Rainer Gemulla, Philipp Rösch, Wolfgang Lehner

Linked Bernoulli Synopses: Sampling along Foreign Keys

Erstveröffentlichung in / First published in:

Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong, 09.-11.07.2008. Springer, S. 6-23. ISBN 978-3-540-69497-7.

DOI: https://doi.org/10.1007/978-3-540-69497-7_4

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

Linked Bernoulli Synopses: Sampling along Foreign Keys

Rainer Gemulla, Philipp Rösch, and Wolfgang Lehner

Database Technology Group
Technische Universität Dresden, Germany
{gemulla,roesch,lehner}@inf.tu-dresden.de

Abstract. Random sampling is a popular technique for providing fast approximate query answers, especially in data warehouse environments. Compared to other types of synopses, random sampling bears the advantage of retaining the dataset's dimensionality; it also associates probabilistic error bounds with the query results. Most of the available sampling techniques focus on table-level sampling, that is, they produce a sample of only a single database table. Queries that contain joins over multiple tables cannot be answered with such samples because join results on random samples are often small and skewed. On the contrary, schema-level sampling techniques by design support queries containing joins. In this paper, we introduce *Linked Bernoulli Synopses*, a schema-level sampling scheme based upon the well-known *Join Synopses*. Both schemes rely on the idea of maintaining foreign-key integrity in the synopses; they are therefore suited to process queries containing arbitrary foreign-key joins. In contrast to Join Synopses, however, Linked Bernoulli Synopses correlate the sampling processes of the different tables in the database so as to minimize the space overhead, without destroying the uniformity of the individual samples. We also discuss how to compute Linked Bernoulli Synopses which maximize the effective sampling fraction for a given memory budget. The computation of the optimum solution is often computationally prohibitive so that approximate solutions are needed. We propose a simple heuristic approach which is fast and seems to produce close-to-optimum results in practice. We conclude the paper with an evaluation of our methods on both synthetic and real-world datasets.

1 Introduction

With the huge amount of data stored in current data warehouse environments, it is impracticable to execute queries directly on the database when human interaction is involved. This applies to explorative queries in data mining and decision-support tasks, which are used as a precursor to more complex analysis tasks; the main goal is to determine which methods and parts of data that are likely to produce interesting results and which will not. It also applies to OLAP and report design, where approximate query processing is able to significantly increase the responsiveness of the system and therefore the productiveness of

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

its users. In all these scenarios, random sampling has proven to be a valuable tool for database summarization. Compared to other types of synopses, random sampling is easy to implement and use, it supports a broad range of queries (including grouping) and it provides probabilistic error bounds.

The main problem with most of the available database sampling schemes is that they focus on only a single table in the database; we refer to these schemes as *table-level sampling schemes*. Queries that contain joins between multiple tables are problematic because joins between random samples in general do not result in a random sample of the join of the respective tables [1,2,3]. To avoid this problem, it is crucial that relationships between multiple tables be known and exploited in the sampling process itself. This is accomplished by *schema-level sampling schemes*: these schemes sample multiple relations at once in such a way that it is possible to use the resulting samples to compute results of queries with joins. In this paper, we restrict attention to foreign-key joins. Such joins are ubiquitous in data warehouse scenarios, where most queries join a fact table with multiple dimension tables along predefined foreign-key paths.

The problem of schema-level sampling becomes tractable when only foreign-key joins are of interest. Indeed, Acharya et al. [2] have shown that it is sufficient to maintain a single sample per table to support any potential foreign-key join. The key idea underlying their Join Synopses is to 1) take a sample of each table, 2) join each sample with all tables to which it has foreign keys and 3) store the joined samples in the synopsis. For example, consider a schema with two tables R_1 and R_2, where R_1 has a foreign key to R_2. Let S_1 and S_2 be uniform random samples of R_1 and R_2, respectively. The Join Synopsis then consists of the two samples $S_1 \bowtie R_2$ and S_2. Observe that—by projection on the attributes of R_1—S_1 can be reconstructed from $S_1 \bowtie R_2$; there is a 1:1 relationship between the tuples in these two relations. The synopsis can be used to answer queries on R_1, on R_2 as well as on $R_1 \bowtie R_2$. To reduce the space requirement of the sampling scheme, [2] also suggest to renormalize the join results. After renormalization, the synopsis contains three "samples": S_1, $R_2 \ltimes S_1$ and S_2. Observe that both $R_2 \ltimes S_1$ and S_2 contain tuples from R_2. If samples S_1 and S_2 have a size of, say, $10,000$ tuples each, the entire synopsis consists of up to $30,000$ tuples—a space overhead of 50%.

In this paper, we develop a new schema-level sampling scheme called *Linked Bernoulli Synopses* (LBS), which reduces the space overhead incurred by Join Synopses. In expectation, the size of LBS is at most as large as the size of the corresponding Join Synopsis; it is often much smaller. The key idea behind LBS is to correlate the sampling processes of S_1 and S_2, while maintaining the uniformity of both samples. Intuitively, given a sample of S_1, we try to reuse as many tuples from $R_2 \ltimes S_1$ as possible for the sample S_2. Our synopses are optimal, that is, it is impossible to find a sampling scheme which produces smaller synopses (with the same sampling fractions for each table) in expectation. For example, when the cardinalities of R_1 and R_2 are equal and there is a 1:1 relationship between both tables (the best case), we require a space budget of $20,000$ tuples to sample $10,000$ tuples from each table; the overhead is reduced to 0%.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

We also address the problem of computing a Linked Bernoulli Synopsis which fits into a given space budget. The problem is challenging because all the sampling steps are correlated; changing the size of one sample might change the size of many others. We treat the problem as an optimization problem and show how it can be solved numerically using results from convex optimization. Finding the optimum solution requires time exponential to the number of relations and linear to their size; approximate solutions are therefore key to the practicability of our methods. In fact, we found that a simple heuristic seems to produce near-optimal results in practice, so that it might be unnecessary to run the entire optimization.

The remainder of the paper is structured as follows: In Section 2, we review Join Synopses in more detail and show how their space consumption can be reduced with a simple modification. We then analyze the modified sampling scheme in terms of (expected) space consumption. In Section 3, we introduce and analyze Linked Bernoulli Synopses. Section 4 discusses the problem of allocating the available space to the different tables in the schema. Preliminary results of an evaluation on synthetic and real-world datasets are presented in Section 5. Section 6 gives a brief overview of related work and we conclude the paper with a summary of our results in Section 7.

