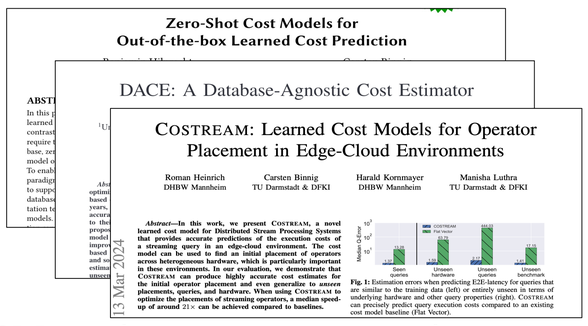
How Good Are Learned Cost Models, Really? Insights from Query Optimzization Tasks

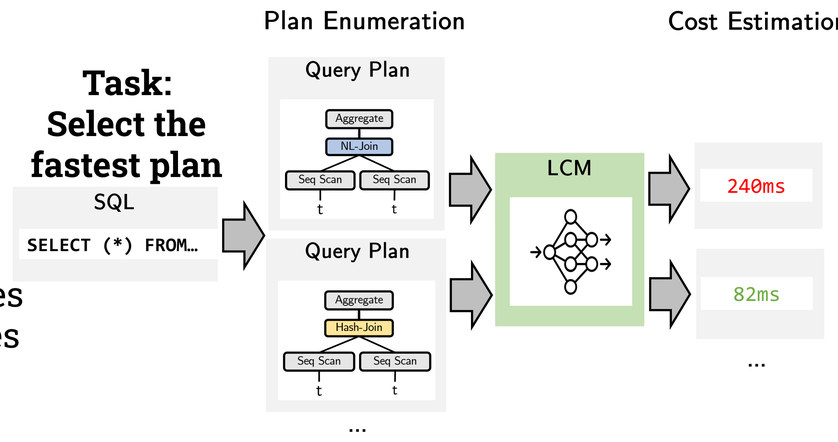
Roman Heinrich, Manisha Luthra, Johannes Wehrstein, Harald Kornmayer, Carsten Binnig
TU Darmstadt, DFKI & DHBW Mannheim

The Rise of Learned Cost Models



- LCMs estimate query execution costs in databases
- LCMs were proposed to overcome the weaknesses of traditional approaches
- LCMS learn from previous query executions
- **LCMs are more precise than traditional approaches!**

LCMs in Query Optimization



Do LCMs provide better plan selections?

A Novel Study

Model	Query Representation	Database-Dependency	Model Architecture
Flat Vector [10]	Flat	DB-agnostic	Regression Tree
MSCN [18]	Flat	DB-specific	Deep Sets
End-To-End [31]	Graph	DB-specific	Tree Structured NN
QPP-Net [27]	Graph	DB-specific	Neural Units
QueryFormer [44]	Graph	DB-specific	Transformer
Zero-Shot [13]	Graph	DB-agnostic	Graph Neural Networks
DACE [23]	Graph	DB-agnostic	Transformer

- Evaluating 7 state-of-the art Learned Cost Models against PostgreSQLs cost model
- Broad variety of LCMs covering various learning paradigms and featurizations
- LCMs were trained on up to 200.000 SPAJ Queries from 20 different databases

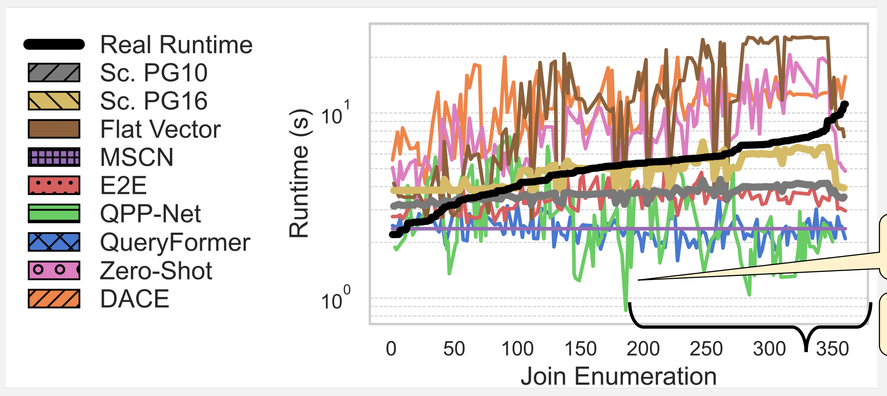
Task 1: Join Ordering

Which order of joins is optimal for a given query?

