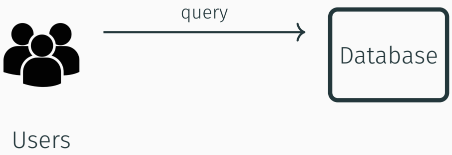
Database Learning: Toward a Database that Becomes Smarter Over Time

Yongjoo Park
Ahmad Shahab Tajik
Michael Cafarella
Barzan Mozafari

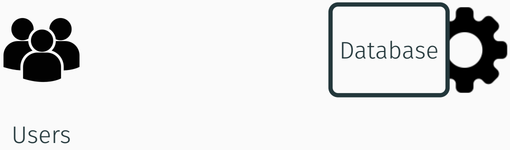
University of Michigan, Ann Arbor

Today's databases

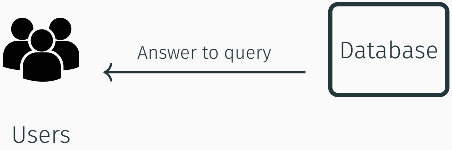
Today's databases



Today's databases



Today's databases



Today's databases

After answering queries,
THE WORK is GONE.

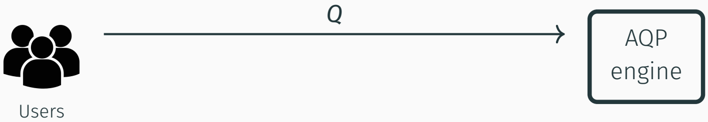
Today’s databases

After answering queries,
THE WORK is GONE.

