CS 764: Topics in Database Management Systems

Lecture 4: Query Optimization

Xiangyao Yu

9/16/2025

Today's Papers: Query Optimization

- Viktor Leis, et al., How Good Are Query Optimizers, Really?. VLDB, 2015
- Viktor Leis, et al., Still Asking: How Good Are Query Optimizers, Really?. VLDB 2025
- Patricia G. Selinger, et al., Access Path Selection in a Relational Database Management System. SIGMOD, 1979

Acknowledgment: Several slides borrowed or adapted from Viktor Leis

Outline

Query optimization

Join order benchmark (JOB)

Cardinality estimation

Cost model

Plan space enumeration

Test of time award paper

Query Optimizer Architecture

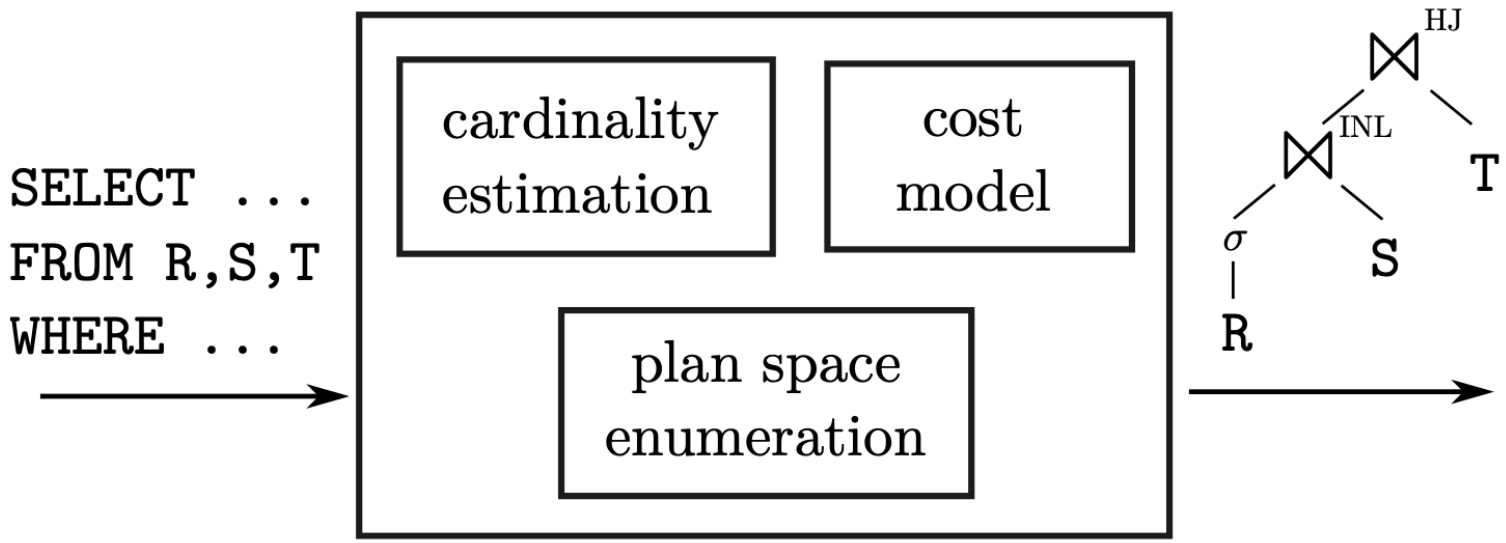


Figure 1: Traditional query optimizer architecture

Goal: Identify the fastest query plan for an input query

– Design complexity: access path, join order, join algorithm, etc.

