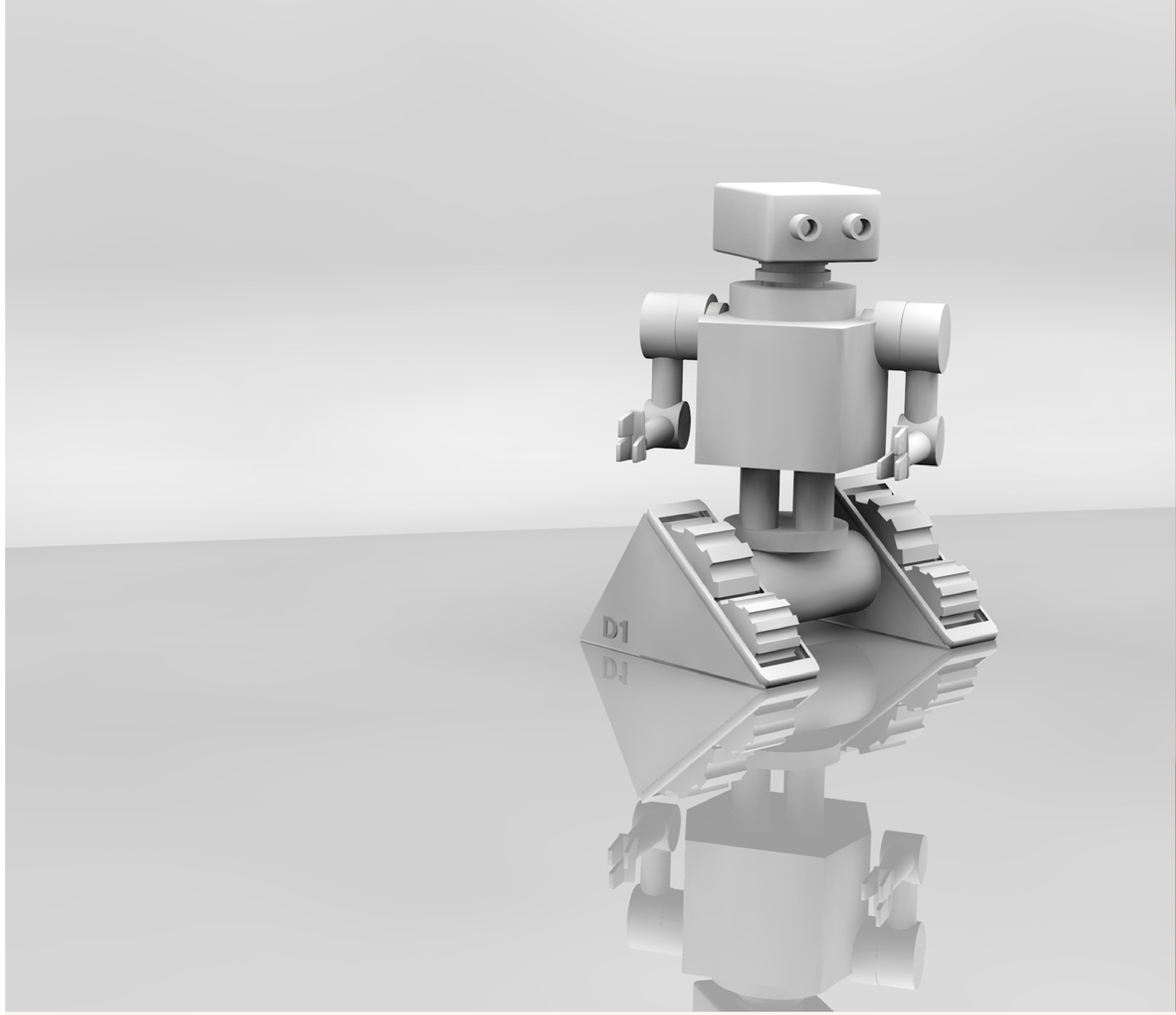
THE UNIVERSITY OF MELBOURNE

Self-Driving Database

Workload-driven Optimisation
& Index Selection



GROUP 40: JULY 20, 2023

Group 40

Qingxuan Yang

1359534

Master of Software Engineering

Runqiu Fei

1166093

Master of Data Science

Xiaoyi Liu

1401212

Master of Bioinformatic

Sunchuangyu Huang

1118472

Master of Data Science

Agenda

How to achieve autonomous database

Components of Self-Driving Databases

1

Workload Predictor

2

Tuner

3

Organizer

WORK LOAD-DRIVEN OPTIMISATION

01 Workload forecasting

02 Behaviour modelling

03 Action planning

i. Workload Forecasting

Workload forecasting involves predicting future workload characteristics. By accurately predicting future workload patterns, these systems can proactively allocate resources and optimize query execution strategies to meet service level agreements (SLAs) and enhance overall efficiency.

- Resource estimation
- Performance diagnosis
- Shift detection

i. Workload Forecasting

QB5000 Framework

- Considers both short-term and long-term forecasting horizons.
- Leverages advanced Machine Learning techniques: Linear Regression, Recurrent Neural Networks, and Kernel Regression.
- Tackles key forecasting challenges: arrival rate patterns, complexity, scalability, and workload evolution.
- Promotes dynamic adaptation to workloads for efficient resource allocation and improved performance.

ii. Behavioural Modeling

Given forecasted workload ➡ estimates and explains costs and benefits of all potential actions taken by DBMS.

