南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

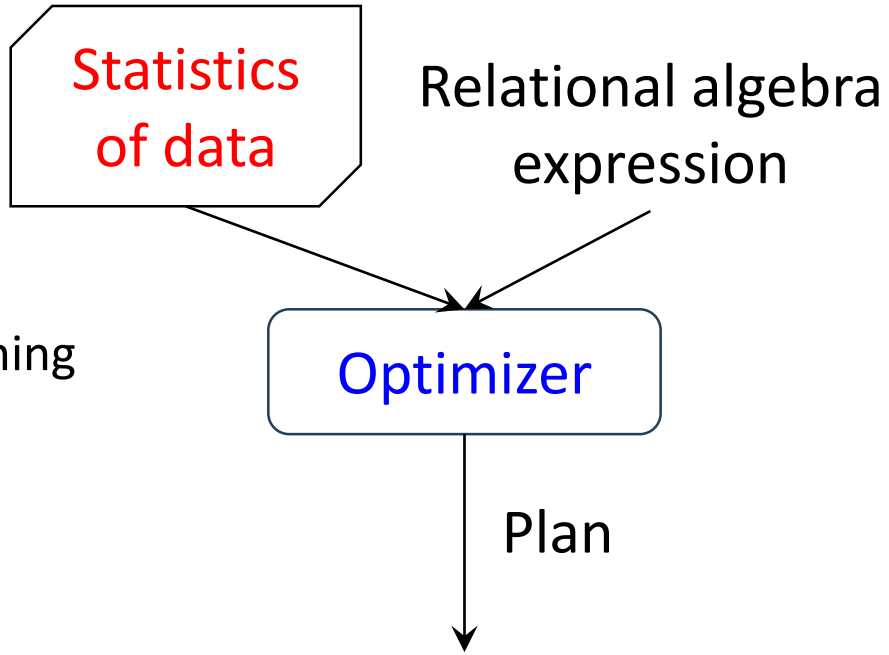
DBGroup

AI赋能的查询优化

南方科技大学
唐　博
tangb3@sustech.edu.cn

Recap: Why Query Optimizer?

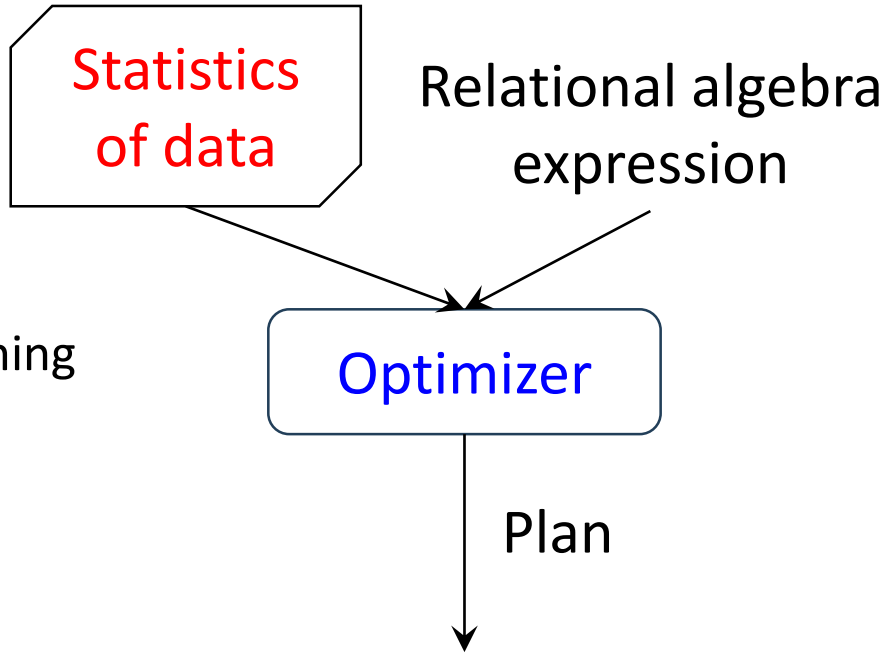
- **Goal**: Find the fastest plan.
- **Techniques**
  - Equivalence rules
  - Join ordering by dynamic programing
  - Cost and size estimation



Recap: Why Query Optimizer?

- **Goal**: Find the fastest plan.
- **Techniques**
  - Equivalence rules
  - Join ordering by dynamic programing
  - Cost and size estimation

Estimation always yields errors,
thus bad plan selection.



Recap: Estimate Selection Size

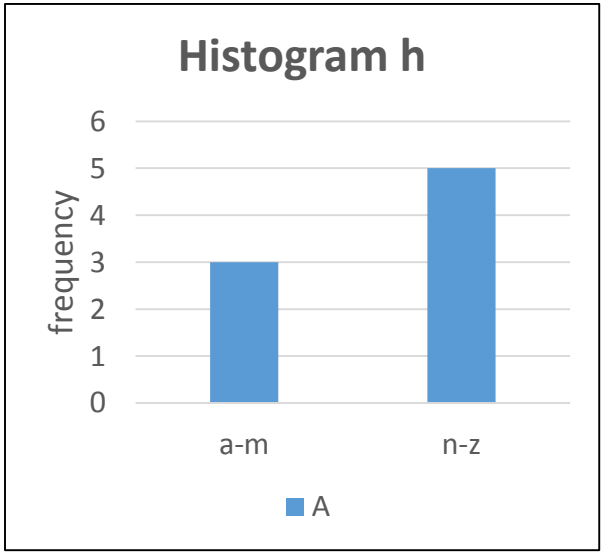
- **Problem**: Let *size* be the estimated number of tuples satisfying the condition, estimate the size of $\sigma_{A=v}(r)$ and A is not a unique attribute.
  - Estimated size = n_r / $V(r, A)$ = 8/4 = 2
  - Actual size = 1
- Estimate the size of $\sigma_{m \le A \le v}(r)$
  - If no histogram,
    - Estimated *size* = $\frac{v-m}{\max(r,A)-\min(r,A)} = \frac{22-13}{26-1} = 0.36$
    - Actual *size* = 2

Relation r

ID	A
1	a
2	a
3	m
4	v
5	z
6	z
7	z
8	z

Recap: Estimate Selection Size

- Estimate the size of $\sigma_{m \le A \le v}(r)$
  - If with histogram,
    - Estimated *size* = 3*(1/13)+5*(9/13) = 3.69
    - Actual *size* = 2



Relation r

ID	A
1	a
2	a
3	m
4	v
5	z
6	z
7	z
8	z

Recap: Estimate Join Size

Case of attributes	*Size of r ⋈ s*	*Remarks*
…	…	…
$R \cap S$ = {A} is not a key for r nor s	$\frac{n_r * n_s}{\max\{V(r,A), V(s,A)\}}$	The lower estimate, assume that every tuple t in r (or s) produces tuples in r ⋈ s

- **Problem:** Estimate the size of $r \bowtie_{r.A=s.A} s$
  - Estimated *size* = $\frac{4 * 4}{4}$ = 4
  - Actual *size* = 3

Relation r

ID	A
1	a
2	b
3	c
4	d

Relation s

ID	A
1	a
2	b
3	d
4	e

Recap: Estimate Join Size

- **Problem:** Generate join order for three relations *r*, *s*, *t*
  - $r(A) \bowtie_A s(A) \bowtie_A t(A)$, Each relation has 1000 tuples.
  - For simplicity, assume the cost of a join operation

    = the # of tuples of input intermediate relation(s).

	{r, s}	{r, t}	{s, t}
Estimated Size	1000	2000	500
Actual Size	100	1000	500
Estimated Cost	0	0	0
Estimated best plan	$r \bowtie s$	$r \bowtie t$	$s \bowtie t$
Actual best plan	$r \bowtie s$	$r \bowtie t$	$s \bowtie t$

	{r, s, t }
Estimated Size	10000
Actual Size	1000
Estimated Cost	500
Estimated best plan	$r \bowtie (s \bowtie t)$
Actual best plan	$(r \bowtie s) \bowtie t$

Recap: Summary

- **Cause of estimate error**
  - Skew distribution of practical data v.s. uniform distribution assumption (for selection).
  - Lack of correlation statistics between columns intra and inter-tables (for join).
- **Impact of estimate error**
  - worse plan selection → more time, more cost, worse user experience.

How to Improve Estimation?

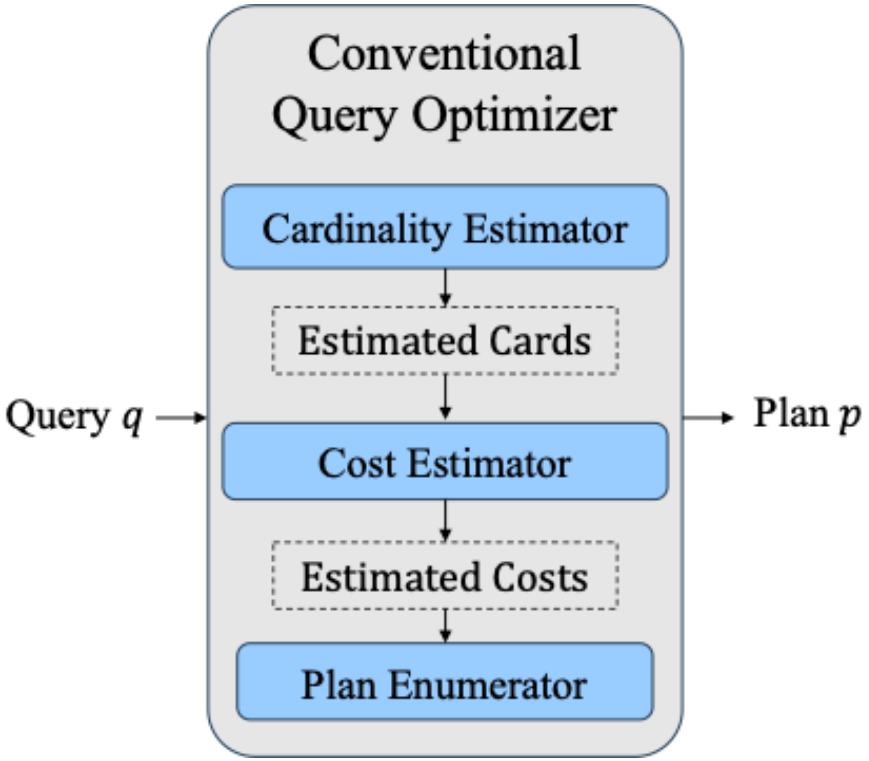
- Improve estimation is to represent data distribution better.
  - For statistics-based methods, better representation → more statistical data (could be more than data itself).
  - Fortunately, **AI does great in learning data distribution**.