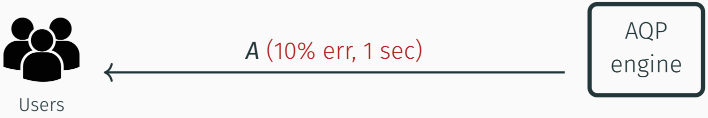
Our Goal: reuse the work

Our high-level approach

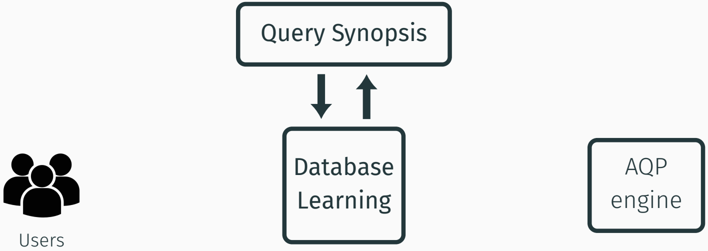
Our high-level approach



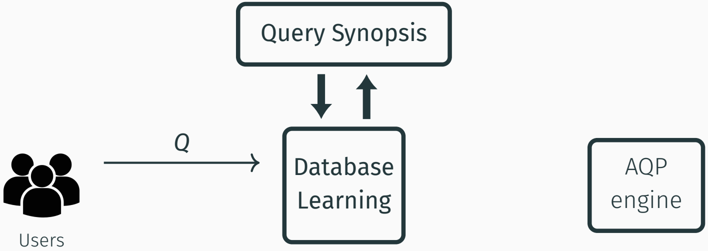
Our high-level approach



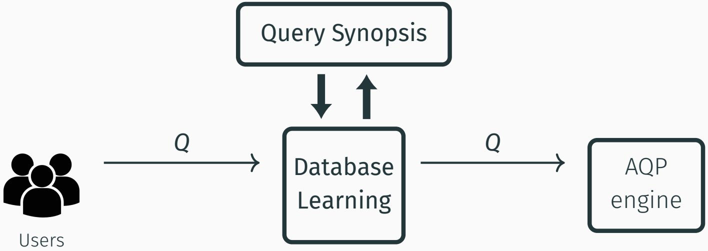
Our high-level approach



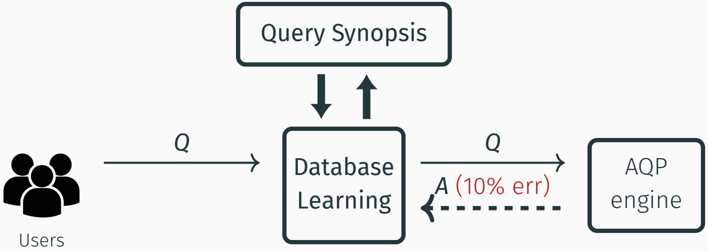
Our high-level approach



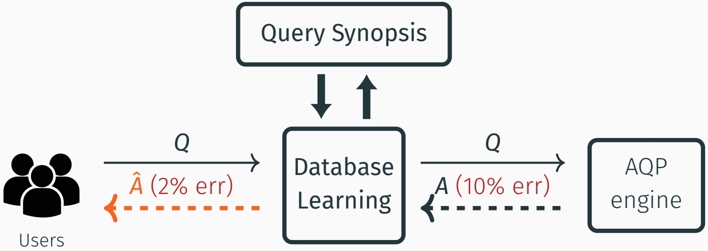
Our high-level approach



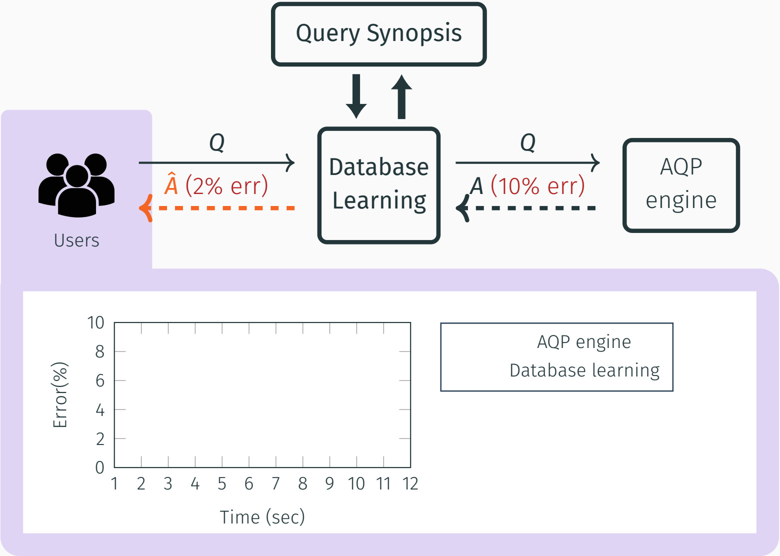
Our high-level approach



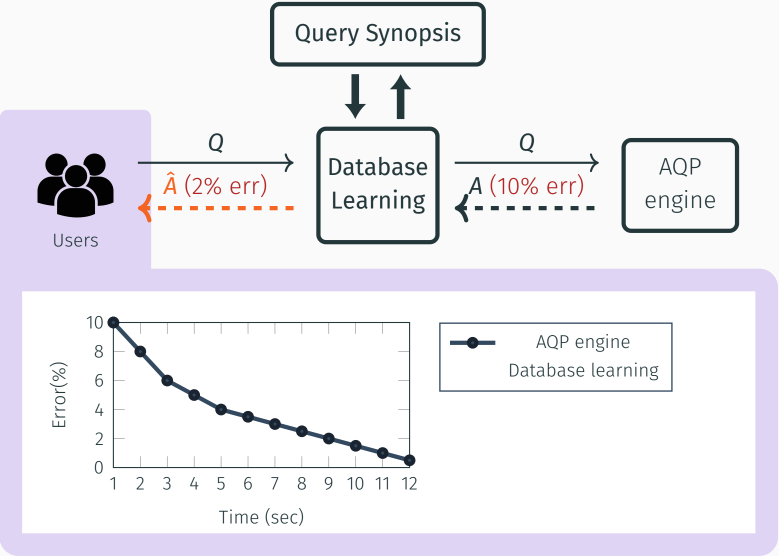
Our high-level approach



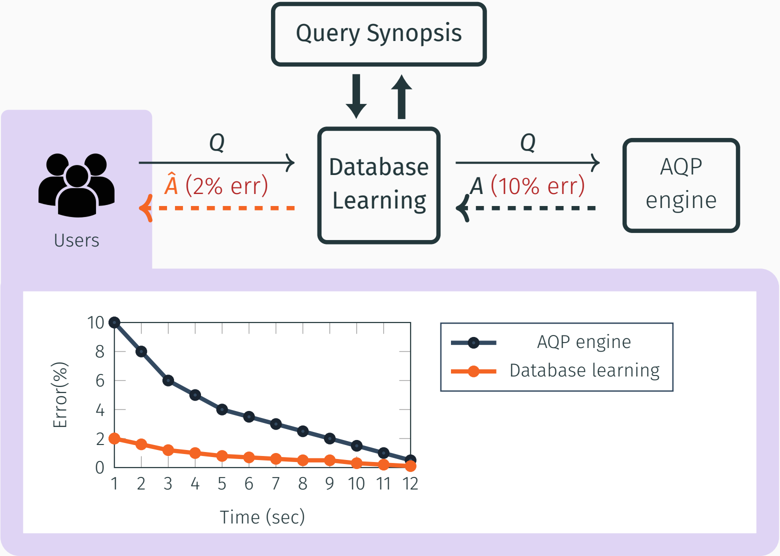
