Join Size Estimation Using Correlated Sampling Final Report

Mengjie Pan
mpan1@uw.edu

1. Introduction

Join size estimation is an important step in query optimization and has been studied extensively in the literature. While several database systems assume independence when estimating join size as an essential part of the query optimization step, many real-world data do not meet this assumption. To this end, Vengerov et al. (2015) propose correlated sampling to deal with the correlation among attributes. The goal of this project is to evaluate the estimated join size produced by correlated sampling on the DBLP dataset, and compare this approach with the estimates returned by postgreSQL. We also study how the performance of correlated sampling changes as the number of joins increases.

Leis et al. (2015) have shown that postgreSQL and several commercial DBMS can have join size underestimation up to 10^6 when joining over multiple tables on the IMDB dataset. PostgreSQL uses a simple formula to calculate the estimate, assuming uniformity, independence, and principle of inclusion, and commercial DBMS presumably calculate estimates assuming independence. None of these estimators uses a sampling approach.

On the other hand, several sampling based approaches have been proposed. Independent Bernoulli sampling is the most simple one, but the variance of the estimator can get very large when infrequent values of the join attribute dominate two tables. End-based sampling addresses the shortcoming of Bernoulli sampling, but it is not suitable in a streaming fashion. Building on Bernoulli sampling and end-based sampling, correlated sampling constructs a small space synopsis for each table by having a uniform hash function on the join attribute values, and then estimates the join size based on the join size of small tables and the sampling probability. To make further improvements, Chen and Yi (2017) propose two-level sampling, which combines the advantages of all three of these sampling methods. Two-level sampling samples the join values in a correlated way, and then uses independent Bernoulli to sample tuples sharing the same join values.

2. Correlated Sampling

In this section, we present the correlated sampling method in more detail.

First we introduce the notion of equivalent classes. For two tables T_i and T_j and join attributes u_{ik}, u_{jl}, we denote $u_{ik} \sim u_{jl}$ whenever the join condition $u_{ik} = u_{jl}$ is present. For any attribute u_{ik}, we denote by $\psi(u_{ik})$ its equivalence class under the relation $\sim$, which includes all other attributes that have to be equal to u_{ik} under the considered join query. We assume that the complex query has K equivalence classes: $\Psi_1, ..., \Psi_K$.

For each equivalence class Ψ_k, we define a uniform hash function h_k from the attribute domain to $[0,1]$.We require, for $h_k \neq h_j$, the hash values of these two attributes be completely independent. For this project, we choose the hash function $h(v) = ((av+b)\ mod\ p)/p$, where $a, b \in [1,p)$ are randomly chosen integers and p is a large prime number that is greater than number of distinct values in the join attribute. We also define the function $\phi : u_{ij} \to k$ to map an attribute to its equivalent class index k.

let U_i be the set of all join attributes for Table T_i and let $|U_i|$ denote the size of this set. For any attribute $u_{ij} \in U_i$ and the value v_{ij} it takes, consider the following event:

$$h_{\phi(u_{ij})}(v_{ij}) < \epsilon_i \tag{1}$$

We refer to this the inclusion condition for u_{ij}.

We define the following two rules for deciding whether a given row r is included in the sample S_i:

1. The **AND** rule: a given row r will be included in the sample S_i if the inclusion condition is satisfied for all join attributes u_{ij} in that row.
2. The **OR** rule: a given row r will be included in the sample S_i if the inclusion condition is satisfied for any join attribute u_{ij} in that row.

Let p_i denote the probability of a given row r of table T_i included in the sample S_i. Then for The **AND** rule, ϵ_i is simply $(p_i)^{\frac{1}{|U_i|}}$. To see this mathematically, recall that the chosen hash function generate independent uniform random variables regardless of correlations among the inputs, so we have

$$\begin{aligned}
&P\left(h_{\phi(u_{i1})}(v_{i1}) < \epsilon_i \wedge ... \wedge h_{\phi(u_{i|U_i|})}(v_{i|U_i|}) < \epsilon_i\right) \\
&= P\left(h_{\phi(u_{i1})}(v_{i1}) < \epsilon_i\right) P\left(h_{\phi(u_{i|U_i|})}(v_{i|U_i|}) < \epsilon_i\right) \\
&= (\epsilon_i)^{|U_i|} = \left((p_i)^{\frac{1}{|U_i|}}\right)^{|U_i|} = p_i
\end{aligned}$$

