SIGMOD
PODS
2024

Tutorial

Learned Query Optimizer: What is New and What is Next

Rong Zhu, Lianggui Weng, Bolin Ding, Jingren Zhou

Institute for Intelligent Computing

Alibaba Group

Alibaba Group Alibaba Cloud TONGYI

Speakers

Rong Zhu
Alibaba Group

Lianggui Weng
Alibaba Group

Bolin Ding
Alibaba Group

Jingren Zhou
Alibaba Group

AI-Native Data Management

Towards being AI Native: Tasks and Values

- Leveraging AI for traditional database operators and improving performance
  - Database tuning: AI models as tuners
  - Statistics estimation: AI models as histograms
  - Query optimization: AI models as optimizers
- Leveraging AI for usability, data analytics, and new applications
  - Text-to-SQL: an LLM-aided translation task
  - Data Manipulation: AI-aided data cleaning, data augmentation, and feature generation
  - Text-to-Insights: AI-aided analytical pipeline generation, copilot
- Data processing systems for AI
  - LLMs as users of the data system, and their evaluation results as feedbacks
  - Multi-modal data and processors (machine learning models)
  - Scalability and composability of data processors

Learned Query Optimizer in AI-Native Data Management

- ❑ Part 1: Preliminaries
- ❑ Part 2: Learn to Cost Plans
- ❑ Part 3: Learn to Explore Plan Space
- ❑ Part 4: Applications and Deployment
- ❑ Part 5: Summary and Future Work

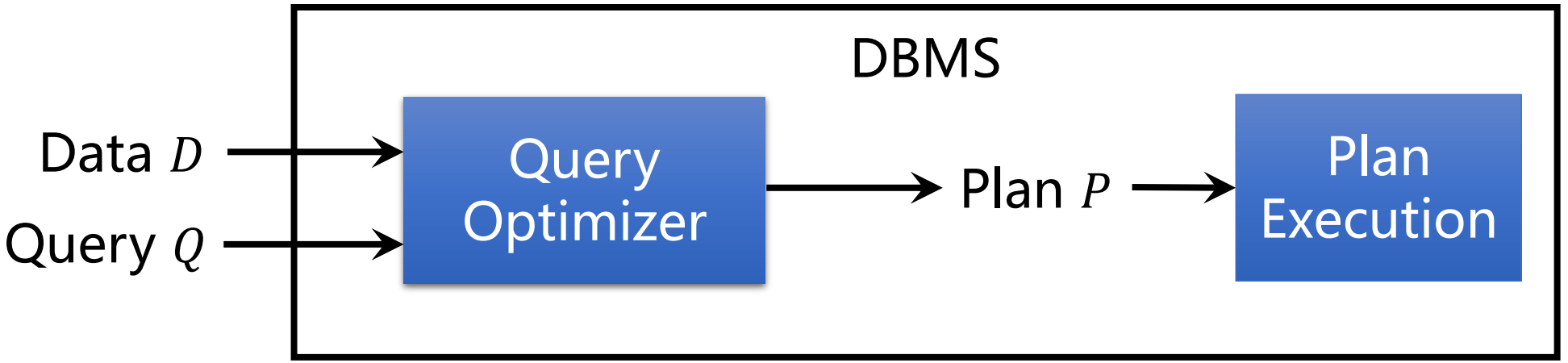
Bolin Ding, Rong Zhu and Jingren Zhou. Learned Query Optimizers, Foundations and Trends in Databases, 2024

Part 1: Preliminaries

- ❑ Query Optimizer
- ❑ Learned Query Optimizer

Query Optimizer: a Well-Defined Problem

- Generating best physical plan for input query
- Directly relates to the system performance
- Extensively studied in the literature

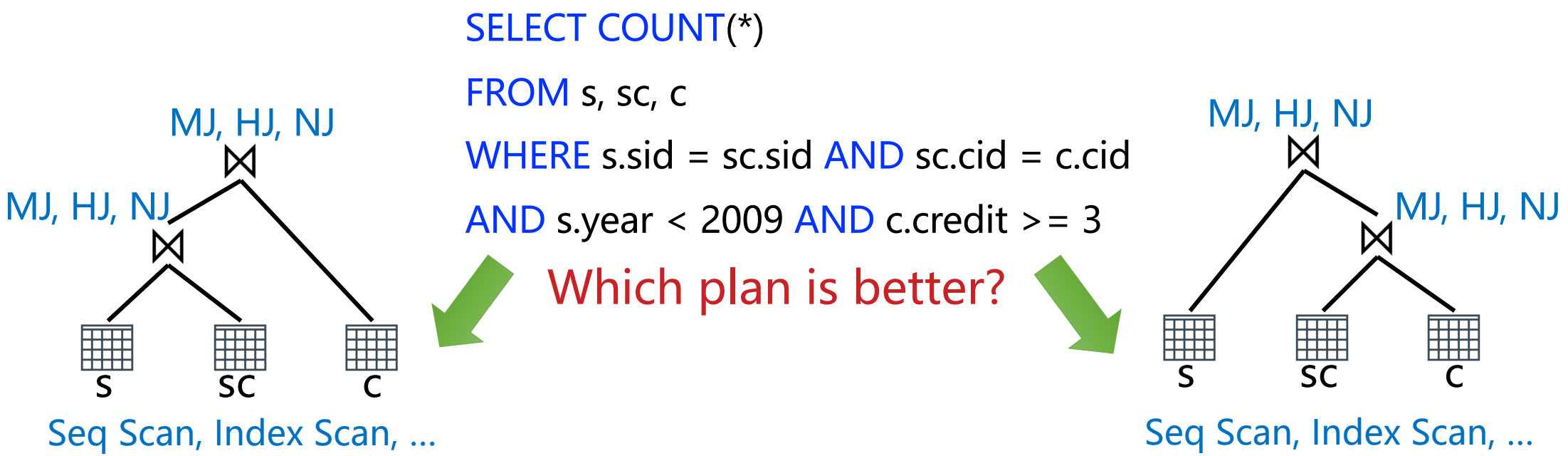


Query Optimizer: a Difficult Task

Undefined part: plan quality

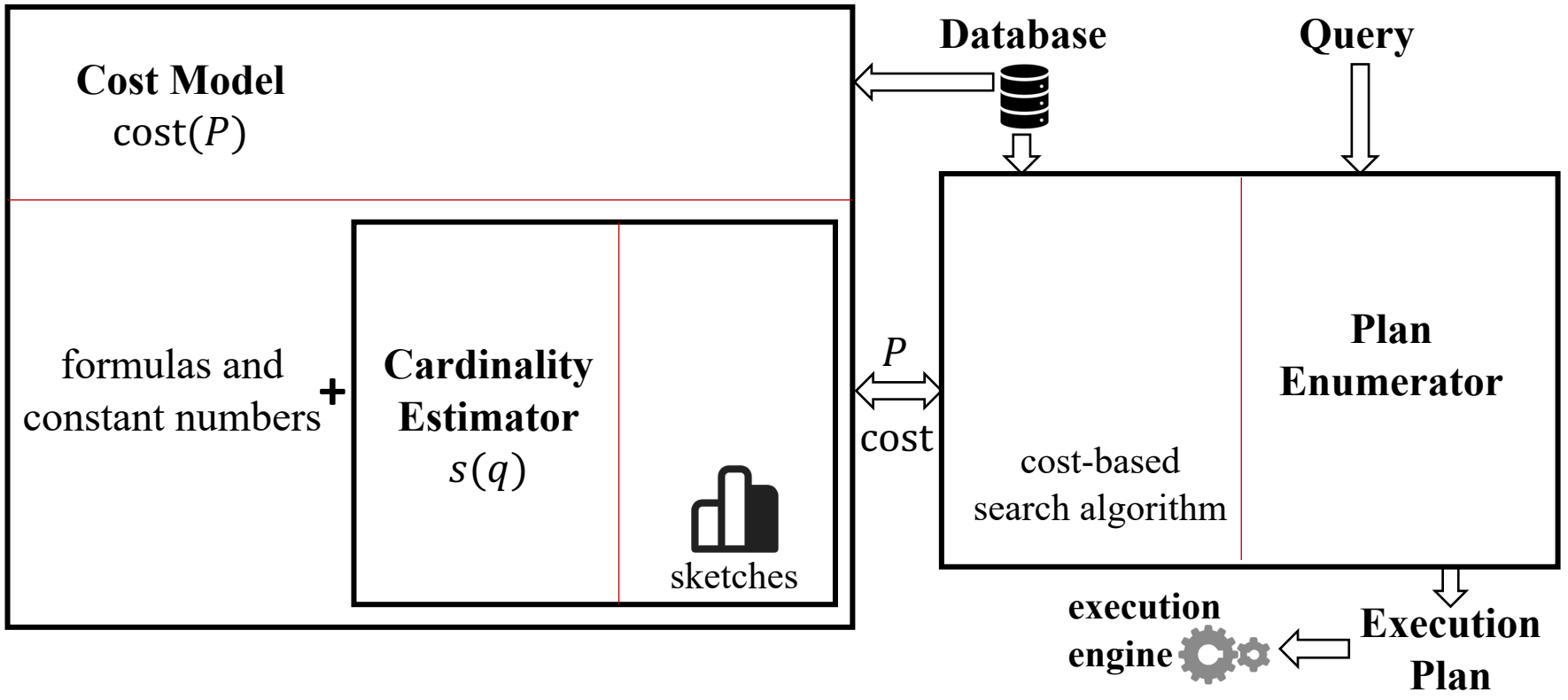
- Determined by data, query, system environment …

Large Search Space: join orders, access paths, …



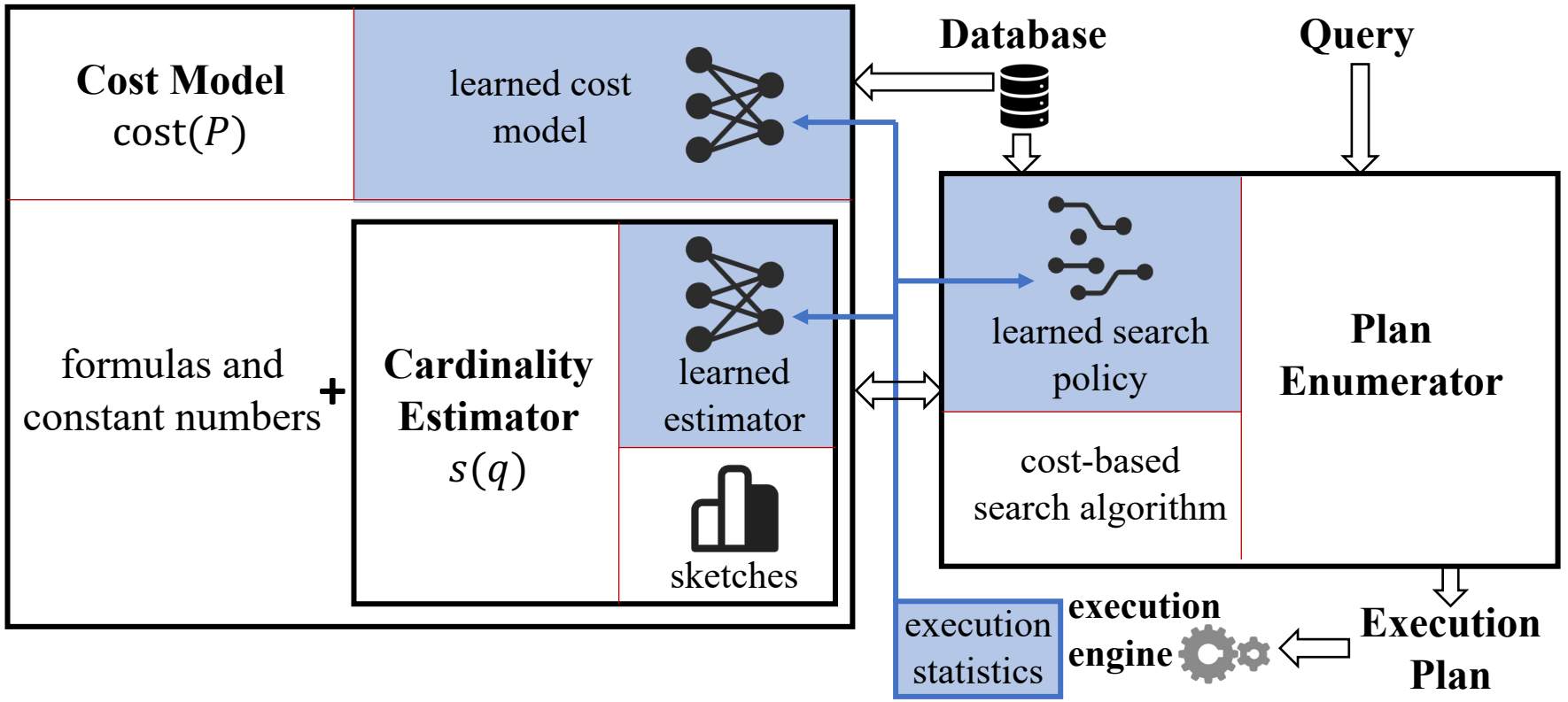
Learned Query Optimizer: a Pioneer in AI4DB

- A typical cost-based query optimizer



Learned Query Optimizer: a Pioneer in AI4DB

- Learned query optimizer: possibilities and opportunities



Learned Parts in Query Optimizer

Learned Query Optimizer in AI-Native Data Management

- ❑ Part 1: Preliminaries
- ❑ Part 2: Learn to Cost Plans
- ❑ Part 3: Learn to Explore Plan Space
- ❑ Part 4: Applications and Deployment
- ❑ Part 5: Summary and Future Work

Part 2: Learn to Cost Plans

- ❑ Cardinality & Cost
- ❑ Learn to Estimate Cardinality
- ❑ Learned Cost Model

Part 2: Learn to Cost Plans

- ❑ Cardinality & Cost
- ❑ Learn to Estimate Cardinality
- ❑ Learned Cost Model

Cardinality & Cost

- Cardinality: size $C(q)$ of the result of a query q
  - Require to estimate before actual execution of queries

- Cost: an approximation to query execution time/resource usage
  - Cost model in forms of experience-driven formulas
  - Example: a join on two relations with size (cardinality) s_1 and s_2

$$\texttt{cost}_{\texttt{J}}(s_1, s_2) = a_{2,0} \cdot s_1^2 + a_{0,2} \cdot s_2^2 + a_{1,1} \cdot s_1 s_2 + a_{1,0} \cdot s_1 + a_{0,1} \cdot s_2 + a_{0,0},$$

Expert-tuning Factors

Estimated Cardinality

Cardinality & Cost

- Cardinality to Cost: the error bound
  - Q-error of estimated cardinality $\hat{s}$ to exact value s

$$\|s/\hat{s}\|_{\mathrm{Q}} = \max(s/\hat{s}, \hat{s}/s)$$

  - Error bound on the cost of a join operator

Proposition 2.1 (Moerkotte *et al.*, 2009). Suppose the Q-errors of $\hat{s}_1$ and $\hat{s}_2$ for estimating s_1 and s_2 are bounded by ϵ, we have the Q-error of $\mathtt{cost}_{\mathrm{J}}(\hat{s}_1, \hat{s}_2)$ for estimating $\mathtt{cost}_{\mathrm{J}}(s_1, s_2)$ bounded by ϵ^2.

  - Error bound on the plan $\hat{P}$ with minimum cost and best plan P^*

$$P^* = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; s) \quad \text{and} \quad \hat{P} = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; \hat{s})$$

Theorem 2.1 (Moerkotte *et al.*, 2009). For a given query q, suppose the Q-error of cardinality estimation for intermediate result of any sub-query is bounded by ϵ, we have $\mathtt{cost}(\hat{P}; s) \leq \epsilon^4 \mathtt{cost}(P^*; s)$.

(Moerkotte et al, 2009) Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors, VLDB 2009

Cardinality & Cost

- Cardinality to Cost: the error bound
  - Q-error of estimated cardinality $\hat{s}$ to exact value s

$$\|s/\hat{s}\|_{\mathrm{Q}} = \max(s/\hat{s}, \hat{s}/s)$$

  - Error bound on the cost of a join operator

Proposition 2.1 (Moerkotte *et al.*, 2009). Suppose the Q-errors of $\hat{s}_1$ and $\hat{s}_2$ for estimating s_1 and s_2 are bounded by ϵ, we have the Q-error of $\mathtt{cost}_{\mathrm{J}}(\hat{s}_1, \hat{s}_2)$ for estimating $\mathtt{cost}_{\mathrm{J}}(s_1, s_2)$ bounded by ϵ^2.

  - Error bound on the plan $\hat{P}$ with minimum cost and best plan P^*

$$P^* = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; s) \quad \text{and} \quad \hat{P} = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; \hat{s})$$

Theorem 2.1 (Moerkotte *et al.*, 2009). For a given query q, suppose the Q-error of cardinality estimation for intermediate result of any sub-query is bounded by ϵ, we have $\mathtt{cost}(\hat{P}; s) \leq \epsilon^4 \mathtt{cost}(P^*; s)$.

worst-case

Inaccurate Cardinality

Inaccurate Cost

Sub-optimal Plan

(Moerkotte et al, 2009) Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors, VLDB 2009

Cardinality & Cost

- Cardinality to Cost: the error bound
  - Q-error of estimated cardinality $\hat{s}$ to exact value s

$$\|s/\hat{s}\|_{\mathrm{Q}} = \max(s/\hat{s}, \hat{s}/s)$$

  - Error bound on the cost of a join operator

Proposition 2.1 (Moerkotte *et al.*, 2009). Suppose the Q-errors of $\hat{s}_1$ and $\hat{s}_2$ for estimating s_1 and s_2 are bounded by ϵ, we have the Q-error of $\mathtt{cost}_{\mathrm{J}}(\hat{s}_1, \hat{s}_2)$ for estimating $\mathtt{cost}_{\mathrm{J}}(s_1, s_2)$ bounded by ϵ^2.

  - Error bound on the plan $\hat{P}$ with minimum cost and best plan P^*

$$P^* = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; s) \quad \text{and} \quad \hat{P} = \arg\min_{P \in \mathbb{P}(q)} \mathtt{cost}(P; \hat{s})$$

Theorem 2.1 (Moerkotte *et al.*, 2009). For a given query q, suppose the Q-error of cardinality estimation for intermediate result of any sub-query is bounded by ϵ, we have $\mathtt{cost}(\hat{P}; s) \leq \epsilon^4 \mathtt{cost}(P^*; s)$.

hopefully

Better Cardinality

Better Cost

Better Plan

(Moerkotte et al, 2009) Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors, VLDB 2009

Part 2: Learn to Cost Plans

- ❑ Cardinality & Cost
- ❑ Learn to Estimate Cardinality
- ❑ Learned Cost Model

Traditional Methods

- Most widely used in modern DBMS
- Histogram: attribute independent assumption $\Pr(A) \approx \prod_i \Pr(A_i)$
  - Fast inference, low storage cost, high estimation error
  - Multi-dimensional histogram: high storage cost
- Sampling: execute the query on a smaller fraction of data
  - Estimation accuracy and inference speed trade-off
  - Kernel-density estimation, join sampling, …

Learned Methods: An Overview

- Data-driven methods:
  - Learned models approximating $\Pr(A)$ of table T on attributes A
  - $C(q) = \Pr(q) * |T|$ for $q \subseteq T$
- Query-driven methods: for a workload (distribution) Q
  - Learned models as a function $C(q)$ for $q \in Q$
- Hybrid methods:
  - Utilize both data and query information

Data-Driven Methods

- Build models to capture data distribution $\Pr(A)$
- Different methods to model $\Pr(A)$
  - Auto-Regression model
  - Probabilistic graphical model (PGM)
  - …

Data-Driven Methods

- Auto-Regression models: chain rule of probability

$$\Pr(A) = \Pr(A_1) \cdot \Pr(A_2|A_1) \cdot \Pr(A_3|A_2, A_1) \cdots \Pr(A_n|A_{n-1}, A_{n-2}, \cdots, A_1)$$

  - Naru: single table
  - NeuroCard: multi-tables as a joined table
  - IAM: adding Gaussian mixture models

- Histogram: attribute independent assumption $\Pr(A) \approx \prod_i \Pr(A_i)$

(Naru) Deep Unsupervised Cardinality Estimation, VLDB 2019
(NeuroCard) NeuroCard: One Cardinality Estimator for All Tables, VLDB 2020
(IAM) Unsupervised Selectivity Estimation by Integrating Gaussian Mixture Models and an Autoregressive Model, EDBT 2022

Data-Driven Methods

- Auto-Regression models: model training

  Train a model to approximating
  $\Pr(x_k|x_{k-1}, x_{k-2}, \cdots, x_1)$ for $x_i \in A_i$

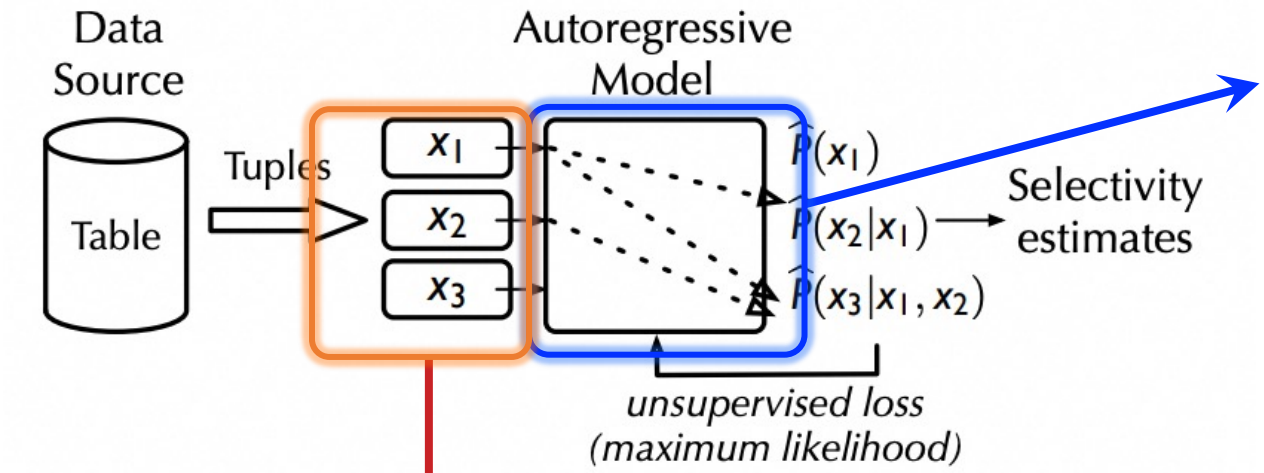


Figure from [Deep Unsupervised Cardinality Estimation, VLDB 2019]

Instantiate the network structures:

DARN, MADE, ResMADE and Transformer

Training data: draw tuple x uniformly random from table

Minimize loss $L = -\frac{1}{|R|}\sum_{x \in R} \log(\Pr(x))$

Data-Driven Methods

- Auto-Regression models: inference (probability computation)
  - Point query: compute the probability from model directly

$$\Pr(a, b, c) = \Pr(A_1 = a) \cdot \Pr(A_2 = b | A_1 = a) \cdot \Pr(A_3 = c | A_2 = b, A_1 = a)$$

  - Range query: sum the probability of all points: low efficiency

$$\Pr(a, b, c \in U) = \sum_{a', b', c' \in U} \Pr(a') \cdot \Pr(b' | a') \cdot \Pr(c' | b', a')$$

  - Speed up by sampling a number of points
    - Tradeoff between efficiency and accuracy

$$\Pr(a, b, c \in U) = \frac{|U|}{|S|} \cdot \sum_{a', b', c' \in S} \Pr(a') \cdot \Pr(b' | a') \cdot \Pr(c' | b', a')$$

Data-Driven Methods

Auto-Regression models: chain rule of probability

$$\Pr(A) = \Pr(A_1) \cdot \Pr(A_2|A_1) \cdot \Pr(A_3|A_2, A_1) \cdots \Pr(A_n|A_{n-1}, A_{n-2}, \cdots, A_1)$$

- Probabilistic graphic models (PGM)
  - Bayesian network: BayesNet and BayesCard
    - Both structural learning and model inference are NP-Hard problems

$$\Pr(A) = \prod_i \Pr(A_i|A_{pa(i)})$$

A	B	C	E	R
y	n	n	y	y
n	n	n	n	y
y	y	y	n	n
		...		

