Lecture #05

Carnegie Mellon University

Special Topics:

Self-Driving Database Management Systems

Index Recommendation III

@DJ_Mooshoo// 15-799 // Spring 2022

TODAY'S AGENDA

Overview
Index Selection Algorithms
Methods
Results
Parting Thoughts

TODAY'S AGENDA

Overview
- Terminology
- Motivation
- Challenges

Index Selection Algorithms

Methods

Results

Parting Thoughts

TERMINOLOGY

Workload and Index

- Simplifying observation: define by attributes

Index Configuration

Potential Index

Syntactically Relevant

Index Candidates

Index Interaction

- Can be positive or negative [0]

MOTIVATION

Problem: There are many approaches to index selection, but comparisons between algorithms is limited.

Goal: compare state-of-the-art index selection algorithms more comprehensibly by:

- Measuring in multiple dimensions
- Developing a standard framework for comparisons

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

CHALLENGES

- Different Goals
  - Maximize Benefit or Benefit/Storage
- Algorithms with Parameters
  - Choosing the right setting for a workload
- Query Cost Estimation
  - DBMS - specific
  - Often not reflective of actual cost

THIS PAPER

- Survey of 8 index selection algorithms
- Provides an evaluation framework that addresses the challenges (in part)
- Present evaluations of the algorithms within the framework

TODAY'S AGENDA

Overview
Index Selection Algorithms
Methods
Results
Parting Thoughts

OVERVIEW OF ALGORITHMS

8 algorithms varying in:

- Approach
- Objective and stop criteria
- Academic/Commercial/Open Source
- Complexity (?)

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

	Drop
Minimization goal	Costs
Stop criterion	# Indexes
Multi-column indexes	No
Index interaction	Yes

DROP (1985)

1. Start with *every* single-column index
2. Drop index that leads to the lowest cost of workload
3. Stop when cost cannot be reduced

Original version uses own cost model

Modifications:

Use the framework for cost estimation

Stop when maximum # indexes is reached

INDEX SELECTION IN RELATIONAL DATABASES
FODO 1985

AutoAdmin (1997)

	AutoAdmin
Minimization goal	Costs
Stop criterion	# Indexes
Multi-column indexes	Yes
Index interaction	Yes

Microsoft SQL Server Tuner

1. Start with per-query candidates
2. Naïve Enumeration
3. Greedy extension
    1. Adding indexes
    2. Adding columns to indexes
4. Stop at maximum # index

Reduce estimation calls using "atomic configurations"

AN EFFICIENT COST-DRIVEN INDEX INDEX SELECTION TOOL FOR MICROSOFT SQL SERVER
VLDB 1997

DTA Anytime (2020)

	DTA
Minimization goal	Costs
Stop criterion	Storage
Multi-column indexes	Yes
Index interaction	Yes

Core approach is the same as AutoAdmin, plus:

1. Also tunes materialized views and partitioning (not evaluated)
2. Considers multi-column indexes from the start
3. Merges query-level candidates
4. Considers index interaction to avoid evaluating suboptimal sets
5. Stop at *any time*

ANYTIME ALGORITHM OF DATABASE TUNING ADVISOR FOR MICROSOFT SQL SERVER
MICROSOFT RESEARCH 2020

DB2 ADVISOR (2000)

	DB2Advis
Minimization goal	$\frac{Costs}{Storage}$
Stop criterion	Storage
Multi-column indexes	Yes
Index interaction	(Yes)

1. All candidates from every query added as hypothetical index
2. Indexes which are used by optimizer added to candidate set
3. Sort candidates by benefit-per-space ratio
4. Randomly vary set to account for index interaction

DB2 ADVISOR: AN OPTIMIZER SMART ENOUGH TO RECOMMEND ITS OWN INDEXES
ICDE 2000

RELAXATION (2005)

	Relaxation
Minimization goal	$\frac{Costs}{Storage}$
Stop criterion	Storage
Multi-column indexes	Yes
Index interaction	Yes

1. Start with optimal index set for each query
   → Original paper exploits optimizer code paths
2. Union of all query-level sets to create (huge) candidate set
3. Reduce candidate set (relaxing) by iteratively:
   → Merging
   → Removing attributes (Prefixing)
   → Promote to
   → Removing Indexes

AUTOMATIC PHYSICAL DATABASE TUNING: A RELAXATION-BASED APPROACH
SIGMOD 2005

CoPhy LP (2011)

