Adaptive query optimization in PostgreSQL

Oleg Ivanov

Postgres Professional

postgrespro.ru

What is query optimization?

How does PostgreSQL optimize queries?

What is adaptive query optimization?

Machine learning and kNN method.

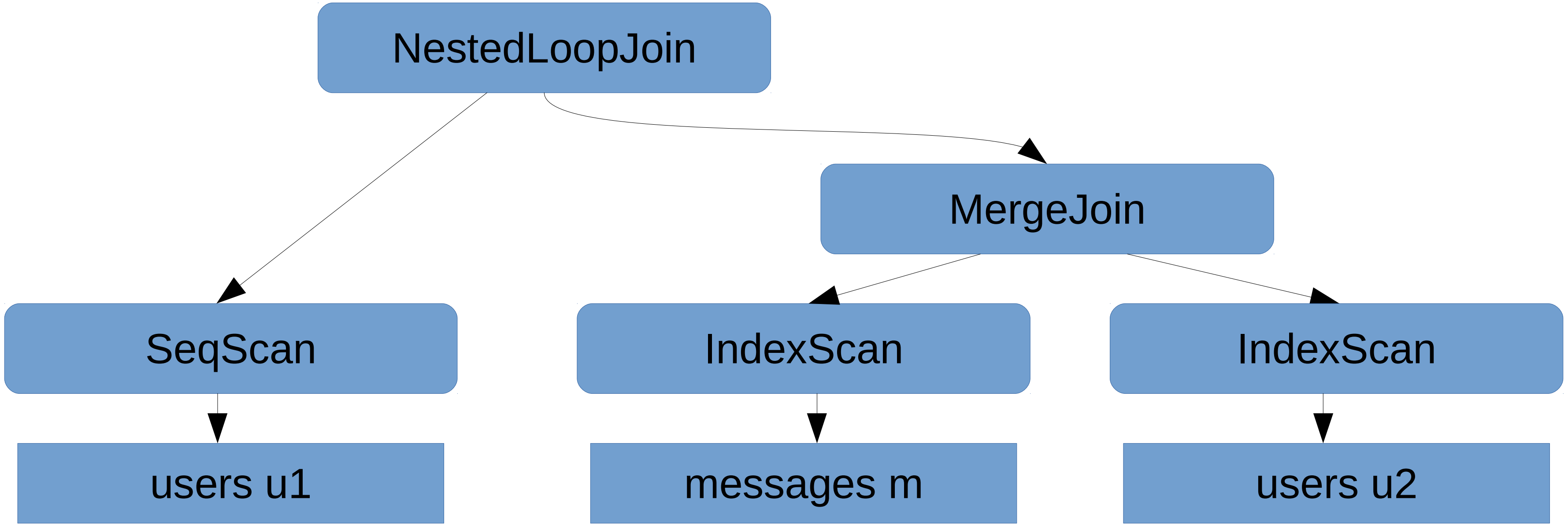
How to use machine learning for adaptive query optimization?

How much can it improve PostgreSQL performance?

Implementaion details: AQO.

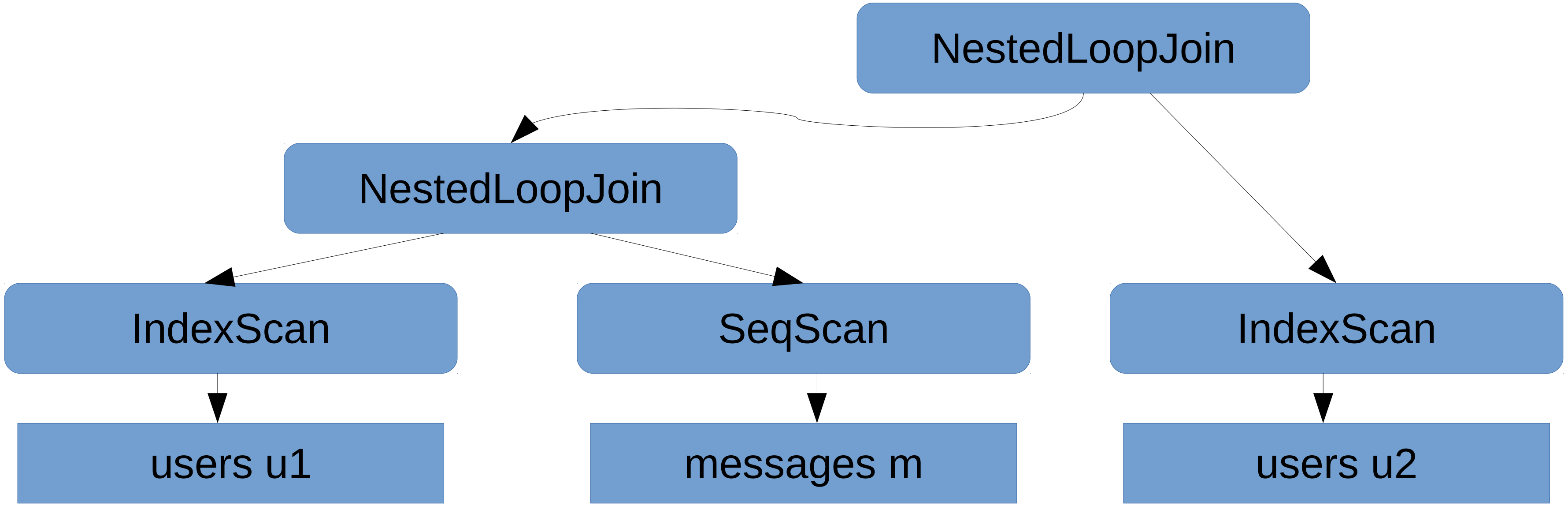
What is query optimization?

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



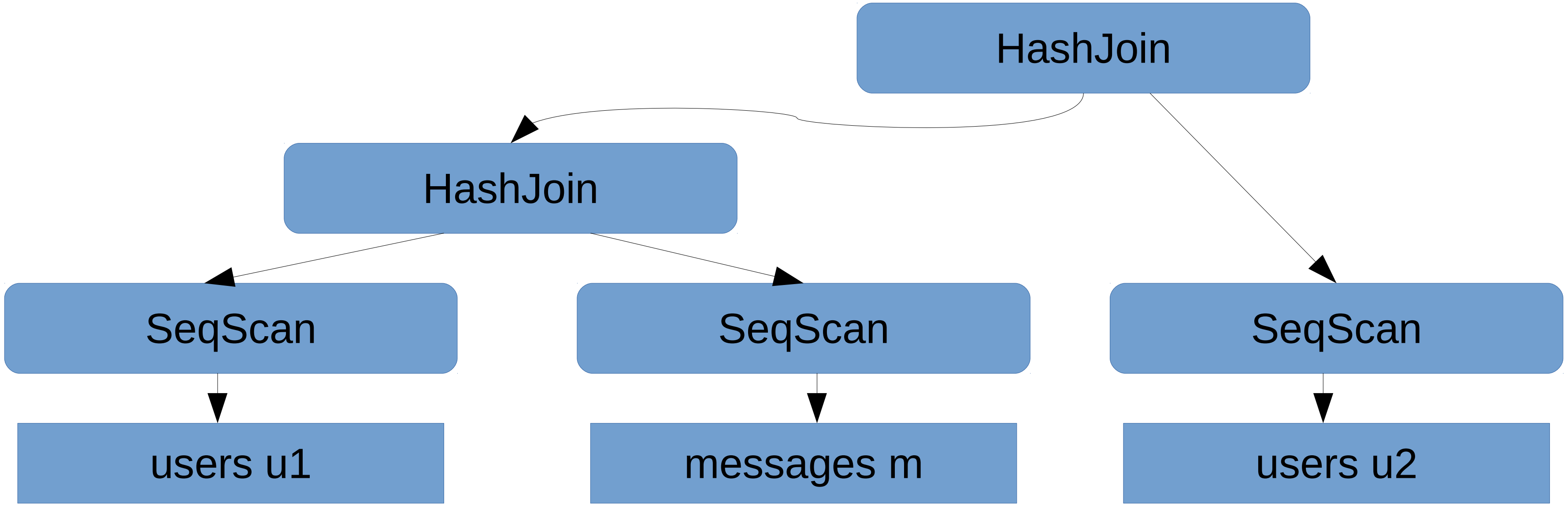
What is query optimization?

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



What is query optimization?

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



What is query optimization?

```
EXPLAIN SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Hash Join  (cost=540.00..439429.44 rows=10003825 width=27)
   Hash Cond: (m.receiver_id = u2.id)
   ->  Hash Join  (cost=270.00..301606.84 rows=10003825 width=23)
         Hash Cond: (m.sender_id = u1.id)
         ->  Seq Scan on messages m  (cost=0.00..163784.25 rows=10003825 width=19)
         ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
               ->  Seq Scan on users u1  (cost=0.00..145.00 rows=10000 width=4)
   ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
         ->  Seq Scan on users u2  (cost=0.00..145.00 rows=10000 width=4)
(9 rows)
```

What is query optimization?

```
EXPLAIN SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Hash Join  (cost=540.00..439429.44 rows=10003825 width=27)
   Hash Cond: (m.receiver_id = u2.id)
   ->  Hash Join  (cost=270.00..301606.84 rows=10003825 width=23)
         Hash Cond: (m.sender_id = u1.id)
         ->  Seq Scan on messages m  (cost=0.00..163784.25 rows=10003825 width=19)
         ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
               ->  Seq Scan on users u1  (cost=0.00..145.00 rows=10000 width=4)
   ->  Hash  (cost=145.00..145.00 rows=10000 width=4)
         ->  Seq Scan on users u2  (cost=0.00..145.00 rows=10000 width=4)
(9 rows)
```

Plan execution cost

Plan node execution cost

Plan node cardinality

How does PostgreSQL optimize queries?

Cost-based query optimization

System R (1974)

Choose the cheapest plan
among all the possible plans

How does PostgreSQL optimize queries?

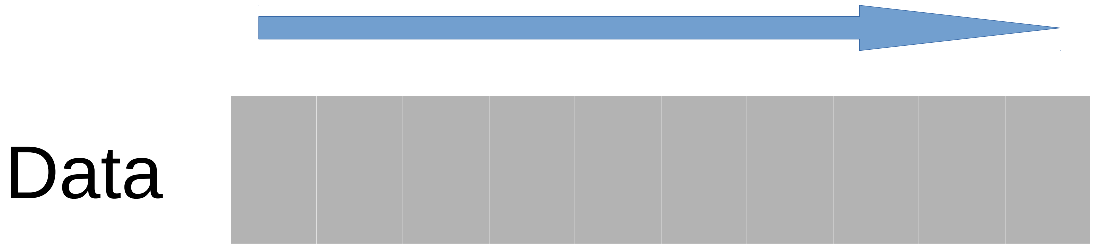
$$Cost = n_s c_s + n_r c_r + n_t c_t + n_i c_i + n_o c_o$$

c_s	seq_page_cost	1.0
c_r	random_page_cost	4.0
c_t	cpu_Tuple_cost	0.01
c_i	cpu_Index_tuple_cost	0.005
c_o	cpu_Operator_cost	0.0025

How does PostgreSQL optimize queries?

```
SELECT * FROM users
WHERE age < 25;
```

SeqScan



$$Cost = n_s c_s + n_o \cdot c_o$$

$$n_s = N_{pages}$$

$$n_o = N_{tuples}$$

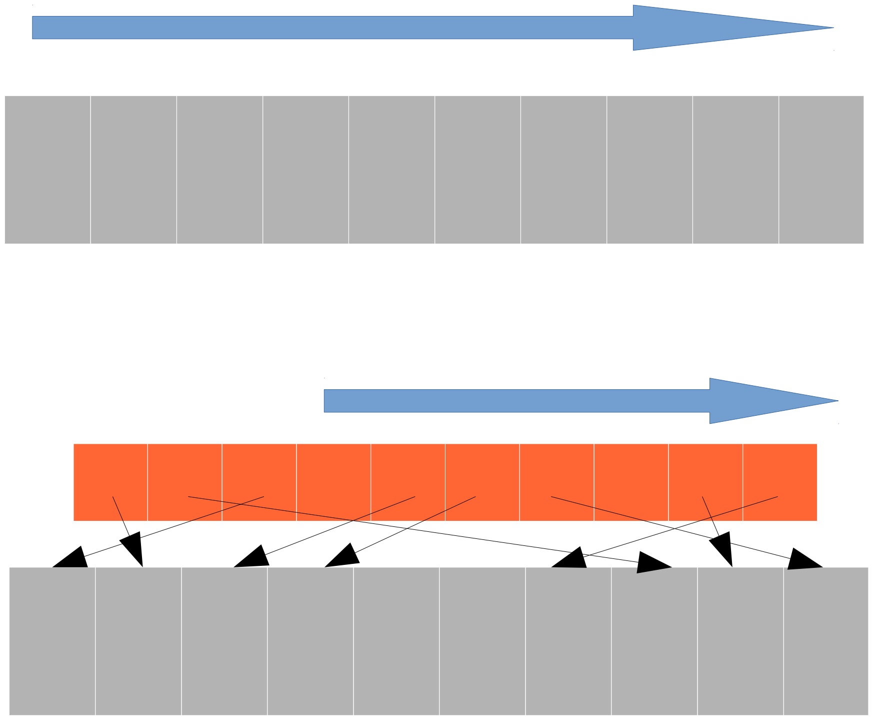
IndexScan

How does PostgreSQL optimize queries?

```
SELECT * FROM users
WHERE age < 25;
```

SeqScan

Data



$$Cost = n_s c_s + n_o \cdot c_o$$

$$n_s = N_{pages}$$

$$n_o = N_{tuples}$$

Index

IndexScan

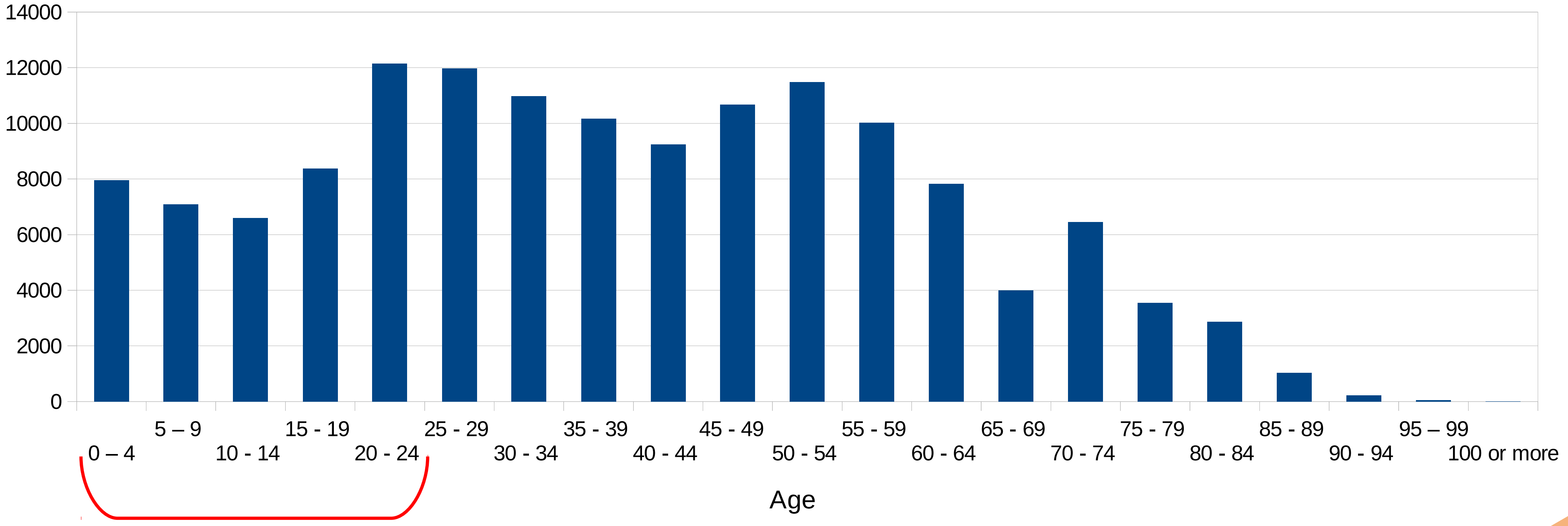
Data

$$Cost = n_r \cdot c_r$$

$$n_r = Cardinality$$

How does PostgreSQL optimize queries?

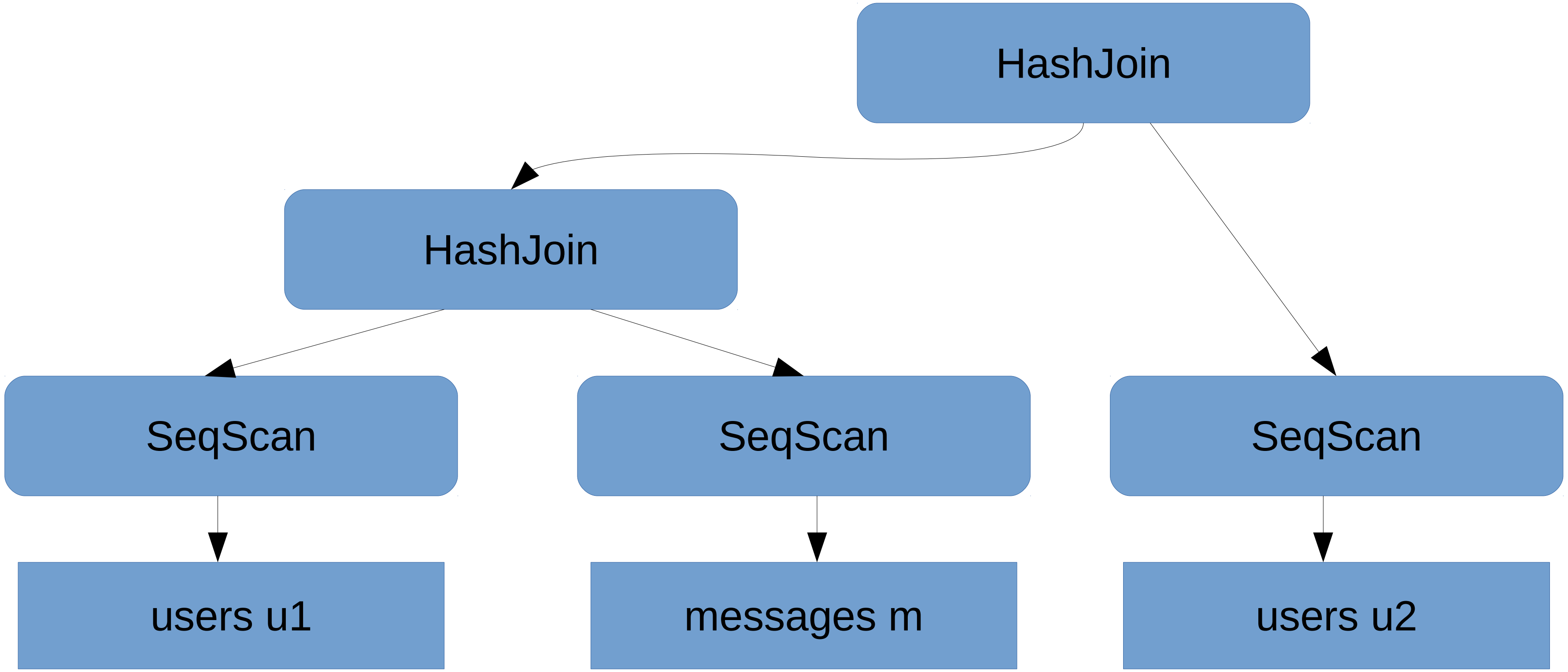
```
SELECT * FROM users
WHERE age < 25;
```



Cardinality

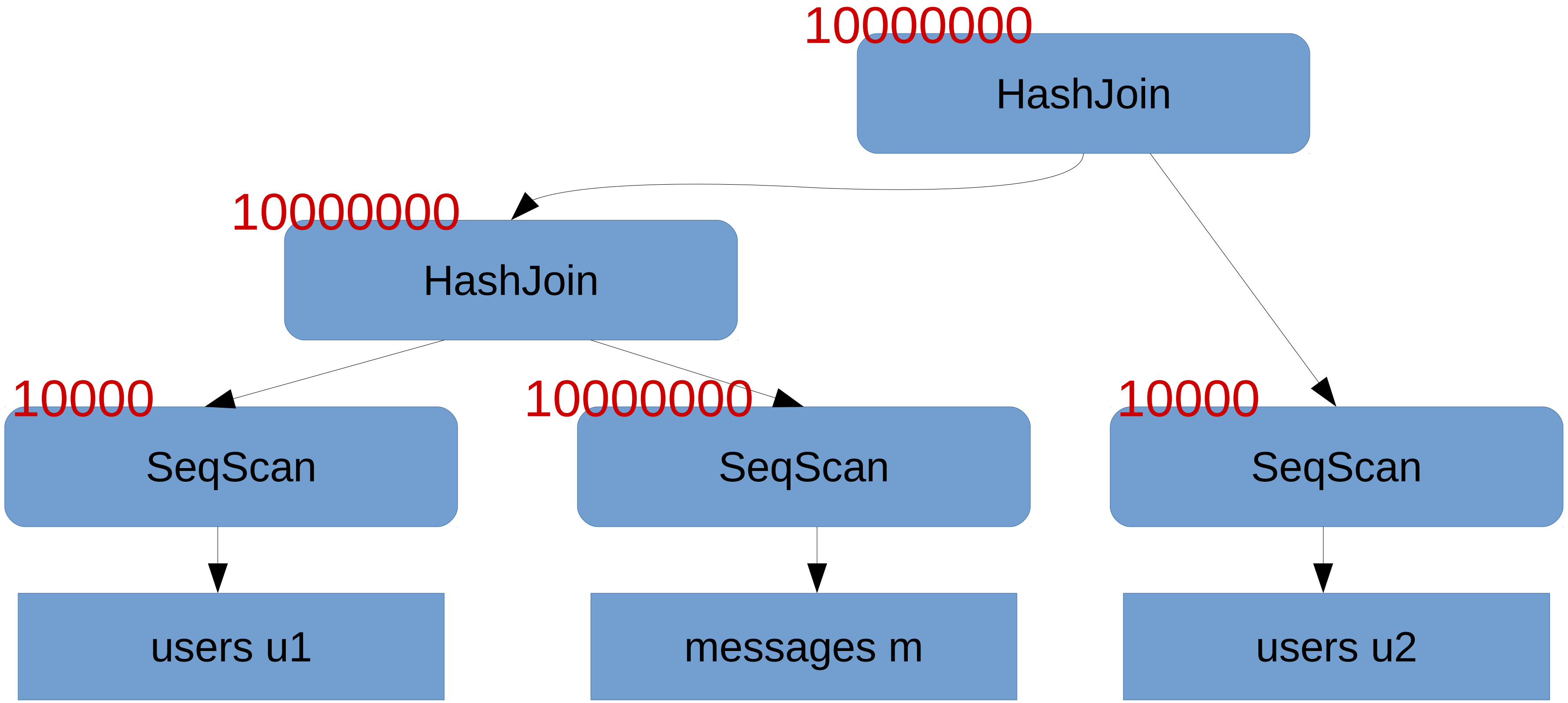
How does PostgreSQL optimize queries?