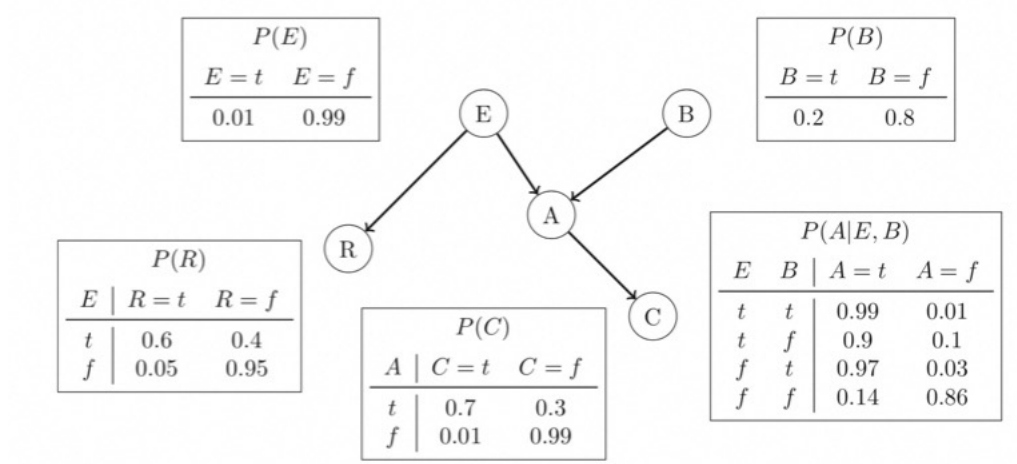


Figure from [A survey on Bayesian network structure learning from data, Progress in Artificial Intelligence, 2019]

(BayesNet) Lightweight Graphical Models for Selectivity Estimation Without Independence Assumptions, VLDB 2011
(BayesCard) BayesCard: A Unified Bayesian Framework for Cardinality Estimation, ArXiv, 2021

Data-Driven Methods

- Different probabilistic graphic models (PGM)
  - Sum-Product network (SPN): DeepDB
    - Model training: split data or attribute
      - Split the data by rows to find local independent attributes
      - Split attributes (columns) and build histograms on local data
    - Model inference: bottom up, linear cost to the node size
    - Accuracy, speed, and model size: sensitive to attribute correlations

c_age	c_region
80	EU
70	EU
60	ASIA
20	EU
...	...
...	...
20	ASIA
25	EU
30	ASIA
70	ASIA

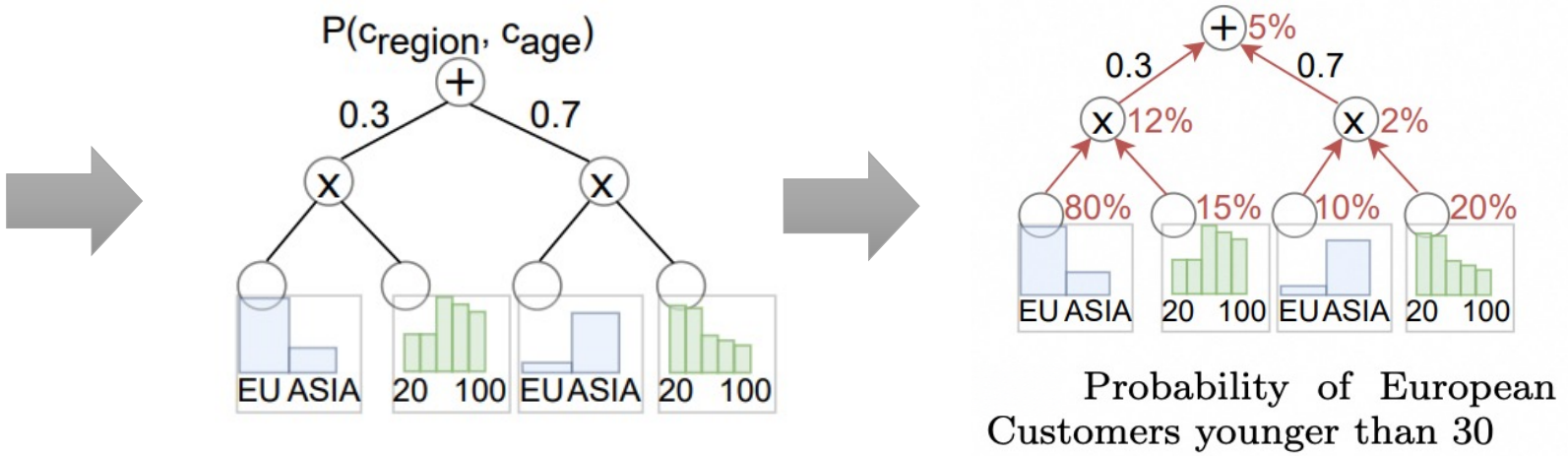
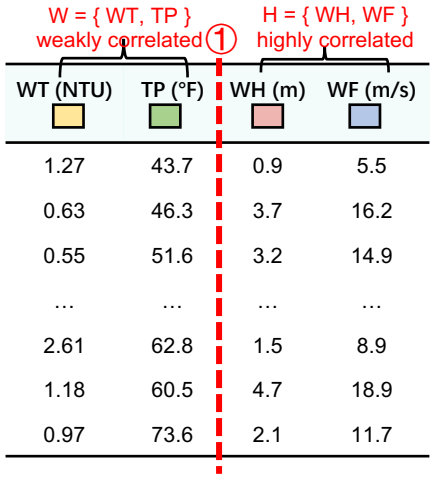
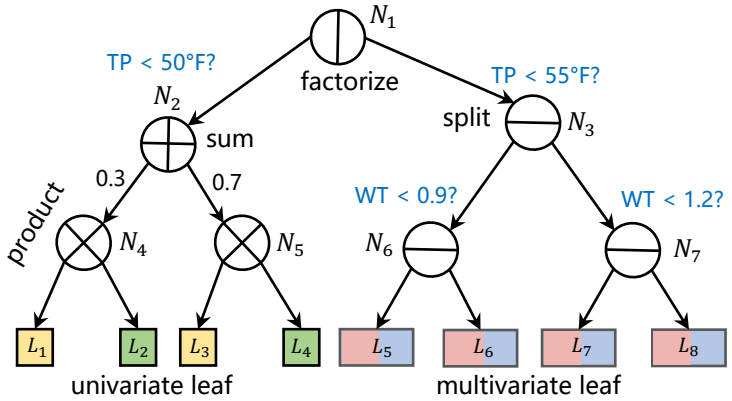


Figure from [DeepDB: Learn from Data, not from Queries!, VLDB 2020]

Data-Driven Methods

- Different probabilistic graphic models (PGM)
  - FSPN: FLAT
  - Adaptively process highly and weakly correlated attributes
    - Factorize node: losslessly split highly and weakly correlated attributes
    - SPN: model weakly correlated attributes
  - Combining the advantages from BN and SPN, no assumptions on correlations
  - High accuracy and small model size, low Inference time





(FLAT) FLAT: Fast, Lightweight and Accurate Method for Cardinality Estimation, VLDB 2021

Data-Driven Methods

- Other techniques
  - FactorJoin:
    histogram on join -> factor graph

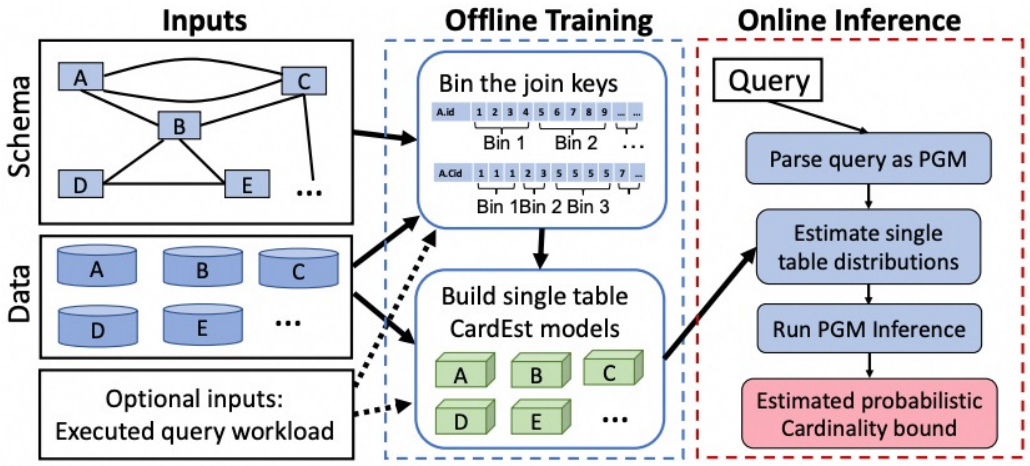


Figure from [FactorJoin: A New Cardinality Estimation Framework for Join Queries, SIGMOD 2023]

  - FACE: normalizing flow

  - Iris: pretraining summarization models

(FactorJoin) FactorJoin: A New Cardinality Estimation Framework for Join Queries, SIGMOD 2023
(FACE) FACE: A Normalizing Flow based Cardinality Estimator, VLDB 2021
(Iris) Pre-training Summarization Models of Structured Datasets for Cardinality Estimation, VLDB 2021

Learned Methods: An Overview

- Data-driven methods:
  - Learned models approximating $\Pr(A)$ of table T on attributes A
  - $C(q) = \Pr(q) * |T|$ for $q \subseteq T$

- Query-driven methods: for a workload (distribution) Q
  - Learned models as a function $C(q)$ for $q \in Q$

- Hybrid methods:
  - Utilize both data and query information

Query-Driven Methods

- Query featurization: from query to vectors
  - Tables
  - Join Conditions
  - Predicates: Columns, Values and Operators

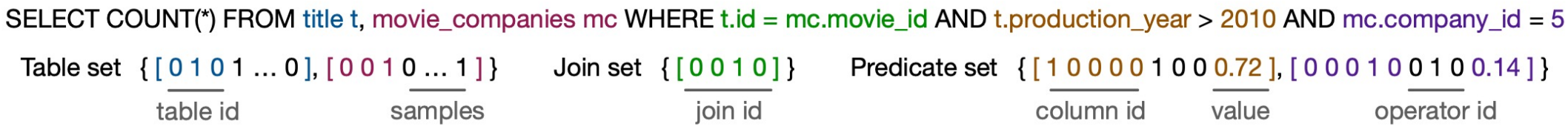


Figure from [Learned Cardinalities: Estimating Correlated Joins with Deep Learning, CIDR 2019]

- Build supervised model mapping q to $C(q)$
  - Statistical models
  - DNN-based models

Query-Driven Methods

- Statistical models:
  - Linear model
  - Tree-based Ensembles
  - XGBoost
  - QuickSel: a mixture of models with overlaps

(Linear) A Black-Box Approach to Query Cardinality Estimation, CIDR 2007
(Tree) Selectivity Estimation for Range Predicates using Lightweight Models, VLDB 2019
(XGBoost) Approximating Selectivity Functions using Low Overhead Regression Models, VLDB 2020
(QuickSel) QuickSel: Quick Selectivity Learning with Mixture Models, SIGMOD 2020

Query-Driven Methods

- DNN-based models:
  - FCN: fully connected neural network
    - FCN+Pool: adding pooling layers
  - MSCN: multi-set convolutional network
    - CRN: learning containment rate between a pair of queries
    - Robust-MSCN: query masking to improve generalization ability

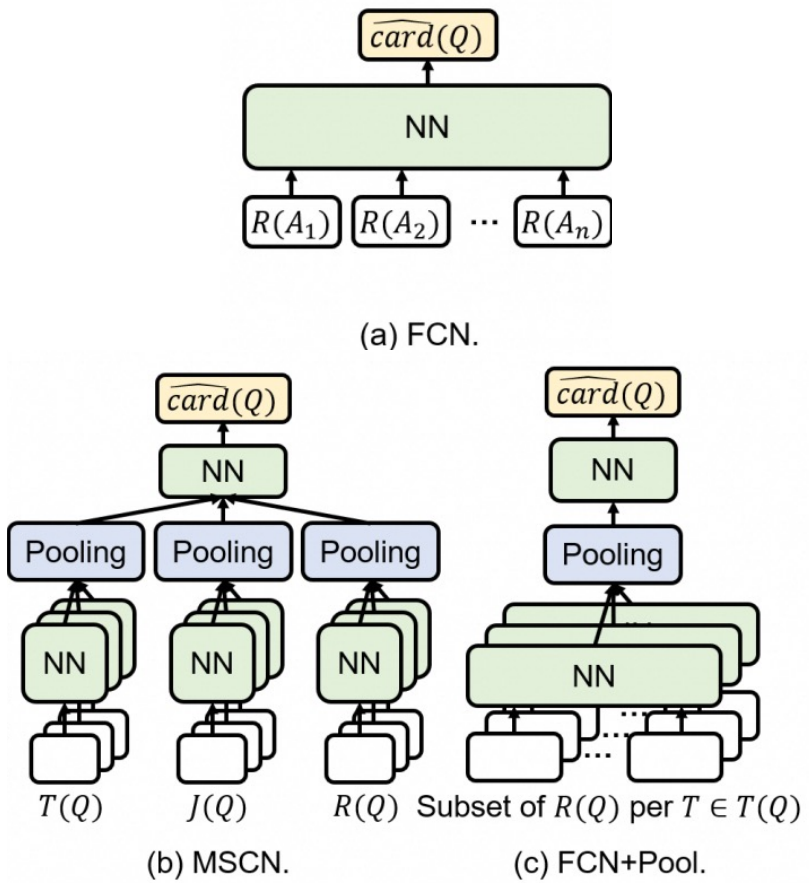


Figure from [Learned Cardinality Estimation: An In-depth Study, SIGMOD 2022]

(FCN) Cardinality estimation using neural networks, CASON 2015
(FCN+Pool) Learned Cardinality Estimation: An In-depth Study, SIGMOD 2022
(MSCN) Learned Cardinalities: Estimating Correlated Joins with Deep Learning, CIDR 2019
(CRN) Improved Cardinality Estimation by Learning Queries Containment Rates. EDBT 2020
(Robust-MSCN) Robust Query Driven Cardinality Estimation under Changing Workloads, VLDB 2023

Query-Driven Methods

- Advanced techniques
  - Fauce: an ensemble of deep models
    - Learn cardinality and uncertainty together
  - GL+: DNNs + segmentation techniques
    - Resolve the data hungry problem
  - NNGP: Bayesian deep learning models
  - LPCE: re-optimization framework

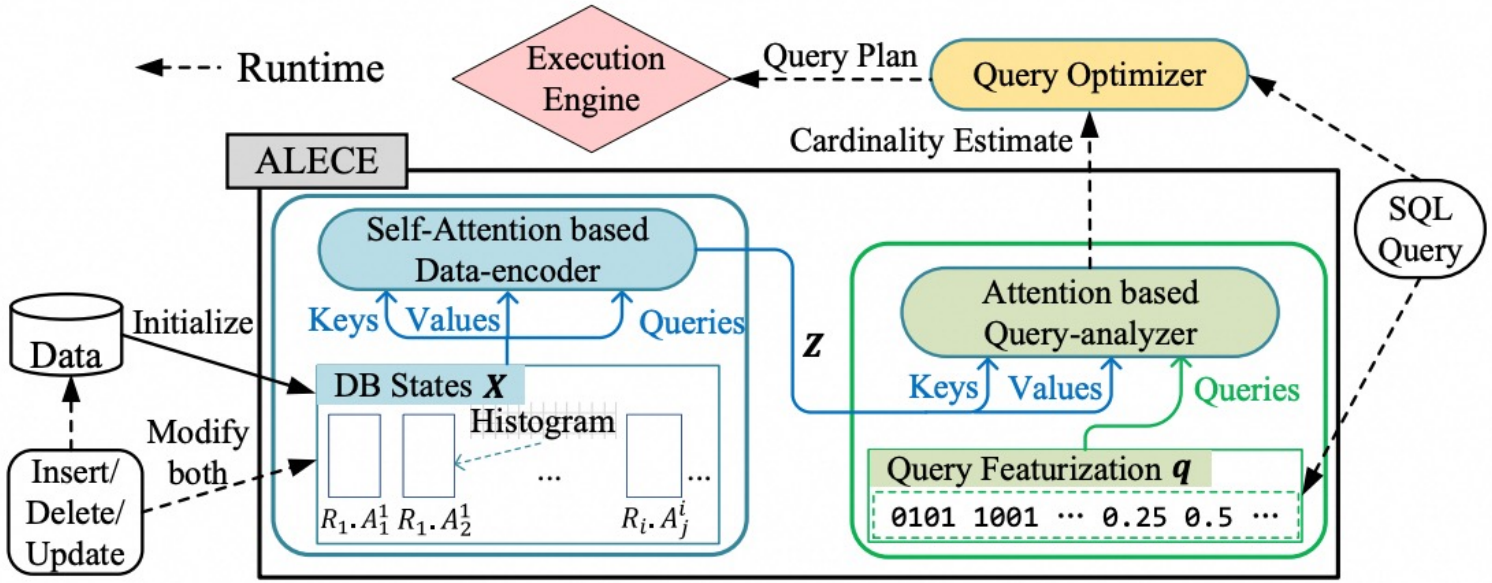
(Fauce) Fauce: Fast and Accurate Deep Ensembles with Uncertainty for Cardinality Estimation, VLDB 2021
(GL+) Learned Cardinality Estimation for Similarity Queries, SIGMOD 2021
(NNGP) Lightweight and Accurate Cardinality Estimation by Neural Network Gaussian Process, SIGMOD 2022
(LPCE) Speeding Up End-to-end Query Execution via Learning-based Progressive Cardinality Estimation, SIGMOD 2023

Learned Methods: An Overview

- Data-driven methods:
  - Learned models approximating $\Pr(A)$ of table T on attributes A
  - $C(q) = \Pr(q) * |T|$ for $q \subseteq T$
- Query-driven methods: for a workload (distribution) Q
  - Learned models as a function $C(q)$ for $q \in Q$
- Hybrid methods:
  - Utilize both data and query information

Hybrid Methods

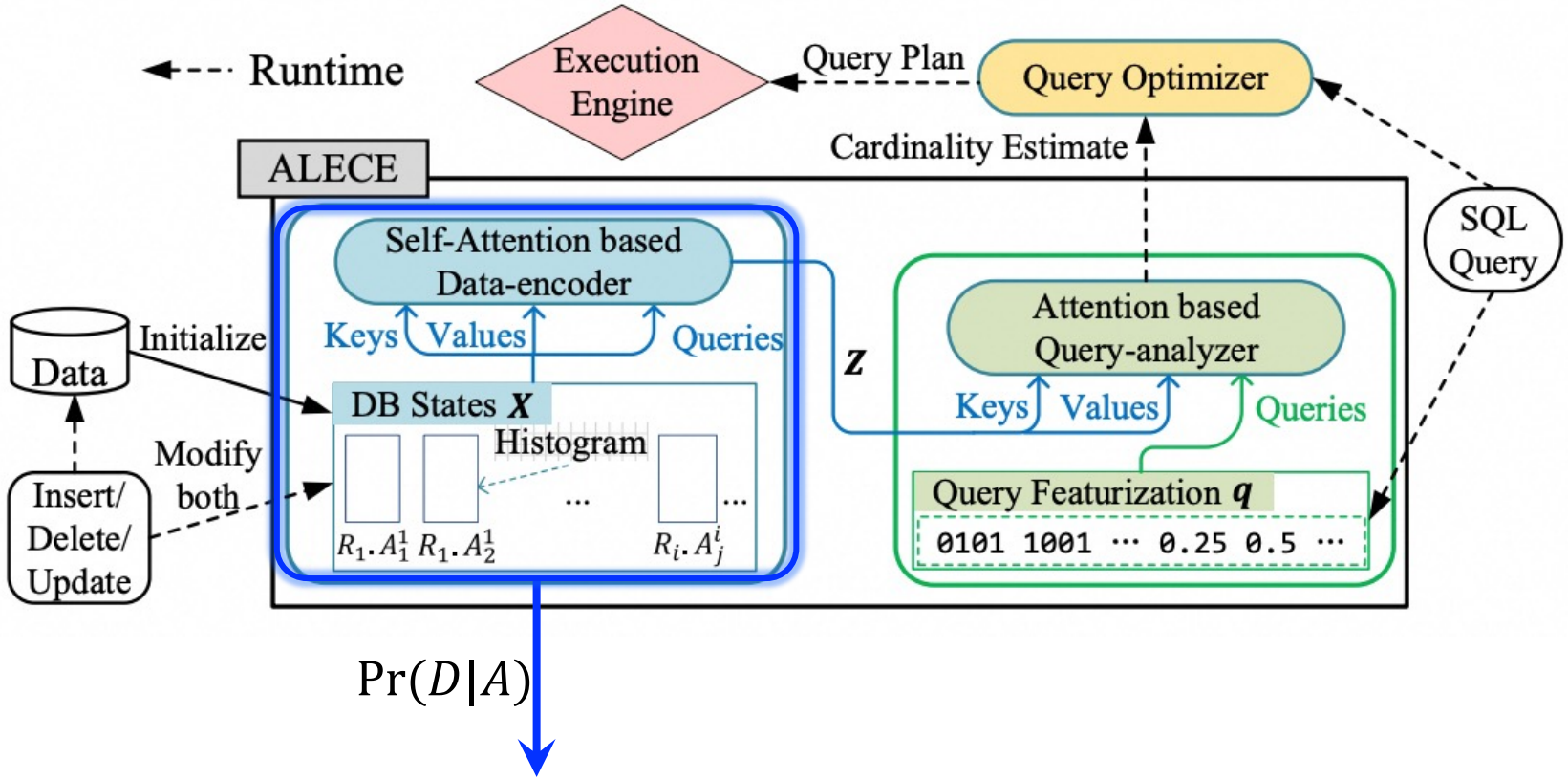
- ALECE: discover the hidden relationships between queries and underlying data using attention + transformer models
  - Data features (e.g., histogram of each attribute) + Query features
  - Process static and dynamic data in the same way