2 Preliminaries

In this section, we review, discuss and analyze Join Synopses. The results presented in this section drive the design of our Linked Bernoulli Synopsis in Section 3.

2.1 Notation

We start by summarizing the notation used throughout this paper. Let $G = (V, E)$ be a schema graph of a relational database with V being the set of vertices and E being a set of directed edges. In more detail, V is the set of tables in the database, while the set $E \subseteq V \times V$ describes foreign-key relationships. An element $(R_1, R_2) \in E$ with $R_1, R_2 \in V$ represents a foreign-key relationship from R_1 to R_2. R_1 is called parent table or predecessor, while R_2 is called child table or successor. For brevity, we write $R_1 \to R_2$ whenever $(R_1, R_2) \in E$. Moreover, we will use $\Rightarrow$ to denote the transitive closure over $\to$; $\nrightarrow$ and $\not\Rightarrow$ denote the inverse of $\to$ and $\Rightarrow$, respectively. The function $\mathrm{pk}_R(t)$ determines the primary key of a tuple $t \in R$, and $\mathrm{fk}_{R_1 \to R_2}(t)$ determines the foreign key of a tuple $t \in R_1$ to table R_2. Thus, when $R_1 \to R_2$, two tuples $t_1 \in R_1$ and $t_2 \in R_2$ join if $\mathrm{fk}_{R_1 \to R_2}(t_1) = \mathrm{pk}_{R_2}(t_2)$. In this case, we say that t_1 references t_2.

Figure 1 shows the example database that we will use as our running example throughout the paper. The database consists of three tables A, B and C; foreign keys are defined between $A.FK$ and $B.PK$ as well as $B.FK$ and $C.PK$. The foreign-key relationships are graphically encoded using arrows between matching tuples. Using the notation above, we have $V = \{A, B, C\}$ and $E = \{(A, B), (B, C)\}$.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

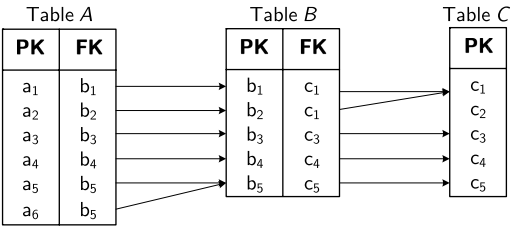


Fig. 1. An example database with 3 tables ($A \to B$ and $B \to C$)

We subsequently assume that the schema graph G is free of cycles, that is, there is no table R with $R \Rightarrow R$. The reason is that—when the schema graph contains cycles—the inclusion of even a single tuple from one of the relations in the cycle might lead to an explosion of the synopsis size. Fortunately, in the setting of data warehouses we are concerned with, cycles in the schema graph do not occur. We also assume that G does not contain multiple edges between two tables; this is not a limitation of our approach but simplifies explanation.

2.2 Join Synopses Revisited

In the following, we take a slightly different view on Join Synopses by incorporating them into the more general concept of schema synopses. In more detail, the *schema synopsis* Ψ_G of a schema graph G consists of a table synopsis for each table in the schema. For brevity, we will omit the schema graph G when referring to the schema synopsis. The *table synopsis* Ψ_R consists of a uniform sample Sample_R and a reference table RefTable_R, both containing items from R. The sample is primarily used for query evaluation, while the reference table is used to maintain foreign-key integrity. In general, the reference table contains all tuples from R which (1) are referenced by a table synopsis of a predecessor of R and (2) are not stored in the sample already. As a matter of notation, we say $t \in \Psi_R$ whenever $t \in \text{Sample}_R \cup \text{RefTable}_R$.

Join Synopses can now be viewed as a special kind of schema synopsis. The algorithm is as follows [2]:

1. Sample each relation independently using Bernoulli sampling. In Bernoulli sampling with sampling rate q, each tuple is included into the sample with probability q and excluded with probability $1 - q$; the process is repeated independently for each tuple. The parameter q essentially controls the desired sample size; see Section 4.
2. Fill the reference tables so that foreign-key integrity is restored. The tables are processed top-down, that is, a table R is processed only after all its predecessors have been processed already. Note that the second step slightly differs from the original Join Synopses because we do not store the same tuple in both the sample and the reference table. For large tables, the resulting space reduction is usually negligible for Join Synopses but sets the foundation for the space reduction possible with Linked Bernoulli Synopses.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

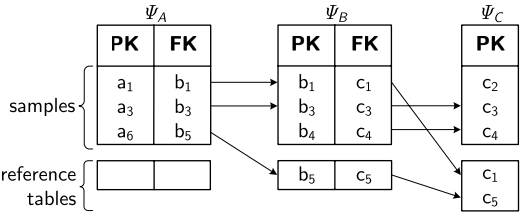


Fig. 2. Join Synopsis with a sampling rate of 50%

Note that both phases of the above algorithm can be interweaved, that is, we can compute both the sample and the reference table of each table in a single scan.

An example for a Join Synopsis of our example database is given in Figure 2. We used a sampling rate of $q = 50\%$ for each table. The reference tables are given below the individual samples. As can be seen, reference tables restore foreign-key integrity. For example, tuple $(a_6, b_5) \in A$ references $(b_5, c_5) \in B$, but the latter is not stored in the sample so that it has been included in the reference table.

2.3 Analysis of Join Synopses

We now analyze the space consumption of the modified Join Synopses described above.[1] Let $R_1, R_2, \ldots, R_k$ be the direct predecessors of a table R. During Join Synopses computation, R is processed after the processing of $R_1, \ldots, R_k$ has completed. Now, let $\mathrm{isRef}_{R_i \to R} : R \to \{\mathrm{true}, \mathrm{false}\}$ for $1 \le i \le k$ be a function which evaluates to true if a tuple $t \in R$ has been referenced by Ψ_{R_i} and to false otherwise. Also set

$$\mathrm{isRef}_R(t) = \mathrm{isRef}_{R_1 \to R}(t) \vee \mathrm{isRef}_{R_2 \to R}(t) \vee \cdots \vee \mathrm{isRef}_{R_k \to R}(t)\,,$$

so that $\mathrm{isRef}_R(t)$ evaluates to true if t is referenced by any predecessor.