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



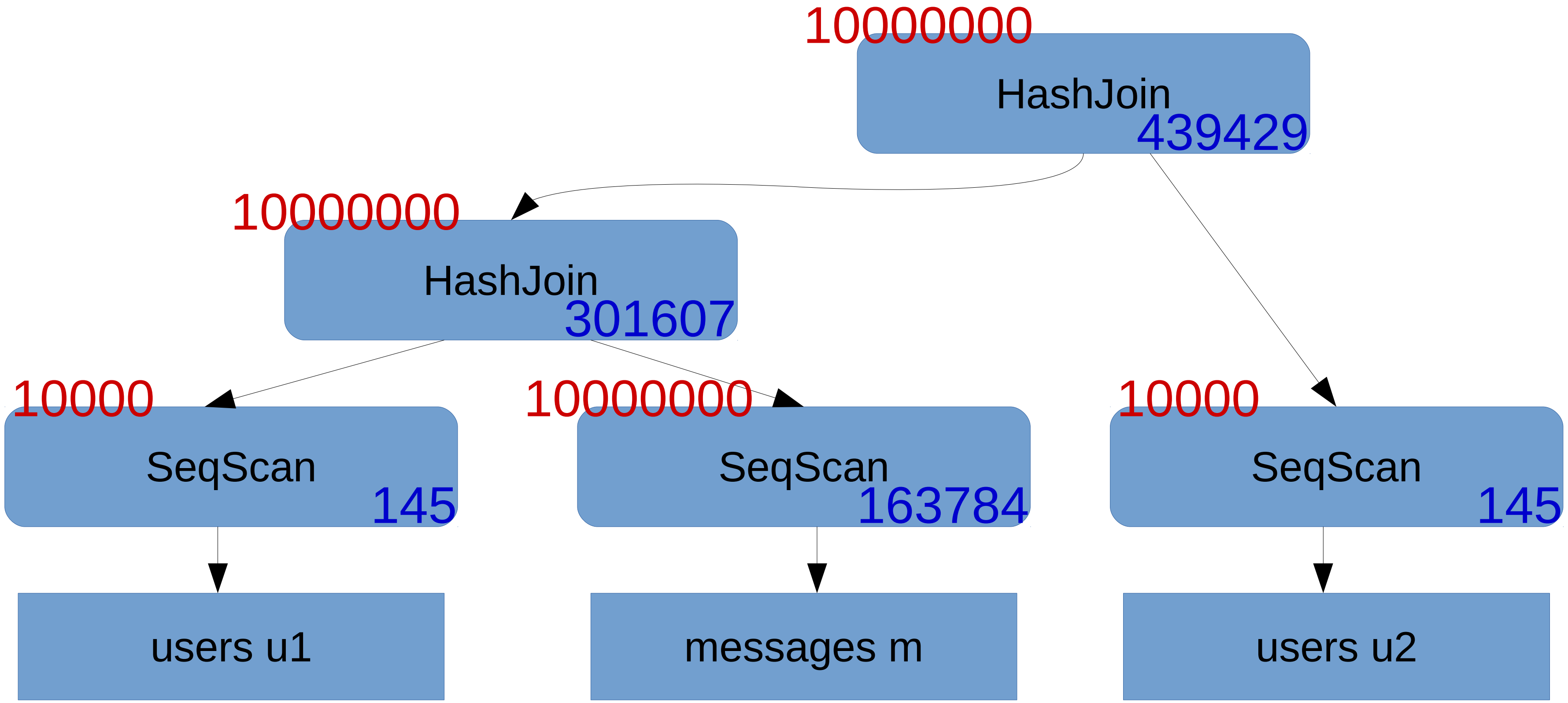
How does PostgreSQL optimize queries?

```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



How does PostgreSQL optimize queries?

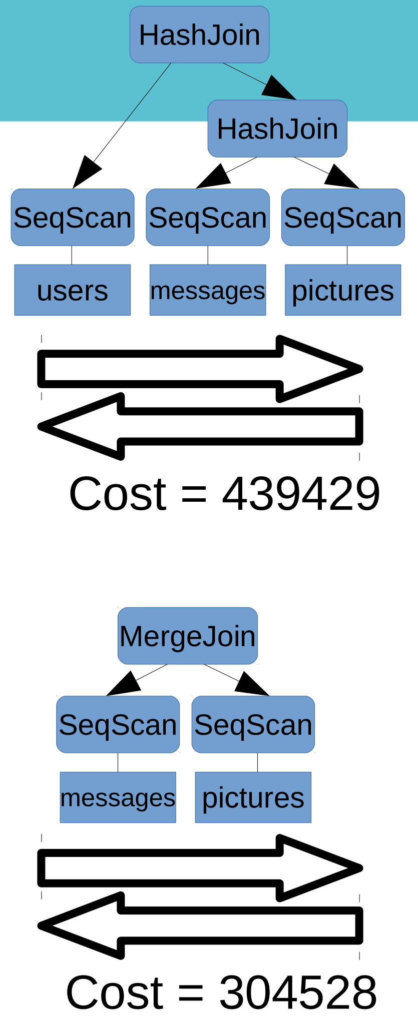
```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



How does PostgreSQL optimize queries?

Optimization method

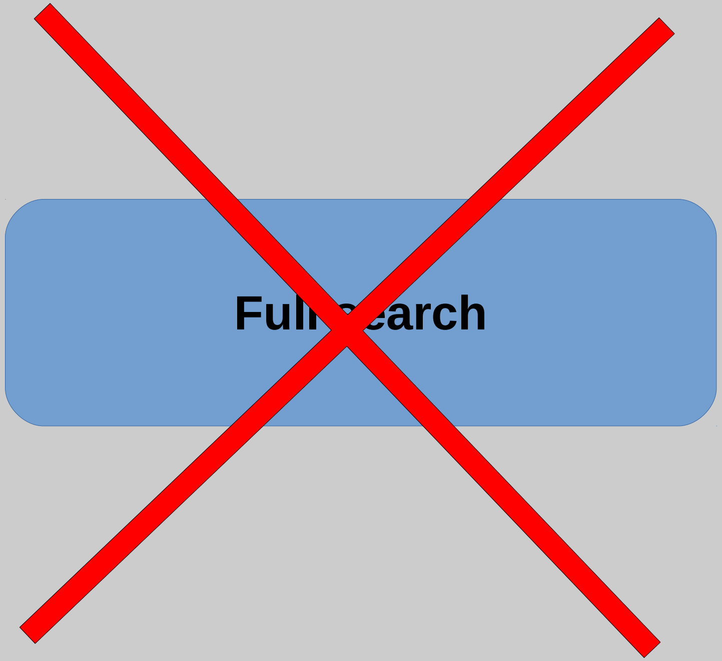
Full search

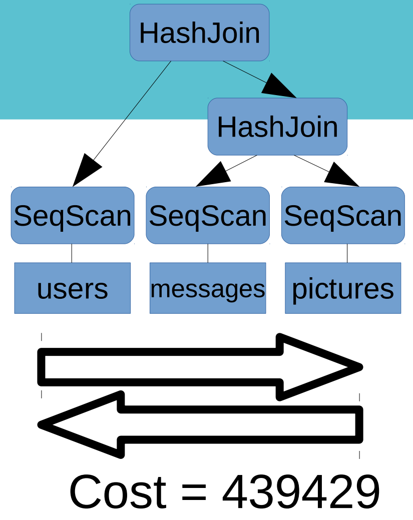


Plan's cost estimation

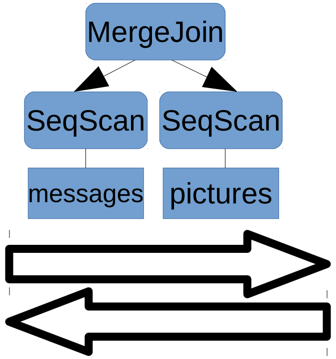
How does PostgreSQL optimize queries?

Optimization method





Cost = 439429



Cost = 304528

Plan's cost estimation

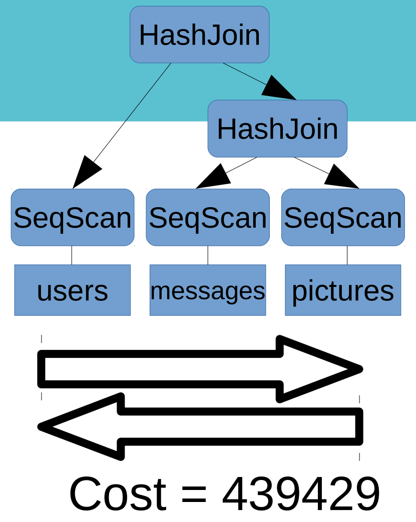
How does PostgreSQL optimize queries?

Optimization method

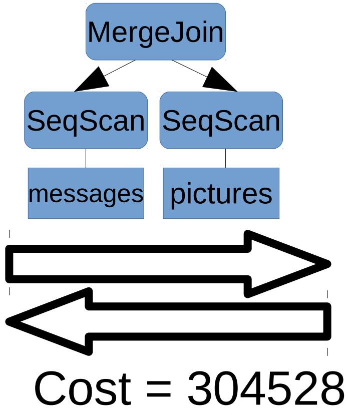
Dynamic programming

or

Genetic algorithm



Cost = 439429



Cost = 304528

Plan's cost estimation

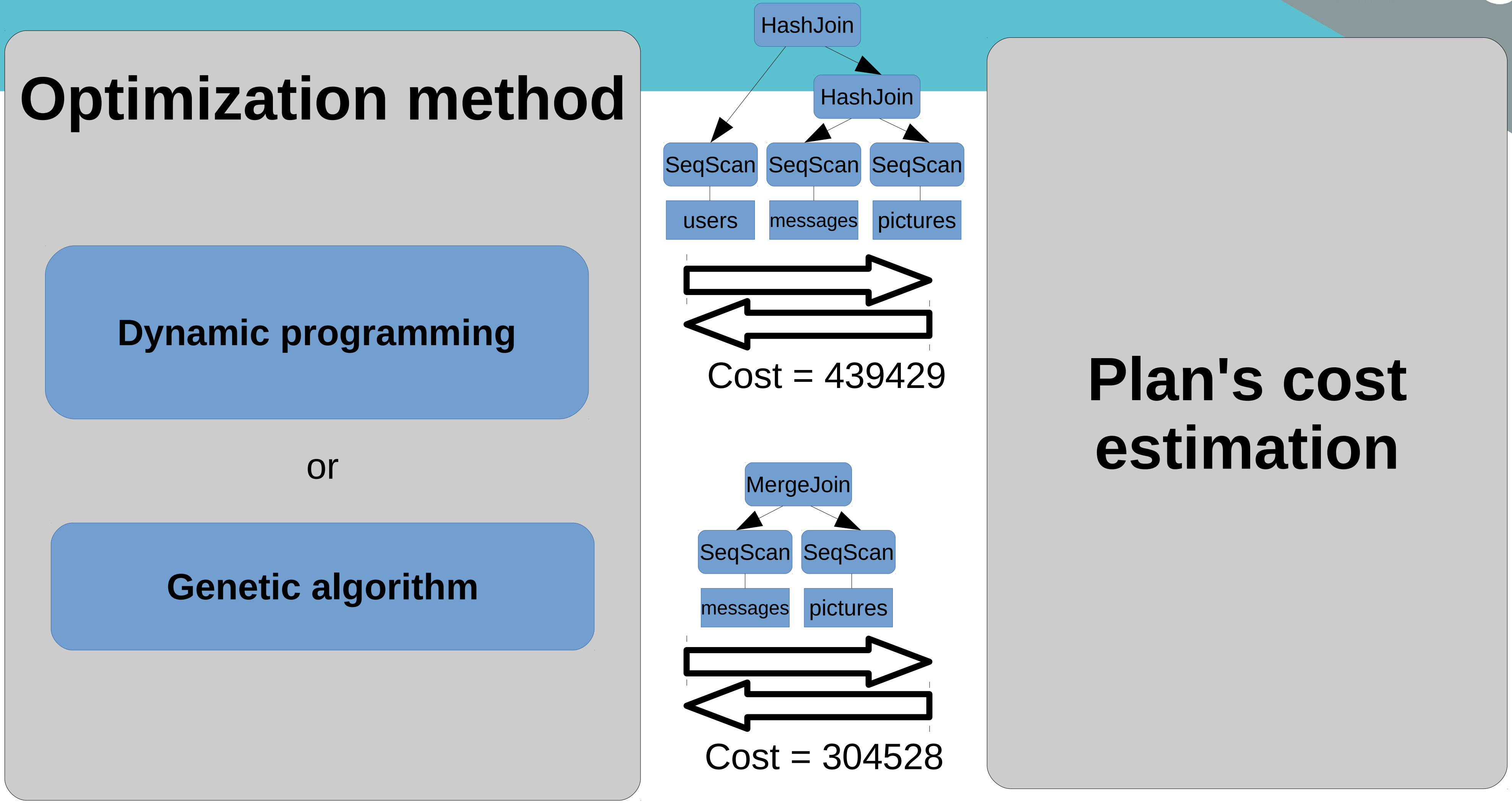
Dynamic programming over subsets

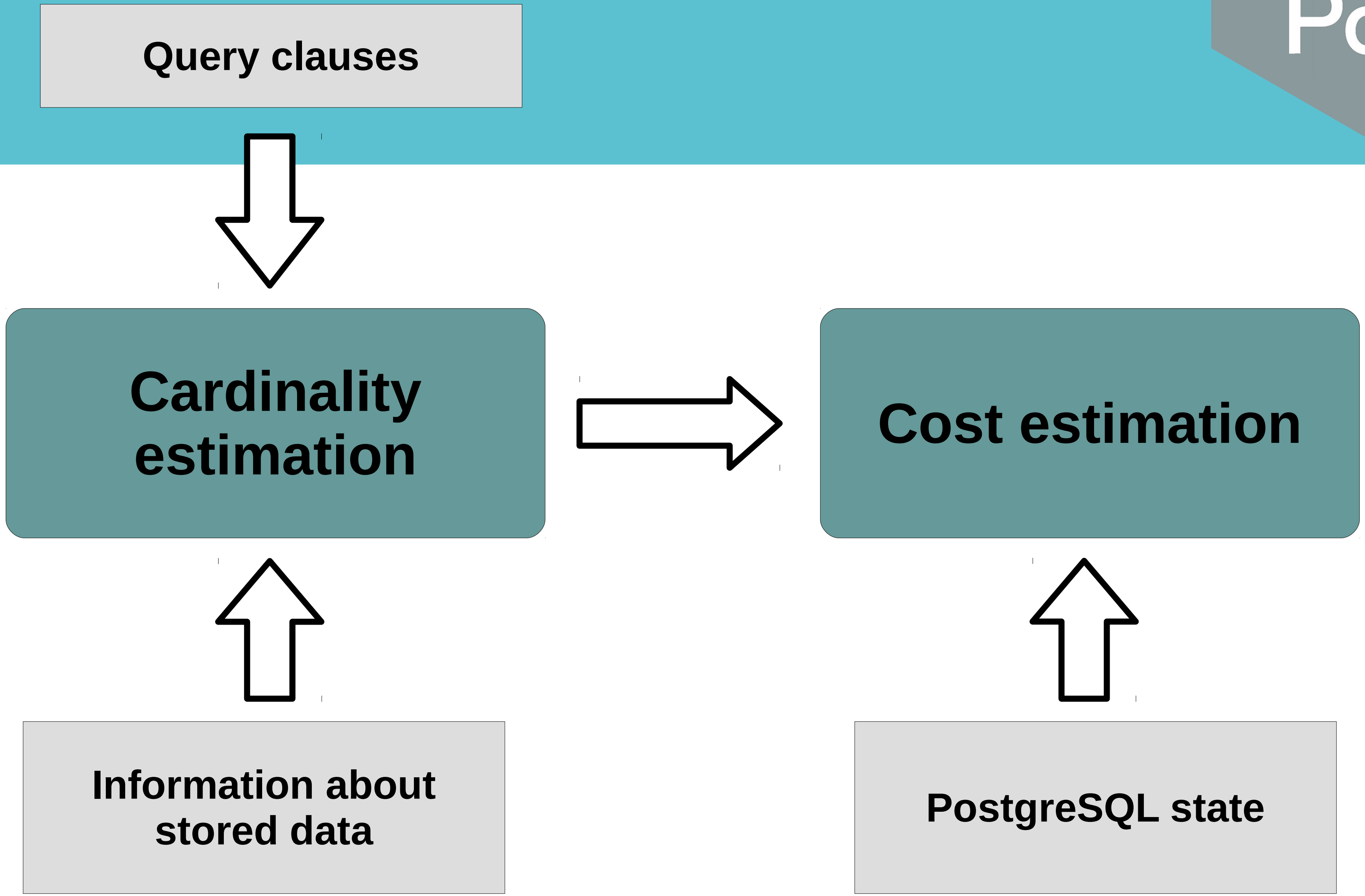
- System R
- Time complexity: 3^n
- Memory consumption: 2^n
- Always finds the cheapest plan

Genetic algorithm

- PostgreSQL
- Common and flexible method
- Can be stopped on every iteration
- No guarantees

How does PostgreSQL optimize queries?