(ALECE) ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads, VLDB 2023

Hybrid Methods

- ALECE: utilize both query and data information

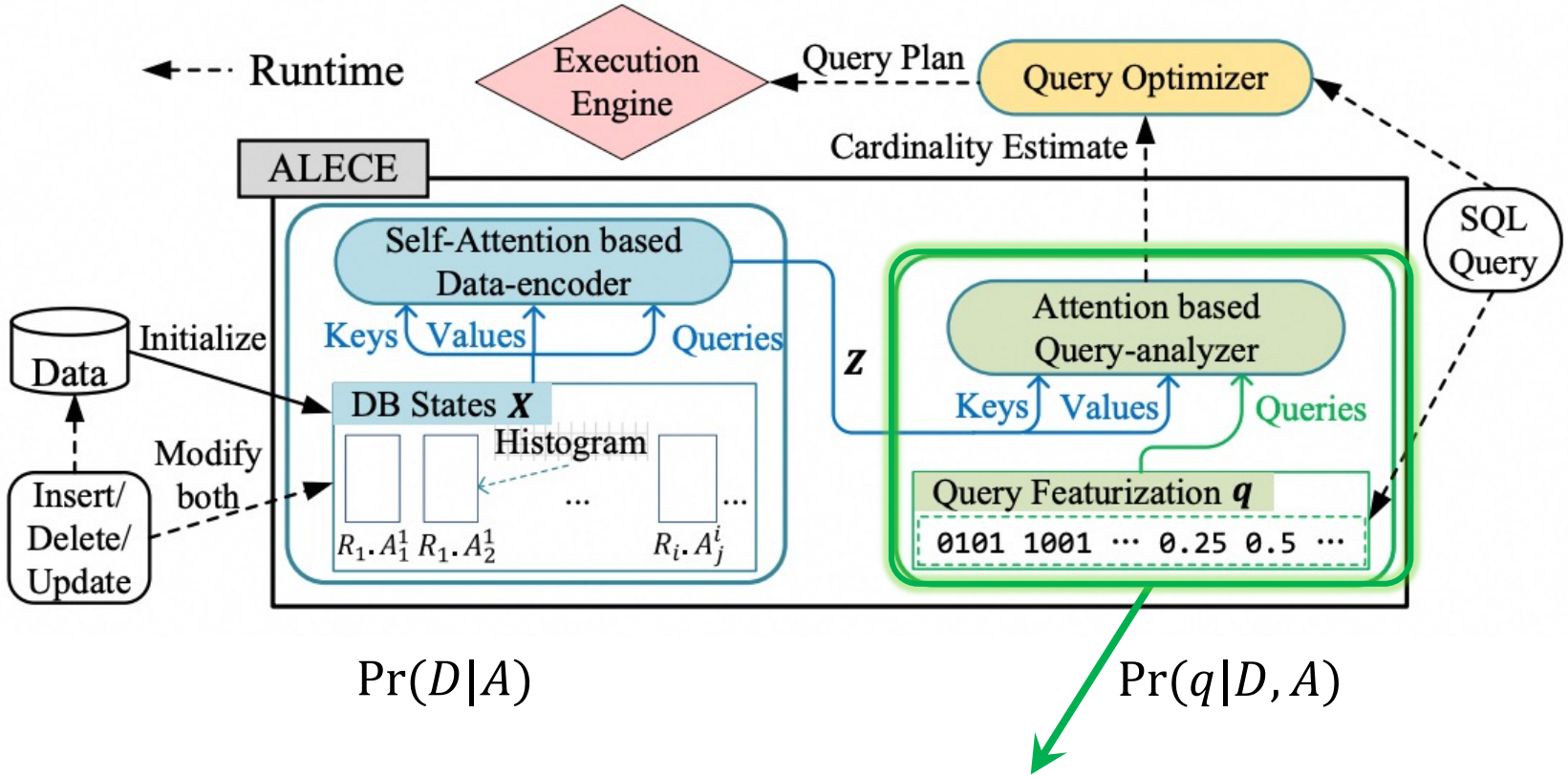


Data Encoder: an attention module learn the information of high-order data distributions using low-level histograms on each attribute

(ALECE) ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads, VLDB 2023

Hybrid Methods

- ALECE: utilize both query and data information



Query Analyzer: an attention module learn the joint information between data distribution and featuralized queries

(ALECE) ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads, VLDB 2023

Benchmark Evaluations

- Cardinality estimation on single table
  - Analyze whether these methods are suitable for deploying
- End-to-end evaluation of query optimization/processing
  - Guidance to select methods in various practical scenarios
- New benchmarks and comprehensive evaluations
  - New dataset STATS: real-world data with complicated distributions
  - End-to-End evaluation on actual DBMS (PostgreSQL)

(Single Table) Are We Ready For Learned Cardinality Estimation?, VLDB 2021
(Space Exploration) Learned Cardinality Estimation: A Design Space Exploration and A Comparative Evaluation, VLDB 2021
(STATS) Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation, VLDB 2021

Benchmark Evaluations

- Learned methods could help, but not always

Category	Method	Workload					
		JOB-LIGHT			STATS-CEB		
		End-to-End Time	Exec. + Plan Time	Improvement	End-to-End Time	Exec. + Plan Time	Improvement
Baseline	PostgreSQL	3.67h	3.67h + 3s	0.0%	11.34h	11.34h + 25s	0.0%
	TrueCard	**3.15h**	**3.15h + 3s**	**14.2%**	**5.69h**	**5.69h + 25s**	**49.8%**
Traditional	MultiHist	3.92h	3.92h + 30s	−6.8%	14.55h	14.53h + 79s	−28.3%
	UniSample	4.87h	4.84h + 96s	−32.6%	$> 25h$	−−	−−
	WJSample	4.15h	4.15h + 23s	−13.1%	19.86h	19.85h + 45s	−75.0%
	PessEst	3.38h	3.38h + 11s	7.9%	6.10h	6.10h + 43s	46.2%
Query-driven	MSCN	3.50h	3.50h + 12s	4.6%	$8.13h$	8.11h + 46s	28.3%
	LW-XGB	4.31h	4.31h + 8s	−17.4%	$> 25h$	−−	−−
	LW-NN	3.63h	3.63h + 9s	1.1%	11.33h	11.33h + 34s	0.0%
	UAE-Q	3.65h	3.55h+356s	−1.9%	11.21h	11.03h+645s	1.1%
Data-driven	NeuroCard^E	3.41h	3.29h + 423s	6.8%	12.05h	11.85h + 709s	−6.2%
	BayesCard	3.18h	3.18h + 10s	13.3%	7.16h	7.15h + 35s	36.9%
	DeepDB	3.29h	3.28h + 33s	10.3%	6.51h	6.46h + 168s	42.6%
	FLAT	3.21h	3.21h + 15s	12.9%	5.92h	5.80h + 437s	47.8%
Query + Data	UAE	3.71h	3.60h + 412s	−2.7%	11.65h	11.46h + 710s	−0.02%

Experimental data from [Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation, VLDB 2021]

Benchmark Evaluations: Dynamic Settings

Data	Model	Insert-heavy											Update-heavy							
		E2E Time(S)	Q-error				P-error				Size (MB)	Building Time(Min)	Latency (ms)	E2E Time(S)	Q-error				P-error			
			50%	90%	95%	99%	50%	90%	95%	99%					50%	90%	95%	99%	50%	90%	95%	99%
STATS	PG	7,790	190	$1.4\cdot10^5$	$1.1\cdot10^6$	$1.8\cdot10^7$	2.60	25.44	41.23	243	-	-	-	4,337	524	43,096	$3.7\cdot10^5$	$1.1\cdot10^7$	1.78	19.52	29.01	178
	Uni-Samp	6,646	1.35	12.47	$>10^{10}$	$>10^{10}$	2.71	16.77	21.65	201	3.34	**0.02**	239	6,002	**1.27**	$9.3\cdot10^9$	$>10^{10}$	$>10^{10}$	1.10	12.86	33.19	153.11
	NeuroCard	>30,000	17.37	1,388	7,402	$3.0\cdot10^5$	2.38	39.18	824	84,415	90.76	23.69	32.42	22,193	18.49	1,252	6,784	$3.1\cdot10^5$	2.10	31.75	347	11,730
	FLAT	>30,000	12.77	1,979	12,897	$8.6\cdot10^5$	2.22	70.08	2,202	8,738	210.33	53.77	13.67	24,319	16.00	2,376	17,811	$9.4\cdot10^5$	2.20	29.76	724	$1.4\cdot10^5$
	FactorJoin	>30,000	22.62	2,593	31,936	$1.6\cdot10^6$	2.35	66.12	876	8,384	1.64	0.42	1.33	>30,000	23.76	2,939	36,198	$2.4\cdot10^6$	2.75	115	912	4,409
	MLP	>30,000	$2.4\cdot10^6$	$>10^{10}$	$>10^{10}$	$>10^{10}$	7.35	491	2,294	6,887	8.63	3.11	3.17	>30,000	$2.3\cdot10^6$	$>10^{10}$	$>10^{10}$	$>10^{10}$	6.66	502	3,025	11,584
	MSCN	27,758	20.09	2,870	17,037	$2.6\cdot10^5$	2.46	87.23	736	8,738	**1.61**	11.87	**1.02**	25,766	18.89	2,382	16,947	$1.8\cdot10^6$	2.56	64.64	246	46,721
	NNGP	12,883	9.88	827	4,652	$2.6\cdot10^5$	1.41	5.31	27.33	3,588	32.44	0.89	32.66	11,985	8.15	567	3,639	$2.5\cdot10^5$	1.52	5.62	13.81	2,753
	ALECE	**2,901**	**1.29**	**5.07**	**11.54**	**62.52**	**1.07**	**1.34**	**1.59**	**2.32**	22.31	9.25	8.25	**2,402**	1.42	**5.06**	**10.40**	**44.69**	**1.08**	**1.45**	**2.13**	**2.86**
	Optimal	2,871	1	1	1	1	1	1	1	1	-	-	-	2,349	1	1	1	1	1	1	1	1
Job-light	PG	6,128	1.59	7.14	20.85	223	1.10	1.32	1.48	2.89	-	-	-	4,747	1.67	6.20	14.09	70.13	1.10	1.39	1.64	3.06
	Uni-Samp	6,401	1.23	2.59	6.53	$>10^{10}$	1.08	1.85	2.79	146	23.95	**0.52**	163	4,491	1.33	3.61	7.50	$>10^{10}$	1.12	1.71	2.26	3.67
	DeepDB	26,636	11.31	663	6,328	$5.4\cdot10^5$	1.66	13.71	26.01	49.57	11.52	11.93	11.92	23,190	10.64	628	4,883	$8.8\cdot10^5$	1.50	11.81	27.71	135
	NeuroCard	27,744	14.73	978	6,766	$4.7\cdot10^5$	1.81	12.40	37.25	144	42.93	9.11	15.90	16,553	13.49	1,000	6,489	$1.5\cdot10^6$	1.60	15.10	25.94	50.68
	FLAT	23,876	11.22	626	5,786	$4.4\cdot10^5$	1.66	13.98	27.26	89.67	11.10	17.83	3.96	22,531	10.13	490	4,221	$1.1\cdot10^6$	1.42	12.87	34.54	195
	FactorJoin	16,665	25.14	3,456	27,938	$2.5\cdot10^6$	1.52	5.45	10.23	47.09	4.66	26.22	1.33	13,071	27.31	6,134	36,155	$2.2\cdot10^6$	1.42	4.83	9.43	22.75
	MLP	5,610	4.30	23.14	59.37	2,411	1.13	1.70	2.10	3.12	6.19	2.68	2.28	5,351	5.59	62.02	138	1,441	1.17	2.02	2.88	4.09
	MSCN	26,665	24.42	1,809	9,217	$1.4\cdot10^5$	2.07	20.98	28.58	106	**1.56**	33.37	**0.77**	29,118	26.77	1,448	7,980	$1.5\cdot10^5$	2.59	19.87	61.18	257
	NNGP	15,430	$4.5\cdot10^8$	$>10^{10}$	$>10^{10}$	$>10^{10}$	3.57	8.39	11.97	24.47	32.44	0.83	29.86	4,959	4.99	59.86	190	50,036	1.20	2.05	2.99	4.96
	ALECE	**5,168**	**1.14**	**2.08**	**3.85**	**14.82**	**1.07**	**1.24**	**1.32**	**1.77**	22.25	7.73	7.28	**3,839**	**1.09**	**1.93**	**3.63**	**16.28**	**1.07**	**1.23**	**1.29**	**1.57**
	Optimal	5,063	1	1	1	1	1	1	1	1	-	-	-	3,792	1	1	1	1	1	1	1	1
TPCH	PG	**3,230**	1.35	4.88	7.90	16.16	1.44	**1.57**	**1.59**	**1.62**	-	-	-	2,385	1.33	5.01	6.87	14.64	**1.41**	**1.56**	**1.59**	**1.62**
	Uni-Samp	>30,000	1.22	2.94	$>10^{10}$	$>10^{10}$	1.81	2.22	45.60	514	23.29	**0.17**	24.52	>30,000	1.30	3.41	$>10^{10}$	$>10^{10}$	1.75	19.35	50.34	211
	NeuroCard	10,616	43.43	7,071	27,735	$3.5\cdot10^5$	3.75	10.91	22.48	106	704.98	34.38	34.77	6,562	45.73	5,896	20,931	$1.2\cdot10^5$	3.16	11.04	16.49	245
	FLAT	11,428	46.04	12,236	70,383	$8.4\cdot10^7$	3.56	10.91	25.71	1,182	168.65	41.42	10.06	7,864	42.32	7,328	26,470	$3.7\cdot10^5$	3.56	15.82	21.70	425
	MLP	7,261	51,475	$1.4\cdot10^7$	$3.5\cdot10^7$	$4.7\cdot10^9$	4.71	14.11	22.74	36.09	8.63	3.57	3.62	8,342	79,317	$3.7\cdot10^7$	$9.1\cdot10^7$	$>10^{10}$	5.12	17.63	32.74	63.39
	MSCN	7,718	36.36	6,709	26,744	$4.4\cdot10^5$	3.58	15.11	25.71	387	**1.58**	23.43	**0.79**	8,353	41.66	5,457	21,433	$1.7\cdot10^5$	3.80	14.86	25.69	472
	NNGP	18,618	$1.3\cdot10^9$	$>10^{10}$	$>10^{10}$	$>10^{10}$	5.56	47.40	59.18	99.55	32.44	0.92	41.76	3,128	25.88	432	986	4,968	1.54	7.98	10.90	20.66
	ALECE	3,233	**1.15**	**2.29**	**3.37**	**8.86**	**1.43**	**1.57**	1.60	**1.62**	22.34	11.62	10.35	**2,380**	**1.25**	**2.11**	**2.95**	**10.87**	**1.41**	**1.56**	**1.59**	**1.62**
	Optimal	3,227	1	1	1	1	1	1	1	1	-	-	-	2,378	1	1	1	1	1	1	1	1

Experimental data from [ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads, VLDB 2023]

Part 2: Learn to Cost Plans

- ❑ Cardinality & Cost
- ❑ Learn to Estimate Cardinality
- ❑ Learned Cost Model

Learned Cost Model

- Cost: an approximation to execution time/resource usage
  - Traditional cost model: expert-tuned formulas on cardinality

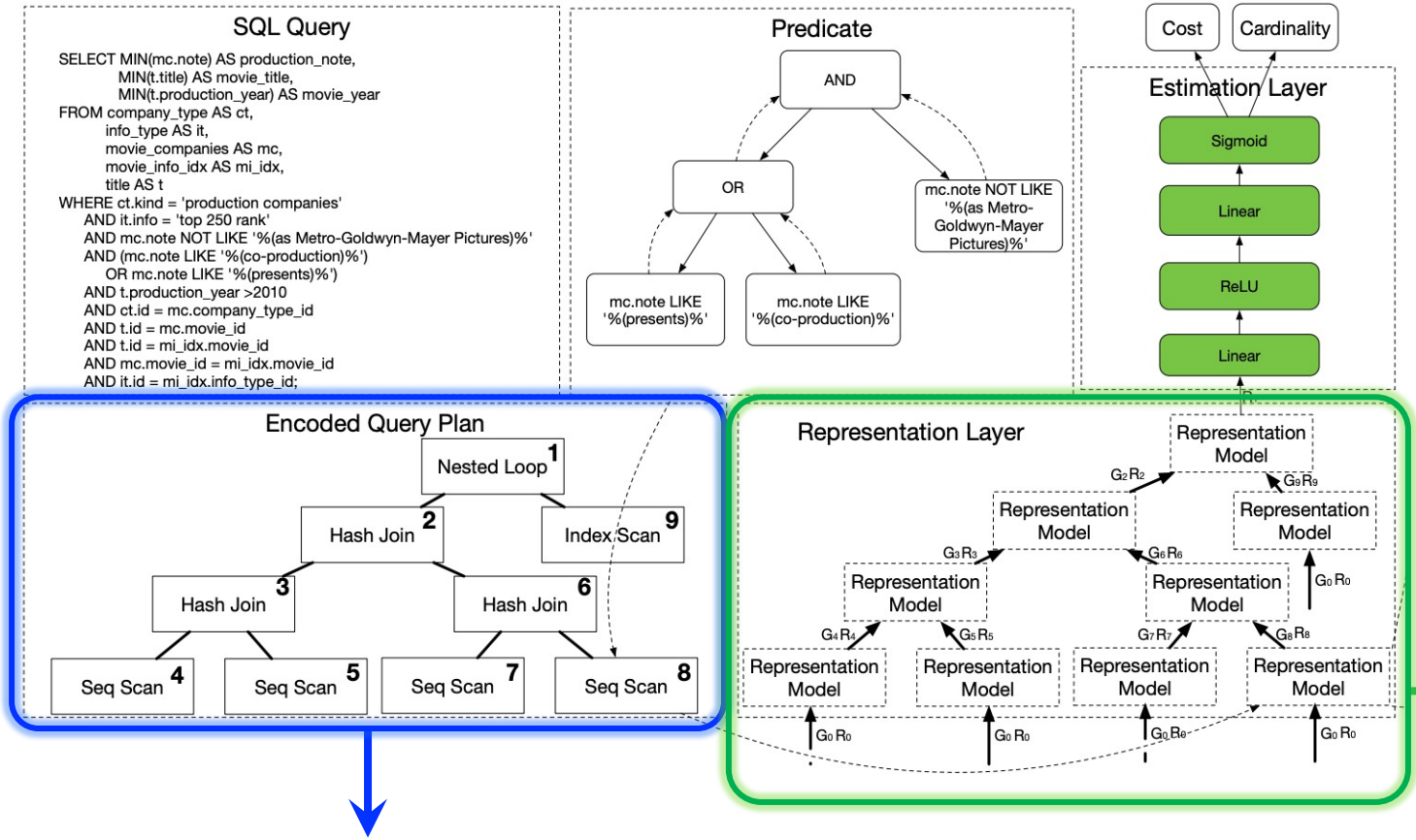
$$\texttt{cost}_{\texttt{J}}(s_1, s_2) = a_{2,0} \cdot s_1^2 + a_{0,2} \cdot s_2^2 + a_{1,1} \cdot s_1 s_2 + a_{1,0} \cdot s_1 + a_{0,1} \cdot s_2 + a_{0,0},$$

- Learned methods
  - Costing a single plan:
    use previous plans to train a cost model
  - Costing plans for concurrent queries:
    consider the correlations between plans and predict the cost

Cost Estimation for Single Plans

A typical LSTM model
$(h_t, m_t) = \text{LSTM}(x_p, h_{t-1}, m_{t-1})$

- Tree LSTM: plan as a sequence



Encode each node in the plan

Output and Memory of left and right child

$(h_p, m_p) = \text{LSTM}(x_p, (h_l, m_l), (h_r, m_l))$

Encoding of node P

Output and Memory of node P

Traverse each node in the plan as a sequence to feed into the LSTM representation model using the node and children information

Figure from [An End-to-End Learning-based Cost Estimator, VLDB 2019]

Cost Estimation for Single Plans

- Tree convolutional network (TCN): plan as a tree
  - Triangle sliding window on each node and its two children

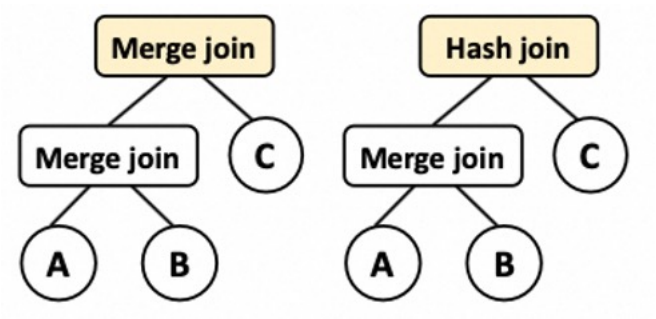
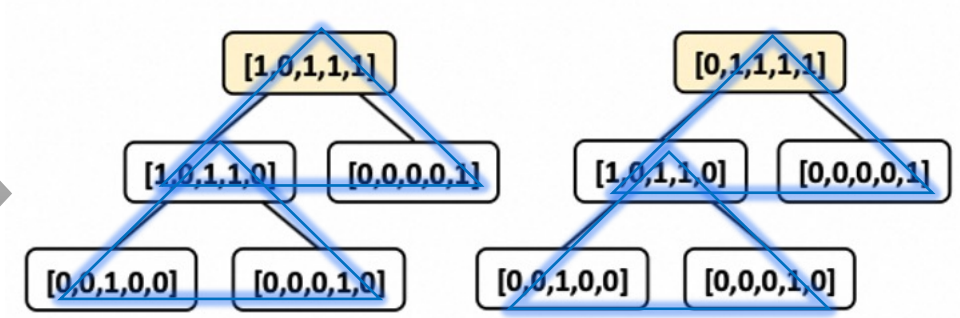


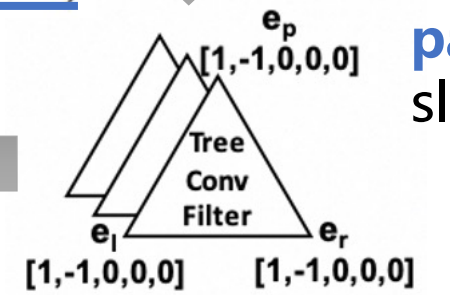
Figure from [Neo: A Learned Query Optimizer, VLDB 2019]



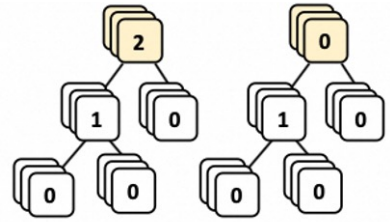
Features on each node

$$\mathbf{x}'_v = \sigma\left(W_{\text{op}} \cdot \mathbf{x}_v + W_{\text{left}} \cdot \mathbf{x}_{\text{left}(v)} + W_{\text{right}} \cdot \mathbf{x}_{\text{right}(v)} + \mathbf{b}\right)$$

Tree Convolution Filter: **shared model parameters** on each sliding window



Recursion



Aggregated results

Additional layers on final output to predict the cost

Cost Estimation for Single Plans

- Other ways to capture the tree structure information
  - Saturn: traverse and compress plans as vectors
  - QueryFormer: a transformer to learn embeddings of plans, appliable to multiple downstream tasks

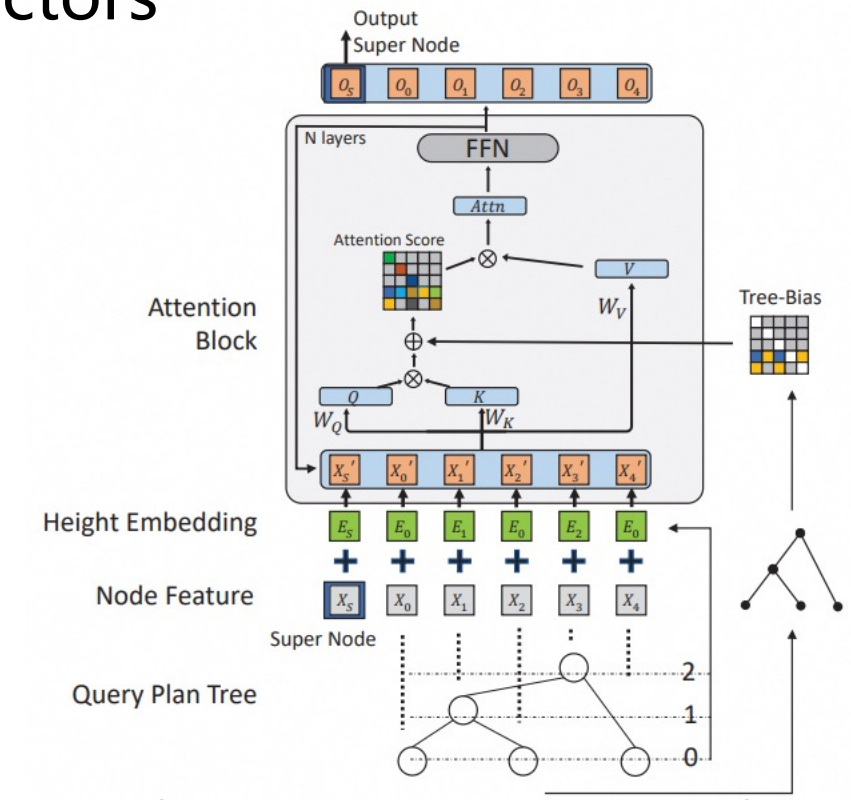


Figure from [QueryFormer: A Tree Transformer Model for Query Plan Representation, VLDB 2022]