Query Optimizer Architecture

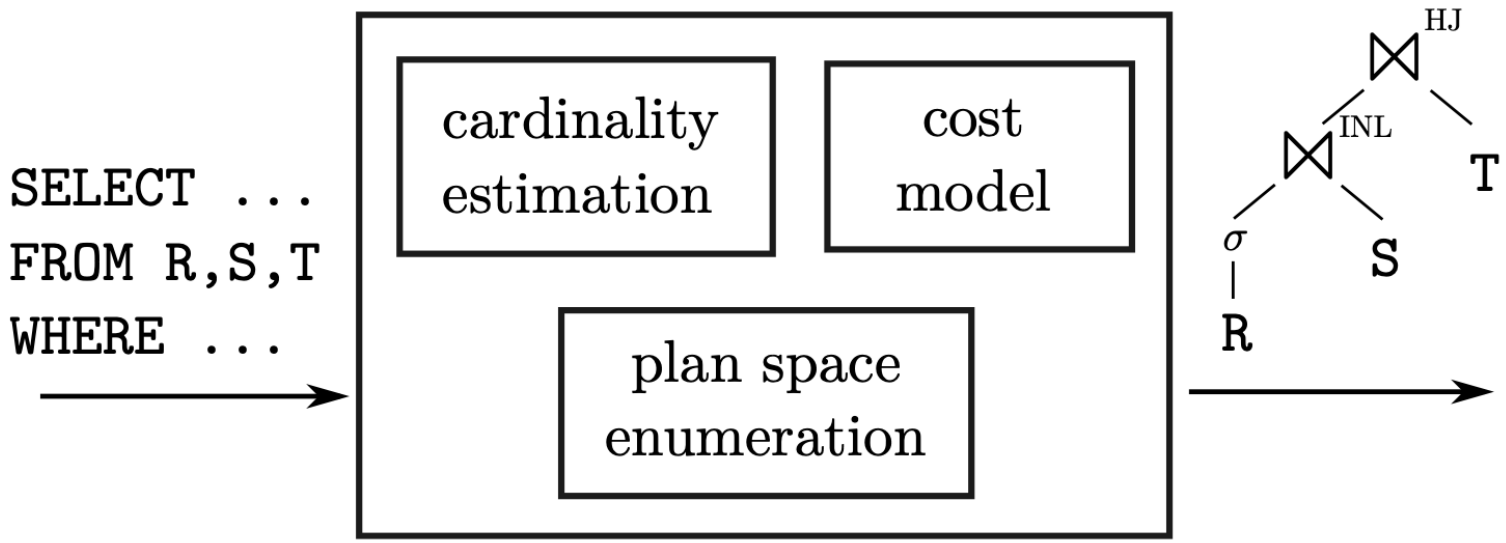


Figure 1: Traditional query optimizer architecture

Query cost = Σ(operator cost)

Operator cost = α * IO_cost(input_card)
+ β * CPU_cost(input_card, output_card)

Outline

Query optimization

Join order benchmark (JOB)

Cardinality estimation

Cost model

Plan space enumeration

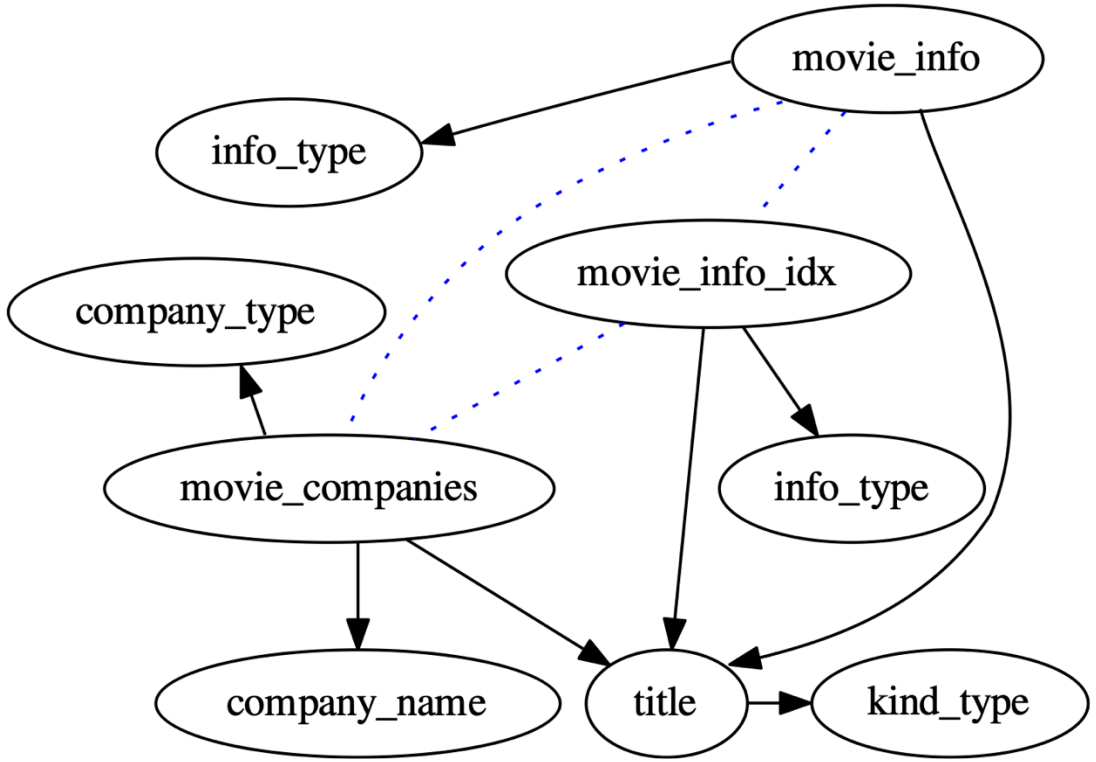
Test of time award paper

Join Order Benchmark: Data Set

- Internet Movie Data Base (IMDB) data set
- around 4GB, 21 relations
- information about movies and related facts about actors, directors, production companies, etc.
- publicly available for non-commercial use

Join Order Benchmark: Queries

```
SELECT cn.name, mi.info, miidx.info
FROM company_name cn, company_type ct,
     info_type it, info_type it2, title t,
     kind_type kt, movie_companies mc,
     movie_info mi, movie_info_idx miidx
WHERE cn.country_code ='[us]'
AND ct.kind = 'production companies'
AND it.info = 'rating'
AND it2.info = 'release dates'
AND kt.kind = 'movie'
AND ... -- (11 join predicates)
```



Outline

Query optimization

Join order benchmark (JOB)

Cardinality estimation

Cost model

Plan space enumeration

Test of time award paper

Cardinality Estimation in Postgres

Conjunctive predicates: assume independence and **multiply the selectivities of the individual selectivity estimates**.

