# Database Learning

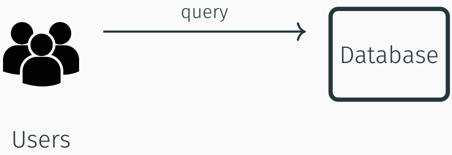
Building databases that become smarter over time

**Yongjoo Park**
Ahmad Shahab Tajik
Michael Cafarella
Barzan Mozafari

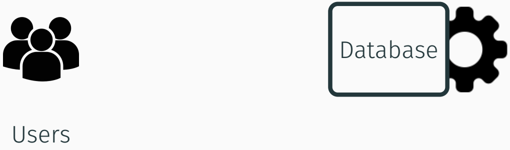
University of Michigan, Ann Arbor

# Today's Databases

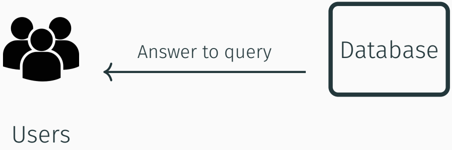
# Today's Databases



# Today's Databases



# Today’s Databases



# Today’s Databases


After answering queries,
THE WORK is almost completely WASTED.

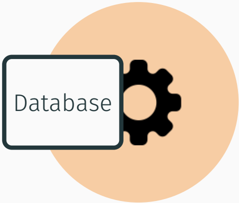
# Today's Databases


After answering queries,
THE WORK is almost completely WASTED.

Small exceptions:

# Today's Databases

Users

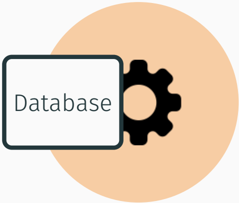



After answering queries,
THE WORK is almost completely WASTED.

Small exceptions:

- Caching

# Today's Databases

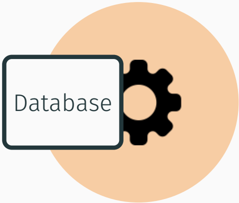




After answering queries,
THE WORK is almost completely WASTED.

Small exceptions:

- Caching
- Identical queries

# Today's Databases

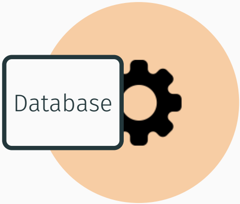

After answering queries,
THE WORK is almost completely WASTED.

Small exceptions:

- Caching
- Identical queries
- Indexing/Materialization hints

# Today’s Databases

Users




After answering queries,

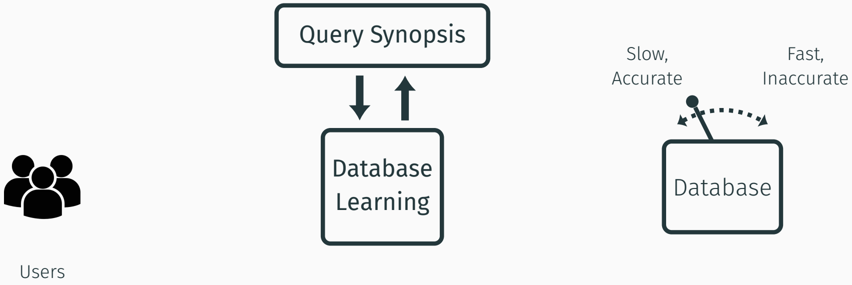
THE WORK is almost completely WASTED.

Small exceptions:

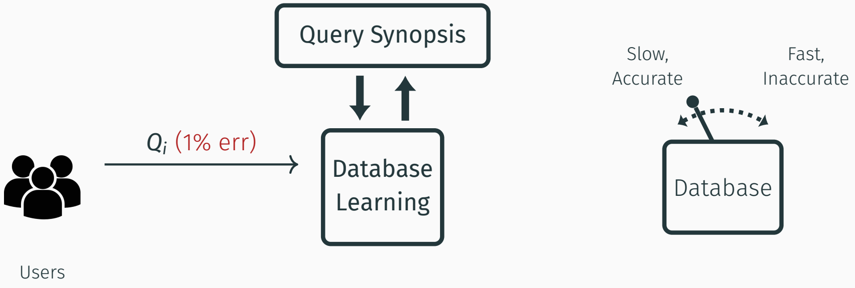
- Caching
- Identical queries
- Indexing/Materialization hints