Our high-level approach



Our high-level approach



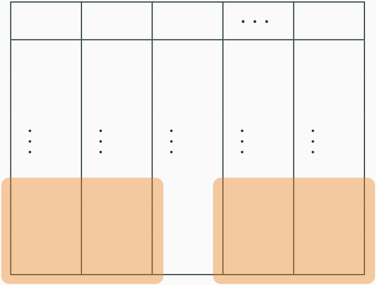
Our high-level approach



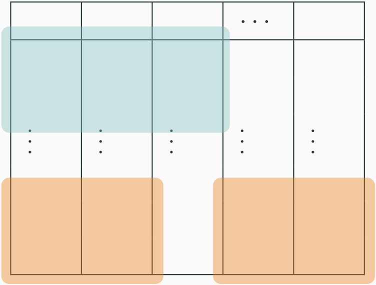
Technical challenges

			…	
⋮	⋮	⋮	⋮	⋮

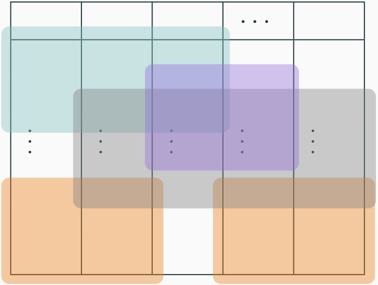
Technical challenges



Technical challenges

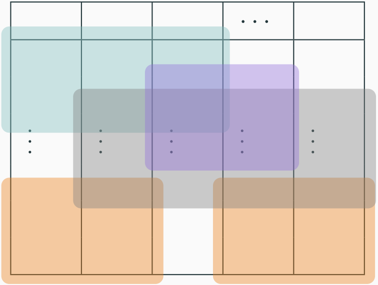


Technical challenges



Queries use the data in different columns/rows.

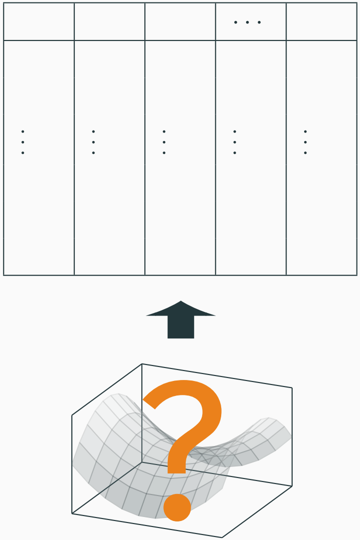
Technical challenges



Queries use the data in different columns/rows.

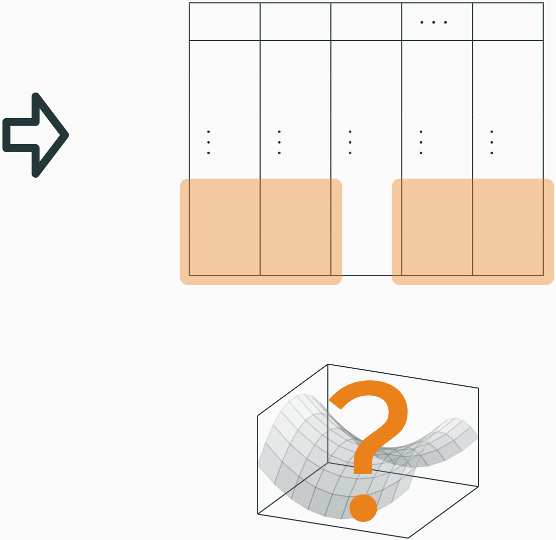
How to leverage those queries for future queries?