To determine the space consumption of the Join Synopsis algorithm, we view $\mathrm{isRef}_R(t)$ as a random function (over the choices made when sampling the predecessors). Our goal is to compute the *reference probability* $\mathrm{pRef}_R(t)$ that tuple t is referenced from any predecessor of R. Assume for a moment that $\mathrm{pRef}_R(t)$ is known for a tuple $t \in R$. The *selection probability* $\mathrm{pSel}_R(t)$ that tuple t is included into the synopsis—either by acception into the sample or by addition to the reference table—is then given by

$$\begin{aligned}\mathrm{pSel}_R(t) &= P(t \in \Psi_R) = P(t \in \mathrm{Sample}_R \vee t \in \mathrm{RefTable}_R)\\ &= P(t \in \mathrm{Sample}_R) + P(t \notin \mathrm{Sample}_R) P(t \in \mathrm{RefTable}_R \,|\, t \notin \mathrm{Sample}_R)\\ &= q_R + (1 - q_R)\,\mathrm{pRef}_R(t),\end{aligned}$$

where q_R is the sampling fraction used for table R. Here, the final equality follows from the independence of the sampling step and the event that the tuple

[1] The analysis is fair with respect to the original algorithms because our modification of the Join Synopsis algorithm leads to a decrease of the space required to store the synopsis.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

is being referenced. We now have all the information we need to compute reference probabilities. Suppose that table R has predecessor R' and denote by $\mathrm{pRef}_{R' \to R}(t)$ the probability that a tuple $t \in R$ is referenced by the synopsis of R'. Since the tuples in R' are sampled independently from each other, this probability is given by:

$$\mathrm{pRef}_{R' \to R}(t) = 1 - \prod_{\substack{t' \in R' \\ \mathrm{fk}_{R' \to R}(t') = \mathrm{pk}_R(t)}} \left(1 - \mathrm{pSel}_{R'}(t')\right).$$

Note that $\mathrm{pRef}_{R' \to R}(t)$ can be computed incrementally, that is, with only a single scan of R'. To see this, suppose that we have already processed a subset R'_0 of the tuples in R' and let $t^+ \in R' \setminus R'_0$ be the currently processed tuple. If t^+ does not reference t, we can simply ignore it. Otherwise, we can use the following relationship

$$\begin{aligned}
\mathrm{pRef}_{R'_0 \cup \{t^+\} \to R}(t) &= 1 - \prod_{\substack{t' \in R'_0 \cup \{t^+\} \\ \mathrm{fk}_{R' \to R}(t') = \mathrm{pk}_R(t)}} \left(1 - \mathrm{pSel}_{R'}(t')\right) \\
&= 1 - \left(1 - \mathrm{pSel}_{R'}(t^+)\right) \prod_{\substack{t' \in R'_0 \\ \mathrm{fk}_{R' \to R}(t') = \mathrm{pk}_R(t)}} \left(1 - \mathrm{pSel}_{R'}(t')\right) \\
&= 1 - \left(1 - \mathrm{pSel}_{R'}(t^+)\right) \left(1 - \mathrm{pRef}_{R'_0 \to R}(t)\right) \qquad (1)
\end{aligned}$$

to update the reference probability. If R' is the sole predecessor of R, then $\mathrm{pRef}_R(t) = \mathrm{pRef}_{R' \to R}(t)$. The discussion of the computation of $\mathrm{pRef}_R(t)$ for tables with multiple predecessors is deferred to Section 3.

Figure 3a shows the selection and reference probabilities for Join Synopses with a sampling fraction of $q = 50\%$ for each table. The reference probabilities are annotated on the arrows, while the selection probabilities are given right next to each tuple (rounded to one digit after decimal point). As can be seen, the selection probabilities—which effectively determine the size of the synopsis—are larger than both the sampling fraction q and the reference probability. The reason is that the sampling steps for each table are performed independently of each other.

Given the sampling fraction q_R, the expected size of the table synopsis of R is given by

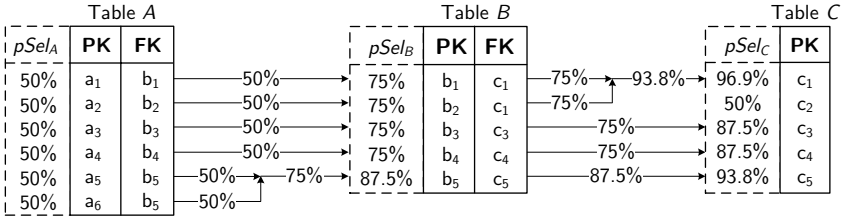
$$E[|\Psi_R|] = \sum_{t \in R} \mathrm{pSel}_R(t) = q_R|R| + (1 - q_R) \sum_{t \in R} \mathrm{pRef}_R(t).$$

In expectation, the entire Join Synopsis consists of

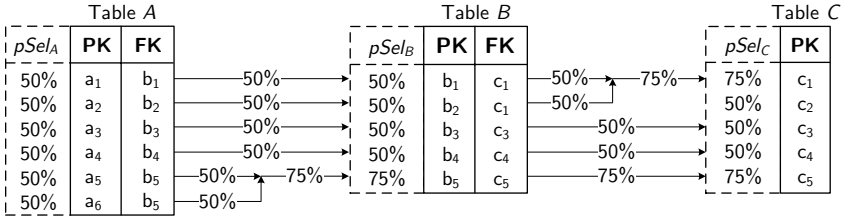
$$E[|\Psi|] = \sum_{R \in V} E[|\Psi_R|] \qquad (2)$$

tuples by the linearity of expected value. In our example, we have $E[|\Psi_A|] = 3$, $E[|\Psi_B|] \approx 3.88$, $E[|\Psi_C|] \approx 4.16$. The expected total synopsis size $E[|\Psi|]$ is 11.04 tuples.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4



(a) Join Synopsis



(b) Linked Bernoulli Synopsis

Fig. 3. Reference and selection probabilities for $q = 50\%$. Reference probabilities are annotated on the arrows; tuples with no incoming edges have zero reference probability.

