o_orderdate >= DATE [startdate] and o_orderdate < DATE [enddate]
```

Figure 3: Query $\mathcal{Q}_a$ used for accuracy experiments. Based on Query $Q5$ from the TPC-D benchmark.

8.1 Experimental testbed

We ran the tests on the TPC-D decision support benchmark. We used a scale factor of 0.3 for generating our test data. This results in a database that is approximately 420 megabytes.[7] Table 4 summarizes the important features of the 8 relations in the TPC-D database.

Our experiments were run on a lightly loaded 296MHz UltraSPARC-II machine having 256 megabytes of memory and running Solaris 5.6. All data was kept on a local disk with a streaming throughput of about 5MB/second.

Query model: The query used for the accuracy experiments is based on query $Q5$ in the TPC-D benchmark and is an aggregate that is computed on the join of `Lineitem`, `Customer`, `Order`, `Supplier`, `Nation` and `Region`. Of the six relations involved in the join, the `Nation` and `Region` relations are sampled in their entirety by Aqua because of their low cardinality. This effectively makes the problem as difficult as estimating an aggregate from a (still complex) four-way join.

The SQL statement for the query is given in Figure 3. It computes the average price of products delivered by suppliers in a nation to customers who are in the same nation. The select conditions take three input parameters — `region`, `startdate` and `enddate`. These restrict suppliers and customers to be within a specific region and focus on business conducted within a specific time interval. In the following experiments, we will vary one or more of these parameters to study the performance for various query selectivities.

In this study, we have focused only on the hard problem of computing approximate aggregates on multi-way joins. Of course, our sampling results extend to the simple case of single table aggregates. Thus, due to space constraints, we do not show any results for the single table case. Besides, those results qualitatively mirror the ones presented in the context of online aggregation for single table aggregates [HHW97].[8]

Space allocation schemes: Recall from Section 5 that we proposed a number of schemes for allocating a given amount of summary space to enable approximate query answering. For the case where certain characterizations of the query mix were known, we presented optimal allocation strategies to minimize overall error. However, for this experimental study, we assume the more realistic scenario where this information is unavailable. Thus, we study the

[7]By the TPC-D specification, a scale factor of 0.3 creates approximately 300 megabytes of data. Due to some overhead in the storage component of Aqua, the data volume is slightly higher than in the specification. We are currently working on reducing this storage overhead.

[8]Of course, we provide a single answer with error bounds where as they have progressively refining output.

six space allocation schemes proposed in Section 5.2, namely, *EquiBase*, *CubeBase*, *PropBase*, *EquiJoin*, *CubeJoin* and *PropJoin*. For the purposes of this experiment, we focus on the four major relations used in $\mathcal{Q}_a$, and allocate base samples and join synopses only on those relations. Therefore, the base sampling schemes divide up the summary space among samples of `Lineitem`, `Customer`, `Order`, and `Supplier`, whereas the join synopses schemes distribute the summary space to join synopses for `Lineitem` (which includes columns from `Customer`, `Order`, and `Supplier`), for `Customer` (which includes columns from `Order`), for `Order` (whose join synopsis is just a base sample), and for `Supplier` (whose join synopsis is also a base sample).

Recall that *PropJoin* gives an equal number of tuples to the various samples whereas *EquiJoin* divides the space equally. Thus, among the various schemes, the source relation in the 4-way join in $\mathcal{Q}_a$, `Lineitem`, is allocated the most space by *PropJoin* since it has the largest tuple and the least space by *EquiJoin*, while *CubeJoin* allocates space in between these two extremes. Likewise, among the base sample schemes, *PropBase* allocates the most space to the base sample of `Lineitem`, then *CubeBase*, then *EquiBase*. In order to avoid clutter in the graphs that follow, we do not plot *CubeJoin* and *CubeBase* and only show numbers for the other four schemes. They cover the entire range of performance for the different schemes.

The experiments also study the sensitivity of the various schemes to the total summary size allocated (parameter *SummarySize* in the figures). *SummarySize* is varied from 0.1% to 3% of the total database size, varying the actual summary size in bytes from 420 KBytes to 12.5 MBytes.