- Time duration
- Resource consumption
- Impact on system performance

ii. Behavioural Modeling

Analytical Models

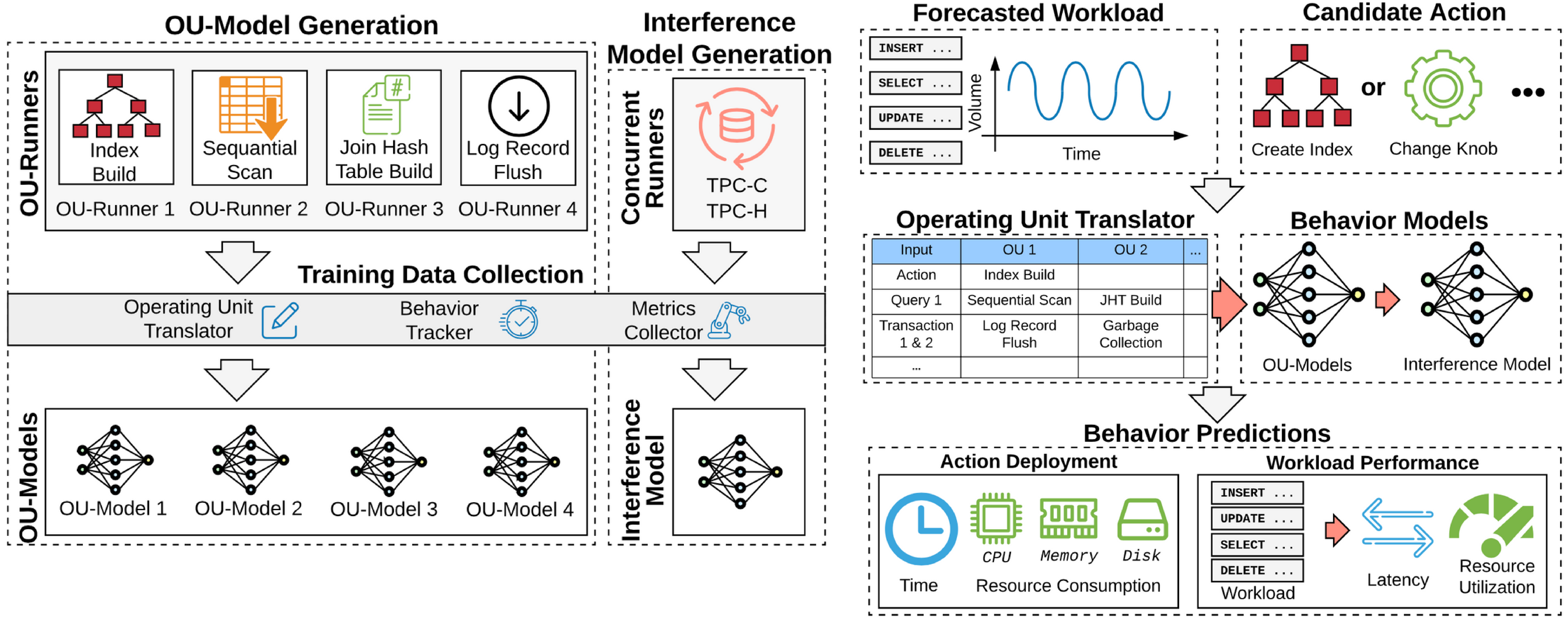
- **human-devised, white-box formulas** -> runtime behaviour of a DBMS
- Target different **workloads** and **runtime components**
  - OLAP workloads: Duggan et al. and Ahmad et al.
  - OLTP workloads: Resource Advisor, DBSeer, and DBSherlock
  - Query execution time: Wu et al.
- **Limitations**: customised for specific DBMS and workload, lack generality

ii. Behavioural Modeling
Machine Learning Models

- Target **isolated query execution**:
  - Query plan as input features: hierarchical classification with PQR trees, subspace projection, and QPPNet with deep neural networks.
  - Combine query-level and operational-level models: runtime tuning of pre-built models, implementation of additional scaling functions
- Target **concurrent operations for a set of queries**:
  - Wu et al.: queuing networks and Markov chains for dynamic query sets
  - GPredictor with graph-based deep learning prediction network.
- **Limitations**: expensive update and retraining

ii. Behavioural Modeling

State-of-the-art model



Input	OU 1	OU 2	...
Action	Index Build		
Query 1	Sequential Scan	JHT Build	
Transaction 1 & 2	Log Record Flush	Garbage Collection	
...			

