ADVANCED DATABASE SYSTEMS

Query Optimizer Cost Models

16

Andy Pavlo
CMU 15-721
Spring 2024
Carnegie Mellon University

LAST CLASS

We were supposed to talk about adaptive query optimization…

COST ESTIMATION

Generate an estimate of the cost of executing a plan for the current state of the database.

→ 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

Estimates are (typically) only meaningful internally.

TODAY'S AGENDA

Cost Model Components
Data Structures
JOB Evaluation
Implementations

OBSERVATION

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

Simple queries are easy to estimate.
More complex queries are not.

COST MODEL COMPONENTS

Choice #1: Physical Costs

→ Predict CPU cycles, disk+network I/O, cache misses, RAM consumption, pre-fetching, etc…

→ Depends heavily on hardware performance.

Choice #2: Logical Costs

→ Estimate result sizes per operator (i.e., **selectivity**).

→ Independent of the operator algorithm.

→ Need estimations for operator result sizes.

Choice #3: Algorithmic Costs

→ Complexity of the operator algorithm implementation.

SELECTIVITY ESTIMATION

The **selectivity** of an operator is the percentage of data accessed for a predicate.

→ Modeled as probability of whether a predicate on any given tuple will be satisfied.

The DBMS estimates selectivities using:

→ Domain Constraints
→ Precomputed Statistics (Zone Maps)
→ Histograms / Approximations
→ Sampling

SELECTIVITY ESTIMATION

Choice #1: Histograms *(Most Common)*

→ Maintain an occurrence count per value (or range of values) in a column.

Choice #2: Sketches

→ Probabilistic data structure that gives an approximate count for a given value.

Choice #3: Sampling

→ DBMS maintains a small subset of each table that it then uses to evaluate expressions to compute selectivity.

Choice #4: ML Model *(Experimental)*

→ Train an ML model that learns the selectivity of predicates and correlations between multiple tables.

SELECTIVITY ESTIMATION: SKETCHES

Maintaining exact statistics about the database is expensive and slow.

Use approximate data structures called **sketches** to generate error-bounded estimates.

→ Count Distinct (HyperLogLog)
→ Quantiles (t-digest)
→ Frequent Items (Count-min Sketch)

Open-source implementations are available (Apache DataSketches)

SELECTIVITY ESTIMATION: SAMPLING

Execute a predicate on a random sample of the target data set.

The # of tuples to examine depends on the size of the table.

Approach #1: Maintain Read-Only Copy

→ Periodically refresh to maintain accuracy.

Approach #2: Sample Real Tables

→ Use **READ UNCOMMITTED** isolation.

→ May read multiple versions of same logical tuple.

SELECTIVITY ESTIMATION: SAMPLING

Modern DBMSs also collect samples from tables to estimate selectivities.

Update samples when the underlying tables changes significantly.

```sql
SELECT AVG(age)
  FROM people
 WHERE age > 50
```

id	name	age	status
1001	Obama	62	Rested
1002	Biden	82	Old
1003	Tupac	25	Dead
1004	Bieber	30	Crunk
1005	Andy	42	Healthy
1006	TigerKing	61	Jailed

⋮

1 billion tuples

Table Sample

1001	Obama	62	Rested
1003	Tupac	25	Dead
1005	Andy	42	Healthy

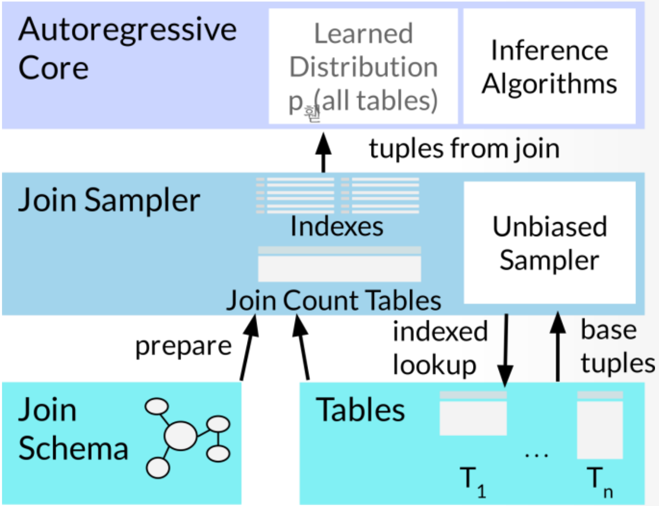
sel(age>50) = 1/3

SELECTIVITY ESTIMATION: ML MODEL

Train a ML model (e.g., transformer) that estimates the selectivity of predicates on data.

→ Potentially identify non-trivial relationships between tables more easily than human-devised methods.

