Technical Perspective on "Reservoir Sampling over Joins"

Surajit Chaudhuri
Microsoft Research
surajitc@microsoft.com

In this technical perspective, I discuss some background and context for the paper "Reservoir Sampling over Joins". Sampling is a well-studied data analysis technique. It enables analysts to accelerate their ability to explore characteristics of large data sets. When we want to create *only a sample* of the output of a complex query over large data sets, a natural question is if it is possible to avoid paying the full cost of executing the query. In some cases, these are standing queries over changing data and for these the system could potentially maintain the samples as data evolve instead of having to obtain samples from scratch.

These challenges have been studied in database research since the late nineties. Two papers in SIGMOD 1999, referenced in this paper by Dai, Hu, and Yi, noted that independent samples of relations cannot be joined to obtain a uniform random sample of the joins among the relations; moreover, for join queries with projection, the projection over a uniform random sample may result in a biased sample. These two papers presented techniques to compute a sample of the output of the join query without computing the full joins, taking advantage of the structure of the query graph, available statistical information, and access paths. The paper by Zhao et al. in SIGMOD 2018 generalized the results of these two earlier papers. All these papers focused on the cases where the join queries are specified on the fly. The challenge of maintaining a sample of the output of a standing join query, as data changes, received relatively less attention. This problem is of interest when the output of a join query is large, e.g., arising from many to many joins. The SIGMOD 2020 paper by Zhou et al. was the first to present a solution to this challenge of maintaining the sample for a join query when data is either added or deleted. They handle many to many multi-join queries with equi-join as well as band join predicates.

This paper builds upon the paper by Zhao et al. but limits its focus to equi-join queries over append-only data. Their key result is that for this important special case, they can maintain a uniform random sample of an acyclic join query in $O(N \log N)$ time when N, the number of tuples in the database, is sufficiently large.

While technically dense, this paper contains several interesting ideas. First, they showed that it is possible to do reservoir sampling on the output of a selection condition on the stream by embedding the selection inside the reservoir sampling algorithm. Thus, they avoid paying the cost of filtering for every tuple up-front. This algorithm may be valuable in other sampling problems. Second, they designed an index, *linear* in the size of the data, that organizes the data into carefully designed buckets, taking into account the equi-join predicates in the query. Their organization of tuples into buckets ensures that while maintaining the index, the number of times any tuple is moved among buckets is *logarithmic in the data size*, key to their result on index maintenance. However, while ensuring these desirable properties, their index *conceptually* contains some *dummy* tuples. Fortunately, their bookkeeping keeps track of these dummy tuples and using the variant of the reservoir sampling with a selection condition mentioned above, their algorithm produces a uniform random sample of the result of a join query.

Despite the progress in this paper and the research community, the current state-of-the-art techniques in computing or maintaining random samples of results of queries do not adequately handle complex SQL – for example, techniques such as those in this paper do not handle nested queries – and no systems have shown their effectiveness on scale-out systems that compute analytics over large datasets. Will this goal ever be a reality? Or do we need to re-imagine other ways to obtain quick and dirty answers to analytic queries?