iii. Action Planning

Forecasted workload and behaviour model estimates ➡

Solves a constrained optimisation problem

&

Plans the best sequence of actions for the DMBS to apply at the correct timing

iii. Action Planning
Static workloads models

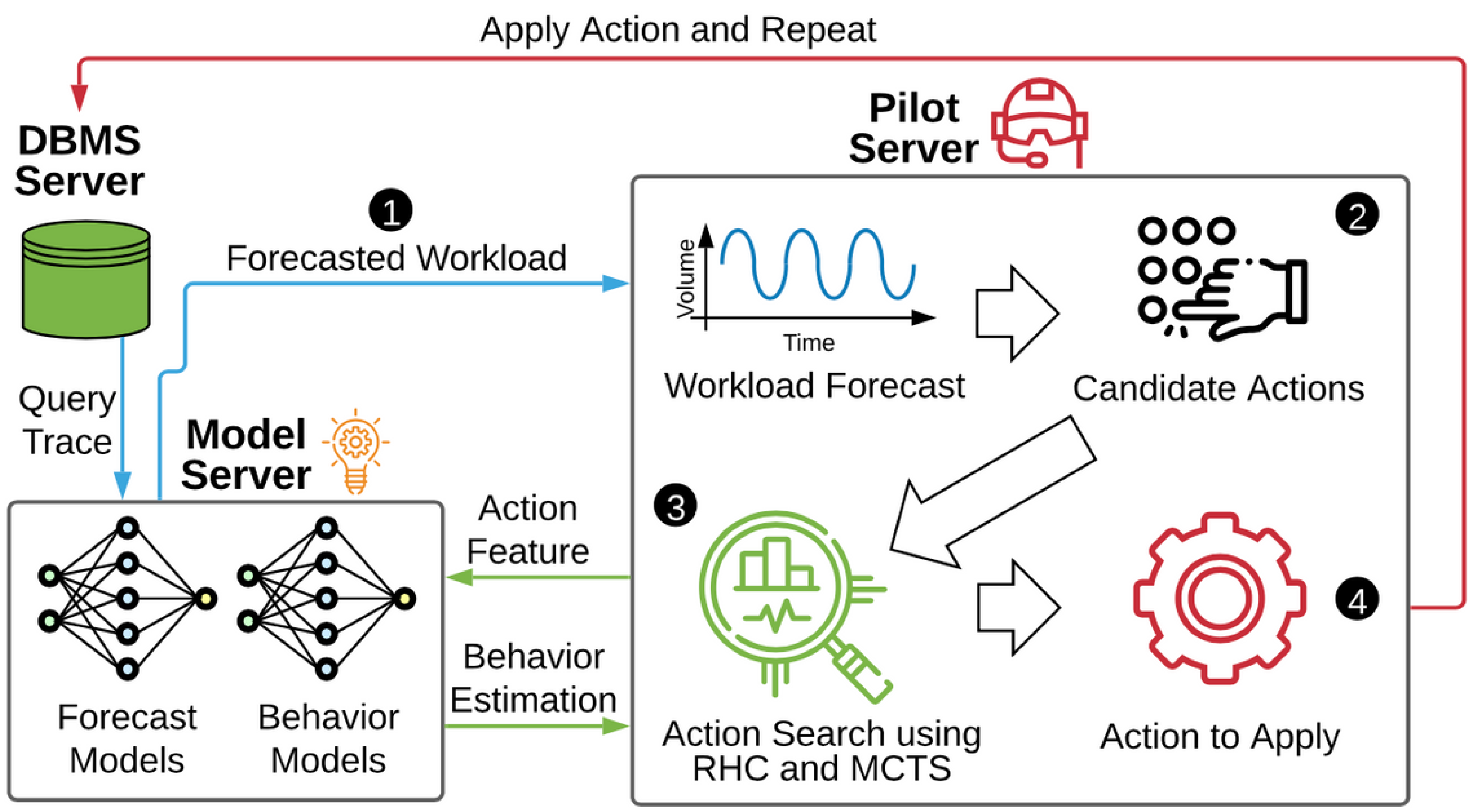
- Generates action for **physical design** or **knob configurations** given representative **static workloads**.
  - Optimised index configuration: Cophy, Kossman et al., Das et al., and DB2 Advisor.
  - Knob configurations: Aken et al., Qtune, and Zhang et al.
- **Limitation**: Lack temporal information
- **Improvement**: GREEDY-SEQ uses static workload tuning models to generate candidate actions, determine the best timing for each action by heuristics, and recursively merge them into a sequence.