	CoPhy
Minimization goal	$\frac{Costs}{Storage}$
Stop criterion	Storage
Multi-column indexes	(Yes)
Index interaction	Yes

1. Formulate index selection as an integer linear program
2. Use off-the-shelf solver to find optimal solution
3. Scalability Issues
   → Binary variables for each (index, query) pair
   → A variable for each subset of candidate set
   → Solution: "Decomposition Heuristic" to reduce problem size [1]

COPHY: A SCALABLE, PORTABLE, AND INTERACTIVE INDEX ADVISOR FOR LARGE WORKLOADS
PVLDB 2011

DEXTER (2017)

	Dexter
Minimization goal	Costs
Stop criterion	Savings (%)
Multi-column indexes	(Limit 2)
Index interaction	Yes

1. Gather queries and runtime information, templatize them
2. Add hypothetical indexes of all single and 2-column index to configuration
3. Run explain to see which indexes are chosen – use these

EXTEND (2019)

	Extend
Minimization goal	$\frac{Costs}{Storage}$
Stop criterion	Storage
Multi-column indexes	Yes
Index interaction	Yes

1. Start with an empty solution set
2. Greedily pick action with the greatest reduction in cost/storage
   - Adding a new index
   - Appending an attribute to an existing index
3. Stop when no cost reduction can be made or storage budget is met

EFFICIENT SCALABLE MULTI-ATTRIBUTE INDEX SELECTION USING RECURSIVE STRATEGIES
ICDE 2019

SUMMARY

- Query-based (DB2Advis and Dexter) vs Index combination-based
  - Speed vs index interactions
- Approach
  - Additive (AutoAdmin, DTA, Extend)
  - Reductive (Drop, Relaxation)

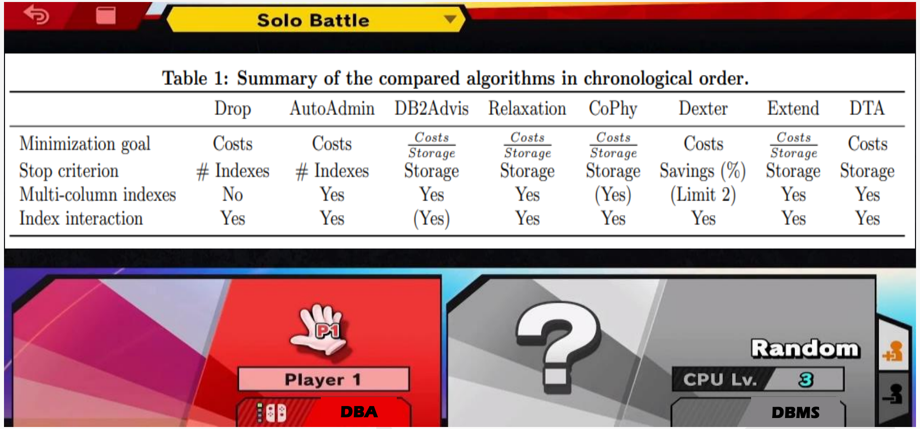


Table 1: Summary of the compared algorithms in chronological order.

	Drop	AutoAdmin	DB2Advis	Relaxation	CoPhy	Dexter	Extend	DTA
Minimization goal	Costs	Costs	$\frac{Costs}{Storage}$	$\frac{Costs}{Storage}$	$\frac{Costs}{Storage}$	Costs	$\frac{Costs}{Storage}$	Costs
Stop criterion	# Indexes	# Indexes	Storage	Storage	Storage	Savings (%)	Storage	Storage
Multi-column indexes	No	Yes	Yes	Yes	(Yes)	(Limit 2)	Yes	Yes
Index interaction	Yes	Yes	(Yes)	Yes	Yes	Yes	Yes	Yes

TODAY'S AGENDA

Overview

Index Selection Algorithms

Methods

- Benchmarks
- Framework
- Evaluation

Results

Parting Thoughts

BENCHMARKS

Table 2: Metrics for the evaluated benchmark schemata and workloads. The number of relevant index candidates was determined by generating all permutations of all syntactically relevant indexes.

