Lecture #16: Cost Models

15-721 Advanced Database Systems (Spring 2024)
https://15721.courses.cs.cmu.edu/spring2024/
Carnegie Mellon University
Prof. Andy Pavlo

1 Cost Models

Cost-Based Query Planning generates an estimate of the cost of executing a particular query plan for the current state of the database. The estimates are only meaningful internally and are independent of the search strategies. Some examples are:

- Size of intermediate results
- Choices of algorithms, access methods
- Resource utilization (CPU, I/O, network)
- Data properties (skew, order, placement)
- Interactions with other queries/tasks in DBMS

The number of tuples processed per operator depends on three factors:

- The access methods available per table
- The distribution of values in the database's attributes
- The predicates used in the query

There are three choices for Cost Model Components:

- **Physical Costs**: Predict CPU cycles, I/O, cache misses, RAM consumption, pre-fetching, etc. Depends heavily on hardware.
- **Logical Costs**: Estimate result sizes per operator. Independent of the operator algorithm. Need estimations for operator result sizes.
- **Algorithmic Costs**: Complexity of the operator algorithm implementation.

2 Selectivity

The selectivity of an operator is the percentage of data accessed for a predicate. The DBMS estimates selectivities using domain constraints, precomputed statistics (Zone Maps), histograms/approximations, and sampling.

Histograms

The most common method. Histograms maintain occurrence count per value in a column.

Sketches

Maintaining exact statistics about the database is expensive and slow. Sketches are data structures used to store approximate statistics with bounded error.

Sampling

Execute a predicate on a random sample of the target dataset. The number of tuples to examine depends on the size of the table. There are two approaches:

- **Maintain Read-Only Copy**: Periodically refresh to maintain accuracy.
- **Sample Real Tables**: Use READ UNCOMMITTED isolation. May read multiple versions of the same logical tuple.

ML Model

Train an ML model (e.g., transformer) that estimates the selectivity of predicates on data. It can potentially discover correlations between tables that are hard for human to observe. This method is still in the early phase.

3 Cardinality

The cardinality is the number of tuples that will be generated per operator and is computed from its selectivity multiplied by the number of tuples in its input. Many DBMSs make simplifying assumptions on cardinality estimation:

- **Uniform Data**: The distribution of values (except for the heavy hitters) is the same.
- **Independent Predicates**: The predicates on attributes are independent.
- **Inclusion Principle**: The domain of join keys overlap such that each key in the inner relation will also exist in the outer table.

Real-world data is often complicated, where these assumptions do not hold.

Correlated Attributes

Consider a database of cars, where one column is models and another column is brand. For a query (brand=”Tesla” AND model=”Model X”), with the independence and uniformity assumptions, the selectivity will be the product of ”Tesla” and ”Model X”’s selectivities. However, in reality, only ”Tesla” makes ”Model X”, hence the real selectivity should be the min value rather than the product of their selectivities.

Column Group Statistics

As a measure to incorporate correlations, the DBMS can track statistics for groups of attributes together rather than just treating them all as independent variables. Some systems (e.g., MSSQL[5]) can automatically build such column groups, while some other systems require manually specifying the groups.

Estimator Quality

The Germans evaluated the correctness of cardinality estimates generated by DBMS optimizers as the number of joins increases. The queries were from the JOB workload based on IMDB data. All the systems produced bad results on cardinality estimation, exhibiting serious underestimation.

What we can learn from the Germans:

- Query optimizer is more important than a fast engine. Cost-based join ordering is necessary.
- Cardinality estimates are routinely wrong. Try to use operators that do not rely on estimates.
- Hash joins + sequential scans are a robust execution model. The more indexes that are available, the more brittle the plans become (but also faster on average).
- Working on accurate models is a waste of time. It’s better to improve cardinality estimation instead.

4 Cost Model Implementations

Postgres Cost Model

Uses a combination of CPU and I/O costs that are weighted by "magic" constant factors. Default settings are obviously for a disk-resident database without a lot of memory: Processing a tuple in memory is 400x faster than reading a tuple from disk; Sequential I/O is 4x faster than random I/O.[4]

IBM DB2 Cost Model

Stores database characteristics in system catalogs for cost estimation[3], including:

- Hardware environment (microbenchmarks)
- Storage device characteristics (microbenchmarks)
- Communications bandwidth (distributed only)
- Memory resources (buffer pools, sort heaps)
- Concurrency Environment (Average number of users, Isolation level/blocking, Number of available locks)

Smallbase Cost Model

A two-phase model that automatically generates hardware costs from a logical model [2].

- **Phase #1: Identify Execution Primitives**: List of operations that the DBMS performs when executing a query. Example: evaluating predicate, index probe, sorting.
- **Phase #2: Microbenchmark**: On start-up, profile operations to compute CPU/memory costs. These measurements are used in formulas that compute operator cost based on table size.

DuckDB Cost Model

We cannot always assume there are statistics because the DBMS may be seeing a data file for the first time. When there are no statistics, the DBMS uses the number of distinct values to determine worst-case cardinality estimation for joins. An interesting point worth mentioning is when HyperLogLog is not available, a "magic" constant of 20% is used for selectivity.[1]

References

[1] T. Ebergen. Join order optimization with (almost) no statistics. Master's thesis, Vrije Universiteit, 2022. URL https://blobs.duckdb.org/papers/tom-ebergen-msc-thesis-join-order-optimization-with-almost-no-statistics.pdf.

[2] S. Listgarten and M.-A. Neimat. Modelling costs for a mm-dbms. In *RTDB*, 1996.

[3] G. M. Lohman, 2010. URL http://infolab.stanford.edu/~hyunjung/cs346/db2-talk.pdf.

[4] Postgres magic constant for physical costs estimation. URL https://www.postgresql.org/docs/current/runtime-config-query.html.

[5] Query Predicate contains multiple correlated columns. URL https://learn.microsoft.com/en-us/sql/relational-databases/statistics/statistics?view=sql-server-ver16#query-predicate-contains-multiple-correlated-columns.