AI-based Query Optimization

- Learned Query Optimization
  - Learned Estimator
  - Learned Optimizer
  - Learned Optimizer Enhancer
- Other AI4DB Topics

Learned Estimator

- Components of Query Optimizer
  - Cardinality Estimator: estimate operator (Selection/Join) size.
  - Cost Model: estimate plan cost.
  - Plan Enumerator: build plans via heuristic methods (e.g., Dynamic Programming).



Simplified architecture of query optimizer

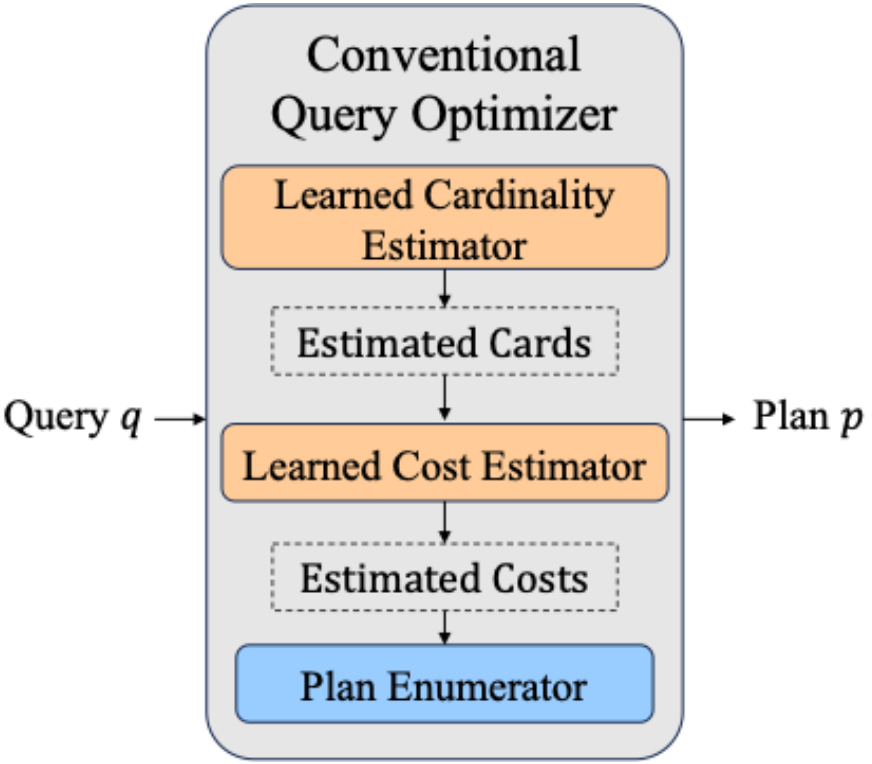
Learned Estimator

- Modules of Learned Estimator
  - Learned Cardinality Estimator:

    estimate operator (Selection/Join)

    size via a learning model.
  - Learned Cost Model: estimate plan

    cost via a learning model.
  - Plan Enumerator: no change.



Architecture of learned estimator

Learned Estimator

- Learned Cardinality Estimator
  - Query-driven: learn a mapping from query to cardinality from the historical workload.
  - Data-driven: learn the distribution directly from data.
  - Hybrid: combine both above methods.
- Learned Cost Model
  - Learn a mapping from plan to cost from the historical workload.

Learned Cardinality Estimator

- Query-driven (MSCN [1])

Step 1 - Query Featurization

SQL: SELECT COUNT(*) FROM title t, movie_companies mc WHERE t.id = mc.movie_id AND t.production_year > 2010 AND mc.company_id = 5;

Transform to three sets of feature vector

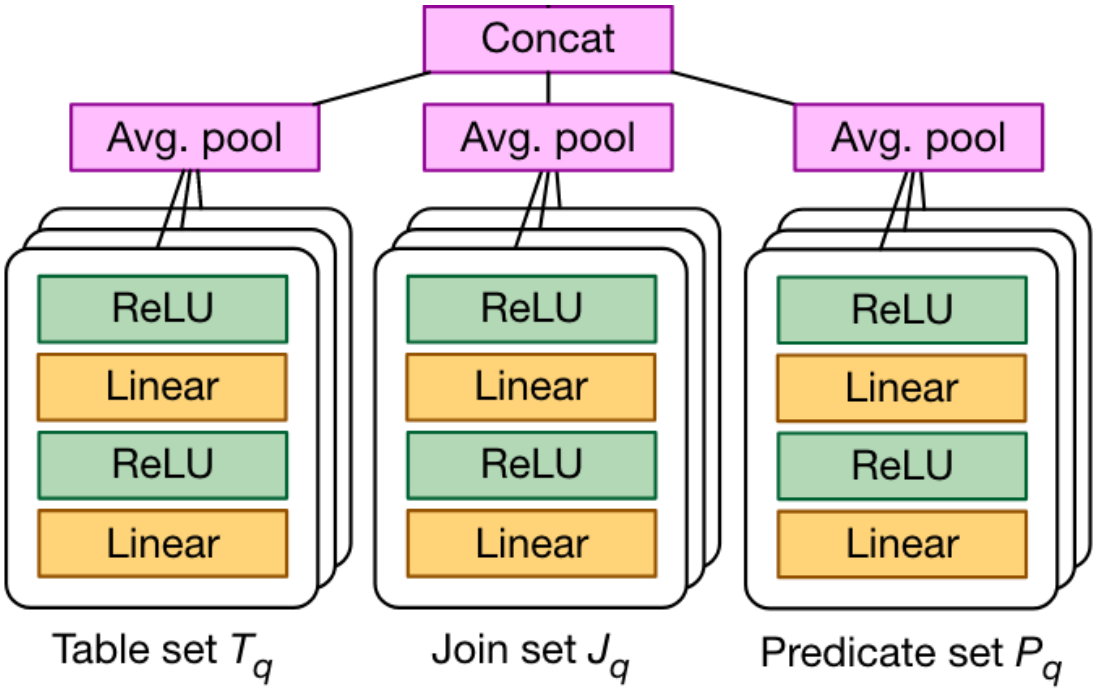
Table set	Join set	Predicate set
{[0 1 0 1 … 0], [0 0 1 0 … 1]}	{[0 0 1 0]}	{[1 0 0 0 0 1 0 0 0.72], [0 0 0 1 0 0 1 0 0.14]}
table id (one hot) / samples	join id (one hot)	column id (one hot) / value (normalized) / operator id (one hot)

[1] Andreas Kipf, et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Learned Cardinality Estimator

- Query-driven (MSCN [1])

Step 2 - Query Representation

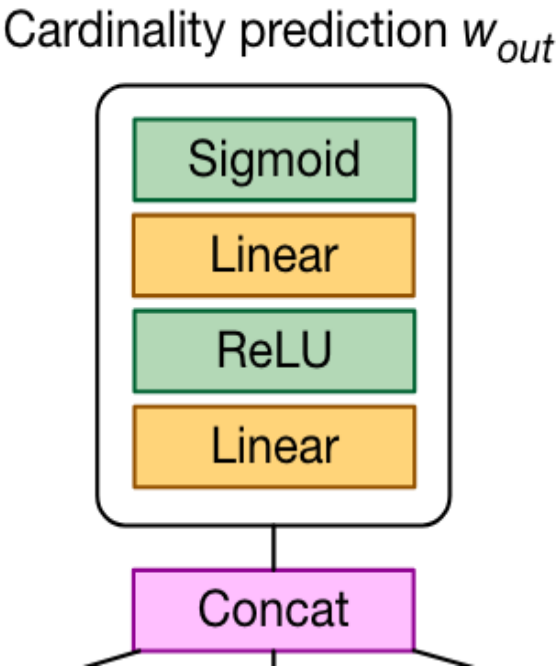


[1] Andreas Kipf, et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Learned Cardinality Estimator

- Query-driven (MSCN [1])

  Step 3 - Cardinality Prediction



[1] Andreas Kipf, et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Learned Cardinality Estimator

- Query-driven (MSCN [1])

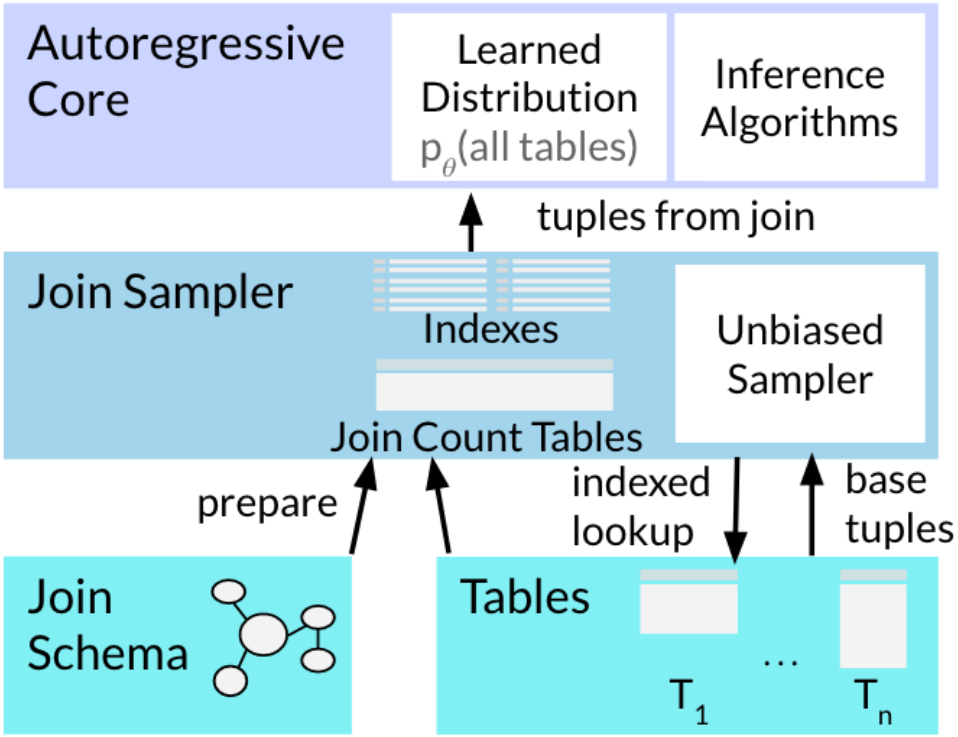
Estimation error on the JOB-light workload

	median	90th	95th	99th	max	mean
PostgreSQL	7.93	164	1104	2912	3477	174
MSCN	**3.82**	**78.4**	**362**	**927**	**1110**	**57.9**

[1] Andreas Kipf, et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. CIDR 2019.