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
JOB	Real-world	21	108	113	73	218	552	1 080
TPC-H	Synthetic uniform	8	61	22	53	398	3 306	29 088
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- **TPC-H (10x)** – Relatively small OLAP benchmark
- **TPC-DS (10x)** – More sophisticated OLAP benchmark
- **Join Order Benchmark** – based on IMDB
  - Queries focused on joins, not a lot of wide column indices
- No writes/updates – purely analytical
  - No index maintenance cost

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

FRAMEWORK

- PostgreSQL 12 chosen due to HypoPG
- Algorithms reimplemented in Python 3
- Key Concept: **Abstraction Layers**
  - CostEvaluation
  - DatabaseConnector
- Cost Estimation Caching

EVALUATION FOCUS

Solution quality with respect to storage constraint

- Cost reduction
- Algorithm Runtime
- Solution Granularity

Potential Indexes

- All (relevant) indexes of width 2

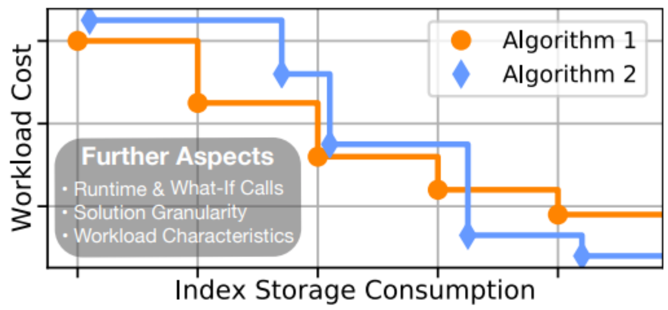


Figure 1: Various dimensions need to be considered for the evaluation of index selection algorithms.

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TODAY'S AGENDA

Overview

Index Selection Algorithms

Methods

Results

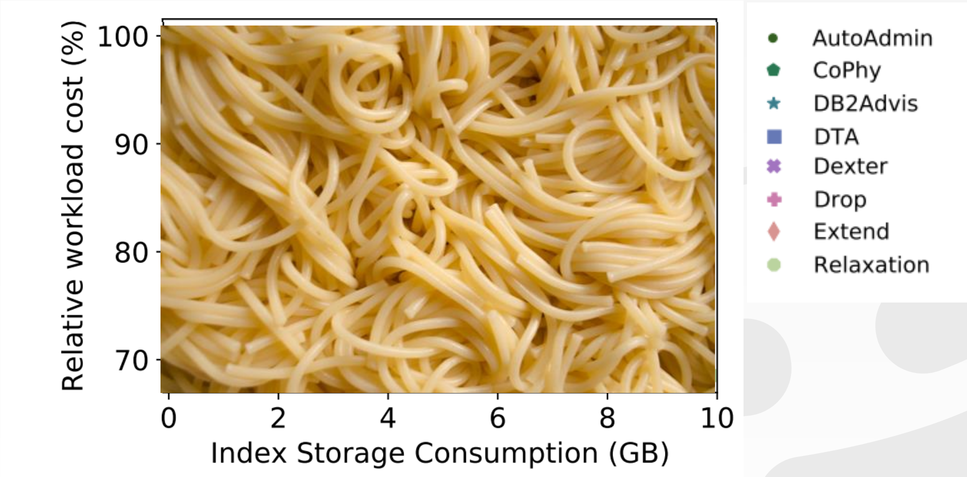
- Per Benchmark
- Further Dimensions
- Important Findings

Parting Thoughts

TPC-H

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-H	Synthetic uniform	8	61	22	53	398	3 306	29 088

- Stop criteria leads to differences in solution characteristic
  - **AutoAdmin** and **Drop** only find solution at 2GB, but it's a good one
  - Due to a "dominating table"
- Best solution depends on storage budget