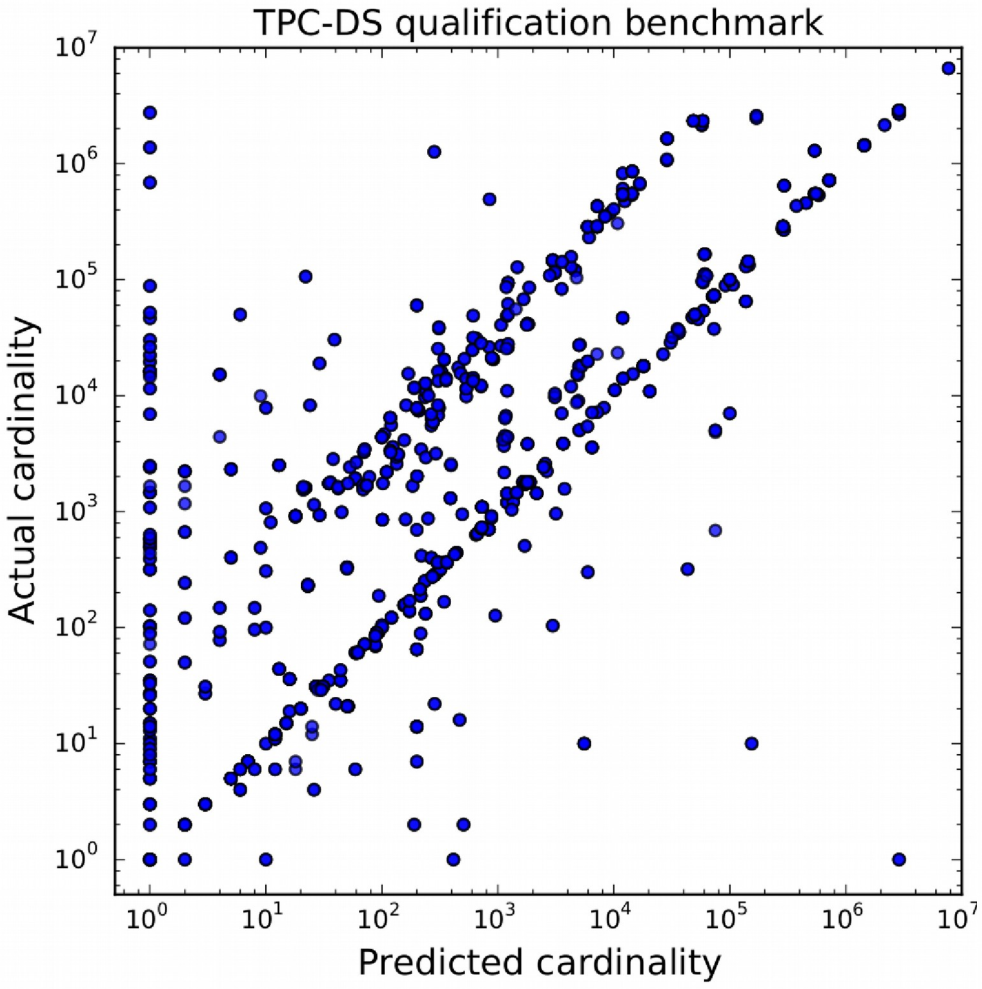
Query clauses

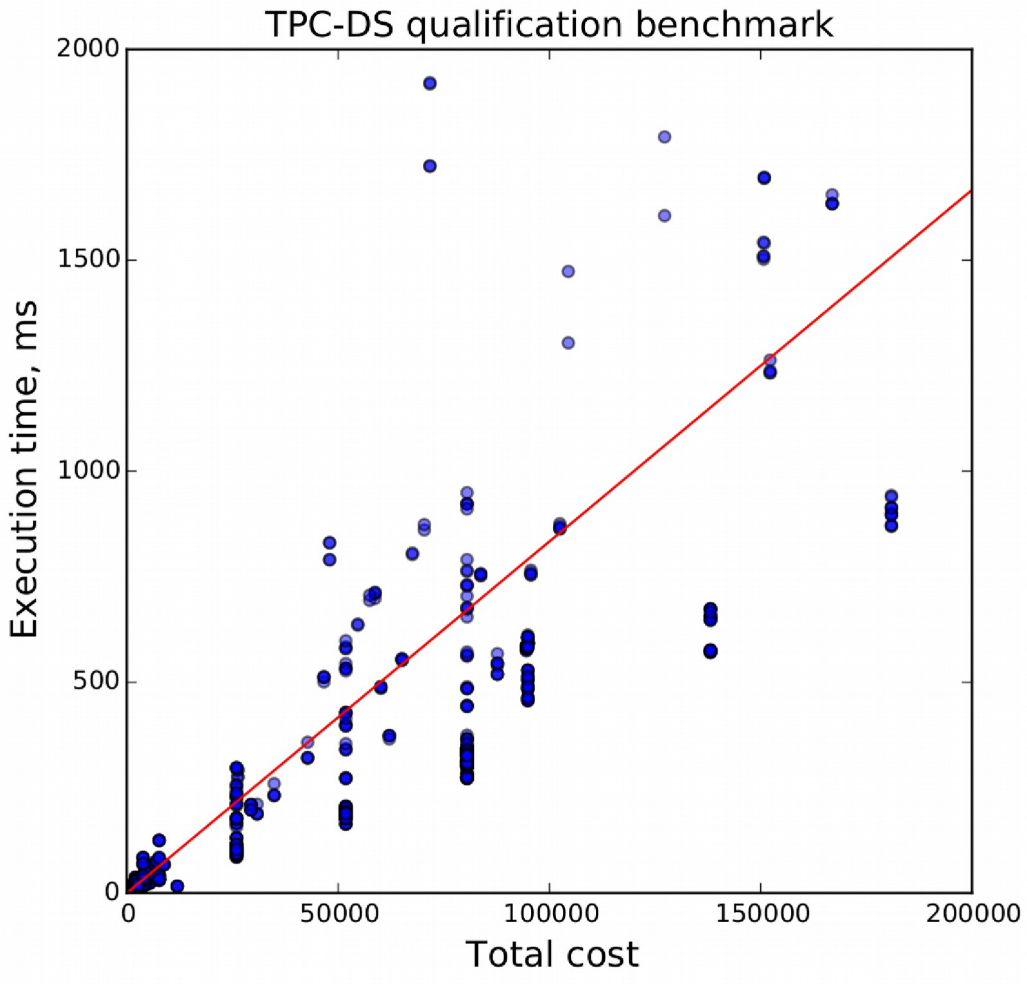
Cardinality estimation

Cost estimation

Information about stored data

PostgreSQL state

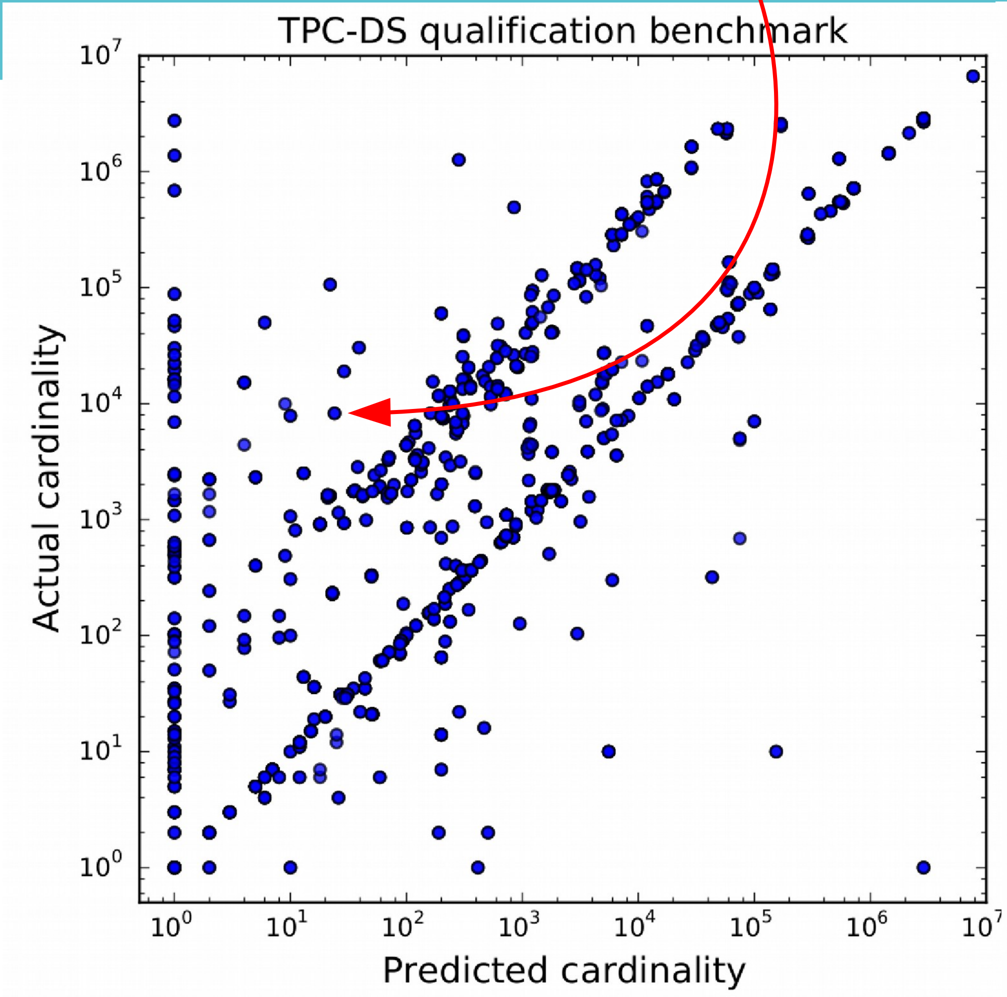


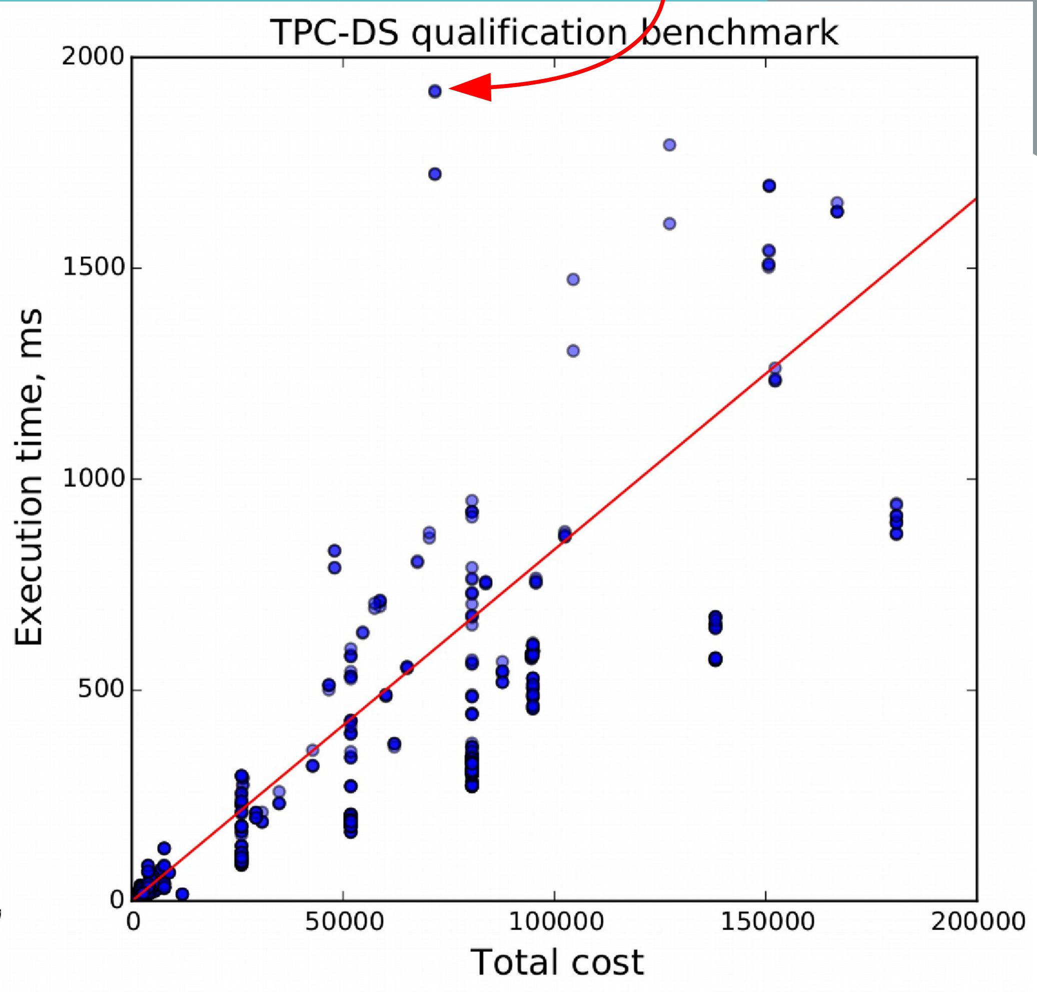


Dataset:
The TPC Benchmark™DS (TPC-DS)
http://www.tpc.org/tpcds/

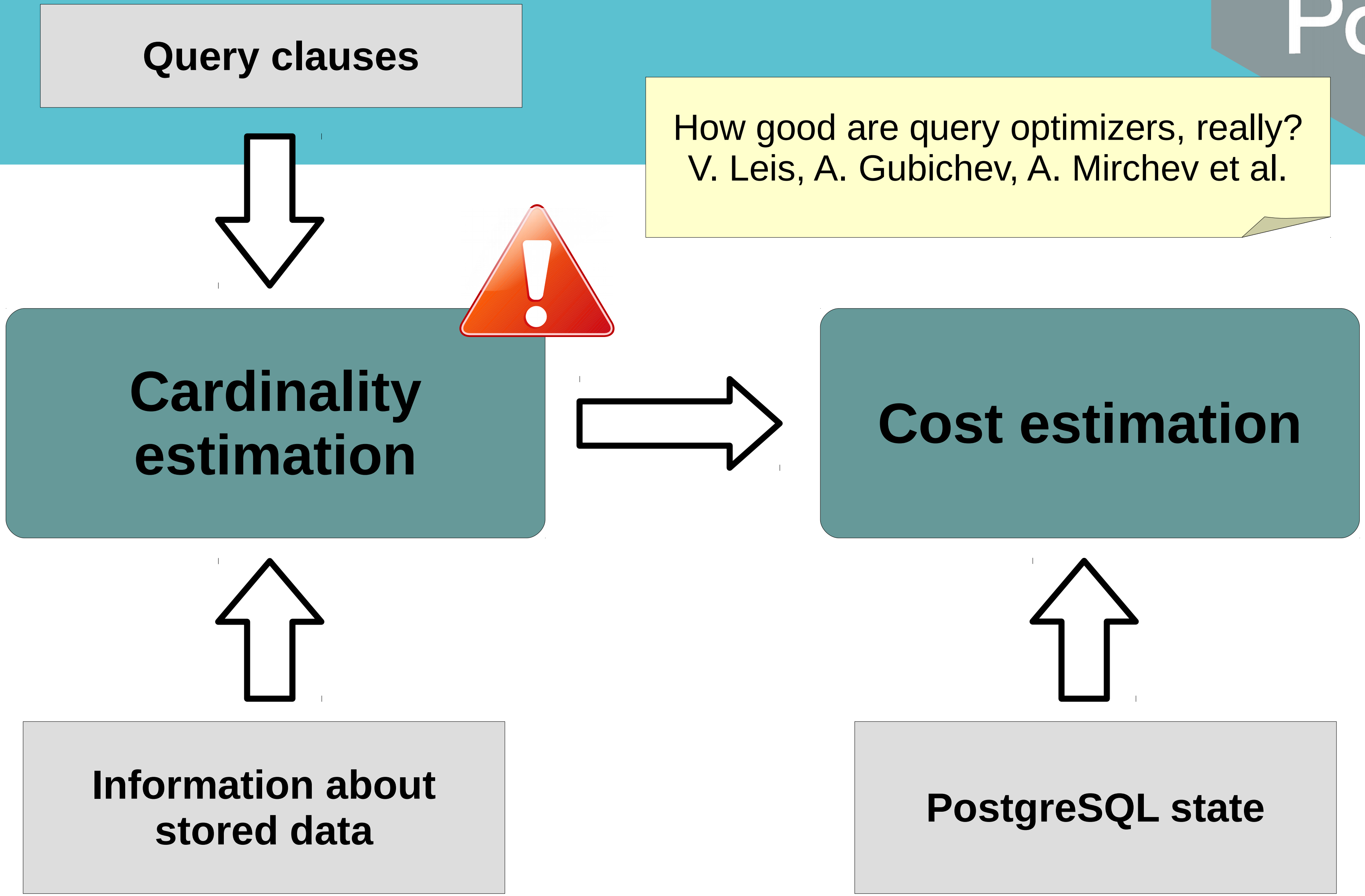
Error: 300 times

Error: 4 times





Dataset:
The TPC Benchmark™DS (TPC-DS)
http://www.tpc.org/tpcds/



Query clauses

Cardinality estimation

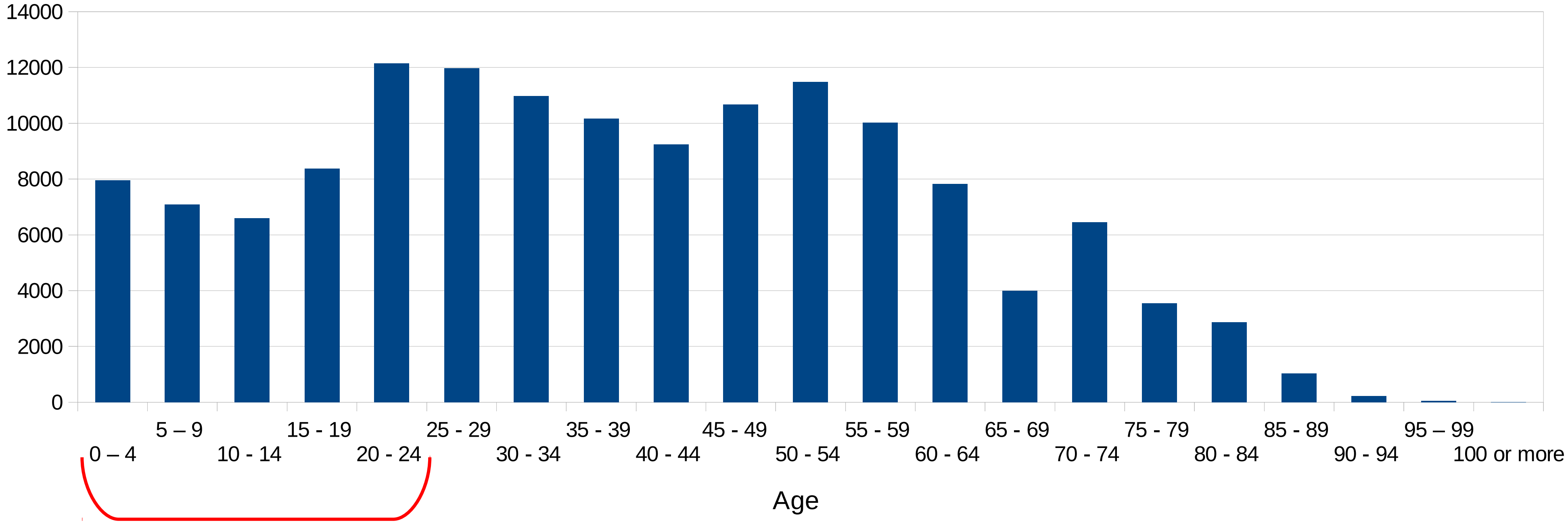
Cost estimation

Information about stored data

PostgreSQL state

How good are query optimizers, really?
V. Leis, A. Gubichev, A. Mirchev et al.

```
SELECT * FROM users
WHERE age < 25;
```



$$Selectivity \simeq 0.3$$

$$Cardinality = N_{tuples} \cdot Selectivity$$

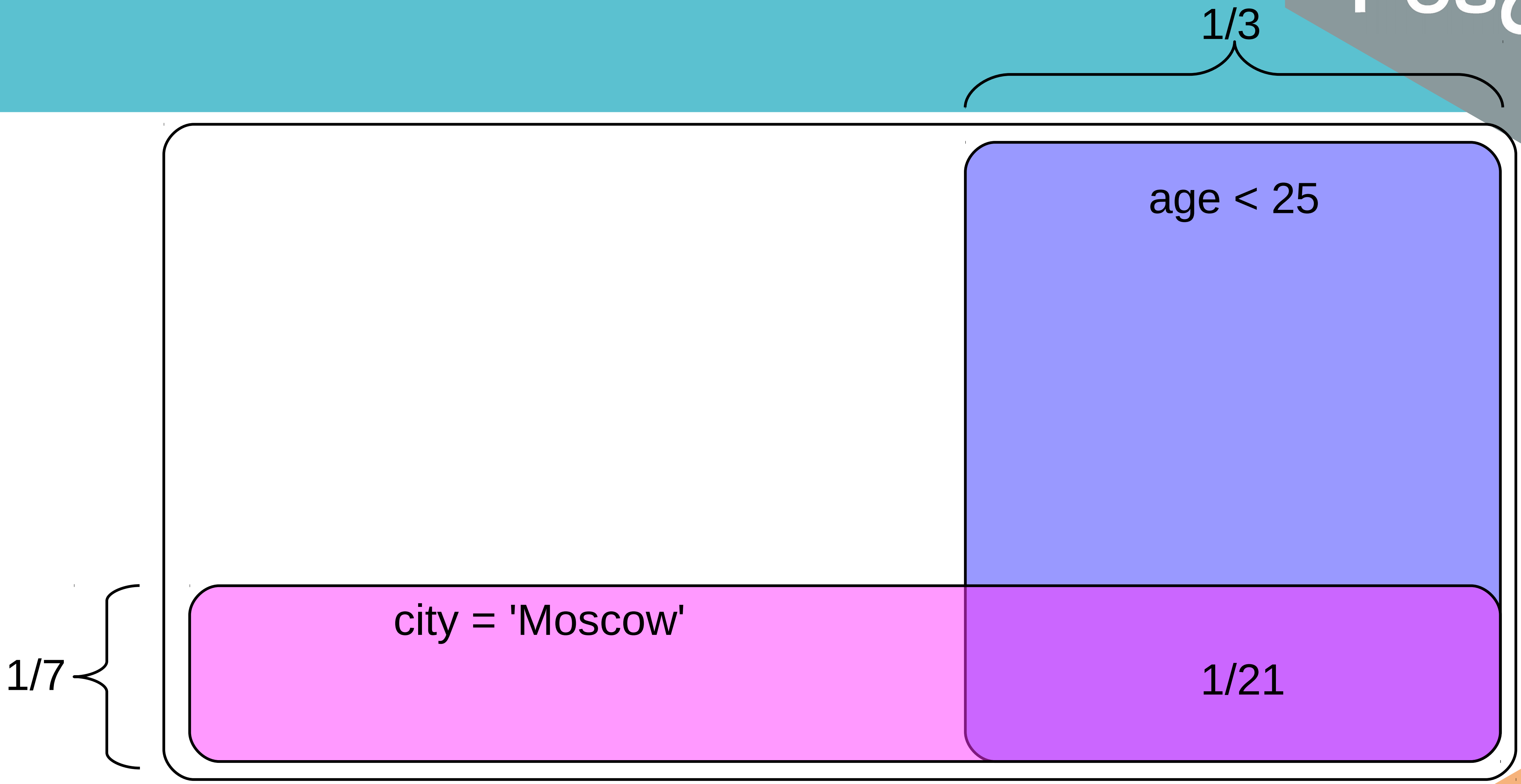
```
SELECT * FROM users
WHERE age < 25 AND city = 'Moscow';
```