8.2 Experimental results

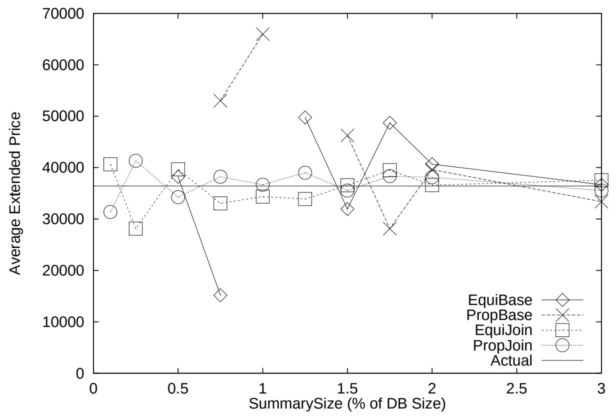
In this section, we present the results of the experimental study. The first three experiments cover the accuracy studies and the final experiment addresses the problem of maintaining join synopses during updates to the underlying data. It should be noted that the graphs presented in this section are a small subset of the results that we obtained. These results have been chosen because they demonstrate the different aspects of approximate query answering using join synopses.

8.2.1 Experiment 1: Join synopsis accuracy

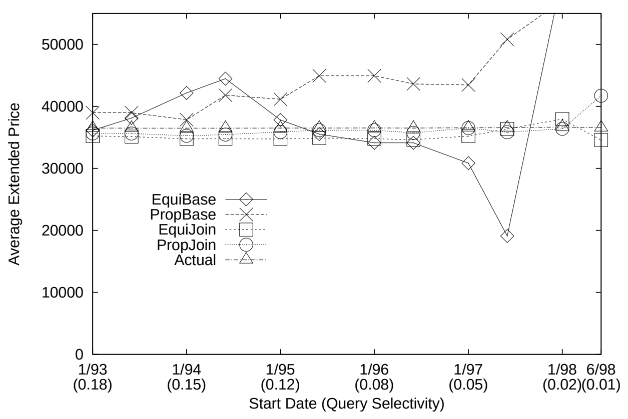
In this experiment, we study the accuracy of the four space allocation schemes for different values of summary size (parameter *SummarySize*) and for different query selectivities. We compare the actual answer of running query $\mathcal{Q}_a$ (Figure 3) on the full TPC-D database against the approximate answers obtained from the different schemes.

Consider Figure 4(a). It plots the average extended price computed by the different schemes for varying summary sizes. The actual answer is shown as a straight line parallel to the x-axis. Following the specification for query $Q5$ in the TPC-D benchmark, the `region` parameter is set to `ASIA` and the selection predicate on the `o_orderdate` column to the range [`1/1/94,1/1/95`].

Consider the two schemes that use only samples of the base relations, *EquiBase* and *PropBase*. Figure 4(a) shows that these schemes produce answers consistently only when the summary size exceeds 1.5% of the database. (For lower sample sizes, the join of the base samples is completely empty!) In fact, it is not until 2% summary size that the approximate answer produced by them comes close to the actual answer. In fact, on the left end of the graph (for smaller summary sizes), these scheme either produce no output at all (e.g., *PropBase* for 1.25% synopsis size), or produce answers that are significantly different from the real answer (with errors close to 100% in some cases).



(a) Date Interval = `[1/1/94,1/1/95]`



(b) *SummarySize* = 1.5%

Figure 4: Behavior of join synopsis and base sample allocation strategies for different (a) summary size values and (b) for different query selectivities.

The schemes based on join synopses, *EquiJoin* and *PropJoin*, on the other hand, not only produce output over the entire range of summary size studied but are also fairly accurate in estimating the correct answer.[9] Even for a summary size of 0.1% (420 Kbytes) shared among all the four join synopses, the results from both the schemes are within 14% of the actual answer! Moreover, the variation in the answers is lower than the variation in the answers from base sampling schemes. The difference between the two types of allocation schemes is further highlighted in Table 5, which shows the number of tuples in the join output for the four schemes. In most cases, the schemes based on join synopses produce at least an order of magnitude more number of tuples than the base sampling schemes do. As expected, *PropJoin* is the most accurate since it assigns the most space to `Lineitem`, the root of the 4-way join.

Figure 4(b) studies the sensitivity of the four allocation schemes for varying selectivities, with the summary size set to 1.5% of the database size. We change the selectivity of query $\mathcal{Q}_a$ by changing the date range in the selection condition on the `o_orderdate` attribute. To control the selectivity, we fixed the parameter `enddate` to `1/1/99`, the tail end of the date range in the TPC-D specification. We varied the `startdate` parameter from `1/1/93` to `6/1/98` in steps of six months. The `startdate`s are shown on the x-axis with the corresponding query selectivity given in brackets below.