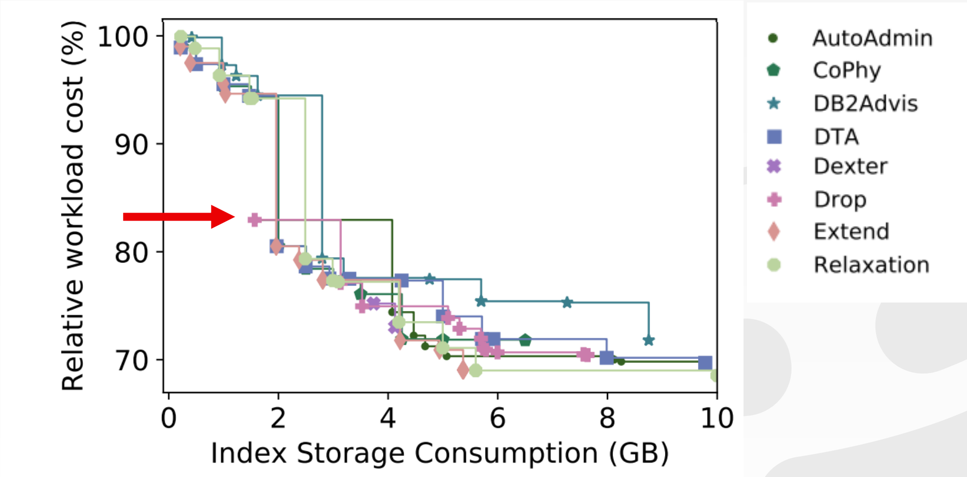


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-H

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-H	Synthetic uniform	8	61	22	53	398	3 306	29 088

- Stop criteria leads to differences in solution characteristic
  - **AutoAdmin** and **Drop** only find solution at 2GB, but it's a good one
  - Due to a "dominating table"
- Best solution depends on storage budget

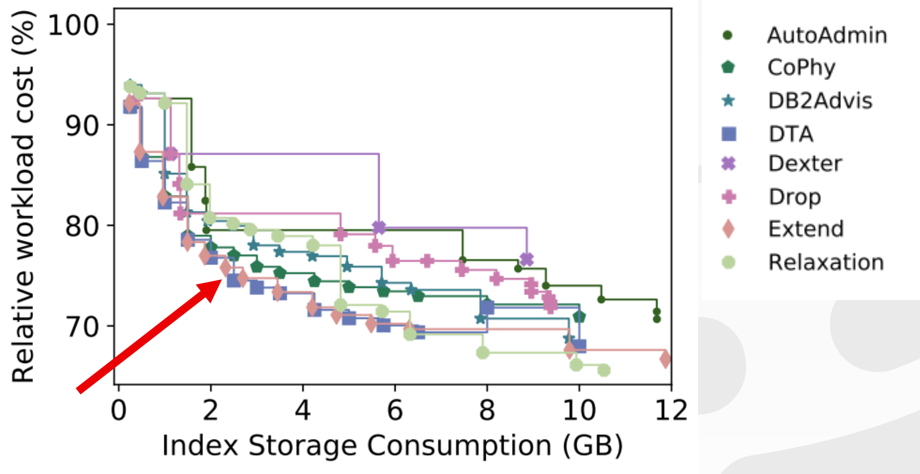


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-DS

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- Extend and DTA are best when budget < 6GB
  - Additive approach
- Extend and Relaxation find best solutions
  - Not the fastest
- Dexter has poor granularity

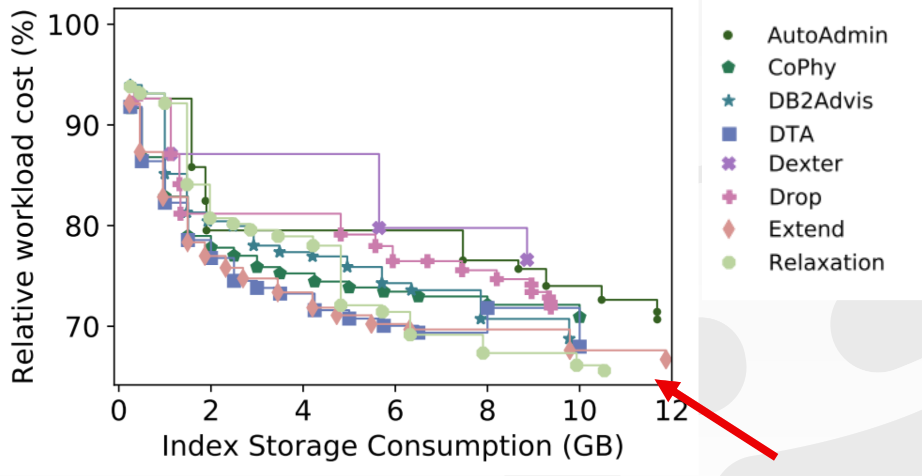


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-DS

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- Extend and DTA are best when budget < 6GB
  - Additive approach
- Extend and Relaxation find best solutions
  - Not the fastest
- Dexter has poor granularity