Only selectivities of individual clauses
(i.e. *marginal* selectivities)
are known

$$Selectivity_{age} = \frac{1}{3}$$

$$Selectivity_{city} = \frac{1}{7}$$

$$Selectivity_{age,city} = ?$$


1/3
age < 25
city = 'Moscow'
1/7
1/21

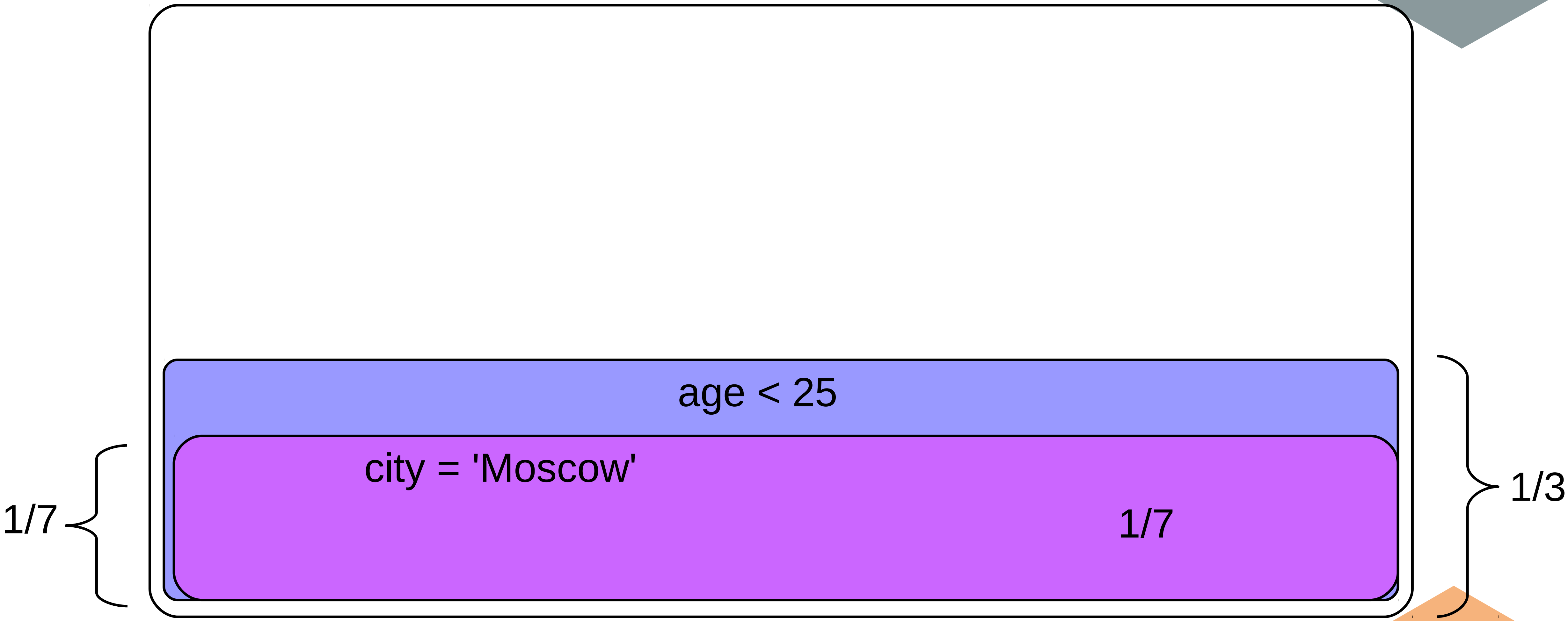
```
SELECT * FROM users
WHERE age < 25 AND city = 'Moscow';
```

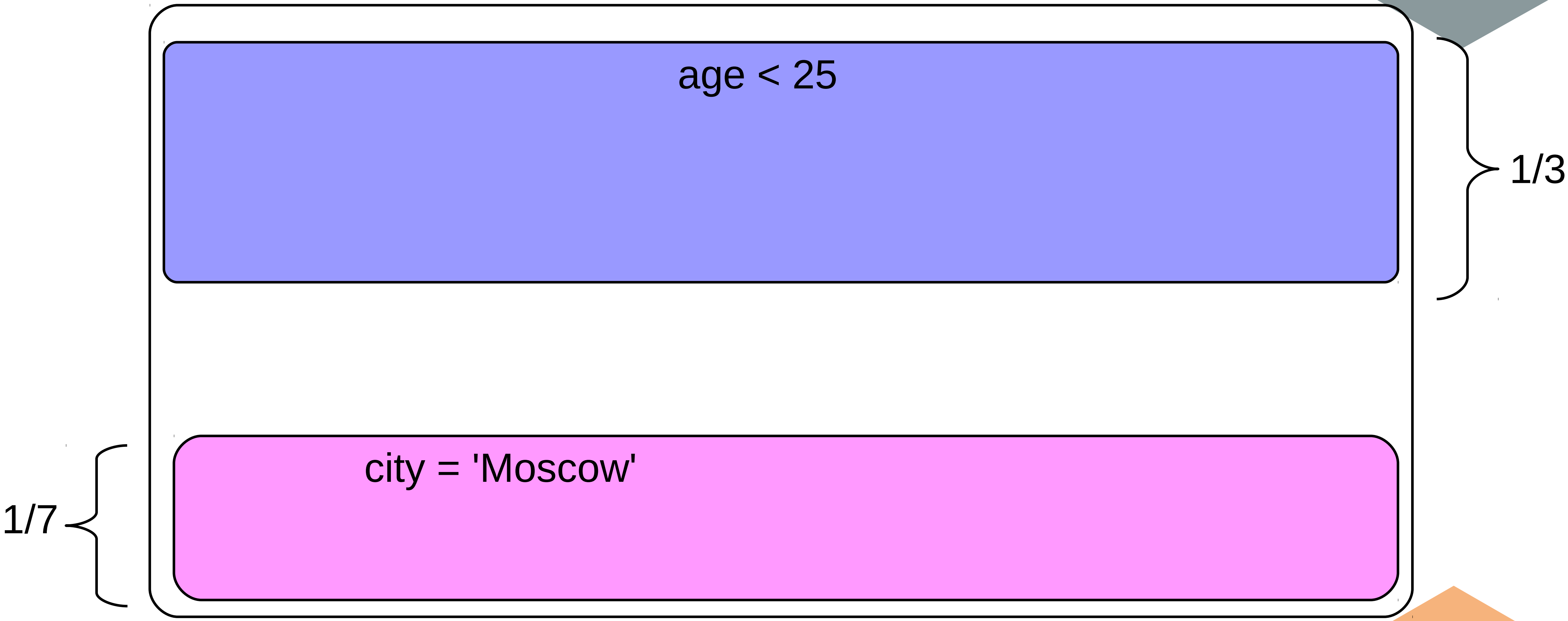
Only selectivities of individual clauses are known

The clauses are considered to be independent:

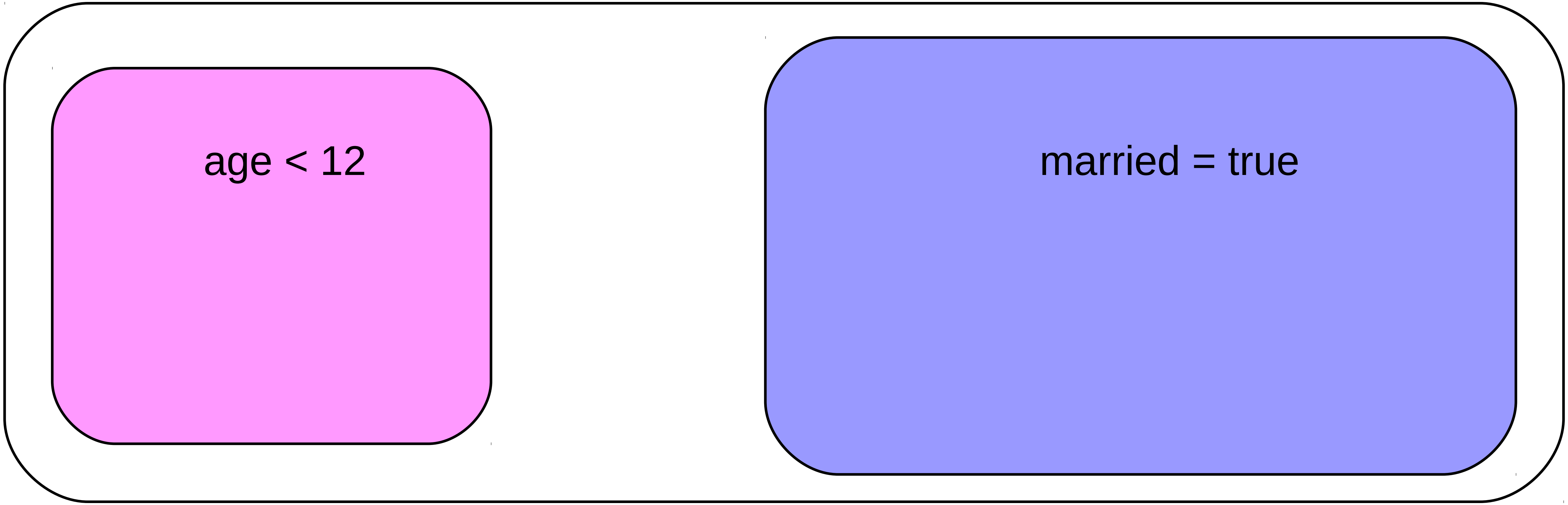
$$Selectivity_{age,city} = Selectivity_{age} \cdot Selectivity_{city}$$

With the exception of $Selectivity_{25<age \text{ AND } age<57} = Selectivity_{25<age<57}$


age < 25
city = 'Moscow'
1/7
1/7
1/3


age < 25
1/3
city = 'Moscow'
1/7

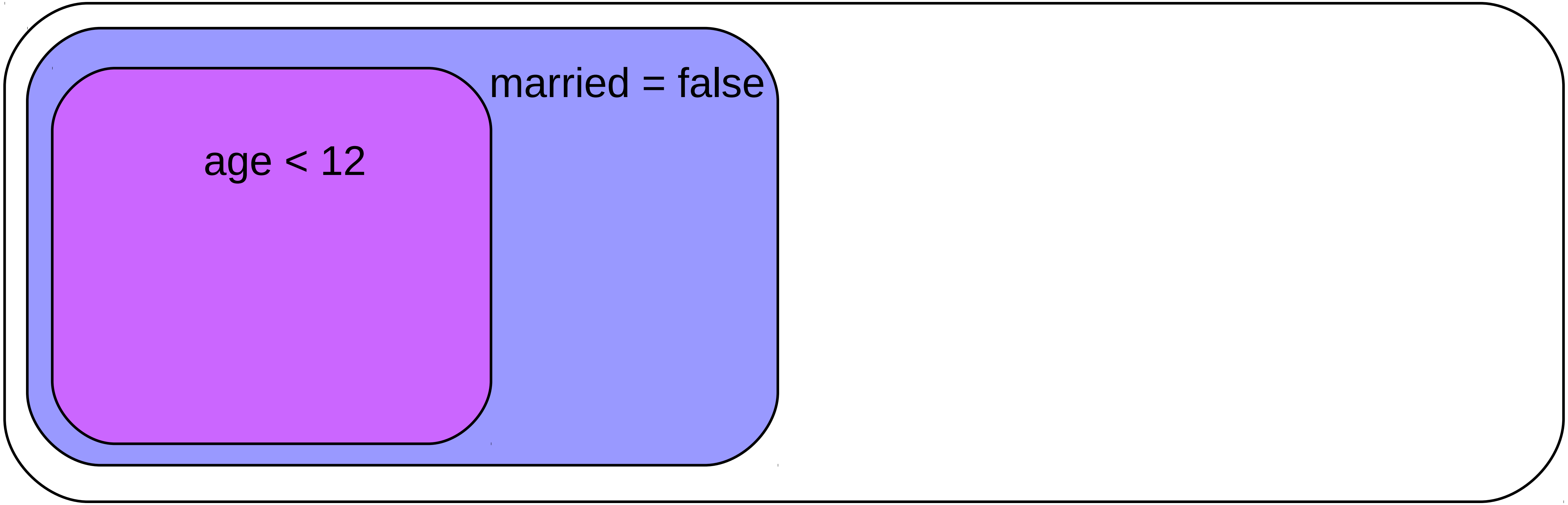
```
SELECT * FROM users
WHERE age < 12 AND married = true;
```



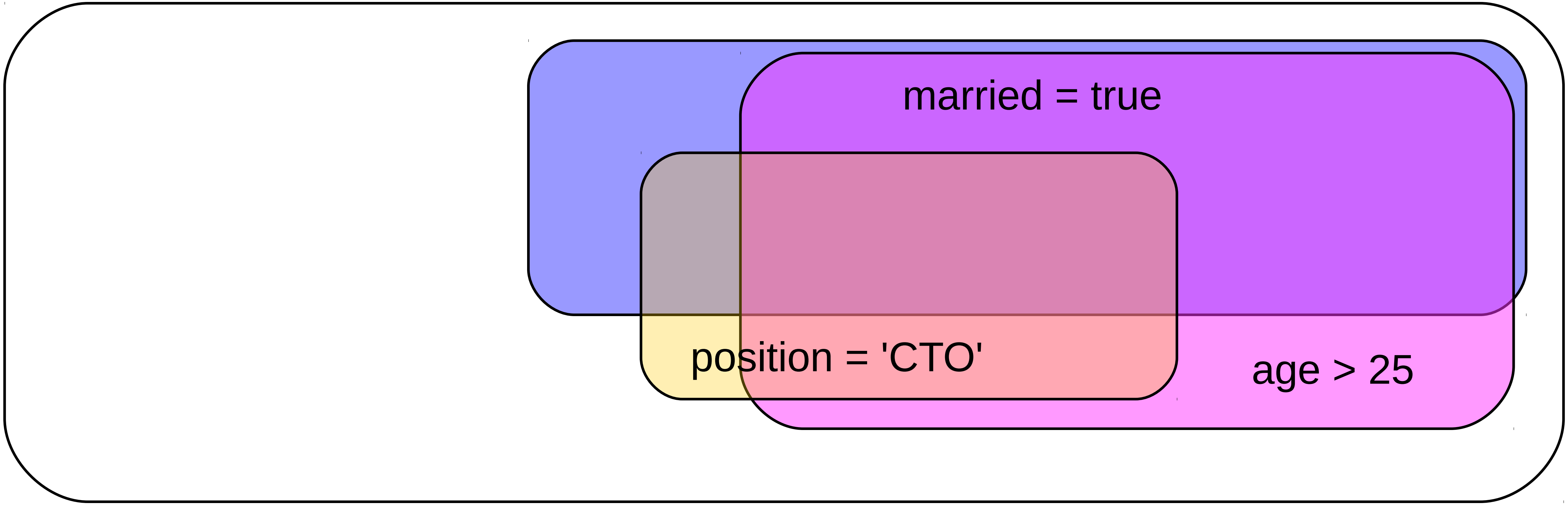
age < 12

married = true

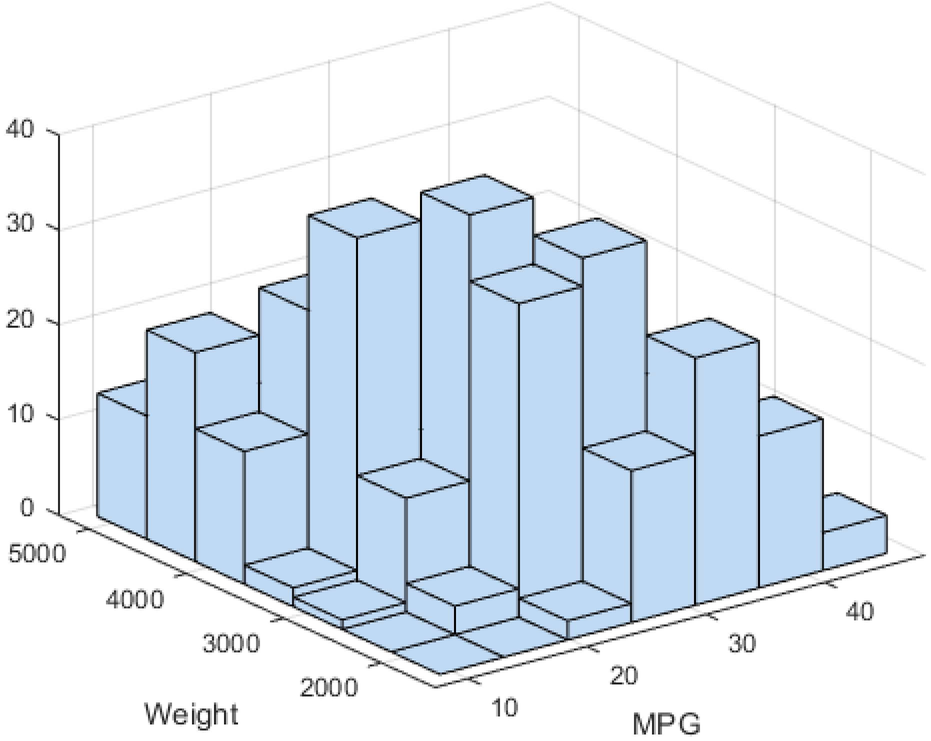
```
SELECT * FROM users
WHERE age < 12 AND married = false;
```



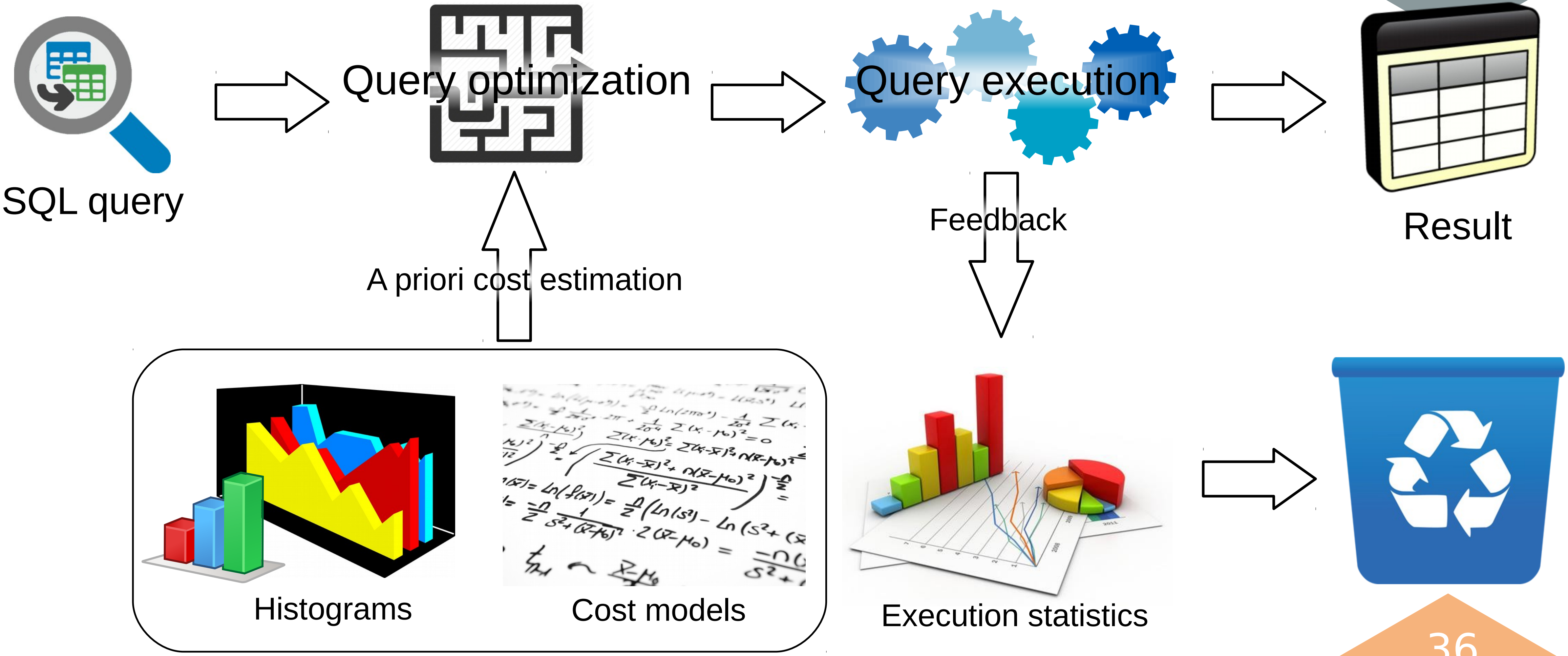
```
SELECT * FROM users
WHERE age > 25 AND married = true
AND position = 'CTO';
```



Multidimensional histograms



What is adaptive query optimization?