Selectivity and summary size have a similar effect on the performance of the base sampling schemes. While the answers returned by the *EquiBase* and *PropBase* techniques are reasonably close to the actual answer when the selectivity is high (left end of the x-axis), the answers fluctuate dramatically as the selectivity decreases. As expected, the join synopsis schemes, *EquiJoin* and *PropJoin*, stay close to the actual answer over the entire range deviating only slightly when the selectivity is down to 1%.

These graphs demonstrate the advantages of schemes based on join synopses over base sampling schemes for approximate join aggregates. Even with a summary size of only 0.1%, join synopses are able to provide fairly accurate aggregate answers.

[9] A detailed analysis of the error bounds on these answers is shown later in Experiment 2.

8.2.2 Experiment 2: Analyzing accuracy bounds

In this experiment, we study the various techniques presented in Section 6 for confidence bounds on estimated answers. Based on the results of the last experiment, we focus only on *PropJoin*. Results for other schemes based on join synopses are qualitatively similar.

Figure 5 plots the error bounds for the *PropJoin* allocation scheme for a summary size of 2%. It shows the 90% confidence bounds of three of the five techniques in Table 1, namely, *Hoeffding*, *Chebychev (estimated σ)*, and *Chebychev (conservative)*.[10] These bounds are compared with bounds based on chunk statistics. The number of chunks are varied on the x-axis. Since the first three bounds are independent of the number of chunks, they are shown only once, for $x = 1$, on the left.[11] We first consider the chunk-independent guaranteed bounds (*Hoeffding* and *Chebychev (conservative)*), then the chunk-independent experimental bounds (*Chebychev (estimated σ)*) and finally the bounds based on chunk statistics.

Recall that among the guaranteed bounds, *Hoeffding* is tighter than *Chebychev (conservative)* for confidence greater than 76% and Figure 5 reflects this. Next, consider the *Chebychev (estimated σ)* bounds. It is tighter by 40% on each end compared to *Hoeffding*. While these bounds are not guaranteed (they are based on large sample assumptions), in all our experiments we never found them to not overlap the real answer.

We now look at three chunk related bounds. *Chunk Std. Dev.* uses the standard deviation of the chunk answers to determine the bound. Again, while not guaranteed, these bounds are tighter than *Hoeffding* and *Chebychev (estimated σ)* and always overlap the exact answer. As the number of chunks increase, the variance increases as expected and the bounds get worse. (Recall that the chunk size is inversely proportionally to the number of chunks since the total amount of space allocated to a join synopsis is fixed.) The *Chunk Min-Max* bound plots a bar from the minimum to the maximum value returned by the chunk results. The bar expands with increasing chunks but always includes the real answer. Finally, we also plot the median, which as shown in Table 2 leads to tighter guaranteed bounds when the desired confidence is above 96%. Since we use a lower confidence bound (90%) in this case, we do not plot the error bar for the median.

One of our motivations for studying empirical bounds such as those obtained by chunking was the following: Even though our techniques could produce estimated answers that were very close to the actual answer, error bounds using known statistical techniques such as *Hoeffding* were comparatively poor. For example, in Figure 5, the estimated answer is away from the actual answer by 4.5%. However, the *Hoeffding* error bound gives a 90% confidence range of $\pm 20\%$. Our experiments confirmed the utility of using empirical bounds to supplement the guaranteed statistical bounds. Note that the guaranteed bounds are based on worst case assumptions on the data distribution for a given MAX and MIN; hence the gap between them and the empirical bounds can be made arbitrarily large by adding a few outliers that increase MAX and MIN while not affecting the exact answer. Also, the use of a small number of chunks (e.g., 5 chunks) to generate chunk-based bounds can further enhance the trust of the user in the estimate produced by the approximate answering system.