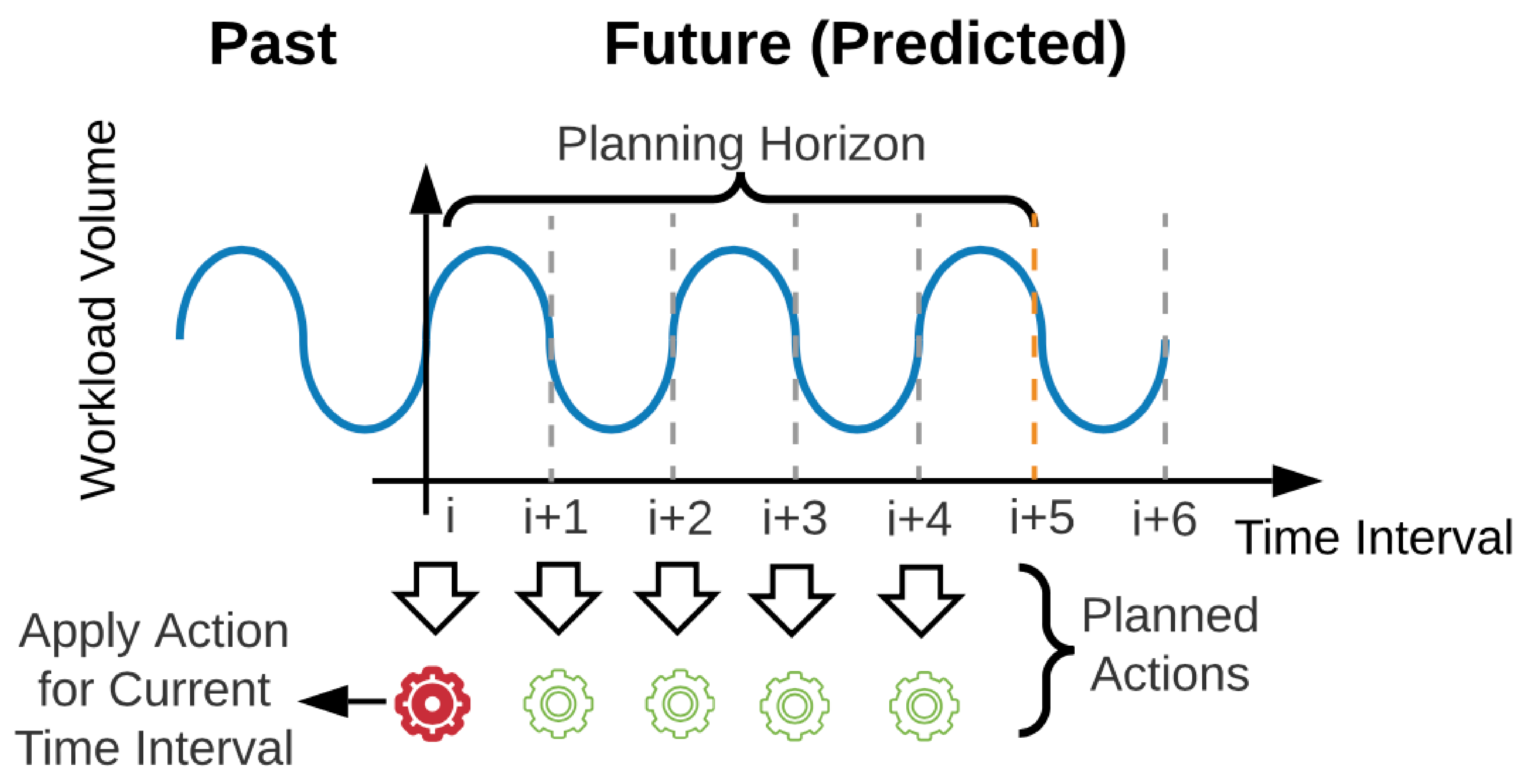
iii. Action Planning
Online tuning models

- Target **realtime database tuning with workload shifts**
    - Bruno and Chaudhuri: modify physical design based on query execution statistics
    - DBA bandits and COLT: index creation and profiling resource allocation through analysis of workload and performance statistics.
- **Limitations**: only solve problems after they occur and do not plan for future actions and workload patterns.

iii. Action Planning

State of the art models





Index Selection

i. What is Index Selection

- Optimum set of indexes
- Lowest query costs
- Under budget constrains

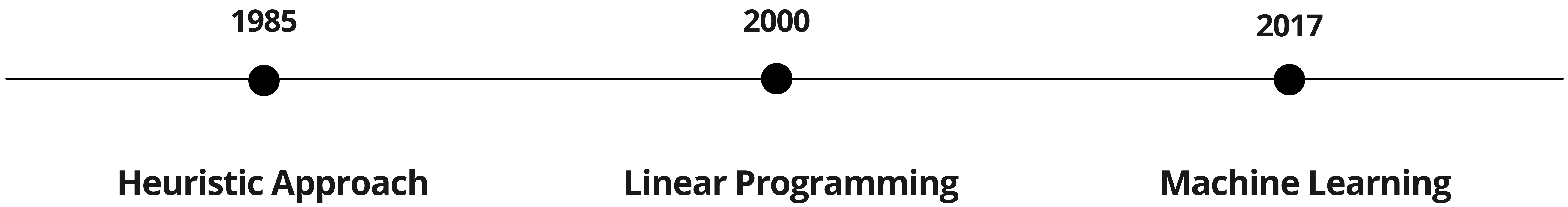
ii. Importance of Index Selection

- Query optimization
- Storage optimization
- Workload optimization

iii. How to select optimal indexes

1. Enumerating Index candidates
2. Selecting the optimum subset of enumerated candidates

iv. Evolution of Index Selection



iv.i Heuristic - Drop (1985)

Heuristic Approach

- Single-column indexes
- **Greedy elimination (*Drop*)**

iv.i Heuristic - AutoAdmin (1997)

Heuristic Approach

- **Multi-column indexes**
- **Restricting index candidate space**
- Greedy(m, k) to select k best candidates
- **“What-if” API**

iv.ii LP Methods - DB2 Advisor (2000)

Linear Programming

- Multi-column indexes
- Utilize “What-if” API
- Restricting index candidate space (like *AutoAdmin*)
- Greedy + Randomization
- **Constraints on storage budget**

iv.iii LP Methods - CoPhy (2011)

Linear Programming

- Multi-column indexes
- Utilize “What-if” API
- More **flexible** candidate enumeration and selection (unlike Greedy)
- **Constraints on storage budget**
- **LP solver (sensitive to LP problem complexity)**

iv.iv Machine Learning

Machine Learning

Multi-Arm Bandit for index tuning

- Kraska et al. proposed by 2022
- Expert agents with online learning capabilities (HTAP)
- Fast and Accurate
- Select an index based on exploration-exploitation