For the **OR** rule, take $\epsilon_i = 1-(1-p_i)^{\frac{1}{|U_i|}}$. Then we have:

$$\begin{aligned}
&P\left(h_{\phi(u_{i1})}(v_{i1}) < \epsilon_i \vee ... \vee h_{\phi(u_{i|U_i|})}(v_{i|U_i|}) < \epsilon_i\right) \\
&= 1 - P\left(h_{\phi(u_{i1})}(v_{i1}) \geq \epsilon_i \wedge ... \wedge h_{\phi(u_{i|U_i|})}(v_{i|U_i|}) \geq \epsilon_i\right) \\
&= 1 - P\left(h_{\phi(u_{i1})}(v_{i1}) \geq \epsilon_i\right) P\left(h_{\phi(u_{i|U_i|})}(v_{i|U_i|}) \geq \epsilon_i\right) \\
&= 1 - (1-\epsilon_i)^{|U_i|} = 1 - \left(1 - 1 - (1-p_i)^{\frac{1}{|U_i|}}\right)^{|U_i|} = p_i
\end{aligned}$$

Assume that we have a complex join query that includes equijoin conditions for tables $T_1, ...T_N$. Vengerov et al. (2015) have shown that, using the **AND** rule, the probability of a particular row r_{out} from the output of that join being included in the join over samples $S_1, ..., S_N$ is

$$P_{inc} = \prod_{k=1}^{K} \min_{i \in \Psi_k} \left((p_i)^{\frac{1}{|U_i|}}\right) \tag{2}$$

Using the **OR** rule, the probability of a particular row r_{out} from the output of that join being included in the join over samples $S_1, ..., S_N$ is

$$P_{inc} = P(\wedge_{i=1}^{N} \vee_{j=1}^{|U_i|} h_{\phi(u_{ij})}(v_{ij}) < 1-(1-p_i)^{\frac{1}{|U_i|}}) \tag{3}$$

To calculate this probability, consider the following event,

$$\{\mathbb{1}_{h_{\phi(u_{11})}(v_{11})<\epsilon_i}, ..., \mathbb{1}_{h_{\phi(u_{N|U_N|})}(v_{N|U_N|})<\epsilon_N}\}$$

For a given query, the strategy is to find all events that will return the output row in the joined samples, calculate each case's probability and add them up.

This probability is the same for all rows that appear in the output of the join, implying a very simple way of estimating the cardinality of the join: compute the join size over samples $S_1, ...S_N$ and divide it by the inclusion probability P_{inc}. Let J' denote the join size over samples $S_1, ...S_N$, then our cardinality estimator is $\hat{J} = J'/P_{inc}$.

Vengerov et al. (2015) have shown that the estimator $\hat{J}$ is unbiased. The variance $var(\hat{J}) = \left(\frac{1}{P_{inc}} - 1\right) \sum_{\vec{v}} F^2(\vec{v})$, where $F(\vec{v})$ is the number of rows in the output of the join that have a combination of join attributes specified by $\vec{v}$. Therefore, the variance of the estimator decreases as p_i increases.

The choice of sampling probability is a practical concern. Assume that p_1 is given (p_1 can be chosen according to the size of table T_1 and the budget), and that T_1 and T_2 both appear in the same equivalence class Ψ. Then for the **AND** rule, the most efficient sampling probability occurs when $p_2 = p_1^{\frac{|U_2|}{|U_1|}}$. So we can start with the sampling probability for a single table in the join graph, and then get the sampling probability of other tables one by one, since every table is connected to one another via some join condition. The same logic can be applied to the **OR** rule, and we derive the formula case by case for queries considered in Section 3.2.

3. Experimental Results

We experiment with how correlated sampling performs on cardinality estimation when multiple tables in the DBLP dataset are joined. We start with joining two tables with one join attribute, and then extend our experiments with more tables and join attributes.