(Saturn) Efficient Learning with Pseudo Labels for Query Cost Estimation, CIKM 2022

Cost Estimation for Plans of Concurrent Queries

- Challenges: queries have complex correlations
- Solutions: modeling the correlations using graphs
  - GPredictor: using graph neural networks for cost prediction

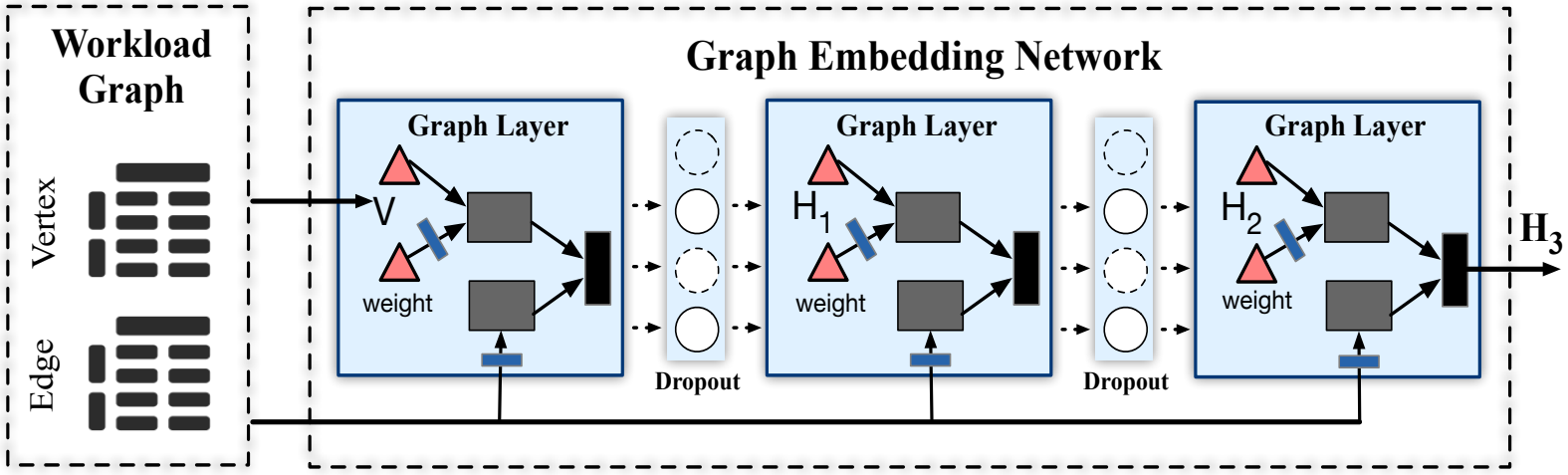


Figure from [Performance Prediction for Concurrent Queries using Graph Embedding, VLDB 2020]

Cost Estimation for Plans of Concurrent Queries

- Prestroid: apply tree convolutional networks for concurrent queries
- RAAL: end-to-end deep cost model with attention mechanism applied in cloud environment

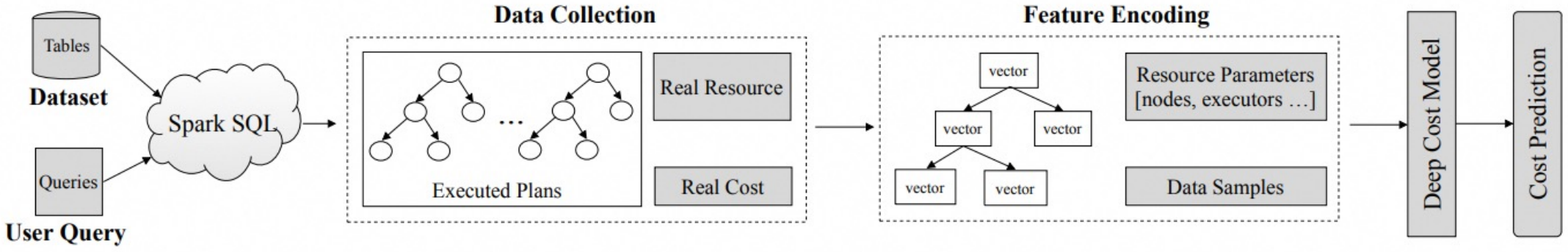


Figure from [A Resource-Aware Deep Cost Model for Big Data Query Processing, ICDE 2022]

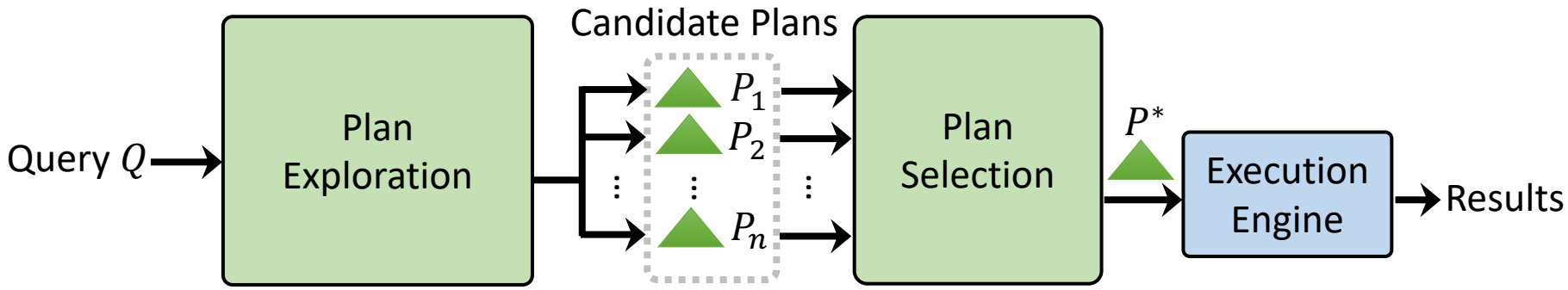
(Prestroid) Efficient Deep Learning Pipelines for Accurate Cost Estimations Over Large Scale Query Workload , SIGMOD 2021

Part 3: Learn to Explore Plan Space

- ❑ Search by Value Networks
- ❑ Search by Steering Existing Query Optimizers
- ❑ Search and Rank Plans
- ❑ Quality Control of Learned Query Optimizers

Plan Search in Query Optimizer

- The general framework



- Traditional plan exploration methods
  - Exact: Dynamic programming
  - Heuristic: Genetic or greedy search …

Part 3: Learn to Explore Plan Space

- ❑ Search by Value Networks
- ❑ Search by Steering Existing Query Optimizers
- ❑ Search and Rank Plans
- ❑ Quality Control of Learned Query Optimizers

Search by Value Networks

- Value network: Predicate the best possible plan P containing partial plan P' of query Q
  - Tree convolutional network or other models

$$\overbrace{[\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}]}^{P_0} \Rightarrow \overbrace{[\mathbf{A} \bowtie \mathbf{B}, \mathbf{C}, \mathbf{D}]}^{P_1} \Rightarrow \overbrace{[\mathbf{A} \bowtie \mathbf{B}, \mathbf{C} \bowtie \mathbf{D}]}^{P_2} \Rightarrow \overbrace{[(\mathbf{A} \bowtie \mathbf{B}) \bowtie (\mathbf{C} \bowtie \mathbf{D})]}^{P_3} \quad P_0, P_1, P_2 \subseteq P_3$$

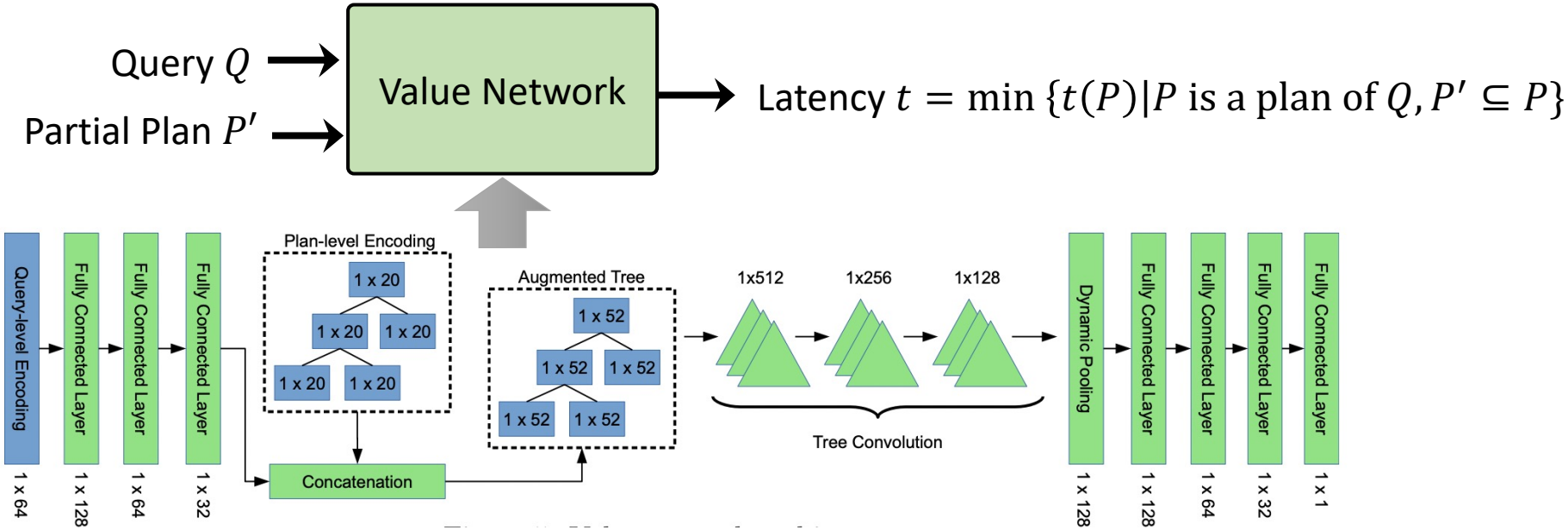
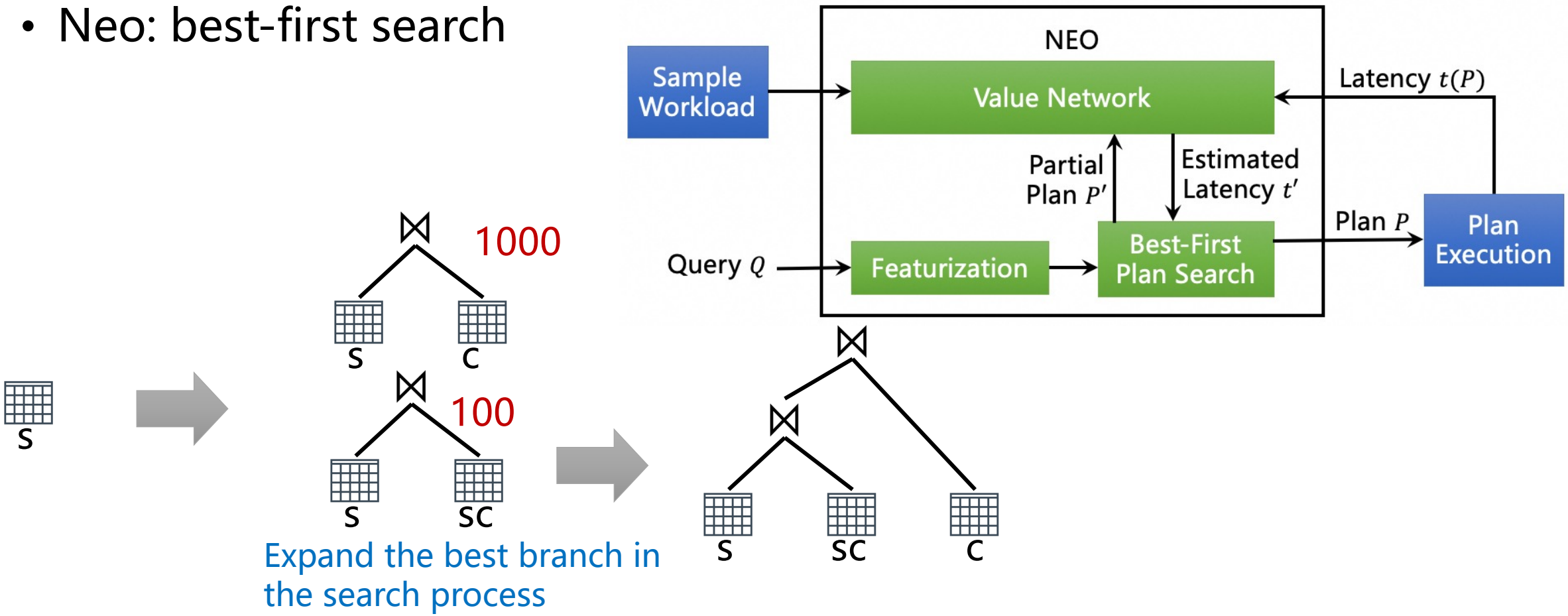


Figure from [Neo: A Learned Query Optimizer, VLDB 2019]

Search by Value Networks

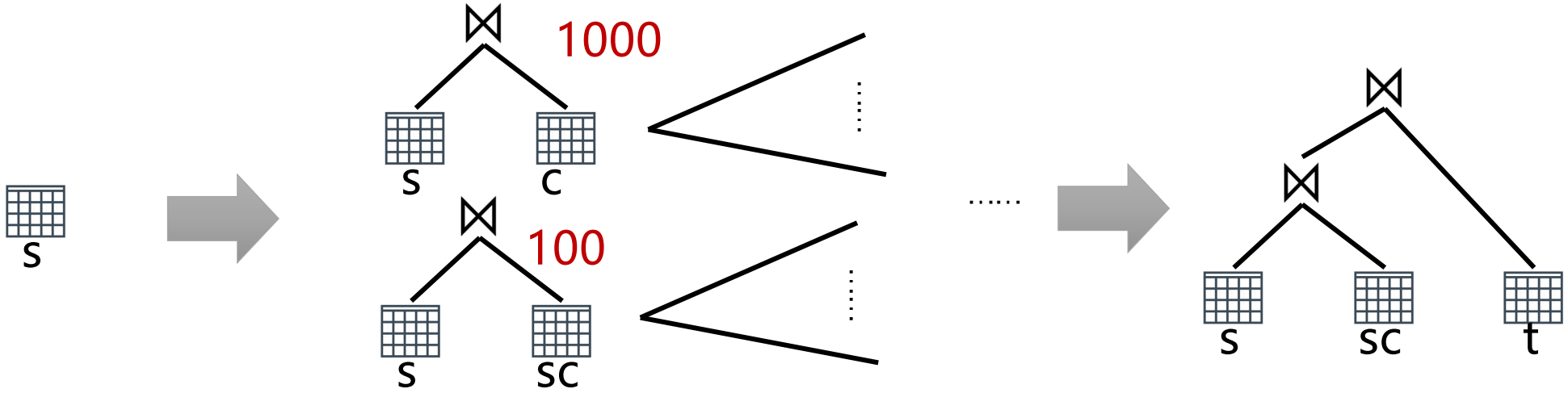
- Search plan space from scratch with value networks
  - Neo: best-first search



(Neo) Neo: A Learned Query Optimizer, VLDB 2019

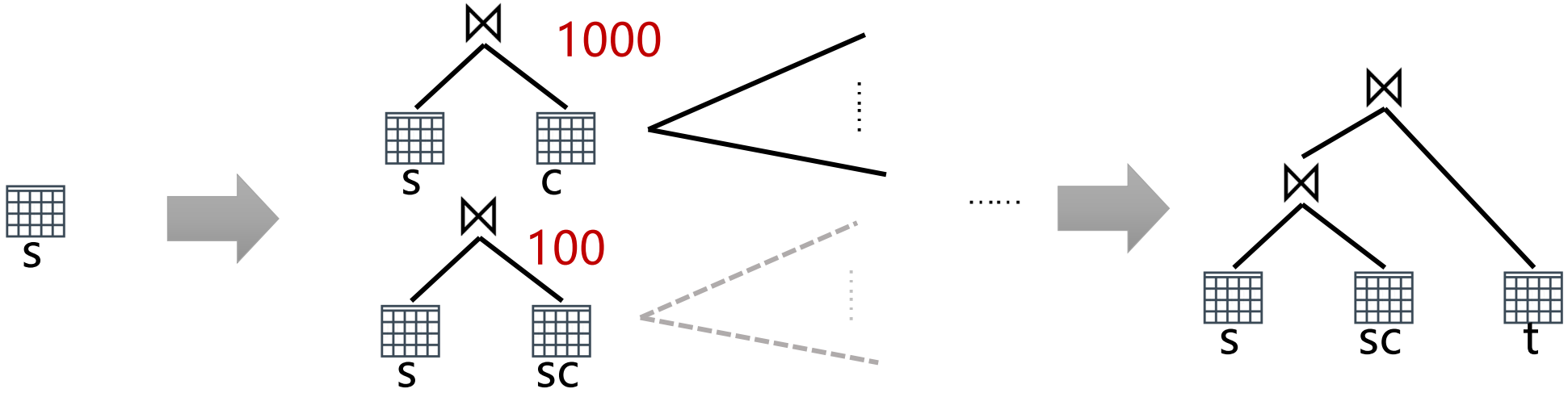
Search by Value Networks

- Search plan space from scratch with value networks
  - Balsa: beam search



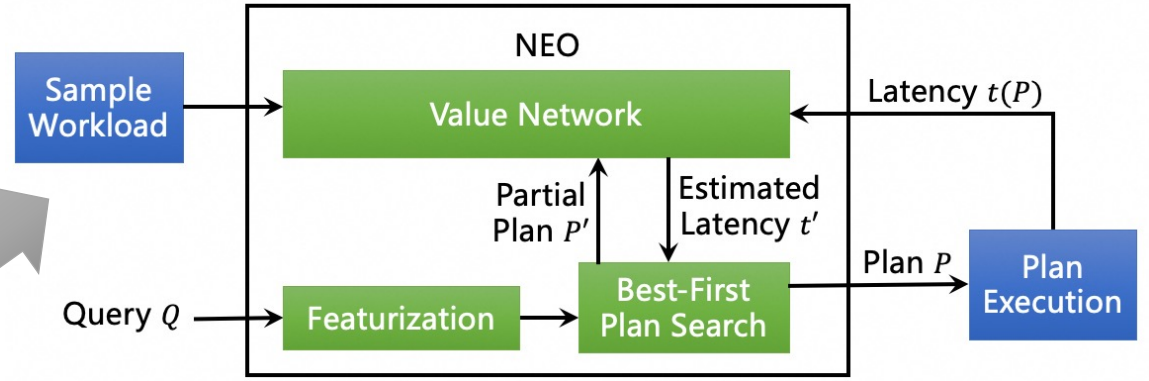
Search by Value Networks

- Search plan space from scratch with value networks
  - LOGER: ϵ-beam search



Expand the a random number (decided by k and ϵ) of branches in the search process

Search by Value Networks



- Model training
  - Exact label is difficult to obtain
  - Using the latency of the best plan we can obtain as training label
  - Update model with collected workload in periodical

- Shortcomings:
  - Cold-start
    - Balsa/BASE: Bootstrap with existing cost model
  - Large search space
    - Not effective to obtain truly good plans

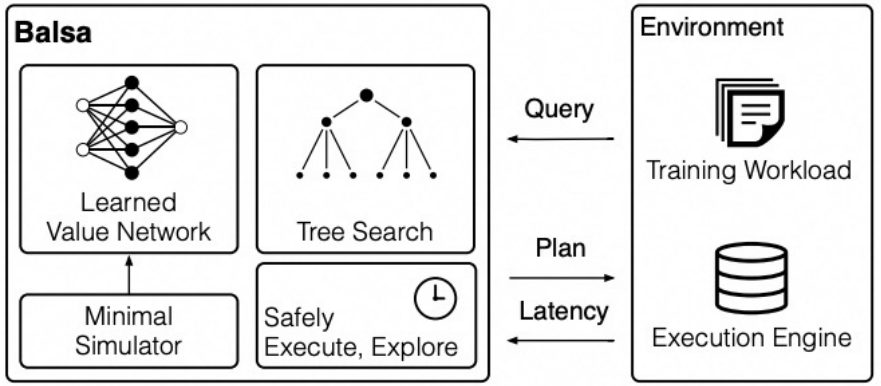


Figure from [Balsa: Learning a Query Optimizer Without Expert Demonstrations, SIGMOD 2022]

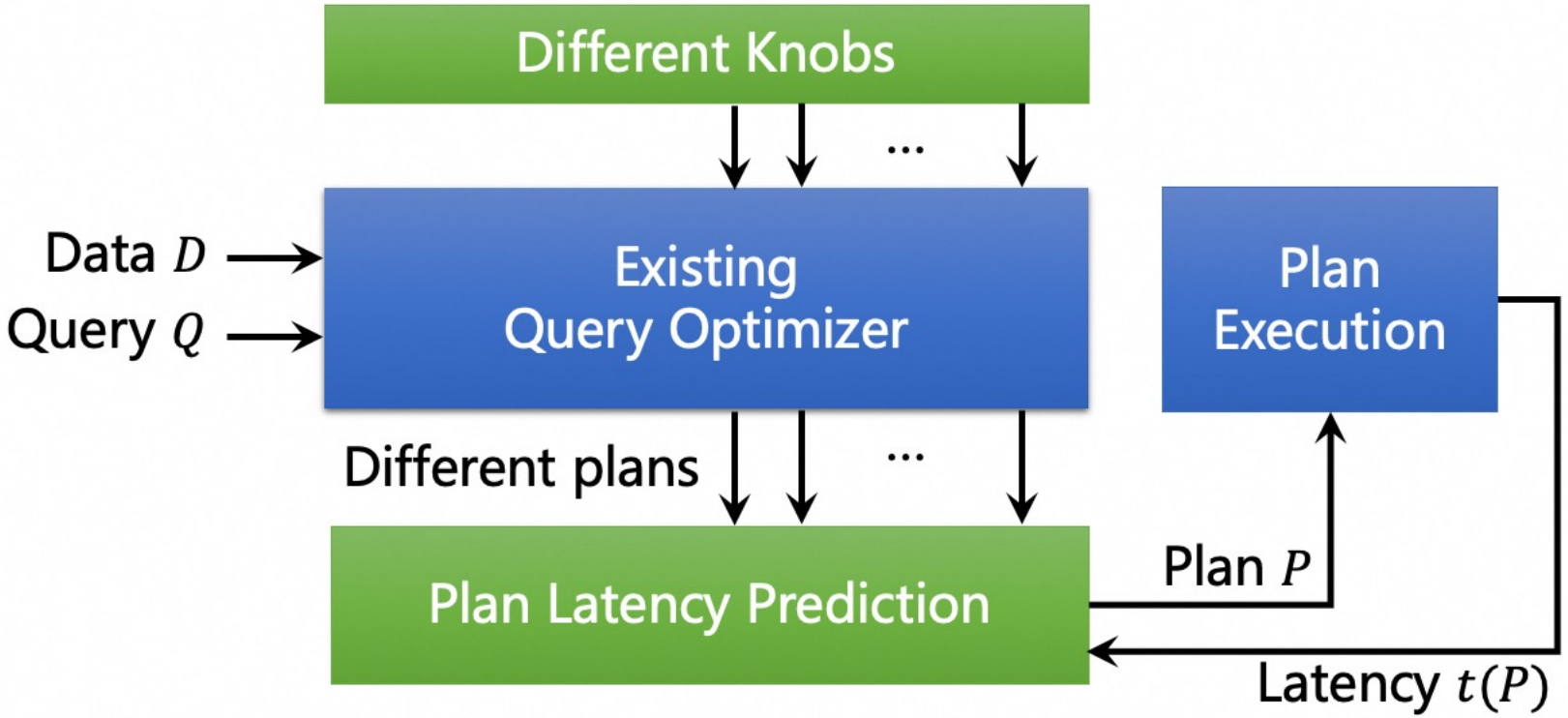
(BASE) BASE: Bridging the Gap between Cost and Latency for Query Optimization, VLDB 2023