## Our Goal: reuse the work.

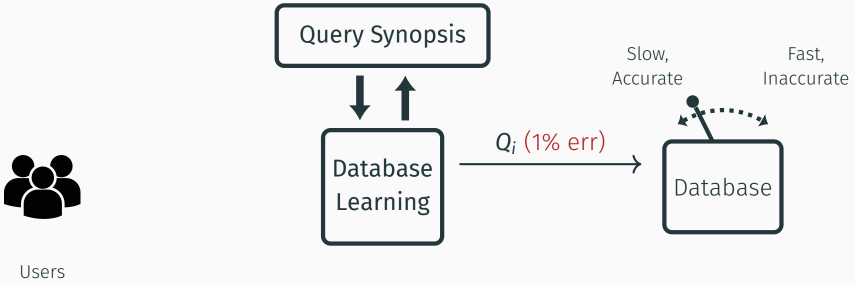
# A New Paradigm in AQP Setting



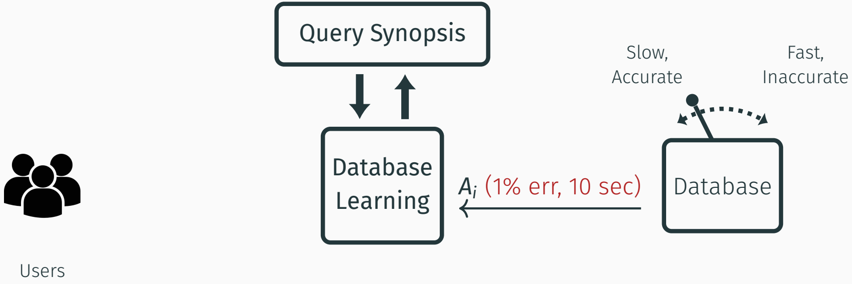
# A New Paradigm in AQP Setting



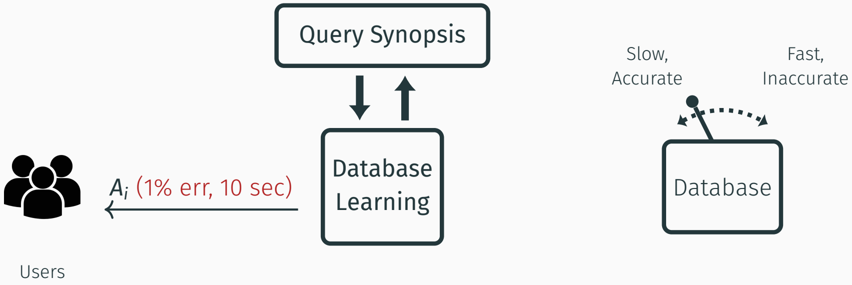
# A New Paradigm in AQP Setting



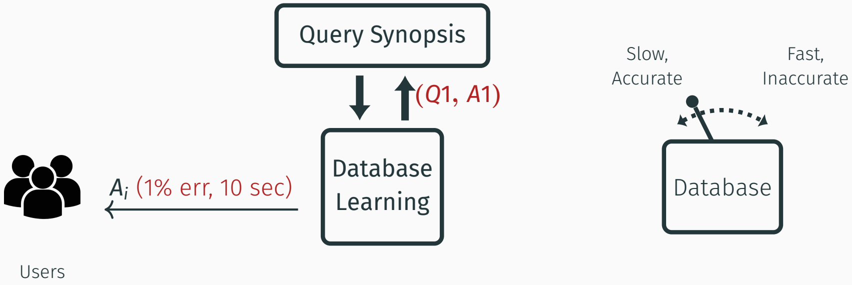
# A New Paradigm in AQP Setting



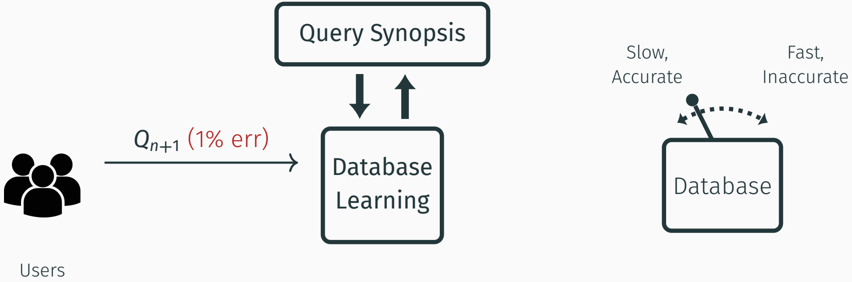
# A New Paradigm in AQP Setting



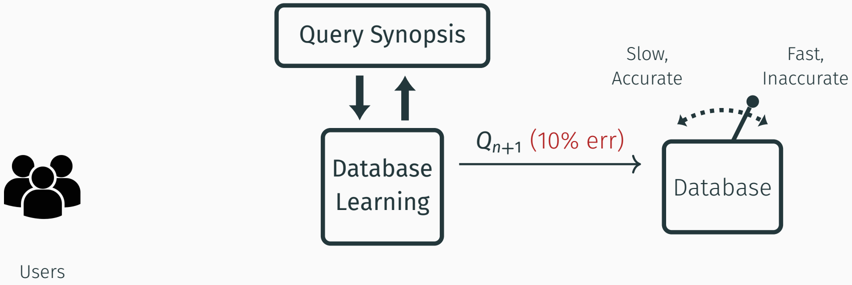
# A New Paradigm in AQP Setting



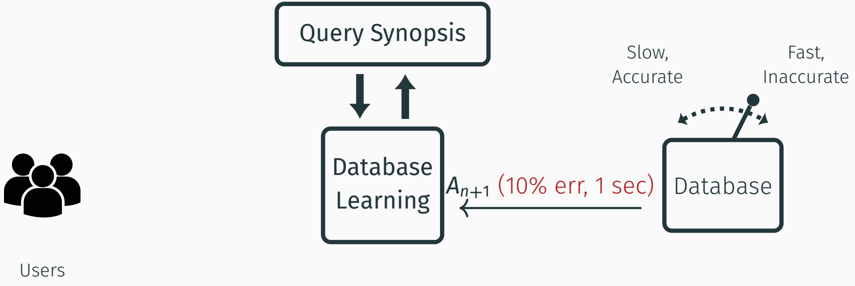
# A New Paradigm in AQP Setting



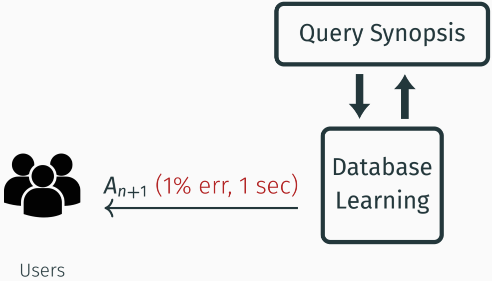
# A New Paradigm in AQP Setting

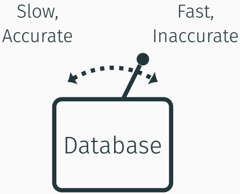