Join output size estimation

$$|T_1 \bowtie_{x=y} T_2| = \frac{|T_1||T_2|}{\max(\text{dom}(x), \text{dom}(y))}$$

dom(x) is the domain cardinality of attribute x, i.e., the number of distinct values of x

Cardinality Estimation in System R

Calculate the selectivity factor F for each boolean factor/predicate

column = value

- If index exists — F = 1/ICARD(index) # distinct keys
- else — 1/10

column1 = column2

- 1 / Max(ICARD(column1 index), ICARD(column2 index))

column > value

- F = (high key value - value) / (high key value – low key value)

pred1 and pred2

- F = F(pred1) * F(pred2)

pred1 or pred2

- F = F(pred1) + F(pred2) – F(pred1) * F(pred2)

Not pred

- F = 1– F(pred)

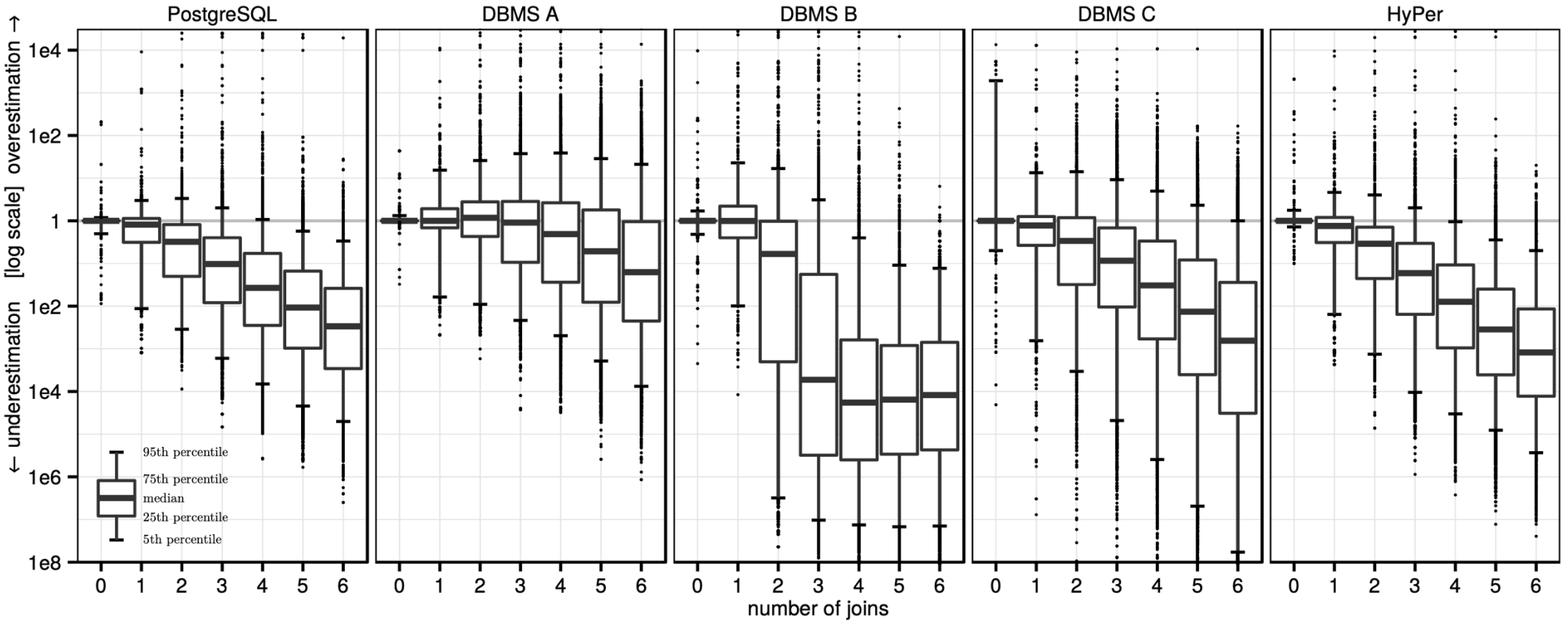
Cardinality Estimation for Base Selection

	median	90th	95th	max
PostgreSQL	1.00	2.08	6.10	207
DBMS A	1.01	1.33	1.98	43.4
DBMS B	1.00	6.03	30.2	104000
DBMS C	1.06	1677	5367	20471
HyPer	1.02	4.47	8.00	2084

Table 1: Q-errors for base table selections

These estimates are relatively accurate

Cardinality Estimation for Joins



- Systematically underestimate
- Several orders-of-magnitude estimation error

Effect of Estimates on Query Performance (1)

