Javier Palomares

Professor Miranker

EE382 Data Engineering

11/9/2018

Physical Plans in SQLite and Postgres

For this assignment, I chose to investigate the query plan SQLite generated assignment. I observed that for several of the join queries, SQLite chooses to scan tables using an automatic index. For the remaining join queries, the behavior was to use full table scans over all tuples of the relations. I investigated the automatic index in SQLite’s documentation found that SQLite might create an automatic index that lasts only for the duration of a single SQL statement. Since the cost of constructing the automatic index is O(T(R)\*logT(R)) and the cost of doing a full table is O(T(R)), an automatic index will be created if SQLite expects that the lookup will be run more than logN times during the course of the SQL statement. Moreover, the current implementation on SQLite only includes nested joins (no merge-join or hash-join). This behavior explains why I observed no difference when joining relations over join keys where secondary indexes exists compared to joining with secondary indexes. Therefore, the best set of indexes in SQLite for the workload is to have no indexes. Because of this, I do not recommend choosing SQLite when join queries are frequent.

.