Tighter Upper Bounds for Join Cardinality Estimates

Walter Cai
Allen School of Computer Science and Engineering, University of Washington
Seattle, Washington
walter@cs.washington.edu

ACM Reference Format:
Walter Cai. 2018. Tighter Upper Bounds for Join Cardinality Estimates. In *SIGMOD/PODS '18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3183713.3183714

1 PROBLEM AND MOTIVATION

Despite decades of research, modern database systems still struggle with multijoin queries. Users will often experience long wait times occurring with unpredictable frequency detracting from the usability of the system. In this work we develop a new method to tighten join cardinality upper bounds. The intention for these bounds is to assist the query optimizer (QO) in avoiding expensive physical join plans. Our approach is as follows: leveraging data sketching, and randomized hashing we generate and tighten theoretical join cardinality upper bounds. We outline our base data structures and methodology, and how these bounds may be introduced to a traditional QO framework as a new statistic for physical join plan selection. We evaluate the tightness of our bounds on GooglePlus community graphs and successfully generate degree of magnitude upper bounds even in the presence of multiway cyclic joins.

2 BACKGROUND AND RELATED WORK

The majority of recent work on cardinality estimation focuses on the use and optimization of sampling methods [5, 7, 10, 12]. Sampling is attractive since it inherently handles all common predicate filters while also delivering unbiased estimates. Nevertheless, a combination of data summarization techniques (e.g. multidimensional histograms), hand-written rules, and strong assumptions on the underlying data remain as the de facto estimation method for production systems [11, 12]. For longer multijoins or cyclic joins, both approaches become increasingly imprecise and inaccurate [9]. These poor estimates lead to poor physical plan selection, which leads to poor performance. We argue providing guaranteed bounds on intermediate join cardinality makes plan selection more robust.

The first step towards these bounds comes from analyzing the connection between information theory and relational databases. Framing relational joins as hypergraphs, Atserias et al. provide upper bounds based on fractional edge covers [2]. More formally, we are given a conjunctive query on relations $\{R_j\}$ with variable sets $\{\boldsymbol{x}_j\}$ where $\boldsymbol{x}_j \subseteq \boldsymbol{x}$:

$$Q(\boldsymbol{x}) : -R_1(\boldsymbol{x}_1), \ldots, R_m(\boldsymbol{x}_n) \tag{1}$$

It is natural to view the schema as a hypergraph H with vertices corresponding to the variables $x \in \boldsymbol{x}$ and hyperedges corresponding to the vertex-sets $\{\boldsymbol{x}_j\}$ associated with each relation. A fractional edge cover $(u_1, \ldots, u_n)$ of $\boldsymbol{x}$ on H is a set of values $u_j \in \mathbb{R}_{\geq 0}$ corresponding to the hyperedges $\{\boldsymbol{x}_j\}$ where for all vertices x, the sum of the u_j values corresponding to hyperedges containing x is at least 1. That is, x is "covered":

$$\forall x \in \boldsymbol{x} : \sum_{j : x \in \boldsymbol{x}_j} u_j \geq 1 \tag{2}$$

We may bound the join cardinality as follows:

$$|Q| \leq \prod_{j=1}^{n} |R_j|^{u_j} \tag{3}$$

This class of bounds is coined the *AGM bound*. Khamis et al. extend this research to include degree parameters [8]. Their contributions generalize the information theory versus relational join analogy to conditional entropic formulations. We refer to this broader class of bounds as the *KNS bound*.

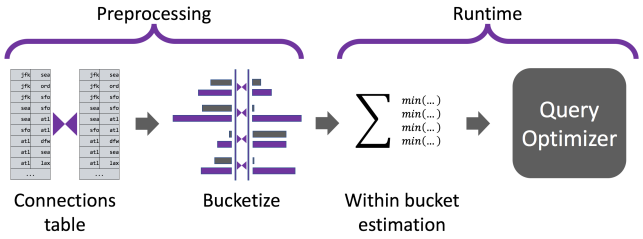

Figure 1: Simple join upper bound generation illustration.

3 APPROACH AND UNIQUENESS

In this project we generate guaranteed join cardinality upper bounds and demonstrate how these bounds are tightened. In Subsection 3.1, we define our core data structure; the *Bound Sketch* (BS). In Subsection 3.2, we describe how the BS is used to generate and tighten theoretical join cardinality upper bounds. We provide an illustration of our method on a simple two table join in Figure 1. Note that while our data structures are constructed offline, they may be called at runtime for any joins involving the preprocessed tables.