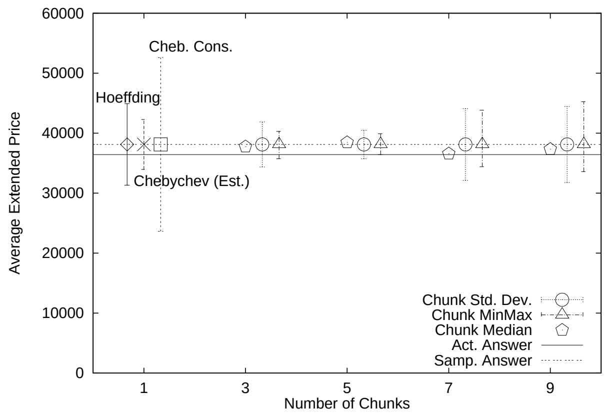
In summary, using the techniques proposed in this paper, one can design a query answering system which not only produces accurate approximate answers for complex multi-way join aggregates but also provides good error bounds

[10] The *CLT* bound is not shown, since as discussed in Section 6.1, it is strictly worse than the *Chebychev (estimated σ)* bound. The *Chebychev (known σ)* bound is not applicable, since σ, the standard deviation of the extended price over all tuples in the database that satisfy the predicate, is not known a priori and would be prohibitively expensive to compute at query time.

[11] Note that some of the bounds are slightly shifted on their x-coordinate to avoid clutter.

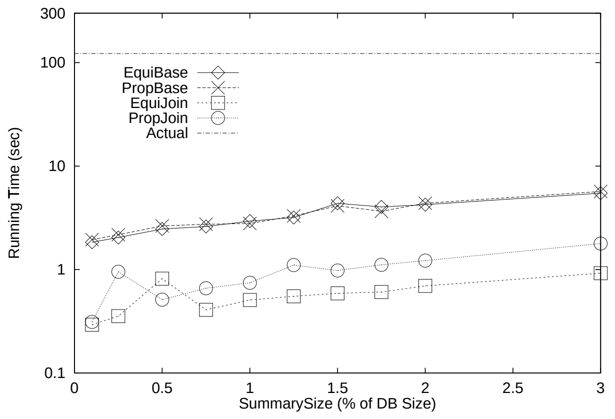
Table 5: Output Size for the various allocation schemes.

SummarySize	Base Samples		Join Synopses	
	EquiBase	*PropBase*	*EquiJoin*	*PropJoin*
0.1%	0	0	6	25
1%	0	2	56	142
1.5%	12	4	104	228
2%	38	44	131	300
3%	38	108	195	453



SummarySize = 2%

Figure 5: Traditional and subsampling error bounds.



Query $\mathcal{Q}_a$

Figure 6: Query execution time for various schemes.

using guaranteed statistical and heuristic empirical techniques.

8.2.3 Experiment 3: Query execution timing

Figure 6 plots the time taken by the various strategies to execute the query (the y-axis is in logscale). The time to execute the actual query is 122 seconds and is shown as a straight line near the top of the figure. As expected, the response times increase with increasing summary size. However, for all the sizes studied, the execution time for the query using join synopses is two orders of magnitude smaller! (The times using base samples are more than an order of magnitude smaller than those computing the actual answer.)

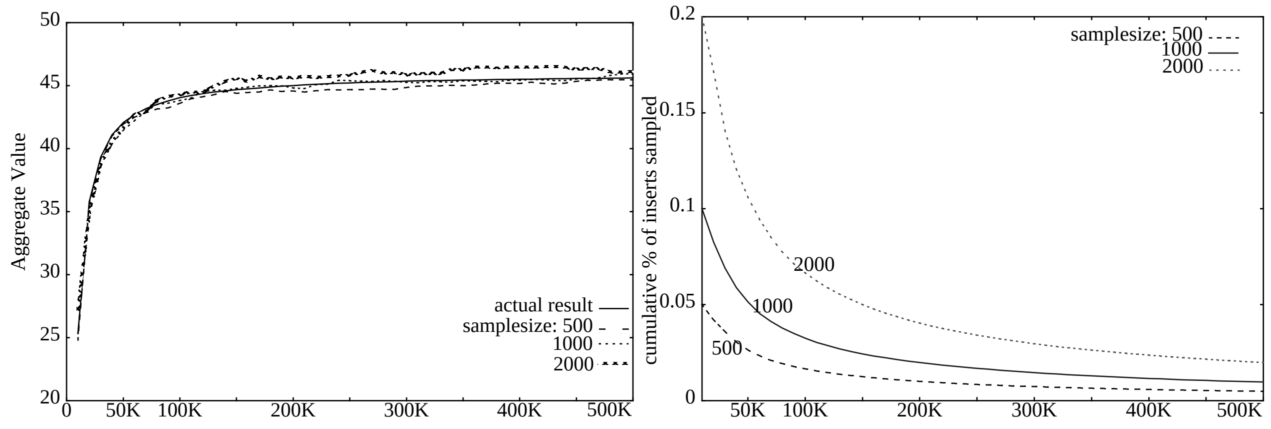
This experiment demonstrates that it is possible to use join synopses to obtain extremely fast approximate answers with minimal loss in accuracy. This is good evidence that applications such as decision support and data warehousing, which can often tolerate marginal loss in result accuracy, can benefit tremendously from the faster responses of approximate query answering systems.