3 Linked Bernoulli Synopses

Linked Bernoulli Synopses are based on Join Synopses, but the synopsis computation is entirely different. The key observation leading to our LBS is that the event of a tuple being referenced by a predecessor already contains some randomness, which can be exploited for sampling. If a tuple is referenced, we *have to* include it into either the sample or the reference table; that is, we have to store it anyway. A tuple in a sample, however, is more valuable because it can be used directly for query answering: larger samples lead to better results. The tuples in the reference tables can be seen as overhead because they are "only" used to preserve foreign-key integrity. Thus, we would like to bias the sample towards referenced tuples, so that the overhead in the reference tables is minimized. LBS perform this biasing in such a way that the resulting sample is still uniform, so that it can be used for query processing as before. In contrast to Join Synopses, however, the samples in LBS are correlated. This correlation does not lead to problems at query time because only one sample is used to answer each query (the sample of the base table of the join) and each sample on its own is a uniform random sample of the respective table.

3.1 Algorithmic Description

We are now ready to describe the computation of Linked Bernoulli Synopses in full detail. The general procedure is as follows:

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

1. Scan the tables in top-down order, that is, whenever a table R is processed, all its predecessors must have been processed already.
2. For each tuple t, decide whether or not the tuple is included into either the sample or the reference table. This decision is based on (1) the reference probability $\mathrm{pRef}_R(t)$ of the tuple and (2) the fact of whether or not the tuple is referenced by a predecessor ($\mathrm{isRef}_R(t)$ is true). Simultaneously, compute (or update) the reference probabilities for every successor of R using equation (1).

The crux of LBS lies in step 2, where we make use of $\mathrm{pRef}_R(t)$ and $\mathrm{isRef}_R(t)$ to drive the sampling process. We now discuss this step in more detail. The key idea is to compare the reference probability $\mathrm{pRef}_R(t)$ with the desired sampling fraction q_R. For each tuple t, there are three cases:

Case 1: $\mathrm{pRef}_R(t) = q_R$, that is, the reference probability and the sampling fraction are equal. In this case, we add the tuple to the sample if and only if it is referenced. Otherwise the tuple is ignored. It follows immediately that $P(t \in \mathrm{Sample}_R) = q_R$.

Case 2: $\mathrm{pRef}_R(t) < q_R$, that is, the reference probability is smaller than the sampling fraction. We directly add t to the sample whenever it is referenced. When t is not referenced, we add it to the sample with probability $(q_R - \mathrm{pRef}_R(t)) \,/\, (1 - \mathrm{pRef}_R(t))$ or ignore it otherwise. The probability that t is included into the sample is given by:

$$P(t \in \mathrm{Sample}_R) = P(\mathrm{isRef}_R(t) = \mathrm{true}) + P(\mathrm{isRef}_R(t) = \mathrm{false}) \frac{q_R - \mathrm{pRef}_R(t)}{1 - \mathrm{pRef}_R(t)}$$
$$= \mathrm{pRef}_R(t) + (1 - \mathrm{pRef}_R(t)) \frac{q_R - \mathrm{pRef}_R(t)}{1 - \mathrm{pRef}_R(t)} = q_R.$$

Case 3: $\mathrm{pRef}_R(t) > q_R$, that is, the reference probability is larger than the sampling fraction. If t is not referenced, we can safely ignore it. If t is referenced, we add it to the sample with probability $q_R / \mathrm{pRef}_R(t)$ or to the reference table otherwise. The probability that t is added to the sample is:

$$P(t \in \mathrm{Sample}_R) = P(\mathrm{isRef}_R(t) = \mathrm{true}) \frac{q_R}{\mathrm{pRef}_R(t)} = \mathrm{pRef}_R(t) \frac{q_R}{\mathrm{pRef}_R(t)} = q_R.$$

This case is the "bad case" because there is a non-zero probability that t is added to the reference table.

To sum up, the tuple is included into the sample with the desired probability of q_R in each of the three cases. Since both the inclusion/exclusion decisions as well as the event of being referenced are independent among the tuples, the algorithm produces a Bernoulli sample with sampling rate q_R. The selection probability $\mathrm{pSel}_R(t)$, that is, the probability that a tuple t is stored in either the sample or the reference table, is given by:

$$\mathrm{pSel}_R(t) = \max\{q, \mathrm{pRef}_R(t)\}.$$

The reference probabilities for LBS are computed from the selection probabilities as before.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

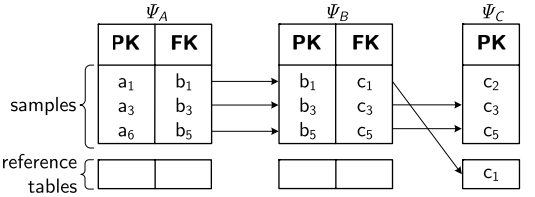


Fig. 4. Linked Bernoulli Synopsis with a sampling rate of 50%

3.2 Example and Analysis

Figure 3b shows the selection and reference probabilities for LBS with a sampling rate of $q = 50\%$ for each table. By inspection, one finds that—for tables B and C—the selection probabilities for LBS are lower than the selection probabilities of Join Synopses; we will formalize this observation below. The expected synopsis sizes are: $E[|\Psi_A|] = 3$, $E[|\Psi_B|] = 2.75$, $E[|\Psi_C|] = 3$ and therefore $E[|\Psi|] = 8.75$ (instead of 11.03).

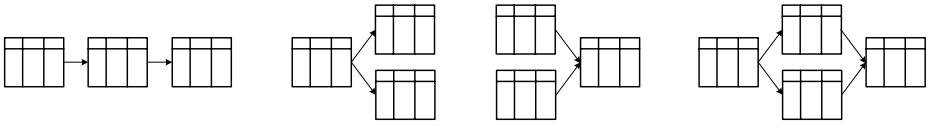
A potential LBS is shown in Figure 4. Compared to Join Synopses (Figure 2), there is no difference in the sample of table A because A does not have any predecessors (all tuples trivially belong to case 2). When sampling table B, we compare the desired sampling rate of 50% with the reference probabilities given in Figure 3b. Tuples b_1 through b_4 all belong to case 1 above, that is, they are included if and only if they are referenced. In the example, this holds true for only b_1 and b_3; b_2 and b_4 are ignored. Note that—given the sample of table A—this process is entirely deterministic. For tuple b_5, the reference probability of 75% is larger than the desired sampling fraction; this is case 3 and—since b_5 is referenced—it is accepted into the sample with probability 2/3 (as in the example) or in the reference table with probability 1/3. Continuing with table C, both c_1 and c_5 belong to case 3. Both tuples are referenced; c_1 has been added to the reference table (probability of acceptance into sample: $\approx 52\%$) and c_5 to the sample ($\approx 53\%$). Tuples c_3 and c_4 belong to case 1, but only c_3 is referenced and therefore added to the sample. Finally, tuple c_2 belongs to case 2 because it has a reference probability of zero. It is accepted into the sample with a probability of 50%.