SQL query

Query optimization

Query execution

Result

A priori cost estimation

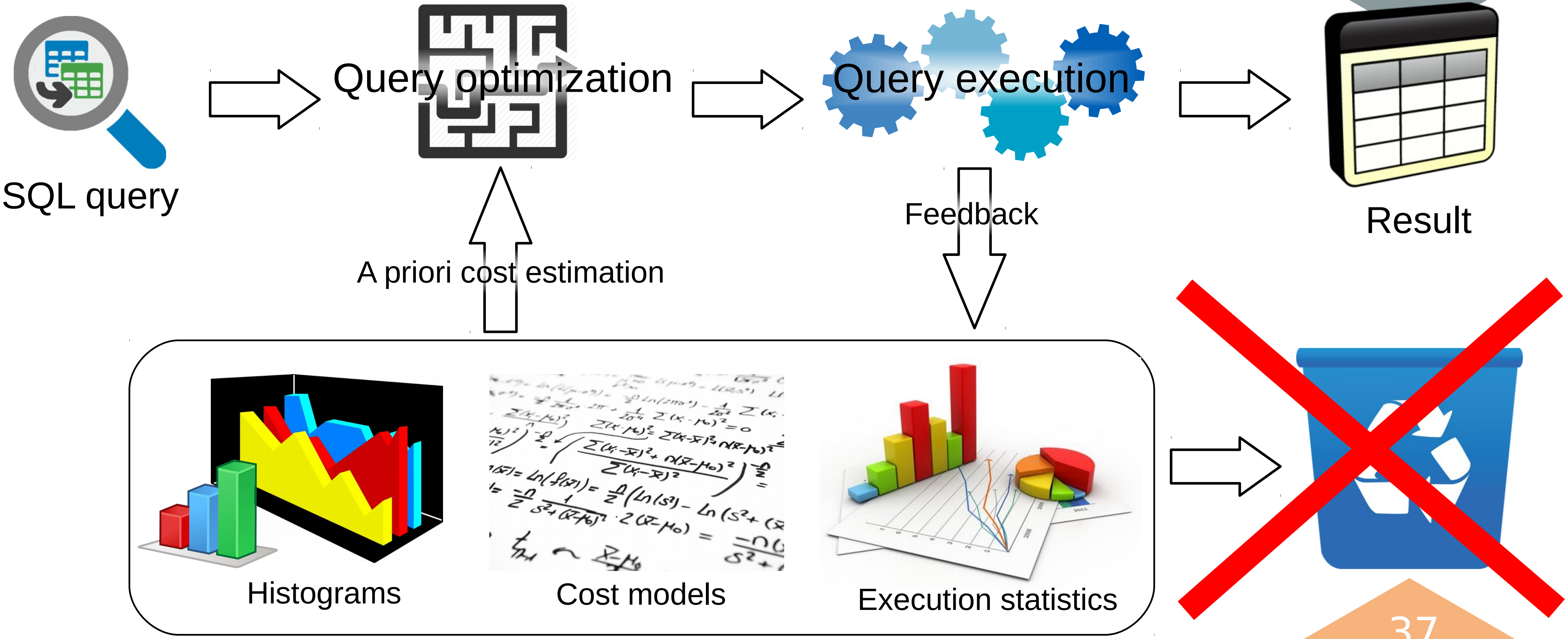
Feedback

Histograms

Cost models

Execution statistics

What is adaptive query optimization?



SQL query

Query optimization

Query execution

Result

A priori cost estimation

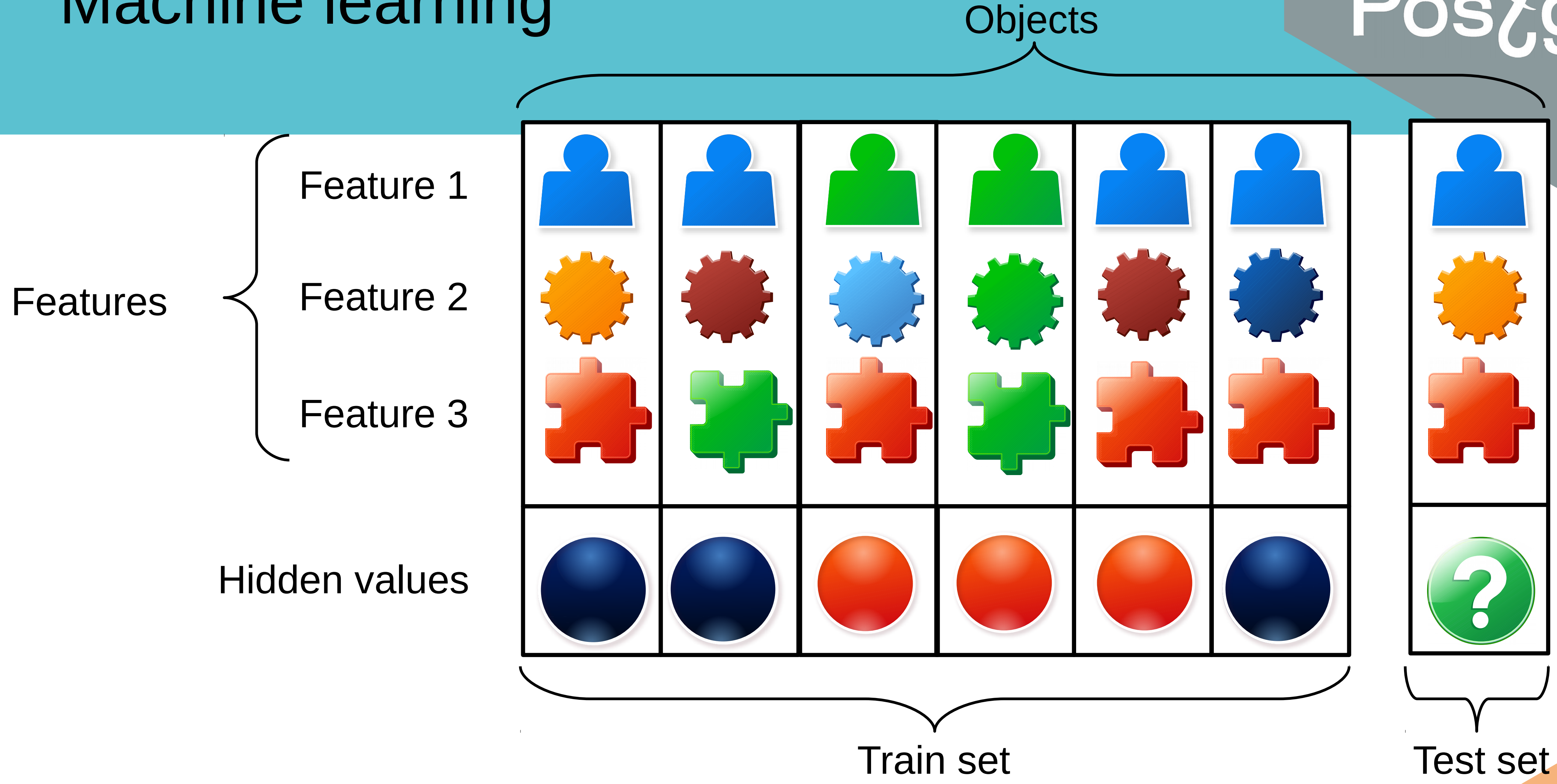
Feedback

Histograms

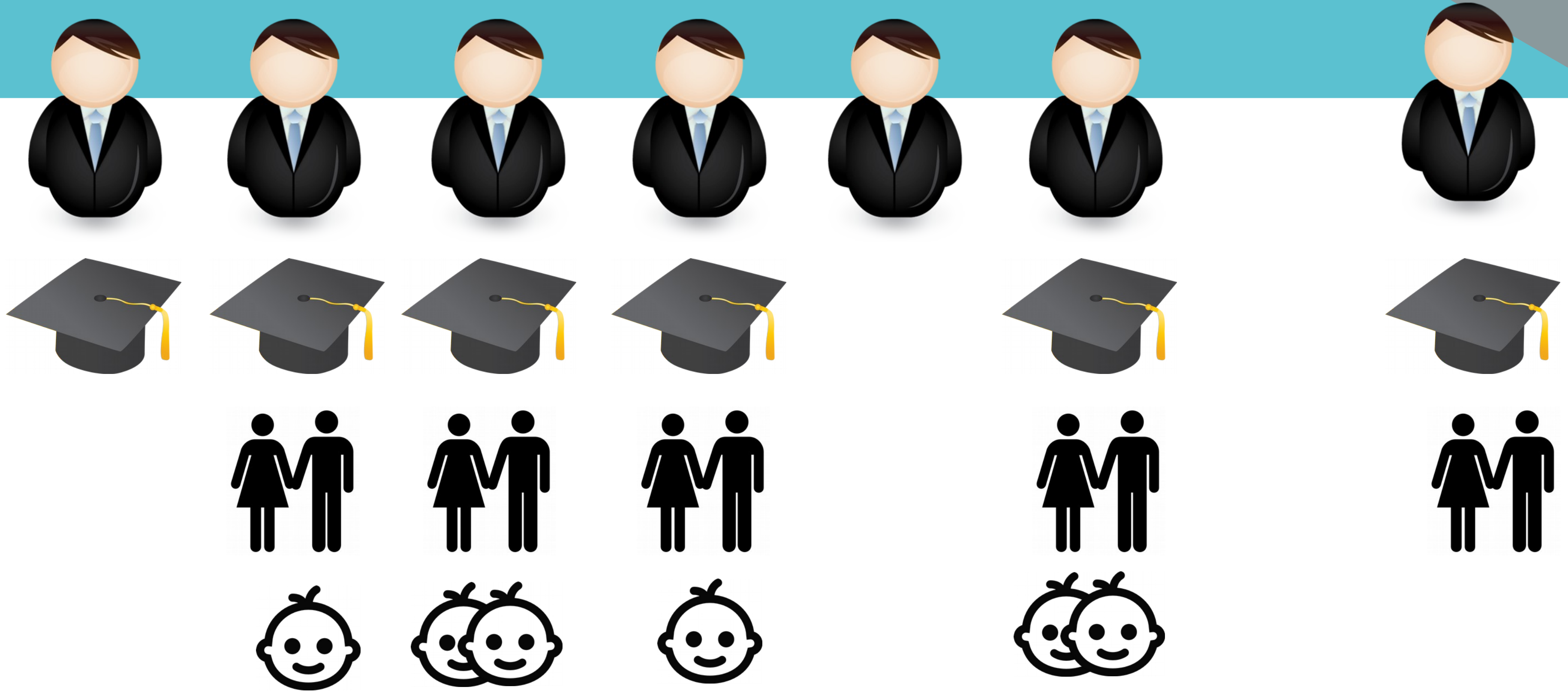
Cost models

Execution statistics

Machine learning

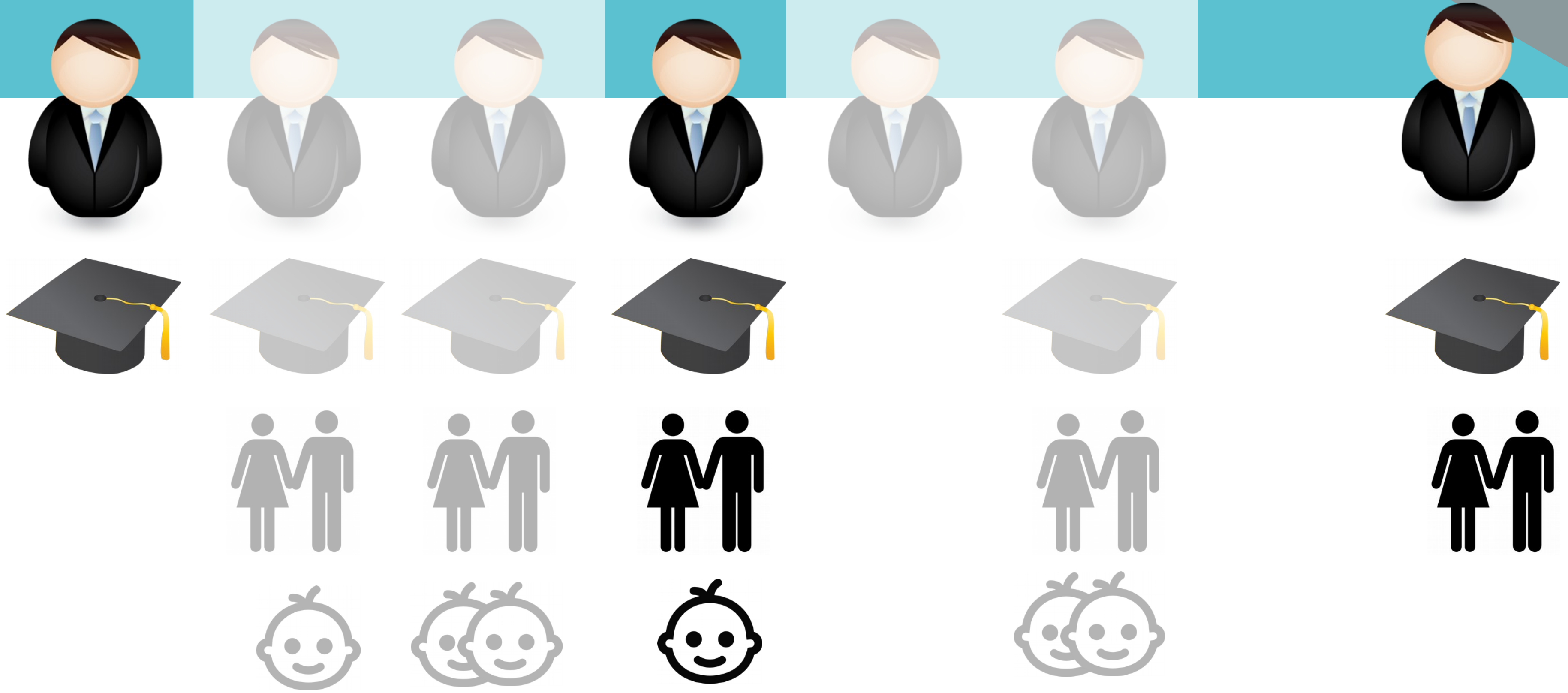


K Nearest Neighbours method



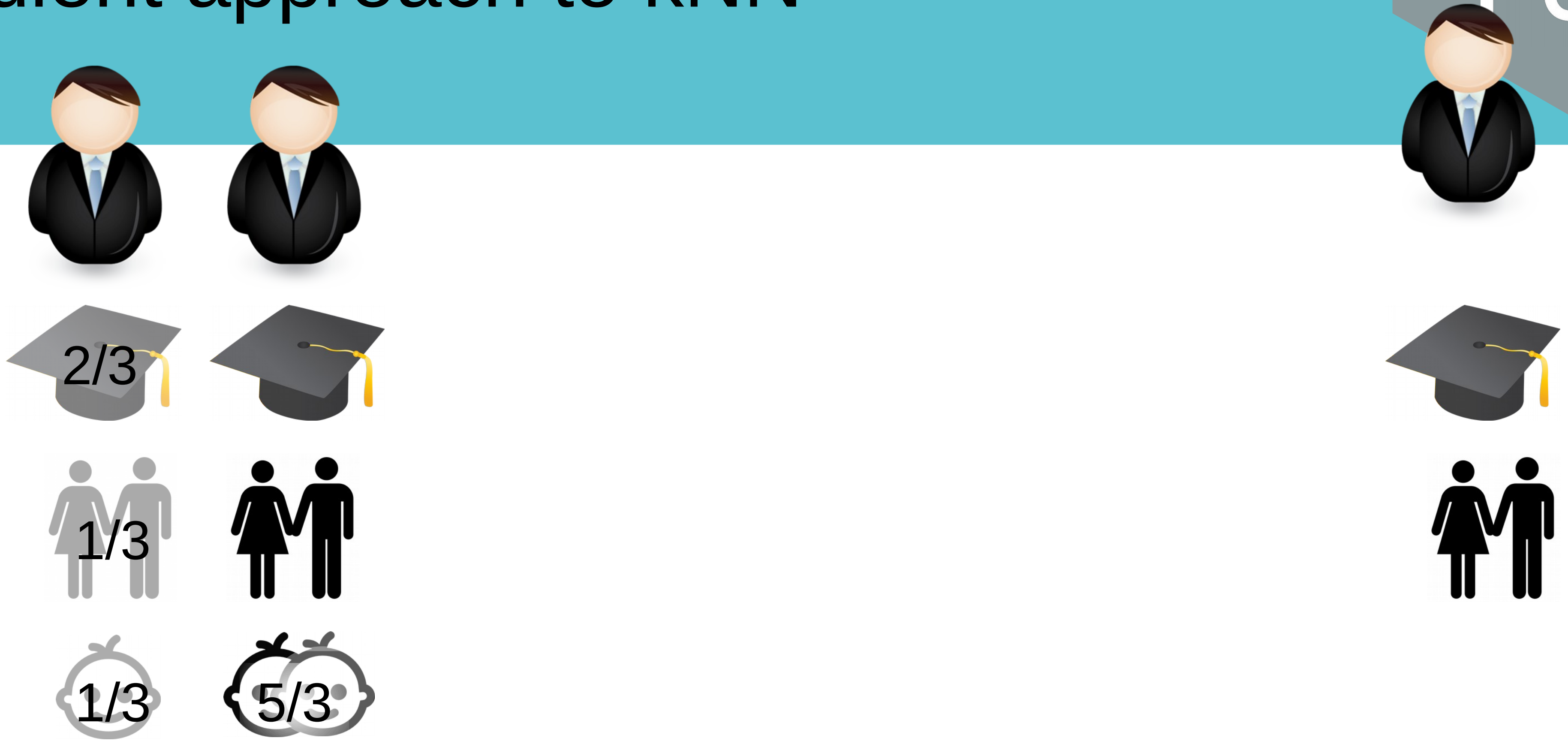
Age	25	47	55	32	22	45	28
Salary	50	120	100	80	30	90	?

K Nearest Neighbours method



Age	25	47	55	32	22	45	28
Salary	50	120	100	80	30	90	?

