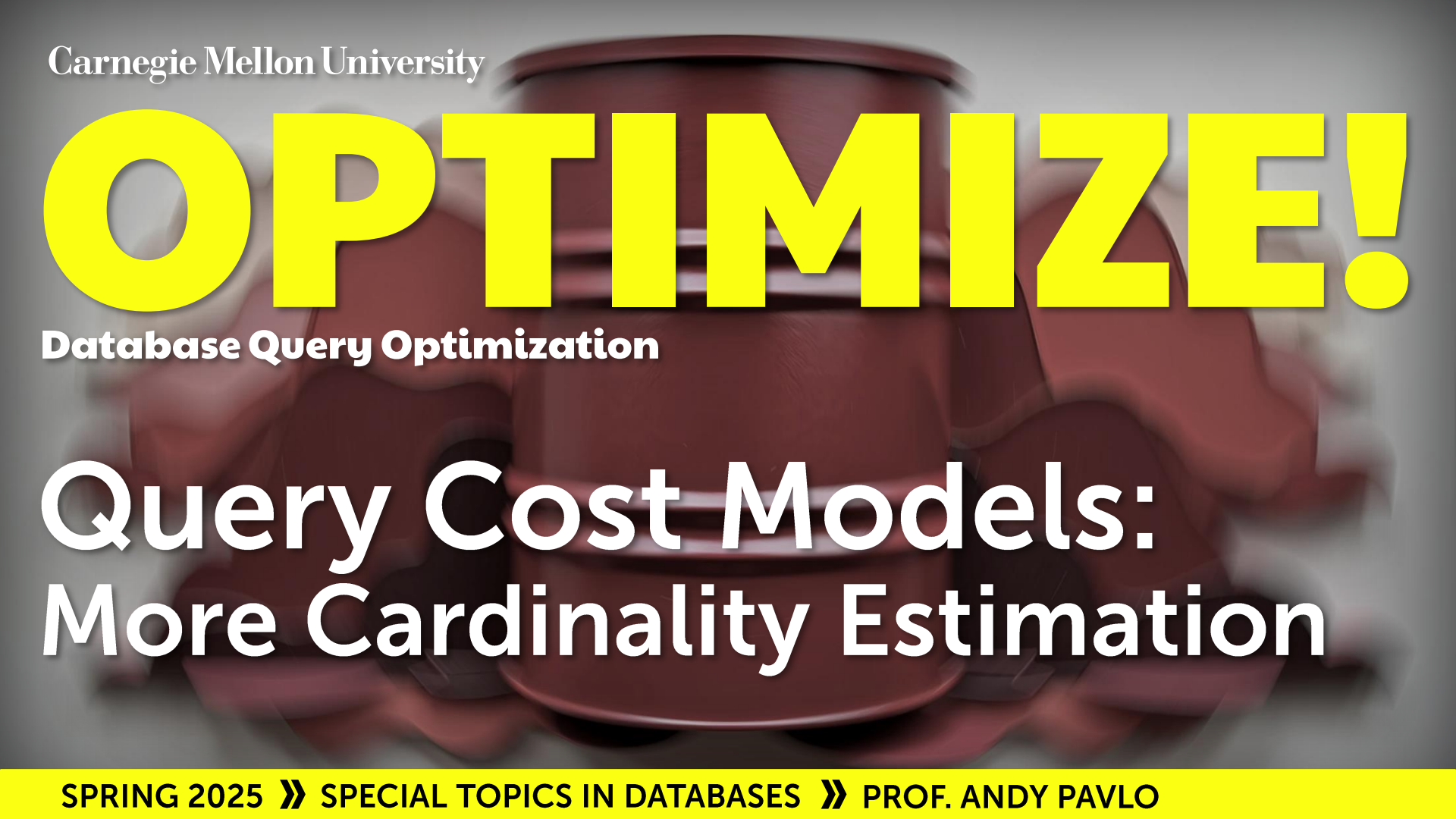

Carnegie Mellon University
OPTIMIZE!
Database Query Optimization
Query Cost Models:
More Cardinality Estimation
SPRING 2025 » SPECIAL TOPICS IN DATABASES » PROF. ANDY PAVLO

ADMINISTRIVIA

Andy still needs to email each group with feedback about their project proposals.

Project #2 status update presentations will be on Monday April 7th

→ Don't wait until then to get started or email me at the last minute about why your project isn't going to work.

UPCOMING DATABASE TALKS

QMDB (PDL/SDI)
→ Thursday Mar 13th @ 12:00pm ET
→ NSH 3305 (with pizza!)

Malloy (DB Seminar)
→ Monday Mar 17th @ 4:30pm ET
→ Zoom

PRQL (DB Seminar)
→ Monday Mar 24th @ 4:30pm ET
→ Zoom

LAST CLASS

We discussed the Germans' survey paper on the effects of cardinality estimation errors in the quality of query plans.

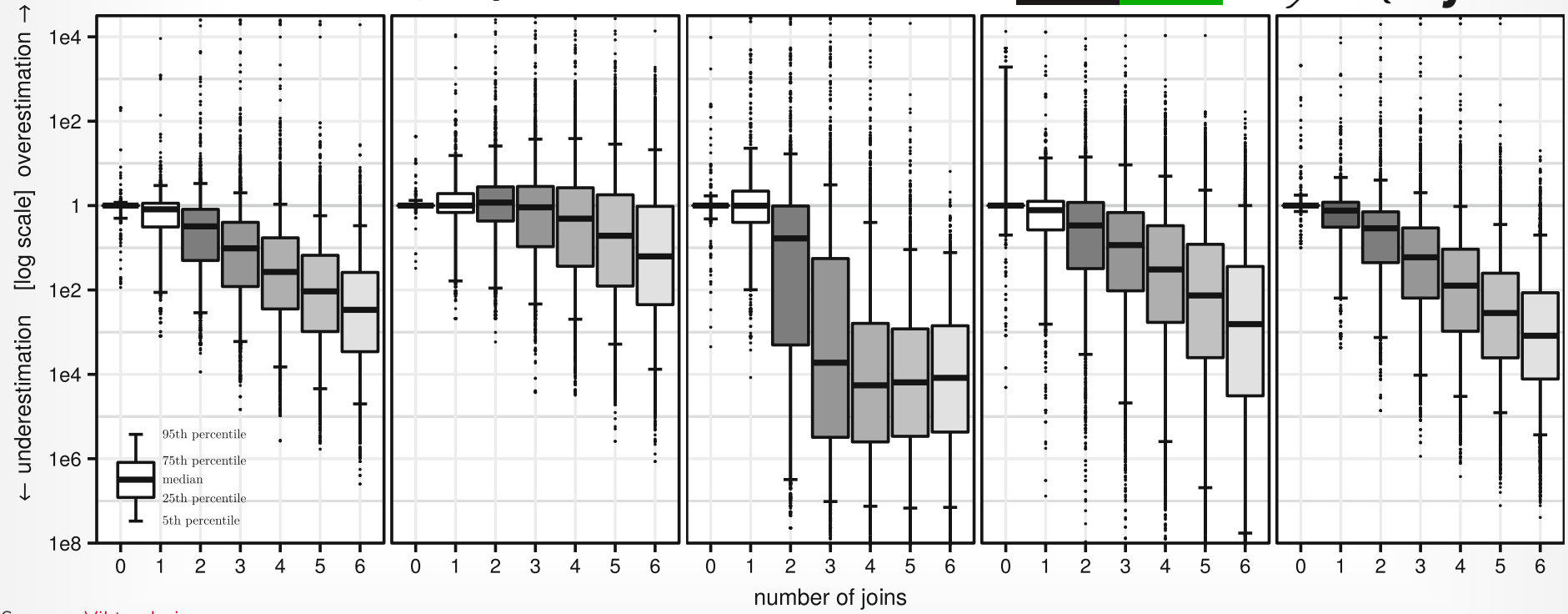
CARDINALITY ESTIMATION

Cardinality estimation is the process of predicting the number of rows that will be returned by a query operation, such as a filter or join, to help the optimizer choose the most efficient execution plan.

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

ESTIMATOR QUALITY

PostgreSQL · Microsoft SQL Server · ORACLE · IBM DB2 · HyPer



Source: Viktor Leis

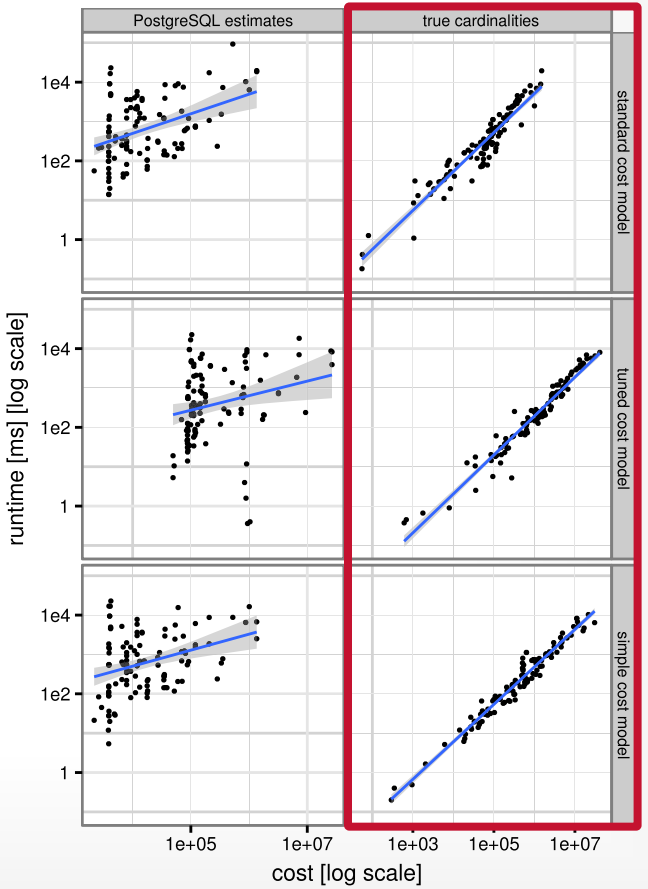
POSTGRES COST MODEL

Cost of an operator is a weighted sum of the # of accessed disk pages and the amount of data processed in memory.

→ Distinguishes between sequential and random I/O.

→ Requires manual tuning the weights based on hardware characteristics.

The Germans replaced Postgres' cost model with simple cost model to see what happens with query plans…



Source: Viktor Leis

OBSERVATION

JOB's workload only contains queries with the **SELECT-PROJECT-JOIN** (SPJ) pattern with indexes on primary and foreign keys.