We now compare formally the selection probability of a tuple using Join Synopses with the selection probability using LBS when using the same sampling rate q_R for both. Assuming that the reference probabilities are the same for both approaches, we have

$$\begin{aligned}
\mathrm{pSel}_R^{\mathrm{JS}}(t) - \mathrm{pSel}_R^{\mathrm{LBS}}(t) &= q + (1-q)\,\mathrm{pRef}_R(t) - \max\{q, \mathrm{pRef}_R(t)\} \\
&= (1-q)\,\mathrm{pRef}_R(t) - \max\{0, \mathrm{pRef}_R(t) - q\} \\
&= \begin{cases} (1-q)\,\mathrm{pRef}_R(t) & \mathrm{pRef}_R(t) \le q \\ q(1-\mathrm{pRef}_R(t)) & \text{otherwise} \end{cases} \\
&\ge 0.
\end{aligned}$$

For $0 < q, \mathrm{pRef}_R(t) < 1$ the inequality becomes strict. In general, the reference probabilities for Join Synopses and Linked Bernoulli Synopses will be different.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4



(a) Chain pattern (b) Split pattern (c) Merge pattern (d) Diamond pattern

Fig. 5. Reference patterns

Using the argument above, it is straightforward to show that the reference probability of Linked Bernoulli Synopses is always smaller than or equal to that of Join Synopses.

3.3 Handling Multiple Predecessors

Until now, we have assumed that every table has at most one predecessor. This assumption is clearly too restrictive in practice. In this section, we discuss how to handle tables with multiple predecessors. The key question we are going to answer is how to compute the reference probability $\mathrm{pRef}_R(t)$ from the reference probabilities $\mathrm{pRef}_{R_i \to R}$ (with R_i being a predecessor of R). To that extent, we distinguish the 4 possible patterns shown in Figure 5. As before, we assume that the schema graph is free of cycles.

In a *chain pattern*, each table has at most one predecessor and at most one successor. This is exactly the situation we looked at in the preceding sections. In a *split pattern*, each table has at most one predecessor but arbitrarily many successors. Again, the formulas established in the preceding sections can be used directly. In a *merge pattern*, each table has arbitrarily many predecessors and at most one successor. Fix a table R and denote by $R_1, \ldots, R_k$ the direct predecessors of R. Since the schema graph is free of cycles, tables $R_1, \ldots, R_k$ do not have any common predecessor. They are therefore sampled independently of each other. It follows that the reference probabilities from each of the R_i to table R are independent and thus

$$\mathrm{pRef}_R(t) = 1 - \prod_{i=1}^{k} \left(1 - \mathrm{pRef}_{R_i \to R}(t)\right). \tag{3}$$

Finally, in a *diamond pattern*, at least two of the predecessors $R_1, \ldots, R_k$ of R share a common predecessor, say R_1 and R_2. In this case, the event that a tuple from R is referenced from R_1 and the event that the same tuple is referenced from R_2 are not necessarily independent. As a consequence, equation (3) cannot be used. In the remainder of this section, we have a closer look at the dependencies introduced in diamond patterns and propose possible workarounds.

We will use the example in Figure 6 for illustration purposes. The example shows 4 tables A, B, C and D, which are arranged in a diamond pattern. The sampling rate has been set to 50% for all tables. Suppose that tables A, B and C have been sampled already and that table D is about to be processed. There are two problems which might occur in this setting:

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

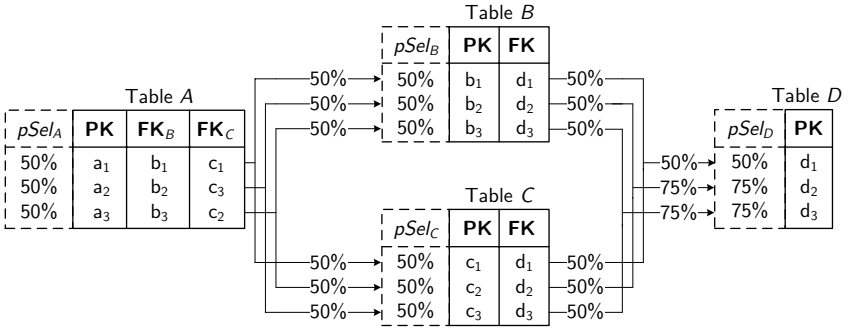


Fig. 6. Reference probabilities for a 50%-sample, dependent references

– *Reference probabilities.* The references to a tuple t might be dependent. In the example, tuple d_1 is referenced whenever either b_1 or c_1 is included into the synopses. The selection probabilities of b_1 and c_1 are 50% each, so that the application of formula (3) indicates that d_1 is referenced with a probability of 75%. However, since in a LBS the samples are correlated, both b_1 and c_1 are included into the sample of tables B and C, respectively, if and only if a_1 has been included in the sample of table A. This event occurs with 50% probability, so that the true reference probability of d_1 is given by $\mathrm{pRef}_D(d_1) = 50\%$.
– *Joint inclusion probabilities.* A more subtle problem is that of joint inclusion probabilities. In the example, both d_2 and d_3 are referenced with a probability of 75%. The references from tables B and C to tuple d_2 are independent, as are the references to tuple d_3. However, if one looks at both d_2 and d_3 simultaneously, one finds that d_2 is referenced whenever d_3 is and vice versa. As a consequence, the sample is biased towards the cases where (1) both d_2 and d_3 are sampled and (2) neither d_2 nor d_3 is sampled. For example, if we ignore the dependencies between the references and proceed as in the previous sections, the joint inclusion probability of d_2 and d_3 is $\approx 44\%$ but should be 25%.

A trivial way of handling the above problems would be to store table D in its entirety. Though simple, this approach is viable in scenarios where table D is very small. For instance, in the schema of the TPC-H benchmark, the only table which is referenced in a diamond pattern is NATION, and this table consists of only 25 tuples. Otherwise, if table D is too large to store it in its entirety, we see two possible solutions: (1) switch back to Join Synopses for table D and all its successors and (2) decide on a per-tuple basis whether to switch back to the Join-Synopses way of sampling or not. The first solution might work well if table D does not have large successors; its main advantage is its simplicity. The second solution is more sophisticated and requires more bookkeeping, but it may reduce the overall space consumption significantly. We omit further details due to lack of space; a detailed description of the second solution can be found in [4].