3.1 The Bound Sketch

We first describe the structure of the BS. Take relation $T(\boldsymbol{y})$, with the variables $x \in \boldsymbol{y}$ and random hash function $h : W \mapsto \{1, \ldots, m\}$. Tuples in T take values in the domain $W^{|\boldsymbol{y}|}$. Let $\boldsymbol{y} = (x_1, x_2, \ldots, x_r)$,

SIGMOD/PODS '18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06.
https://doi.org/10.1145/3183713.3183714

and let $[m]$ denote $\{1, \ldots, m\}$. For each index value $I \in [m]^r$, we define the subset T^I of relational instance T as those tuples $t \in T$ that hash to the index array I. That is:

$$T^I = \left\{ t \in T : h(t[x_1]) = I[x_1], \ldots, h(t[x_r]) = I[x_r] \right\} \tag{4}$$

The BS on T is a collection of $|\boldsymbol{y}|$+1-many $|\boldsymbol{y}|$-dimensional tensors of the form $m \times \cdots \times m$. Each cell of the initial tensor contains a *count* value c. We define the I-th entry of this tensor as $c^I = |T^I|$. Each cell of the remaining $|\boldsymbol{y}|$ tensors contains a *degree* value d. Each of these tensors corresponds to a variable in $\boldsymbol{y}$. We define the degree parameter corresponding to variable $x \in \boldsymbol{y}$ as the maximum degree for variable x from the subset T^I. More formally:

$$d^I[x] = \max_{\substack{w \in W, \\ h(w) = I[x]}} \left| \left\{ t \in T^I : t[x] = w \right\} \right| \tag{5}$$

Alternatively, we may attain the value $d^I[x]$ from the following SQL query:

```
SELECT MAX(c) FROM (SELECT COUNT(*) AS c FROM T^I GROUP BY x);
```

Finally, we describe the BS formally as:

$$\Big(\left[c^I\right]_{I \in [m]^{|\boldsymbol{y}|}}, \underbrace{\quad \ldots \quad, \left[d^I[x]\right]_{I \in [m]^{|\boldsymbol{y}|}}, \quad \ldots \quad}_{|\boldsymbol{y}|\text{-many}} \Big) \tag{6}$$

where x is a variable in $\boldsymbol{y}$.

3.2 Our Cardinality Bounds

Next, we describe how the BS is used to generate guaranteed and tight cardinality upper bounds. Consider schema instance

$$\{R_1(\boldsymbol{x}_1), \ldots, R_n(\boldsymbol{x}_n)\} \tag{7}$$

and database instance D on the relations $\{R_k\}$. Define index array $I \in [m]^{|\boldsymbol{x}|}$ as before, and let $I[\boldsymbol{x}_j]$ be the subarray of I whose values correspond to the variables present in $\boldsymbol{x}_j \subseteq \boldsymbol{x}$. Define a database instance D^I as a subset of database instance D where for each relational instance $R_i(D)$, we take the subset

$$R_i\left(D^I\right) = \left\{ t \in R_i(D) : h\left(t\left[x\right]\right) = I[x], \quad \forall x \in \boldsymbol{x}_j \right\} \tag{8}$$

That is, the set of all tuples in $R_i(D)$ which hash to the correct index values with respect to the variables in $\boldsymbol{x}_j$. Define database instance subset $D^I = \left\{R_1\left(D^I\right), \ldots, R_n\left(D^I\right)\right\}$. Given a conjunctive query $Q(D)$, observe that we may reconstruct the full conjunctive query using only these D^I:

$$Q(D) = \bigcup_{I \in [m]^{|\boldsymbol{x}|}} Q\left(D^I\right) \tag{9}$$

As an illustrative example we describe a triangle query:

$$Q(x, y, z) : -R(x, y), S(y, z), T(z, x) \tag{10}$$

At first, for each of the tables R, S, T, we consider only the maximum degree and count values for the entire table (tensors of size 1×1). We generate a collection of distinct upper bound formulas based on the count and degree statistics we described in Subsection 3.1:

$$|Q(D)| \leq \begin{cases} c_R d_S[y], & c_S d_T[z], & c_T d_R[x] \\ c_R d_T[x], & c_S d_R[y], & c_T d_S[z] \\ c_R^{\frac{1}{2}} c_S^{\frac{1}{2}} c_T^{\frac{1}{2}} \end{cases} \tag{11}$$

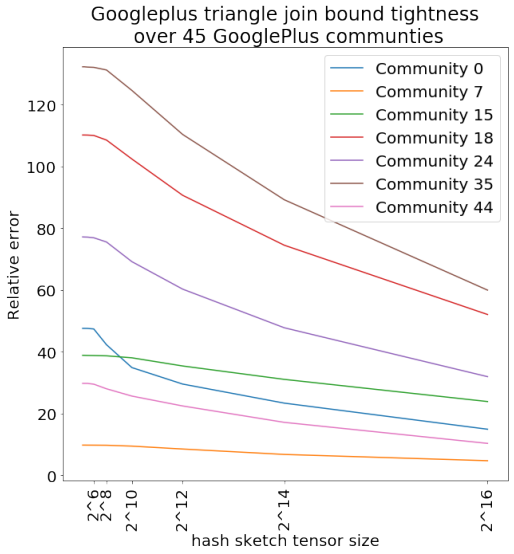

Figure 2: Tightening of bounds for GooglePlus triangles.

