An Experimental Study of Online Database Index Selection Using MTS

COMP90055 - Research Project Presentation

Tanzid Sultan

The Need for Online Index Selection

- <u>The simple reason why indexes are important in a database:</u>

```
SELECT * FROM Employees
WHERE City = 'x';
```

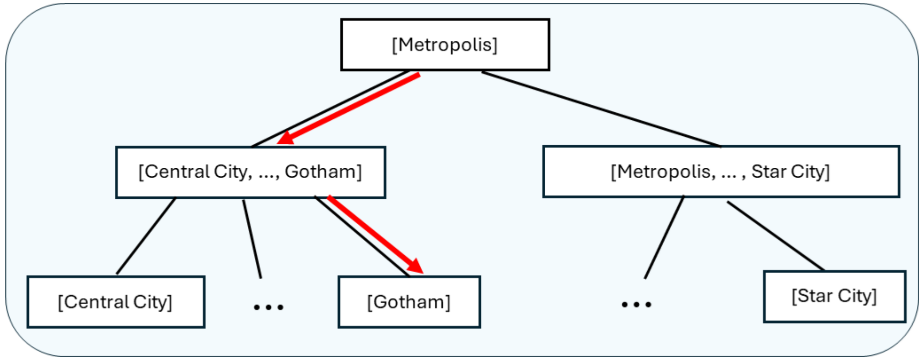
Employees Table

ID	Name	Age	City	Occupation
1	Alice	30	Gotham	Data Analyst
2	Bob	22	Metropolis	Grave Digger
3	Charlie	51	Gotham	Product Manager
…	…	…	…	…

\# of rows in table = 10^9 $\Rightarrow$ Full Table Scan = 10 minutes

\# of rows matching predicate = 120 $\Rightarrow$ Index Scan = 0.01 seconds

B+ Tree Index on City Column



1000x **speedup** with index scan!

The Need for Online Index Selection

- Just create an index on every column of every table?

- Don't have infinite memory and time! (specially when database already large and complex)

- Also, what about **dynamic** workloads and system environments?
  - a static index configuration won't be best all the time (need to adapt)
  - memory budget might vary over time (need to adapt)
  - hand selection by human admin is impractical (need to automate)

- **Solution** → An **Online Algorithm** which **automatically** creates and drops indexes based on evolving workloads and system needs

Online Index Selection Problem Formulation and MTS

- Queries $\boldsymbol{q_i}$ arrive sequentially → at each step algorithm decides on changing the index configuration (creating/dropping indexes) → then executes $\boldsymbol{q_i}$

- Objective is to minimize the ***total workload cost:***

$$C_{tot}(\text{Workload}) = \sum_{i=1}^{n} \underbrace{C_{tr}(s_{i-1}, s_i)}_{\textit{transition cost}} + \underbrace{C_{exe}\,(q_i, s_i)}_{\textit{execution cost}}$$

- Decision has to be made using **limited knowledge**: only workload/queries seen so far

- One-to-One correspondence with the problem of ***metrical task systems*** (Borodin et. al. 1992):

Task ↔ Query
State ↔ Index Configuration

WFA and WFIT for Online Index Selection

- The ***Work Function Algorithm (WFA) is*** an **optimal deterministic algorithm** for MTS (Borodin et. al. 1992), i.e. achieves lowest possible competitive ratio upper-bound.

- WFA *mimics the* optimal offline algorithm but uses only limited information.

- Should we use WFA for online index selection?

Pros	Cons
• Simple algorithm • Strong performance guarantee	• Exponentially large configuration space • Computationally intractable

- Stable partitions to the Rescue! **WFIT- Divide and Conquer** (Schnaitter and Polyzotis, 2012)

A Major Flaw of WFIT & Our Research Goals

- A **major weak-point** in WFIT → uses internal cost model

(Internal cost model means cost model of the DB query optimizer, accessed through ***what-if interface***.)

- What's wrong with using *what-if*?
  - query optimizer's **cost estimates are highly error-prone**
  - high overhead of **invoking what-if interface makes WFIT too slow**

- Research goals:

  0. Leverage WFIT for online index selection
  1. **Extend WFIT - replace what-if with a lightweight external cost model**
  2. **Benchmark WFIT against current state-of-the-art MAB**

A light-weight external cost model for access path prediction

- Difficulty in training ***end-to-end ML cost model*** (feature engineering, sample inefficiency, model complexity, offline training, etc.)