Learned Cardinality Estimator

- Data-driven (NeuroCard [1])

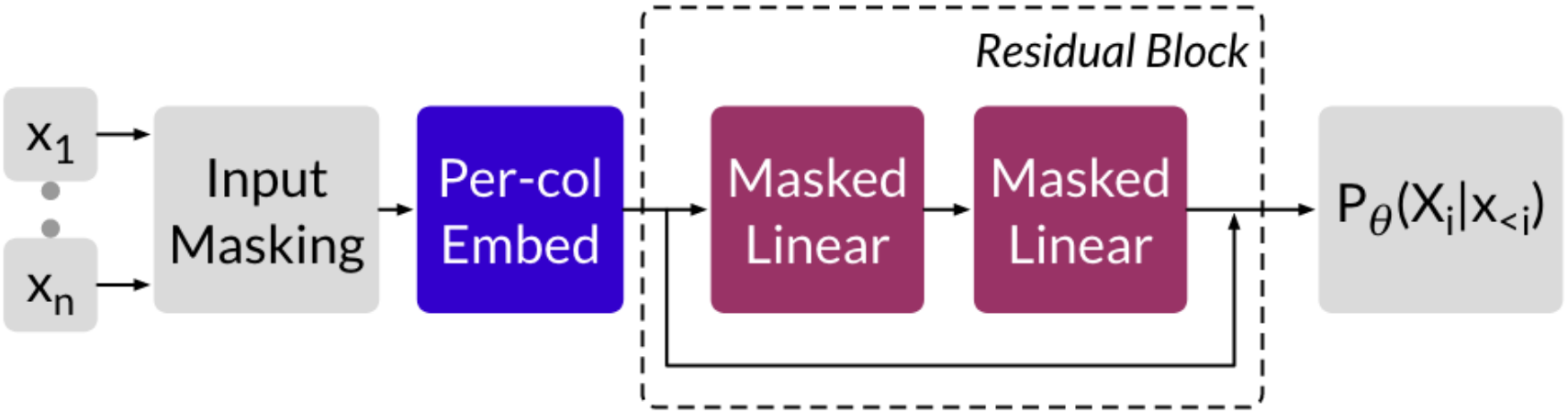


Overview of NeuroCard

[1] Zongheng Yang, et al. NeuroCard: One Cardinality Estimator for All Tables. VLDB 2021.

Learned Cardinality Estimator

- Data-driven (NeuroCard [1])



Default architecture of the autoregressive model

[1] Zongheng Yang, et al. NeuroCard: One Cardinality Estimator for All Tables. VLDB 2021.

Learned Cardinality Estimator

- Data-driven (NeuroCard [1])

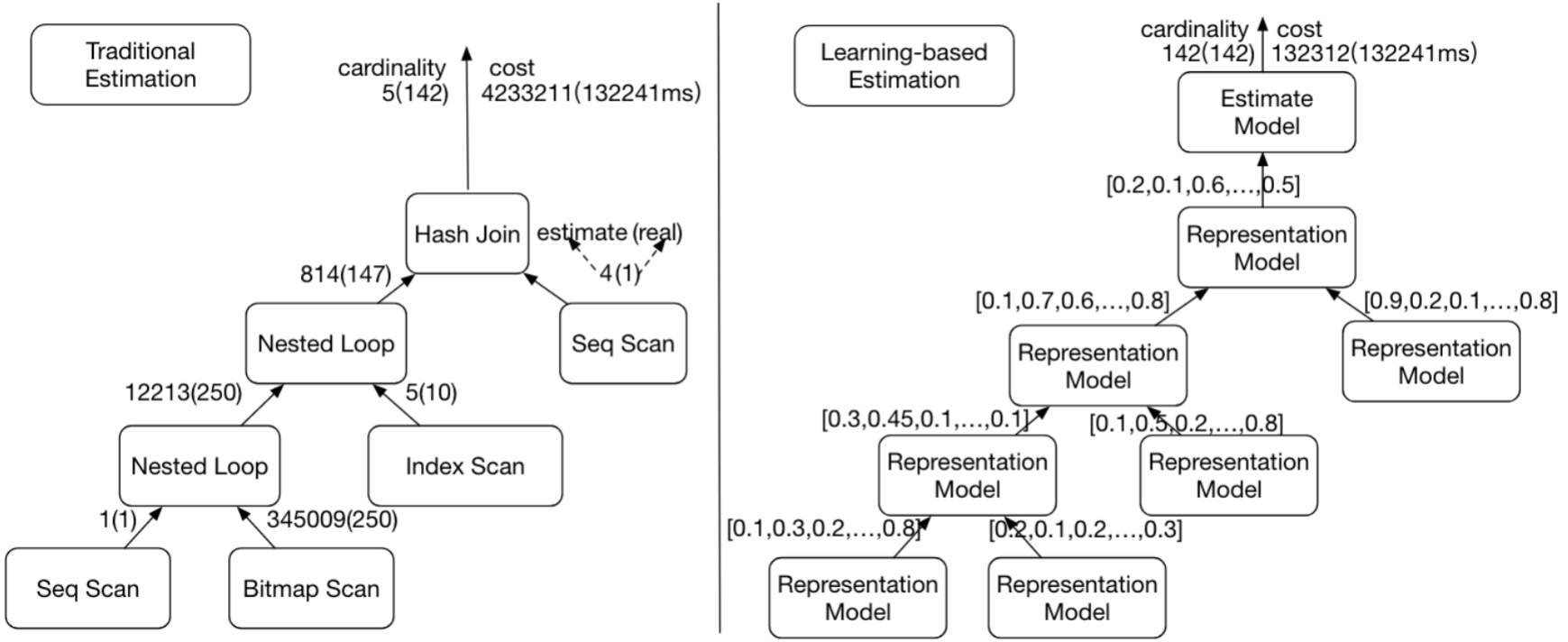
Estimation error on the JOB-light workload

	median	95th	99th	max
PostgreSQL	7.97	797	$3*10^3$	10^3
MSCN	3.01	136	$1*10^3$	10^3
NeuroCard	**1.57**	**5.91**	**8.48**	**8.51**

[1] Zongheng Yang, et al. NeuroCard: One Cardinality Estimator for All Tables. VLDB 2021.

Learned Cost Model

- T-LSTM[1]: both cardinality and cost are inaccurate

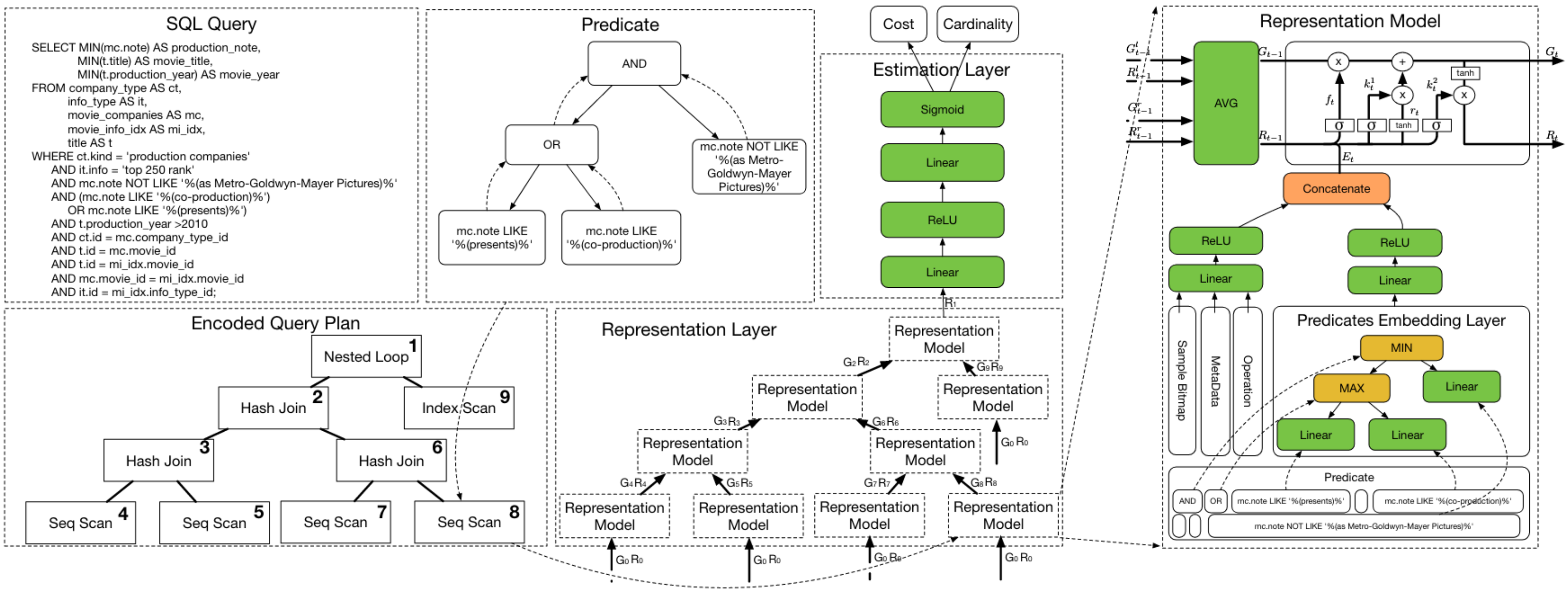


Comparison of traditional cost estimation and learning-based cost estimation

[1] Ji Sun, Guoliang Li. An End-to-End Learning-based Cost Estimator. VLDB 2019.

Learned Cost Model

- T-LSTM[1]: estimate both cardinality and cost



Two level tree model

[1] Ji Sun, Guoliang Li. An End-to-End Learning-based Cost Estimator. VLDB 2019.

Learned Estimator Analysis

- Learning historical workload or data for better estimation
- Pros:
  - Clear task definition and optimization procedure
- Cons:
  - Better estimations do not guarantee superior performance
  - Hard to integrate into DBMS
  - Significant overhead for multi-join queries
  - Insufficient robustness

Learned Optimizer

How to design a learned-based optimizer to improve the overall query processing performance?