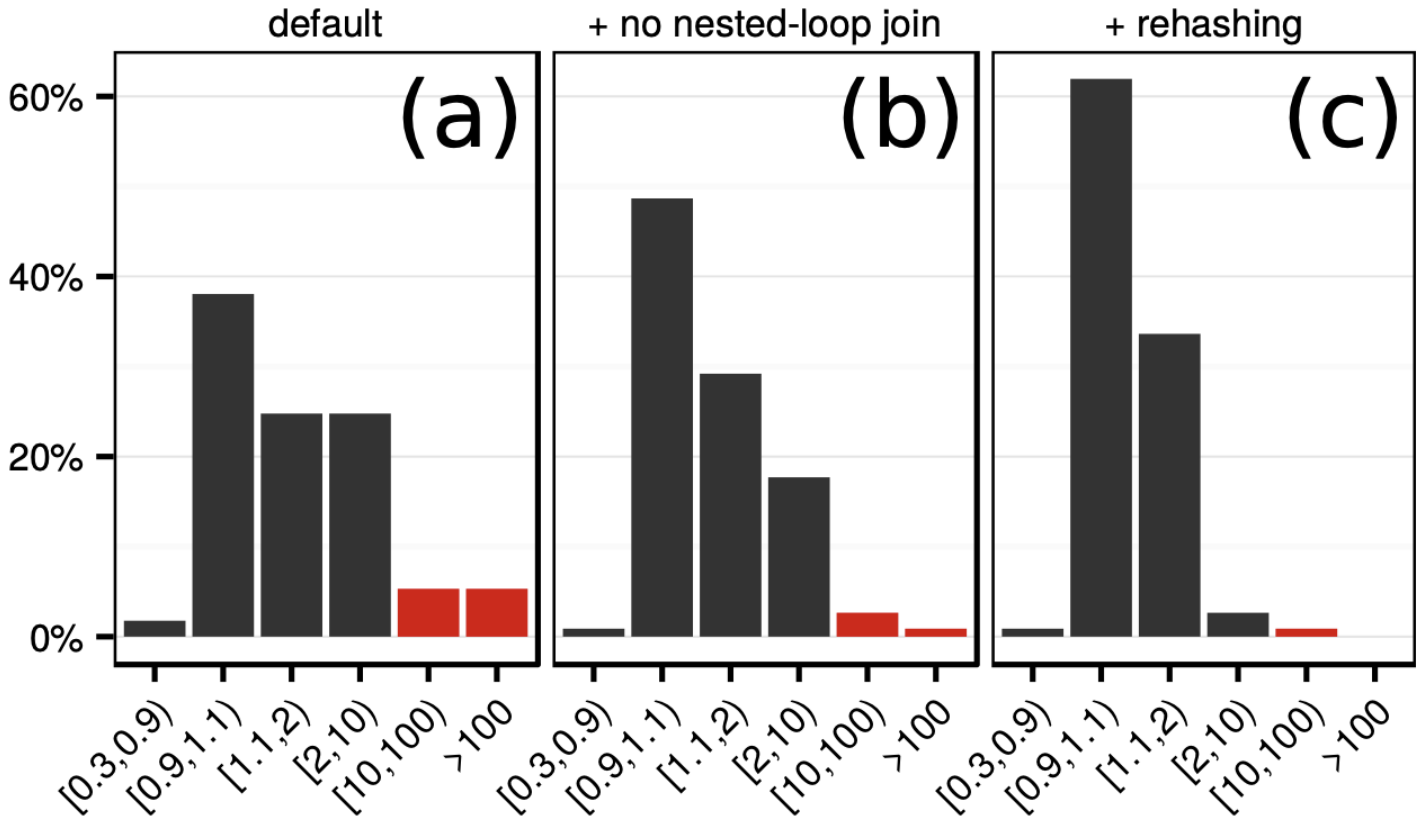


Figure 6: Slowdown of queries using PostgreSQL estimates w.r.t. using true cardinalities (primary key indexes only)

Removing **nested-loop join** and adding **runtime optimizations** can improve the accuracy of cardinality estimation

Effect of Estimates on Query Performance (2)