These formulas represent a subset of the KNS bound and each corresponds to a different entropic formula. For a full discussion of these formulae, we refer the reader to [8]. As we increase the hash size m, we generate values from the KNS bound for each subinstance D^I. The sum over all index combinations provides an upper bound on D:

$$|Q(D)| \leq \sum_{I \in [m]^{|\boldsymbol{x}|}} \min \begin{cases} c_{R^I} d_{S^I}[y], & c_{S^I} d_{T^I}[z], & c_{T^I} d_{R^I}[x] \\ c_{R^I} d_{T^I}[x], & c_{S^I} d_{R^I}[y], & c_{T^I} d_{S^I}[z] \\ c_{R^I}^{\frac{1}{2}} c_{S^I}^{\frac{1}{2}} c_{T^I}^{\frac{1}{2}} \end{cases} \tag{12}$$

4 EVALUATION, & FUTURE DIRECTIONS

We evaluate our bounds on a collection of 45 GooglePlus community user-follower edge-sets [6]. The edge counts within the communities range between 228,521 and 1,614,977, and the cardinality of the self-join triangles derived from these respective communities range between 1,791,588 and 130,322,694. In Figure 2 we include the progressive tightening of our bounds on triangle queries as we increase BS size. Each line represents the proportion of our upper bound over the true join cardinality for a single GooglePlus community. We observe a tradeoff space between BS size and the tightness of our bounds.

We argue that tight cardinality upper bounds may be easily used in existing systems. The simplest approach is to inject the upper bounds directly into the query optimizer in place of the cardinality estimates. The bounds will reflect correlation across tables and expensive join orderings will be avoided. Alternatively, the bounds are paired with a sample-based unbiased estimate. These pairs are then used to generate a distributional view of join cardinality similar to past work in robust query optimization [3, 4] and bounded query execution [1].

ACKNOWLEDGMENTS

This project is supported by NSF grant AITF 1535565.

REFERENCES

[1] Michael Armbrust, Kristal Curtis, Tim Kraska, Armando Fox, Michael J. Franklin, and David A. Patterson. 2011. PIQL: Success-Tolerant Query Processing in the Cloud. *CoRR* abs/1111.7166 (2011). arXiv:1111.7166 http://arxiv.org/abs/1111.7166

[2] Albert Atserias, Martin Grohe, and DÃąniel Marx. 2013. Size Bounds and Query Plans for Relational Joins. *SIAM J. Comput.* 42, 4 (2013), 1737–1767. https://doi.org/10.1137/110859440 arXiv:https://doi.org/10.1137/110859440

[3] Brian Babcock and Surajit Chaudhuri. 2005. Towards a Robust Query Optimizer: A Principled and Practical Approach. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*. ACM, New York, NY, USA, 119–130. https://doi.org/10.1145/1066157.1066172

[4] Shivnath Babu, Pedro Bizarro, and David DeWitt. 2005. Proactive Re-optimization. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*. ACM, New York, NY, USA, 107–118. https://doi.org/10.1145/1066157.1066171

[5] Yu Chen and Ke Yi. 2017. Two-Level Sampling for Join Size Estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. ACM, New York, NY, USA, 759–774. https://doi.org/10.1145/3035918.3035921

[6] Google. 2017. *GooglePlus*. https://plus.google.com/

[7] Srikanth Kandula, Anil Shanbhag, Aleksandar Vitorovic, Matthaios Olma, Robert Grandl, Surajit Chaudhuri, and Bolin Ding. 2016. Quickr: Lazily Approximating Complex AdHoc Queries in BigData Clusters. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM, New York, NY, USA, 631–646. https://doi.org/10.1145/2882903.2882940

[8] Mahmoud Abo Khamis, Hung Q. Ngo, and Dan Suciu. 2016. What do Shannon-type inequalities, submodular width, and disjunctive datalog have to do with one another? *CoRR* abs/1612.02503 (2016). arXiv:1612.02503 http://arxiv.org/abs/1612.02503

[9] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (Nov. 2015), 204–215. https://doi.org/10.14778/2850583.2850594

[10] Viktor Leis, Bernharde Radke, Andrey Gubichev, Alfons Kemper, and Thomas Neumann. 2017. Cardinality Estimation Done Right: Index-Based Join Sampling. In *CIDR*.

[11] Postgres Development Core Team. 2017. *PostgreSQL*. https://www.postgresql.org/

[12] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil Chakkappen. 2015. Join Size Estimation Subject to Filter Conditions. *PVLDB* 8 (2015), 1530–1541.