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

4 Computing a Synopsis with a Memory Bound

In the previous section, we assumed that the desired sampling rate for each table is given beforehand. In practice, however, it might be difficult to decide on the values of the individual sampling fractions. A more realistic approach is to start with a space budget and to automatically set the sampling fractions so that the space budget is not exceeded and the sample sizes are maximized. To simplify the ongoing discussion, we assume that the space budget is given in number of tuples. That is, for a given budget M, the goal is to find sampling fractions which ensure that $|\Psi| \leq M$ with high probability.

4.1 An Optimization Problem

Suppose that the schema contains tables $R_1, \ldots, R_n$. Denote by $q_1, \ldots, q_n$ the sampling fraction used for each respective table, and let $q = (q_1, \ldots, q_n)$ denote a vector of these sampling fractions. There are many possible choices for q and we have to quantify which choices are considered good and which are not. Suppose that there is a function f so that $f(q) > f(q')$ whenever the sampling rates in q are considered more valuable than those in q'. We can now treat the problem as an optimization problem, that is, we want to find a vector q^* which maximizes the objective function $f(q)$ with respect to the constraint $g(q) \leq M$, where $g(q)$ encodes the (expected) space budget. Using these two functions, the optimization problem can be stated as:

$$\begin{aligned} &\text{Maximize} \\ &\qquad f(q_1, \ldots, q_n) \\ &\text{with respect to} \\ &\qquad 0 < q_1, \ldots, q_n \leq 1 \\ &\qquad g(q_1, \ldots, q_n) \leq M. \end{aligned}$$

The function f can be derived from workload information or from information about the intended usage of the synopsis. In the absence of such information, a suitable choice for f is the geometric mean of the individual sampling fractions or, even simpler, its n-th power:

$$f_{GEO}(q_1, \ldots, q_n) = q_1 q_2 \cdots q_n.$$

Some insight into this choice for f is given in the next section. In any case, we make the assumption that both f and g are monotonically increasing functions of q. In other words, for any $\Delta q = (\Delta q_1, \ldots, \Delta q_n)$ with $\Delta q_i \geq 0$, we assume that

$$f(q + \Delta q) \geq f(q) \qquad \text{and} \qquad g(q + \Delta q) \geq g(q).$$

This assumption virtually always holds in practice because larger sampling rates lead to larger samples (f) which in turn lead to a larger synopsis size (g).

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

The monotonicity of f and g introduces structure into the optimization problem, which can be exploited for solving it. In [5], Tuy proposes an outer-approximation algorithm for monotonic optimization called the *polyblock algorithm*. The time complexity of the polyblock algorithm is exponential in the number of tables, so that it can only be used when the number of tables is not too large. But even when the number of tables is small, the polyblock algorithm requires frequent evaluations of the constraint function g. Exact computation of g according to equation (2) is expensive because a table scan of every table in the schema is required. Therefore, for large problem sizes, it is impractical to compute the optimum solution and approximate algorithms are needed.

4.2 A Heuristic Solution

Our heuristic solution is based on two simplifications. First, we do not compute g exactly but make use of a lower bound g_l for which a closed-form expression exists and which can be evaluated quickly without accessing the database. It is easy to see that

$$g_l(q_1, \ldots, q_n) = |R_1|q_1 + \cdots + |R_n|q_n$$

provides the desired lower bound; we simply ignore the size of the reference table. As a consequence, replacing g by g_l will produce oversized synopses. Depending on the data, this may or may not be significant; we show below that the size of the reference tables is often negligible when f_{GEO} is used as the objective function. Second, the optimum solution can be computed analytically for the combination of g_l and f_{GEO}. To see this, consider an n-dimensional hypercube with edges of length $q_1|R_1|, \ldots, q_n|R_n|$. Then, f_{GEO} is proportional to the volume of the hypercube, which in turn is maximized when all edges have equal length. It follows the f_{GEO} is maximized when

$$q_i \propto \frac{1}{|R_i|}$$

for $1 \leq i \leq n$.[2] We refer to this allocation scheme as *equi-size allocation* because—when the reference tables are ignored—the same number of tuples is sampled from every table in expectation. For traditional Join Synopses, the equi-size allocation scheme is known to produce good results [2].

In the following, we argue that the size of the reference tables is often negligible for equi-size allocation. To see this, consider two tables R_1 and R_2 with $R_1 \rightarrow R_2$ and set $r = |R_1|/(|R_1| + |R_2|)$. For a space budget of M tuples and equi-size allocation, we set $q_1 = rM/|R_1|$ and $q_2 = (1-r)M/|R_2|$. There is no reference table for R_1, so that we focus on R_2. Recall that a tuple $t \in R_2$ is added to the reference table if and only if $\text{pRef}_{R_2}(t) > q_2$ (case 3 in Section 3). Perhaps surprisingly, all tuples from R_2 that are referenced up to $k = |R_1|/|R_2|$ times will not be added to the reference table. To see this, start from the Bernoulli

[2] When one of the q_i exceeds 1, we set it to 1 and repeat the process for the remaining sampling fractions.

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

inequality $(1+x)^k > 1+kx$ and set $x = -q_1 = -q_2/k$. It immediately follows that $1-(1-q_1)^k < q_2$. The expression on the left hand side is equal to $\mathrm{pRef}_{R_2}(t)$ when t is referenced exactly k times; the inequality also holds when t is referenced fewer than k times. Thus, only tuples which are referenced $k+1$ or more times have a non-zero chance of being included in the reference tables. There are at most $|R_1|/(k+1)$ such tuples and only some of them are added to the reference table. Since dependent tables are typically smaller than their parents, the number of tuples in the reference table is expected to be low.

5 Experiments

We ran a variety of experiments in order to evaluate the effectiveness of Linked Bernoulli Synopses. Most of the experiments directly compare Join Synopses (JS) with Linked Bernoulli Synopses (LBS); the main issue we are trying to address is the extent to which LBS are able to reduce the overhead for storing reference tables. We also evaluate how close the equi-size allocation scheme comes to optimum allocation in terms of resulting sample sizes and query accuracy.