Our idea



Our idea

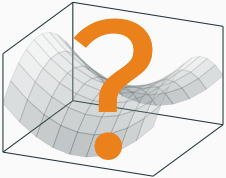
Q1



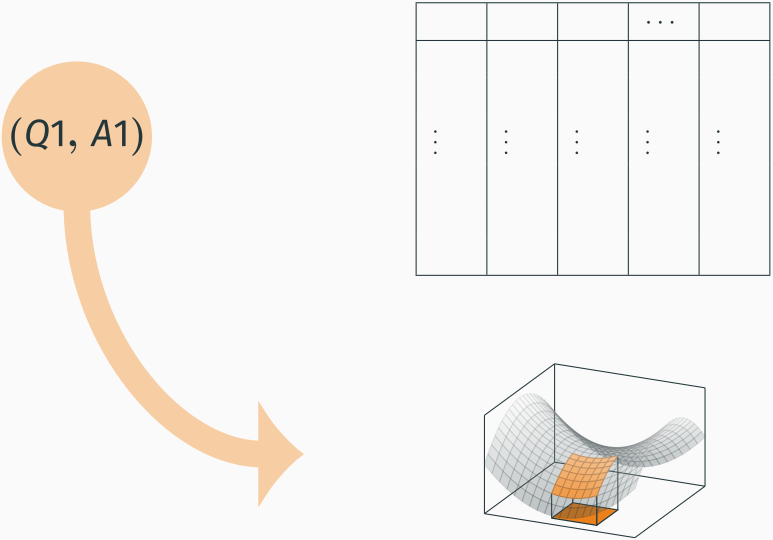
Our idea

$(Q1, A1)$

			$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

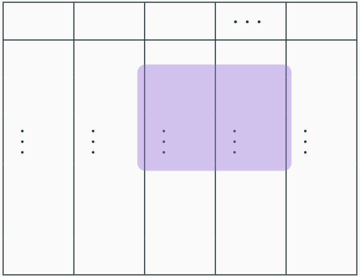


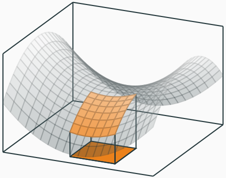
Our idea



Our idea

Q2

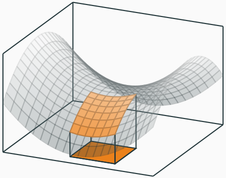




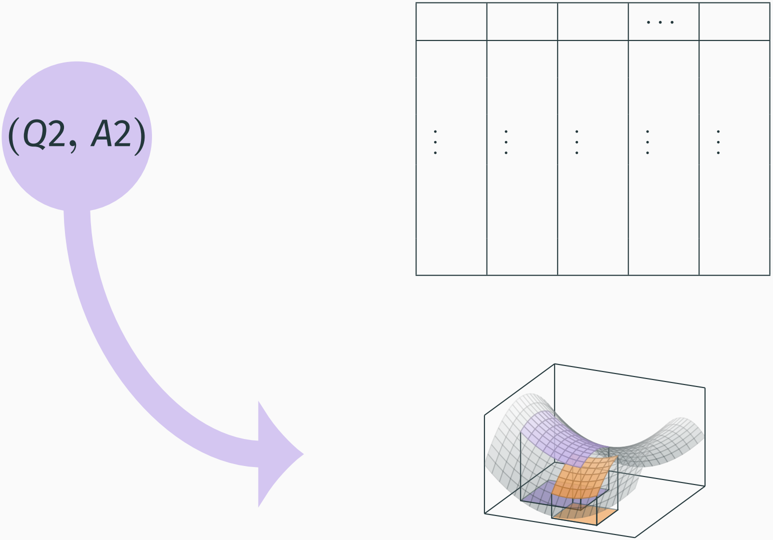
Our idea

$(Q2, A2)$

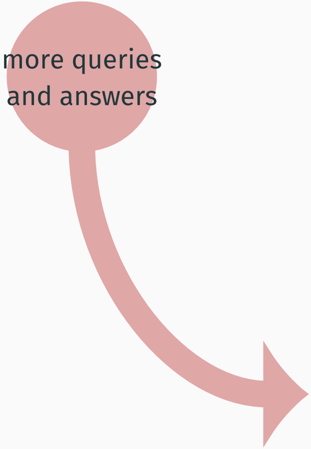
			$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$



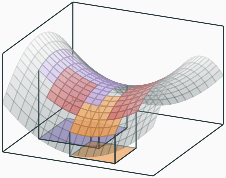
Our idea



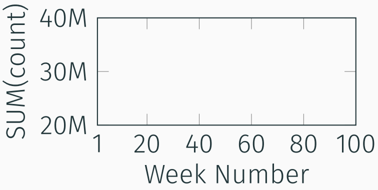
Our idea



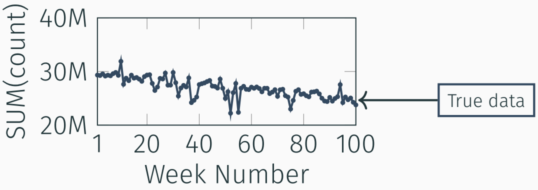
			$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$