Source: Kukjin Lee

OBSERVATIO

JOB's workload only contains queri
SELECT-PROJECT-JOIN (SPJ) p
indexes on primary and foreign ke

```
SELECT MIN(an.name) AS alternative_name,
       MIN(chn.name) AS character_name,
       MIN(t.title) AS movie
FROM aka_name AS an,
     char_name AS chn,
     cast_info AS ci,
     company_name AS cn,
     movie_companies AS mc,
     name AS n,
     role_type AS rt,
     title AS t
WHERE ci.note IN ('(voice)',
                  '(voice: Japanese version)',
                  '(voice) (uncredited)',
                  '(voice: English version)')
  AND cn.country_code ='[us]'
  AND mc.note IS NOT NULL
  AND (mc.note LIKE '%(USA)%'
       OR mc.note LIKE '%(worldwide)%')
  AND n.gender ='f'
  AND n.name LIKE '%Ang%'
  AND rt.role ='actress'
  AND t.production_year BETWEEN 2005 AND 2015
  AND ci.movie_id = t.id
  AND t.id = mc.movie_id
  AND ci.movie_id = mc.movie_id
  AND mc.company_id = cn.id
  AND ci.role_id = rt.id
  AND n.id = ci.person_id
  AND chn.id = ci.person_role_id
  AND an.person_id = n.id
  AND an.person_id = ci.person_id;
```

Source: Kukjin Lee

OBSERVATION

JOB's workload only contains queries with the **SELECT-PROJECT-JOIN** (SPJ) pattern with indexes on primary and foreign keys.

The Germans' survey only injected the true cardinalities for all operators in a query plan.

Source: Kukjin Lee

OBSERVATION

JOB's workload only contains queries with the **SELECT-PROJECT-JOIN** (SPJ) pattern with indexes on primary and foreign keys.

The Germans' survey only injected the true cardinalities for all operators in a query plan.

Some DBMSs can mask the adverse effects of an optimizer's bad cardinality estimates via adaptivity.

Source: Kukjin Lee

TODAY'S AGENDA

Background

Row-store Evaluation

Column-store Evaluation

Bitmap Filtering

Adaptive Joins

MSSQL CARDINALITY ESTIMATION SURVEY

Measure how cardinality estimation errors effect query plan quality in a DBMS with a state-of-the-art optimizer and execution engine.

→ Compare optimizer behavior when the database contains row-oriented (b+trees) vs. columnar indexes (bitmaps).

→ Evaluated both synthetic and real-world workloads.

Identify the minimal subset of logical expressions that need accurate estimates to generate the best plan.

ANALYZING THE IMPACT OF CARDINALITY ESTIMATION ON EXECUTION PLANS IN MICROSOFT SQL SERVER
VLDB 2023

MSSQL: CARDINALITY INJECTION

Compute a hash for each logical expression in the optimizer's memo.

→ Recursively hash children expressions.

Reverse (decode) each logical expression back into a SQL statement.

→ Again, recursively decode children expressions.

Execute SQL and store in catalog. Then use hash during Cascades search to retrieve true cardinality.

WORKLOADS

Cardinality Estimation Benchmarks

→ JOB, CEB, STATS

Industry Benchmarks

→ TPC-DS, TPC-H, DSB

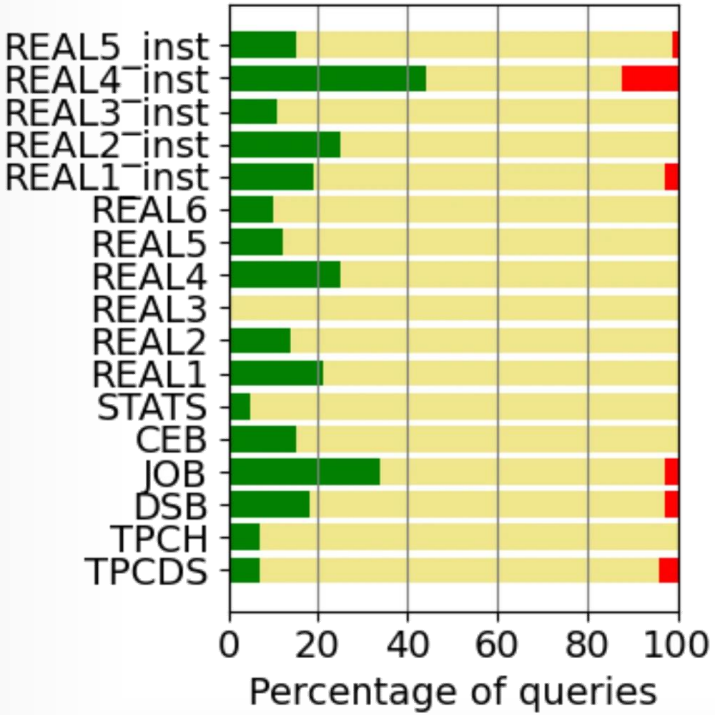
√ Group By
√ Outer Join
√ Nested Queries
√ CTE
√ Union/Intersect

Real Workloads

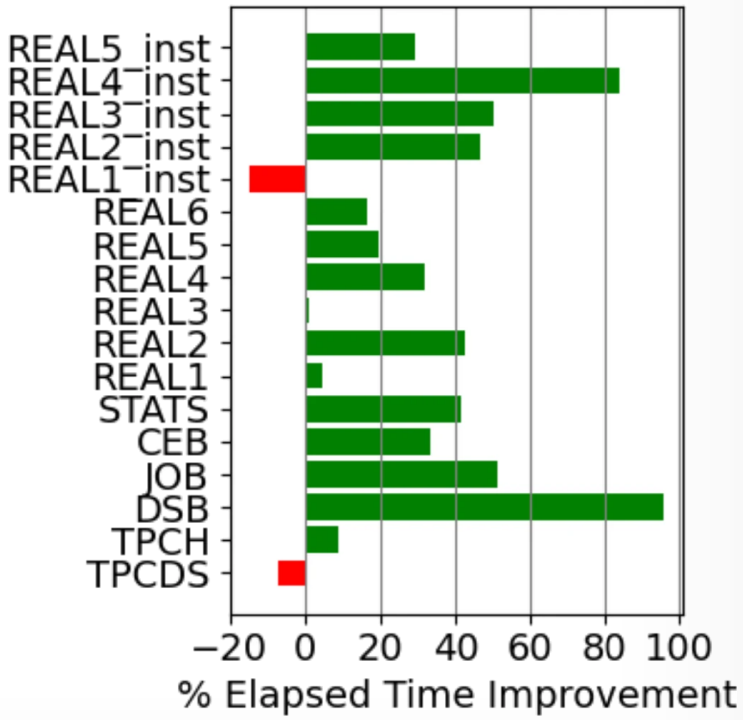
→ Internal and external applications running on MSSQL.
→ Raw query traces + parameterized query instances.