- Easier **to focus directly on access path prediction** → more relevant for index selection in large disk-based databases.

- Start with **simple heuristic-based approach**:

> Input: *query*, *configuration*
>
> ***for*** *each table*:
> - **enumerate all possible access paths** (i.e. full table scan + possible index scans)
> - **estimate the disk I/O cost for each access path** (heuristic-based)
> - **pick cheapest access path**

- This **works surprisingly well**! (comparable to *what-if* in accuracy + 500x faster)

Heuristic-based Selectivity Estimation

- Disk I/O cost for index scans derived using **selectivity estimates**, i.e. **fraction of rows** that need to be retrieved based on predicates.

- Example:

```
SELECT * FROM Employees
WHERE City = 'x'
AND Occupation = 'y';
```

Employees Table

ID	Name	Age	City	Occupation
1	Alice	30	Gotham	Data Analyst
2	Bob	22	Metropolis	Grave Digger
3	Charlie	51	Gotham	Product Manager
...	...	...	...	...

$$\text{selectivity} = P(\text{City} = \text{x},\ \text{Occupation} = \text{y}) \approx P(\text{City} = \text{x})\ P(\text{Occupation} = \text{y})$$

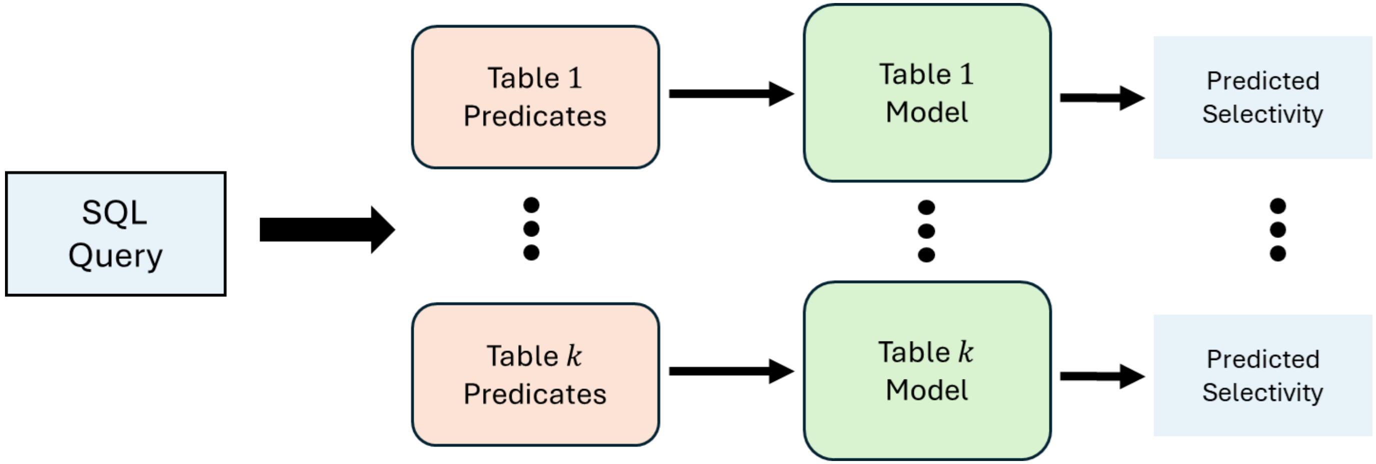
- **Attribute independence assumption** → often **violated due to correlations**

- **Single attribute distributions**, e.g. $P(\text{City})$, from **database catalog statistics**→ often **inaccurate/outdated or unavailable**

Towards Better Selectivity Estimation I: Learned Selectivity Prediction

- **Learning** to perform **selectivity prediction for equality/range predicates**.

- A **separate model for each table**.