# A New Paradigm in AQP Setting

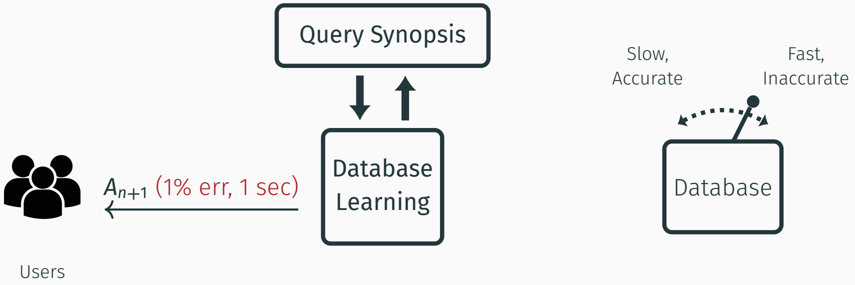


# A New Paradigm in AQP Setting



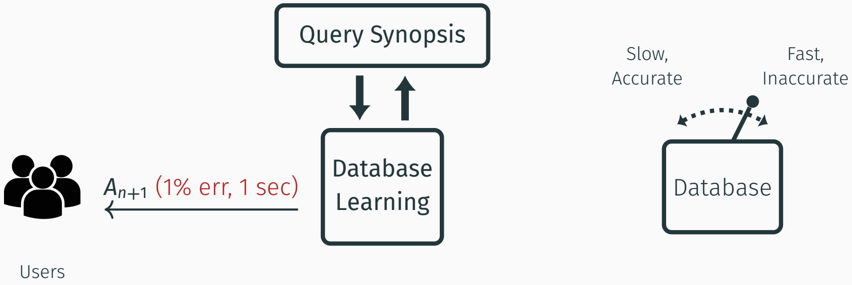



# A New Paradigm in AQP Setting



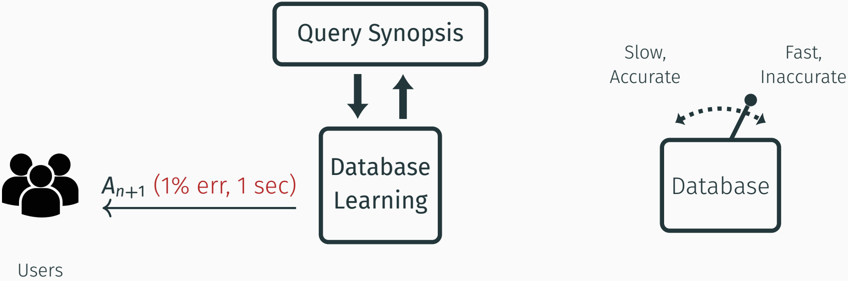

1. User: enjoys **1% error bound in 1 second!**

# A New Paradigm in AQP Setting




1. User: enjoys **1% error bound in 1 second!**
2. Formally, always more accurate

# A New Paradigm in AQP Setting




1. User: enjoys **1% error bound in 1 second!**
2. Formally, always more accurate
3. Popularity of analytic workloads $\Rightarrow$ **Approximate solutions**
   - BlinkDB, SnappyData, Yahoo Druid, Facebook Presto, Infobright, etc.