3.1. Data Description

The DBLP dataset schema consists of three tables: `publication`, `author`, and `authored`. Table `publication` has primary key `pubid`; table `author` has primary key `id`; table `authored` has primary key pair (`id`, `pubid`), where foreign key `id` references primary key `id` in `author` and foreign key `pubid` references primary key `pubid` in `publication`. The `authored` table contains 11537965 entries, with 3962010 distinct values of `pubid` and 2019845 distinct values of `id`.

3.2. Queries

Suppose we want to get a table that gives the information on collaboration. Then we need to join table `author` with itself on the condition that the attribute `pubid` in the two tables should be the same. To get a table `collaboration` with columns `id1`, `id2` denoting a pair of author who have written a specific paper together, we need to run the following query:

```
select A1.id, A2.id
from authored A1, authored A2
where A1.pubid=A2.pubid and A1.id!=A2.id;
```

Due to the scope of this project, we do not consider the selectivity criteria when estimating cardinality. So the query we consider is simply joining over two tables:

```
select A1.id, A2.id
from authored A1, authored A2
where A1.pubid=A2.pubid;
```

Going from two-table joins to three table joins, suppose we want to find all papers with at least three authors. We consider the following query:

```
select A1.pubid
from authored A1, authored A2, authored A3
where A1.pubid=A2.pubid and
A2.pubid=A3.pubid;
```

Of course, to really get all papers with at least three authors, we would need `A1.id` $\neq$ `A2.id`, `A1.id` $\neq$ `A3.id`, and `A2.id` $\neq$ `A3.id`. But we omit the selectivity for now.

Even though we have experimented with three-table joins, we notice that there is only one equivalence class

(of `pubid`) with the experiments above. This makes the **AND** and **OR** trivial because $|U_i| = 1$ for all i. For the above two queries, we experiment with $p_i = \epsilon_i = [0.001 \;\; 0.005 \;\; 0.01]$.

We now consider a query that joins over tables with at least one $|U_i| > 1$. Suppose we want to get authors who have written at least two papers together. Then we would run the following query:

```
select A1.id, A2.id
from authored A1, authored A2,
authored A3 ,authored A4
where A1.pubid=A2.pubid and
A3.pubid=A4.pubid and A1.pubid≠A3.pubid and
A1.id=A3.id and A2.id=A4.id and A1.id ≠
A2.id and A3.id ≠ A4.id;
```

This is a bit more complicated than the scope of this project, so we instead consider:

```
select A1.id, A2.id
from authored A1, authored A2,
authored A3 ,authored A4
where A1.pubid=A2.pubid and
A3.pubid=A4.pubid and A1.id=A3.id and
A2.id=A4.id;
```

This query allows us to study how correlated sampling using the **AND** rule and the **OR** rule performs on cardinality estimation.

This is a loop query where $|U_1| = |U_2| = |U_3| = |U_4| = 2$, and therefore we choose $p_1 = p_2 = p_3 = p_4$. For both the **AND** rule and the **OR** rule, we experiment with $p_i = [0.001 \;\; 0.005 \;\; 0.01 \;\; 0.03 \;\; 0.05]$. We can find the corresponding ϵ_i using the formulas in Section 2.

For each query, we sample tables 1000 times and get 1000 estimated cardinality. We show in the next section the distribution of estimated cardinality, and whether the average estimated cardinality converge to the true value.

3.3. Results

3.3.1. Joining over one attribute

Figures 1 and 2 show the distribution of estimated cardinality over 1000 simulations for two-table and three-table joins without outliers, at different levels of sampling probability p_i. The algorithm gives better estimates for two-table joins than those for three-table joins. We observe that as the sampling probability increases, the median estimated cardinality gets closer to true values and variances decrease.

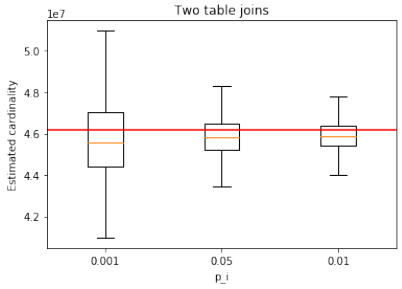

Figure 1. Distribution of estimated cardinality when joining over two tables (outliers not shown). The horizontal red line represents the true cardinality, and the orange line represents the median estimate.