<u>This approach is powerful because</u>:

- **online** learning → more **accurate/up-to-date statistics**
- can handle **joint selectivities over attributes** → learn **correlations**, no independence assumption
- simple prediction task, simple features → **lightweight** model can be used

Towards Better Selectivity Estimation II: Learned CDFs

- **Learning** the **cumulative distribution function** (CDF) of table attributes via **monotonic regression.**

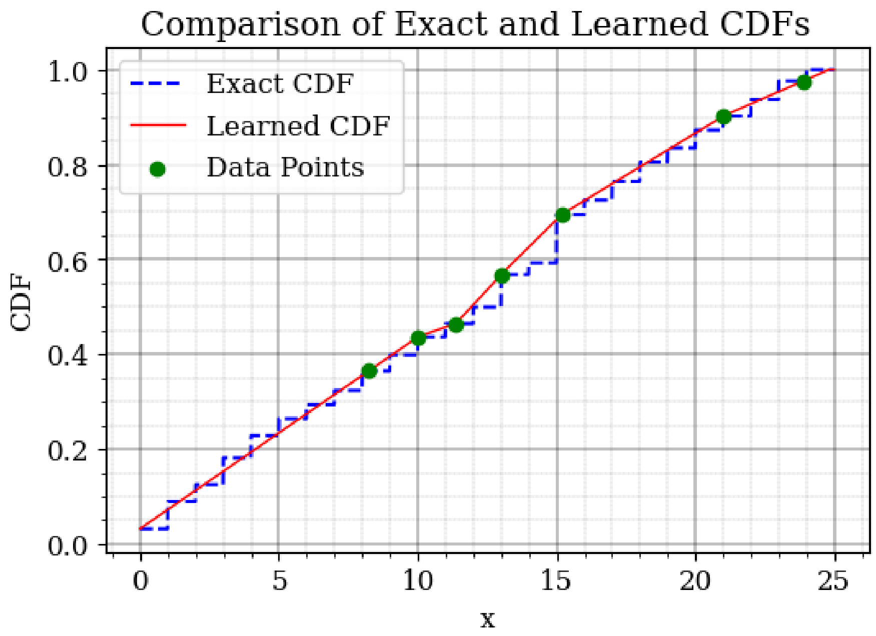
$$CDF_A(x) = P(A \leq x)$$

- **Selectivity can be derived using the CDF**, e.g. for this range predicate $\{a_1 \leq A \leq a_2\}$:

$$\text{selectivity} = P(a_1 \leq A \leq a_2) = CDF_A(a_2) - CDF_A(a_1)$$

<u>This approach is powerful because</u>:

- **online** learning
- **high sample efficiency**
- handle **joint selectivities** via **multidimensional monotonic regression**
- **very lightweight**



Preliminary Results - Benchmarking WFIT vs. MAB

- Experimental environment:
  - Linux Ubuntu OS, 20GB RAM, Intel Core-i7 16 cores, 1TB SSD
  - PostgreSQL + HypoPG extension

- Datasets
  - SSB, TPC-H, TPC-H skew (all SF 10)

- Workload Types - Static, Dynamic Shifting, Dynamic Random

- Query execution with cold cache (OS and PG buffer caches empty at start of each query execution)

Experiment:
- SSB Dataset
- Static Workload - 10 templates
- 8 rounds

	WFIT + what-if	**WFIT + ext.**	**MAB**	**No Index**
$C_{tot}(W)$	$583\ s$	$\mathbf{507\ s}$	$1021\ s$	$2258\ s$
$C_{tr}(W)$	$\mathbf{140\ s}$	$152\ s$	$455\ s$	$0\ s$
$C_{exe}(W)$	$392\ s$	$\mathbf{355\ s}$	$566\ s$	$2258\ s$

Conclusions & Future Directions

- WFIT algorithm holds up well against current SOTA MAB!
- External cost model significantly faster (500x) than what-if interface.
- Learned selectivity estimation may be a promising approach.

- However, still have room for improvement:
  - Extend access path enumeration to incorporate multiple index scans for a single table
  - Extend to multi-attribute joint selectivity prediction and multidimensional CDF
  - Scope for speeding up WFIT, certain aspects are “embarrassingly parallel” → multi-process parallelization

- On experimental side:
  - Need to benchmark on larger real-world datasets with highly skewed data, e.g. imbd
  - Use more diverse, complex and larger workloads
  - Design experiments that test ability of learned models to handle joint attributes with high correlation