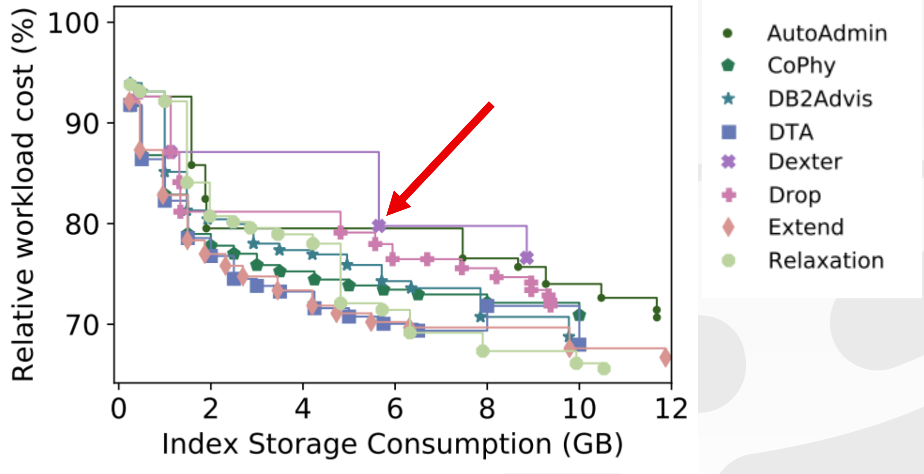


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-DS

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n=1$	$n=2$	$n=3$	$n=4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- Extend and DTA are best when budget < 6GB
  - Additive approach
- Extend and Relaxation find best solutions
  - Not the fastest
- Dexter has poor granularity

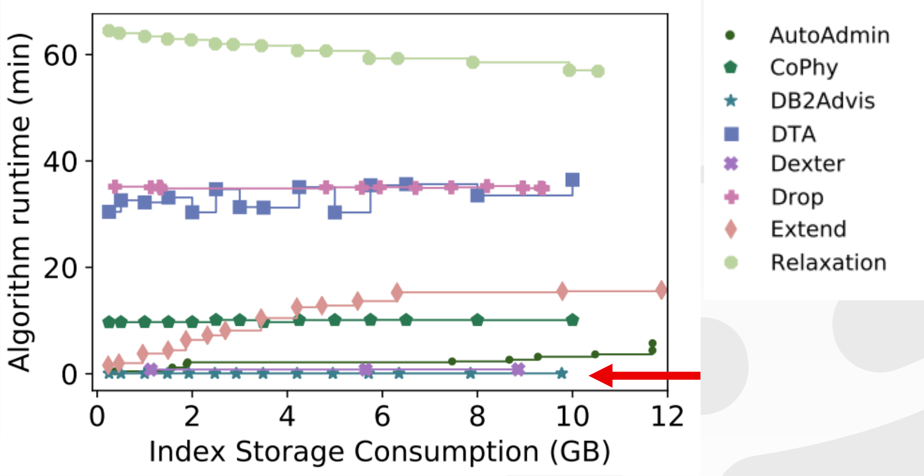


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-DS

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- DB2Advis is fast
  - Because it only calls CostEstimation on 2 configurations ("all" and "none")
- Relaxation scales poorly with number of potential indexes
  - Compared to TPC-H

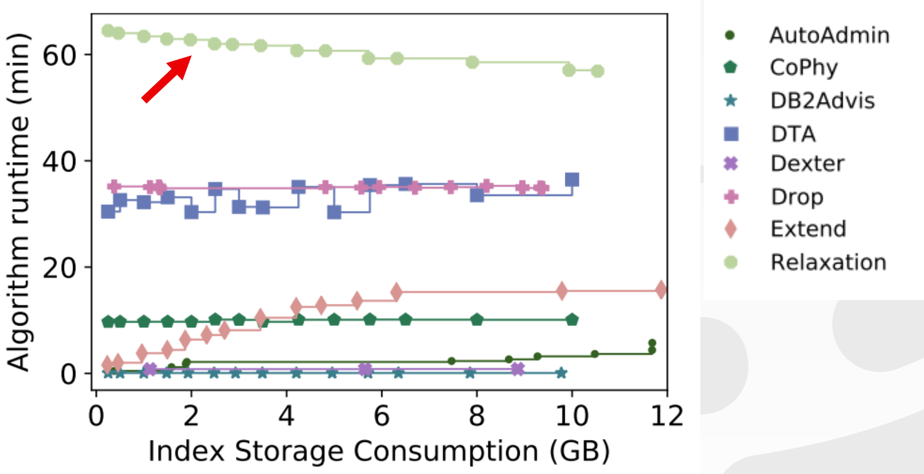


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

TPC-DS

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n = 1$	$n = 2$	$n = 3$	$n = 4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