8.2.4 Experiment 4: Join synopsis maintenance

In this section, we show experimental results demonstrating that join synopses can be maintained with very minimal overhead. Such join synopses can give very good approximate answers even when updates significantly change the nature of the underlying data. We base this section on a join between the `Lineitem` and `Order` tables. The

```
select avg(l_quantity) from lineitem, order
    where l_orderkey = o_orderkey and o_orderstatus = F
```

Figure 7: Join synopsis maintenance query $\mathcal{Q}_m$.



(a) Result accuracy (b) Maintenance cost

Figure 8: The left side of the figure shows aggregate values computed from join synopses of various sizes. The right side shows the cost of online maintenance of the samples. Both of them are plotted against $500,000$ updates to the `Lineitem` table.

query used retrieves the average quantity of tuples from the `Lineitem` table that have a particular value for the `o_orderstatus` column. The SQL statement for the query is given in Figure 7.

We consider the maintenance of a join synopsis for `Lineitem` as tuples are inserted into the `Lineitem` table, using the algorithm of Section 7. Note that insertions into other tables in the schema can safely be ignored in maintaining the `Lineitem` join synopsis. Figure 8(a) plots the aggregate values computed from join synopses of different sizes. Even for extremely small sizes, the join synopsis is able to track the actual aggregate value quite closely despite significant changes in the data distribution. Figure 8(b) then shows that maintenance of join synopses is very inexpensive, by plotting the average fraction of the inserted `Lineitem` tuples that are actually inserted into the join synopsis. In accordance with the algorithm of Section 7, we go to the base data only when a tuple is inserted into the join synopsis. It is clear from the figure that this number is a small fraction of the total number of tuples inserted. (For example, when maintaining a sample of 1000 tuples and processing $500,000$ inserts, we go to the base data only 4822 times.)

8.2.5 Summary of experiments

The experimental results in this section empirically demonstrate the validity of the techniques proposed in this paper. The results show that join synopses can be used to compute approximate join aggregates extremely quickly, and that the performance of join synopses is superior to that of base sampling schemes. Further, the results also show that join synopses can be maintained inexpensively during updates. Finally, the results indicate that empirical error bounds are a good complement to traditional guaranteed bounds on approximate answers.

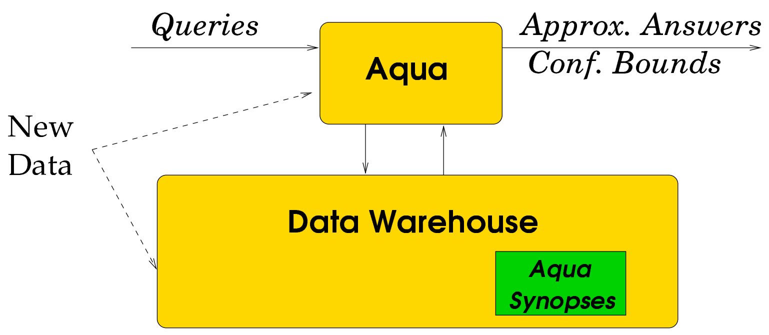


Figure 9: The Aqua architecture.

```
select sum(l_quantity)
  from lineitem, order
  where l_orderkey = o_orderkey
  and o_orderstatus = F
```

(a) Original query

```
select 100*sum(l_quantity)
  from LOsynopsis
  where o_orderstatus = F
```

(b) Rewritten query

```
select 100*sum(l_quantity), chunkid
  from LOsynopsis
  where o_orderstatus = F
  groupby chunkid
```

(c) Rewritten for chunks

Figure 10: Query rewriting to use join synopses.

8.3 Architecture

As discussed in the introduction, the work in this paper was done as part of the Approximate QUery Answering (Aqua) project [GMP97a] at Bell Labs. We are currently in the process of implementing the Aqua system. In this section, we briefly describe certain aspects of the Aqua system relevant to the implementation of join synopses. This section can also be viewed as one possible way to implement join synopses in a DBMS.