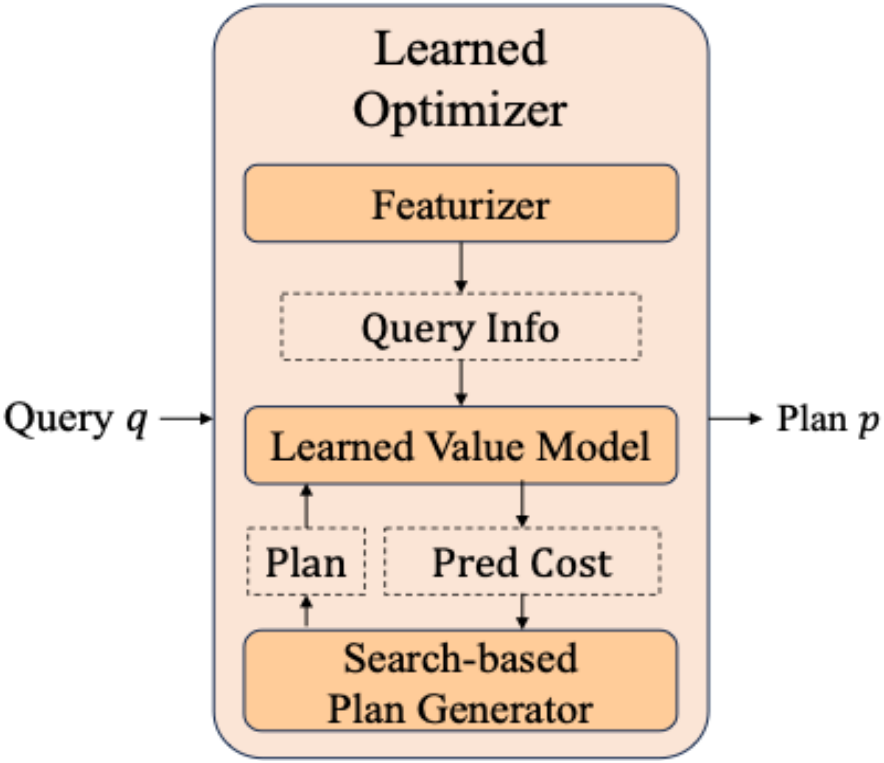
Learned Optimizer

How to design a learned-based optimizer to improve the overall query processing performance?

Design a reinforcement learning-based optimizer that learns from its plan construction decision!

Learned Optimizer

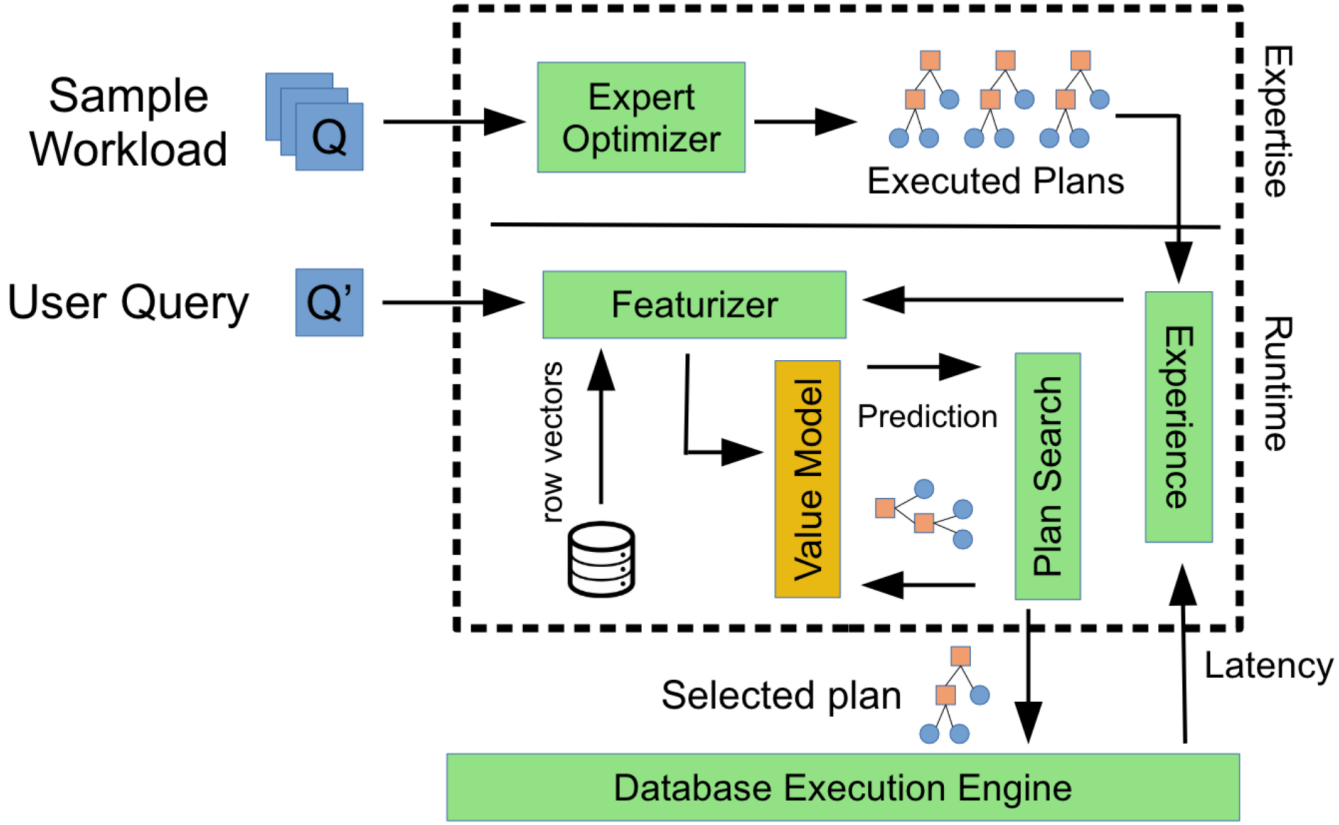
- Components of Learned Optimizer
  - Featurizer: extract features of query.
  - Learned Value Model: evaluate plan construction decision.
  - Search-based Plan Generator: efficient heuristic method to construct a plan (e.g., best-first search, beam search).



Architecture of learned optimizer

Learned Optimizer

- Neo [1]

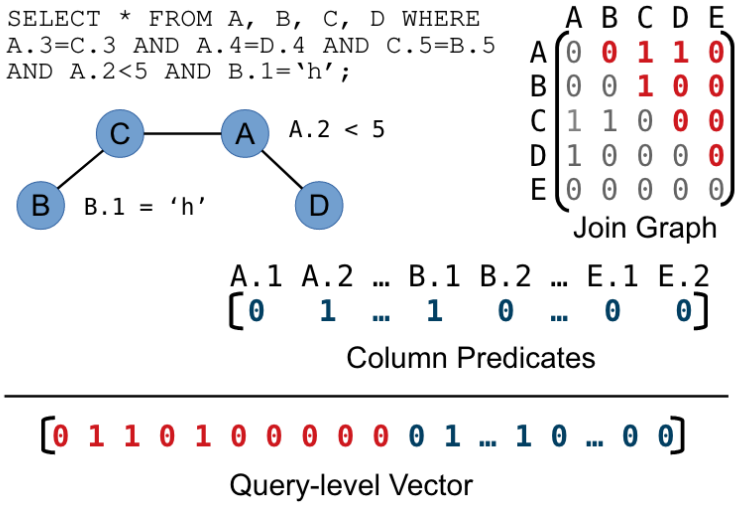


Architecture of Neo

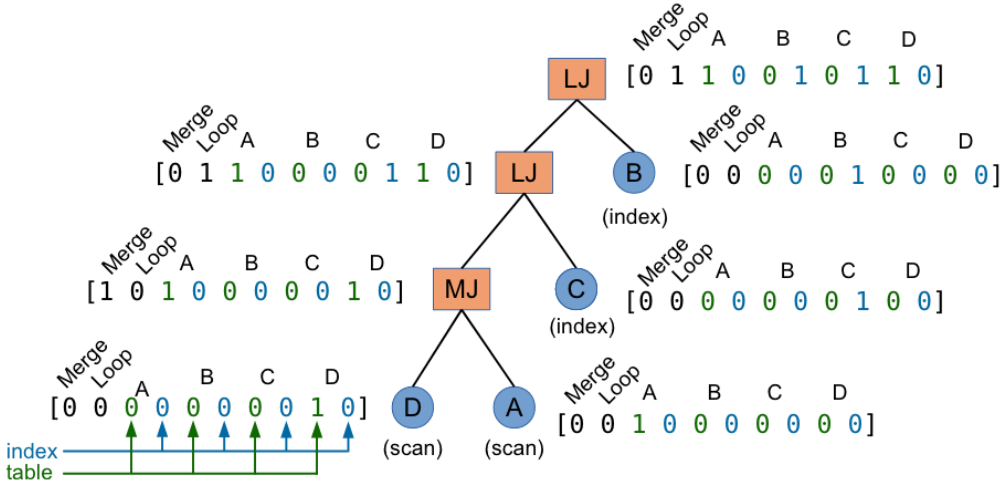
[1] Ryan Marcus, et al. Neo: A Learned Query Optimizer. VLDB 2019.