- DB2Advis is fast
  - Because it only calls CostEstimation on 2 configurations ("all" and "none")
- Relaxation scales poorly with number of potential indexes
  - Compared to TPC-H

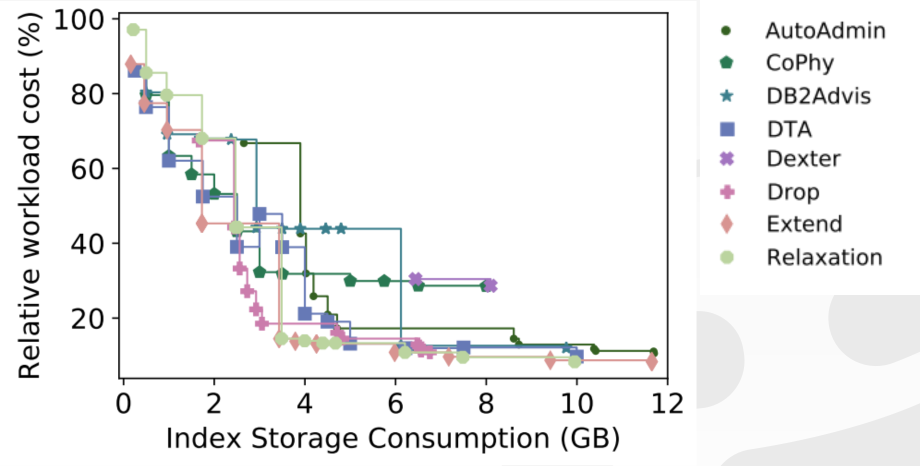


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

JOIN ORDER BENCHMARK

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n=1$	$n=2$	$n=3$	$n=4$
JOB	Real-world	21	108	113	73	218	552	1 080

- Workload lends itself to massive speedups
  - Adding relatively small
- Dexter has poor granularity

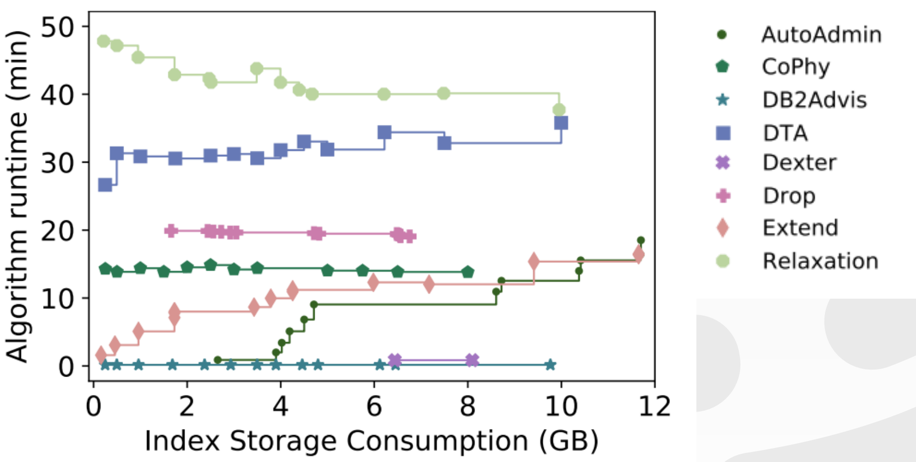


MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

JOIN ORDER BENCHMARK

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n=1$	$n=2$	$n=3$	$n=4$
JOB	Real-world	21	108	113	73	218	552	1 080

- Reduction vs Additive Approach
  - Runtime decreases/increases with respect to storage budget
  - Initial candidate set has large impact on runtime



MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

INFLUENCE OF PARAMETERS

- DB2Advis – Try Variation
  - Candidate set too large for random variation to reliably find improvements
  - Could be helpful in databases with fewer tables/attributes
- AutoAdmin – Naïve Enumerations
  - Increasing k=1 to 2 increases runtime 3-10x
  - Sometimes smaller k leads to *better* solution
- DTA – Runtime Limits
  - Running **9 minutes** leads to a solution within 3% of running **14 hours**