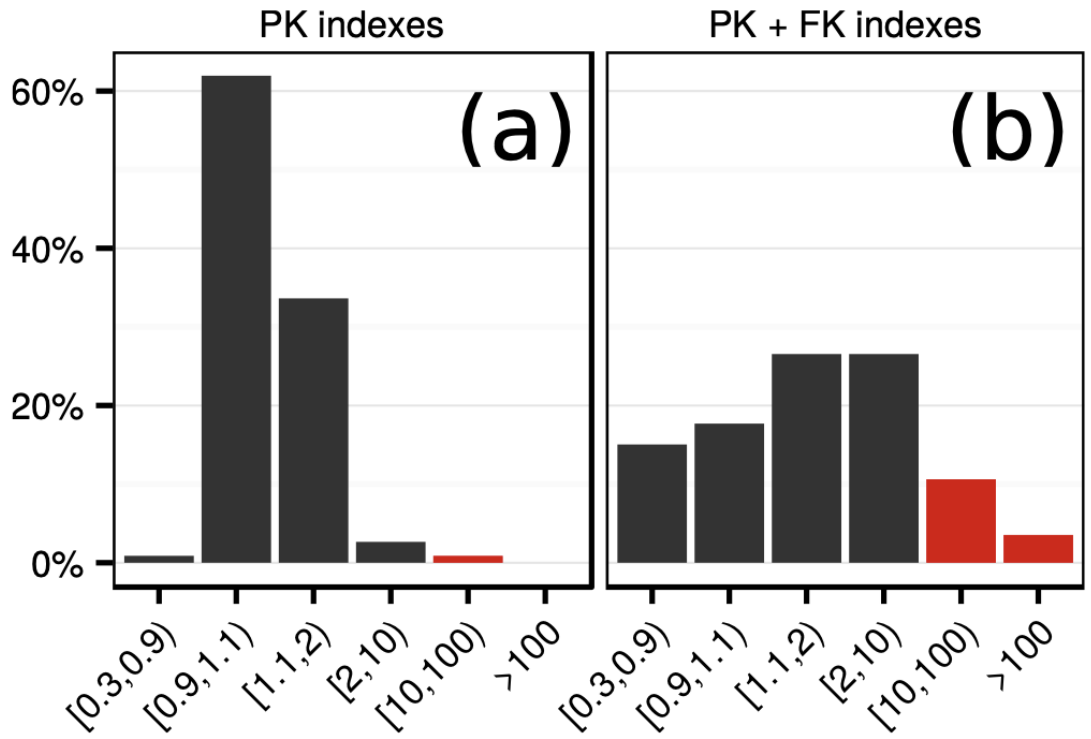


Figure 7: Slowdown of queries using PostgreSQL estimates w.r.t. using true cardinalities (different index configurations)

More indexes further reduce the cardinality estimation accuracy

Outline

Query optimization

Join order benchmark (JOB)

Cardinality estimation

Cost model

Plan space enumeration

Test of time award paper

Cost Model in System R

Cost = IO cost + Computation cost

= **#I/Os** + W * **RSICARD** (output cardinality)

IO Cost in System R

Calculate the number of pages access through IO

segment scan

- IO = TCARD(T)/P # segment pages

unique index matching (e.g., EMP.ID = '123')