Budget Aware Monte-Carlo Tree Search

- Kossmann et al. proposed by 2022
- Budget limits computing performance
- Simulate-based techniques
- Recommend the best index within limited budgets

v. Challenges of Compare Index Selection

Index selection in databases:

- Computationally expensive and time-consuming to identify indexes for large workloads
- Complicated performance of evaluation metrics due to diverse optimization goals and various definitions of budget.
- Unknown database spec for benchmarking

vi. Experiment Results

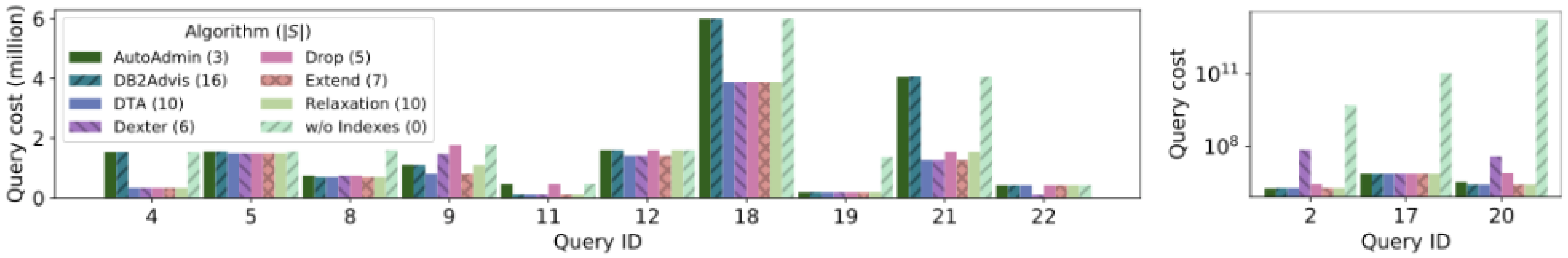
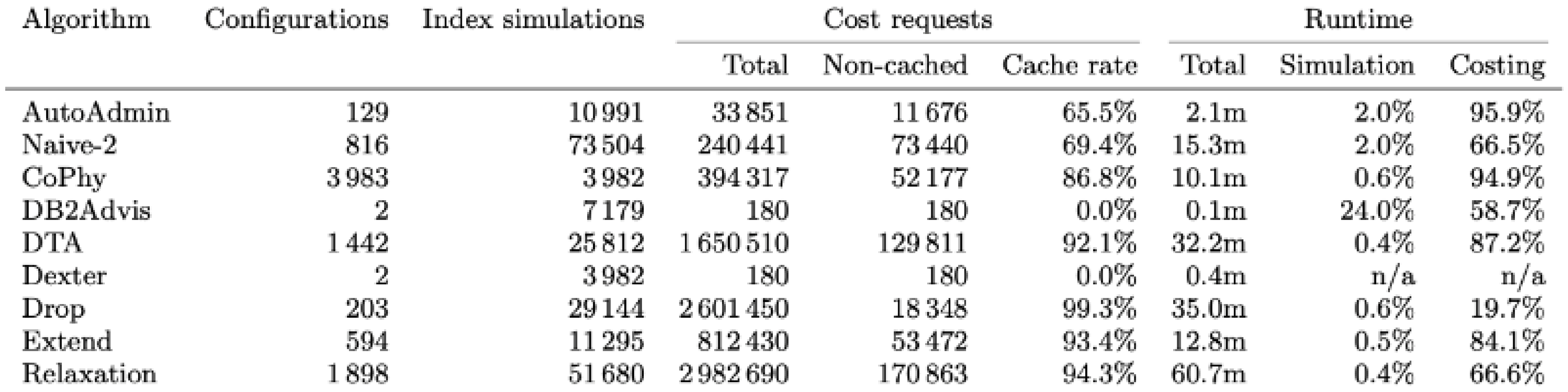


Figure: Estimated query processing costs for TPC-H (SF 10) on PostgreSQL with budget 5GB (Kossmann et al. 2022)

vi.i Experiment Results - Cache rate



Algorithm	Configurations	Index simulations	Cost requests			Runtime		
			Total	Non-cached	Cache rate	Total	Simulation	Costing
AutoAdmin	129	10 991	33 851	11 676	65.5%	2.1m	2.0%	95.9%
Naive-2	816	73 504	240 441	73 440	69.4%	15.3m	2.0%	66.5%
CoPhy	3 983	3 982	394 317	52 177	86.8%	10.1m	0.6%	94.9%
DB2Advis	2	7 179	180	180	0.0%	0.1m	24.0%	58.7%
DTA	1 442	25 812	1 650 510	129 811	92.1%	32.2m	0.4%	87.2%
Dexter	2	3 982	180	180	0.0%	0.4m	n/a	n/a
Drop	203	29 144	2 601 450	18 348	99.3%	35.0m	0.6%	19.7%
Extend	594	11 295	812 430	53 472	93.4%	12.8m	0.5%	84.1%
Relaxation	1 898	51 680	2 982 690	170 863	94.3%	60.7m	0.4%	66.6%

Figure: Algorithm cost estimation for TPC-DS (SF 10), storage consumption ≈ 5 GB, Kossmann et al (2022)

vi.ii Experiment Results (MCTS vs. MAB)

