Synopsis-based Sampling for Cardinality Estimation of SPARQL queries

Lei Gai[1], Tengjiao Wang[1], and Wei Chen[1]

Peking University, P.R.China {lei.gai, tjwang, pekingchenwei}@pku.edu.cn

Abstract. The abstract

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

SPARQL is a formal way to structurally express users' analytical/semantical tasks. For many of today's tasks, user often need to pose ad hoc complex queries involving huge tables.

The research community has long realized the need for accurate cardinality estimation in query optimization. Unfortunately, despite many eminent works on this topic, it is still not a well solved problem considering the trade-off between several factors.

We can see two line of work. synopsis

Due to the multiplicative nature of joins, errors in cardinality estimation typically propagate exponentially through a larger query plan (cite SIGMOD91, ioannidis). This means even small improvement can dramatically improve the quality of estimation available to the query optimizer, thus boot the overall query performance (cite VLDB2015, how good are query optimzier, really?)

Joins are expensive, especially on large data with complicated correlations. The sample operator cannot be push through a join sample, i.e., $sample(R \bowtie)S) \neq sample(R) \bowtie (S)$. This means sample is not reusable to the next sampling. (cite SIgMOD2018 random sampling over joins revisited, CIDR2017, cardinality estimation done right-index-based join sampling).

The optimizer's cost model relies heavily upon the underlying cardinality model.

2 Preliminaries

2.1 Sequential Sampling

2.2 multi-way join sampling

simple random sample of size k, each element in the underlying population is picked with equal probability (uniform), and the procedure is repeated k times independently (independent).

the fundamental challenge for the problem i that the sampling operation cannot be pushed down through a join operator.

To obtain uniform and independent samples, Two methods were proposed (cite Chaudhuri). One is to reject samples from $sample(R) \bowtie sample(S)$ with appropriate probability. The other takes samples from R non-uniformly, guided by the statistical information on S. (However, they only considered the problem over two-relation joins).

(cite SIGMOD 2018) gives a generic framework to handle multi-way joins, and with selection predicates. It leverage the join size upper bound (cite AGM method), depending on what prior information is available about the underlying data, and offers different trade-offs between sample production latency and throughput.

common join attribute, join predicate.

If the sample size is n and the dataset size is N, SRS generates a sample by taking every k-th data item ($k = \lceil \frac{N}{n} \rceil$).

2.3 cardinality estimation

(cite VLDB2015, how good are query optimzier, really?)

- **estimate for triple pattern (base table)**
- **estimate for Joins**

2.4 self-join, sampling

(AGMS) Alon, N., Gibbons, P.B., Matias, Y., Szegedy, M. Tracking Join and Self-Join Sizes in Limited Storage. In Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '99), ACM Press, New York, NY, 1999, 10-20.

3 Data Model

Idea: use an oracle to tell when a key is heavy [Kumar Xu 06] Adjust sampling probability accordingly

Can use a "sketch" data structure to play the role of oracle Like a hash table with collisions, tracks approximate frequencies E.g. (Counting) Bloom Filters, Count-Min Sketch

Track probability with which key is sampled, use HT estimators Set probability of sampling key with (estimated) weight w as $1/(1 + \epsilon w)$ for parameter ϵ : decreases as w increases Decreasing e improves accuracy, increases sample size

4 Sampling

For each tuple in R, choose it for the sample with probability f, independent of other tuples. In real world DB, choose each tuple in R is prohibitively costly. we

can easily convert it to Sampling with/-out Replacement (SWoR). Sampling a slightly larger fraction $\hat{f}$ to ensure that we get at least an f-fraction, as can be shown by the Chernoff bound, and then rejecting an appropriate number of the samples to ensure that we get exactly an f-fraction.

Given a RDF dataset $\mathcal{D}$ of d triples, and a Conjunctive SPARQL query with k triple patterns $TP = \{P_i\}$, $i = 1, \ldots, k$. We use $\sigma_{P_i}(\mathcal{D})$ to represent the set of triples that match triple pattern P_i. For two triple pattern P_1 and P_2 with a join attribute A, let $n_1(v)$ and $n_2(v)$ be distinct number of triples in $\sigma_{P_1}(\mathcal{D})$ and $\sigma_{P_2}(\mathcal{D})$ that contains value v in A respectively. Clearly, $\sum_{v \in A} n_1(v) = |\sigma_{P_1}(\mathcal{D})|$, and $\sum_{v \in A} n_2(v) = |\sigma_{P_2}(\mathcal{D})|$. The cardinality of $P_1 \bowtie P_2$ can be estimated as

$$n = |P_1 \bowtie P_2| = \sum_{v \in A} n_1(v) \cdot n_2(v) \tag{1}$$

For $n_i(v)$, if there are synopsis available to capture the correlation among triple elements, such multi-dimensional histogram or index like RDF-3X, $n_i(v)$ can be directly estimated or even get exactly. otherwise more common cases we should rely on applying *independence assumption* for S,P and O.

To computing $\textsc{Sampling}(TR_1 \bowtie TR_2, f)$

References