- IO = 1 data page + 1-3 index page

clustered index matching

- IO = F(preds) * (NINDEX(I) + TCARD(T)) # index pages & # data pages

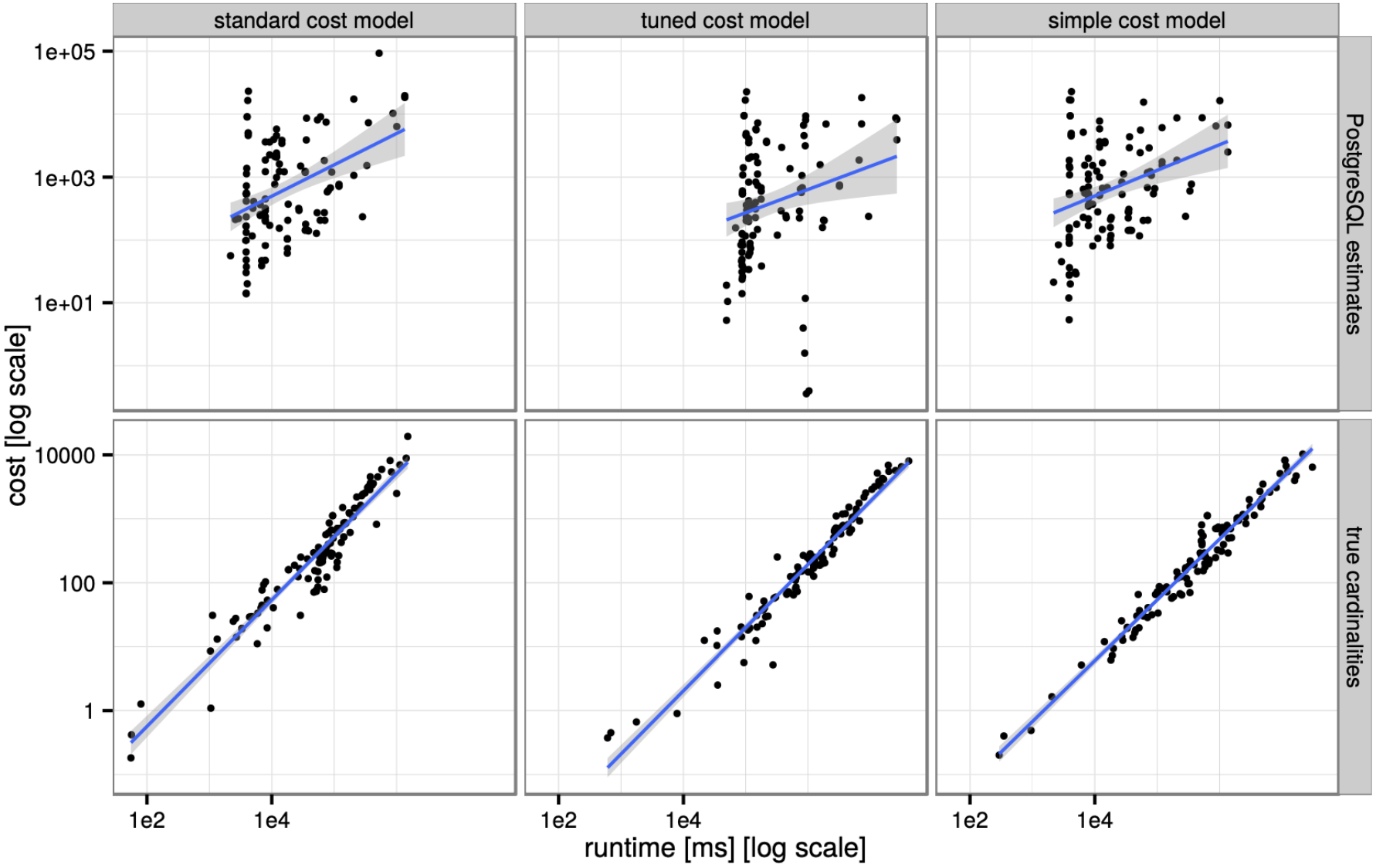
non-clustered index matching

- IO = F(preds) * (NINDEX(I) + NCARD(T)) # index pages & # data page accesses

clustered index no matching

- IO = NINDEX(I) + TCARD(T)

Cost Model



Cardinality estimation is much more crucial than the cost model.

Outline

Query optimization

Join order benchmark (JOB)