ROWSTORE EVALUATION

% of Queries Changed by 2x or more



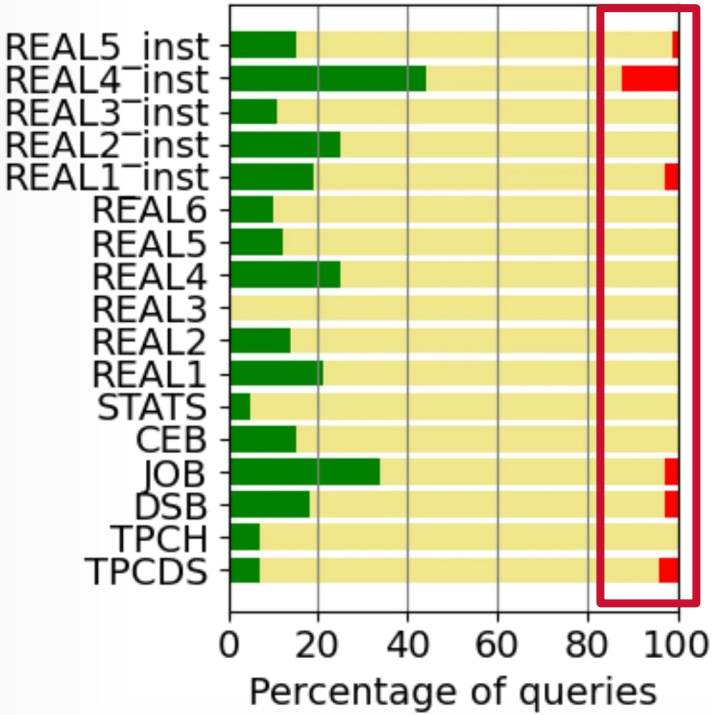
Workload Improvement



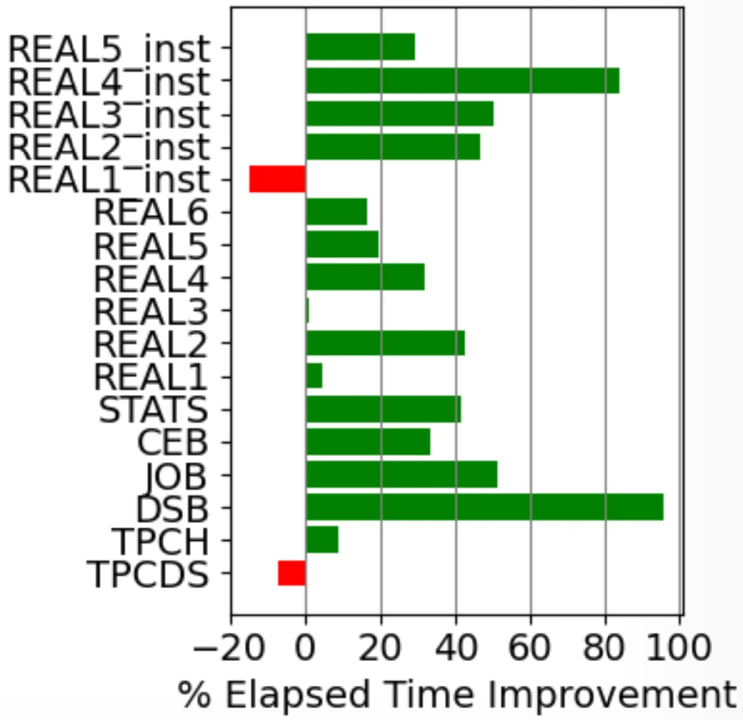
Source: Vivek Narasayya

ROWSTORE EVALUATION

% of Queries Changed by 2x or more



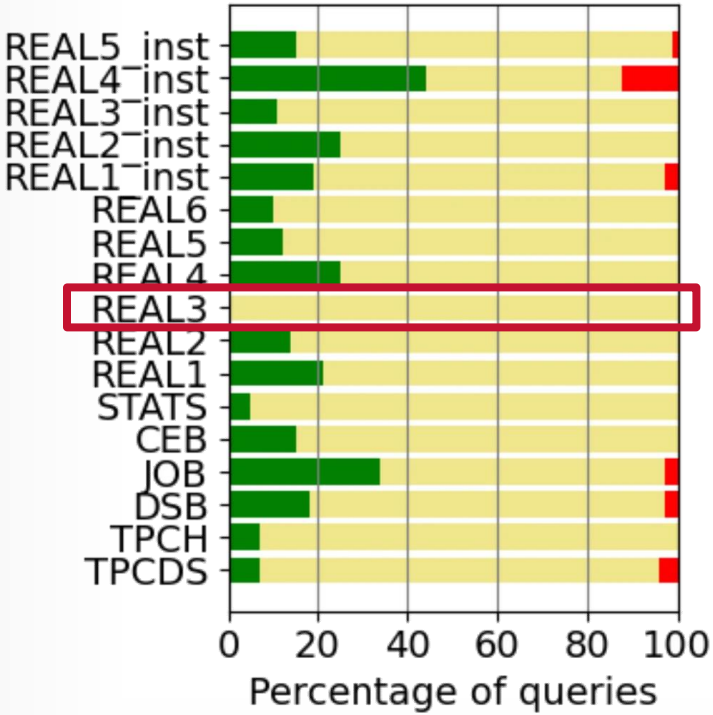
Workload Improvement



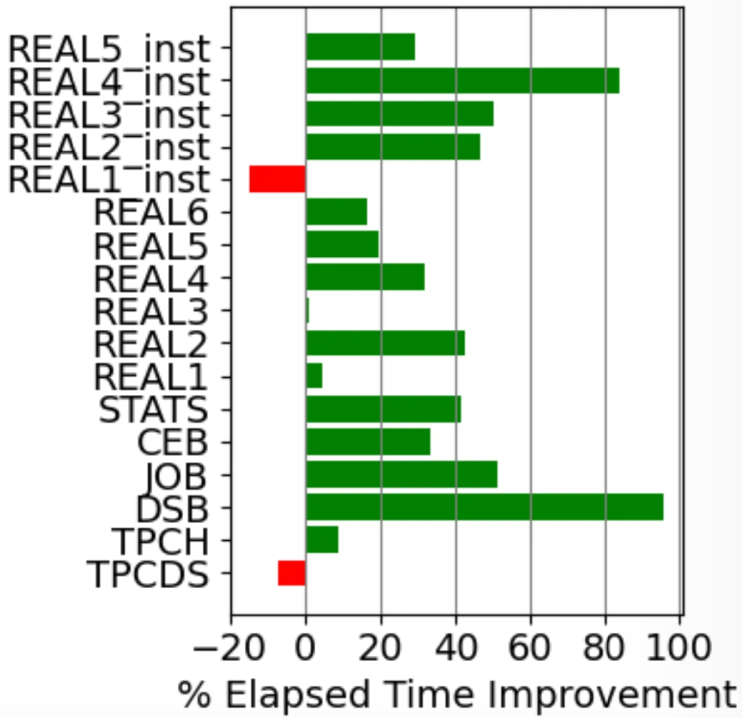
Source: Vivek Narasayya

ROWSTORE EVALUATION

% of Queries Changed by 2x or more

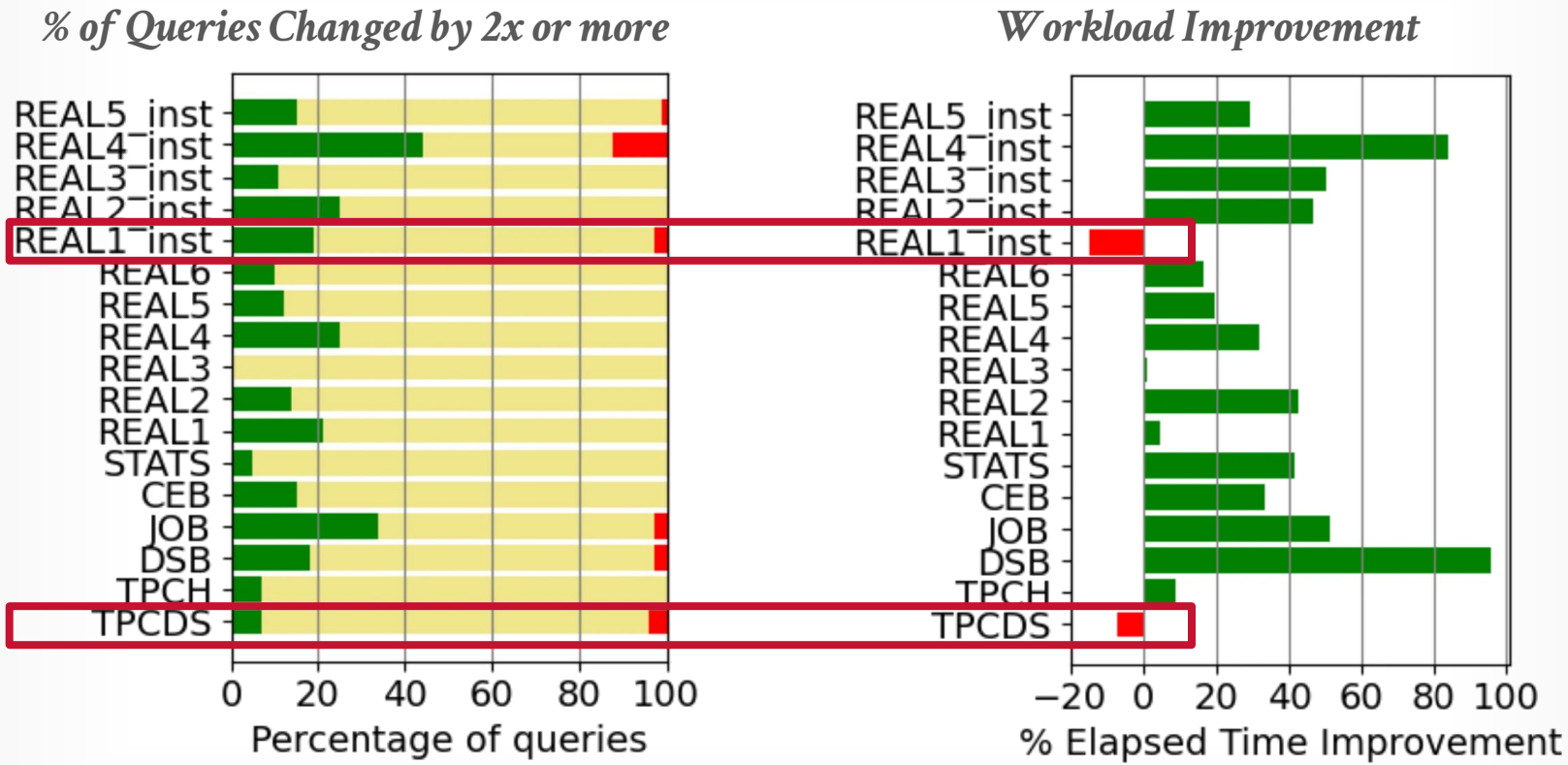


Workload Improvement



Source: Vivek Narasayya

ROWSTORE EVALUATION



Source: Vivek Narasayya

ROWSTORE DRILLDOWN

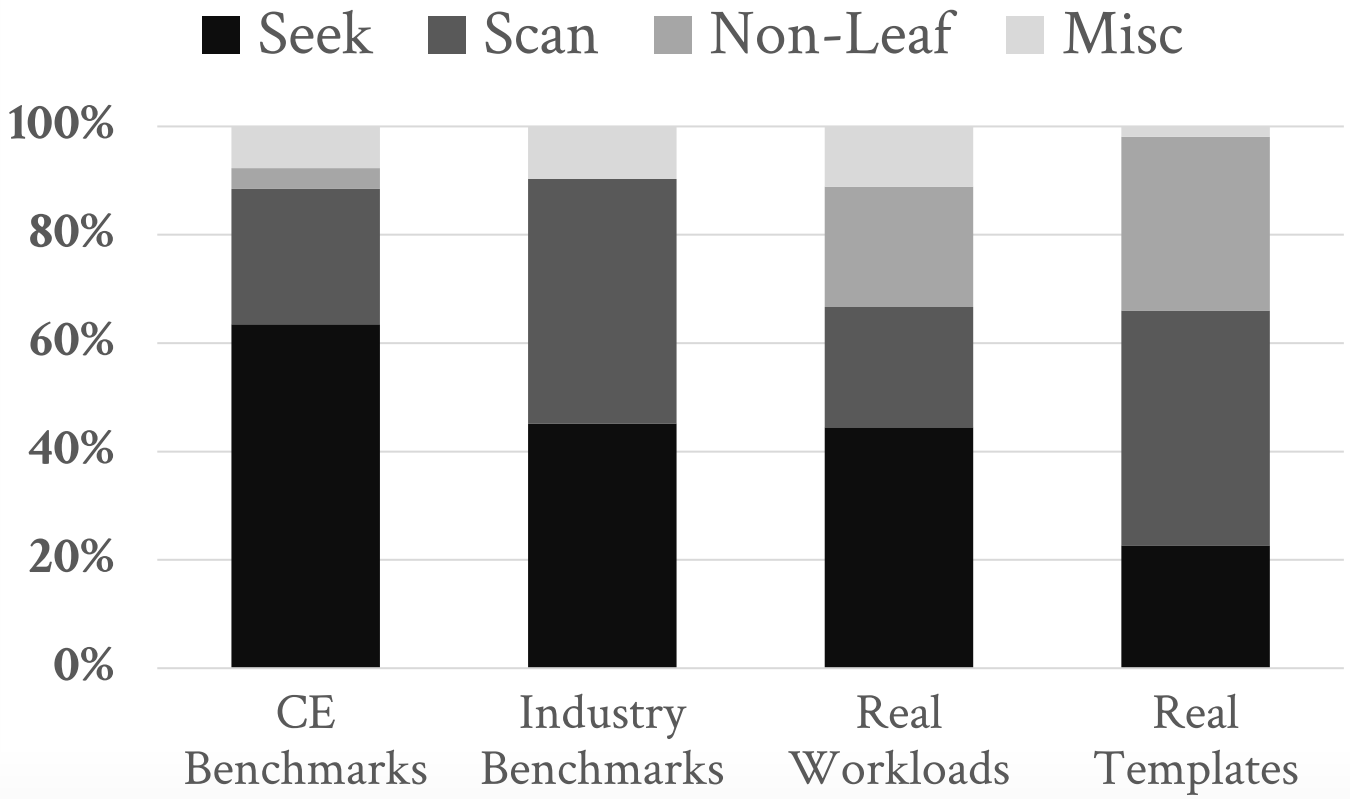
The next question to answer is why queries get faster with better cardinality estimation.

→ It's not just because the optimizer picks a better join order.
→ The Germans observed that slow plans in Postgres were due to excessive use of index scans.