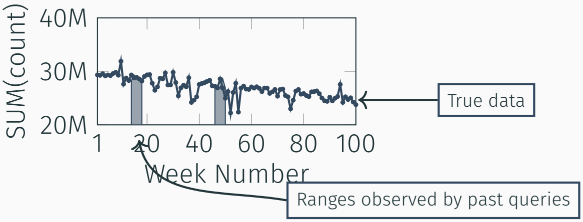
Concrete example



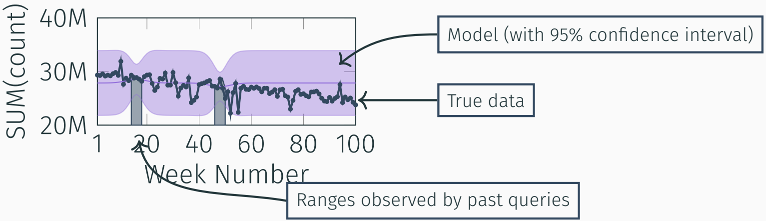
Concrete example

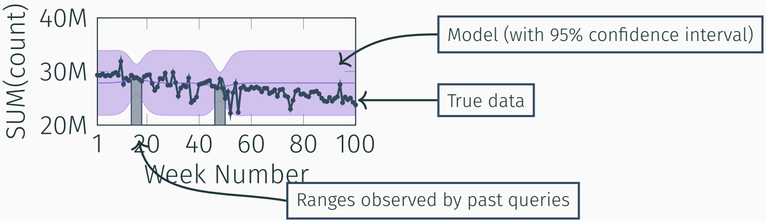


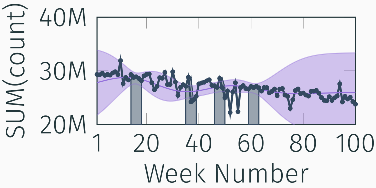
Concrete example



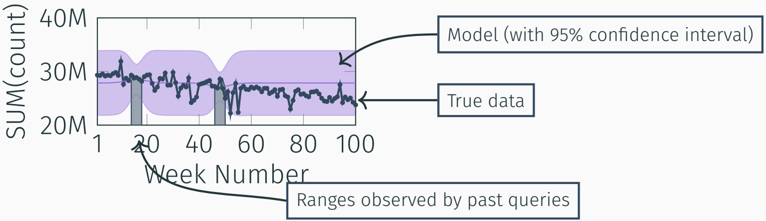
Concrete example

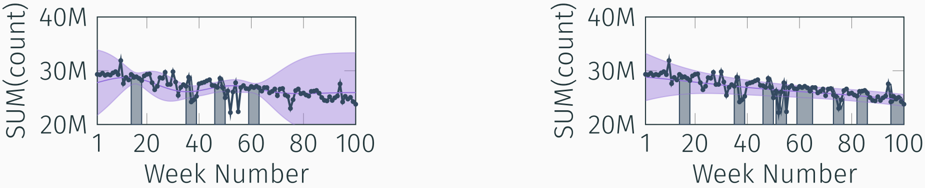



SUM(count)
40M
30M
20M
1
20
40
60
80
100
Week Number
Model (with 95% confidence interval)
True data
Ranges observed by past queries

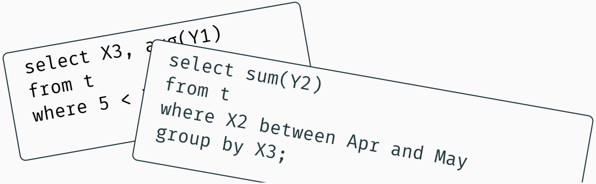

SUM(count)
40M
30M
20M
1
20
40
60
80
100
Week Number

Concrete example



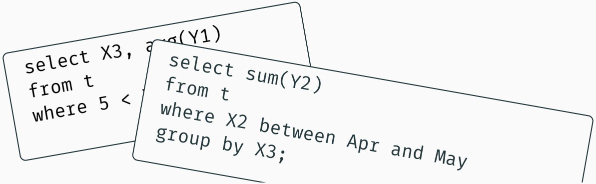


Design goals

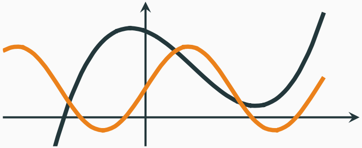


1. Support a wide class of SQL queries

Design goals

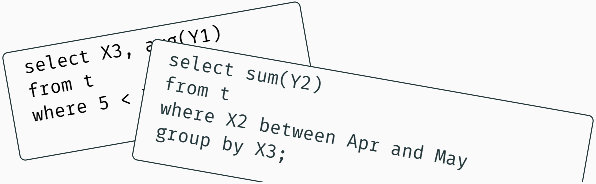


1. Support a wide class of SQL queries

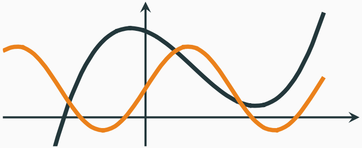


2. No Assumptions about Data

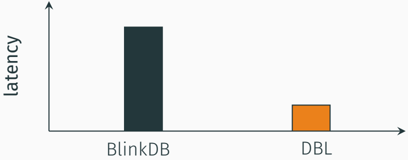
Design goals



1. Support a wide class of SQL queries



2. No Assumptions about Data



3. Lightweight

Our Approach

Problem statement

Problem statement

Problem:

Given 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.

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 $\leq$ original answer's error bound

(in practice, much more accurate)

if the error bounds provide the same probabilistic guarantees.