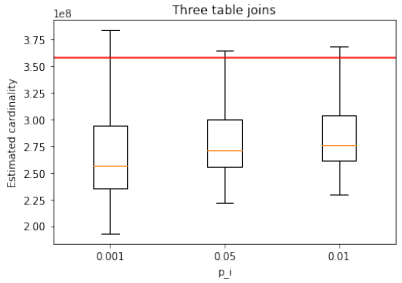

Figure 2. Distribution of estimated cardinality when joining over three tables (outliers not shown). The horizontal red line represents the true cardinality, and the orange line represents the median estimate.

The median alone does not tell that the estimator is unbiased. To study empirically whether the estimator is unbiased, we take average of estimates, and calculate its q-error, define in Leis et al. (2015) as the factor by which an estimate differs from the true cardinality. Table 1 shows that the estimators for two-table join and three-table join are unbiased, since the q-error is close to one when we have large enough sampling probability.

	two-table	three-table
p_i=0.001	1.00	1.12
p_i=0.005	1.00	1.02
p_i=0.01	1.00	1.02

Table 1. Q-error of average of estimates over 1000 simulations when joining over single attribute

Because the plots don't show the outliers, we further investigate the estimator using median, 90 percentile, 95 percentile and maximum q-error. Table 2 shows that median, 90%, 95%, and maximum q-error of estimated cardinality at different levels of sampling probability. We observe that small sampling probability and large number of joins are associated with large q-errors.

	median	90th	95th	max
p_i=0.001, two-table	1.03	1.08	1.09	3.07
p_i=0.005, two-table	1.02	1.04	1.05	1.41
p_i=0.01, two-table	1.01	1.03	1.04	1.2
p_i=0.001, three-table	1.41	1.63	1.7	69.97
p_i=0.005, three-table	1.33	1.49	1.54	14.69
p_i=0.01, three-table	1.31	1.45	1.5	7.71

Table 2. Q-error of estimator when joining over single attribute

For reference, PostgreSQL gives 104206615 for two-table joins, and 941155854 for three-table joins. This is a q-error of 2.26 for two-table joins and a q-error of 2.63 for three-table joins. Both of these numbers are greater than 95% q-error for correlated sampling even at $p = 0.001$!

3.3.2. Joining over multiple attributes

In this section, we present experimental results on joining four tables with four equivalence classes in total.

Table 3 shows that the correlated sampling estimator using the **AND** rule and **OR** rule are unbiased. For the **AND** rule, the q-errors of average of estimates are perfect: they are exactly one at each p_i. For the **OR** rule, the q-errors of average of estimates are very good: they are approaching one as p_i increases.

Tables 4 and 5 show the q-errors of the estimator using the **AND** rule and the **OR** rule, respectively. Notice that at each p_i, the **OR** rule gives less precise estimates than the **AND**. This could be explained by the fact that, for a given p_i, ϵ_i used with the **AND** rule is larger, which leads to a larger P_{inc} and thus a smaller variance of the estimator.

Figure 3 shows the distribution of estimated cardinality when joining over multiple attributes using the **AND** rule. We observe that the distribution is symmetric and the variance of the estimator decreases as p_i increases. Figure 4 shows the distribution of estimated cardinality when joining over multiple attributes using the **OR** rule. We observe that the distribution is right-skewed and the median estimate approaches true cardinality as p_i increases. In theory, variance should decrease with increasing p_i. However, this is only the case with the **OR** rule when $p_i \geq 0.01$. my intuition is that $p_i = 0.001$ and $p_i = 0.005$ are just too small for the **OR** rule to work. This is because, even

though the **AND** rule and the **OR** rule both produce $|S_i| \approx |T_i| \cdot p_i$, the join size over $S_1, ..., S_N$ are larger with the **AND** rule.