Part 3: Learn to Explore Plan Space

- ❑ Search by Value Networks
- ❑ Search by Steering Existing Query Optimizers
- ❑ Search and Rank Plans
- ❑ Quality Control of Learned Query Optimizers

Search by Steering Existing Query Optimizer

- Tune the existing query optimizer to search different paths in the plan space
  - Reduce the search space
  - Easy for deployment



Different Tuning Knobs

- Bao: tune the hint set
  - Enable/Disable certain types of operations
    - Disable the nested loop join

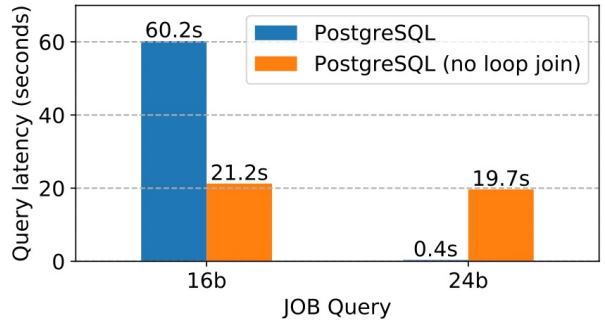


Figure from [Bao: Making Learned Query Optimization Practical, SIGMOD 2021]

- HyperQO: tune the leading hints on join order

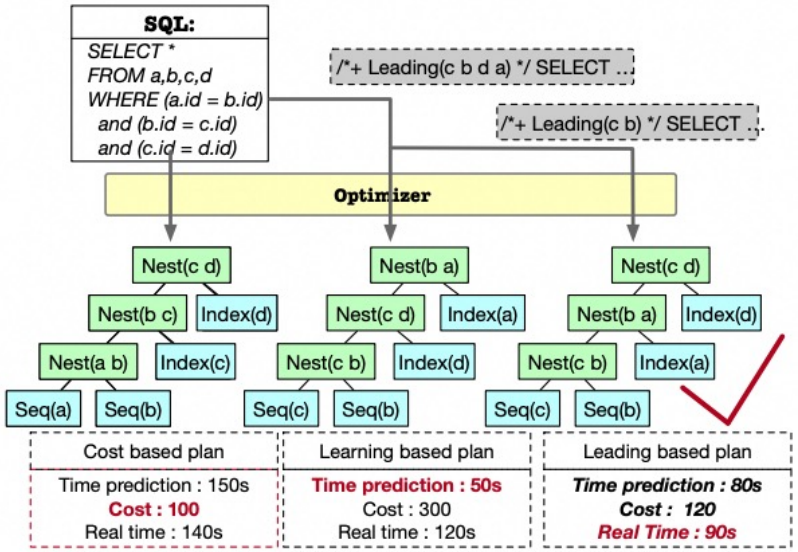
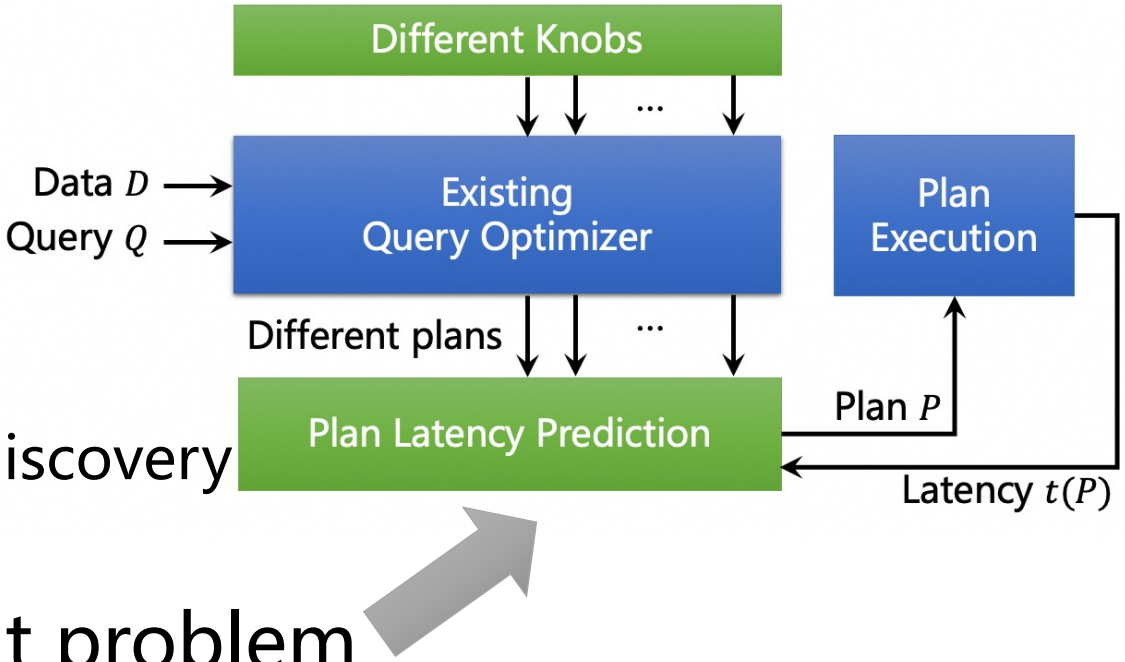


Figure from [Cost-based or Learning-based? A Hybrid Query Optimizer for Query Plan Selection, VLDB 2022]

Disadvantages



- Large space of knobs
  - h hints → 2^h possible hint sets
  - AutoSteer: Automated suitable hint-set discovery
- Latency prediction is a very difficult problem
  - Large learning space: data, query, system and other factors …
  - Select in-optimal plans
  - Slow model updating
  - …

(AutoSteer) AutoSteer: Learned Query Optimization for Any SQL Database, VLDB 2023

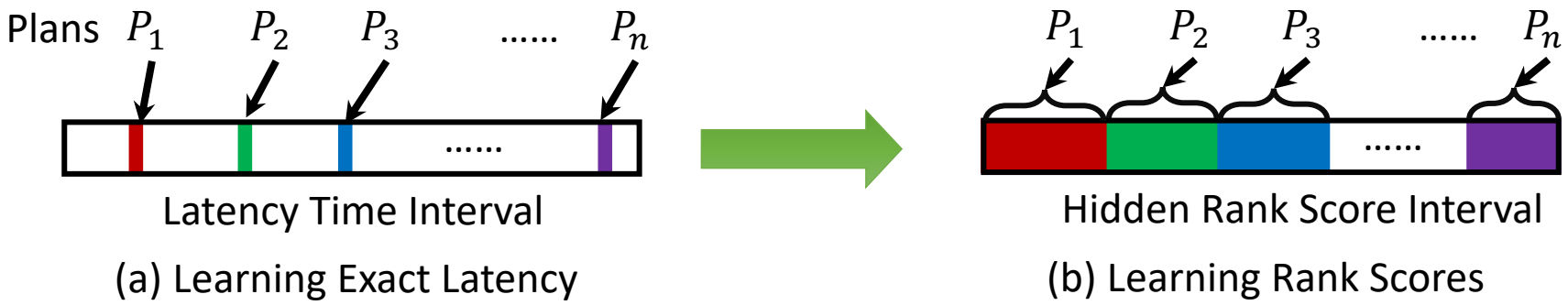
Part 3: Learn to Explore Plan Space

- ❑ Search by Value Networks
- ❑ Search by Steering Existing Query Optimizers
- ❑ Search and Rank Plans
- ❑ Performance Quality Control

Query Optimizer: A Better Solution

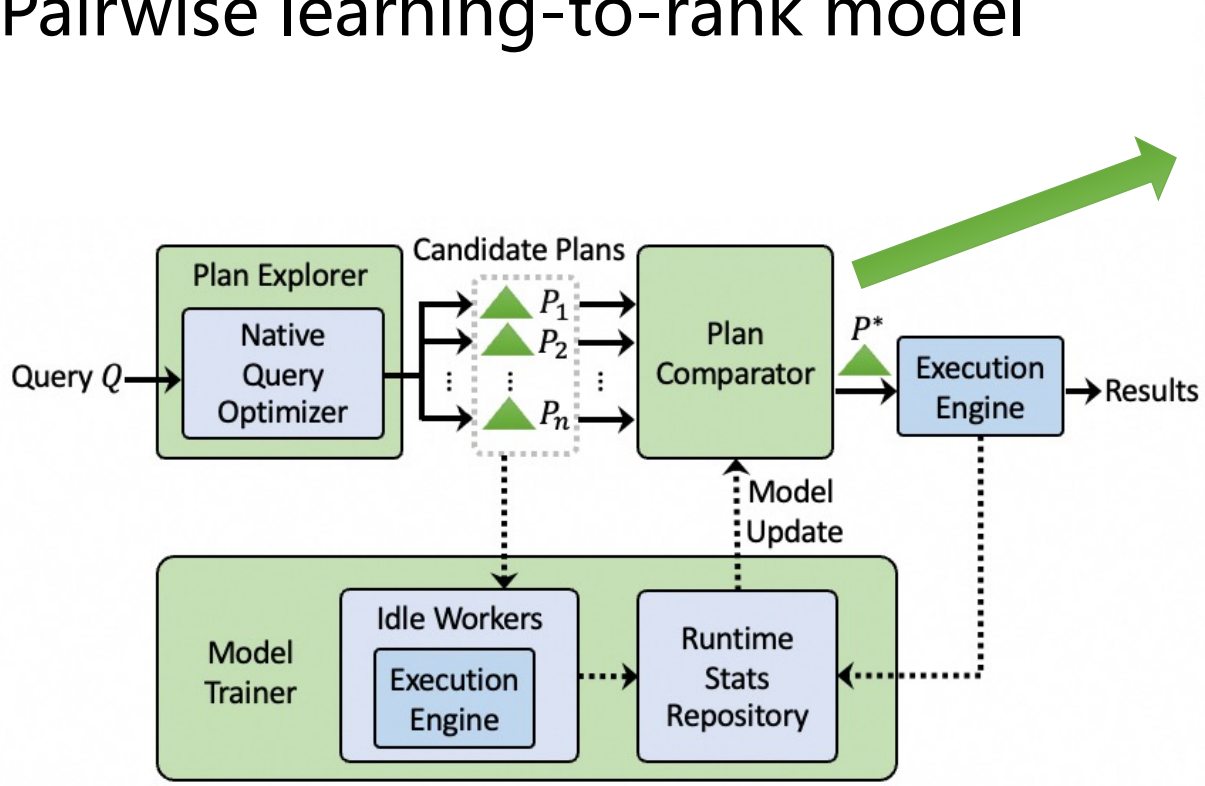
- Is it really necessary to know the exact time of each plan?
- No, we just need to know which plan is better!

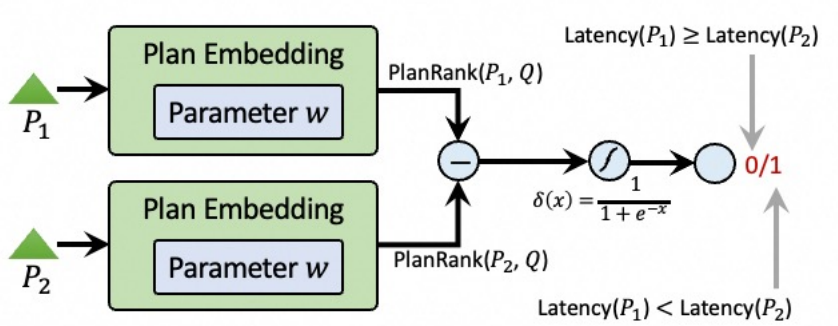
- From learning-to-predict to learning-to-rank
  - Low learning difficulty with more benefits
  - Accurate model, fast learning and updating…



Query Optimizer: Learn to Rank Plans

- Lero: learning-to-rank based query optimizer
  - Pairwise learning-to-rank model





Plan comparator $\mathbf{CmpPlan}(P_1, P_2)$

Commutativity

- $\mathbf{CmpPlan}(P_1, P_2) = 1 - \mathbf{CmpPlan}(P_2, P_1)$

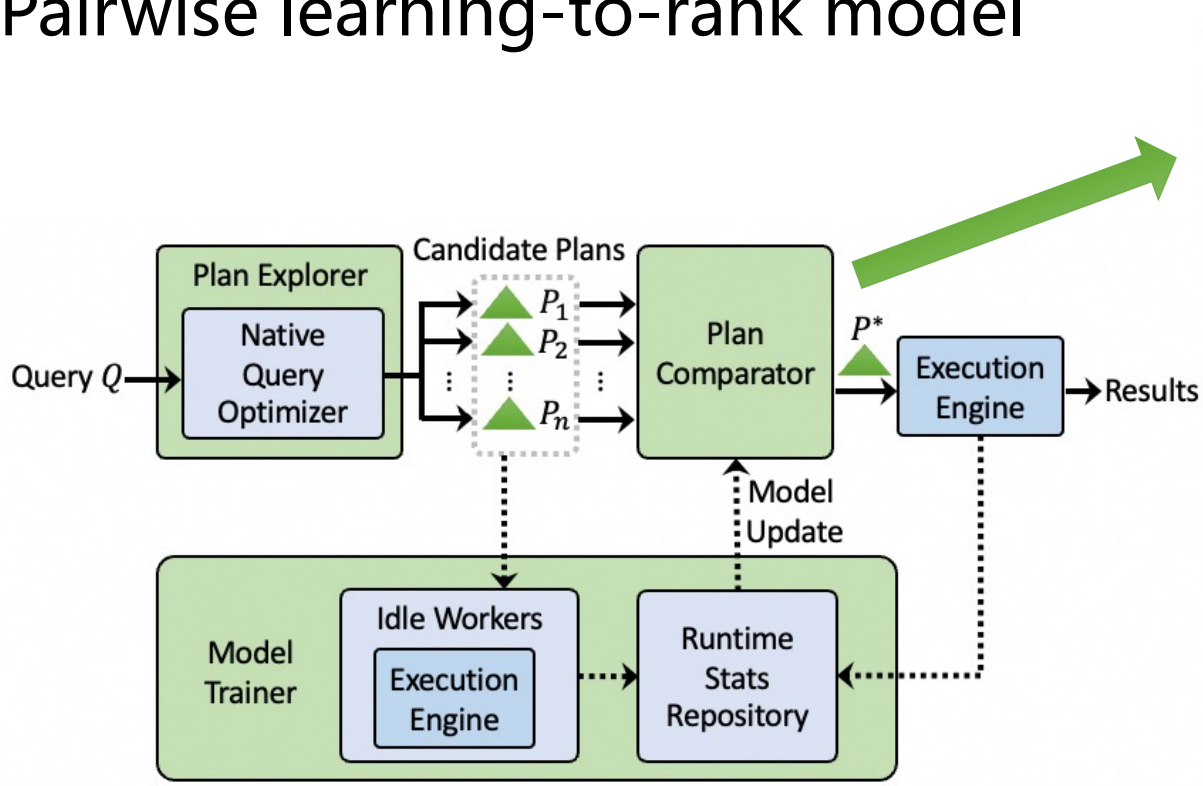
Transitivity

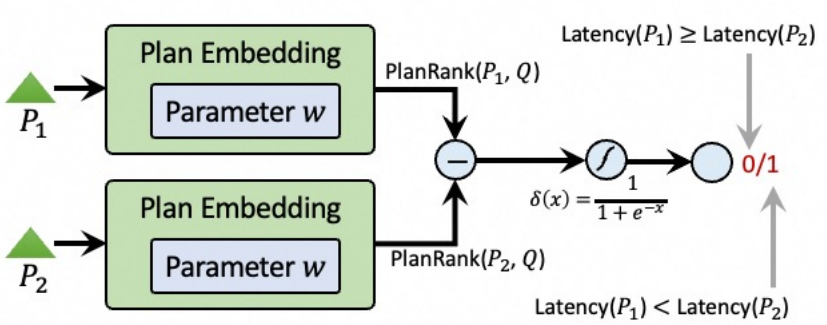
- $P_1 > P_2,\ P_2 > P_3 \Rightarrow P_1 > P_3$

(Lero) Lero: A Learning-to-Rank Query Optimizer, VLDB 2023

Query Optimizer: Learn to Rank Plans

- Lero: learning-to-rank based query optimizer
  - Pairwise learning-to-rank model



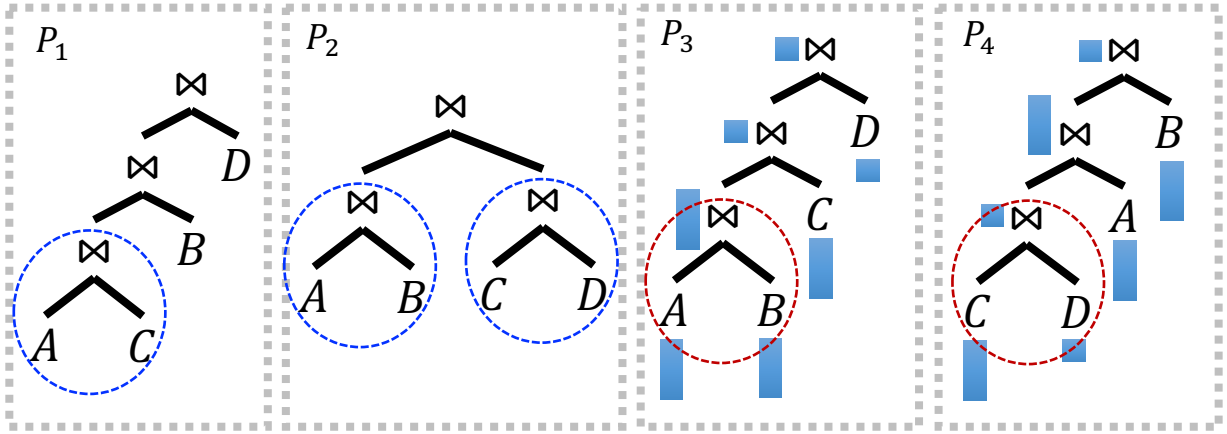


- Embedding each pair of plans with shared parameters
- 0/1 signal rather than exact time as training label → more robust
- pre-training the embedding model to align with the existing cost model → inheriting the wisdom, no cold-start

Query Optimizer: Learn to Rank Plans

- Plan exploration in Lero: tune cardinality of all sub-queries
  - Group sub-queries together
  - Scale up/down their cardinality → could find better one with theoretical analysis

- Plan diversity
  - Different plan shapes
  - Different join orders



Benchmark Evaluation

- Lero performs better than others
  - Static and dynamic settings

Table 1: Overall performance of different query optimizers.

Query Optimizer	Execution Time (in hour)			
	STATS	IMDB	TPC-H	TPC-DS
PostgreSQL	20.19	1.15	0.94	1.68
Bao	15.32	0.47	1.17	1.55
Bao+	13.85	0.41	0.89	1.57
Lero	**11.32**	**0.35**	**0.74**	**1.47**
Fastest Found Plan	**10.73**	**0.19**	**0.72**	**1.39**

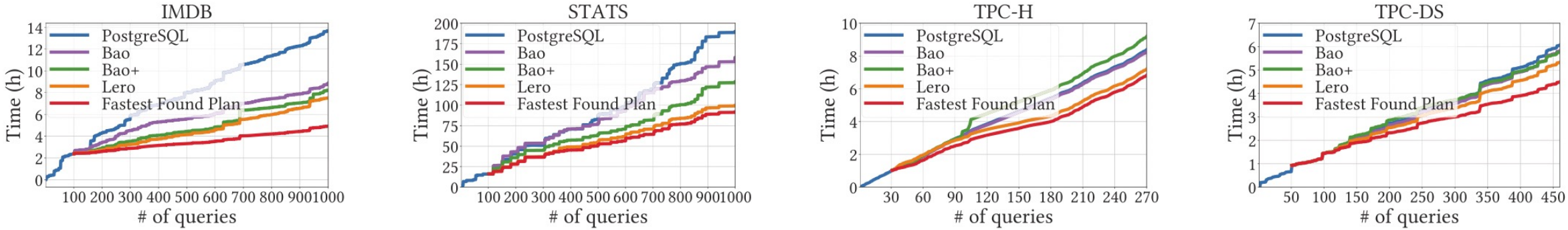


Figure 6: Performance curve of different query optimizers since deployment on benchmarks.

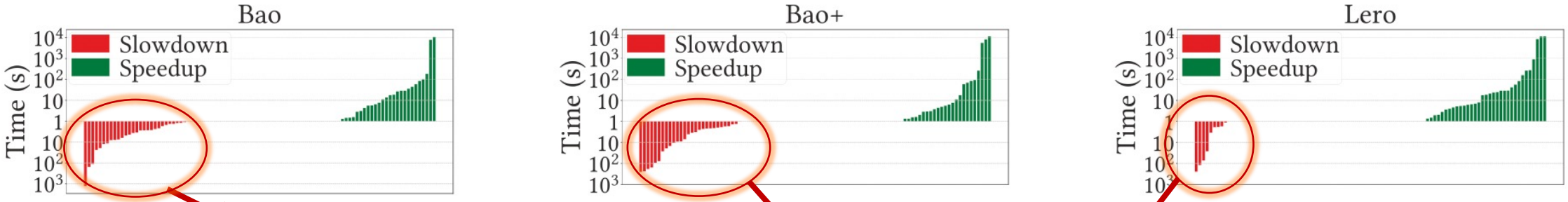


Figure 5: Per-query execution time of different query optimizers in comparison with PostgreSQL on the STATS benchmark.

Could we ultimately eliminate such performance regressions without affecting the benefits?