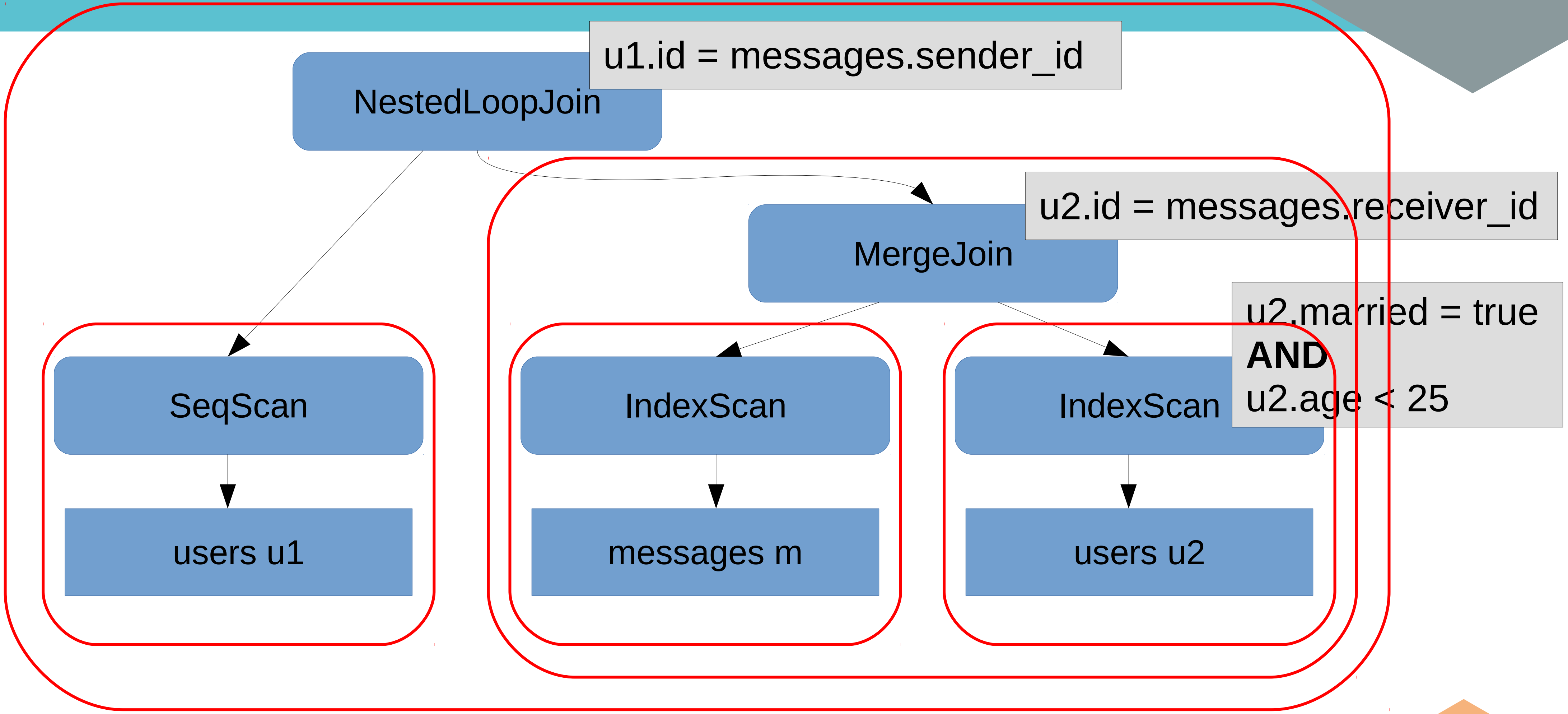
Gradient approach to kNN

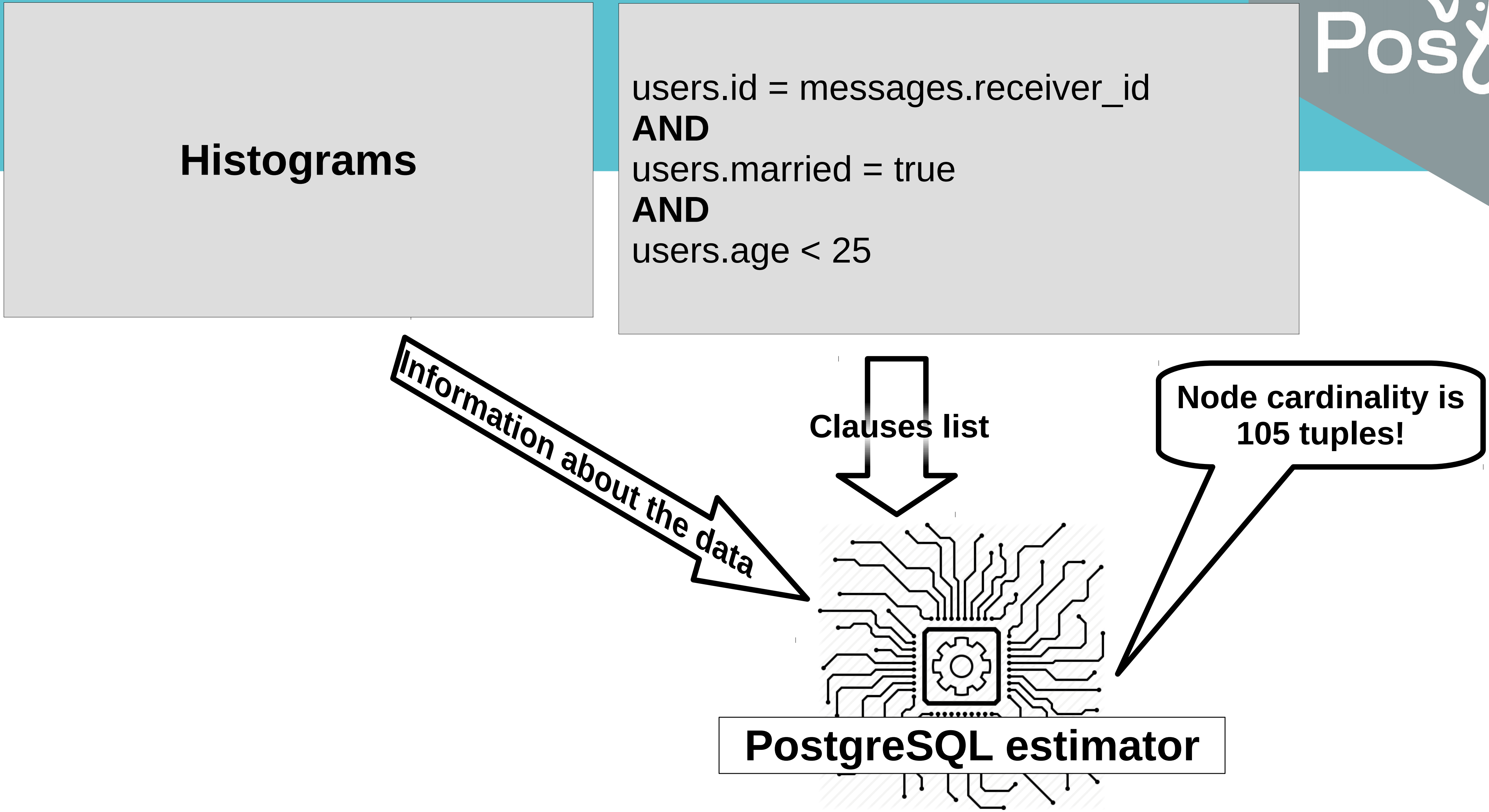


Age	27	47	28
Salary	53	103	?

How to use machine learning for adaptive query optimization?

The object is a node with its subtree





Histograms

users.id = messages.receiver_id
AND
users.married = true
AND
users.age < 25

Information about the data

Clauses list

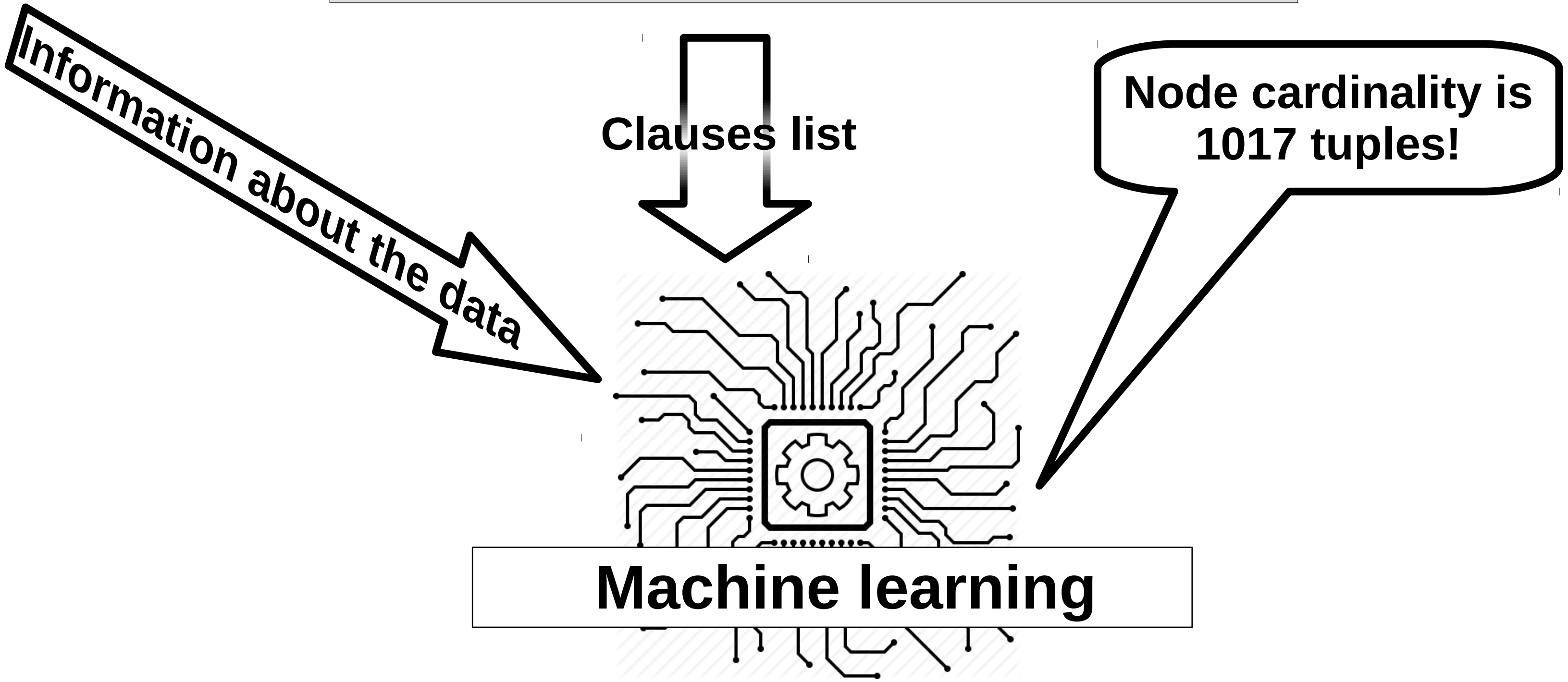
Node cardinality is 105 tuples!

PostgreSQL estimator

Clause selectivities

- 0.0001
- 0.73
- 0.23

users.id = messages.receiver_id
AND
users.married = **const**
AND
users.age < **const**



Information about the data

Clauses list

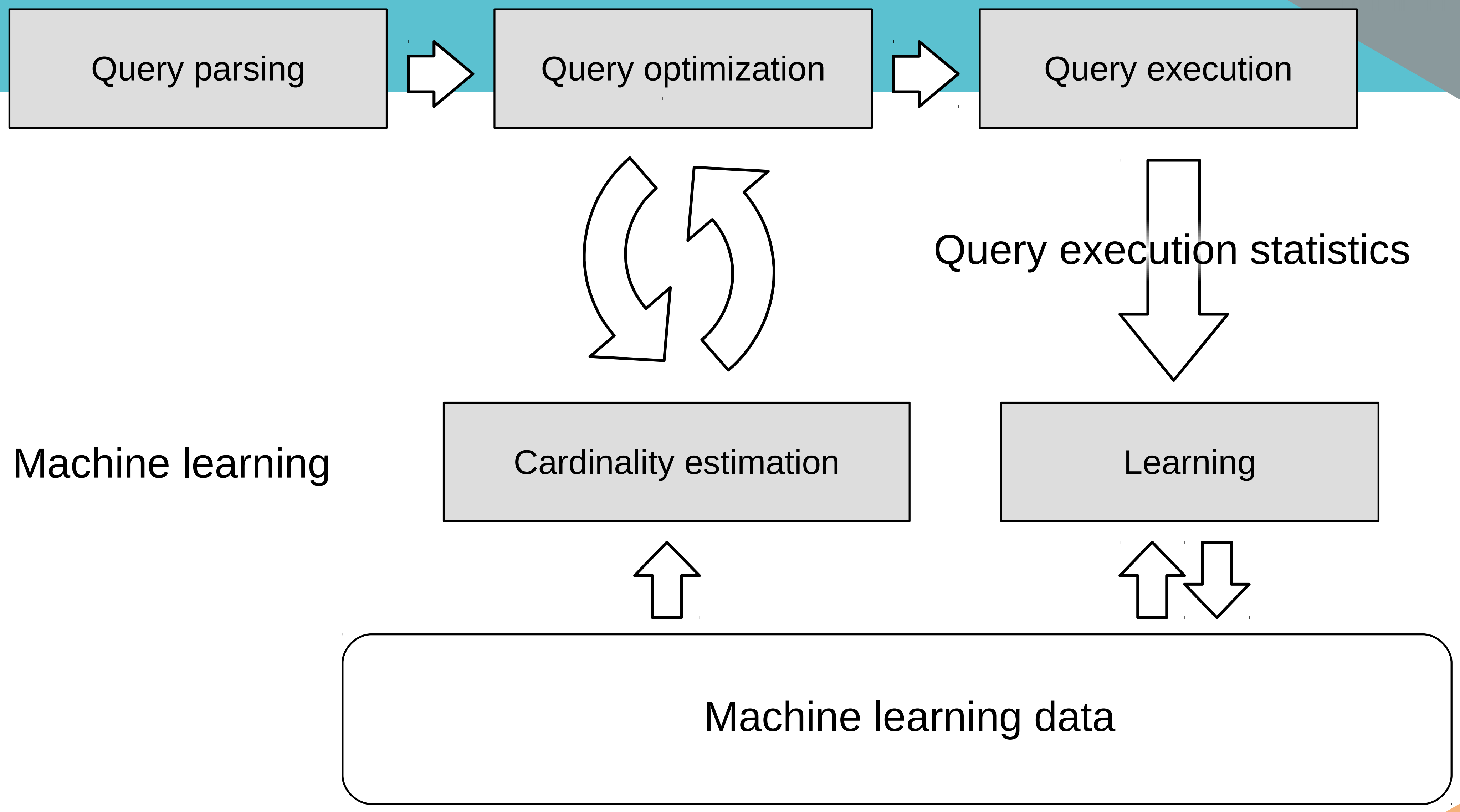
Node cardinality is 1017 tuples!

Machine learning

Machine learning problem statement

		Object is a plan node
Features	users.id = messages.receiver_id	0.0001
	users.married = **const**	0.73
	users.age < **const**	0.23
Hidden value	Node cardinality	?

Workflow


Query parsing
Query optimization
Query execution
Query execution statistics
Machine learning
Cardinality estimation
Learning
Machine learning data

Theoretical properties

- Will it converge?

Yes, in the finite number of steps

- How fast will it converge?

Don't know (in practice in a few steps)

- What guarantees on obtained plans or regressor do we have?

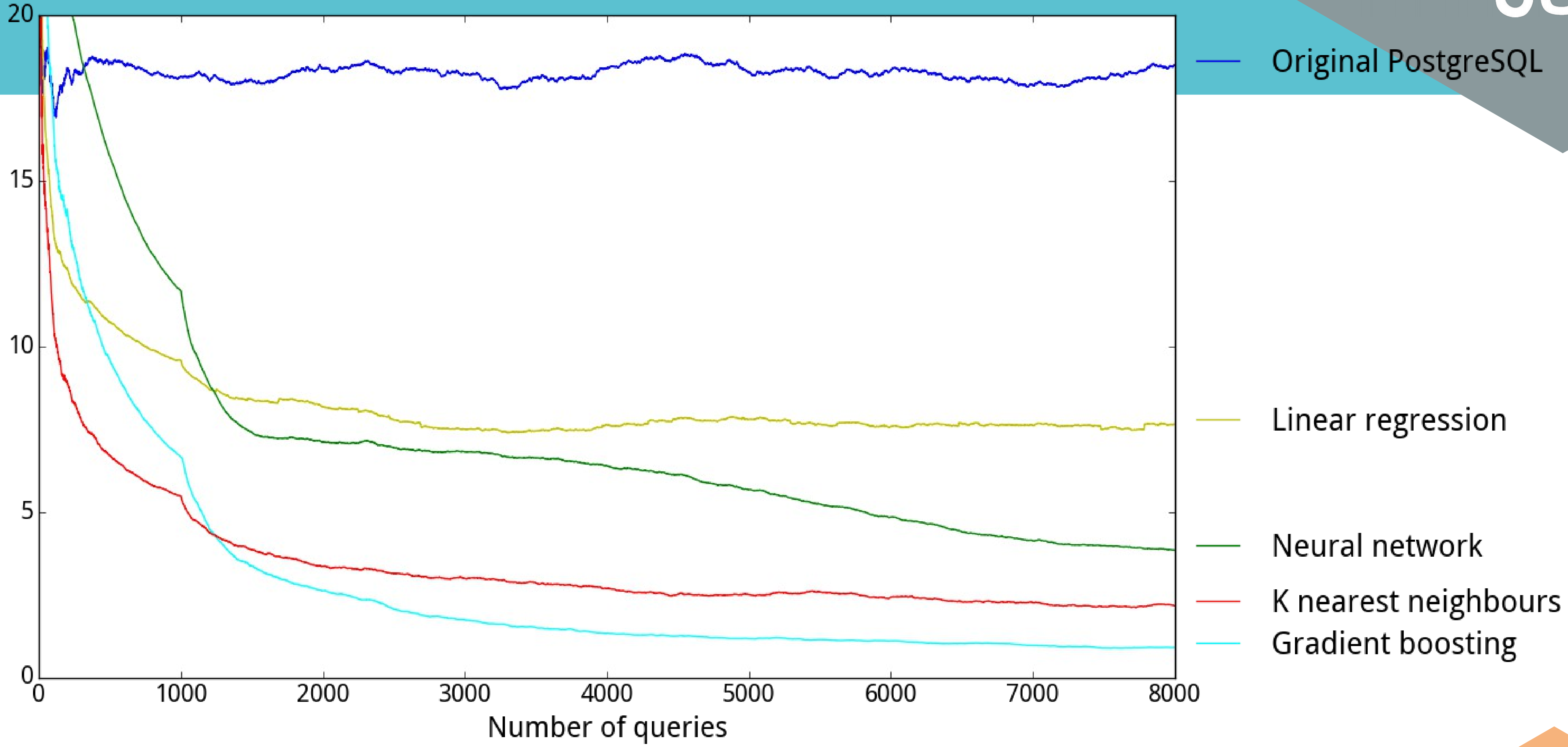
Predictions are correct for all executed paths

With perfect cost model obtained plans are not worse

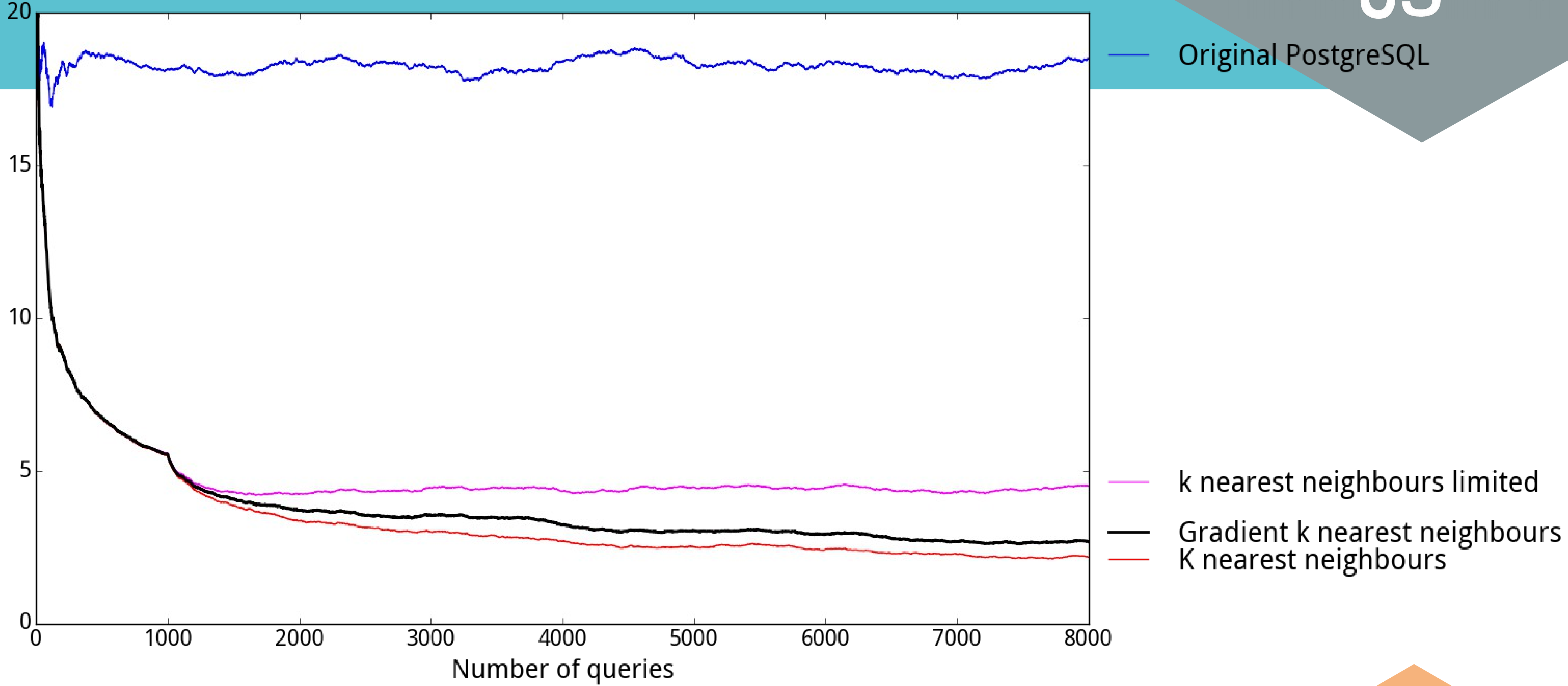
How much can it improve PostgreSQL performance?