# From Machine Learning To Database Learning

Machine Learning: Past Observations $\Rightarrow$ Future Predictions

# From Machine Learning To Database Learning

| | | | |
|---|---|---|---|
| Machine Learning: | Past Observations | $\Rightarrow$ | Future Predictions |
| Database Learning: | **Past Answers** | $\Rightarrow$ | **Future Answers** |

# From Machine Learning To Database Learning

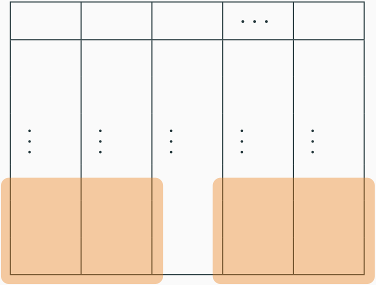
| | | | |
|---|---|---|---|
| Machine Learning: | Past Observations | $\Rightarrow$ | Future Predictions |
| Database Learning: | **Past Answers** | $\Rightarrow$ | **Future Answers** |

The more past queries, the more Accurate and Faster

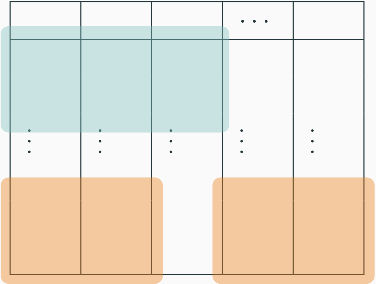
# Primary Benefit: Exploratory Workloads

| | | | ... | |
|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

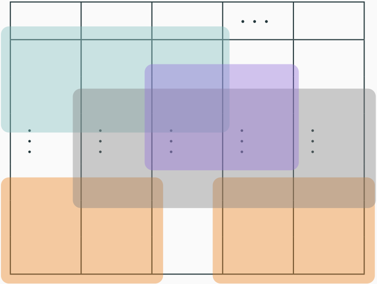
# Primary Benefit: Exploratory Workloads



# Primary Benefit: Exploratory Workloads

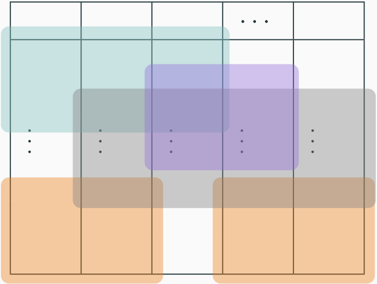


# Primary Benefit: Exploratory Workloads



Queries use the data in different columns/rows.

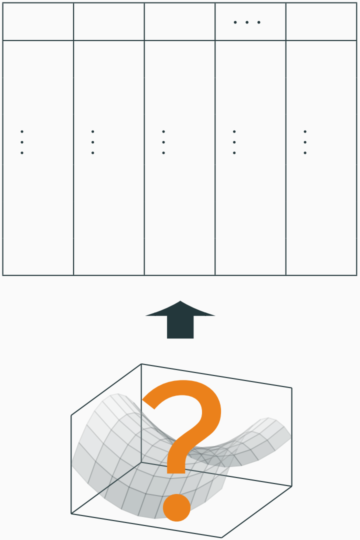
# Primary Benefit: Exploratory Workloads



Queries use the data in different columns/rows.

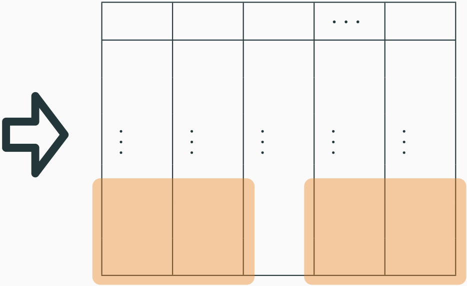
How to leverage those queries for future queries?

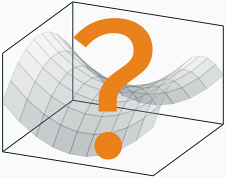
# Our Idea



# Our Idea

*Q*1

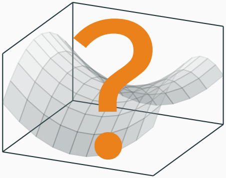




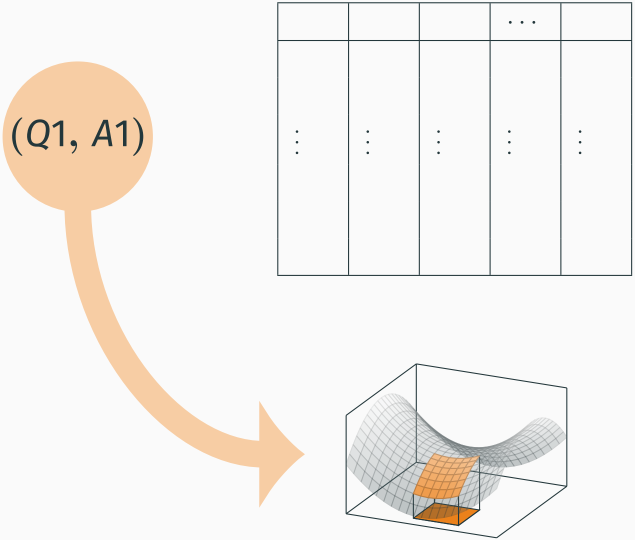
# Our Idea

$(Q1, A1)$

| | | | $\cdots$ | |
|---|---|---|---|---|
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