Overview of our technique

```sql
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

Overview of our technique

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

① →

Random variables
(our uncertainty on answers)

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

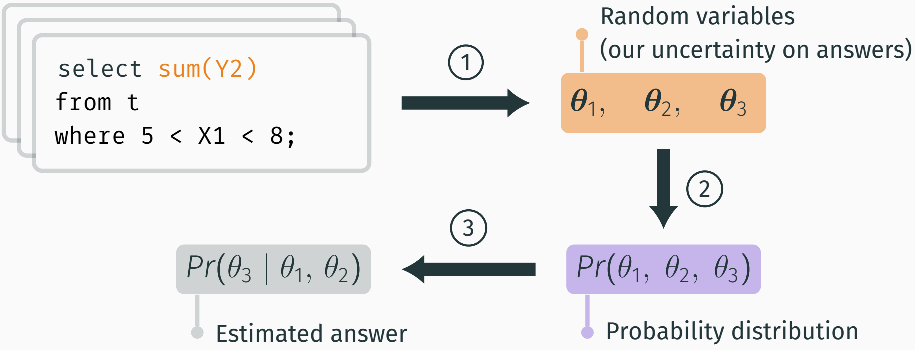
↓ ②

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

Probability distribution

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

Overview of our technique



Two aggregations involve common values

$\rightarrow$ correlation between answers

How to define random variables

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

How to define random variables

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

We define a random variable θ
for every combination of:

How to define random variables

We define a random variable θ
for every combination of:

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

- Aggregate function

How to define random variables

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

We define a random variable θ for every combination of:

- Aggregate function
- Selection predicates

How to define random variables

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

We define a random variable θ
for every combination of:

- Aggregate function
- Selection predicates

```
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

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

We define a random variable θ
for every combination of:

- Aggregate function
- Selection predicates

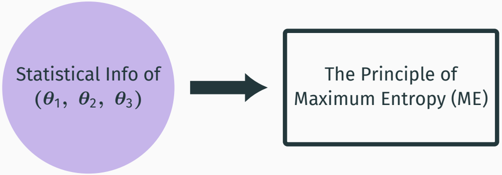
```
select X3, avg(Y1), sum(Y2)
from t
where 5 < X1 < 8
  and X2 between Apr and May
group by X3;
```

What if your query is complex?

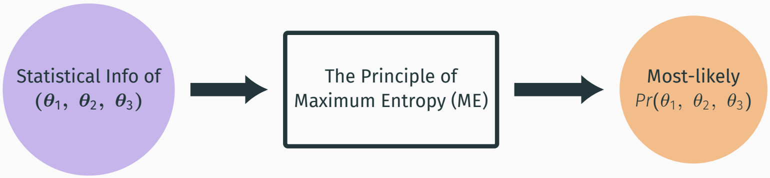
How to determine the probability distribution

The Principle of
Maximum Entropy (ME)

How to determine the probability distribution



How to determine the probability distribution

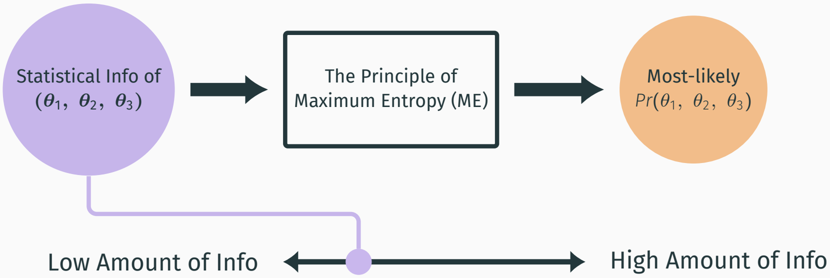


How to determine the probability distribution

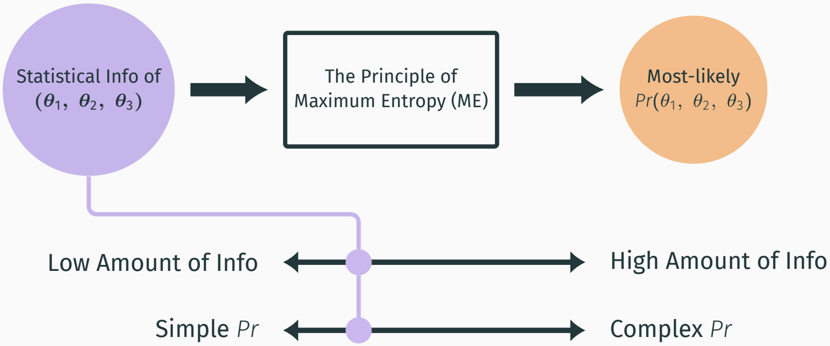
Statistical Info of $(\theta_1, \theta_2, \theta_3)$ $\rightarrow$ The Principle of Maximum Entropy (ME) $\rightarrow$ Most-likely $Pr(\theta_1, \theta_2, \theta_3)$