5.1 Experimental Setup

We implemented JS and LBS on top of DB2 using Java 1.6. The experiments were conducted on an Athlon AMD XP 3000+ system running Linux with 2 GB of main memory.

We make use of both synthetic and real-world data. The synthetic datasets are based on the TPC-H database of 1GB size. We used a Zipfian distribution for both values (prices, quantities, etc.) and foreign keys. We fixed the skew parameter for values to $z = 0.5$; the skew parameter for foreign keys is modified across the experiments. For our real-world experiments, we make use of the CDBS database[3]. The database contains information about radio and television broadcast services in the United States; only the 14 radio-related tables (without comment tables) were used in our experiments. The sizes of the tables range from $28,000$ to 1.4 million tuples.

5.2 Space Consumption

In a first set of experiments, we evaluated the effectiveness of LBS in comparison to JS on both synthetic and real-world datasets. We computed both synopses for various sampling fractions and datasets and recorded the space overhead required for the reference tables. The space overhead is defined as the size of the reference tables with respect to the size of just the samples, that is, we determine how much space is used for non-sample tuples. We used the equi-size allocation scheme throughout all the experiments.

[3] `http://www.fcc.gov/mb/cdbs.html`

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

Synthetic Data. Our experiments on synthetic datasets try to determine the key factors that influence the overhead of JS and LBS. We generated several TPC-H datasets with different parameters to examine the impact of skew in the foreign-key columns. We also experimented with varying synopsis sizes. For simplicity, we define the synopsis size as the size of just the sample part of the synopsis with respect to the size of the original tables.

Data skew. In a first experiment, we only consider the orders (O) and customer (C) table of the TPC-H schema. We varied the skew parameter of the foreign keys ($O \to C$) from 0 (uniformly distributed) to 1 (heavily skewed); each tuple of C is referenced at least once. We used a sampling fraction of 0.55% for O and 5.5% for C; these settings correspond to equi-size allocation with a synopsis size of 1%. The results are shown in Figure 7a. JS have a high overhead on uniformly distributed keys, but the overhead decreases with increasing skew. The reason for this behavior is that, when the skew is low, almost every tuple in the sample of O corresponds to a different customer, which in turn has to be added to the reference table. When the skew is high, however, a large subset of the orders are placed by only a small subset of the customers; the number of distinct foreign keys in the sample of O therefore decreases in expectation. The overhead of LBS is consistently smaller than the overhead of JS. For a skew value of $z = 0$, no reference tables are needed at all; see the discussion at the end of Section 4.2. With increasing skew, some tuples are referenced with a higher probability than their desired sampling rate (case 3), so that they are added to the reference tables from time to time. If the skew increases further, the number of referenced tuples decreases rapidly and the same effect as for JS can be observed.

Number of unreferenced tuples. In the next experiment, we proceeded as before but modified the fraction f of unreferenced customers. A fraction of $f = 40\%$ means that 40% of the customers did not place any order. For the remaining customers, we used a skew parameter of $z = 0.5$. Figure 7d plots the space overhead for various choices of f. For JS, the space overhead decreases as the f increases. The reason is that the number of distinct customers in the sample of O drops as f increases so that less space is required for reference tables. LBS performs better when the values of f are not too extreme. When the value of f increases, so does the space overhead because more and more customers are referenced with probability larger than the sampling fraction (case 3).

Number of tables. We next evaluated the impact of the number of tables. We started with just lineitem and orders and subsequently added customer, part-supp, part and supplier (in this order). The total size of the synopsis was set to 1%. The skew parameter was set to $z = 0.5$. The results are shown in Figure 7b. As can be seen, LBS outperform JS, especially when the number of tables is high. The reason is that an increasing number of tables lead to an increasing number of transitive references, which have to be stored in the reference tables. For Linked Bernoulli Synopses, this effect is reduced to a minimum.

Synopsis size. In a final experiment, we evaluated the impact of the synopsis size when sampling the 6 tables mentioned above. Figure 7c plots the space overhead

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

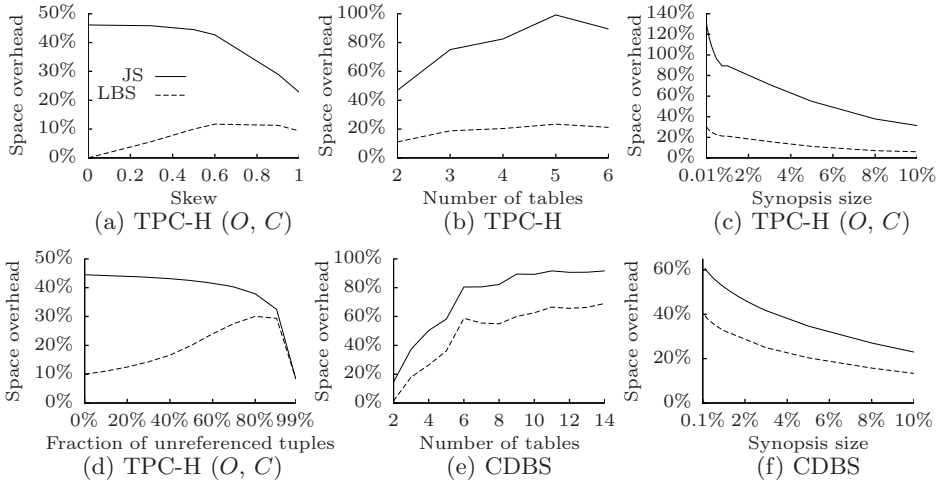


Fig. 7. Space overhead for Linked Bernoulli Synopses and Join Synopses

in dependency of the size of the sample part of the synopsis. As can be seen, the overhead decreases with increasing sample size because more and more tuples qualify for the sample and therefore do not have to be stored in the reference tables. Especially for small synopses, LBS have a significantly smaller overhead.

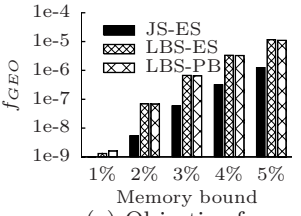
Real-world Data. We now report our results on the CDBS data. We modified the synopsis size between 0.1% and 10%. As can be seen in Figure 7f, the space overhead decreases with increasing synopsis size for both JS and LBS. The difference between the two is not as dramatic as it has been in the synthetic datasets because the CDBS tables contain large numbers of unreferenced tuples (up to 90%). Figure 7e shows the influence of the number of tables for a synopsis size of 1%. Again, LBS has lower overhead than JS.