Aqua is designed as a module that sits on top of a DBMS managing a data warehouse. Aqua precomputes statistical summaries on the relations in the warehouse. Currently, the statistics take the form of *samples* and *histograms* which are stored as regular relations inside the warehouse; they are also incrementally maintained up-to-date as the base data is updated.

Aqua answers user queries using the precomputed summaries. Approximate answers are provided by rewriting the user query over the summary relations and executing the new query. The rewriting involves suitably scaling the results of certain operators within the query. Finally, the query and the approximate answer are analyzed to provide guarantees on the quality of the answer (Section 6). The high-level architecture of Aqua is depicted in Figure 9.

We now give two examples of query rewriting in Aqua that illustrates the use of join synopses, highlighting the rewriting process with and without chunks. Further details on the rewriting procedure are provided elsewhere [GMP97a].

Consider the query in Figure 10(a). It is a variation of the update query $\mathcal{Q}_m$ shown in Figure 7, except that it computes the SUM rather than the AVG aggregate. Further, assume that `LOsynopsis` is a 1% sample of the join between the `Lineitem` and `Order` tables. When the query is submitted to Aqua, it identifies the join being computed in the query and rewrites the query to refer to the `LOsynopsis` table. The rewritten query submitted to the warehouse is shown in Figure 10(b). (Calculation of error bounds is not shown here for simplicity.)

Recall that we use chunks to provide empirical guarantees on the approximate answers computed by Aqua. Since the number of chunks is fixed before query time, one approach to implementing chunks is to materialize them as separate tables and run the transformed query on each chunk. However, this can be expensive. Instead, to avoid this overhead, we make use of the group-by operation, as follows. We add an extra column to each join synopsis relation and populate it randomly with a value in the range $[1 \ldots k]$, where k is the desired number of chunks. This is done at the time the join synopsis is created, not at query time. At query time, we rewrite the query to include this extra column as a (possibly additional) group-by column, and execute the query. In a postprocessing step, we assemble the estimate (e.g., take the median or the average) and the bound, and return the result. In this way, all chunks are handled with a single query. Figure 10(c) shows the query in Figure 10(a) transformed to used chunks. Note that *chunkid* is a column that is added to the `LOsynopsis` table to identify chunks. (Once again, computation of error bounds is not shown in the figure for simplicity.)

In addition to the issue of join synopses, we are investigating several other issues in Aqua. These include the use of other synopsis data structures, such as those discussed in [GM98c, GM98a], and providing approximate answers for general (non-aggregate) queries that return a set of tuples, either by using samples [GMP97a] or histograms [IP98]. Further details on Aqua can be found in [GMP97a, GPA$^+$98].

9 Conclusions

In this paper, we have focused on the important problem of computing approximate answers to aggregates computed on multi-way joins. For data warehousing environments with schemas that involve only foreign-key joins, we have proposed join synopses as a solution to this problem. We have shown that schemes based on join synopses provide better performance than schemes based on base samples for computing approximate join aggregates. Further, we have also shown that join synopses can be maintained efficiently during updates to the underlying data. Finally, we have explored the use of empirical confidence bounds for approximate answers and have shown that they are a good complement to traditional guaranteed bounds.

Sampling is becoming increasingly essential in data warehousing and other applications. Hence, it is important to eliminate any fundamental problems that limit its applicability to complex queries. This paper identifies one such problem and presents a complete solution to it. As part of this solution, we also develop novel error analysis techniques.

Acknowledgements

The Aqua project is joint work with Yossi Matias, one of the project initiators. Additional contributors to Aqua are Torsten Suel and S. (Muthu) Muthukrishnan. We thank Yossi for his contributions to the results in Section 6, Muthu for providing Lemma 4.5, and Yair Bartal for his work on the proof of Theorem 5.1. We also thank Minos Garofalakis, Hank Korth and Avi Silberschatz for discussions related to this work.

References

[AMS96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proc. 28th ACM Symp. on the Theory of Computing*, pages 20–29, May 1996. Full version to appear in JCSS special issue for STOC'96.

[AZ96] G. Antoshenkov and M. Ziauddin. Query processing and optimization in Oracle Rdb. *VLDB Journal*, 5(4):229–237, 1996.

[BDF$^+$97] D. Barbará, W. DuMouchel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. The New Jersey data reduction report. *Bulletin of the Technical Committee on Data Engineering*, 20(4):3–45, 1997.

[BM96] R. J. Bayardo, Jr. and D. P. Miranker. Processing queries for first-few answers. In *Proc. 5th International Conf. on Information and Knowledge Management*, pages 45–52, November 1996.