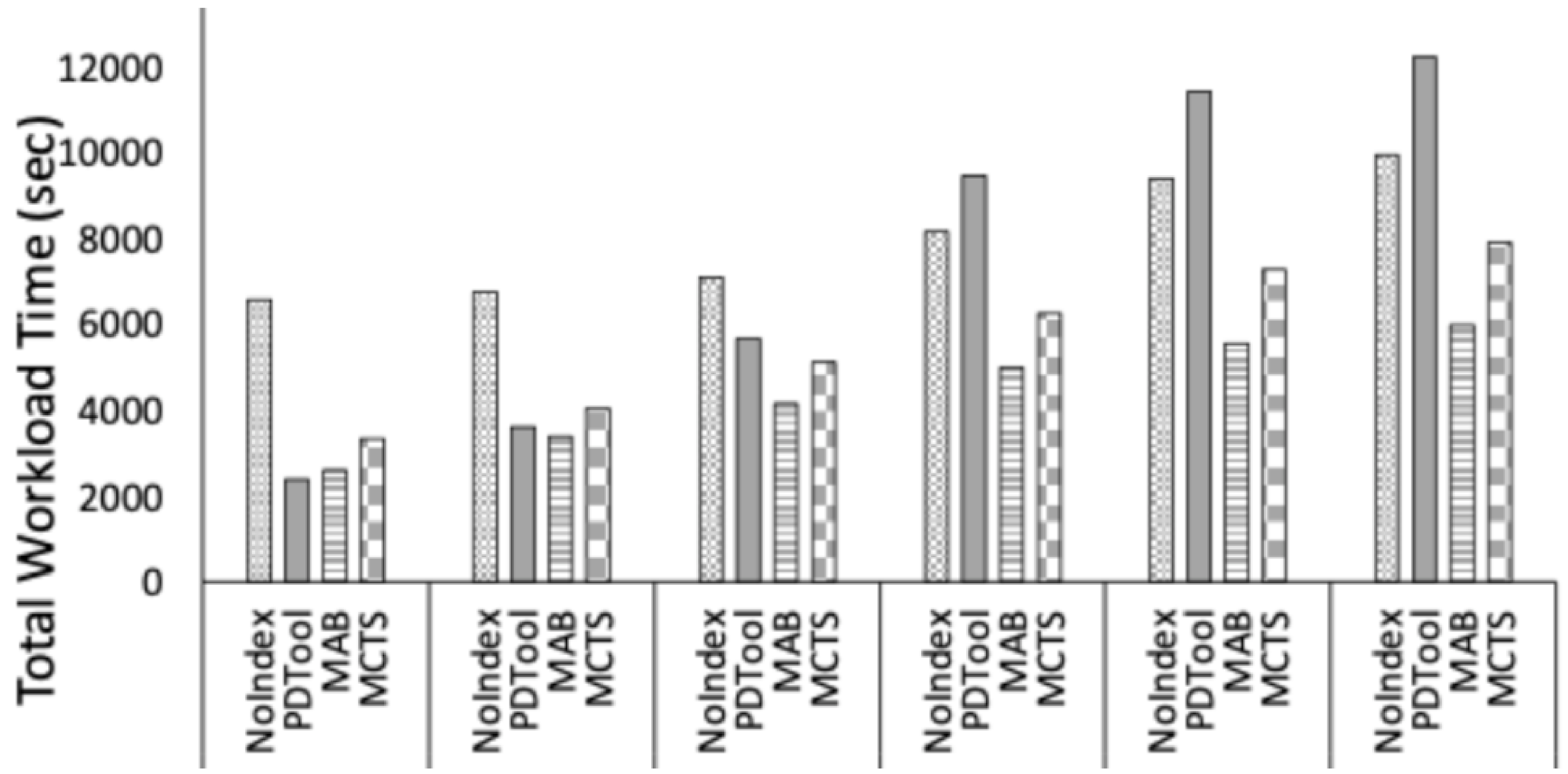


Figure: Transaction Analytic Ratio between MAB and MCTS budget-aware approach (Kraska et al., 2022)

vi.iii Discussion

1. Evolution of index selection
2. Two domination approaches
    a. Linear Programming
    b. Machine Learning
3. Performance testing on TPC-H and TPC-DS benchmarks demonstrated significant improvements

Summary

Workload-Driving Optimisation

- Behavior modeling includes:
  - Analytical models
  - Machine Learning models
- Action planning involves:
  - Use of forecasted workloads and behavioral model estimates to plan action sequence
  - State-of-art algorithms: ModelBot2 & PB0
  - Future work needed for less restricted optimization and complex feature tuning

Index Selection

- Evolution from heuristic methods to linear programming and machine learning techniques
- All algorithms demonstrated improvements in operational cost and index storage
- Integrated machine learning within DBMS will be the future

Q & A

Reference

[1] Transaction Processing Performance Council (TPC). *TPC Benchmark H Standard Specification*. Technical Report. 2023. URL: https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf.

[2] Transaction Processing Performance Council (TPC). *TPC-DS Benchmark Standard Specification*. Technical Report. 2023. URL: https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-ds_v2.1.0.pdf.

[3] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. “Automated Selection of Materialized Views and Indexes in SQL Databases”. In: *VLDB ’00: Proceedings of the 26th International Conference on Very Large Data Bases*. 2000, pp. 496–505.