Learned Optimizer

- Neo [1] - Featurizer



Query-level encoding

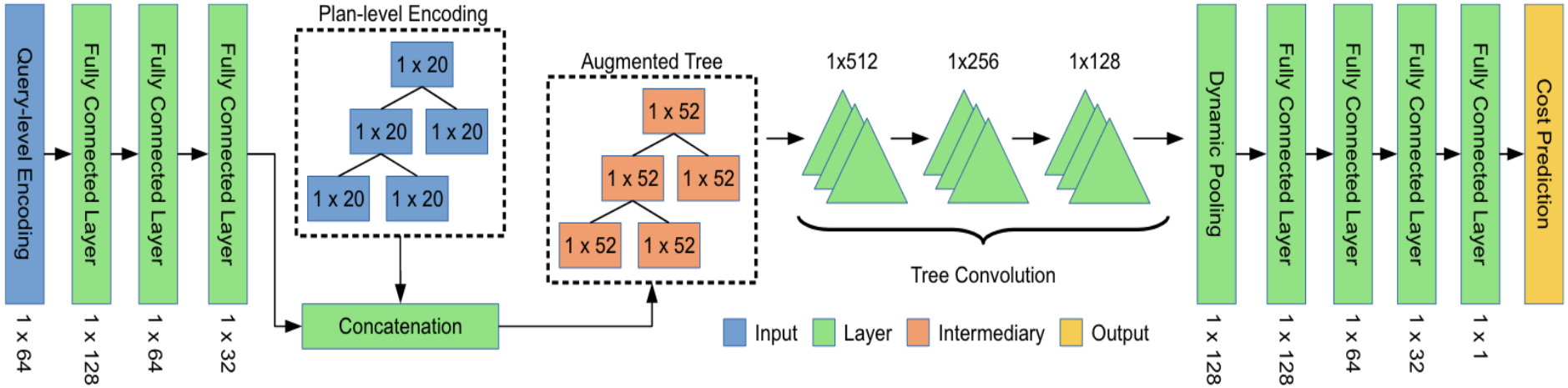


Plan-level encoding

[1] Ryan Marcus, et al. Neo: A Learned Query Optimizer. VLDB 2019.

Learned Optimizer

- Neo [1] - Value Network

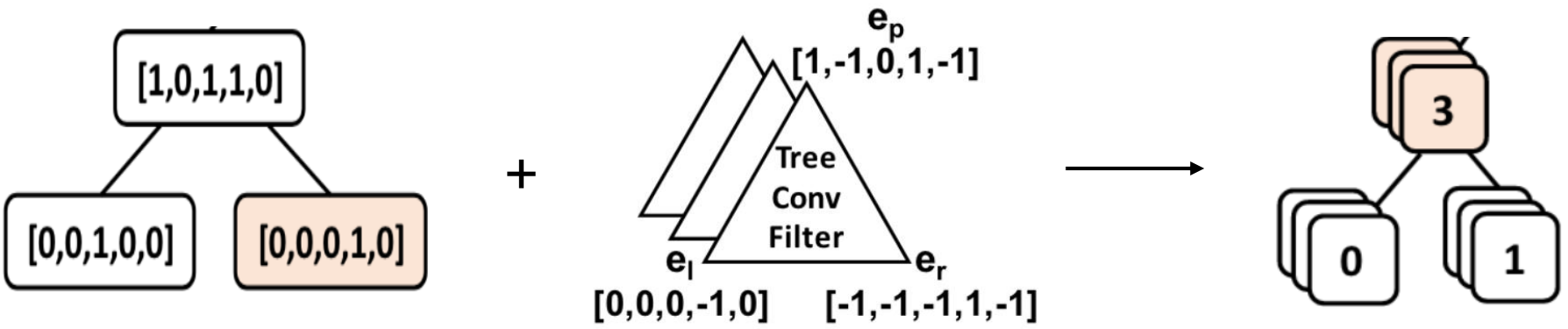


Architecture of Neo's value network

[1] Ryan Marcus, et al. Neo: A Learned Query Optimizer. VLDB 2019.

Learned Optimizer

- Neo [1] - Value Network - Tree Convolution Example



Features of nodes | Tree conv filters | Output

[1] Ryan Marcus, et al. Neo: A Learned Query Optimizer. VLDB 2019.

Learned Optimizer Analysis

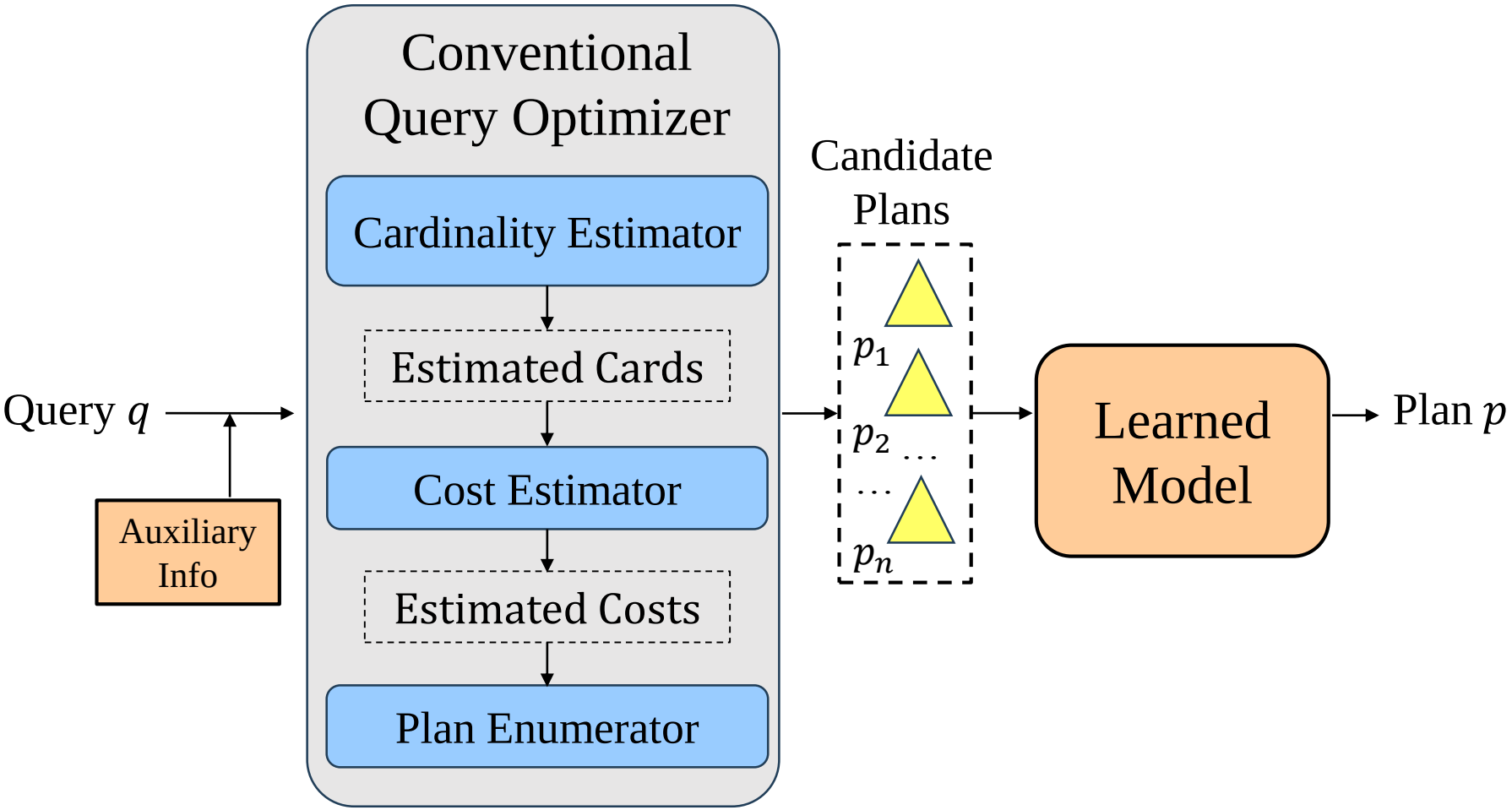
- Learn to construct plans via reinforcement learning
- Pros:
  - High potential for performance improvement
  - Available for database without a query optimizer
- Cons:
  - Substantial training overhead
  - Risk of worse plan generation (Insufficient robustness)

Learned Optimizer Enhancer

How to design a learned-based optimizer with

less training overhead and better robustness?

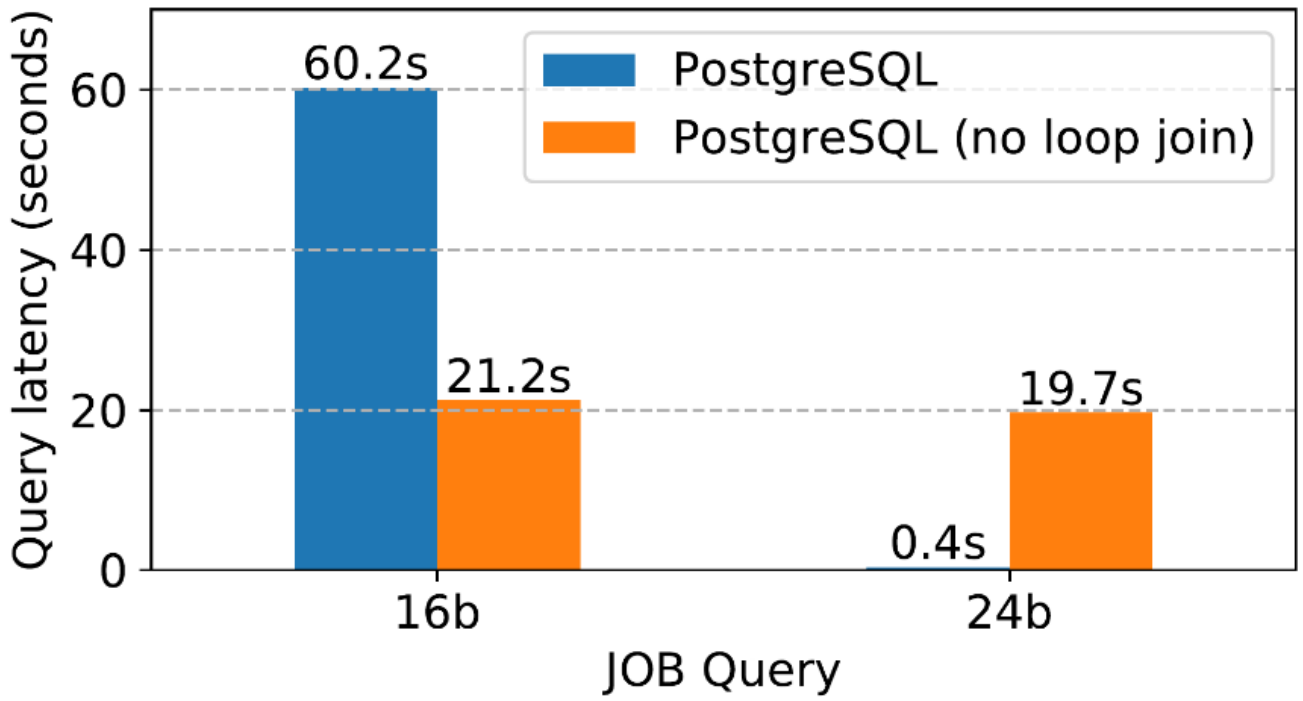
Learned Optimizer Enhancer



Architecture of learned optimizer enhancer

Learned Optimizer Enhancer

- Bao [1]