This work is still early and no DBMS incorporates them (AFAIK).



NEUROCARD: ONE CARDINALITY ESTIMATOR FOR ALL TABLES
VLDB 2020

RESULT CARDINALITY

The number of tuples that will be generated per operator is computed from its selectivity multiplied by the number of tuples in its input.

Many DBMSs make simplifying assumptions about data to support complex expressions…

RESULT CARDINALITY: ASSUMPTIONS

Assumption #1: Uniform Data

→ The distribution of values (except for the heavy hitters) is the same.

Assumption #2: Independent Predicates

→ The predicates on attributes are independent

Assumption #3: Inclusion Principle

→ The domain of join keys overlap such that each key in the inner relation will also exist in the outer table.

CORRELATED ATTRIBUTES

Consider a database of automobiles:
- → # of Makes = 10, # of Models = 100

And the following query:
- → `(make="Honda" AND model="Accord")`

With the independence and uniformity assumptions, the selectivity is:
- → $1/10 \times 1/100 = 0.001$

But since only Honda makes Accords the real selectivity is $1/100 = 0.01$

Source: Guy Lohman

COLUMN GROUP STATISTICS

The DBMS can track statistics for groups of attributes together rather than just treating them all as independent variables.

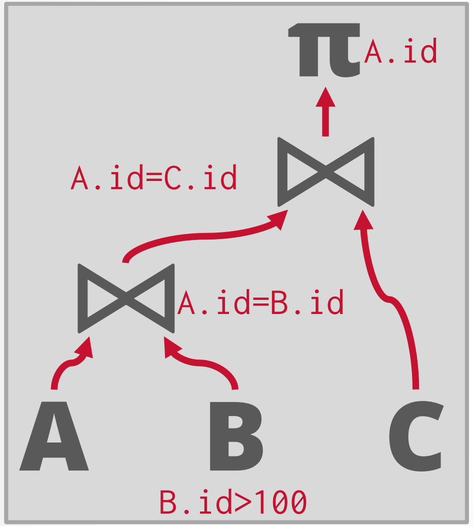
→ Some systems automatically build multi-column statistics if they are already used in an index together (MSSQL).

→ Otherwise, a human manually specifies target columns.

Also called Column Group Statistics (Db2) or Extended Statistics (Oracle).

ESTIMATION PROBLEM

```
SELECT A.id
  FROM A, B, C
 WHERE A.id = B.id
   AND A.id = C.id
   AND B.id > 100
```



Compute the cardinality of base tables

$\mathbf{A} \rightarrow |\mathbf{A}|$

$\mathbf{B}.\mathrm{id}>100 \rightarrow |\mathbf{B}| \times \mathit{sel}(\mathbf{B}.\mathrm{id}>100)$

$\mathbf{C} \rightarrow |\mathbf{C}|$

Compute the cardinality of join results

$$\mathbf{A} \bowtie \mathbf{B} = (|\mathbf{A}| \times |\mathbf{B}|) \;/\; \mathit{max}(\mathit{sel}(\mathbf{A}.\mathrm{id}=\mathbf{B}.\mathrm{id}), \mathit{sel}(\mathbf{B}.\mathrm{id}>100))$$

$$(\mathbf{A} \bowtie \mathbf{B}) \bowtie \mathbf{C} = (|\mathbf{A}| \times |\mathbf{B}| \times |\mathbf{C}|) \;/\; \mathit{max}(\mathit{sel}(\mathbf{A}.\mathrm{id}=\mathbf{B}.\mathrm{id}), \mathit{sel}(\mathbf{B}.\mathrm{id}>100), \mathit{sel}(\mathbf{A}.\mathrm{id}=\mathbf{C}.\mathrm{id}))$$