Experimental data from [Lero: A Learning-to-Rank Query Optimizer, VLDB 2023]

Part 3: Learn to Explore Plan Space

- ❑ Search by Value Networks
- ❑ Search by Steering Existing Query Optimizers
- ❑ Search and Rank Plans
- ❑ Quality Control of Learned Query Optimizers

Learned Query Optimizer: Quality Control

- HyperQO: an ensemble model to learn multiple prediction results of each plan
  - Plans having large variance are filtered before selection
  - Small variance ≠ accurate estimation
    - Bad plans may be reserved if all models make the same mistake

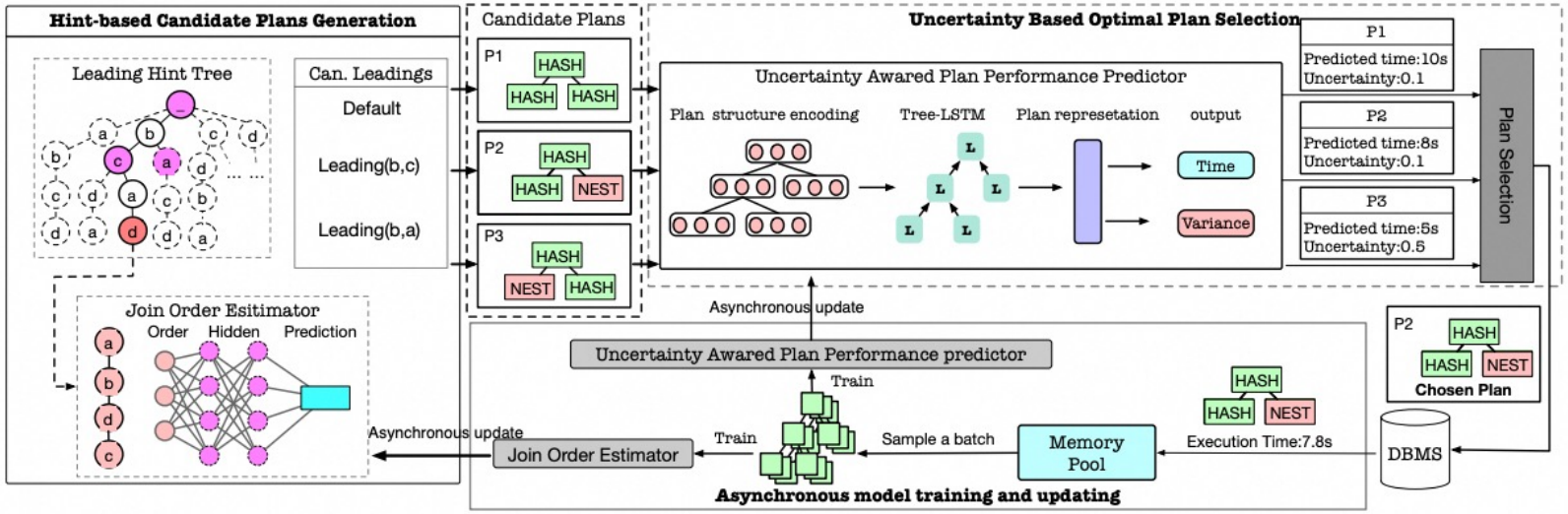


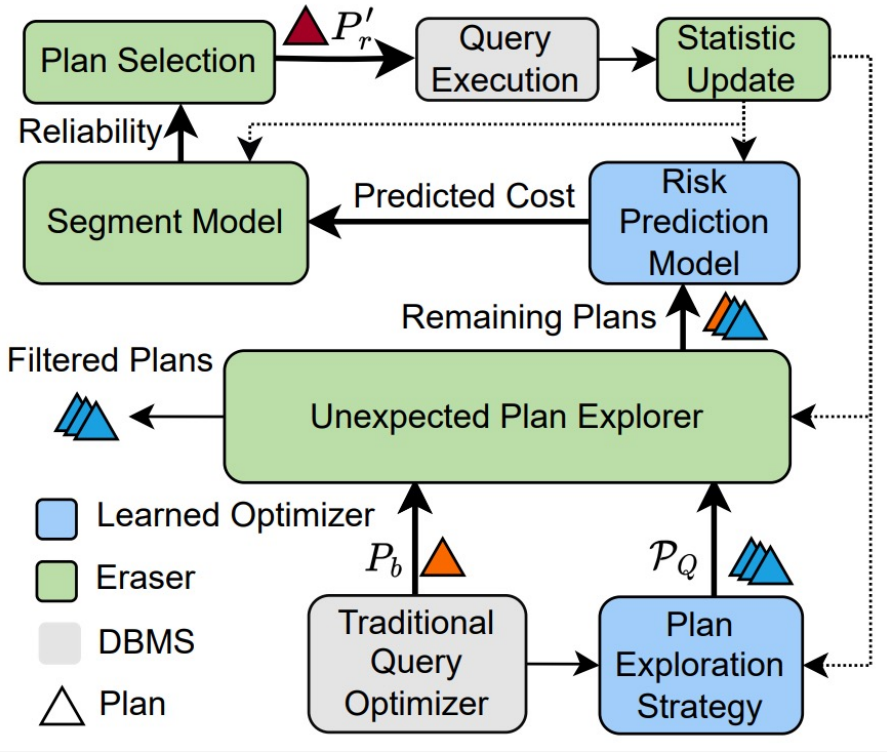
Figure from [Cost-based or Learning-based? A Hybrid Query Optimizer for Query Plan Selection, VLDB 2022]

Learned Query Optimizer: Quality Control

- Eraser: eliminate regressions while preserving benefits

- Unexpected plan explorer
  - A coarse-grained filter removing all highly risky plans with unseen feature values

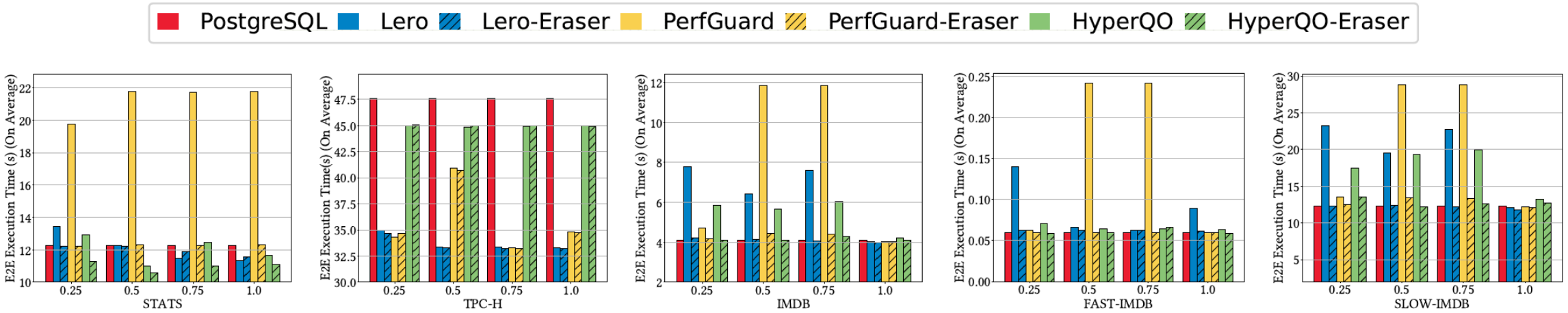
- Segment model
  - A fine-grained model to evaluate the quality of each plan for plan selection



(Eraser) Eraser: Eliminating Performance Regression on Learned Query Optimizer, VLDB 2024

Learned Query Optimizer: Quality Control

- Evaluation results of Eraser
  - Learned query optimizer performs worse than traditional one
    - Eraser could eliminate the performance regressions
  - Learned query optimizer performs better than traditional one
    - Eraser has very little impact on the performance improvements



Experimental data from [Eraser: Eliminating Performance Regression on Learned Query Optimizer, VLDB 2024]

Part 4: Applications and Deployment

- Prototype Applications
  - Learned cost model for SCOPE in Microsoft
  - Learn to steer SCOPE in Microsoft
  - OpenGauss

- PilotScope: A Middleware System for Deployment

Part 4: Applications and Deployment

- ❑ Prototype Applications
  - ❑ Learned cost model for SCOPE in Microsoft
  - ❑ Learn to steer SCOPE in Microsoft
  - ❑ OpenGauss

- ❑ PilotScope: A Middleware System for Deployment

Learned Cost Model in SCOPE

- SCOPE: the big data system for data analytics in Microsoft
- Learned cost models could improve plan quality of production jobs

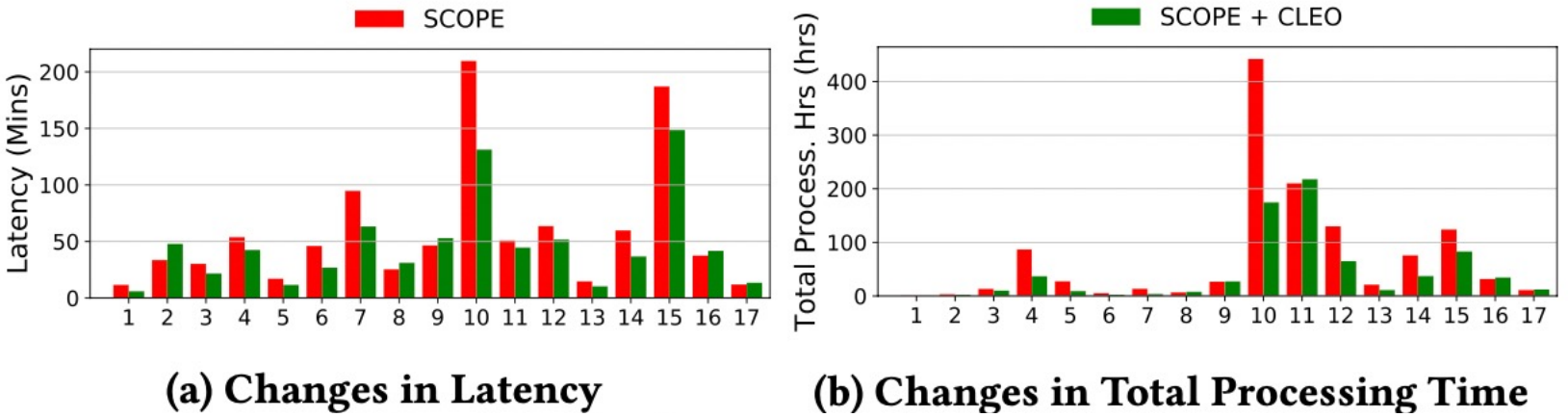


Figure from [Cost Models for Big Data Query Processing: Learning, Retrofitting, and Our Findings, SIGMOD 2020]

SCOPE and Its Hints

- SCOPE: tuning different hints
  - 256 hints in total, 100-150 hints are used frequently
  - Typically 10 – 20 hints are used in a single job
  - Analyze to discover suitable hint set to steer SCOPE

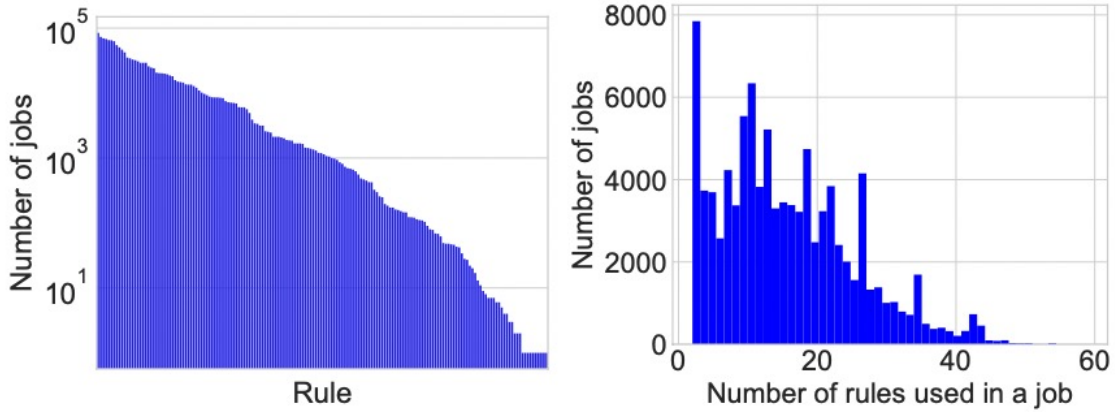
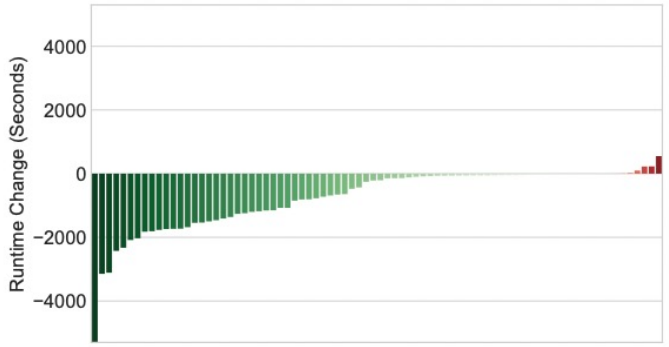


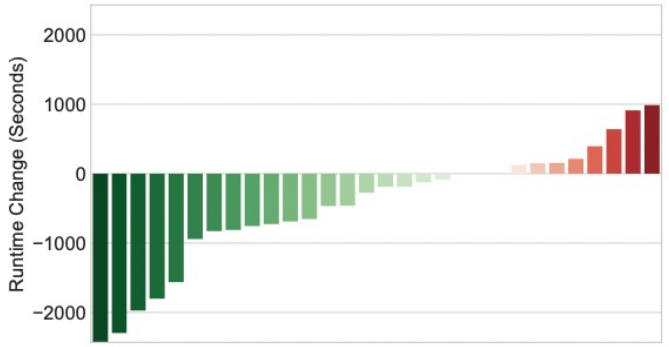
Figure from [Steering Query Optimizers: A Practical Take on Big Data Workloads, SIGMOD 2021]

Learn to Steer SCOPE

- Online deployment: large improvements, small regressions

	1			2			3		
	Mean	90P	99P	Mean	90P	99P	Mean	90P	99P
Best	5458	14K	14.8K	19.8K	26K	27K	2966	13.8K	15.3K
Default	6461	16.3K	18.3K	20.7K	26.9K	28.9K	3304	14.7K	16.8K
Learned	**5724**	**14.7K**	**15.4K**	**20.2K**	**26.2K**	**27K**	**3252**	**14.6K**	16.8K





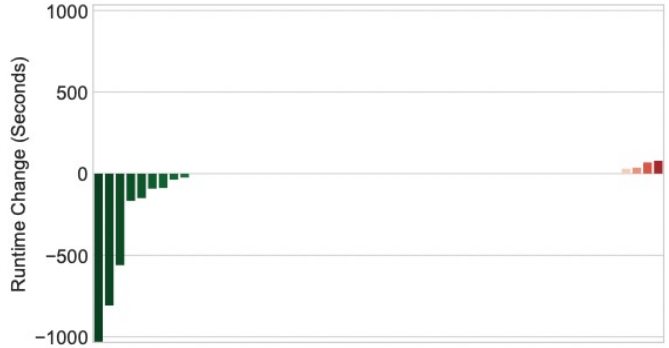


Figure and Table from [Steering Query Optimizers: A Practical Take on Big Data Workloads, SIGMOD 2021]

OpenGauss: Automatic DB

- Multiple learned components are deployed into GaussDB
  - Query rewriter
  - Cardinality/Cost Estimator
  - Plan Enumerator
  - …
- Traditional Methods as backups
- Difficult to extend to other AI4DB methods and DB systems

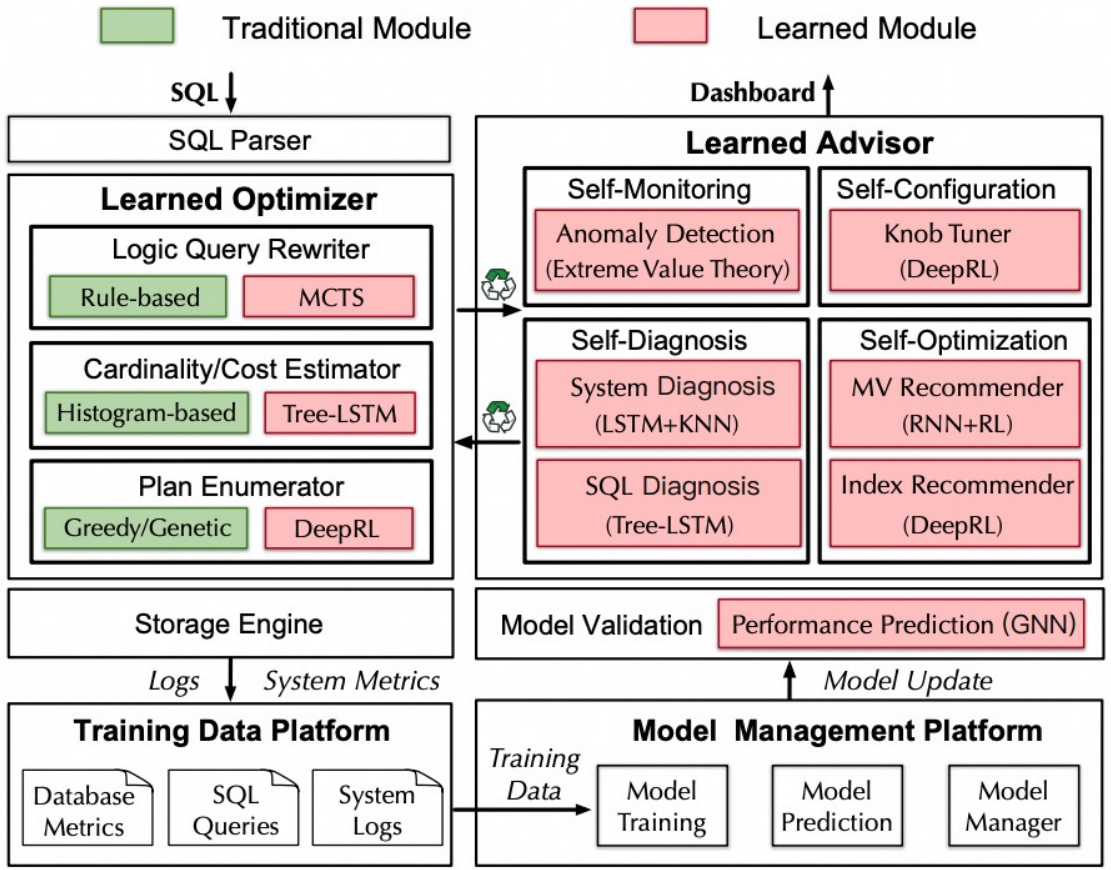


Figure from [openGauss: An Autonomous Database System, VLDB 2021]

Part 4: Applications and Deployment

- Prototype Applications
  - Learned cost model for SCOPE in Microsoft
  - Learn to steer SCOPE
  - OpenGauss

- PilotScope: A Middleware System for Deployment

PilotScope: Motivations

- AI4DB algorithms are developing very fast ……

- A fundamental problem is still not well studied
  - Could they actual contribute to real world DBMS?
    - If so, to what extent?
    - If not, why not?
  - This could only be answered by actual deployment (gold standard)
    - Examine in real systems, on real data, report real performance
    - The pros & cons of each method

  ……

  → guide future research towards the right direction

Cost Model

Query Optimizer

Knob Tuning

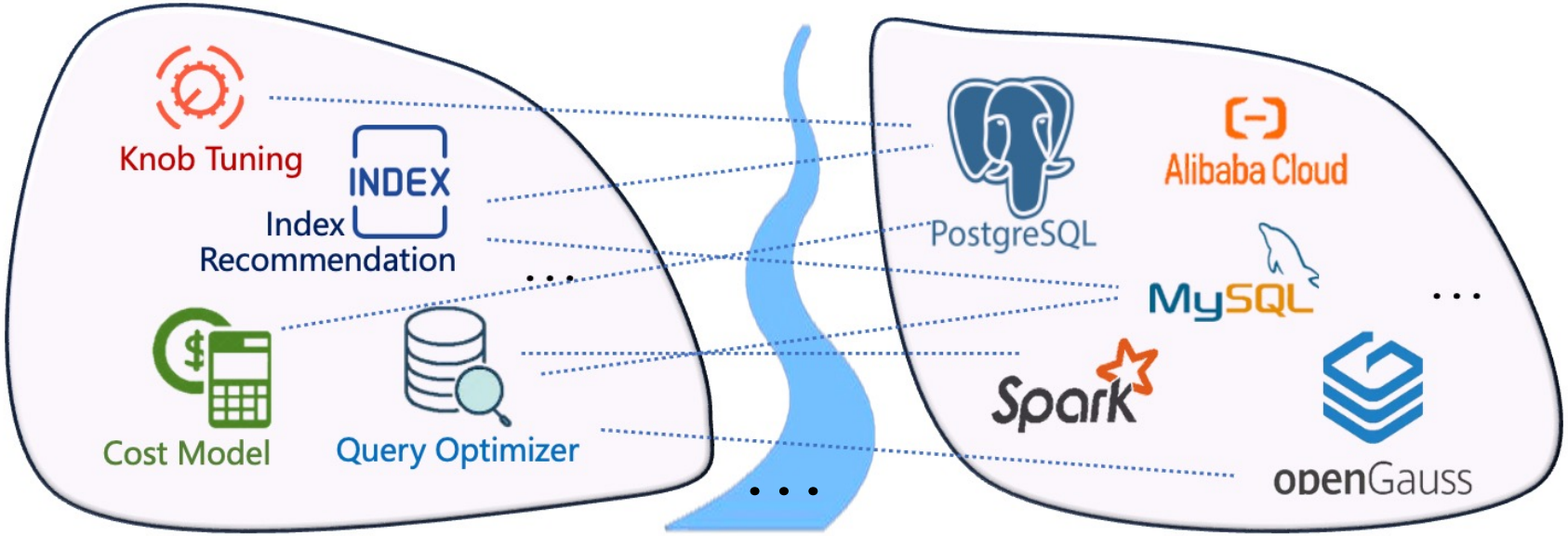
…

Index Recommendation

PilotScope: Existing Solutions

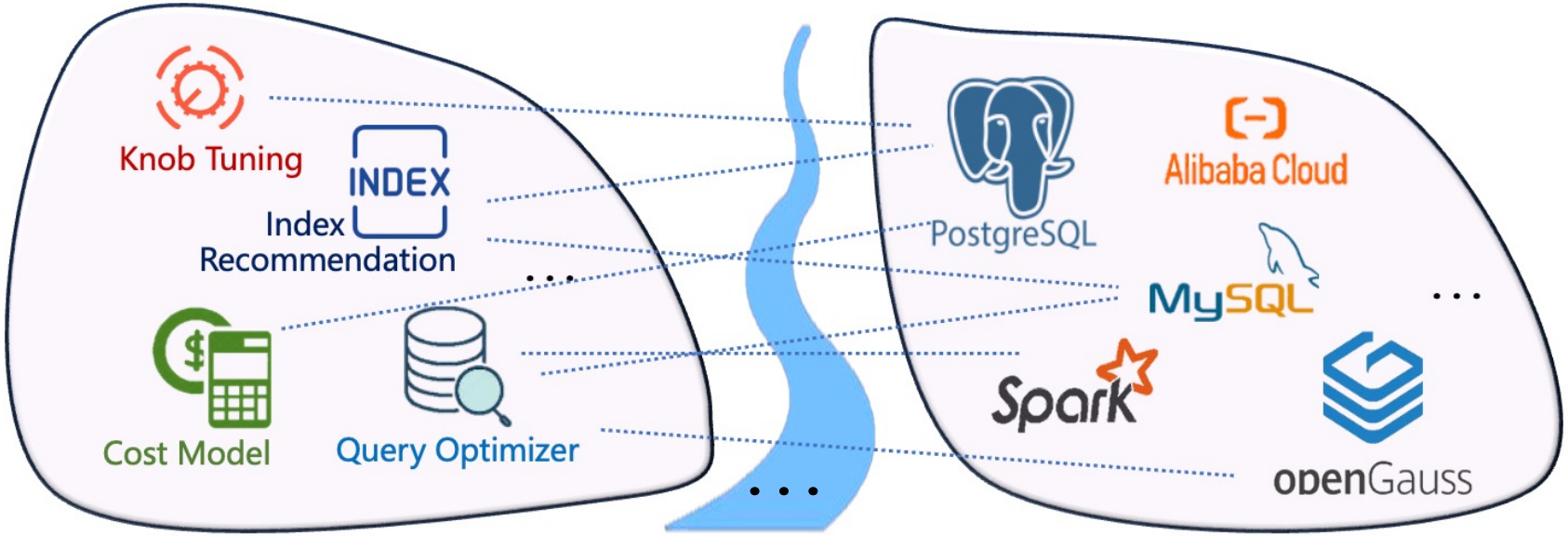
- Deploying specific algorithm into specific DB
  - Learned query optimizer/cost model → SCOPE
  - Knob tuning/query optimizer → openGauss
  - Workload forecasting/behavior modeling → NoisePage