[CK98] M. J. Carey and D. Kossmann. Reducing the braking distance of an SQL query engine. In *Proc. 24th International Conf. on Very Large Data Bases*, pages 158–169, August 1998.

[CR94] C. M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 161–172, May 1994.

[GGMS96] S. Ganguly, P. B. Gibbons, Y. Matias, and A. Silberschatz. Bifocal sampling for skew-resistant join size estimation. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 271–281, June 1996.

[GM98a] P. B. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 331–342, June 1998.

[GM98b] P. B. Gibbons and Y. Matias. Selecting estimation procedures and bounds for approximate answering of aggregation queries. Technical report. In preparation., 1998.

[GM98c] P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. To appear in *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, AMS., 1998.

[GMP97a] P. B. Gibbons, Y. Matias, and V. Poosala. Aqua project white paper. Technical report, Bell Laboratories, Murray Hill, New Jersey, December 1997.

[GMP97b] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proc. 23rd International Conf. on Very Large Data Bases*, pages 466–475, August 1997.

[GPA$^+$98] P. B. Gibbons, V. Poosala, S. Acharya, Y. Bartal, Y. Matias, S. Muthukrishnan, S. Ramaswamy, and T. Suel. AQUA: System and techniques for approximate query answering. Technical report, Bell Laboratories, Murray Hill, New Jersey, February 1998.

[Haa96] P. J. Haas. Hoeffding inequalities for join-selectivity estimation and online aggregation. Technical Report RJ 10040, IBM Almaden Research Center, San Jose, CA, 1996.

[Haa97] P. J. Haas. Large-sample and deterministic confidence intervals for online aggregation. In *Proc. 9th International Conf. on Scientific and Statistical Database Management*, August 1997.

[HHW97] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 171–182, May 1997.

[HNS94] P. J. Haas, J. F. Naughton, and A. N. Swami. On the relative cost of sampling for join selectivity estimation. In *Proc. 13th ACM Symp. on Principles of Database Systems*, pages 14–24, May 1994.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. 21st International Conf. on Very Large Data Bases*, pages 311–322, September 1995.

[HÖT88] W.-C. Hou, G. Özsoyoğlu, and B. K. Taneja. Statistical estimators for relational algebra expressions. In *Proc. 7th ACM Symp. on Principles of Database Systems*, pages 276–287, March 1988.

[IP98] Y. Ioannidis and V. Poosala. Histogram-based techniques for approximating set-valued query-answers. Submitted for publication, 1998.

[Koo80] R. P. Kooi. *The Optimization of Queries in Relational Databases*. PhD thesis, Case Western Reserve University, September 1980.

[LN95] R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. *J. Computer and System Sciences*, 51(1):18–25, 1995.

[LNS90] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 1–12, May 1990.

[MSY96] Y. Matias, S. C. Sahinalp, and N. E. Young. Performance evaluation of approximate priority queues. Presented at *DIMACS Fifth Implementation Challenge: Priority Queues, Dictionaries, and Point Sets*, organized by D. S. Johnson and C. McGeoch, October 1996.

[MVN93] Y. Matias, J. S. Vitter, and W.-C. Ni. Dynamic generation of discrete random variates. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, pages 361–370, January 1993.

[MVY94] Y. Matias, J. S. Vitter, and N. E. Young. Approximate data structures with applications. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 187–194, January 1994.

[OR92] F. Olken and D. Rotem. Maintenance of materialized views of sampling queries. In *Proc. 8th IEEE International Conf. on Data Engineering*, pages 632–641, February 1992.

[PIHS96] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 294–305, June 1996.

[Poo97] V. Poosala. *Histogram-based estimation techniques in databases*. PhD thesis, Univ. of Wisconsin-Madison, 1997.

[SAC$^+$79] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. T. Price. Access path selection in a relational database management system. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 23–34, June 1979.

[Sch97] D. Schneider. The ins & outs (and everything in between) of data warehousing. Tutorial in the *23rd International Conf. on Very Large Data Bases*, August 1997.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge - Base Systems*, volume 1. Computer Science Press, Rockville, Maryland, 1988.

[VL93] S. V. Vrbsky and J. W. S. Liu. Approximate—a query processor that produces monotonically improving approximate answers. *IEEE Trans. on Knowledge and Data Engineering*, 5(6):1056–1068, 1993.