Low Amount of Info $\longleftrightarrow$ High Amount of Info

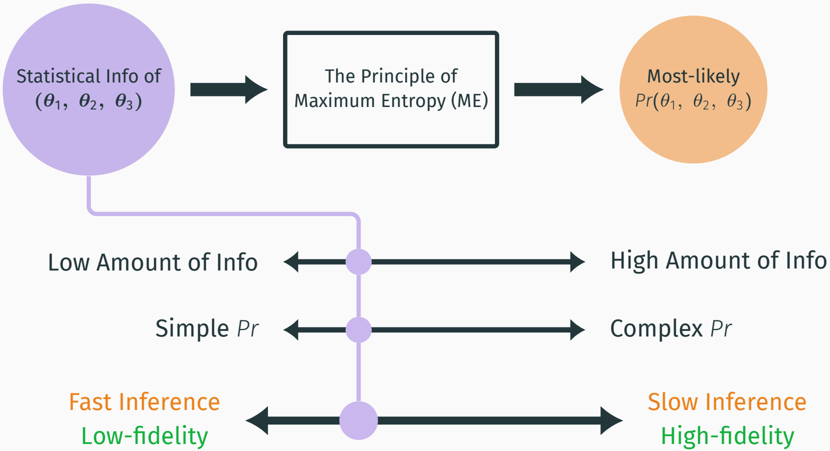
How to determine the probability distribution



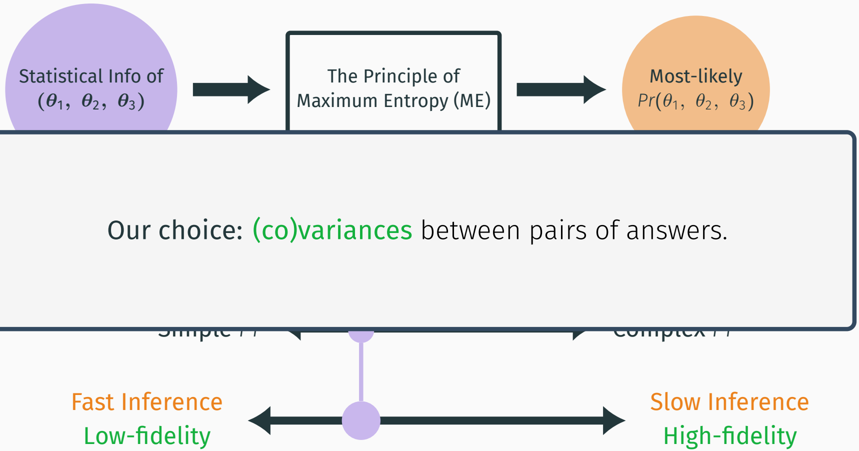
How to determine the probability distribution



How to determine the probability distribution



How to determine the probability distribution



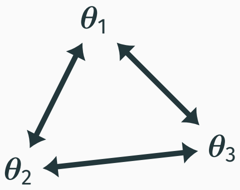
Our choice: (co)variances between pairs of answers.

Most-likely probability distribution

θ_1

θ_2 θ_3

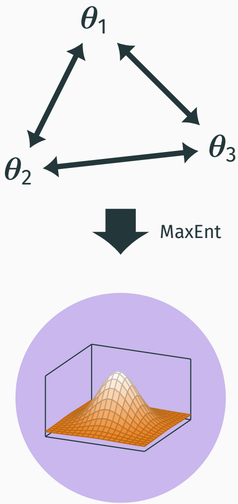
Most-likely probability distribution



Statistical Information:

Mean, variances, covariances

Most-likely probability distribution

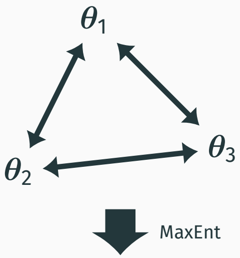


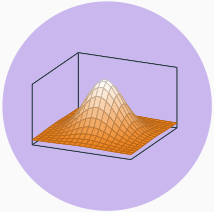
Statistical Information:

Mean, variances, covariances

Multivariate normal distribution

Most-likely probability distribution





Statistical Information:

Mean, variances, covariances

Multivariate normal distribution

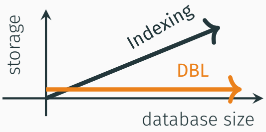
Fast inference using a closed form

Benefits of database learning

Database learning vs. indexing

Benefits of database learning

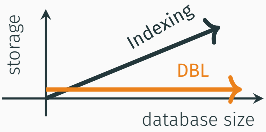
Database learning vs. indexing



1. Little storage overhead

Benefits of database learning

Database learning vs. indexing

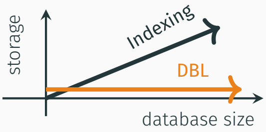


1. Little storage overhead

Database learning vs. materialized view

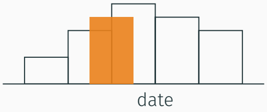
Benefits of database learning

Database learning vs. indexing



1. Little storage overhead

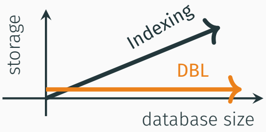
Database learning vs. materialized view



2. Without alignment

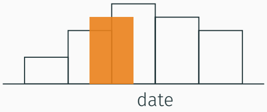
Benefits of database learning

Database learning vs. indexing

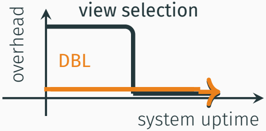


1. Little storage overhead

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)