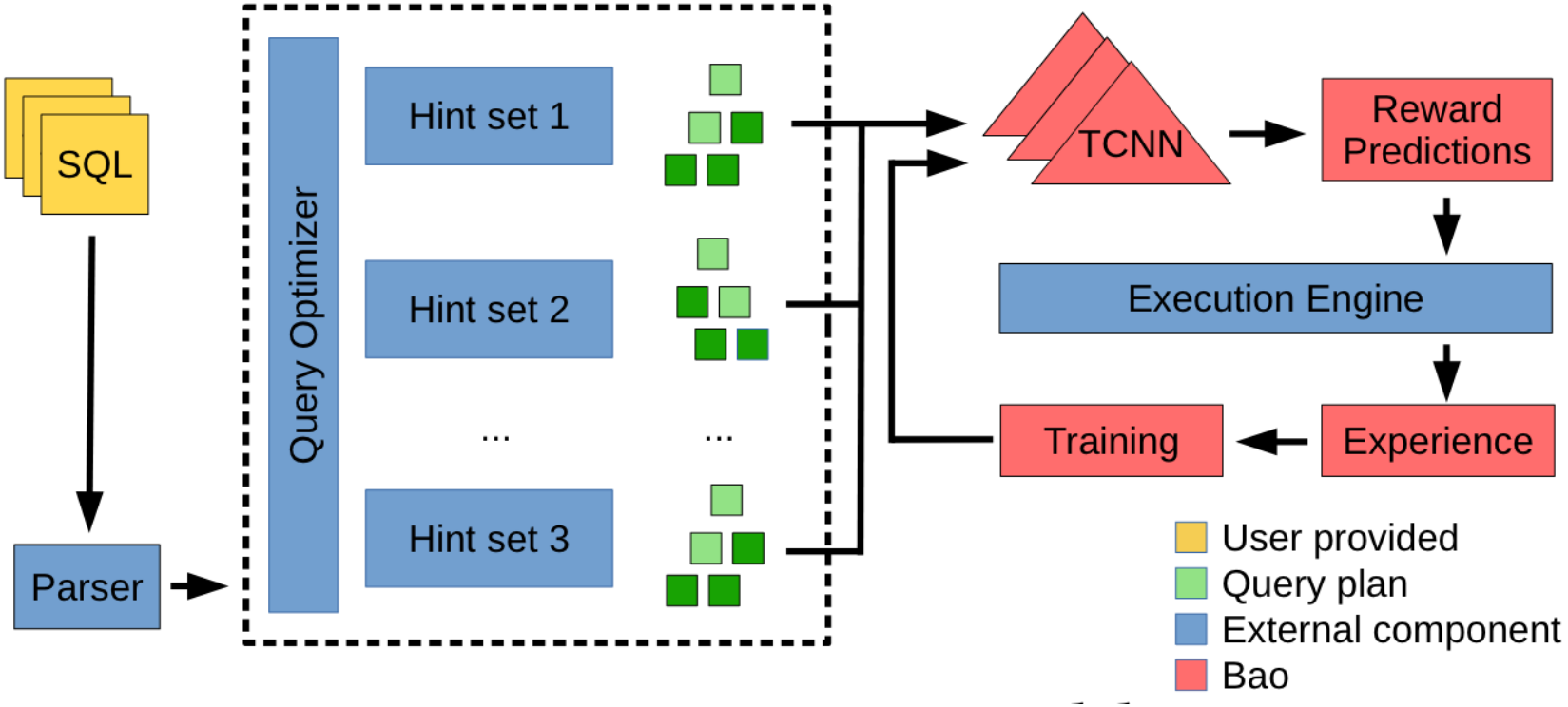


Disabling loop join improve (16b) but harm (24b) for JOB in PostgreSQL

[1] Ryan Marcus, et al. Bao: Making Learned Query Optimization Practical. SIGMOD 2021.

Learned Optimizer Enhancer

- Bao [1]

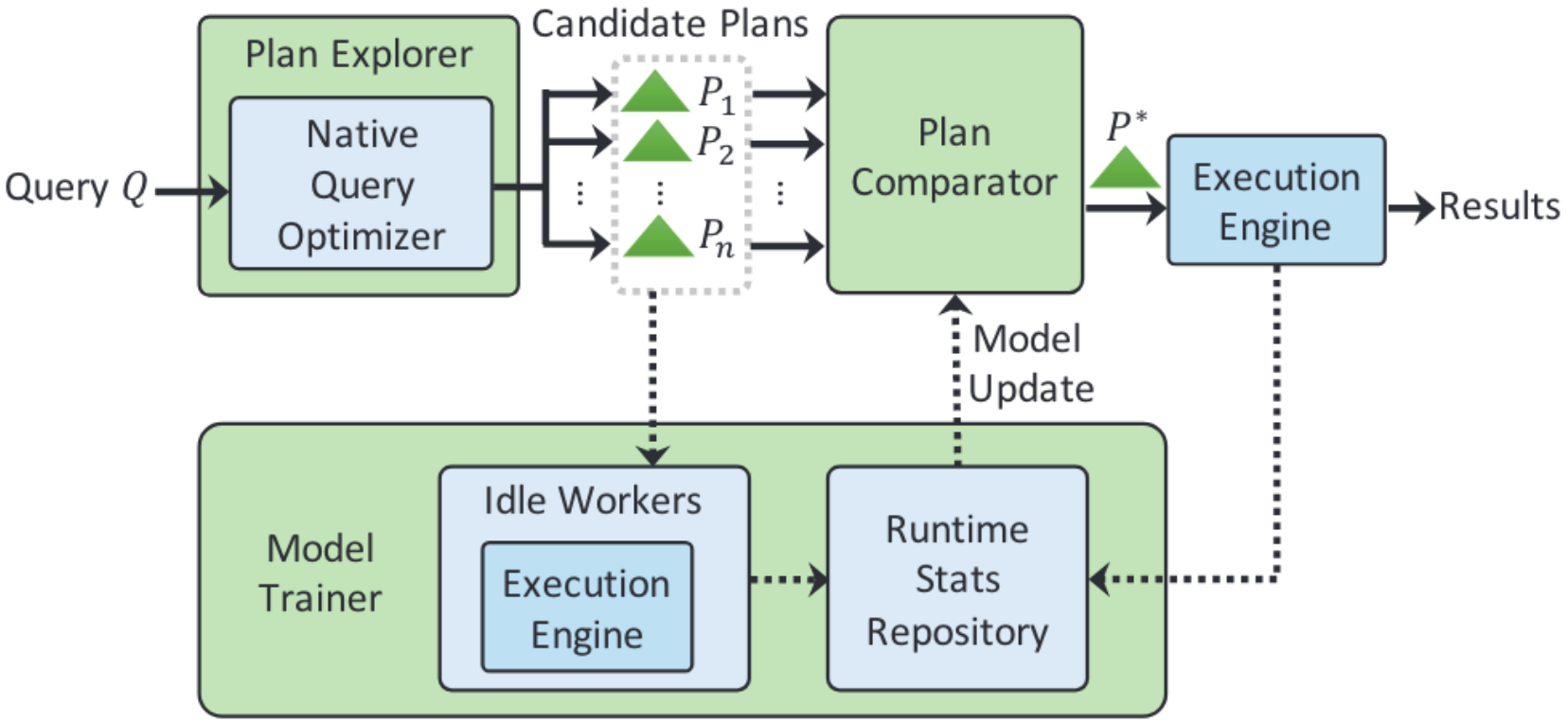


Architecture of Bao

[1] Ryan Marcus, et al. Bao: Making Learned Query Optimization Practical. SIGMOD 2021.

Learned Optimizer Enhancer

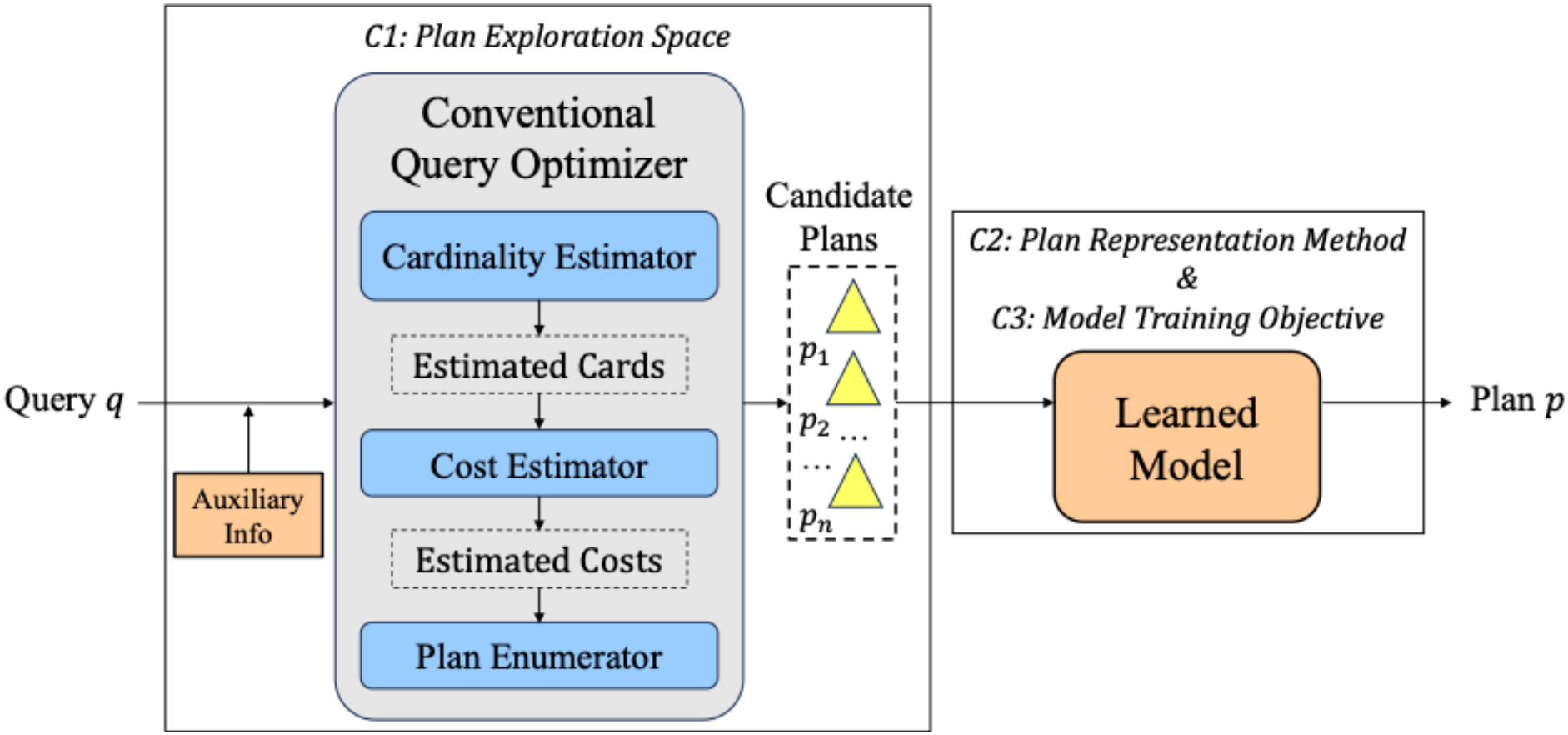
- Lero [1]



Architecture of Lero

[1] Rong Zhu, et al. Lero: A Learning-to-Rank Query Optimizer. VLDB 2023.

