Database Learning: Toward a Database that Becomes Smarter Over Time

Yongjoo Park
Ahmad Shahab Tajik
Michael Cafarella
Barzan Mozafari

University of Michigan, Ann Arbor
Today’s databases

Users

Database

After answering queries, the work is gone.

Our Goal: reuse the work.
Today’s databases

Users → query → Database

Answer to query

After answering queries, THE WORK is GONE.

Our Goal: reuse the work
Today’s databases

Users

Database

After answering queries, the work is gone.

Our Goal: reuse the work.
Today’s databases

Users

Answer to query

Database

After answering queries, the work is gone.

Our Goal: reuse the work
Today’s databases

After answering queries,
THE WORK is GONE.
Today’s databases

After answering queries, 
THE WORK is GONE.

Our Goal: reuse the work
Our high-level approach

Users

AQP engine
Our high-level approach
Our high-level approach

Users → A (10% err, 1 sec) → AQP engine
Our high-level approach

- Users
- Database Learning
- Query Synopsis
- AQP engine

(10% err, 1 sec)

(2% err)
Our high-level approach
Our high-level approach
Our high-level approach
Our high-level approach

Query Synopsis

Database Learning

AQP engine

Users

Q

Â (2% err)

Q

A (10% err)
Our high-level approach

- **Users**
  - Query Synopsis
  - Database Learning
  - AQP engine

- AQP (10% err)
- A (10% err)
- \( \hat{A} \) (2% err)

![Graph showing error and time](image)

AQP engine
Database learning
Our high-level approach

**Diagram Description:**
- **Users** provide queries (Q).
- **Query Synopsis** generates an approximation (A) with 2% error.
- **Database Learning** adjusts the approximation based on feedback (A' with 10% error).
- The adjusted approximation is then used by the **AQP engine**.

**Graph:**
- Time (sec) on the x-axis.
- Error (%) on the y-axis.
- The error decreases over time, with a significant reduction from 10% to 2% error within 12 seconds.

**Key Points:**
- The AQP engine continuously learns and refines queries to reduce error.
- Database learning adapts based on user queries and feedback loops.
Our high-level approach

Query Synopsis

Users → AQP engine

Database Learning

A (10% err) → AQP engine

\[ Q \rightarrow \hat{A} \text{ (2\% err)} \]

Error(%)

1 2 3 4 5 6 7 8 9 10 11 12

Time (sec)

AQP engine

Database learning

AQP engine

Database learning
## Technical challenges

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Technical challenges

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Queries use the data in different columns/rows.

How to leverage those queries for future queries?
Technical challenges

Queries use the data in different columns/rows. How to leverage those queries for future queries?
Technical challenges

Queries use the data in different columns/rows.
Technical challenges

Queries use the data in different columns/rows.

How to leverage those queries for future queries?
Our idea

\[ \text{(Q1; A1)} \]

\[ \text{(Q2; A2)} \]

more queries and answers...
Our idea

Q1

more queries and answers...
Our idea

(Q1, A1)
Our idea

(Q1, A1)
Our idea

Q2

\[
\begin{array}{ccc}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{array}
\]

...
Our idea

(Q2, A2)

\[
\begin{array}{cccc}
\vdots & \vdots & \vdots & \cdots \\
\vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots \\
\end{array}
\]
Our idea

(Q2, A2)

more queries and answers...
Our idea

more queries and answers
Concrete example
Concrete example

Week Number

SUM(count)

1 20 40 60 80 100

True data

Ranges observed by past queries

Model (with 95% confidence interval)
Concrete example

![Graph showing true data and ranges observed by past queries.]