  ……



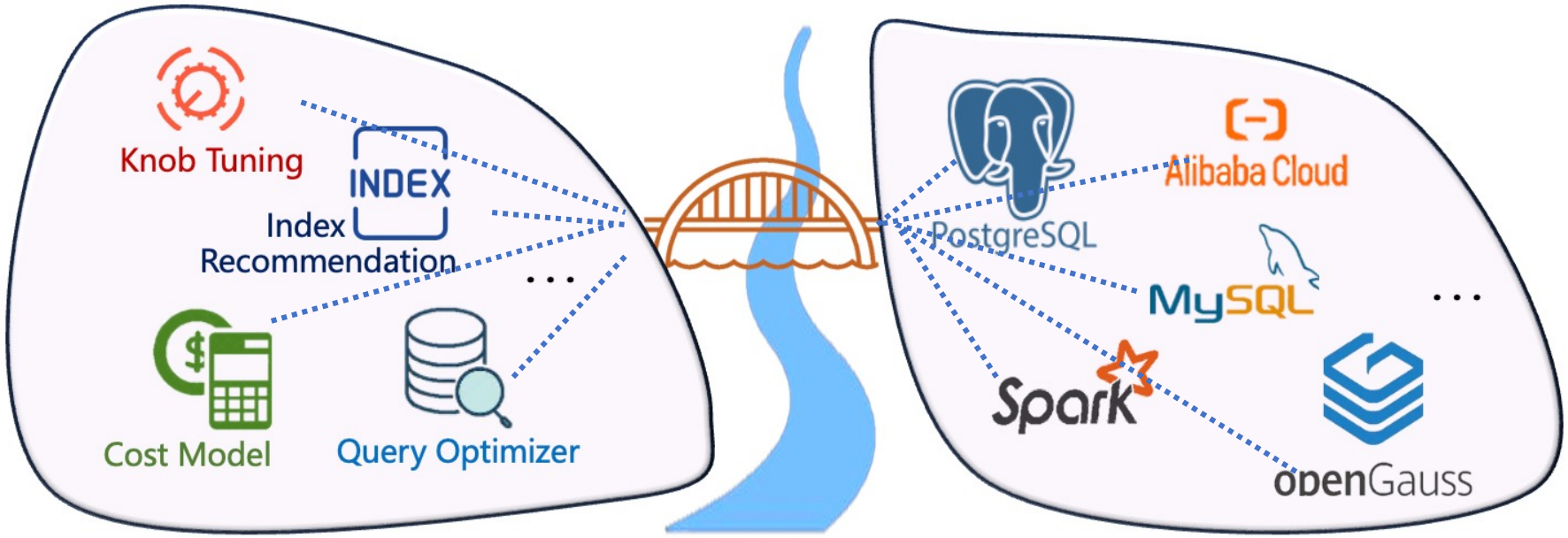
PilotScope: Existing Solutions

- Limitations of existing solutions
  - Not general to other AI4DB algorithms & databases
  - Very expensive: $O(m * n)$ tailored solutions for m algorithms & n databases
  - Not friendly to developers
    - Require close cooperation and learning between ML and DB developers



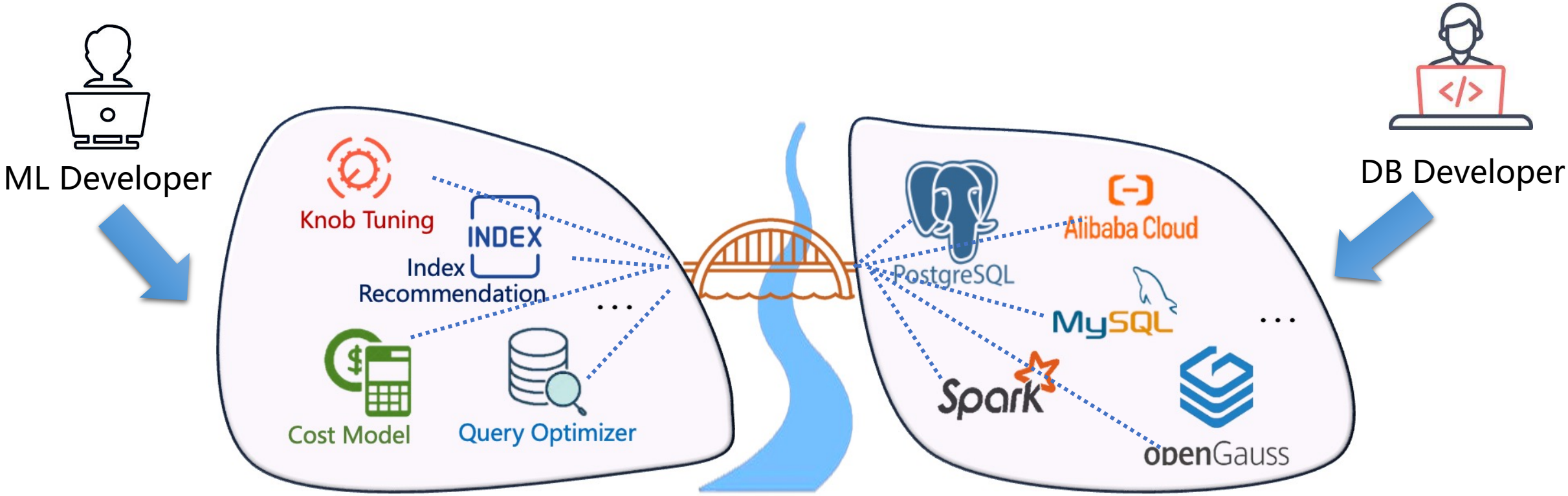
PilotScope: Our Principle

- What we desire of the AI4DB deployment?
  - One system generally appliable for multiple algorithms & databases
    - General to algorithm: support different AI4DB tasks
    - General to system: the same ML program could run on different systems
    - Easy to extensible to other AI4DB algorithms/database systems



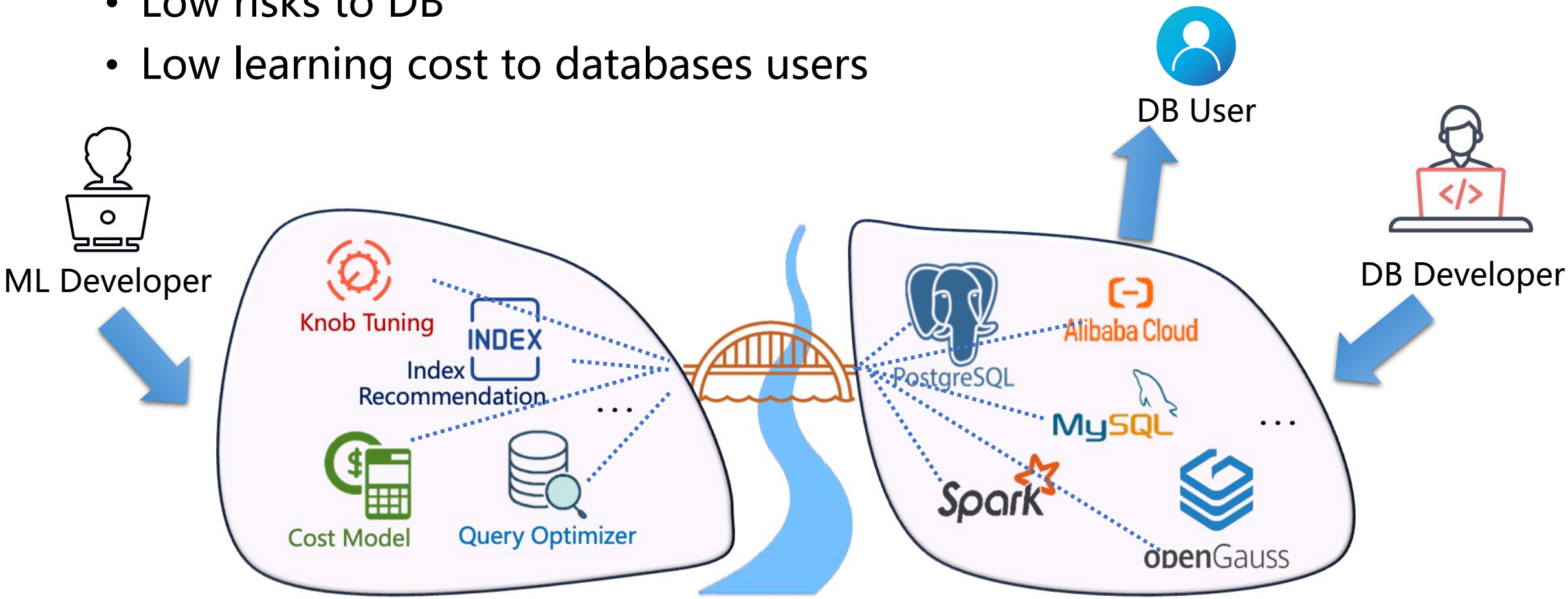
PilotScope: Our Principle

- What we desire of the AI4DB deployment?
  - ML & DB developers could work independently
    - Developers could focus on writing their own programs
    - Not need to know details in other side



PilotScope: Our Principle

- What we desire of the AI4DB deployment?
  - The minimal impact to the native databases
    - Low risks to DB
    - Low learning cost to databases users



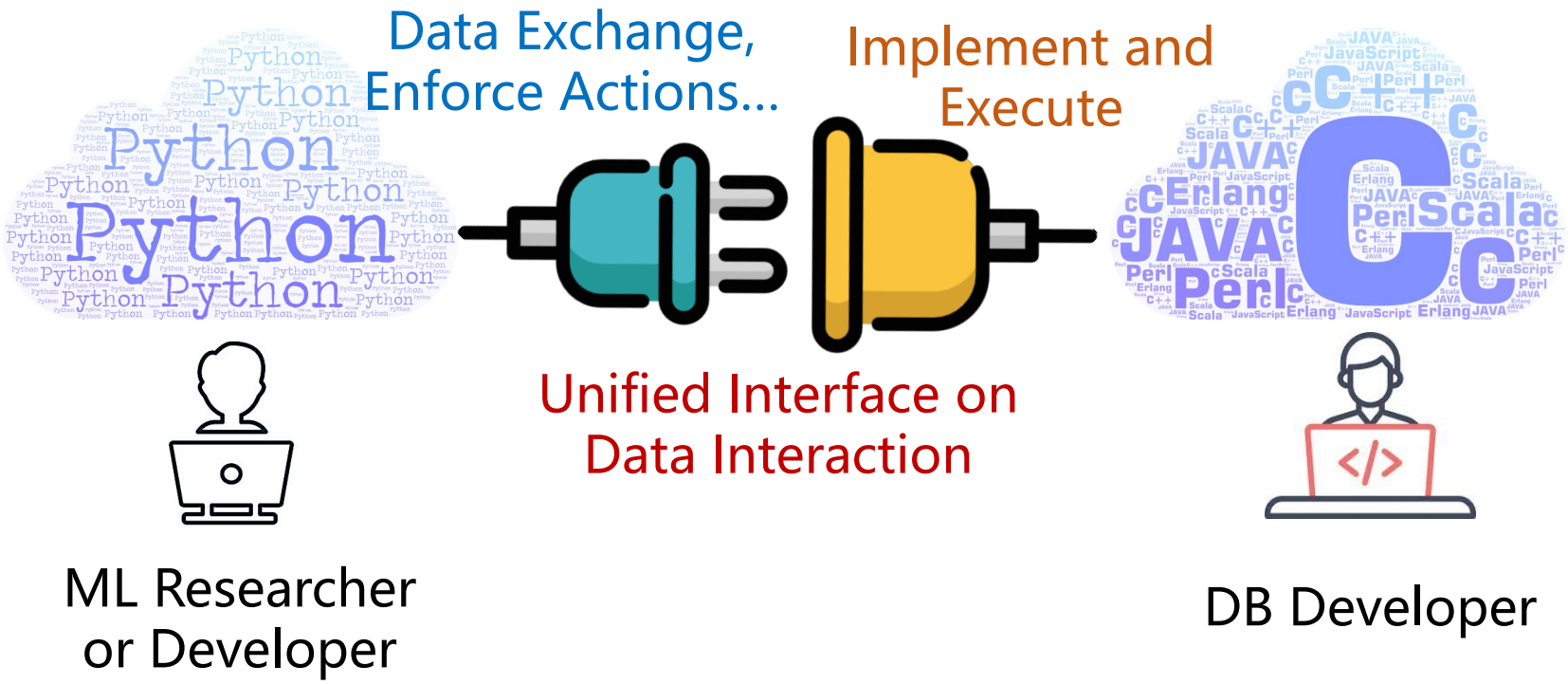
PilotScope: Challenges

- A desirable deployment system is very challenging
- ML and DB have very different paradigms: language, dependency, …
- AI4DB algorithms are very diversified
  - An AI4DB task usually involves a long pipeline on data interaction
- Database systems are very complex
  - Difficult and risky to modify the codebase

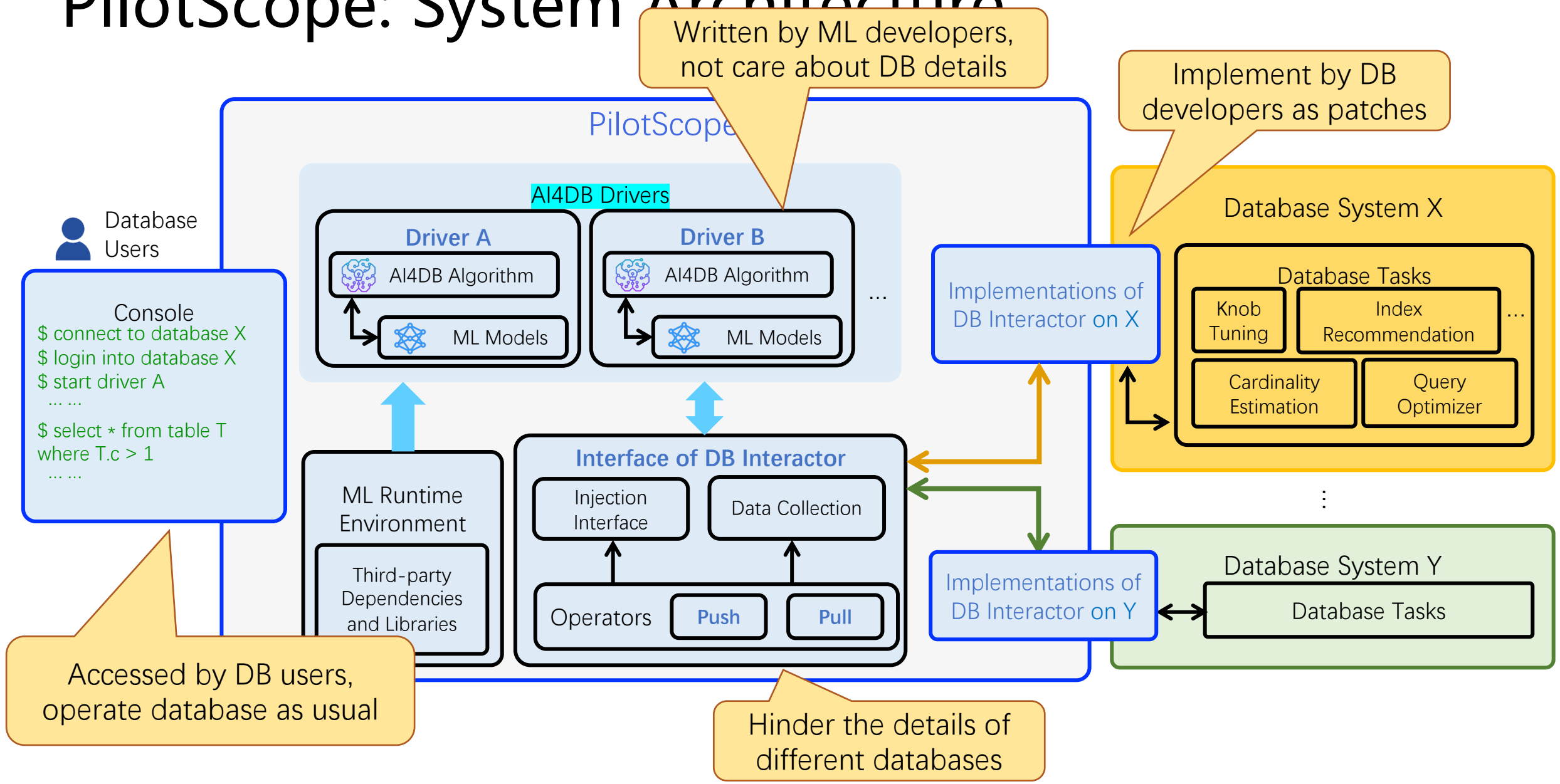
  ……

PilotScope: Framework

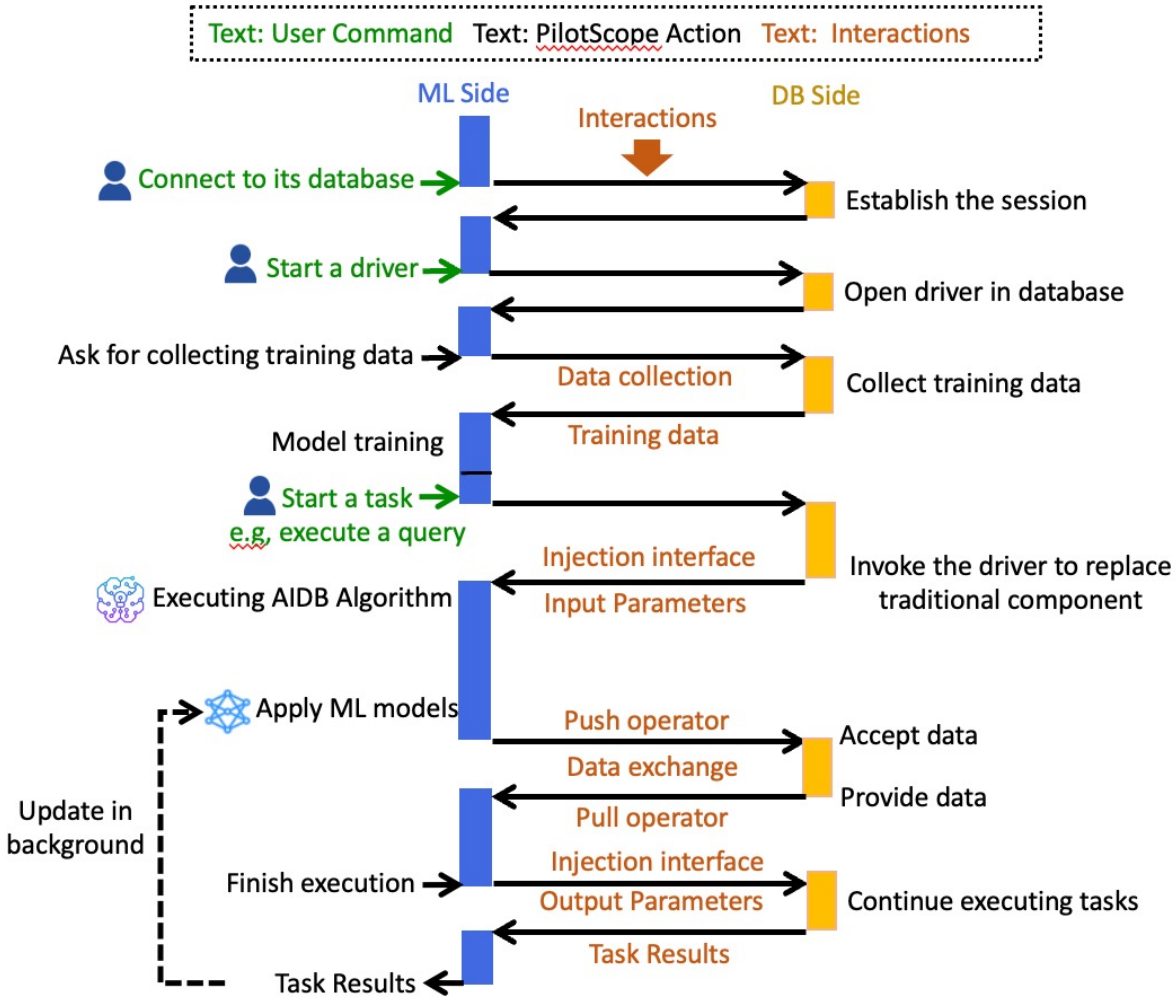
- How to bridge the gap between AI and DB?

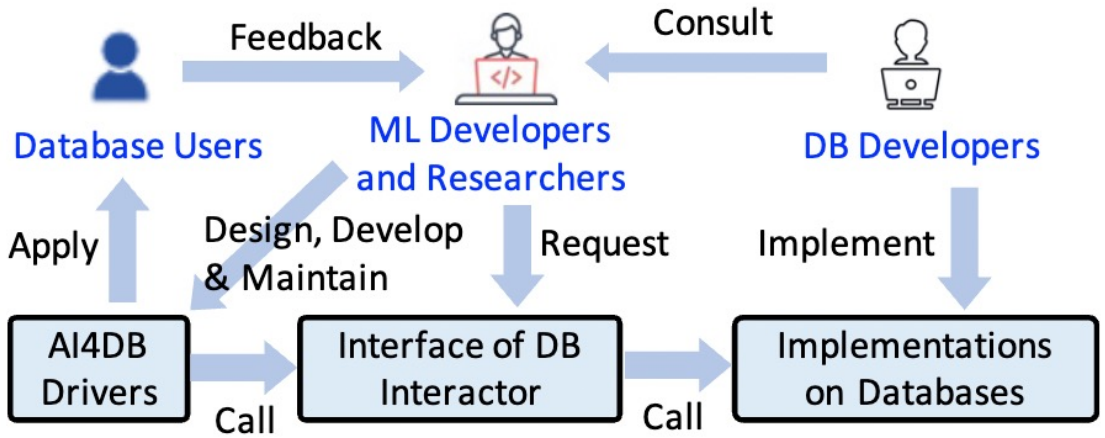


PilotScope: System Architecture



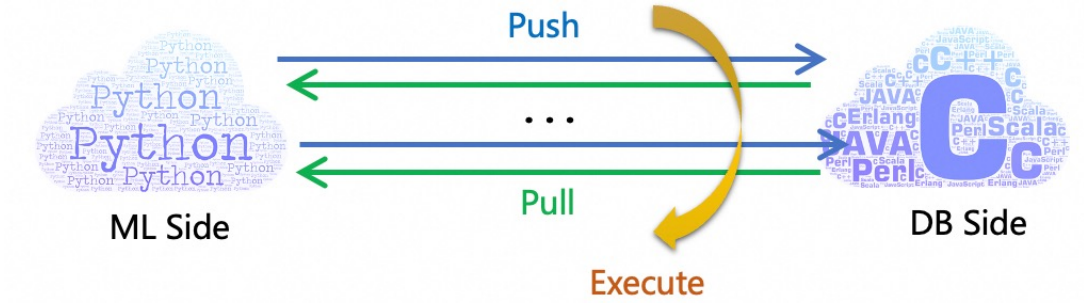
PilotScope: Workflow





PilotScope: DB Interactors

- Flexible APIs
  - Push: send data to DB
  - Pull: fetch data from DB
  - Any combination in a session
  - Execute: trigger execution

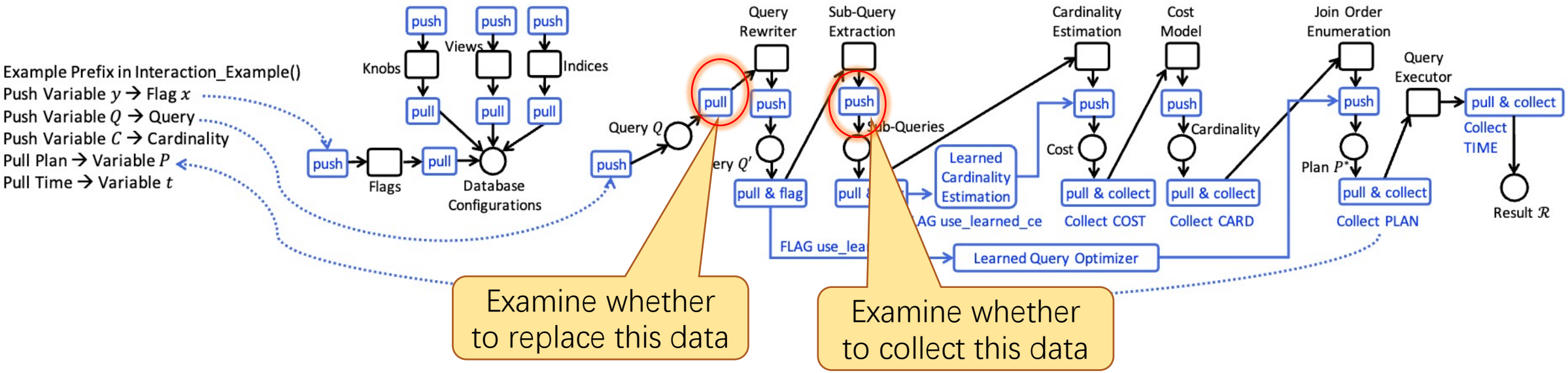


- Supported data types
  - Knob/hint/index
  - Sub-query/cardinality/cost
  - Plan/Latency
  - …

```
def test_all_data_interactor(self):
    knob = {"max_parallel_workers_per_gather": 0}
    hint = {"enable_hashjoin": "off", "enable_mergejoin": "on"}
    query = ("select count(*) from posts as p, postlinks as pl, posthistory "
             "as ph where p.id = pl.postid and pl.postid = ph.postid and "
             "p.creationdate>=1279570117 and ph.creationdate>=1279585800 "
             "and p.score < 50;")
    data_interactor = PilotDataInteractor(self.config)
    data_interactor.push_hint(hint)
    data_interactor.push_knob(knob)
    data_interactor.pull_physical_plan()
    data_interactor.pull_subquery_card()
    data_interactor.pull_estimated_cost()
    data_interactor.pull_record()
    data_interactor.execute(query)
```