Experimental evaluation

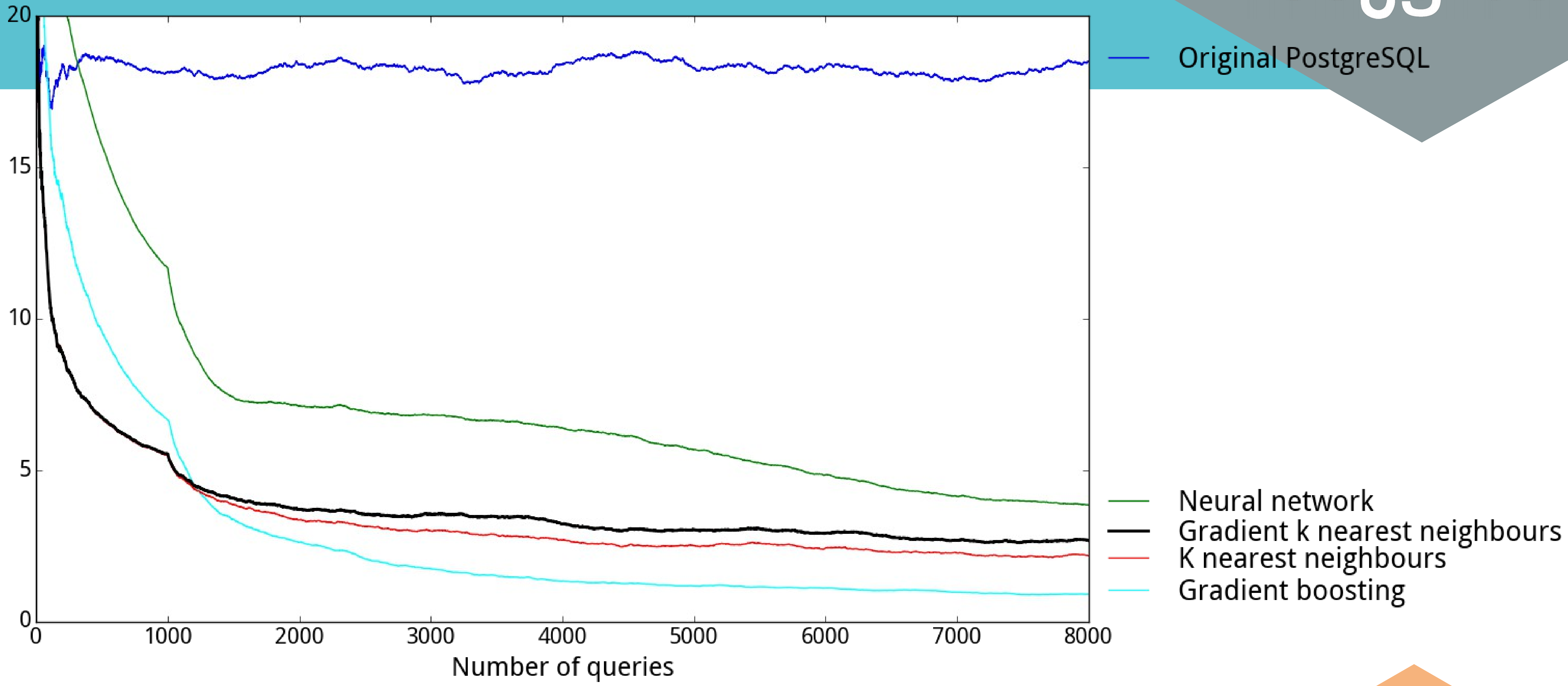
Estimation error



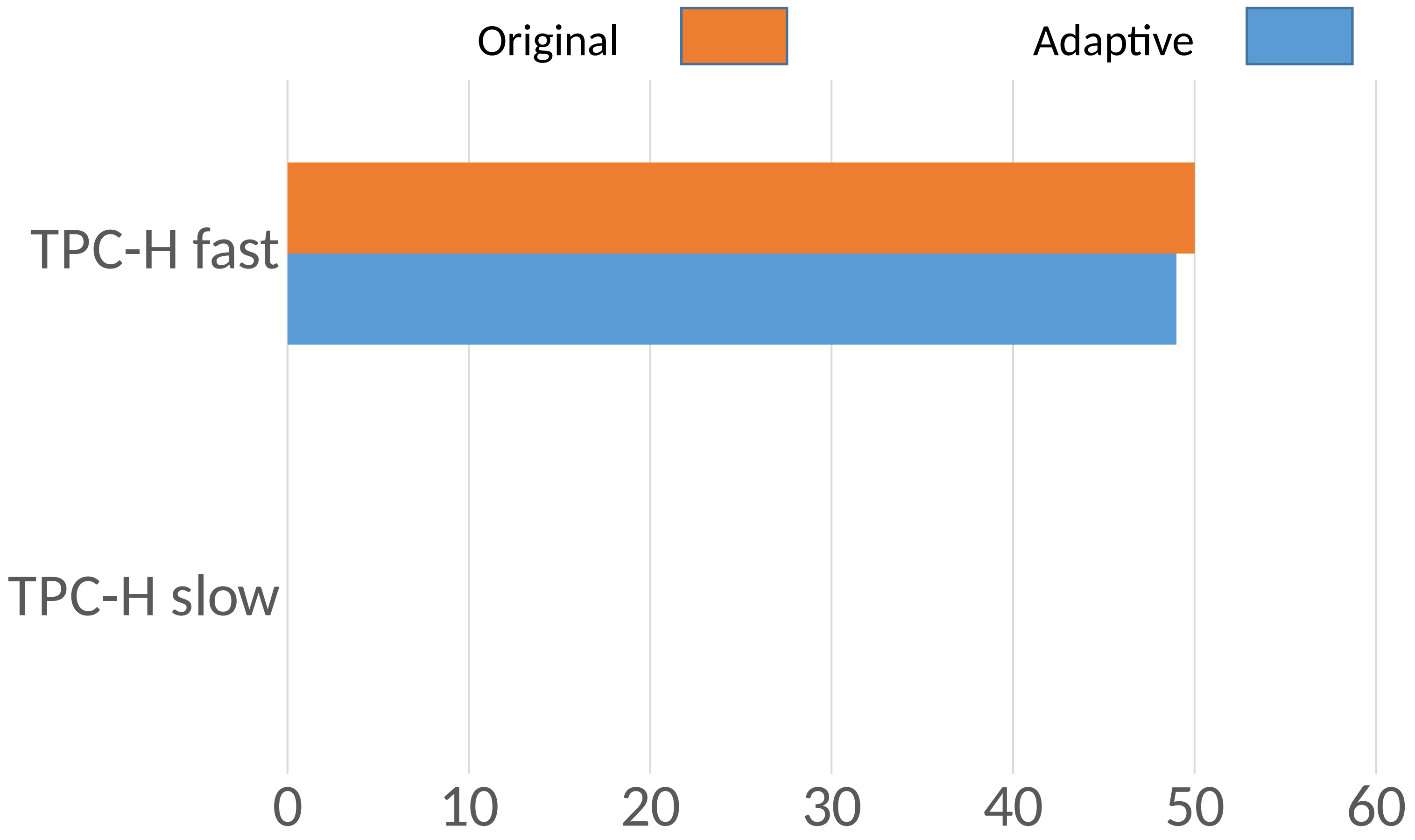
Estimation error



Estimation error

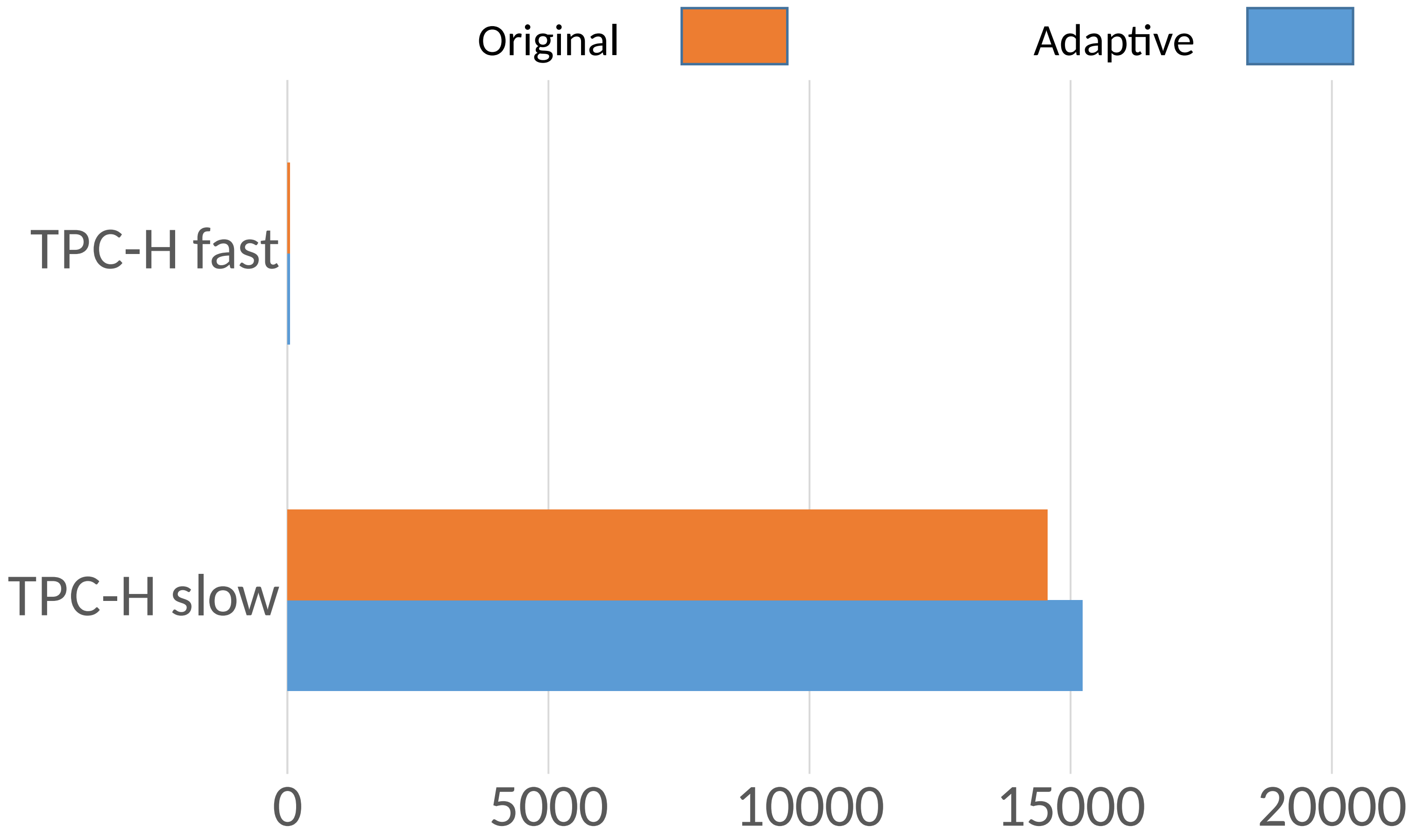


Performance improvement



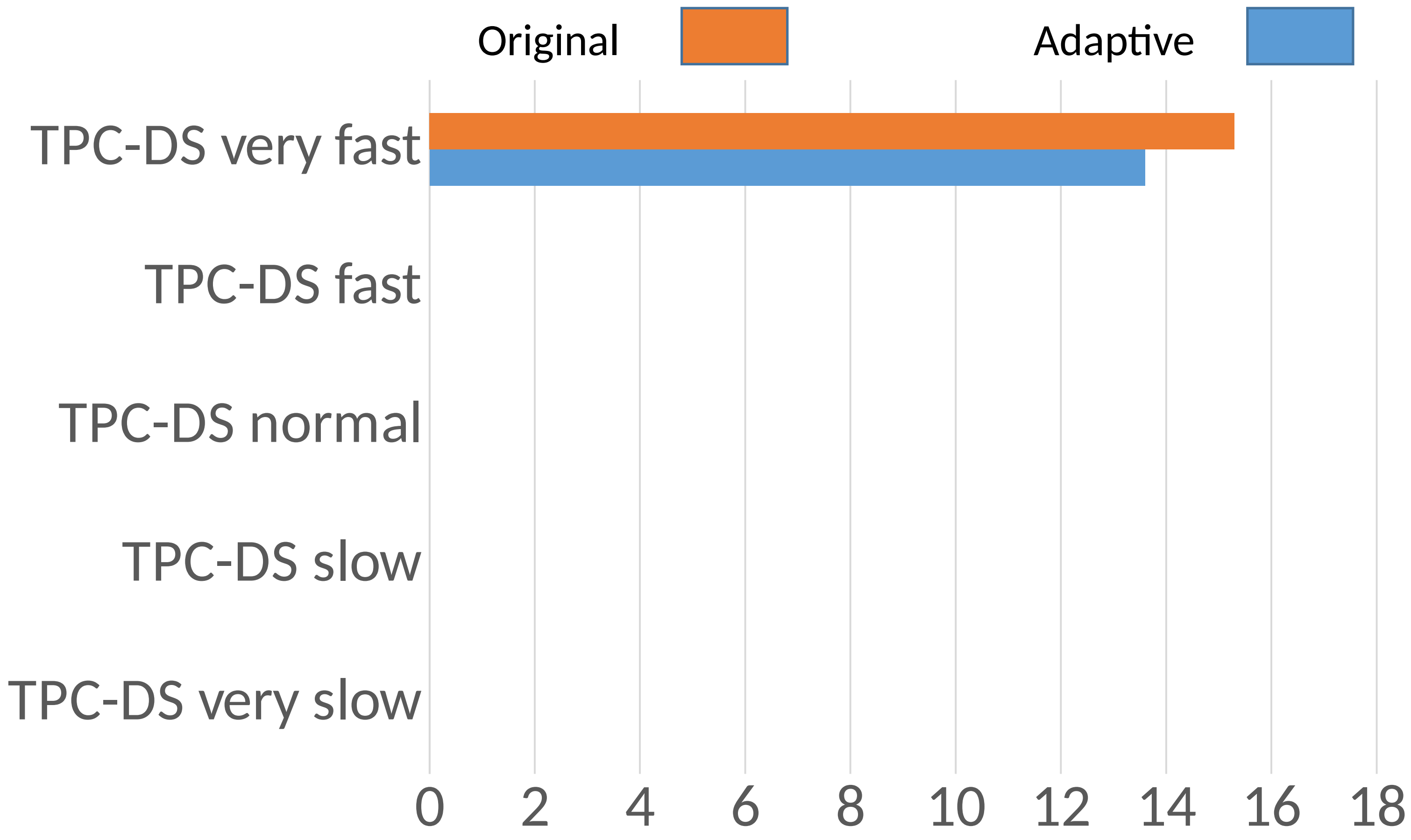
+1.3%

Performance improvement



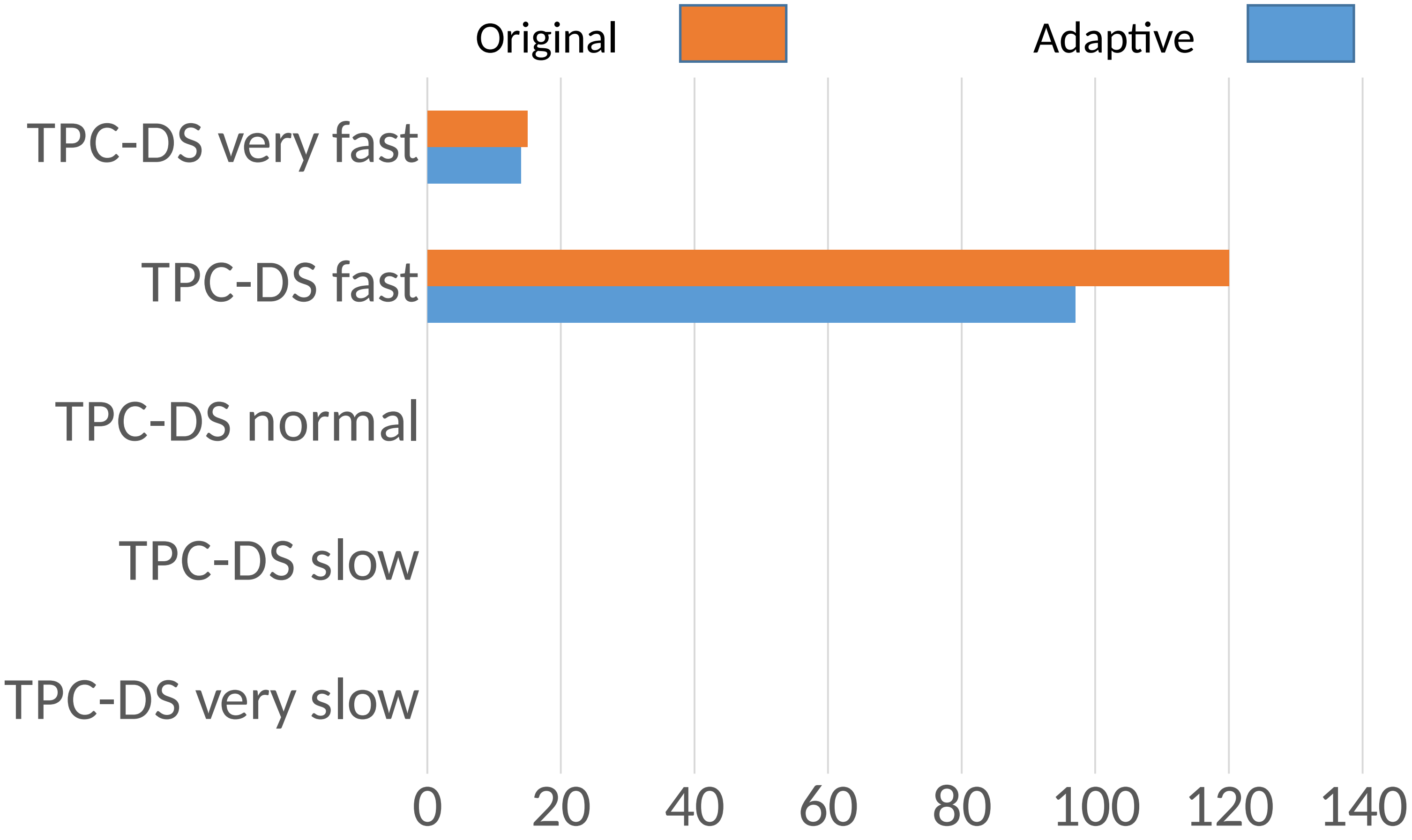
-4.4%

Performance improvement



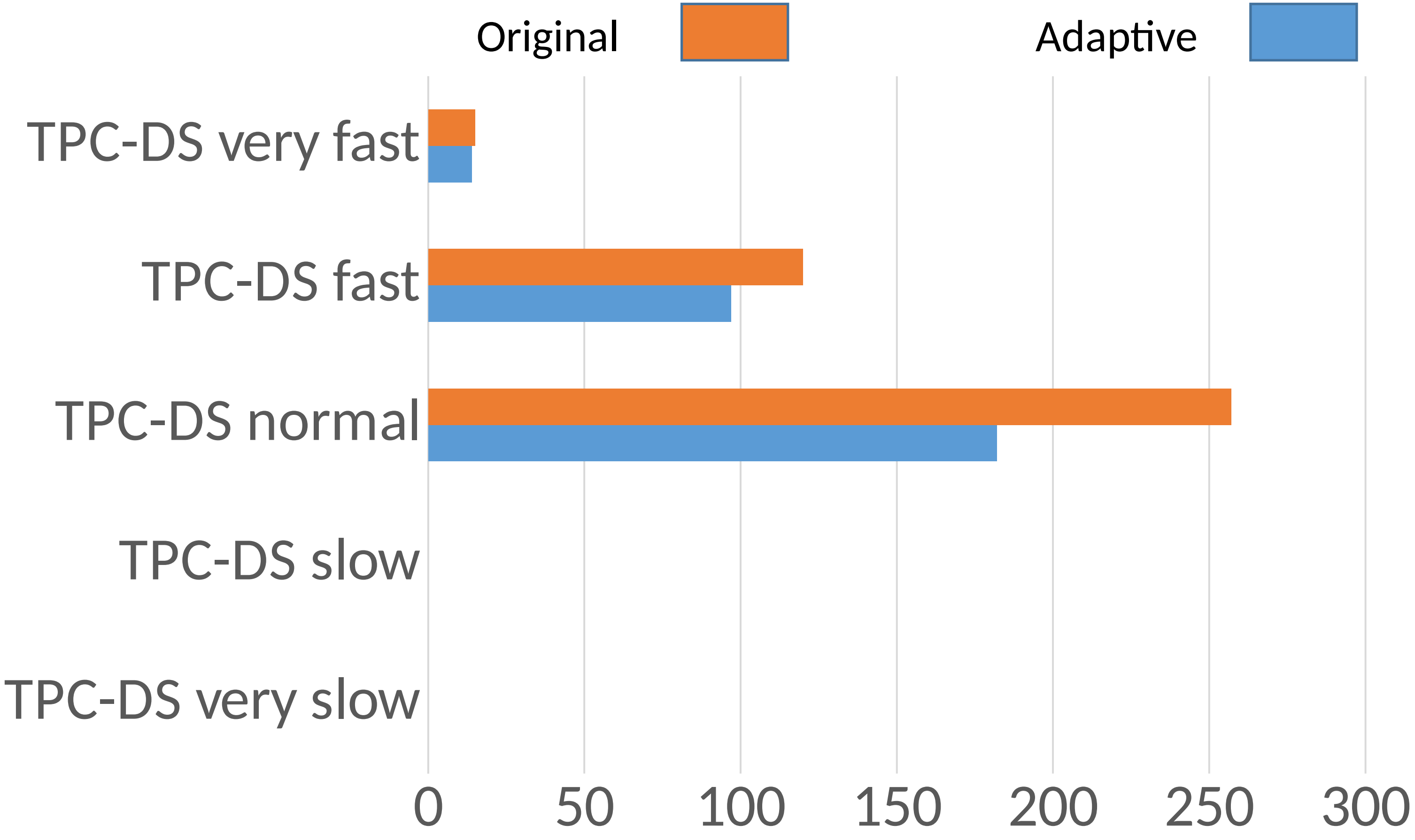
+12%

Performance improvement



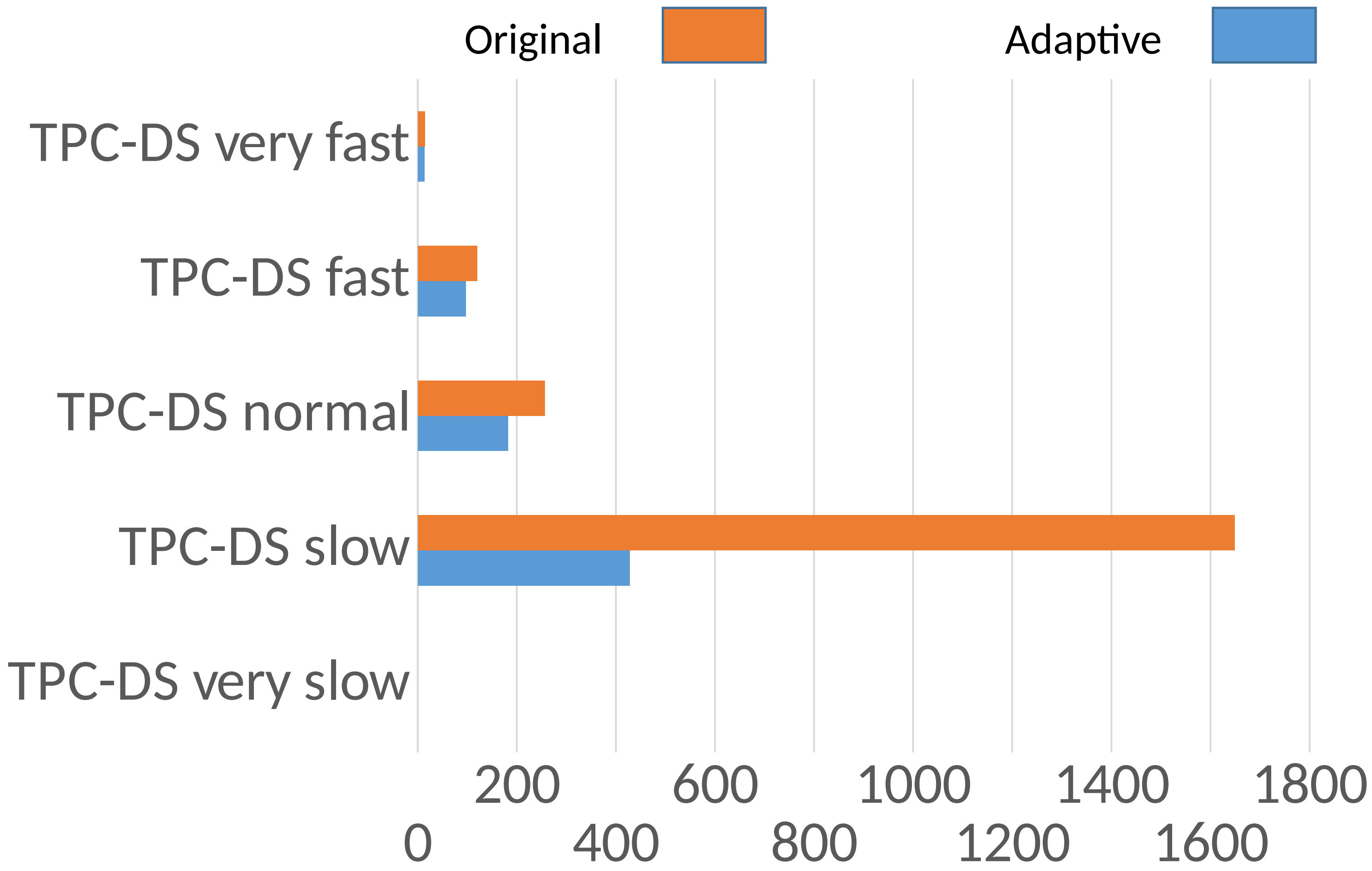
+24%