For reference, PostgreSQL gives 692764 for this loop-join query over four tables (a q-error of 1345!) Therefore, the estimated cardinality given by PostgreSQL is a lot worse than even the max estimate returned by correlated sampling over 1000 simulations. We see that correlated sampling outperforms PostgreSQL more as the query becomes more complicated with more join attributes.

	AND rule	**OR rule**
p_i=0.001	1.00	1.12
p_i=0.005	1.00	1.04
p_i=0.01	1.00	1.01
p_i=0.03	1.00	1.01
p_i=0.05	1.00	1.00

Table 3. Q-error of average of estimates over 1000 simulations when joining over multiple attributes

AND rule	median	90th	95th	max
p=0.001	1.20	1.63	1.81	3.95
p=0.005	1.10	1.25	1.30	1.62
p=0.01	1.07	1.18	1.22	1.49
p=0.03	1.04	1.10	1.12	1.24
p=0.05	1.03	1.08	1.09	1.20

Table 4. Q-error of estimator when joining over multiple attributes using the **AND** rule

OR rule	median	90th	95th	max
p=0.001	1.86	2.33	2.62	118.06
p=0.005	1.54	1.83	2.01	42.73
p=0.01	1.39	1.71	2.06	12.83
p=0.03	1.18	1.40	1.49	6.28
p=0.05	1.11	1.27	1.35	3.34

Table 5. Q-error of estimator when joining over multiple attributes using the **OR** rule

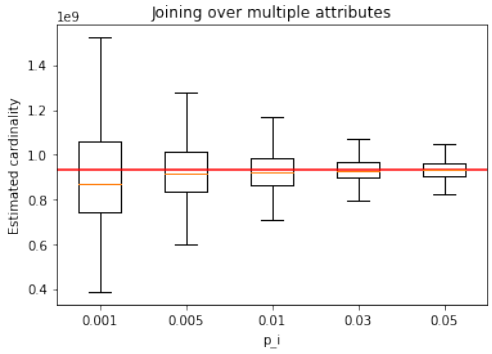

Figure 3. Distribution of estimated cardinality when joining over multiple attributes using the **AND** rule (outliers not shown). The horizontal red line represents the true cardinality, and the orange line represents the median estimate.

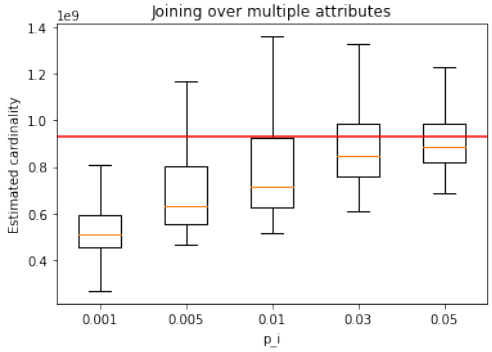

Figure 4. Distribution of estimated cardinality when joining over multiple attributes using the **OR** rule (outliers not shown). The horizontal red line represents the true cardinality, and the orange line represents the median estimate.

4. Conclusion

In this project, we investigate how correlated sampling performs on queries using DBLP dataset, and we compare its performance to PostgresSQL's estimate. For a simple two-table join over one attribute and three table joins over one attribute, PostgreSQL's estimates are less precise than 95% of the estimates by correlated sampling, even at the smallest sampling probability. For a loop-join query that involves four tables and four equivalence classes, PostgreSQL's estimate is a lot worse than the worst estimate produced by correlated sampling over 1000 simulations.

Of the queries we have considered, correlated sampling produces a skewed distribution of estimates on the three-table joins and loop-join with the **OR** rule. We have provided empirical evidence that correlated sampling estimator is unbiased on all three queries we have experimented with. In addition, we have discovered that with the **OR** rule, we cannot choose the proportion of sampled tables to be too small, since that will just not give us a good enough join size over sampled tables.

References

Yu Chen and Ke Yi. Two-level sampling for join size estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 759–774. ACM, 2017.

Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015.

David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P Chakkappen. Join size estimation subject to filter conditions. *Proceedings of the VLDB Endowment*, 8(12):1530–1541, 2015.