ESTIMATOR QUALITY

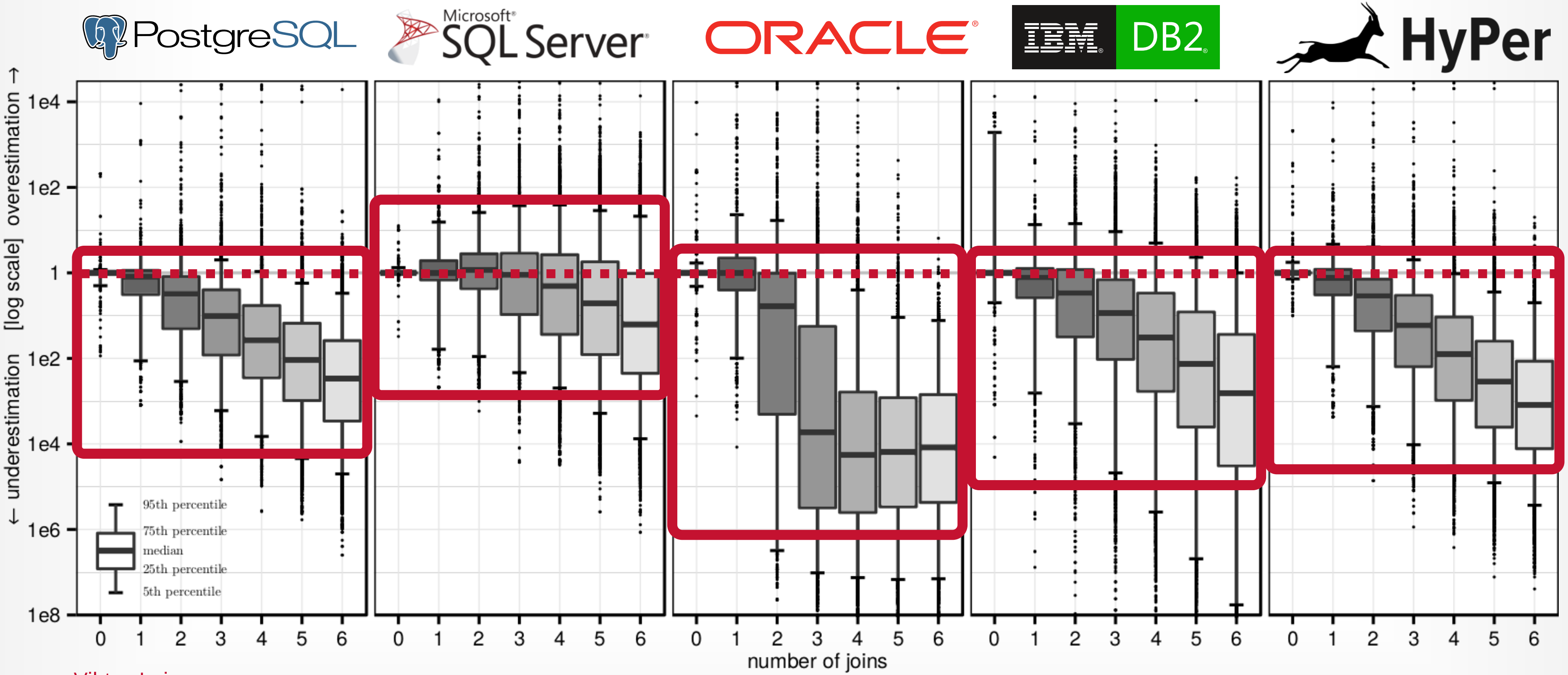
Evaluate the correctness of cardinality estimates generated by DBMS optimizers as the number of joins increases.

→ Let each DBMS perform its stats collection.
→ Extract measurements from query plan.

Compared five DBMSs using 100k queries from the JOB workload based on IMDB data.