For each query improved by 2x or more, categorize based on which class of operators contribute the majority of the improvement:

→ Seek Operators
→ Scan Operators
→ Non-Leaf Operators
→ Miscellaneous / Catch-All

ROWSTORE DRILLDOWN



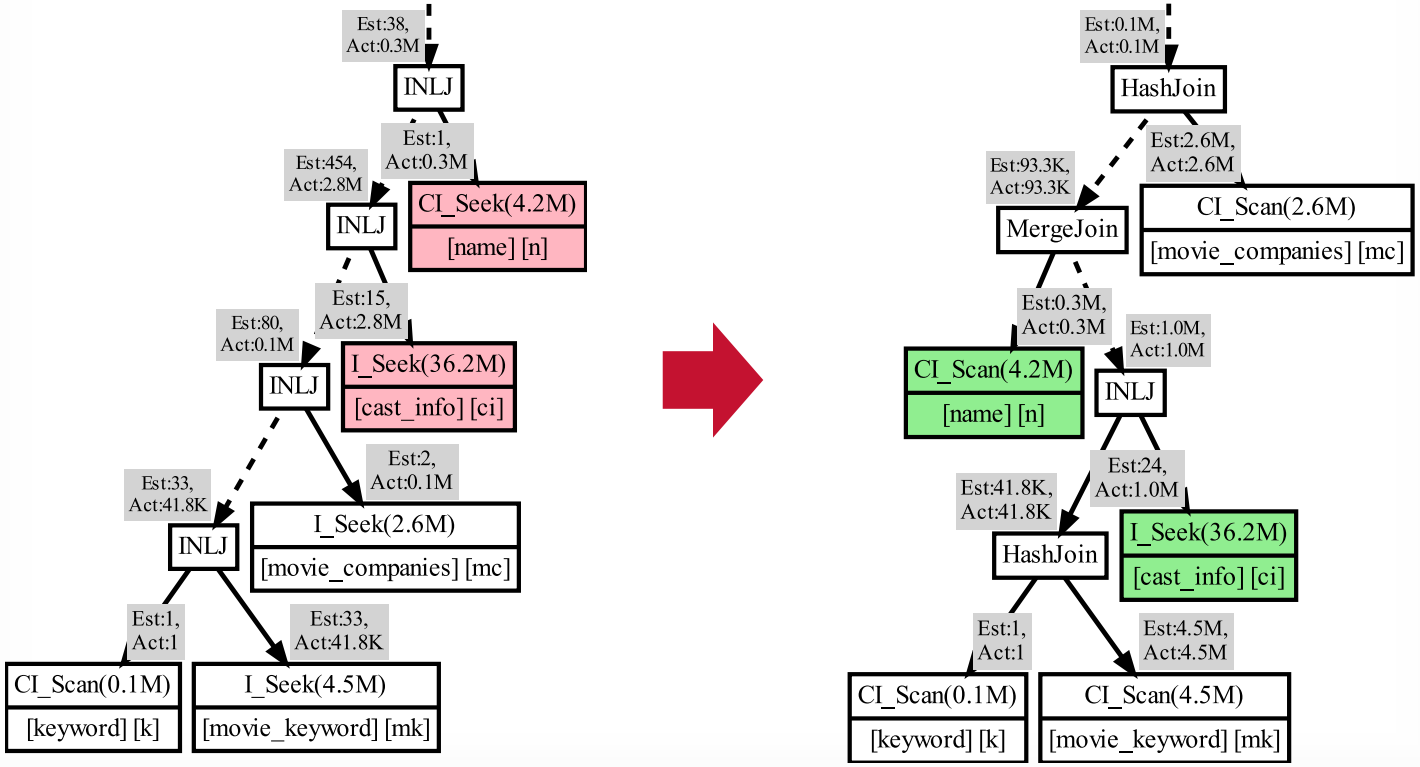
Source: Vivek Narasayya

JOB QUERY 17A: PLAN IMPROVEMENTS

```
SELECT min(n.name) AS member_in_charnamed_american_movie,
       min(n.name) AS a1
FROM cast_info AS ci,
     company_name AS cn,
     keyword AS k,
     movie_companies AS mc,
     movie_keyword AS mk,
     name AS n,
     title AS t
WHERE cn.country_code ='[us]'
  AND k.keyword ='character-name-in-title'
  AND n.name LIKE 'B%'
  AND n.id = ci.person_id
  AND ci.movie_id = t.id
  AND t.id = mk.movie_id
  AND mk.keyword_id = k.id
  AND t.id = mc.movie_id
  AND mc.company_id = cn.id
  AND ci.movie_id = mc.movie_id
  AND ci.movie_id = mk.movie_id
  AND mc.movie_id = mk.movie_id;
```