5.3 Memory Bounds

In a final experiment, we computed synopses that fit into a prespecified amount of space. We used all tables of the TPC-H database; the nation and region table have been sampled entirely. The foreign-key skew was set to $z = 0.5$ and f_{GEO} was used as objective function. We ran three different combinations: JS with equi-size allocation (JS-ES), LBS with equi-size allocation (LBS-ES), and LBS with polyblock allocation (LBS-PB). In the latter case, we restricted the number of steps of the optimization algorithm to $1,000$; this corresponds to $1-2$ days of computation. In contrast, synopsis computation with equi-size allocation is a matter of minutes. The memory bound (samples and reference tables) was varied from 1% to 5%.

Figure 8 plots the objective function f_{GEO} for each combination and memory bound (log plot). The value of the objective function increases with an increasing synopsis size because more space is available for samples. In all cases, the

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4



(a) Objective func.

	Q_1	Q_2	Q_3	Q_4
JS	3.51%	3.95%	3.28%	0.18%
LBS	2.69%	3.06%	2.43%	0.14%
	(−23.4%)	(−22.5%)	(−25.9%)	(−22.2%)

(b) Average relative error (1% size)

Fig. 8. Memory-bounded synopses (synthetic data)

LBS-based approaches achieve significantly larger values of the objective function than JS; they are between 8.9 to 20.8 times larger. Comparing LBS-ES and LBS-PB, one finds that both approaches perform similarly. In an additional experiment, we allowed 14,000 optimization steps for LBS-PB with a 1% memory bound (16 days) and found that the resulting value of the objective function was roughly 50% above the one achieved by LBS-ES. Thus, there is room for improvement, but the high computational cost of LBS-PB renders it impractical.

A different view of the results for a 1% memory bound is given in Figure 8b, where we compare the estimation error achieved by both JS and LBS with equi-size allocation. The resulting per-table sample size was 7,407 tuples and 12,296 tuples, respectively. The columns denote the average relative error of the approximate answer for four different queries (over 1,000 independent runs). Q_1 determines the average order value of customers from Germany ($O \bowtie C \bowtie N$), Q_2 the average balance of these customers ($C \bowtie N$), Q_3 the turnover generated by European suppliers ($L \bowtie PS \bowtie S \bowtie N \bowtie R$), and Q_4 computes the average retail price of a part (P). As can be seen in the figure, the increase in sample size for LBS is directly reflected in the precision of the estimates.

6 Related Work

There exists a variety of sampling techniques for approximate query processing. These techniques can be divided into table-level and schema-level sampling schemes.

Table-level sampling. A table-level sample represents a single table (or view) of a database. Most research focuses on sampling techniques which produce good or optimal samples for a specific purpose such as aggregation queries [6,7] or group-by queries [8,9]. It might be possible to combine some of these techniques with the ideas presented in this paper. For example, a combination of LBS with the weighted sampling scheme in [6] requires an appropriate adjustment of the reference probabilities.

Schema-level sampling. Schema-level samples summarize more than one table as well as the relationships between them. The difficulty of joins over random samples is examined in [3,2]. Acharya et al. [2] also propose the Join Synopsis algorithm from which our Linked Bernoulli Synopses have been derived. In

Final edited form was published in "Scientific and Statistical Database Management. 20th International Conference, SSDBM 2008. Hong Kong 2008", S. 6-23, ISBN 978-3-540-69497-7
https://doi.org/10.1007/978-3-540-69497-7_4

contrast to Join Synopses, Linked Bernoulli Synopses correlate the individual samples so that the space consumption of the synopses is minimized.

Other schema-level synopses. Apart from sampling, other synopses have been proposed for approximate query processing over joins. In [10], Spiegel and Polyzotis propose the *Tuple Graph* as a data structure which is able to represent complex relations between tables. *Probabilistic Relational Models* [11] also exhibit statistical dependencies between attributes of multiple tables. Both techniques focus on selectivity estimation of complex queries but are not applicable to approximate query processing.

7 Conclusion

In this paper, we introduced a novel schema-level sampling scheme called Linked Bernoulli Synopses. The scheme computes a uniform sample of every table in the database; foreign-key integrity is maintained for all sampled tuples. Our approach is based on Join Synopses but correlates the sampling processes of the individual tables. As a consequence, the size of the resulting synopsis is significantly reduced without affecting the quality of approximate answers. Indeed, the saved space can be used to store larger samples, which in turn decreases the estimation error.

References

1. Olken, F.: Random Sampling from Databases. Ph.d. thesis, Lawrence Berkeley National Laboratory (1993)
2. Acharya, S., Gibbons, P.B., Poosala, V., Ramaswamy, S.: Join Synopses for Approximate Query Answering. In: SIGMOD, pp. 275–286 (1999)
3. Chaudhuri, S., Motwani, R., Narasayya, V.: On Random Sampling over Joins. In: SIGMOD, pp. 263–274 (1999)
4. Gemulla, R., Rösch, P., Lehner, W.: Linked Bernoulli Synopses: Sampling Along Foreign Keys (Full Version). Technical report (2007), `http://wwwdb.inf.tu-dresden.de/publications`
5. Tuy, H.: Monotonic optimization: Problems and solution approaches. SIAM J. on Optimization 11(2), 464–494 (2000)
6. Chaudhuri, S., Das, G., Datar, M., Narasayya, R.M.V.R.: Overcoming Limitations of Sampling for Aggregation Queries. In: ICDE, pp. 534–544 (2001)
7. Rösch, P., Gemulla, R., Lehner, W.: Designing Random Sample Synopses with Outliers. In: ICDE (2008)
8. Acharya, S., Gibbons, P., Poosala, V.: Congressional Samples for Approximate Answering of Group-By Queries. In: SIGMOD, pp. 487–498 (2000)
9. Babcock, B., Chaudhuri, S., Das, G.: Dynamic Sample Selection for Approximate Query Processing. In: SIGMOD, pp. 539–550 (2003)
10. Spiegel, J., Polyzotis, N.: Graph-Based Synopses for Relational Selectivity Estimation. In: SIGMOD, pp. 205–216 (2006)
11. Getoor, L., Taskar, B., Koller, D.: Selectivity Estimation using Probabilistic Models. In: SIGMOD, pp. 461–472 (2001)