PilotScope: Implementation

- APIs in the interface: patches before/after each component
  - Built-in tools: commands/hooks, …



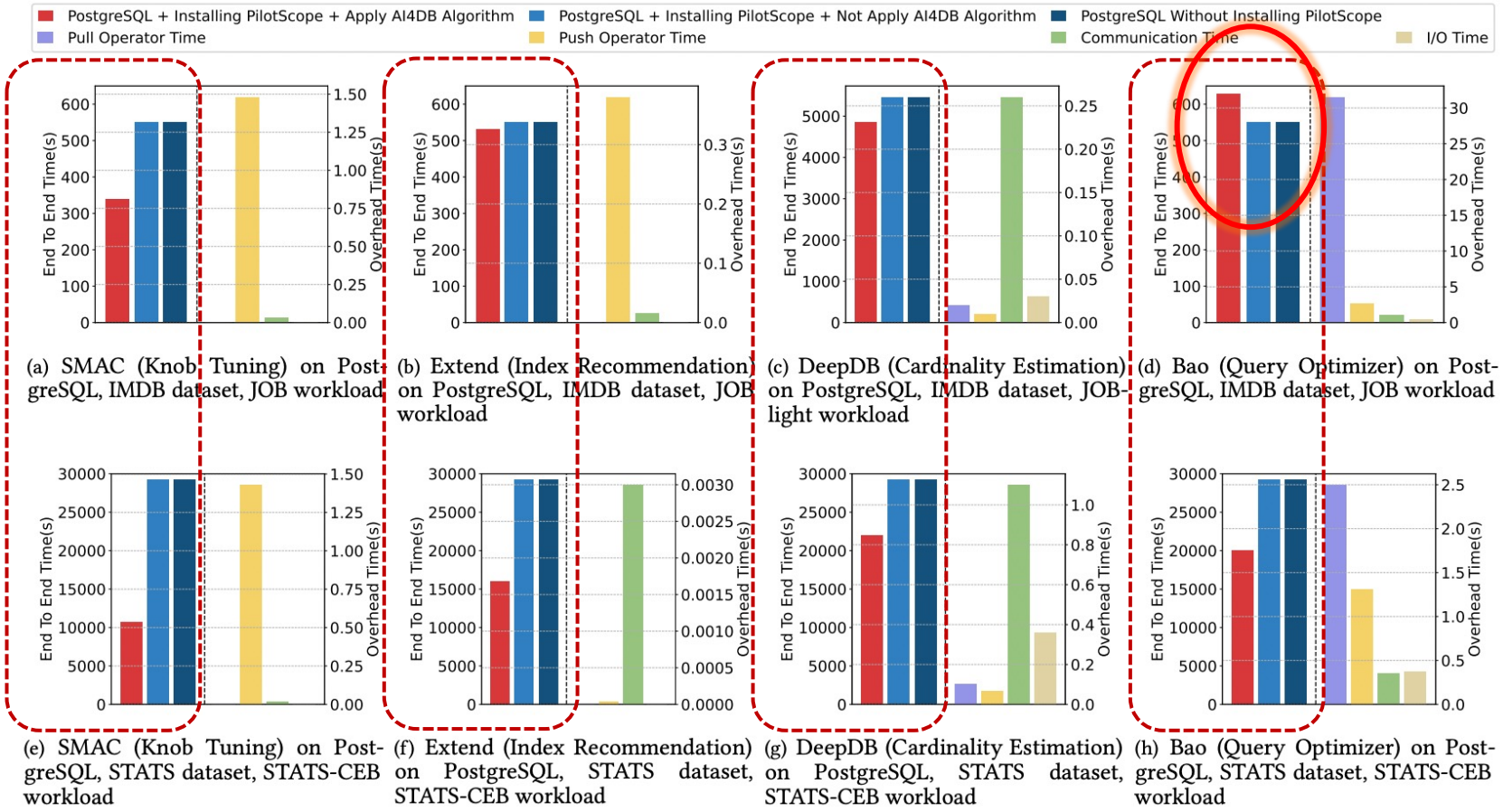
PilotScope: Applications

- 4 AI4DB tasks, 15+ algorithms on PostgreSQL and Spark
- Easy to extend to similar tasks: view advisor, learned cost model…

Task	Algorithm	Description
Knob Tuning	SMAC [26] GP-BO [44]	Bayesian Optimization
	DDPG [56]	Reinforcement Learning
Index Recommendation	AutoAdmin [17] DB2Advisor [49] Extend [45] DTA [1]	Enlarging Small Set
	Drop [51] Relaxation [14]	Reducing Large Set
	Cophy [20]	Linear Programming
Cardinality Estimation	MSCN [30]	Query-Driven Model
	NeuroCard [53] DeepDB [25]	Data-Driven Model
E2E Query Optimizer	Bao [40]	Tuning Hint
	Lero [59]	Scaling Cardinality

PilotScope: Evaluations

- AI4DB algorithms may not always work well
  - By PilotScope, we know how/when it could perform well/bad → guidelines for future research



(a) SMAC (Knob Tuning) on PostgreSQL, IMDB dataset, JOB workload

(b) Extend (Index Recommendation) on PostgreSQL, IMDB dataset, JOB workload

(c) DeepDB (Cardinality Estimation) on PostgreSQL, IMDB dataset, JOB-light workload

(d) Bao (Query Optimizer) on PostgreSQL, IMDB dataset, JOB workload

(e) SMAC (Knob Tuning) on PostgreSQL, STATS dataset, STATS-CEB workload

(f) Extend (Index Recommendation) on PostgreSQL, STATS dataset, STATS-CEB workload

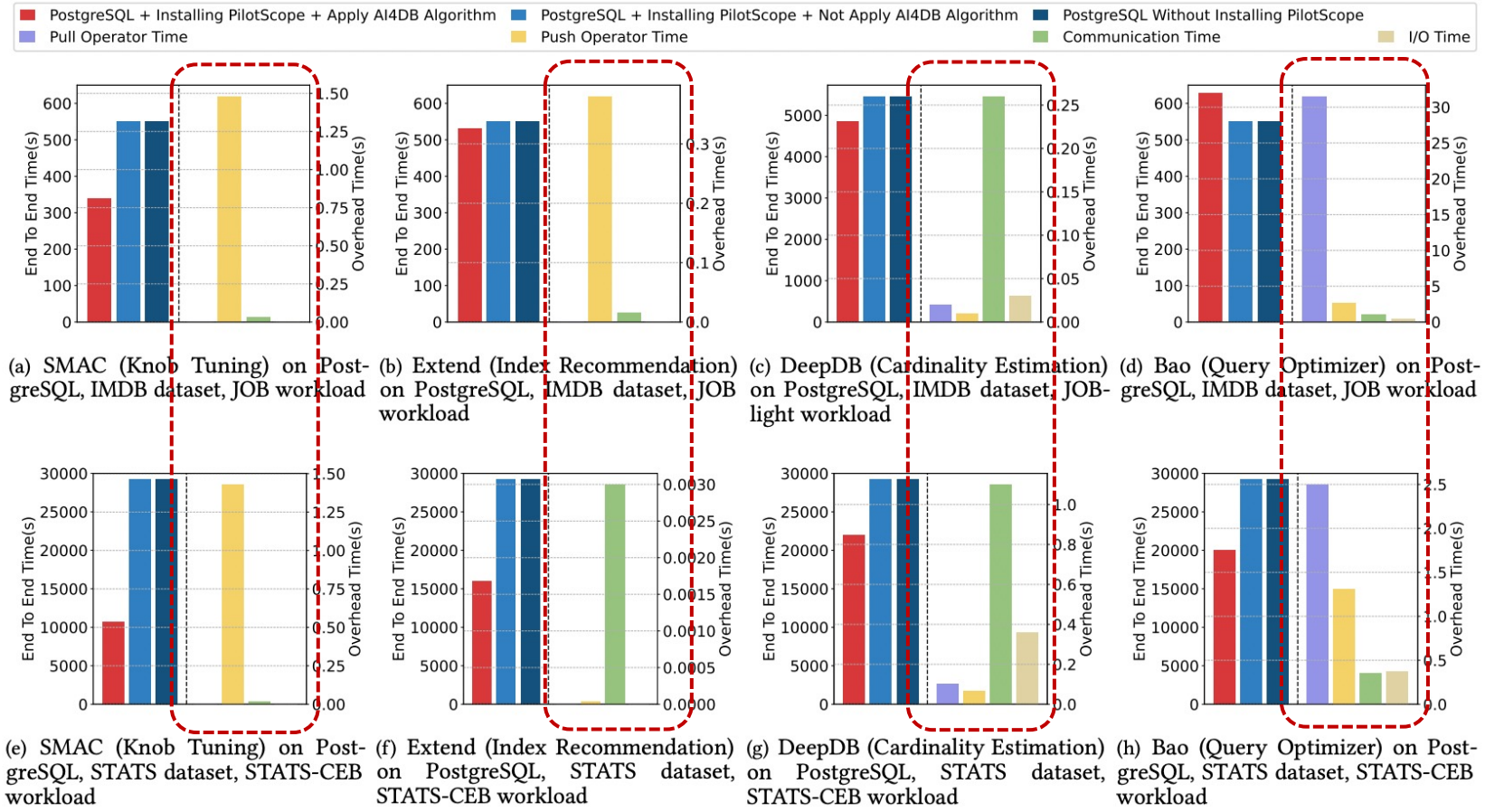
(g) DeepDB (Cardinality Estimation) on PostgreSQL, STATS dataset, STATS-CEB workload

(h) Bao (Query Optimizer) on PostgreSQL, STATS dataset, STATS-CEB workload

Experimental data from [PilotScope: steering databases with ML drivers , VLDB 2024]

PilotScope: Evaluations

- Overhead of PilotScope is negligible



(a) SMAC (Knob Tuning) on PostgreSQL, IMDB dataset, JOB workload

(b) Extend (Index Recommendation) on PostgreSQL, IMDB dataset, JOB workload

(c) DeepDB (Cardinality Estimation) on PostgreSQL, IMDB dataset, JOB-light workload

(d) Bao (Query Optimizer) on PostgreSQL, IMDB dataset, JOB workload

(e) SMAC (Knob Tuning) on PostgreSQL, STATS dataset, STATS-CEB workload

(f) Extend (Index Recommendation) on PostgreSQL, STATS dataset, STATS-CEB workload

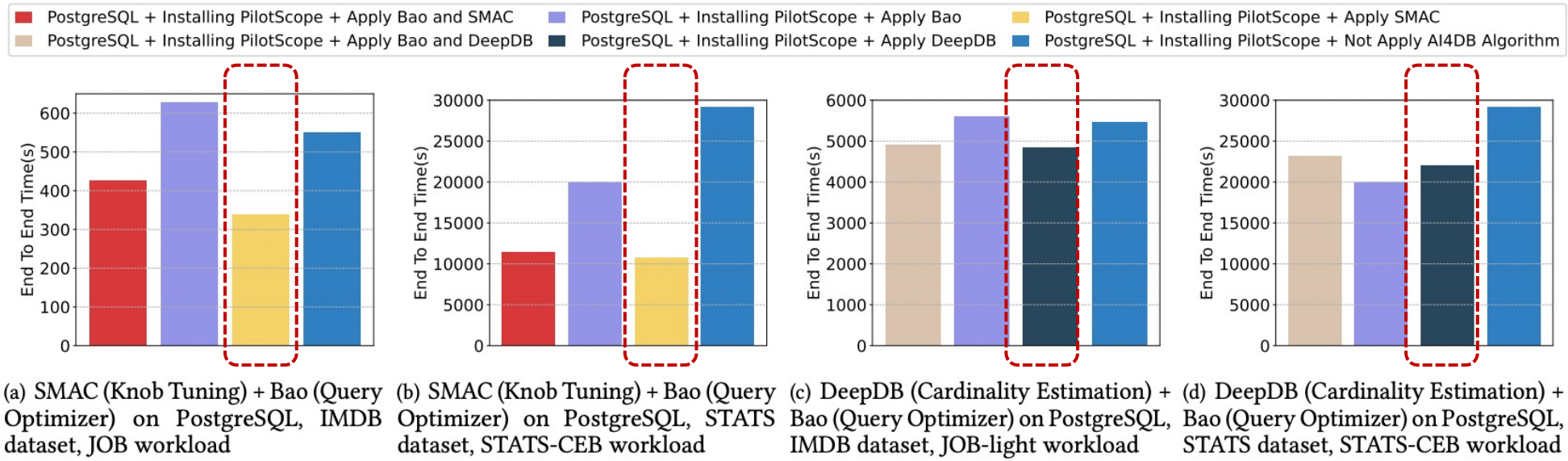
(g) DeepDB (Cardinality Estimation) on PostgreSQL, STATS dataset, STATS-CEB workload

(h) Bao (Query Optimizer) on PostgreSQL, STATS dataset, STATS-CEB workload

Experimental data from [PilotScope: steering databases with ML drivers , VLDB 2024]

PilotScope: Evaluations

- PilotScope could support to deploy and run multiple AI4DB algorithms at the same time
- 1 + 1 > 2? → not always hold, correlations are complex
  - A new and interesting topic in AI4DB



Experimental data from [PilotScope: steering databases with ML drivers , VLDB 2024]

PilotScope: More to Come

- Open-Source: https://github.com/alibaba/pilotscope

- Welcome for the contributions to build this community together

language Python language C language Scala license Apache 2.0 contributions Welcome

docs Usage Guideline docs Develop Guideline docs API Reference

AI4DB driver Knob Tuning AI4DB driver Index Recommendation AI4DB driver Cardinality Estimation

AI4DB driver E2E Query Optimizer

database PostgreSQL 13.1 database Spark 3.1

Figure generated by Tongyi Wanxiang (https://tongyi.aliyun.com/wanxiang/)

PilotScope: Demo

Demo Time!

Part 5: Summary and Future Opportunities

- ❑ Summary
- ❑ Future Opportunities

Summary

- Query optimizer is a suitable experimental plot for AI4DB → the prosperity of learned methods are developed in recent

- Learn to cost plans
  - Cardinality plays the most important role
  - Cardinality estimation: data-driven, query-driven and hybrid methods
  - Cost model: models for single/concurrent queries
  - Exhibit advantages in benchmark evaluations, but still not ready for actual deployment with some shortcomings

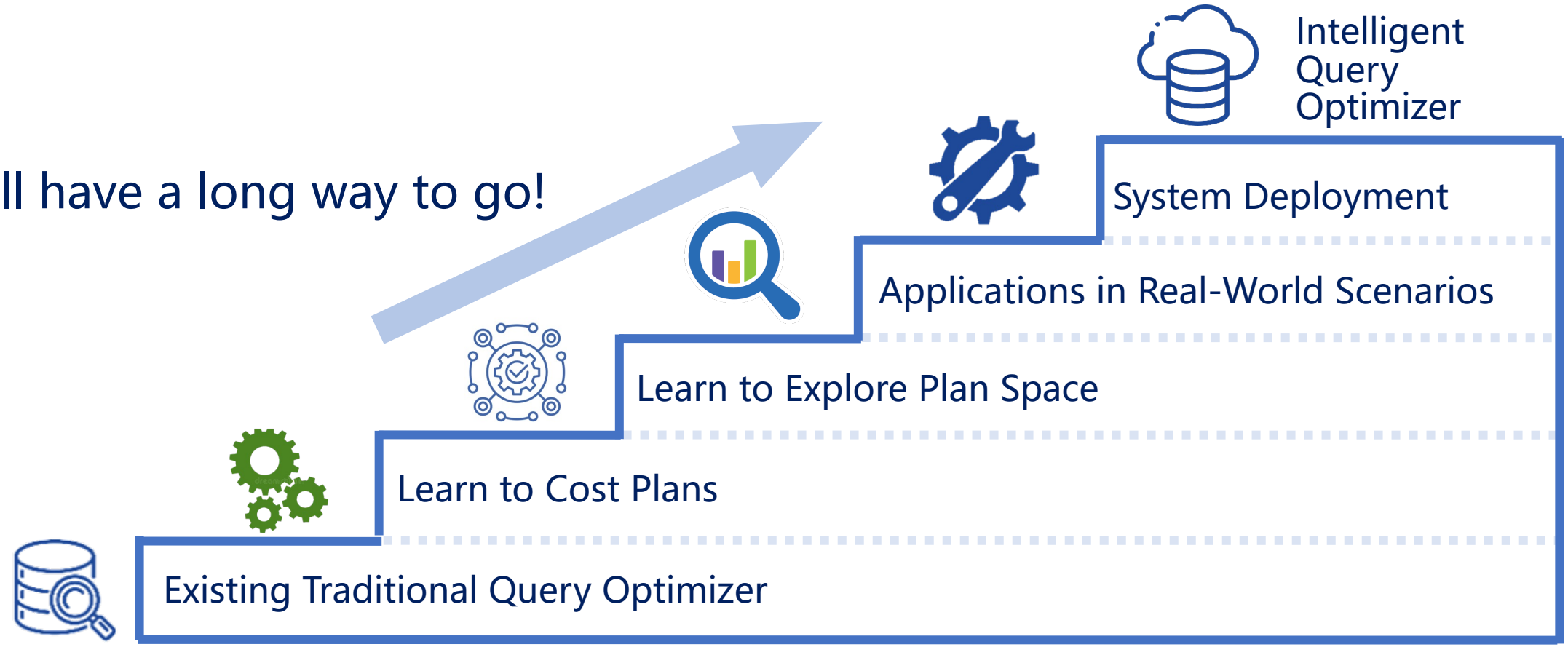
Summary

- Learn to explore plan space
  - Learn the value network to explore plans from scratch
  - Learn to steer existing query optimizers with different knobs
  - Lero: from learning-to-predict to learning-to-rank
  - Eraser: eliminate performance regressions

- Applications and deployment
  - Some attempts on customized systems and tasks
  - PilotScope: a middleware for ease of deployment, generally appliable to multiple algorithms and systems

Future Work

- Final goals: practical and intelligent query optimizer and beyond



Future Work

- Opportunity I: cardinality estimation
  - Fusion of data-driven and query-driven methods
  - Adaptive and general methods: different data, workloads, OLAP/OLTP, …

  - Pretrained cross-database model is possible routine
    - One model generally appliable to different databases/settings
    - Very friendly to deploy and apply

Future Work

- Opportunity II: optimize cardinality (cost) estimation with others tasks together
  - Complex relations of knob/index/… to query optimizer
  - 1 + 1 > 2, towards a more powerful query optimizer

Future Work

- Opportunity III: LLMs and learned query optimizer
  - LLMs could help
    - Text2SQL
    - Query rewriter
    - Data cleaning

Examples that AI/LLMs Help a Lot

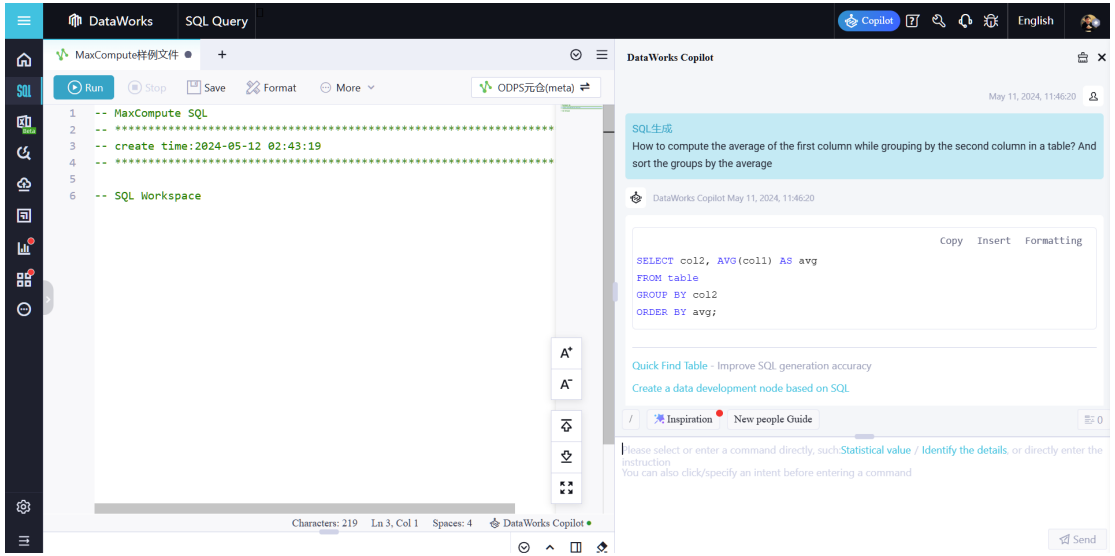
- Text-to-SQL (DataWorks Copilot)

Prompt for LLM

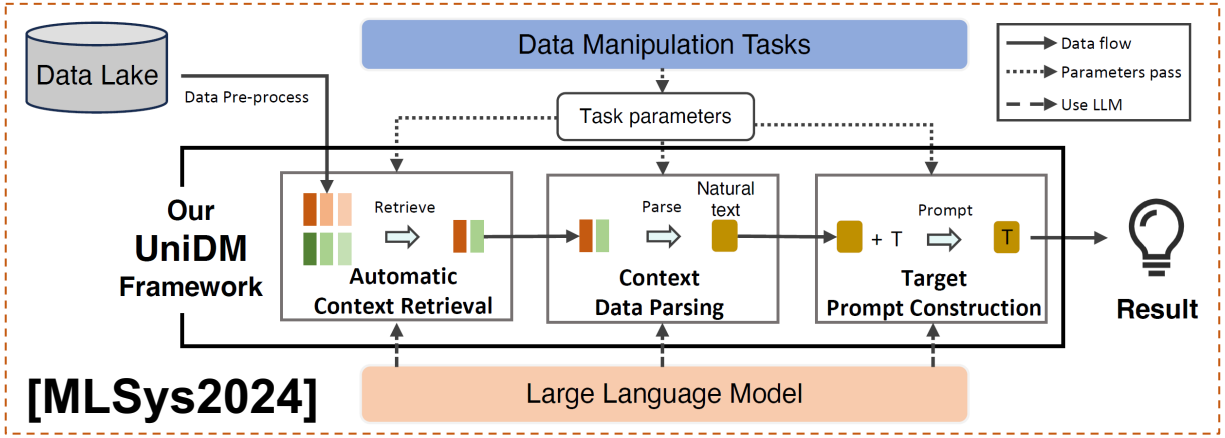
(1) Meta-data (table names, info, key-foreign key)
(2) SQL examples to join the tables
(3) (instruction, SQL) example pairs
(4) Text-to-SQL instruction (user's input)

(1)-(3) *are hidden from the end user*
[4] *is the user's input*

[VLDB2024]



- AI-aided data manipulation (UniDM)
  - Handling missing data
  - Data transformation
  - Error detection



[MLSys2024]

Future Work

- Opportunity III: LLMs and learned query optimizer
  - Can LLM help query plan generation?
  - How to fuse LLM with the information of data distributions?
    - Interplay between large and small models
      - Combine LLM reasoning and data knowledge representations together
      - Efficiency and effectiveness balance

Future Work

- Opportunity IV: extend the techniques to more AI4DB or even DB4AI tasks beyond query optimizer
  - Index recommendation
  - View advisors
  - System diagnosis
  - …

  - From intelligent query optimizer to intelligent database

Learned Query Optimizer: What is New and What is Next

Q & A

red.zr@alibaba-inc.com
lianggui.wlg@alibaba-inc.com
bolin.ding@alibaba-inc.com
jingren.zhou@alibaba-inc.com