- **Week Number**
  - 1, 20, 40, 60, 80, 100

- **SUM(count)**
  - 20M, 30M, 40M

**True data**

**Ranges observed by past queries**
Concrete example
Concrete example

Model (with 95% confidence interval)

True data

Ranges observed by past queries

SUM(count)
Concrete example
Design goals

1. Support a **wide class of SQL queries**
Design goals

1. Support a wide class of SQL queries

2. No Assumptions about Data
Design goals

1. Support a **wide class of SQL queries**

2. **No Assumptions** about Data

3. **Lightweight**
Our Approach
Problem statement

Problem:
Given past queries ($q_1; \ldots; q_n$), a new query ($q_{n+1}$), and their approximate answers, find the most likely answer to the new query ($q_{n+1}$) and its estimated error.

Our result:
Under a certain model assumption, our answer's error bound is original answer's error bound (in practice, much more accurate) if the error bounds provide the same probabilistic guarantees.
Problem:

Given past queries \((q_1, \ldots, q_n)\), a new query \((q_{n+1})\), and their approximate answers, find the most likely answer to the new query \((q_{n+1})\) and its estimated error.

Our result: Under a certain model assumption, our answer's error bound is much more accurate than the original answer's error bound if the error bounds provide the same probabilistic guarantees.
Problem: Given past queries \((q_1, \ldots, q_n)\), a new query \((q_{n+1})\), and their approximate answers, find the most likely answer to the new query \((q_{n+1})\) and its estimated error.

Our result: Under a certain model assumption, our answer’s error bound \(\leq\) original answer’s error bound (in practice, much more accurate) if the error bounds provide the same probabilistic guarantees.
Overview of our technique

```
select avg(Y2)
from t
where 6 < X1 < 8;
```
Overview of our technique

```
select sum(Y2)
from t
where 5 < X1 < 8;
```
Overview of our technique

select \text{sum}(Y2) 
from t 
where 5 < X1 < 8;

select \text{avg}(Y2) 
from t 
where 6 < X1 < 8;

select \text{sum}(Y2) 
from t 
where 5 < X1 < 8;

Random variables
(our uncertainty on answers)

$\theta_1, \theta_2, \theta_3$
Overview of our technique

```
select sum(Y2)
from t
where 5 < X1 < 8;
```

Random variables
(our uncertainty on answers)

\[ \theta_1, \theta_2, \theta_3 \]

\[ Pr(\theta_1, \theta_2, \theta_3) \]

Probability distribution

Two aggregations involve common values!

Correlation between answers
Overview of our technique

Two aggregations involve common values
→ correlation between answers
Overview of our technique

Two aggregations involve **common values**

→ **correlation** between answers
How to define random variables

We define a random variable for every combination of:

```sql
select X3,
    avg(Y1),
    sum(Y2)
from t
where 5 < X1 < 8
    and X2 between Apr and May
group by X3;
```

What if your query is complex?
How to define random variables

We define a random variable $\theta$
for every combination of:

```sql
select sum(Y2) from t where 5 < X1 < 8;
```
How to define random variables

We define a random variable $\theta$ for every combination of:

```
select sum(Y2)
from t
where 5 < X1 < 8;
```

Aggregate function

What if your query is complex?
How to define random variables

We define a random variable $\theta$ for every combination of:

```sql
select sum(Y2)
from t
where 5 < X1 < 8;
```

Aggregate function

Selection predicates
How to define random variables

We define a random variable $\theta$ for every combination of:

```sql
select sum(Y2)
from t
where 5 < X1 < 8;
```

Aggregate function
Selection predicates

What if your query is complex?

```sql
select X3, avg(Y1), sum(Y2)
from t
where 5 < X1 < 8
  and X2 between Apr and May
group by X3;
```
How to define random variables

We define a random variable $\theta$ for every combination of:

```sql
select sum(Y2)
from t
where 5 < X1 < 8;
```

What if your query is complex?

```
select X3, { avg(Y1), sum(Y2) }
from t
where 5 < X1 < 8
  and X2 between Apr and May
group by X3;
```
How to determine the probability distribution

The Principle of Maximum Entropy (ME)

The Principle of Maximum Entropy (ME)
How to determine the probability distribution