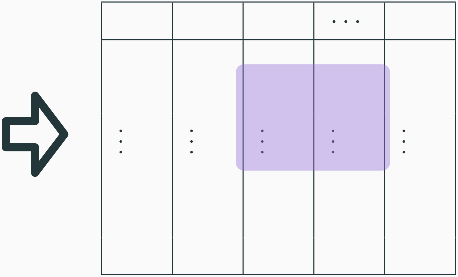


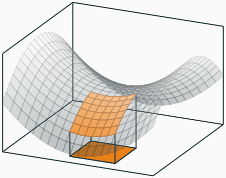
# Our Idea



# Our Idea

*Q2*

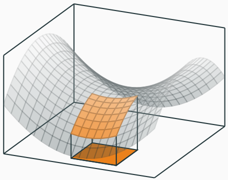




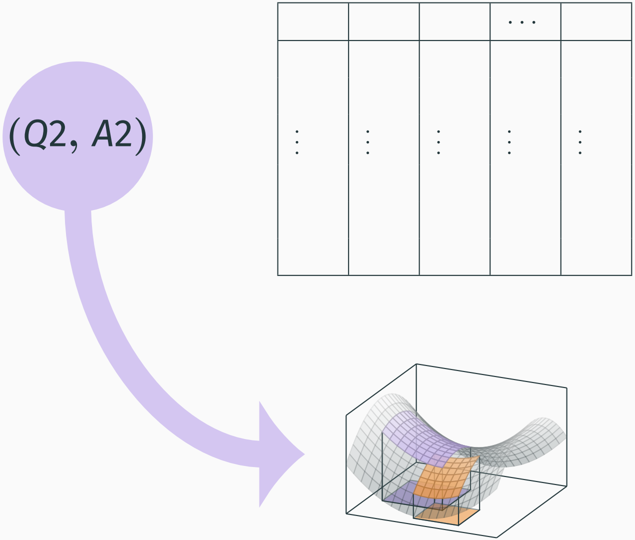
# Our Idea

$(Q2,\, A2)$

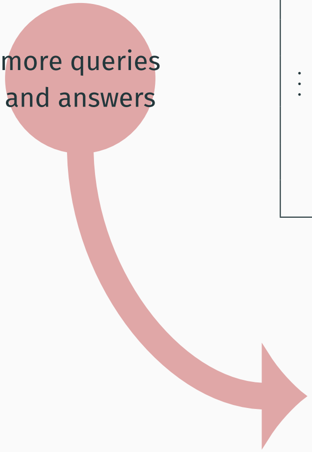
| | | | $\cdots$ | |
|---|---|---|---|---|
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |



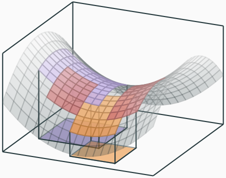
# Our Idea



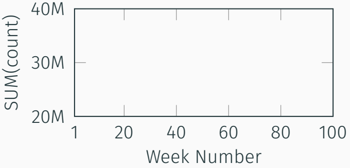
# Our Idea



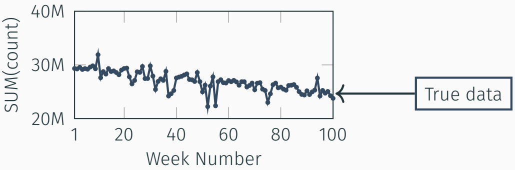

| | | | … | |
|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |



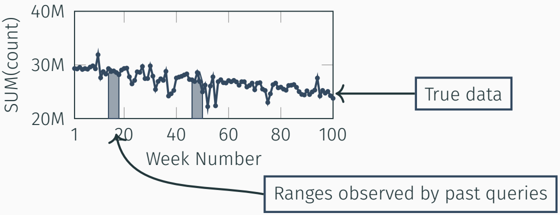
# Concrete Example



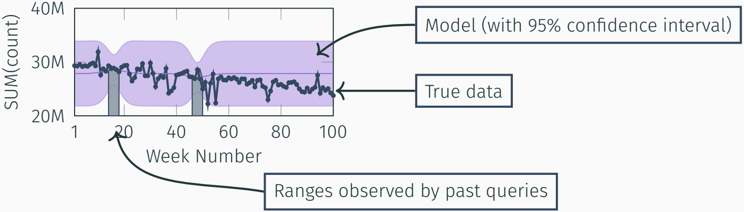
# Concrete Example



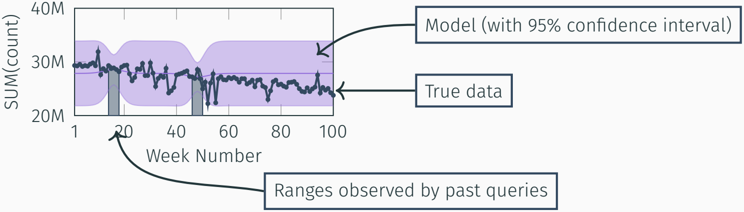
# Concrete Example

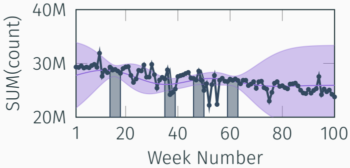


# Concrete Example

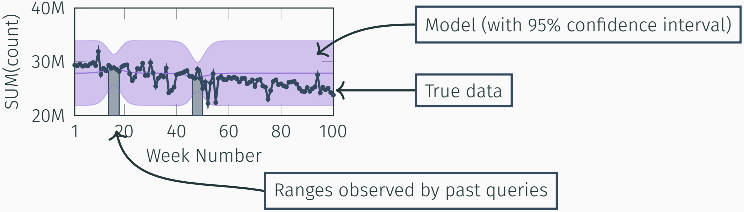


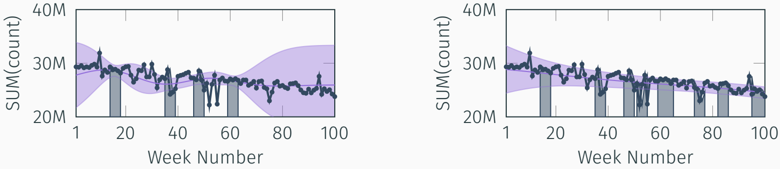
# Concrete Example



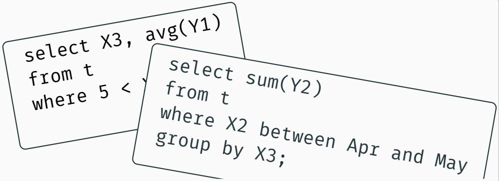



# Concrete Example



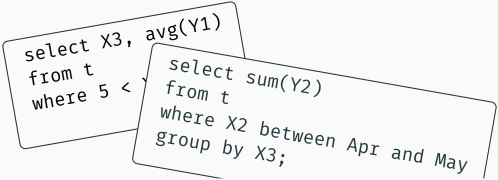



# Design Goals

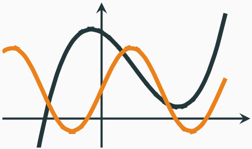



1. Support a wide class of SQL queries

# Design Goals

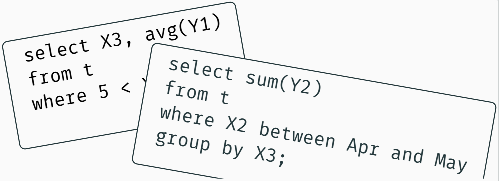



1. Support a wide class of SQL queries

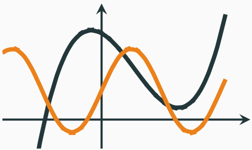


2. No Assumptions about Data

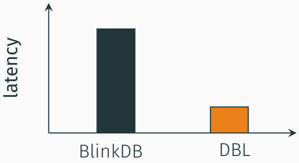
# Design Goals




1. Support a wide class of SQL queries



2. No Assumptions about Data




3. Lightweight

# Our Approach

# Problem Statement

# Problem Statement

**Problem:**

Given queries, $\{q_1, \ldots, q_{n+1}\}$, and their approximate answers,

Find the most likely answer to $q_{n+1}$ and its estimated error.

# Problem Statement

**Problem:**

Given queries, $\{q_1, \ldots, q_{n+1}\}$, and their approximate answers,

Find the most likely answer to $q_{n+1}$ 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.

# Our Model Assumption

Assumption:

Goodness of the principle of maximum entropy [?]

# Our Model Assumption

Assumption:

Goodness of the principle of maximum entropy [?]

$\rightarrow$ Not overly confident on the unobserved data.

# Our Model Assumption

Assumption:

Goodness of the principle of maximum entropy [?]

$\rightarrow$ Not overly confident on the unobserved data.

Justification:

“ *agrees with everything that is known, but carefully avoids assuming anything that is not known — [?]* ”

# Our Model Assumption

Assumption:

Goodness of the principle of maximum entropy [?]

$\rightarrow$ Not overly confident on the unobserved data.

Justification:

“ *agrees with everything that is known, but carefully avoids assuming anything that is not known — [?]* ”

We provide empirical justifications in our report [?].

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

1

Random variables
(our uncertainty on answers)

$\theta_1, \quad \theta_2, \quad \theta_3$

# Overview of Our Technique

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

① →

Random variables
(our uncertainty on answers)

$\boldsymbol{\theta}_1, \quad \boldsymbol{\theta}_2, \quad \boldsymbol{\theta}_3$

↓ ②

$Pr(\theta_1,\ \theta_2,\ \theta_3)$

Probability distribution

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

① →

Random variables
(our uncertainty on answers)

$\boldsymbol{\theta}_1, \quad \boldsymbol{\theta}_2, \quad \boldsymbol{\theta}_3$