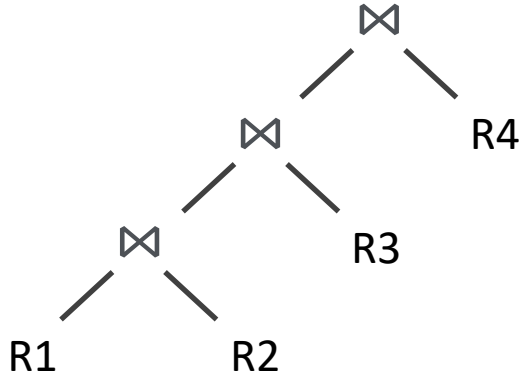
Cardinality estimation

Cost model

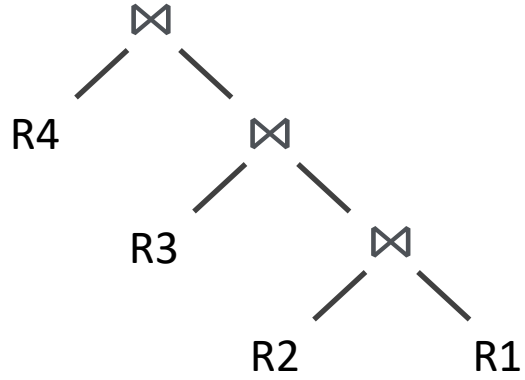
Plan space enumeration

Test of time award paper

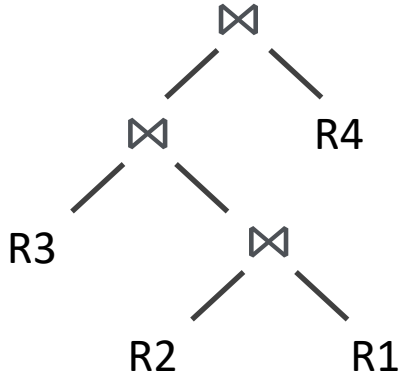
Plan Space – Join Order



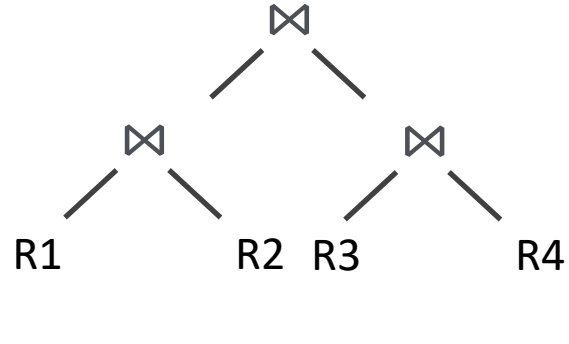
left-deep tree



right-deep tree

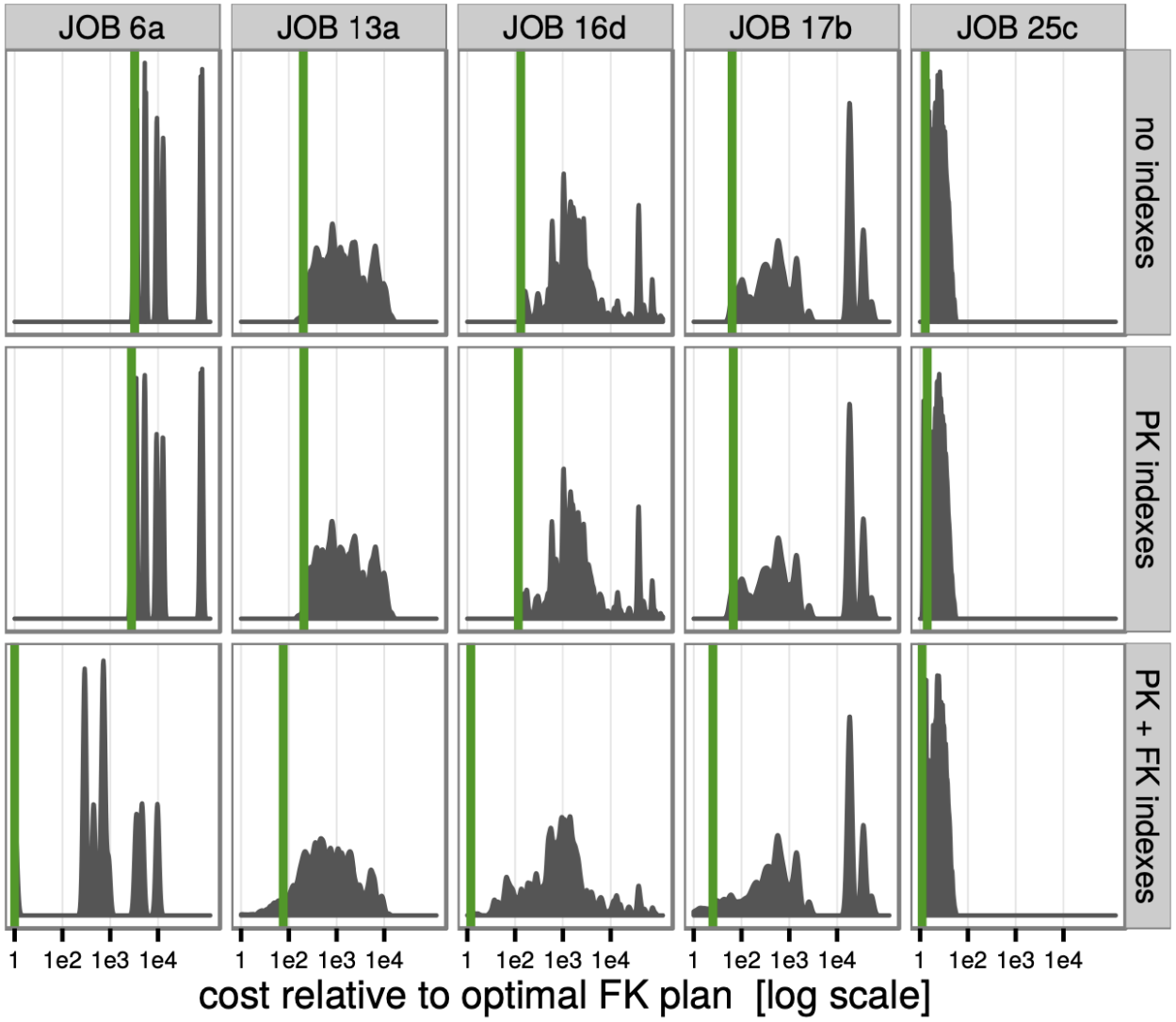


zig-zag tree



bushy tree

Cost Distribution (True Cardinalities)



Plan space enumeration is still important

Heuristics vs. Exhaustive Enumeration

PK indexes:

	PostgreSQL estimates			true cardinalities		
	median	95%	max	median	95%	max
DP[1]	1.03	1.85	4.79	1.00	1.00	1.00
QP-1000[2]	1.05	2.19	7.29	1.00	1.07	1.14
GOO[3]	1.19	2.29	2.36	1.19	1.64	1.97

PK + FK indexes:

	PostgreSQL estimates			true cardinalities		
	median	95%	max	median	95%	max
DP	1.66	169	186367	1.00	1.00	1.00
QP-1000	2.52	365	186367	1.02	4.72	32.3
GOO	2.35	169	186367	1.20	5.77	21.0

[1] optimal plan
[2] best of 1000 random plans
[3] Greedy Operator Ordering

Cardinality estimation is more crucial than enumeration strategies

Restricted Tree Shapes (true cardinalities)

	PK indexes			PK + FK indexes		
	median	95%	max	median	95%	max
zig-zag	1.00	1.06	1.33	1.00	1.60	2.54
left-deep	1.00	1.14	1.63	1.06	2.49	4.50
right-deep	1.87	4.97	6.80	47.2	30931	738349

Right deep is bad for JOB (but good for star-schema)

Bushy trees are not very necessary

Outline

Query optimization

Join order benchmark (JOB)

Cardinality estimation

Cost model

Plan space enumeration

Test of time award paper

2015-2025 Query Optimization Research

Learning-based approaches

- Learning cardinalities or better plans
- Learned approaches have not yet been widely adopted