[4] Mumtaz Ahmad et al. “Predicting completion times of batch query workloads using interaction-aware models and simulation”. In: *Proceedings of the 14th International Conference on Extending Database Technology*. 2011, pp. 449–460.

[5] Surajit Chaudhuri and Vivek Narasayya. “Anytime Algorithm of Database Tuning Advisor for Microsoft SQL Server”. June 2020. URL: https://www.microsoft.com/en-us/research/publication/anytime-algorithm-of-database-tuning-advisor-for-microsoft-sql-server/.

Reference

[6] Surajit Chaudhuri and Vivek R Narasayya. "An efficient, cost-driven index selection tool for Microsoft SQL server". In: *VLDB*. Vol. 97. San Francisco. 1997, pp. 146–155.

[7] Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. "CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads". In: *Proc. VLDB Endow.* 4.6 (Mar. 2011), pp. 362–372. DOI: 10.14778/1978665.1978668.

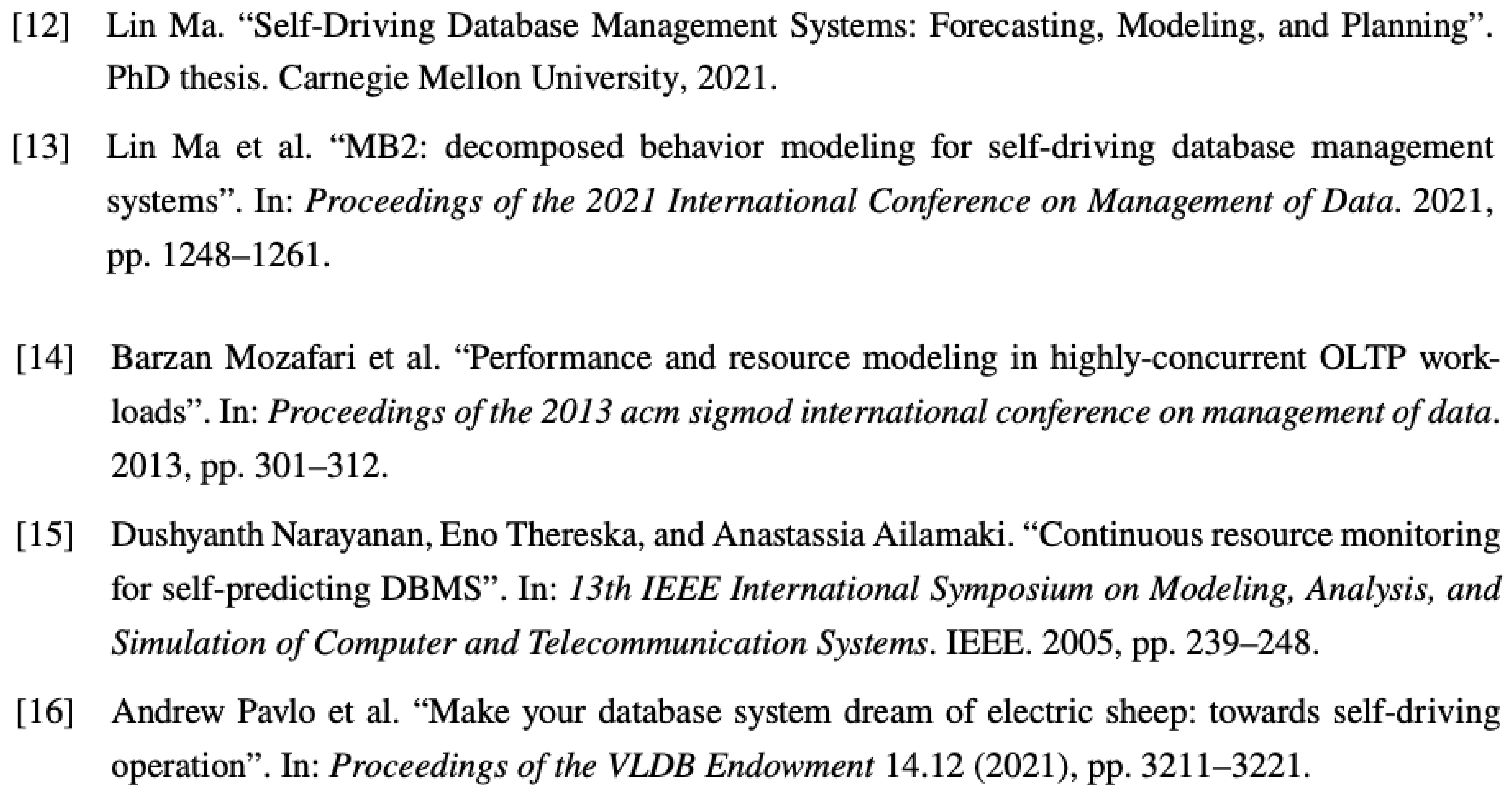
[8] Jennie Duggan et al. "Performance prediction for concurrent database workloads". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 2011, pp. 337–348.

[9] Jan Kossmann and Rainer Schlosser. "Self-driving database systems: a conceptual approach". In: *Distributed and Parallel Databases* 38 (2020), pp. 795–817.