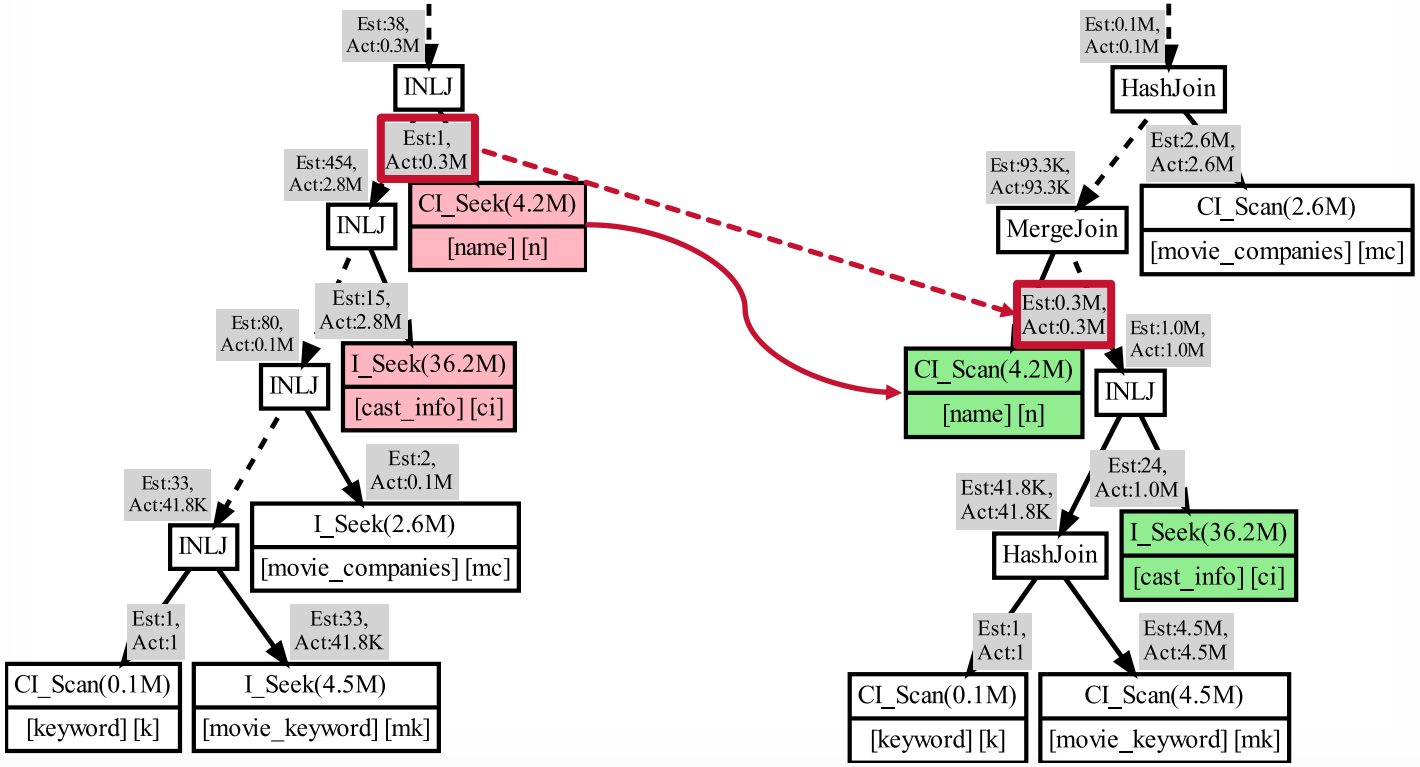
Source: Vivek Narasayya

JOB QUERY 17A: PLAN IMPROVEMENTS



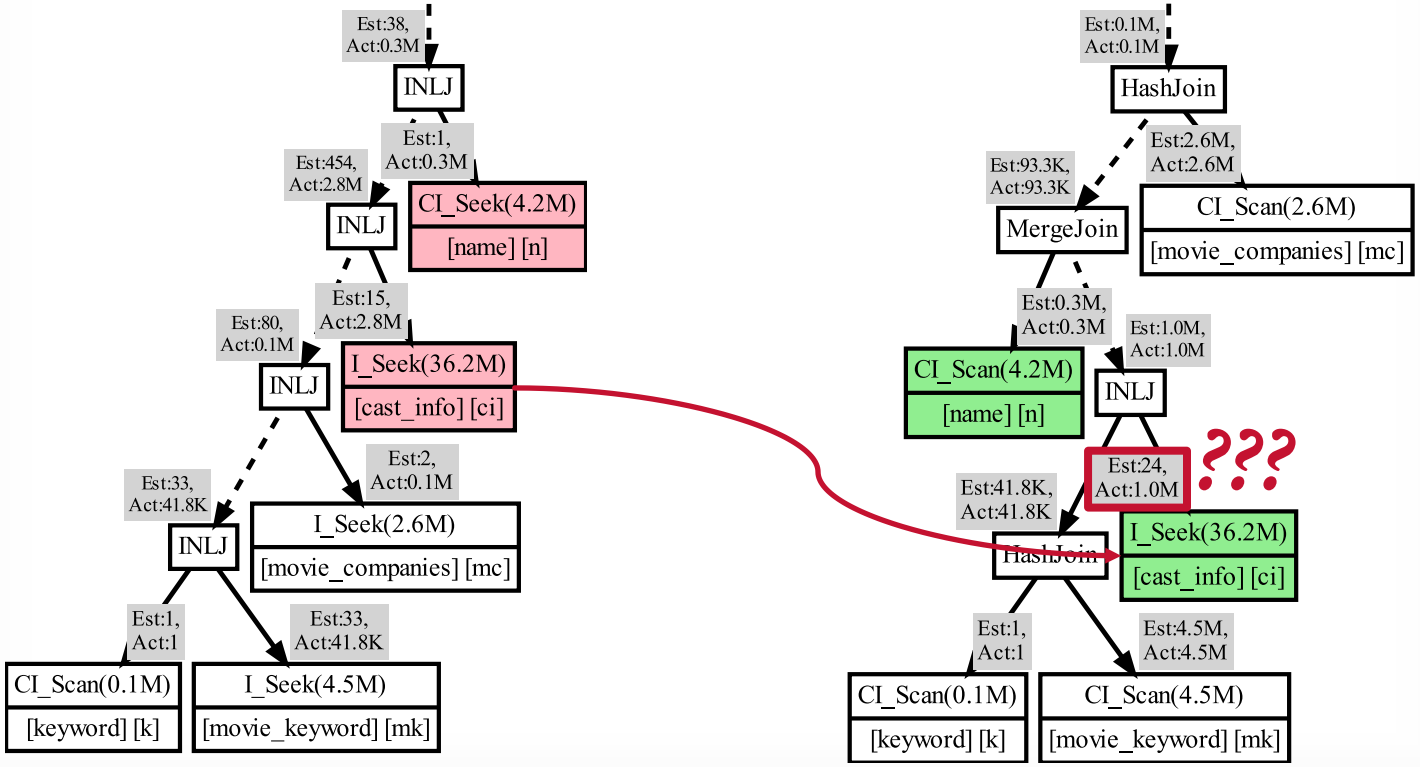
Source: Vivek Narasayya

JOB QUERY 17A: PLAN IMPROVEMENTS



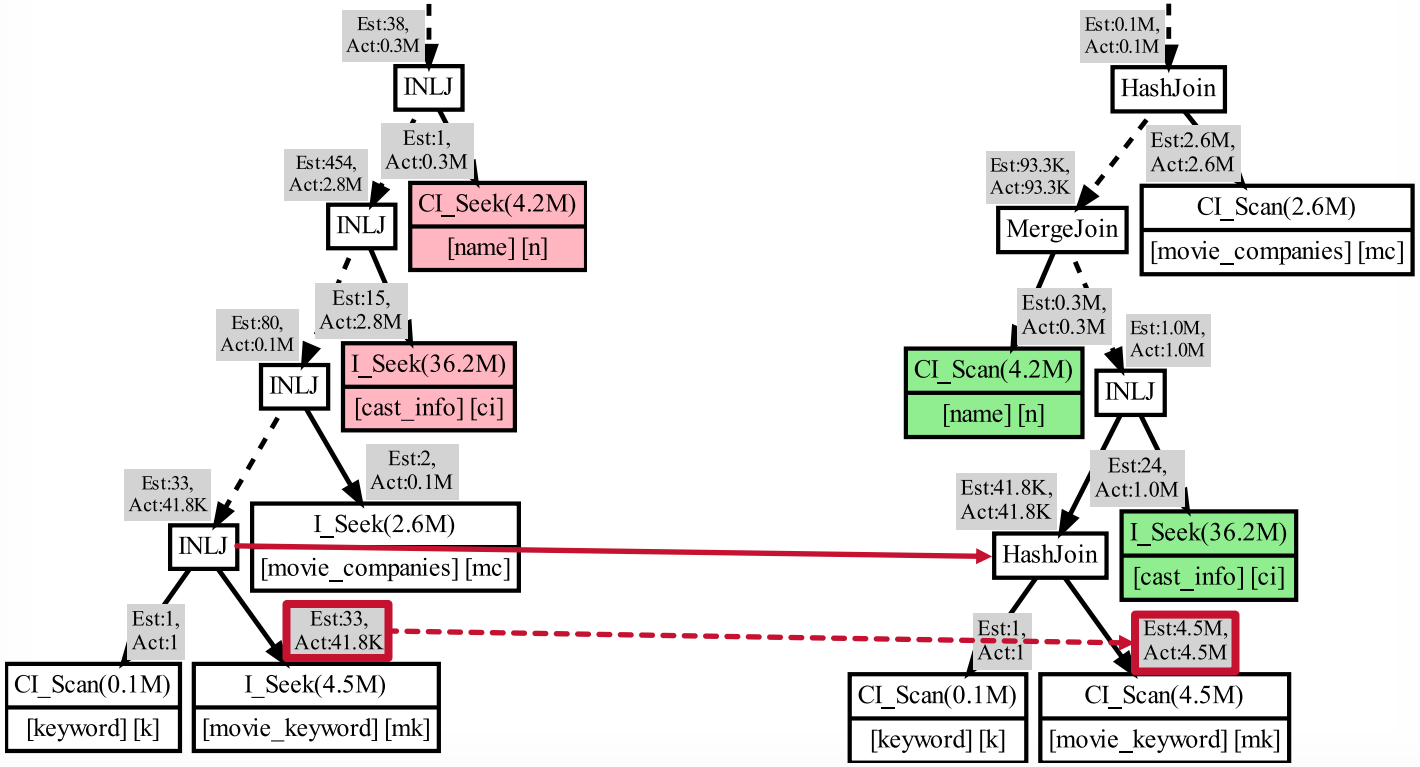
Source: Vivek Narasayya

JOB QUERY 17A: PLAN IMPROVEMENTS



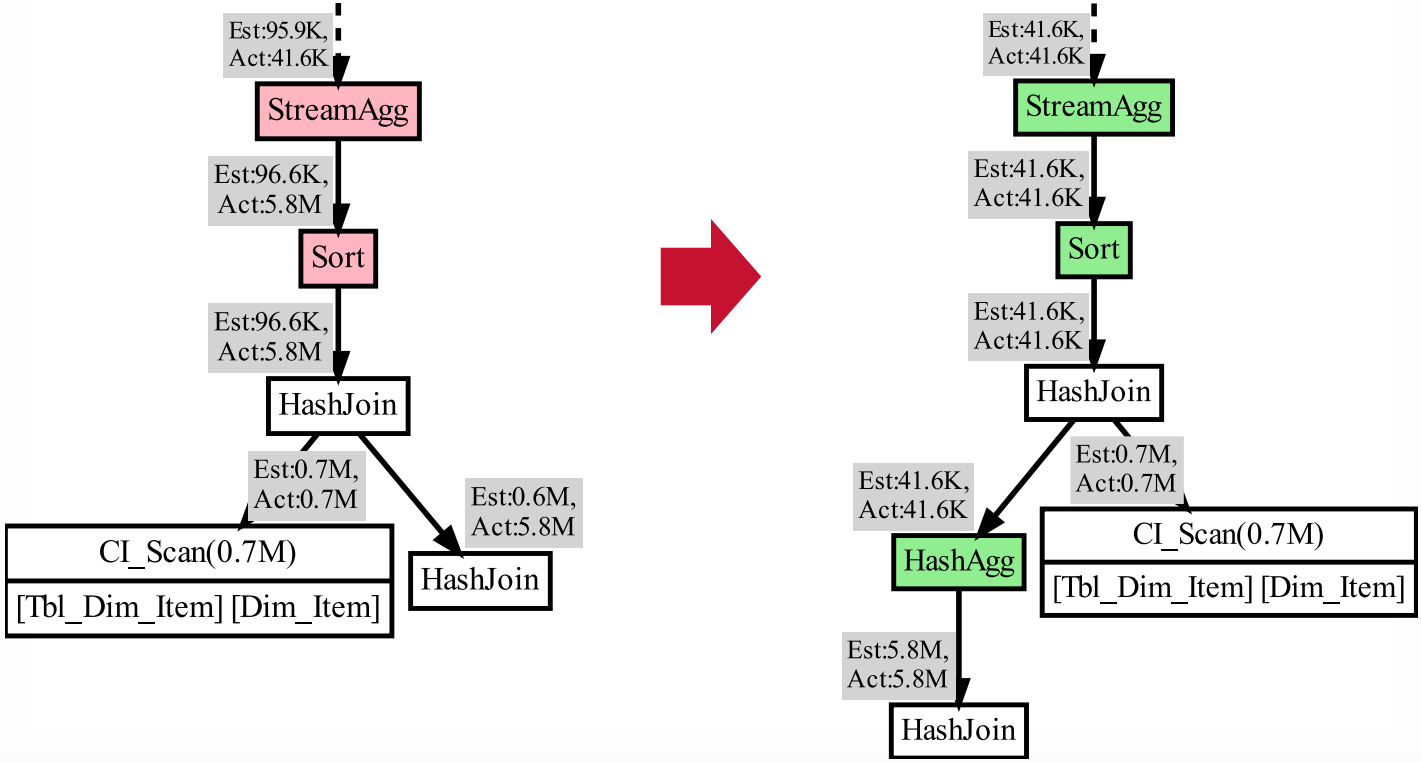
Source: Vivek Narasayya

JOB QUERY 17A: PLAN IMPROVEMENTS



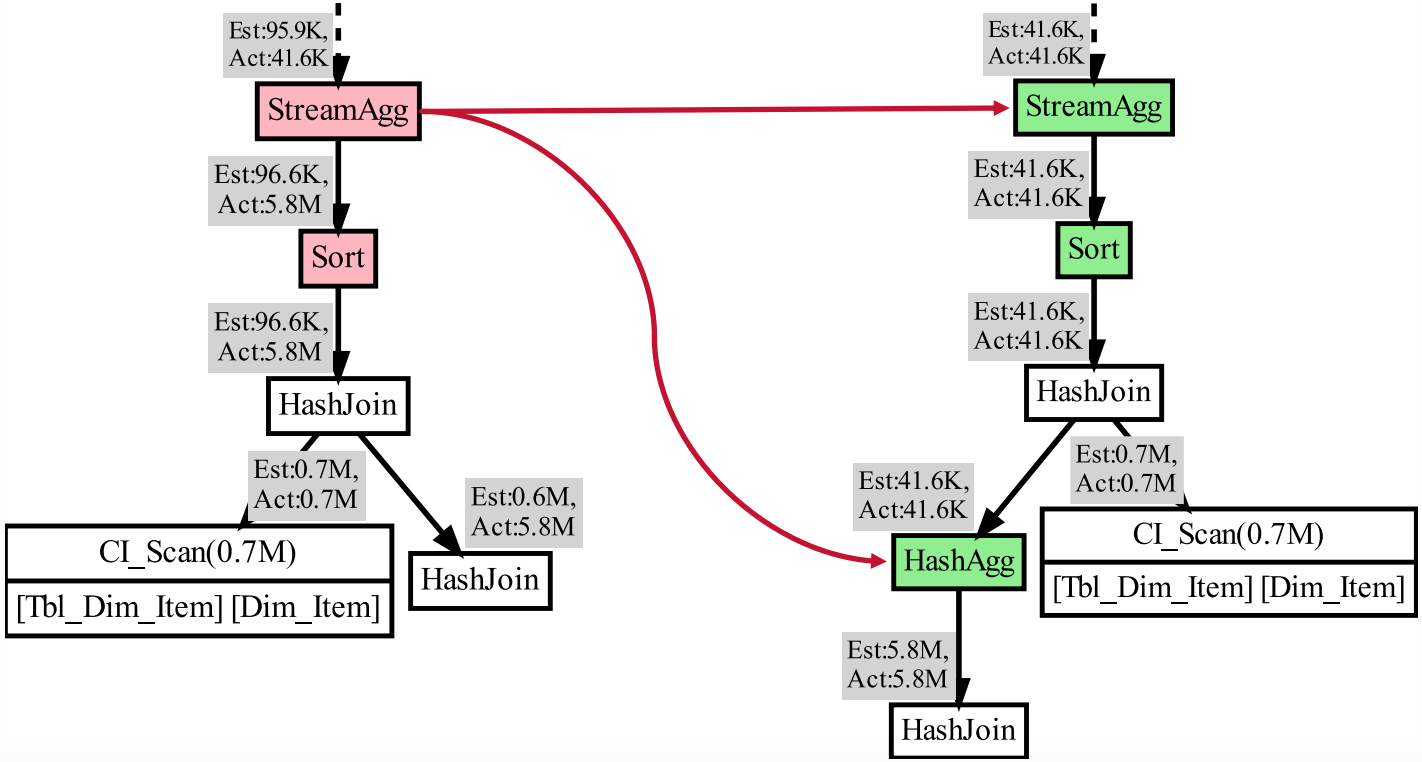
Source: Vivek Narasayya

REAL1 QUERY: AGGREGATION PUSHDOWN



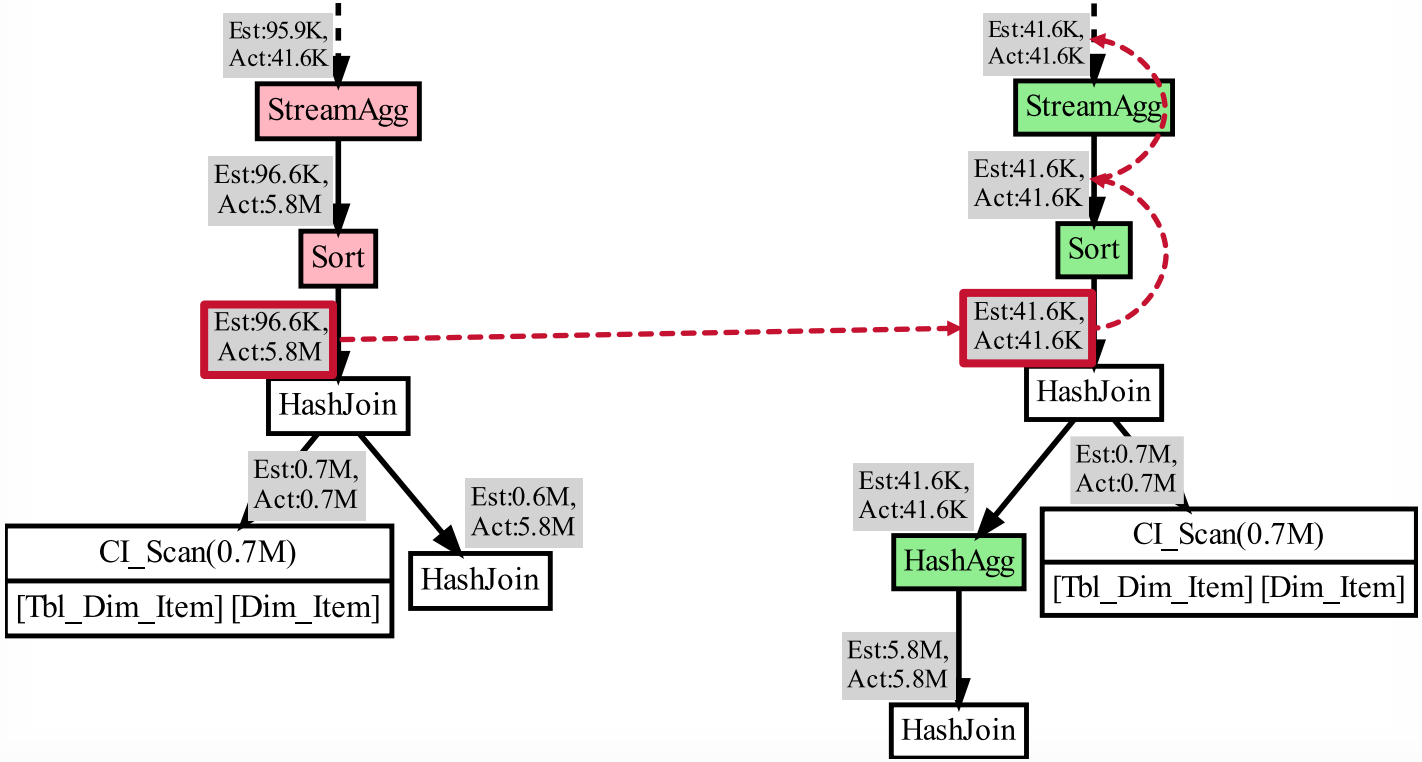
Source: Vivek Narasayya

REAL1 QUERY: AGGREGATION PUSHDOWN



Source: Vivek Narasayya

REAL1 QUERY: AGGREGATION PUSHDOWN



Source: Vivek Narasayya

REASON FOR CE ERRORS

In real workloads, the most common reason for cardinality estimation errors that lead to poor plan s is data skew in one or more join columns.