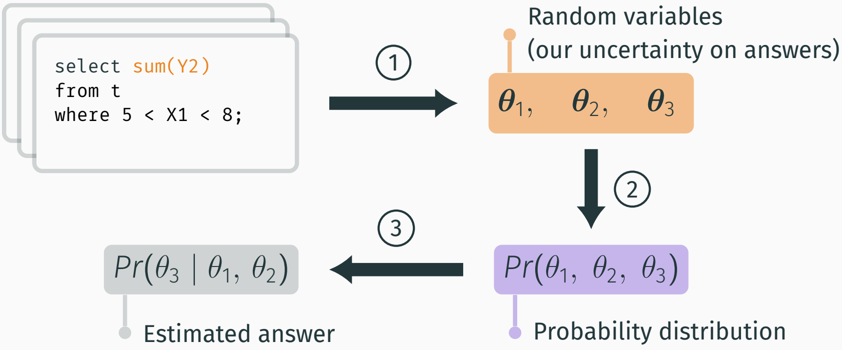
② ↓

$Pr(\theta_1,\ \theta_2,\ \theta_3)$

Probability distribution

Two aggregations involve common values
$\rightarrow$ correlation between answers

# Overview of Our Technique



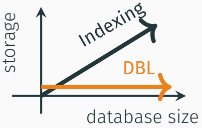

Two aggregations involve common values
$\rightarrow$ correlation between answers

# Benefits of Database Learning

Database Learning vs. Indexing

# Benefits of Database Learning

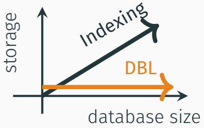
## Database Learning vs. Indexing




1. Little Storage Overhead

# Benefits of Database Learning

Database Learning vs. Indexing

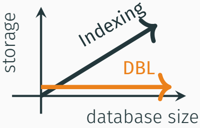



1. Little Storage Overhead

Database Learning vs. Materialized View Selection

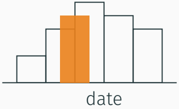
# Benefits of Database Learning

## Database Learning vs. Indexing




1. Little Storage Overhead

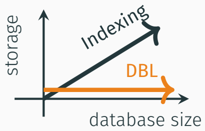
## Database Learning vs. Materialized View Selection




2. Without Alignment

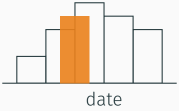
# Benefits of Database Learning

## Database Learning vs. Indexing

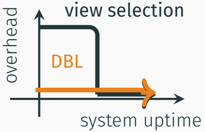



1. Little Storage Overhead

## Database Learning vs. Materialized View Selection




2. Without Alignment




3. No Upfront Overhead

# Experiment

# Implementation and Experiments

1. Using **Spark SQL** as a backend
   - NoLearn: Sampling-based AQP engine
   - Verdict: Our database learning system

# Implementation and Experiments

1. Using **Spark SQL** as a backend
   - NoLearn: Sampling-based AQP engine
   - Verdict: Our database learning system

2. Datasets:
   - `Customer1`: Query log from an analytic DB vendor
   - `TPC-H`: 100G TPC-H dataset

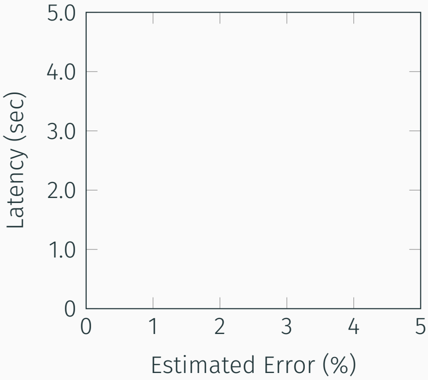
# Implementation and Experiments

1. Using **Spark SQL** as a backend
   - NoLearn: Sampling-based AQP engine
   - Verdict: Our database learning system
2. Datasets:
   - `Customer1`: Query log from an analytic DB vendor
   - `TPC-H`: 100G TPC-H dataset
3. Environment:
   - 5 Amazon EC2 workers (`m4.2xlarge`)
   - SSD-backed HDFS for Spark's data loading

# Generality of VERDICT

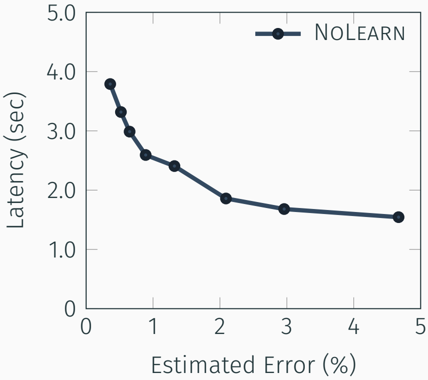
| Dataset | # Analyzed | # Supported | Percentage |
|---|---:|---:|---:|
| `Customer1` | 3,342 | 2,463 | 73.7% |
| `TPC-H` | 21 | 14 | 63.6% |