[10] Jan Kossmann et al. "Magic Mirror in My Hand, Which is the Best in the Land? An Experimental Evaluation of Index Selection Algorithms". In: *Proc. VLDB Endow.* 13.12 (July 2020), pp. 2382–2395. DOI: 10.14778/3407790.3407832.

[11] Tim Kraska et al. "No DBA? No regret! Multi-armed bandits for index tuning of analytical and HTAP workloads with provable guarantees". In: *Proceedings of the 2022 International Conference on Management of Data*. 2022.

Reference



[12] Lin Ma. “Self-Driving Database Management Systems: Forecasting, Modeling, and Planning”. PhD thesis. Carnegie Mellon University, 2021.

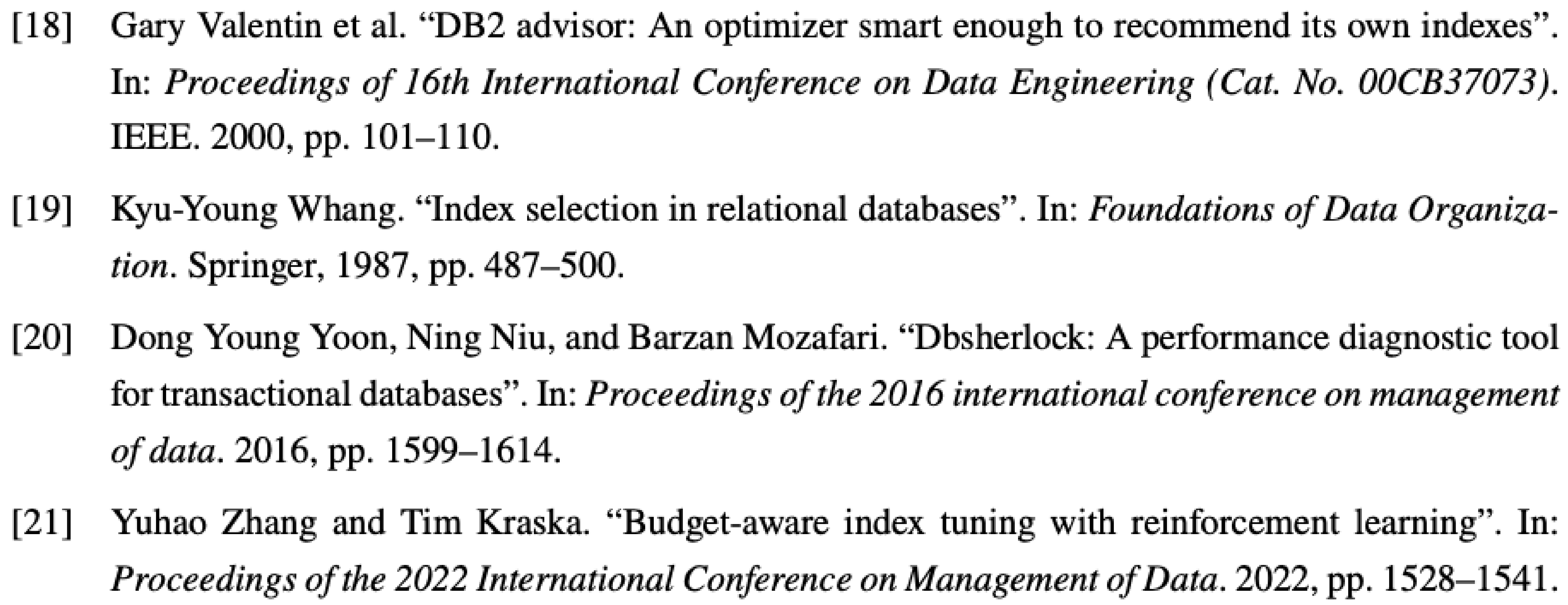
[13] Lin Ma et al. “MB2: decomposed behavior modeling for self-driving database management systems”. In: *Proceedings of the 2021 International Conference on Management of Data*. 2021, pp. 1248–1261.

[14] Barzan Mozafari et al. “Performance and resource modeling in highly-concurrent OLTP workloads”. In: *Proceedings of the 2013 acm sigmod international conference on management of data*. 2013, pp. 301–312.

[15] Dushyanth Narayanan, Eno Thereska, and Anastassia Ailamaki. “Continuous resource monitoring for self-predicting DBMS”. In: *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE. 2005, pp. 239–248.

[16] Andrew Pavlo et al. “Make your database system dream of electric sheep: towards self-driving operation”. In: *Proceedings of the VLDB Endowment* 14.12 (2021), pp. 3211–3221.

Reference



[18] Gary Valentin et al. “DB2 advisor: An optimizer smart enough to recommend its own indexes”. In: *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. IEEE. 2000, pp. 101–110.

[19] Kyu-Young Whang. “Index selection in relational databases”. In: *Foundations of Data Organization*. Springer, 1987, pp. 487–500.

[20] Dong Young Yoon, Ning Niu, and Barzan Mozafari. “Dbsherlock: A performance diagnostic tool for transactional databases”. In: *Proceedings of the 2016 international conference on management of data*. 2016, pp. 1599–1614.

[21] Yuhao Zhang and Tim Kraska. “Budget-aware index tuning with reinforcement learning”. In: *Proceedings of the 2022 International Conference on Management of Data*. 2022, pp. 1528–1541.