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

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)

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

Our experimental claims

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

Dataset	# Analyzed	# Supported	Percentage
`Customer1`	3,342	2,463	73.7%
`TPC-H`	21	14	66.7%

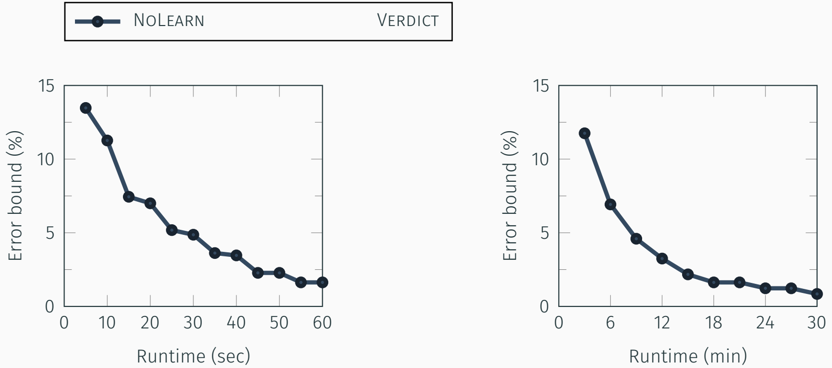
Unsupported queries:

1. Nested queries (that cannot be flattened)
2. Textual filters: `city like '%arbor%'`

Runtime-error trade-off

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

Number of past queries fixed to 50



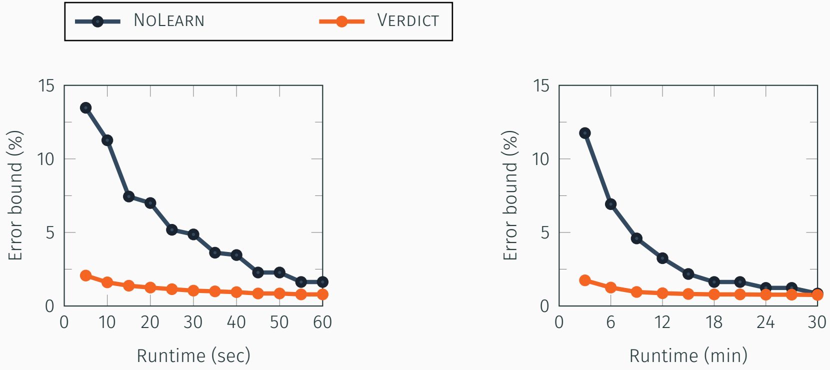
(a) Data in Memory

(b) Data on SSD

Runtime-error trade-off

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

Number of past queries fixed to 50

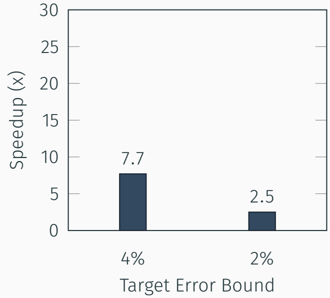


(a) Data in Memory

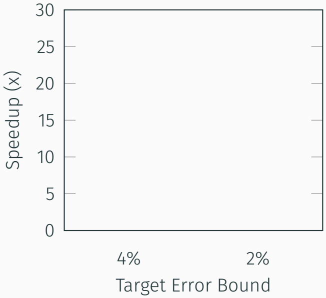
(b) Data on SSD

Speedup

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



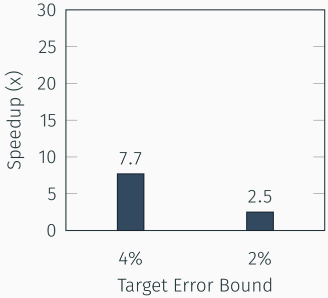
(a) Data in memory



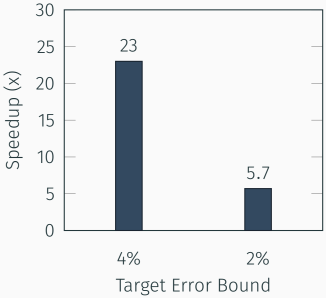
(b) Data on SSD

Speedup

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



(a) Data in memory



(b) Data on SSD

Memory and computational overhead

1. Memory overhead:

Memory and computational overhead

1. Memory overhead:
    - Queries and their answer, some matrices and their inverses

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

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:

	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%)

Thank You!