***These may be dependent on workload and DBMS**

FURTHER DIMENSIONS

- Index selection order
  - For a specific algorithm and workload, what indexes are selected - and when?
  - Fine-grain analysis of algorithm
- Runtime cost breakdown

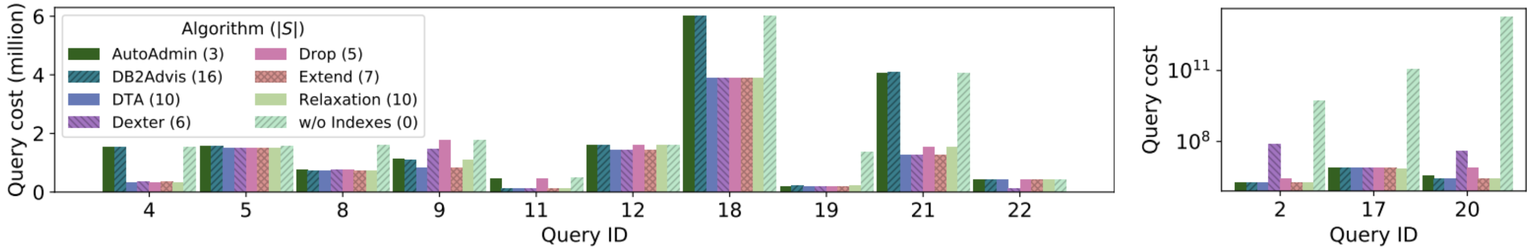


Figure 4: Estimated query processing costs for TPC-H (scale factor 10) on PostgreSQL. Queries 1, 3, 6, 7, 10, 13, 14, 15, and 16 are omitted as their costs were not affected by indexes for a budget of 5 GB. Expensive queries (2, 17, 20) depicted with log (right), others (left) with linear scale. *S* is the final index configuration.

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

FURTHER DIMENSIONS

- Cost Request Caching
  - Although the cache is an implementation detail, it does allow us to obtain useful information about what index configurations each algorithm tries

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%

MAGIC MIRROR IN MY HAND, WHICH IS THE BEST IN THE LAND?
VLDB 2020

FURTHER DIMENSIONS

- Index width threshold
  - Extend is the only algorithm that can handle w >= 4

Benchmark	Dataset	Relations	Attributes	Queries	Relevant n-column candidates			
					$n=1$	$n=2$	$n=3$	$n=4$
TPC-DS	Synthetic skewed	24	429	99	248	3 734	68 052	1 339 536

Table 4: Cost request timings including index simulation for two TPC-DS queries; DNF exceeds 30min.

Index width	Relevant indexes		Time	
	Query 13	Query 64	Query 13	Query 64
1 column	22	49	13ms	12ms
2 columns	132	287	97ms	44ms
3 columns	870	1 889	33s	5s
4 columns	5 910	14 393	DNF	231s

IMPORTANT FINDINGS

- Different weaknesses surface in different scenarios
- Minimization Goal affects performances, especially at small / large storage constraints
- Solution Granularity depends on Workload, Approach, Budget
- Costing takes up majority of runtime for most approaches

IMPORTANT FINDINGS

- No overall "best" index selection algorithm
  - Workload
  - Storage Budget
  - DBMS
  - Runtime constraints

TODAY'S AGENDA

Overview
Index Selection Algorithms
Methods
Results
Parting Thoughts

PARTING THOUGHTS

Main Contribution

- A platform for evaluating index selection algorithms that abstracts away the cost model and DBMS*

* If DBMS exposes interface for hypothetical indexes

Evaluate algorithms on equal footing

- Different dimensions needed to be more comprehensive
  - More workloads (Transactional)
  - Cost models which account for index maintenance
- "Fairness" in evaluation
  - Benefit vs Benefit/Storage – Apples to Oranges?
  - Workloads

REFERENCES

[0] Karl Schnaitter, Neoklis Polyzotis, and Lise Getoor. 2009. *Index interactions in physical design tuning: modeling, analysis, and applications.* Proc. VLDB Endow. 2, 1 (August 2009), 1234–1245.

[1] Rainer Schlosser and Stefan Halfpap. 2020. *A Decomposition Approach for Risk-Averse Index Selection.* In 32nd International Conference on Scientific and Statistical Database Management (SSDBM 2020).