# Latency-Error Trade-off



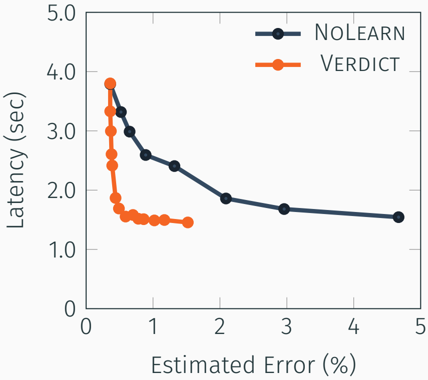

`Customer1` dataset in memory

# Latency-Error Trade-off



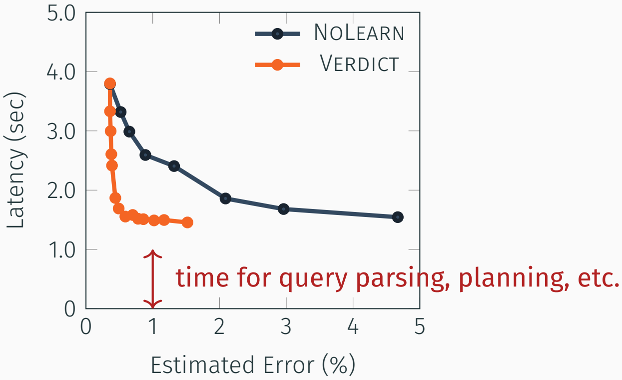

`Customer1` dataset in memory

# Latency-Error Trade-off



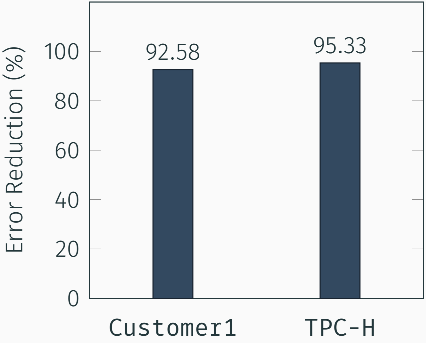

`Customer1` dataset in memory

# Latency-Error Trade-off



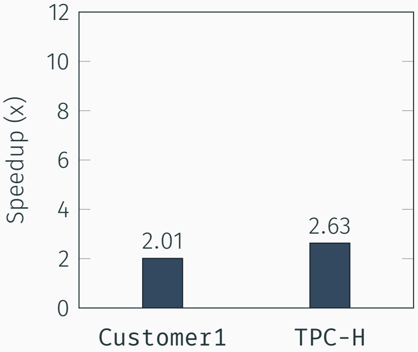

`Customer1` dataset in memory

# Error Reduction



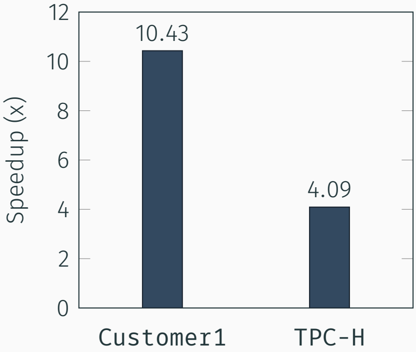

For the same time budget; Data on SSD

# Speedup (Data in Memory)



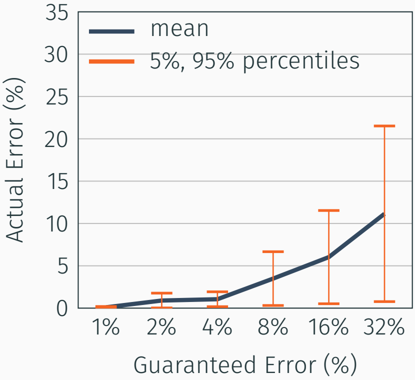

For the same target error of 2%.

# Speedup (Data on SSD)




For the same target error of 2%.

# Reliability of Estimated Error Guarantees




Guaranteed error = from 95% confidence interval

# Memory and Computational Overhead

1. Memory Overhead:
   - 150KB per query for the `Customer1` dataset
   - 8KB per query for the `TPC-H` dataset

# Memory and Computational Overhead

1. Memory Overhead:
   - 150KB per query for the `Customer1` dataset
   - 8KB per query for the `TPC-H` dataset

2. Computational Overhead:

| Latency | Cached | | No-Cache | |
|---|---|---|---|---|
| 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. Database Learning:

   *Answers to past queries* $\rightarrow$ boost your AQP!

# Conclusion and Future Work

1. Database Learning:

   *Answers to past queries* $\rightarrow$ boost your AQP!

2. Our prototype, VERDICT, demonstrated:

# Conclusion and Future Work

1. **Database Learning**:

   *Answers to past queries* $\rightarrow$ boost your AQP!

2. Our prototype, **VERDICT**, demonstrated:

   - Support 73.7% real-world analytical queries

# Conclusion and Future Work

1. Database Learning:

   *Answers to past queries* → boost your AQP!

2. Our prototype, **Verdict**, demonstrated:

   - Support 73.7% real-world analytical queries
   - Error reduction up to 95% compared to existing AQP engines

# Conclusion and Future Work

1. **Database Learning**:

   *Answers to past queries* → boost your AQP!

2. Our prototype, **VERDICT**, demonstrated:
   - Support 73.7% real-world analytical queries
   - Error reduction up to 95% compared to existing AQP engines

**Our next goal: Active Database Learning**

Aims to build a probabilistic model of data
*even before* any queries submitted

Thank You!

# References I