Better benchmarks: SQLStorm benchmark

Runtime approaches

- Runtime feedback
- Limited runtime adaptivity

Future Directions

Yannakakis revival

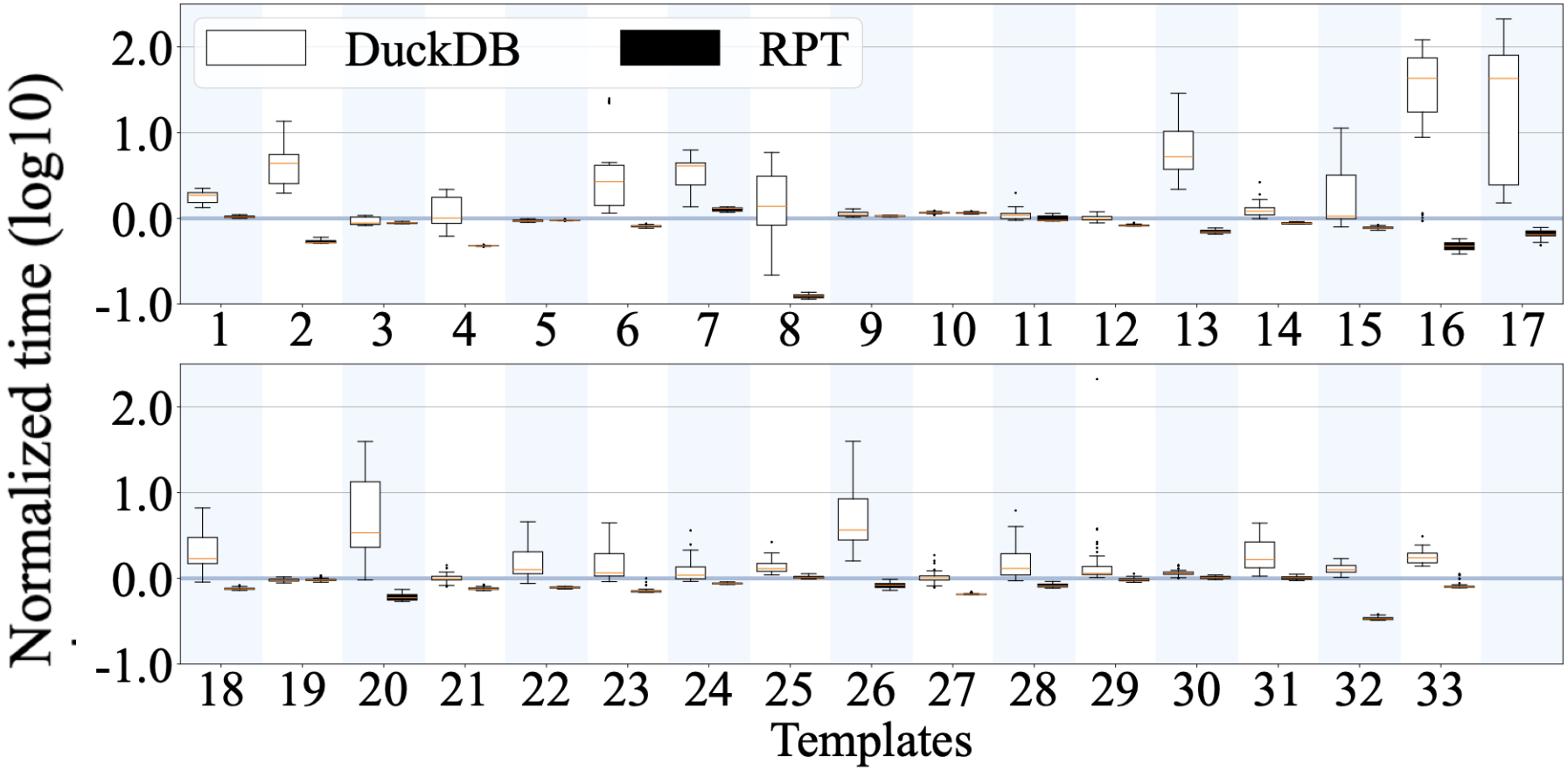
- Predicate transfer

Challenge #1: In data lake, meta data is usually not available

Challenge #2: Regressions can prevent innovation

- Even if a particular technique noticeably **improves the average performance** of a large workload, the presence of **regressions can prevent its adoption**

Predicate Transfer on JOB



No index used in this experiment

Join order does not really matter after applying predicate transfer

Questions

- Will learned cardinality estimators change conclusions?
  - High training and inference costs
  - Difficulty adapting to dynamic environments
  - Challenges in obtaining high-quality training data
  - Unpredictability due to black-box nature
- Pivot more toward runtime adaptation rather static optimization?
- What about distributed databases?

Project Ideas—Query Optimization

- Rerun some experiments in the paper with PT enabled
- Evaluate PT on SQLStorm benchmark

Before Next Lecture

Submit review for

Mike Stonebraker, et al. C-store: a column-oriented DBMS, VLDB 2005