Statistical Info of $(\theta_1, \theta_2, \theta_3)$

The Principle of Maximum Entropy (ME)

Most-likely

Low Amount of Info

High Amount of Info

Simple

Pr

Complex

Pr

Fast Inference

Low-fidelity

Slow Inference

High-fidelity

Our choice:

(co)variances between pairs of answers.
How to determine the probability distribution

Statistical Info of \((\theta_1, \theta_2, \theta_3)\)  \rightarrow  The Principle of Maximum Entropy (ME)  \rightarrow  Most-likely \(Pr(\theta_1, \theta_2, \theta_3)\)
How to determine the probability distribution

Statistical Info of ($\theta_1, \theta_2, \theta_3$) \rightarrow The Principle of Maximum Entropy (ME) \rightarrow Most-likely $Pr(\theta_1, \theta_2, \theta_3)$

Low Amount of Info \leftrightarrow High Amount of Info

Our choice: (co)variances between pairs of answers.
How to determine the probability distribution

Statistical Info of $(\theta_1, \theta_2, \theta_3)$ → The Principle of Maximum Entropy (ME) → Most-likely $Pr(\theta_1, \theta_2, \theta_3)$

Low Amount of Info ↔ High Amount of Info

Our choice: (co)variances between pairs of answers.
How to determine the probability distribution

The Principle of Maximum Entropy (ME)

Most-likely $Pr(\theta_1, \theta_2, \theta_3)$

Statistical Info of $(\theta_1, \theta_2, \theta_3)$

Low Amount of Info  High Amount of Info

Simple $Pr$  Complex $Pr$

Our choice: (co)variances between pairs of answers.
How to determine the probability distribution

Statistical Info of $(\theta_1, \theta_2, \theta_3)$ → The Principle of Maximum Entropy (ME) → Most-likely $Pr(\theta_1, \theta_2, \theta_3)$

Low Amount of Info → Simple $Pr$ → Fast Inference → Low-fidelity

High Amount of Info → Complex $Pr$ → Slow Inference → High-fidelity
How to determine the probability distribution

The Principle of Maximum Entropy (ME)

Most-likely \( Pr(\theta_1, \theta_2, \theta_3) \)

Our choice: (co)variances between pairs of answers.

Simple 11  
Fast Inference  
Low-fidelity

Complex 11  
Slow Inference  
High-fidelity

Statistical Info of \( (\theta_1, \theta_2, \theta_3) \)
Most-likely probability distribution

\[ \theta_1 \]

\[ \theta_2 \quad \theta_3 \]
Most-likely probability distribution

Statistical Information:
Mean, variances, covariances
Most-likely probability distribution

Statistical Information:
Mean, variances, covariances

Multivariate normal distribution
Most-likely probability distribution

Statistical Information:
Mean, variances, covariances

Multivariate normal distribution
Fast inference using a closed form
Benefits of database learning

Database learning vs. indexing

1. Little storage overhead
2. Without alignment
3. No upfront overhead
4. 
5. 
6. 
7. 
8. 
9. 
10. 
11. 
12. 
13.
Benefits of database learning

Database learning vs. indexing

1. Little storage overhead
Benefits of database learning

Database learning vs. indexing

1. Little storage overhead

Database learning vs. materialized view
Benefits of database learning

Database learning vs. indexing

1. Little storage overhead

Database learning vs. materialized view

2. Without alignment
Benefits of database learning

Database learning vs. indexing

1. Little storage overhead

Database learning vs. materialized view

2. Without alignment

3. No upfront overhead
Experiment
Experiment setup

1. Two systems:
   - NOLEARN: Approximate query processing engine (The longer runtime, the more accurate answer)
   - VERDICT: Our database learning system (on top of NOLEARN)
1. Two systems:
   • **NOLEARN**: Approximate query processing engine (The longer runtime, the more accurate answer)
   • **VERDICT**: Our database learning system (on top of NOLEARN)