Learned Optimizer Enhancer



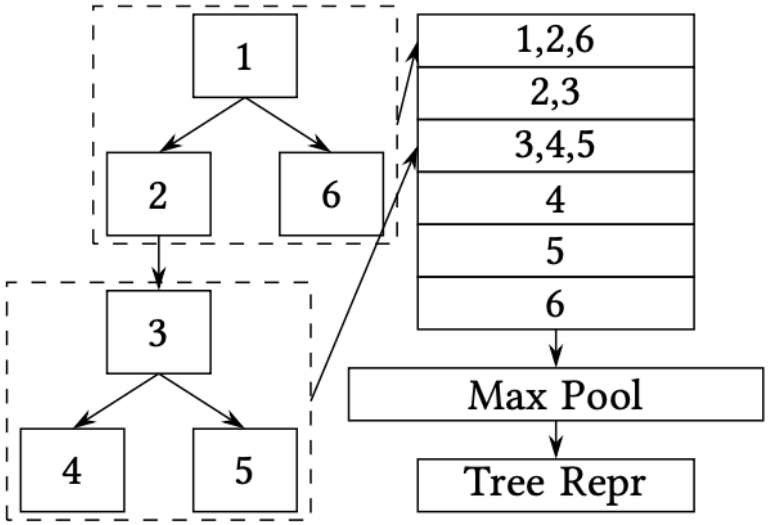
Architecture of learned optimizer enhancer

Learned Optimizer Enhancer

- C1: Plan Exploration Space
  - Existing methods
    - Bao: variant hint assignment (e.g., enable hash join, enable index scan).
    - Lero: cardinality perturbation (e.g., some cardinalities * 10 or * 0.1).
  - Limitations of existing methods
    - Bao: Fails when different query components demand distinct hints.
    - Lero: Limited to local plan space exploration.
    - Neither method directly explores diverse join order.

Learned Optimizer Enhancer

- C2: Plan Representation Method
  - Existing methods
    - Both Bao and Lero adopt Tree-CNN
  - Limitations of existing methods
    - Loss of global structure information: The max pooling-based feature aggregation in Tree-CNN fails to preserve hierarchical dependencies between nodes.



Tree-CNN procedure

Learned Optimizer Enhancer

- C3: Model Training Objective
  - Existing methods
    - Bao: learn to predict the execution time, taking MSE the loss function

$$\mathrm{MSE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\widehat{y_i} - y_i)^2,$$

$\widehat{y_i}$ is the predicted time and y_i is the actual time.

    - Lero: learn to rank the plans of query, taking cross-entropy loss function

$$l = -\log(\sigma(\widehat{y_i} - \widehat{y_j})),$$

$\widehat{y_i}$ and $\widehat{y_j}$ are the predicted time plan i and j, σ is sigmoid function.

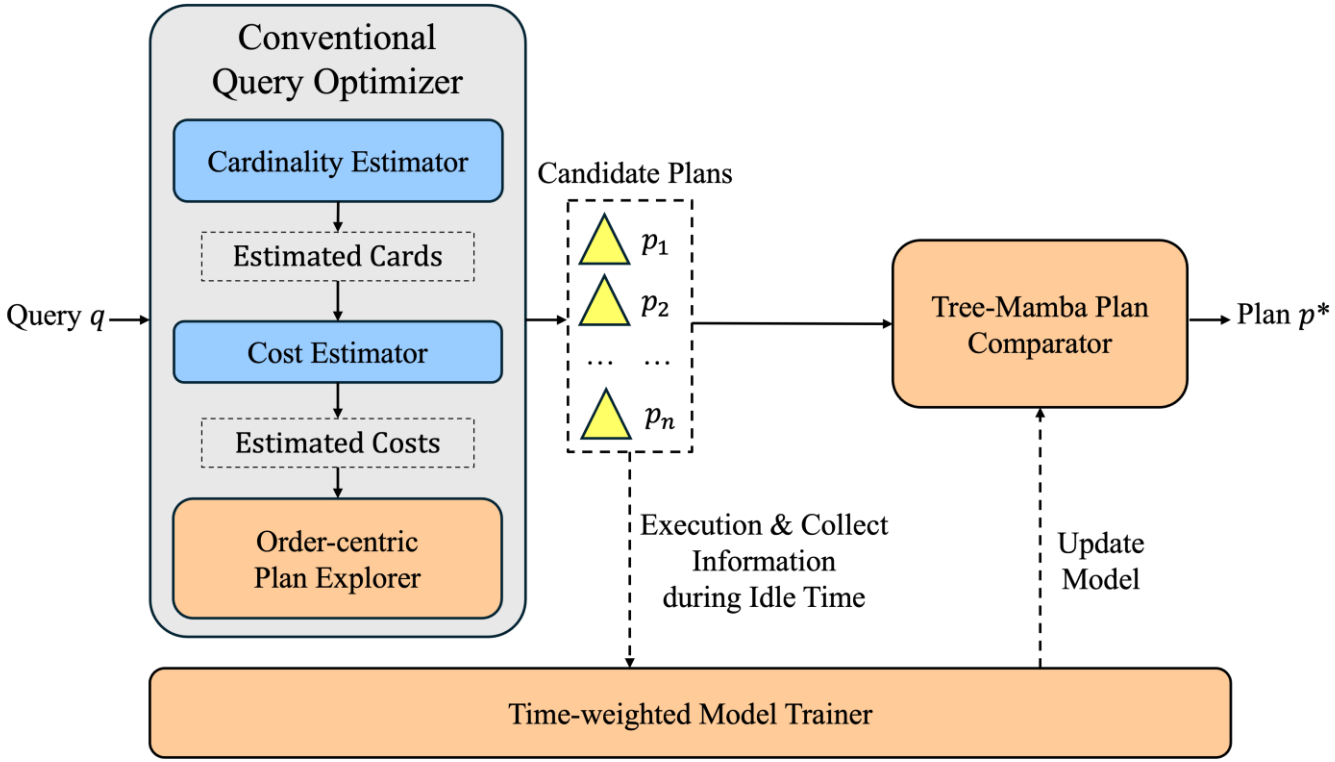
This loss function means that plan i has larger actual time than plan j.

Learned Optimizer Enhancer

- C3: Model Training Objective
  - Limitations of existing methods
    - Bao: learn to predict the execution time is impractical due to inherent estimation errors in query optimization.
    - Lero: learn-to-rank enhances the model training but neglects sample importance differentiation
      - For example, two query pairs with execution time differences of 0.1s vs. 10s are treated equally by Lero's loss function, despite the latter being significantly more critical for performance optimization

Learned Optimizer Enhancer

- Athena [1]