HOW GOOD ARE QUERY OPTIMIZERS, REALLY?
VLDB 2015

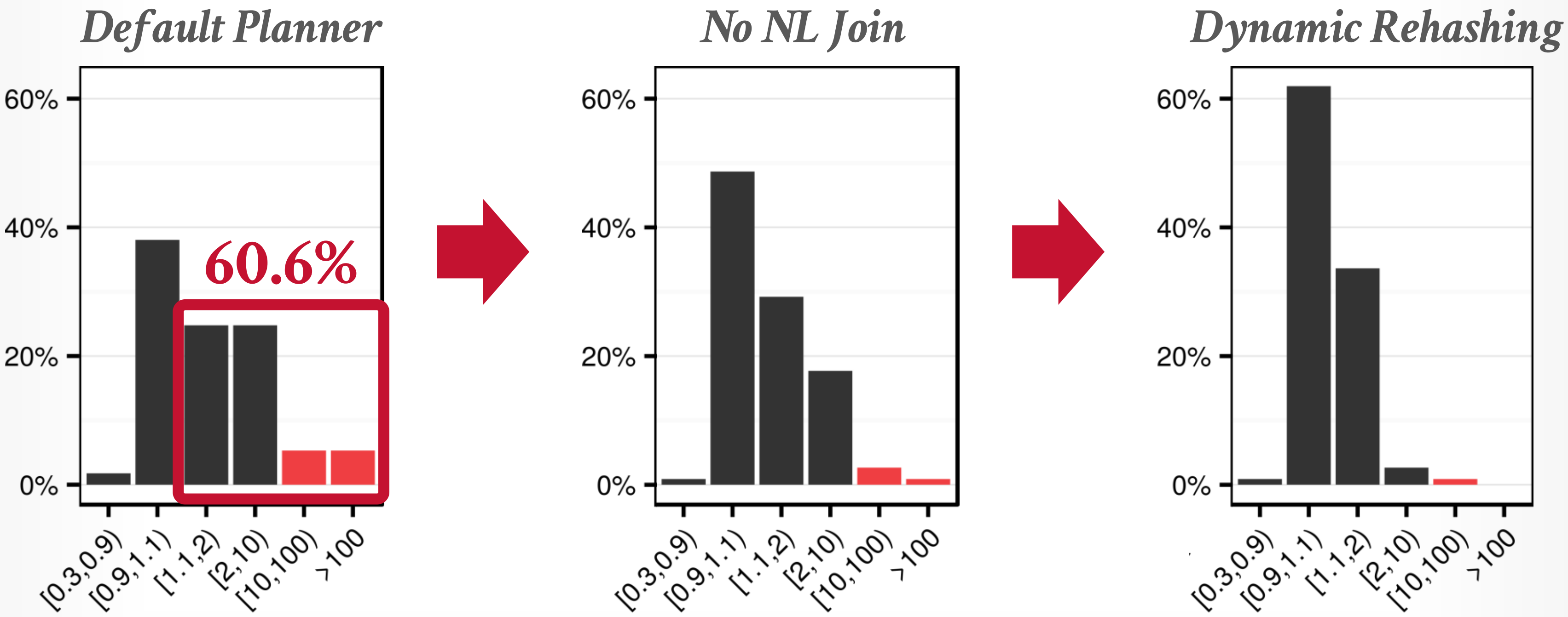
ESTIMATOR QUALITY



Source: Viktor Leis

EXECUTION SLOWDOWN

PostgreSQL v9.4 – JOB Workload



Slowdown compared to using true cardinalities

Source: [Viktor Leis]

LESSONS FROM THE GERMANS

Query opt 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 + seq scans are a robust exec 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
→ Better to improve cardinality estimation instead

Source: Viktor Leis

COST MODEL IMPLEMENTATIONS

PostgreSQL

IBM Db2

Smallbase (TimesTen)

DuckDB

POSTGRESQL 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.

19.7.2. Planner Cost Constants

The *cost* variables described in this section are measured on an arbitrary scale. Only their relative values matter, hence scaling them all up or down by the same factor will result in no change in the planner's choices. By default, these cost variables are based on the cost of sequential page fetches; that is, `seq_page_cost` is conventionally set to `1.0` and the other cost variables are set with reference to that. But you can use a different scale if you prefer, such as actual execution times in milliseconds on a particular machine.

> **Note:** Unfortunately, there is no well-defined method for determining ideal values for the cost variables. They are best treated as averages over the entire mix of queries that a particular installation will receive. This means that changing them on the basis of just a few experiments is very risky.

`seq_page_cost` (`floating point`)

Sets the planner's estimate of the cost of a disk page fetch that is part of a series of sequential fetches. The default is 1.0. This value can be overridden for tables and indexes in a particular tablespace by setting the tablespace parameter of the same name (see ALTER TABLESPACE).

`random_page_cost` (`floating point`)

IBM DB2 COST MODEL

Database characteristics in system catalogs

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

Source: Guy Lohman

SMALLBASE COST MODEL

Two-phase model that automatically generates hardware costs from a logical model.

Phase #1: Identify Execution Primitives

→ List of ops that the DBMS does when executing a query
→ Example: evaluating predicate, index probe, sorting.

Phase #2: Microbenchmark

→ On start-up, profile ops to compute CPU/memory costs
→ These measurements are used in formulas that compute operator cost based on table size.

MODELLING COSTS FOR A MM-DBMS
REAL-TIME DATABASES 1996

DUCKDB COST MODEL

Cannot assume there are statistics because the DBMS may be seeing a data file for the first.

When there are no statistics, the DBMS uses number of distinct values to determine worst-case cardinality estimation for joins.

→ Assumes primary-foreign key joins.
→ Assume independence and uniformity of data.
→ If HyperLogLog is available, use that when possible (e.g., `value=10`). Otherwise, assume 20% selectivity.

JOIN ORDER OPTIMIZATION WITH (ALMOST) NO STATISTICS
TOM EBERGEN - VRIJE UNIVERSITEIT MS THESIS (2022)

PARTING THOUGHTS

If every DBMS has the same execution engine design (i.e., Velox, DataFusion), then query optimization is what will distinguish one system from another. This is highly dependent on the efficacy of the cost model's cardinality estimation.

The combination of sampling + sketches are the way to achieve accurate cardinality estimations.

NEXT CLASS

[Project Status Update #2]