Requirement for Cost Models: Rank between different join orders

Method: Exhaustively iterate over all join orders of JOB-Light queries. Let cost models predict and analyze their predictions and plan selections.

Example: Predict Runtime for all possible Join Orders of JOB-Light Query Nr 33



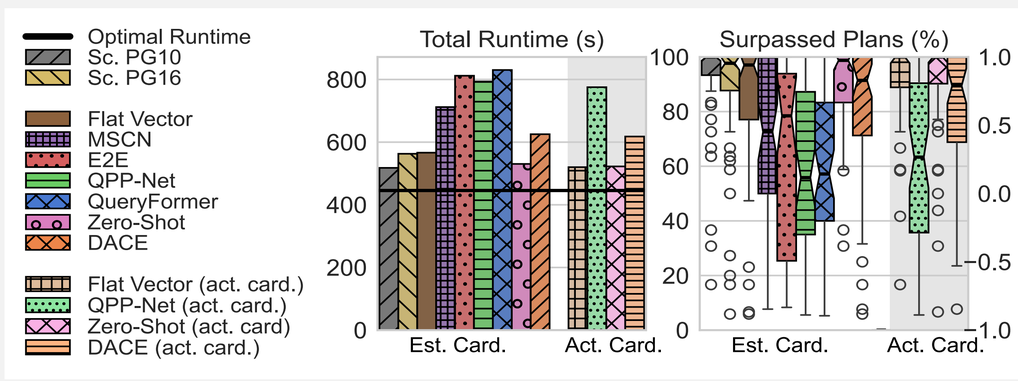
Most LCMS do not understand the costs of the join order!

Novel Metrics

Selected Runtime (s)
How fast is the selected plan?

Surpassed Plans (%)
How optimal is the selected plan?

Results over all JOB-Light Queries



Traditional Models are outperforming LCMs for join ordering … even if LCMs have access to actual cardinalities

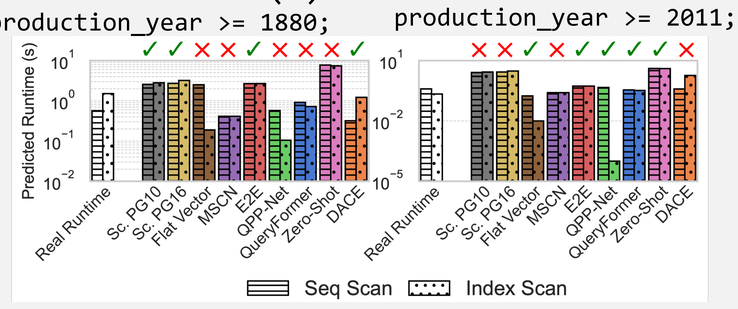
Task 2: Access Path Selection

How to optimally access a given table?

Requirement for Cost Models: Decide between Sequential Access and Index Look-Up

Method: Compare predictions for IndexScan and Sequential Scan. Let cost models predict and analyze their predictions and plan selections.

Example: Find Optimal Access Path for `column title.production_year`

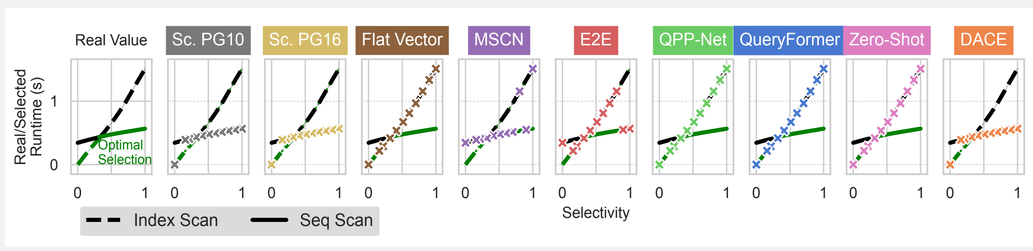


Most cost models do not select the correct access path!