2. Datasets:
   • **Customer1**: 536GB data and query log from a customer
   • **TPC-H**: 100GB TPC-H dataset
1. Two systems:
   - **NOLEARN**: Approximate query processing engine (The longer runtime, the more accurate answer)
   - **VERDICT**: Our database learning system (on top of NOLEARN)

2. Datasets:
   - **Customer1**: 536GB data and query log from a customer
   - **TPC-H**: 100GB TPC-H dataset

3. Environment:
   - 5 Amazon EC2 workers (**m4.2xlarge**) + 1 master
   - SSD-backed HDFS for Spark’s data loading
1. VERDICT supports a large portion of real-world queries
Our experimental claims

1. VERDICT supports a large portion of real-world queries

2. VERDICT achieves speedup compared to NOLEARN
Our experimental claims

1. VERDICT supports a large portion of real-world queries

2. VERDICT achieves speedup compared to NOLEARN

3. VERDICT works with small memory and computational overhead
## Generality of VERDICT

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Analyzed</th>
<th># Supported</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer1</td>
<td>3,342</td>
<td>2,463</td>
<td>73.7%</td>
</tr>
<tr>
<td>TPC-H</td>
<td>21</td>
<td>14</td>
<td>66.7%</td>
</tr>
</tbody>
</table>

Unsupported queries:

1. Nested queries (that cannot be flattened)
2. Textual filters:  
   ```
   city like '%arbor%
   ```
Results on the **TPC-H** dataset (the paper has the **Customer1** results)

Number of past queries fixed to 50

**Runtime-error trade-off**
Runtime-error trade-off

Results on the TPC-H dataset (the paper has the Customer1 results)

Number of past queries fixed to 50

(a) Data in Memory

(b) Data on SSD
The results on the Customer1 dataset (the paper has the TPC-H results)

(a) Data in memory

(b) Data on SSD
The results on the **Customer1** dataset (the paper has the **TPC-H** results)

**Speedup**

(a) Data in memory

<table>
<thead>
<tr>
<th>Target Error Bound</th>
<th>Speedup (x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4%</td>
<td>7.7</td>
</tr>
<tr>
<td>2%</td>
<td>2.5</td>
</tr>
</tbody>
</table>

(b) Data on SSD

<table>
<thead>
<tr>
<th>Target Error Bound</th>
<th>Speedup (x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4%</td>
<td>23</td>
</tr>
<tr>
<td>2%</td>
<td>5.7</td>
</tr>
</tbody>
</table>
1. Memory overhead:

- Queries and their answer, some matrices and their inverses
- 23.2 KB per query for the Customer1 dataset
- 15.8 KB per query for the TPC-H dataset

2. Computational overhead:

- Latency for memory
- Latency for SSD
- NoLearn: 2.083 sec, 52.50 sec
- Verdict: 2.093 sec, 52.51 sec
- Overhead: 0.010 sec (0.48%), 0.010 sec (0.02%)
1. Memory overhead:
   - Queries and their answer, some matrices and their inverses
1. Memory overhead:
   - Queries and their answer, some matrices and their inverses
   - 23.2 KB per query for the Customer1 dataset
   - 15.8 KB per query for the TPC-H dataset
Memory and computational overhead

1. Memory overhead:
   - Queries and their answer, some matrices and their inverses
   - 23.2 KB per query for the Customer1 dataset
   - 15.8 KB per query for the TPC-H dataset

2. Computational overhead:

<table>
<thead>
<tr>
<th></th>
<th>Latency for memory</th>
<th>Latency for SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOLEARN</td>
<td>2.083 sec</td>
<td>52.50 sec</td>
</tr>
<tr>
<td>VERDICT</td>
<td>2.093 sec</td>
<td>52.51 sec</td>
</tr>
<tr>
<td>Overhead</td>
<td>0.010 sec (0.48%)</td>
<td>0.010 sec (0.02%)</td>
</tr>
</tbody>
</table>
Thank You!