This leads to heavy over/under-estimation based on the skew value and whether the query plan's filter operators will remove (in)frequent values.

MSSQL: COLUMNSTORE

In the early 2010s, MSSQL added compressed columnstore indexes.

→ Originally used fractured mirror approach where original data remains in row store.

→ Later added ability to skip row and use columnstore indexes as core table storage.

Supports **INSERT**, **UPDATE**, and **DELETE** operations:

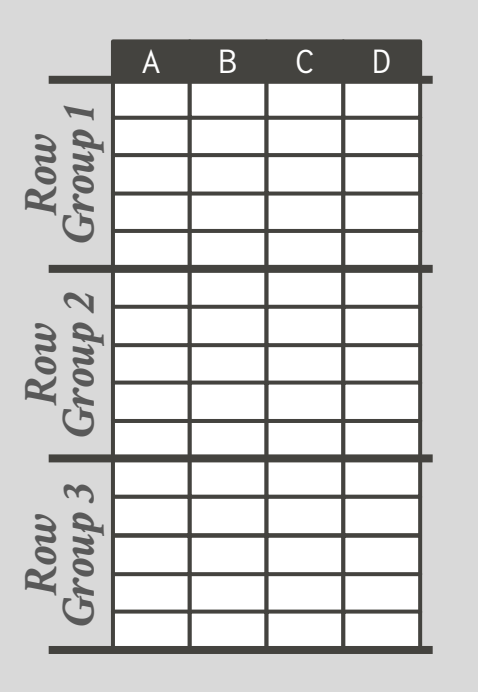
→ Use a **delta store** for modifications. DBMS seamlessly combines results from columnstore and delta store.

→ Deleted tuples are marked in a bitmap.

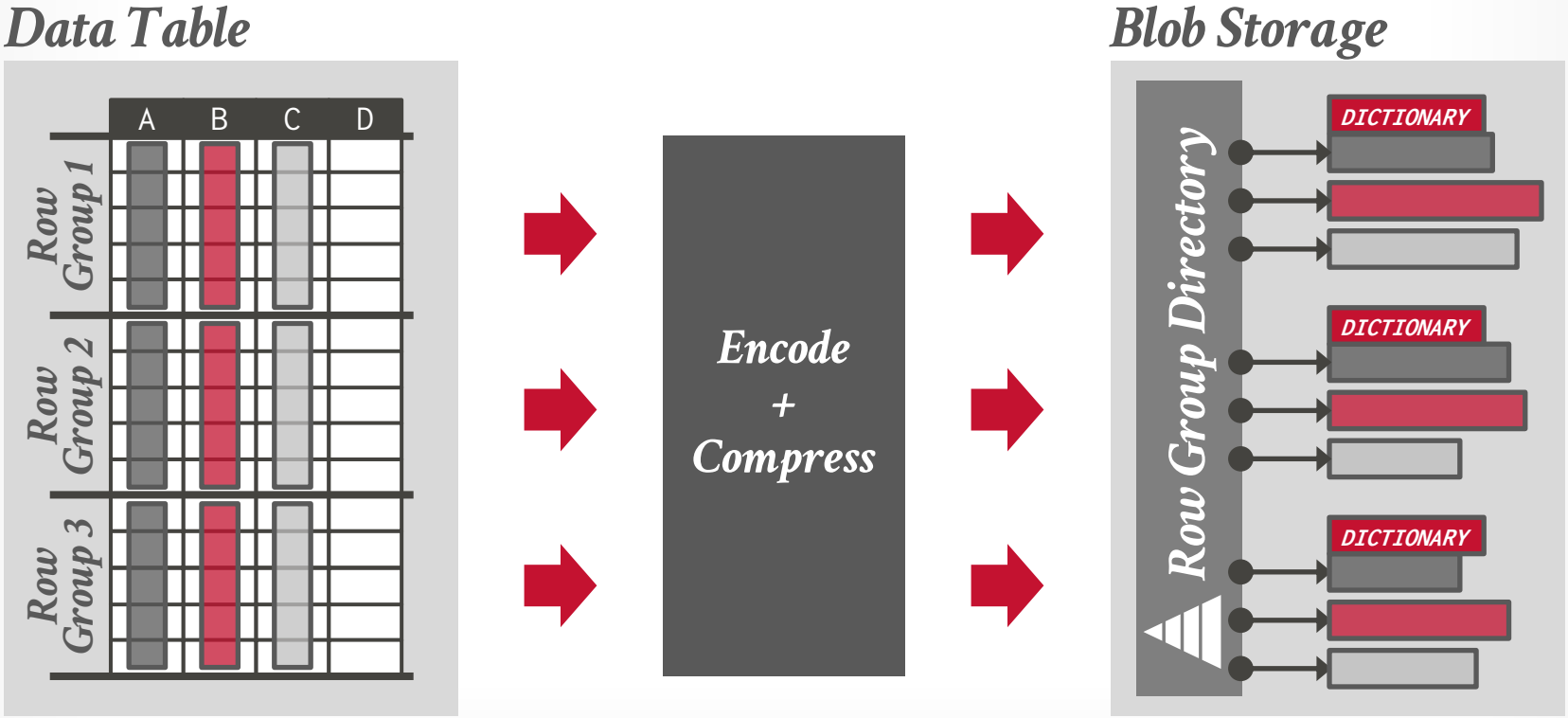
SQL SERVER COLUMN STORE INDEXES
SIGMOD 2011

MSSQL: COLUMNSTORE

Data Table



MSSQL: COLUMNSTORE



MSSQL: QUERY PROCESSING

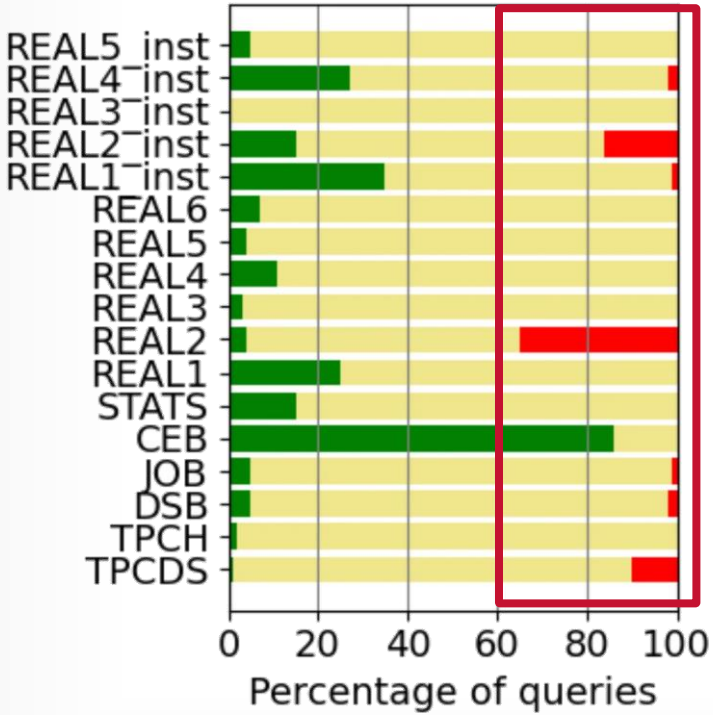
The DBMS's optimizer is aware of the columnar indexes and can choose them over row-store indexes when appropriate.