Novel Metrics

Balanced Accuracy
How accurate are the selections?

Results Over Whole Selectivity Range



Traditional Models are outperforming LCMs for access path selection. They prefer IndexScans too often.

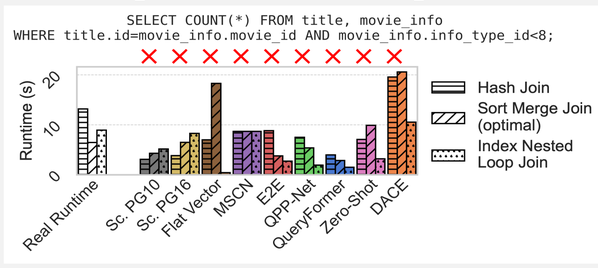
Task 3: Join Operator Selection

What is the optimal join operator implementation?

Requirement for Cost Models: Select between join algorithms (Hash, Sort, NestedLoop)

Method: Compare predictions different join implementations. Let cost models predict and analyze their predictions and plan selections.

Example: Select Join Operator for a Two-Way Join Query

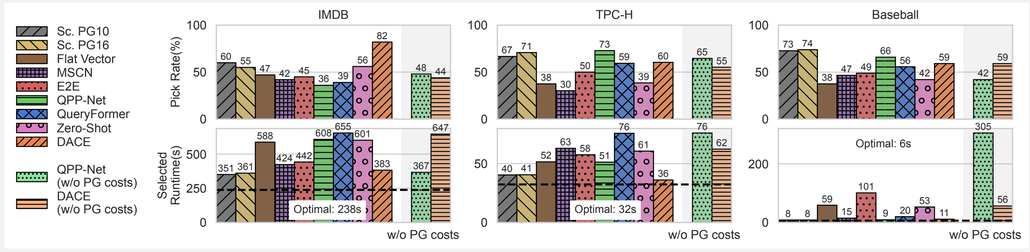


No LCM selects the optimal sort merge join for this query

Novel Metrics

Pick Rate
How accurate are the selections?

Full Results over Three Datasets



LCMs do less often pick the correct join operator than traditional approaches. This leads to longer execution runtimes.

What do we learn?

Don't Look Only at Estimation Accuracy - but at the Plan Selection!

- Evaluate and optimize your model against plan selection
- Use ranking metrics such as Selected Runtime, or the Accuracy over Access Path Selection

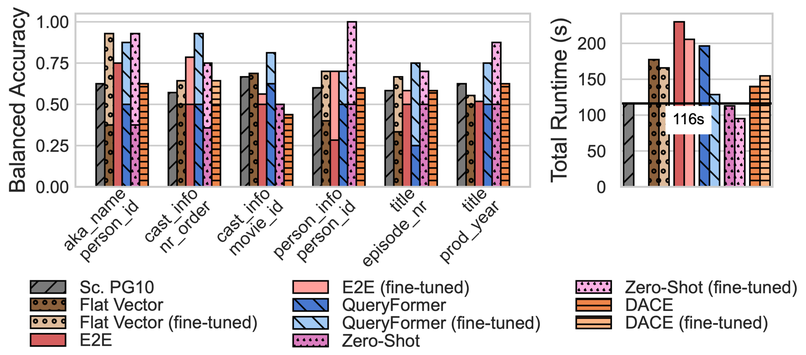
Recommendations for Model Design:

- Learning form Query Plans, not only from SQL
- Simple model architectures are often on par with complex models
- DB-agnostic (i.e. zero-shot/global) models achieved the best results
- Histograms and sample bitmaps do not show significant benefits
- Don't throw expert knowledge away – models using Postgres estimates performed better

How can we improve?

Overcoming the Training Data Bias

- Current strategies only learn from the plans provided and selected by PostgreSQL
- Future LCMs need to learn from sub-optimal plans
- Example: Fine-tuning for Access Path Selection



Paper

Code