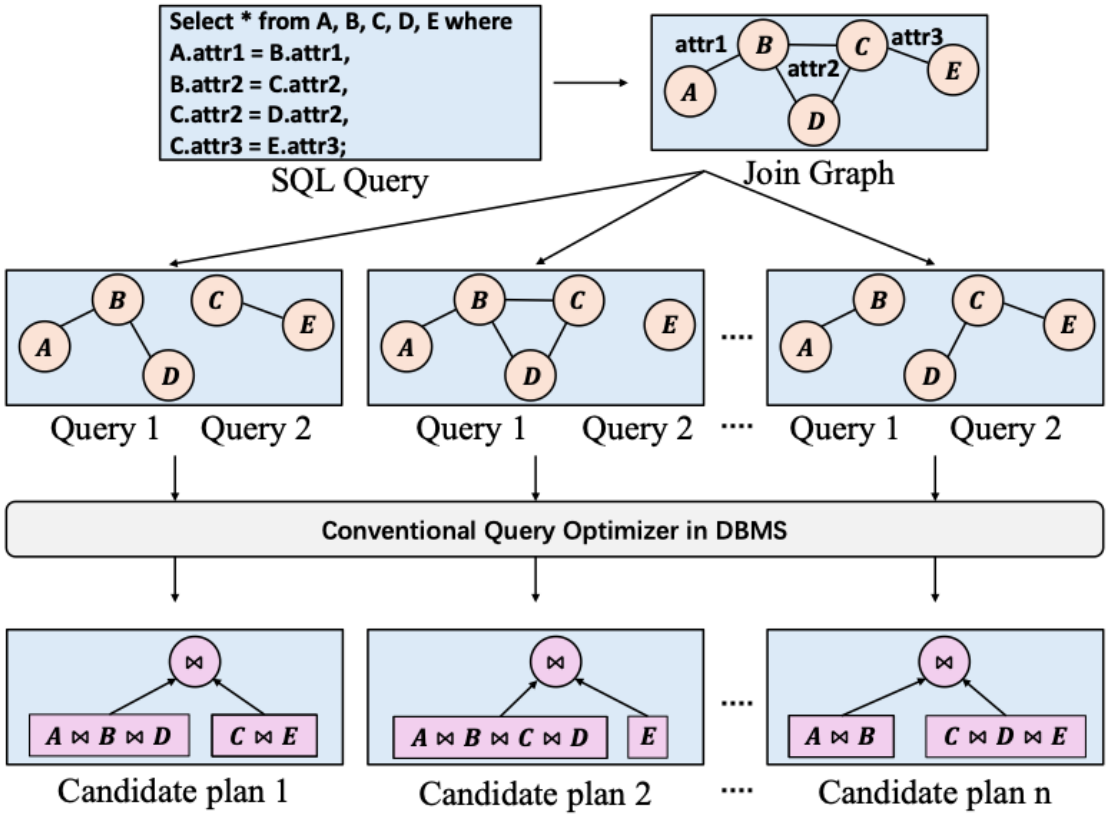


Architecture of Athena

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

- Athena [1] - Order-centric Plan Explorer

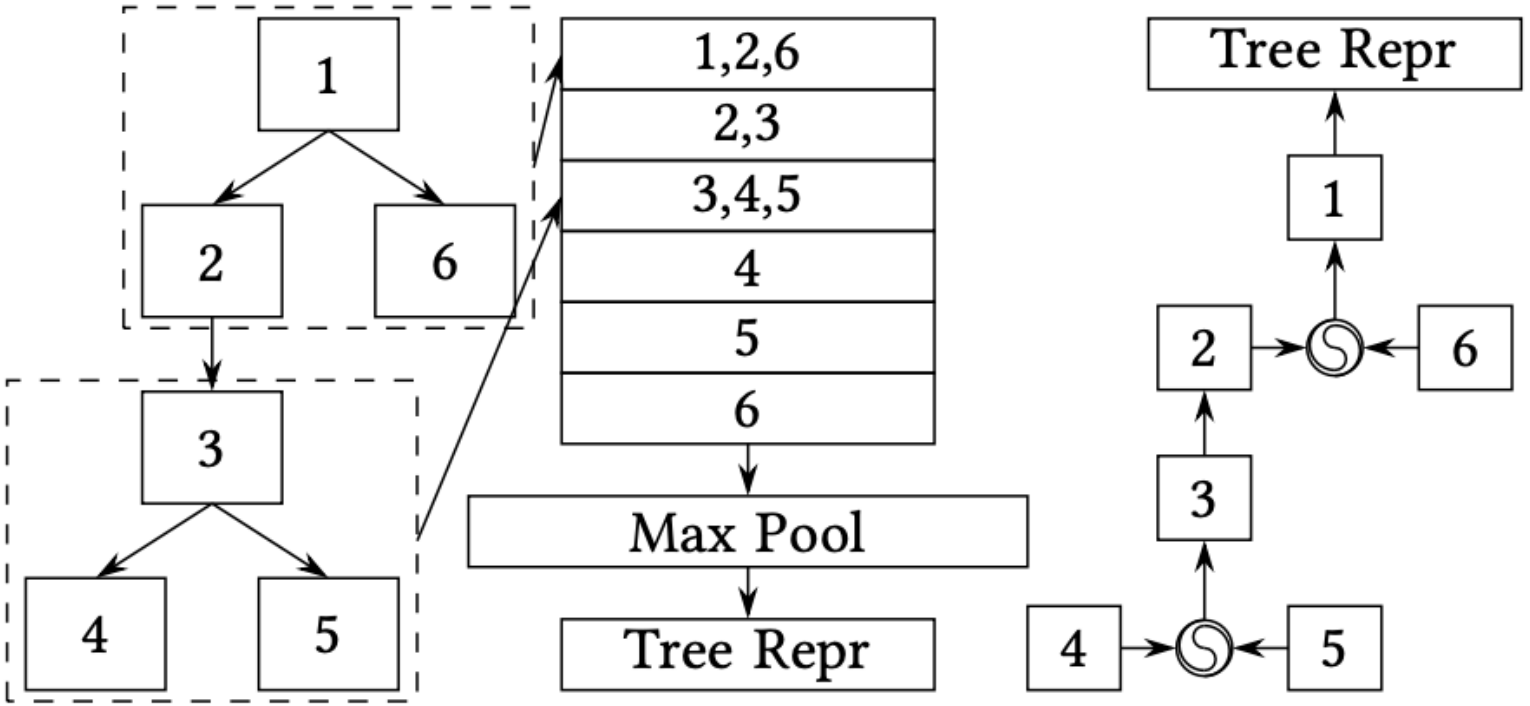


Plan explorer of Athena

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

- Athena [1] - Tree-Mamba Plan Comparator



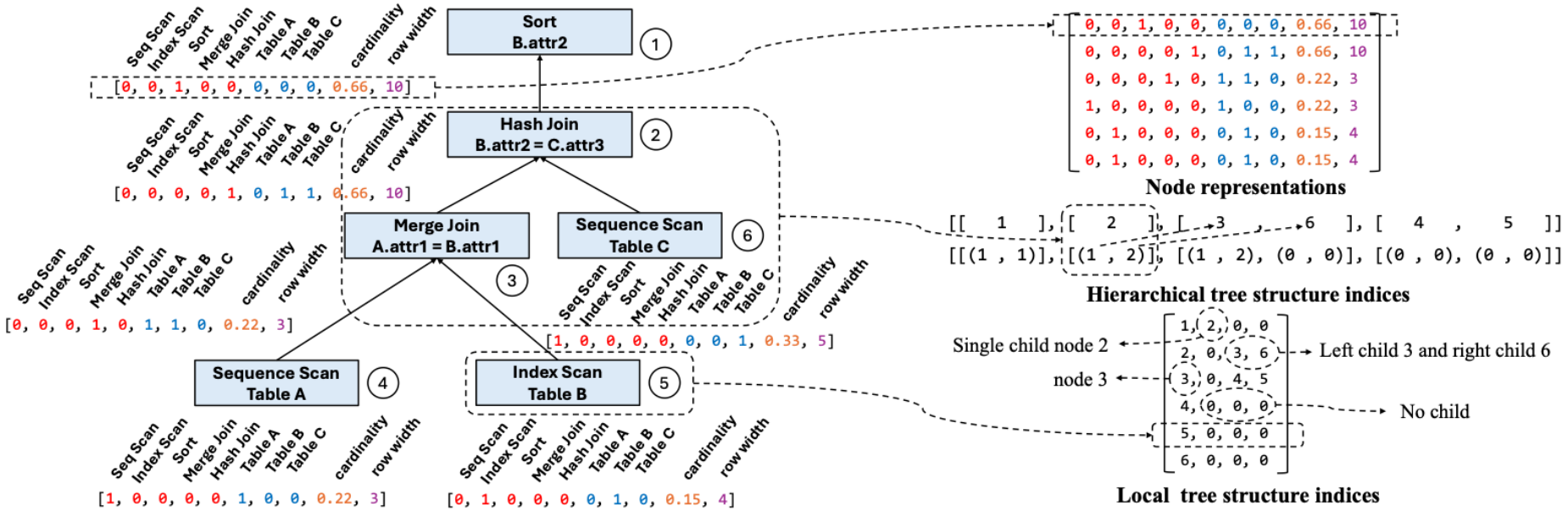
(a) Tree-CNN procedure

(b) Tree-RNN procedure

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

❖ Athena [1] - Tree-Mamba Plan Comparator - Model

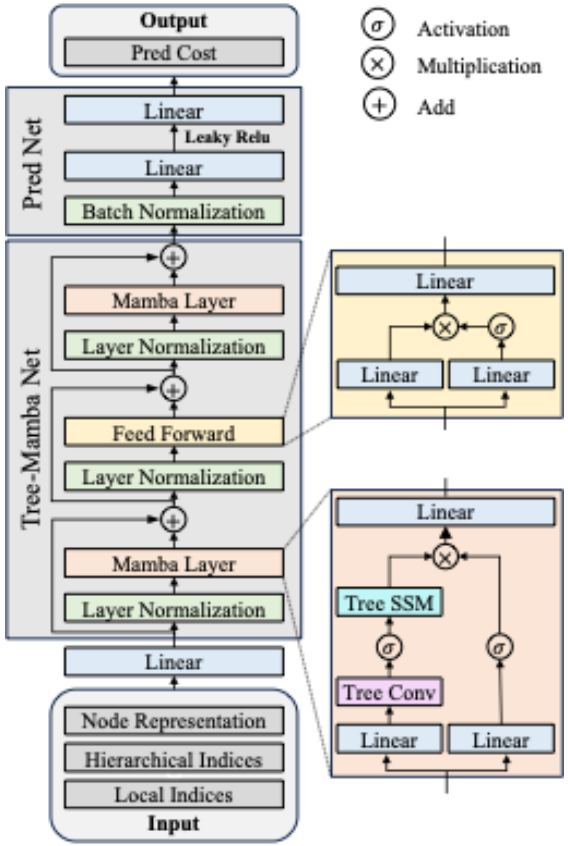


Featurizer of Athena

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

- Athena [1] - Tree-Mamba Plan Comparator - Model
  - New design
    - New Tree Conv could distinguish nodes with one or two child nodes.
    - Adopt Mamba to represent the tree. structure data with larger hidden space
    - Optimized matrix multiplication.



Model design of Athena

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

- Athena [1] - Time-weighted Model Trainer

$$P(p_1 \triangleright p_2) = \text{sigmoid}\left(M(p_1) - M(p_2)\right)$$

$$\mathcal{L}'(\mathcal{B};\tau,\gamma) = -\frac{\sum_{i=1}^{n} w_i^{\frac{1}{\tau}} \text{TaiLr}\left(P(p_1 \triangleright p_2);\gamma\right)}{\sum_{i=1}^{n} w_i^{\frac{1}{\tau}}}$$

$$\text{TaiLr}(P;\gamma) = \frac{\log(\gamma + (1-\gamma)P)}{1-\gamma}$$

Pair time difference weight

Distinguish the importance of different pair sample

TaiLr loss function

Avoid overconfidence for low loss sample

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Learned Optimizer Enhancer

- Athena [1]

The end-to-end latency (in seconds) on 4 benchmarks

Workload	JOB	STATS-CEB	TPC-DS	DSB
PostgreSQL	1666	8181	3773	1979
Bao	>40611	>14605	2089	2399
Lero	1661	7841	1106	1649
Athena	**954**	**4185**	**663**	**722**

[1] Runzhong Li, Qilong Li, Bo Tang, et al. Athena: An Effective Learning-based Framework for Query Optimizer Performance Improvement. SIGMOD 2025.

Other AI4DB Topics

- Self-tuning Database Configuration
- Automated Index Recommendation
- Learned Index
- Security Enhancement and Anomaly Detection
- Automated Operations and Resource Scheduling
- …

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

DBGroup

谢谢！

DBGroup @ SUSTech

Dr. Bo Tang (唐博)

tangb3@sustech.edu.cn