A Appendix

Proof. (of Theorem 5.1) Since additional sample points only decrease Equation 1, the optimal solution has $\sum_i n_i s_i = N$. Assume first that $s_i = 1$ for all i. Let r be the number of source relations. Consider the optimal choice of n_i's that minimizes Equation 1. Consider any $j, k \leq r$, $j \neq k$. If we set $n_j = x$ and $C = N - \sum_{l \neq j,k} n_l$ so that $n_k = C - x$ then we have that in the optimal solution the expression

$$\frac{f_j}{\sqrt{x}} + \frac{f_k}{\sqrt{C - x}}$$

is minimized for x. To find the value of x minimizing the expression above we find when its derivative is zero:

$$\frac{1}{2}f_j x^{-3/2} + \frac{1}{2}f_k(C-x)^{-3/2} = 0.$$

Substituting back $n_j = x$ and $n_k = C - x$ we get

$$\frac{n_j}{n_k} = \left(\frac{f_j}{f_k}\right)^{2/3}.$$

Since this holds for every pair $j \neq k$ and $\sum_i n_i = N$, we get that

$$n_i = N\frac{{f_i}^{2/3}}{\sum_j {f_j}^{2/3}}.$$

Now consider the general case where the s_i may be distinct and greater than one. Let $n'_i = n_i s_i$ and $f'_i = f_i\sqrt{s_i}$. Then we can rewrite the expression to minimize as $\sum_i \frac{f'_i}{\sqrt{n'_i}}$ where $\sum_i n'_i = N$. Using the solution derived above we get that the expression is minimized for

$$n'_i = N\frac{{f'_i}^{2/3}}{\sum_j {f'_j}^{2/3}} = N\frac{{f_i}^{2/3}{s_i}^{1/3}}{\sum_j {f_j}^{2/3}{s_j}^{1/3}}.$$

Therefore we have

$$n_i = n'_i/s_i = N\frac{(f_i/s_i)^{2/3}}{\sum_j {f_j}^{2/3}{s_j}^{1/3}}.$$

■

Lemma A.1 *Consider a family of bounds of the form*

$$\Pr(|e_{(k)} - \mu| \geq t) \leq c\delta^{1/k}, \tag{4}$$

for some $c \geq 1$ and some δ, $0 < \delta < 1$. Then for all $k > 1$, the bounds obtained by applying Equation 4 with no chunking are smaller than the bounds obtained by applying Equation 4 to the chunk estimators $e_{(k)}$ and then applying Equation 3 to the median, e^*_k, of the chunk estimators.

Proof. Let $k > 1$ be the number of chunks. We plug in $\rho = 1 - c\delta^{1/k}$ into Equation 3, resulting in

$$\begin{aligned}\Pr(|e^*_k - \mu| \geq t) &= \sum_{i=0}^{\lfloor\frac{k}{2}\rfloor}\binom{k}{i}(1-c\delta^{1/k})^i(c\delta^{1/k})^{k-i}\\ &= c^k\delta + \sum_{i=1}^{\lfloor\frac{k}{2}\rfloor}\binom{k}{i}(1-c\delta^{1/k})^i(c\delta^{1/k})^{k-i} > c\delta,\end{aligned}$$

since each term in the summation is positive when $0 < \delta < 1$, and $c^k \geq c$. Thus the error, δ, without chunking is less than the error taking the median of $k > 1$ chunks. ■

Lemma A.2 *In each of the techniques described in Section 6.2 using chunking and taking the median, the error bound, t, is minimized when all the chunks are the same size.*

Proof. Let k be the number of chunks, and let $n_1, \ldots, n_k$ be the chunk sizes. Then, $\sum_{i=1}^k n_i = n$. For the median technique, the ρ in Equation 2 is determined from Equation 3, based on the given p and k. For this ρ, we wish to minimize the t in Equation 2. Thus we wish to minimize the maximum of the chunk error bounds $t_1, \ldots, t_k$; this is accomplished when all such bounds are equal. In each of the techniques described in Section 6.2, the bounds are functions of ρ, MAX, MIN, σ, $\hat{\sigma}$, and/or n_i. Of these, only n_i may differ from chunk to chunk (e.g., recall that $\hat{\sigma}$ is computed over the entire sample), and t_i is proportional to $1/\sqrt{n_i}$. Thus all t_i are equal precisely when $n_1 = n_2 = \cdots = n_k = n/k$. ■