Even though the original MSSQL execution engine uses row-oriented / Volcano model, MSFT added vector-at-a-time operators operate directly on columnar indexes.

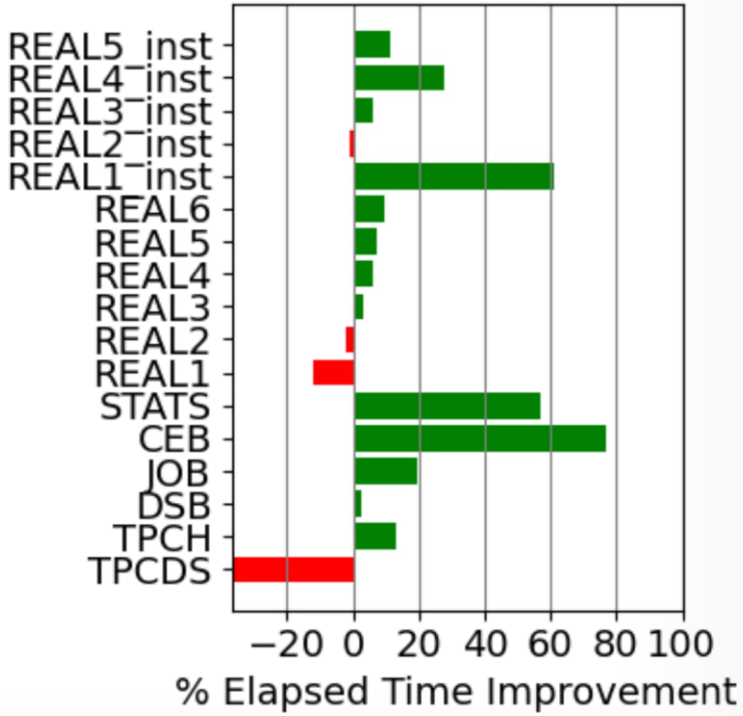
ENHANCEMENTS TO SQL SERVER COLUMN STORES
SIGMOD 2013

COLUMNSTORE EVALUATION

% of Queries Changed by 2x or more



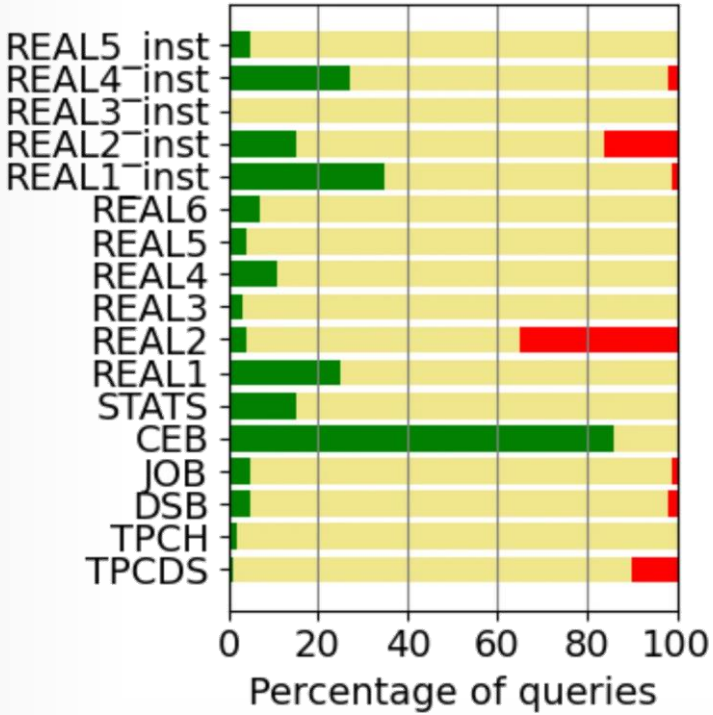
Workload Improvement



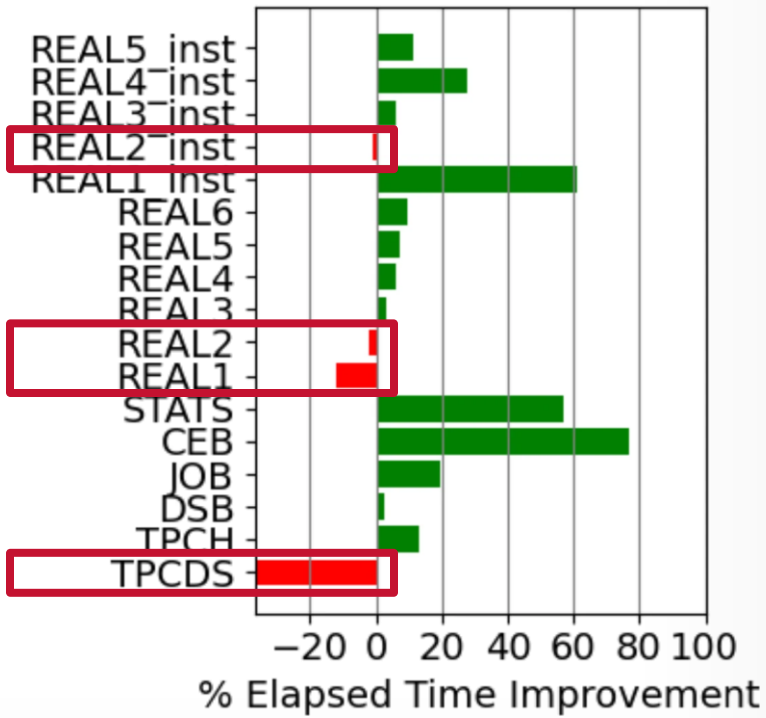
Source: Vivek Narasayya

COLUMNSTORE EVALUATION

% of Queries Changed by 2x or more



Workload Improvement



Source: Vivek Narasayya

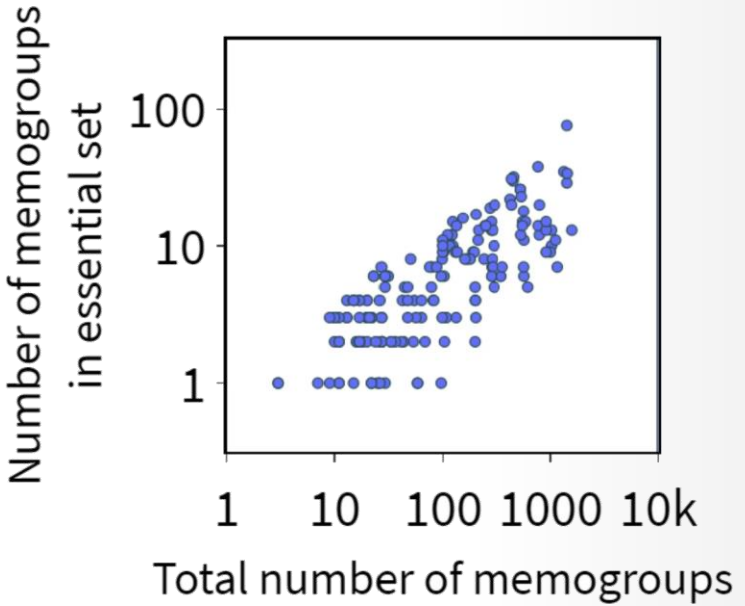
ESSENTIAL LOGICAL GROUPS

The set of logical groups in the optimizer's memo is ***essential*** iff:

→ Injecting the set's groups with exact estimates yields the same plan as when all groups have exact estimates.

→ It is the minimal set of essential groups.

The size of a query's essential set is small (less than 10%, median is 5).



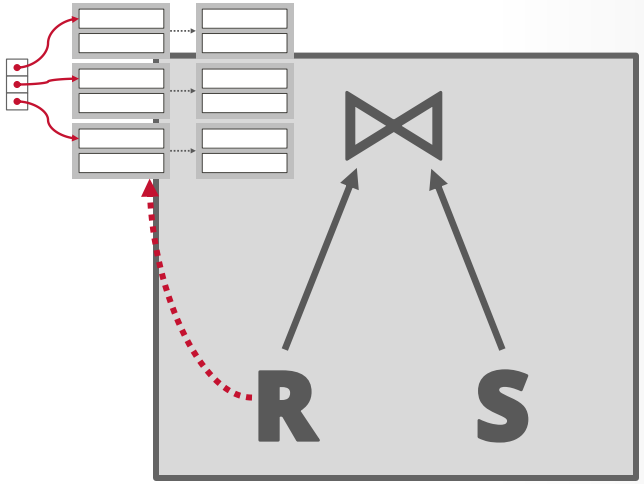
Source: Vivek Narasayya

BITMAP FILTERING

Create a probe filter (Bloom Filter) as the DBMS builds the hash table on the outer table in the first phase.

→ Always check the filter before probing the hash table.

→ Faster than probing hash table because the filter fits in CPU cache.



This technique is sometimes called ***sideways information passing.***

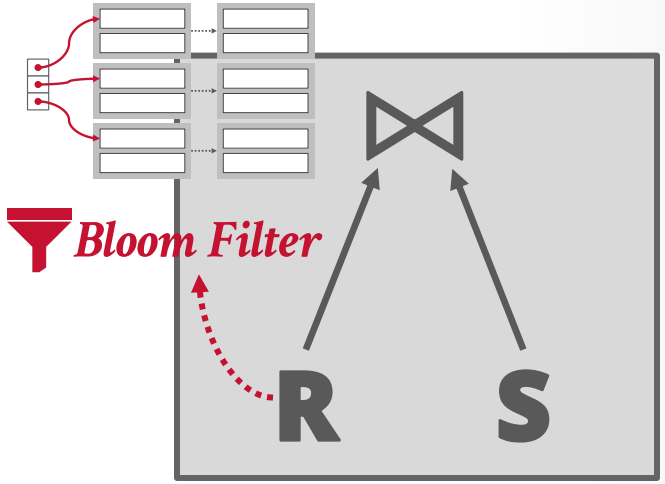
MICRO ADAPTIVITY IN VECTORWISE
SIGMOD 2013

BITMAP FILTERING

Create a probe filter (Bloom Filter) as the DBMS builds the hash table on the outer table in the first phase.

→ Always check the filter before probing the hash table.

→ Faster than probing hash table because the filter fits in CPU cache.



This technique is sometimes called ***sideways information passing.***

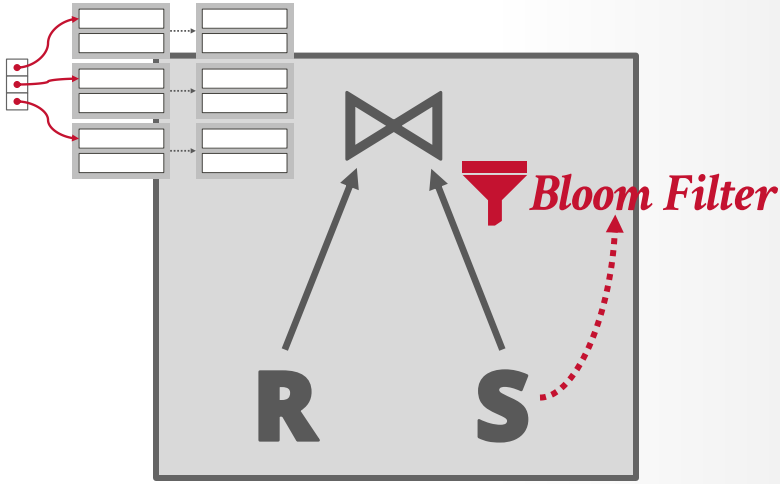
MICRO ADAPTIVITY IN VECTORWISE
SIGMOD 2013

BITMAP FILTERING

Create a probe filter (Bloom Filter) as the DBMS builds the hash table on the outer table in the first phase.

→ Always check the filter before probing the hash table.

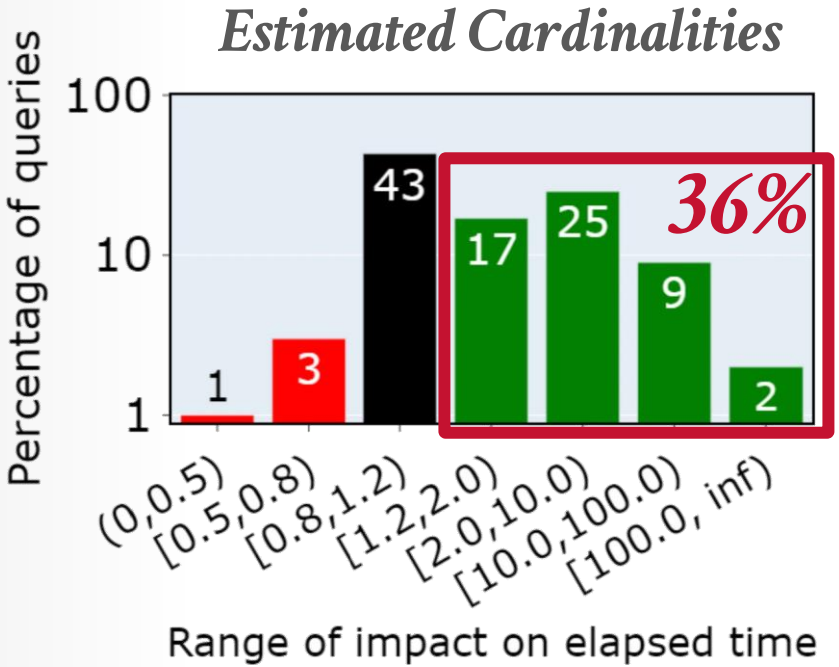
→ Faster than probing hash table because the filter fits in CPU cache.

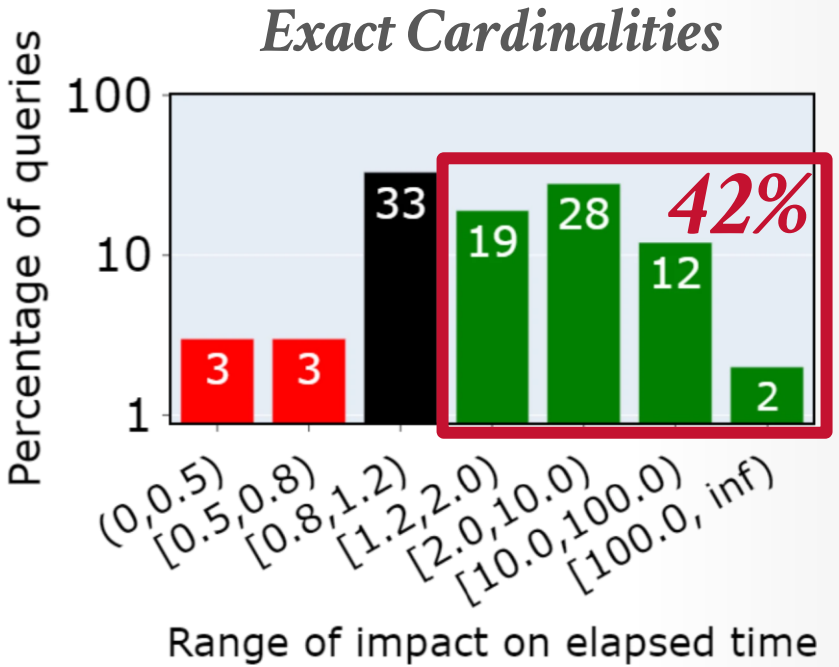


This technique is sometimes called ***sideways information passing.***

MICRO ADAPTIVITY IN VECTORWISE
SIGMOD 2013

BITMAP FILTERING EVALUATION





Source: Vivek Narasayya

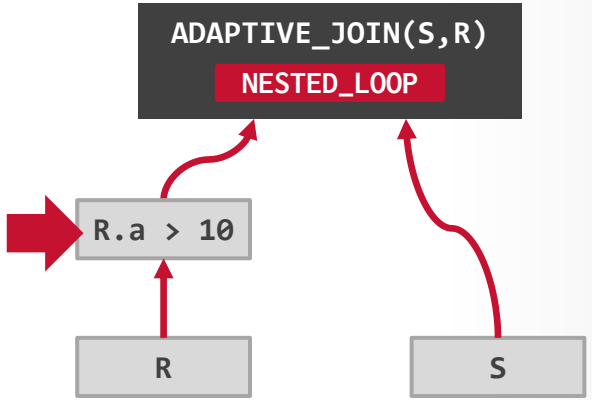
ADAPTIVE JOINS

Special operator that can switch between a hash join and nested loop join based on observed data during query execution.

→ User defines a row count threshold to specify when the DBMS switches from one join type to another.

Only makes query plan resilient to estimation errors on the outer child of a join operator.

```
SELECT *
  FROM R JOIN S
    ON R.id = S.id
 WHERE R.a > 10;
```



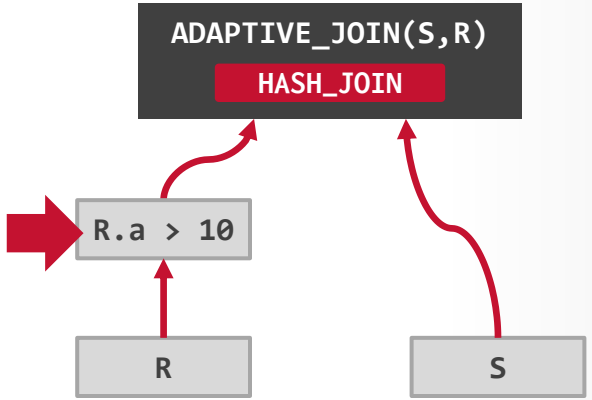
ADAPTIVE JOINS

Special operator that can switch between a hash join and nested loop join based on observed data during query execution.

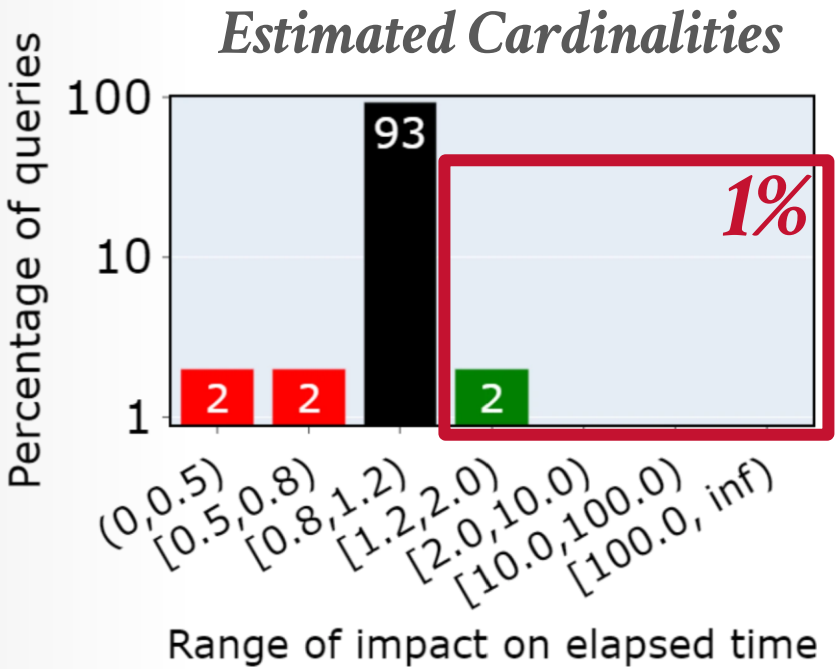
→ User defines a row count threshold to specify when the DBMS switches from one join type to another.

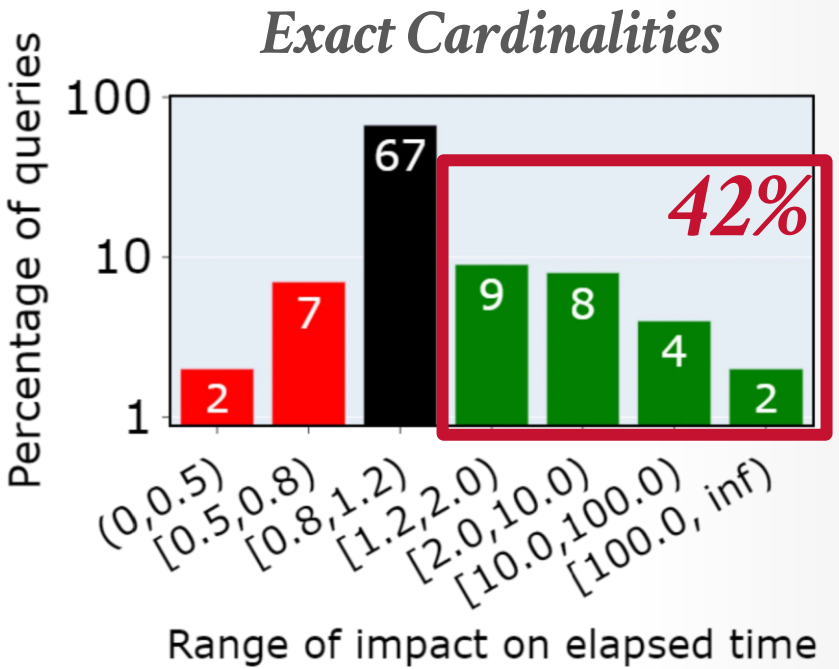
Only makes query plan resilient to estimation errors on the outer child of a join operator.

```
SELECT *
  FROM R JOIN S
    ON R.id = S.id
 WHERE R.a > 10;
```



ADAPTIVE JOINS EVALUATION





Source: Vivek Narasayya

PARTING THOUGHTS

Accurate cardinality estimation helps almost as many queries in a columnstore as in a rowstore.

A small fraction of queries will have performance regression when using exact cardinality estimates.

Only 1-5% of logical subexpressions in a query need accurate estimates to generate the best plan.

NEXT CLASS

Applying ML to Cardinality Estimation