Performance improvement



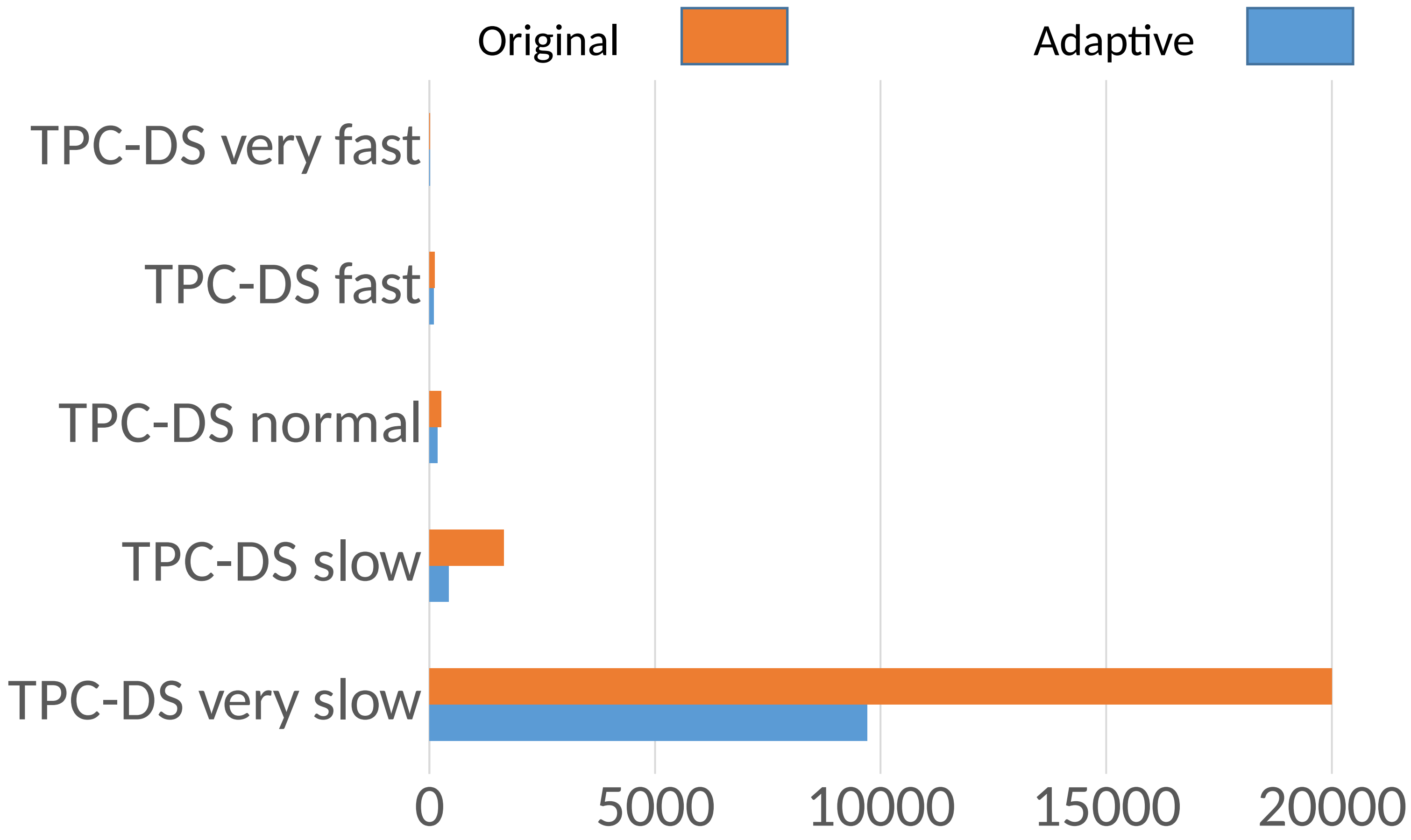
+41%

Performance improvement



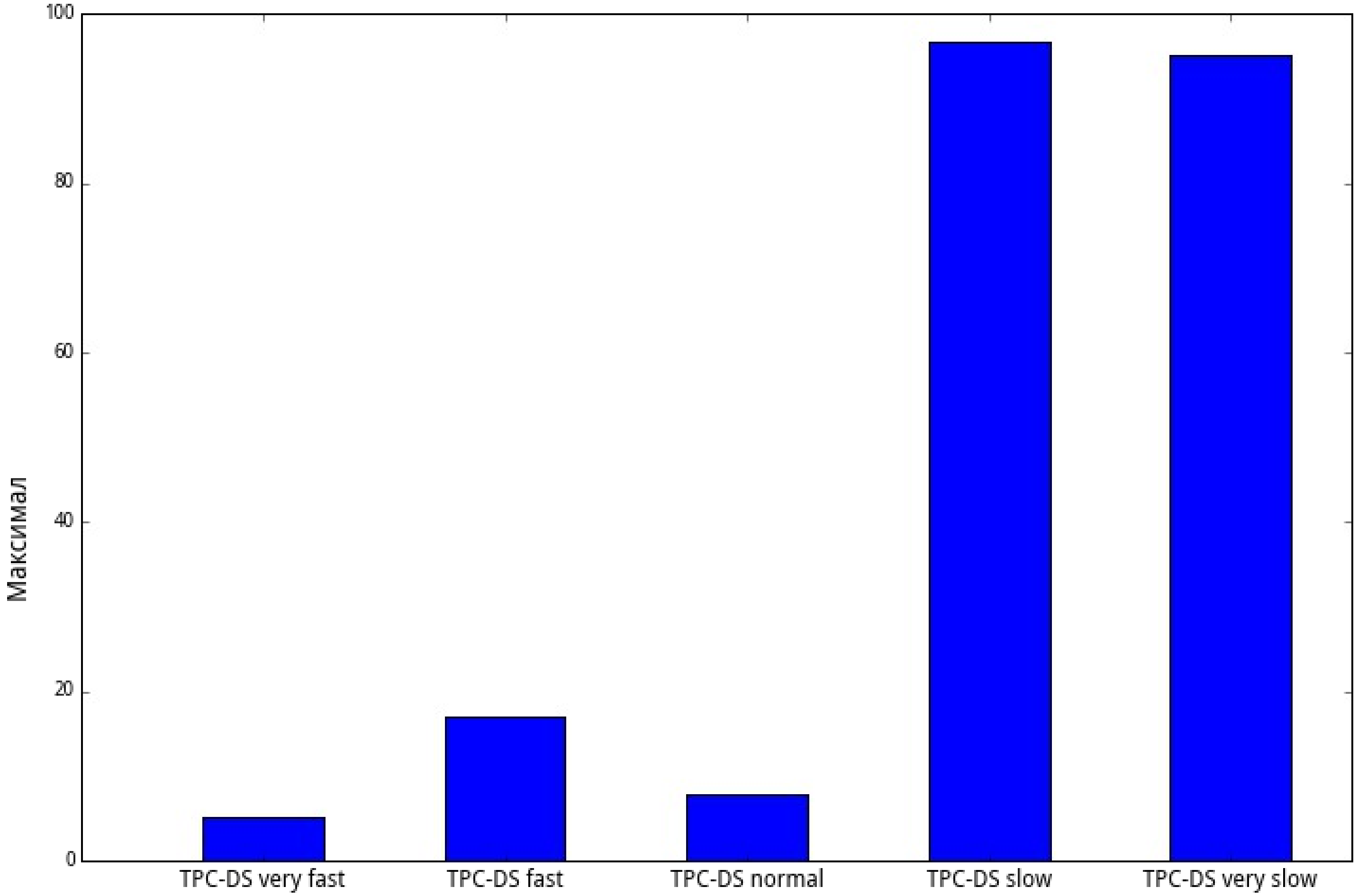
+285%

Performance improvement



+115%

Maximum acceleration



Overheads

Experimental evaluation

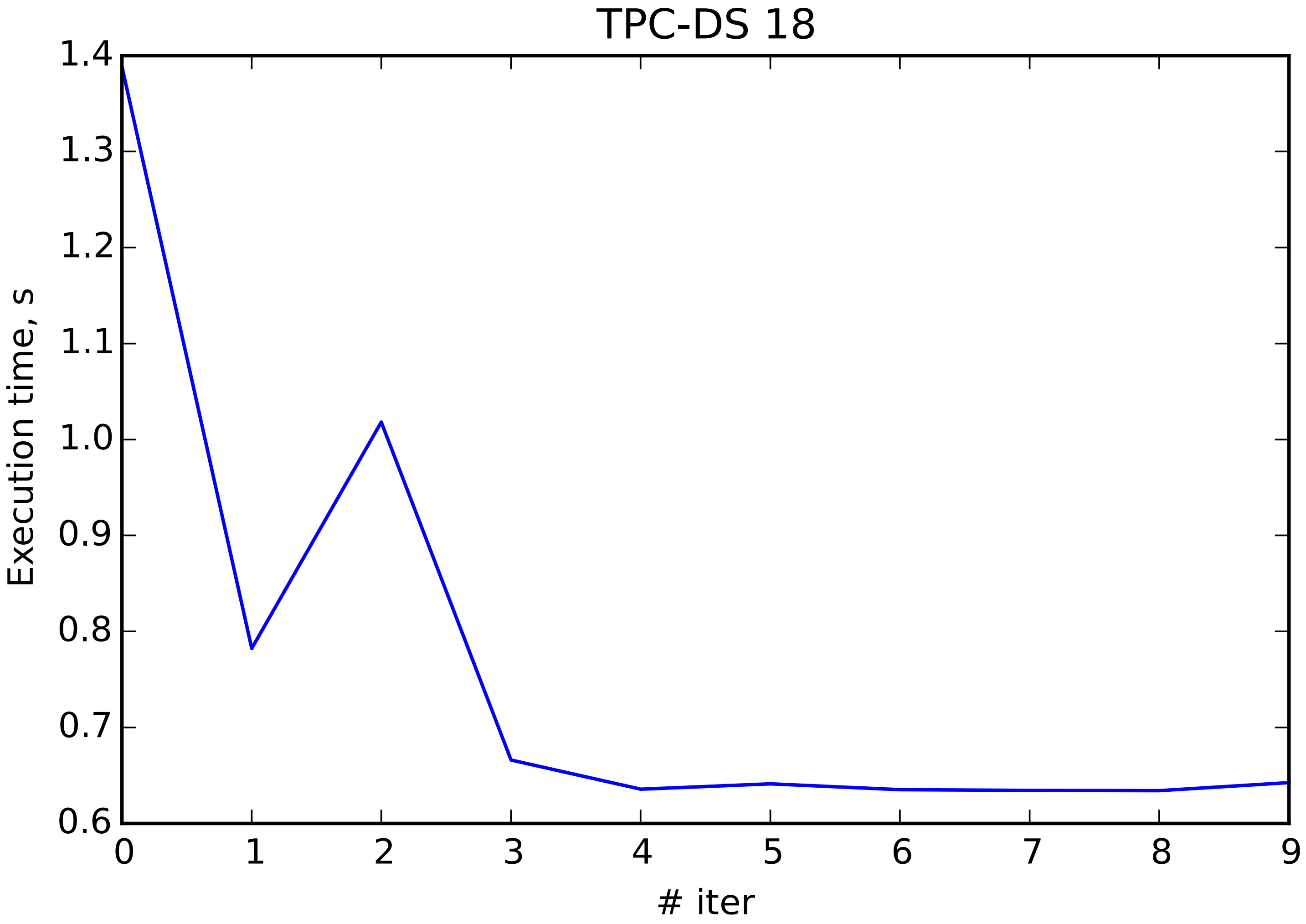
Slowdown for genetic algorithm is not more than 2 seconds

Slowdown for dynamic programming is not more than 30 ms

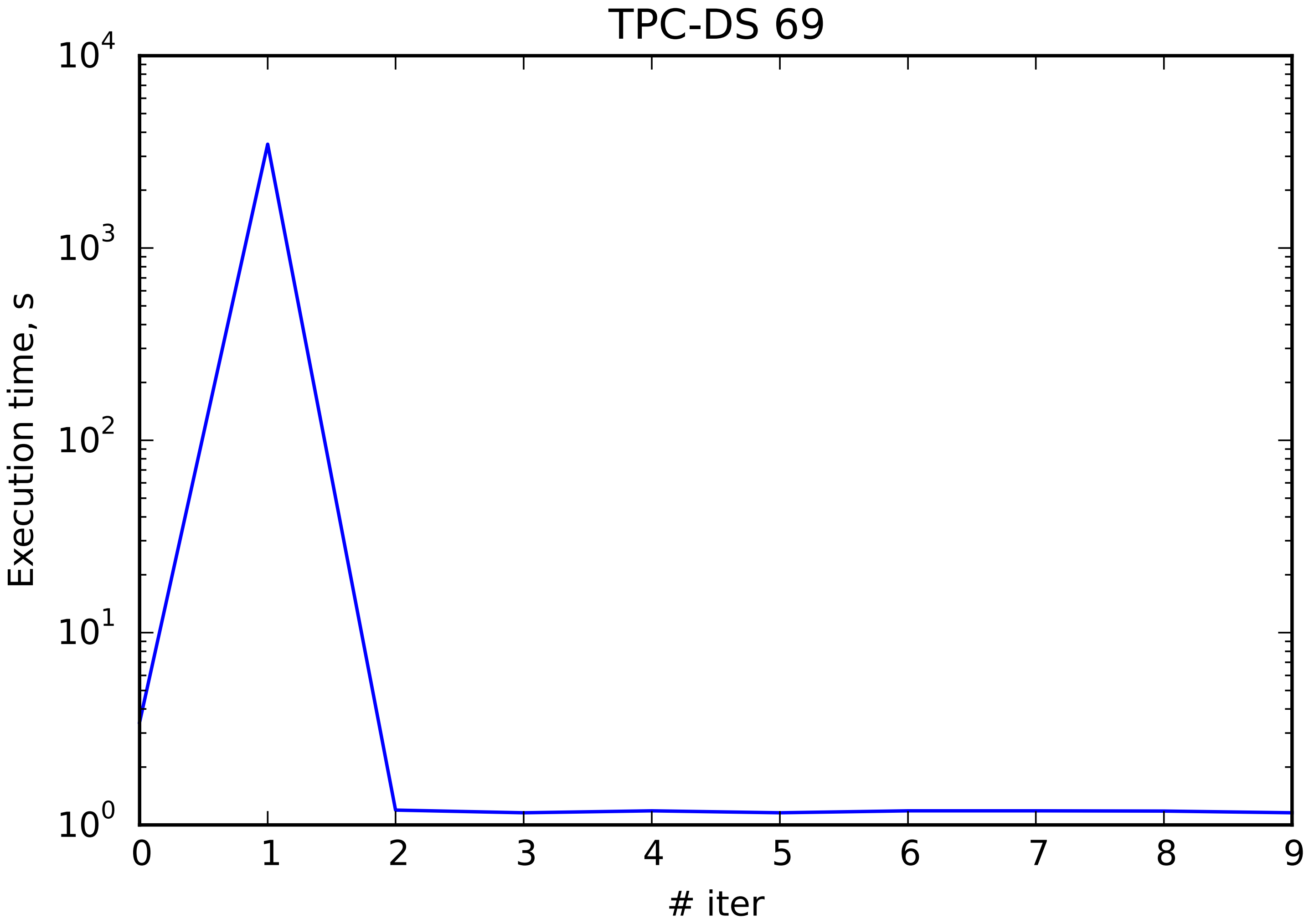
Applicability

Complex analytical queries
with a repeating pattern.

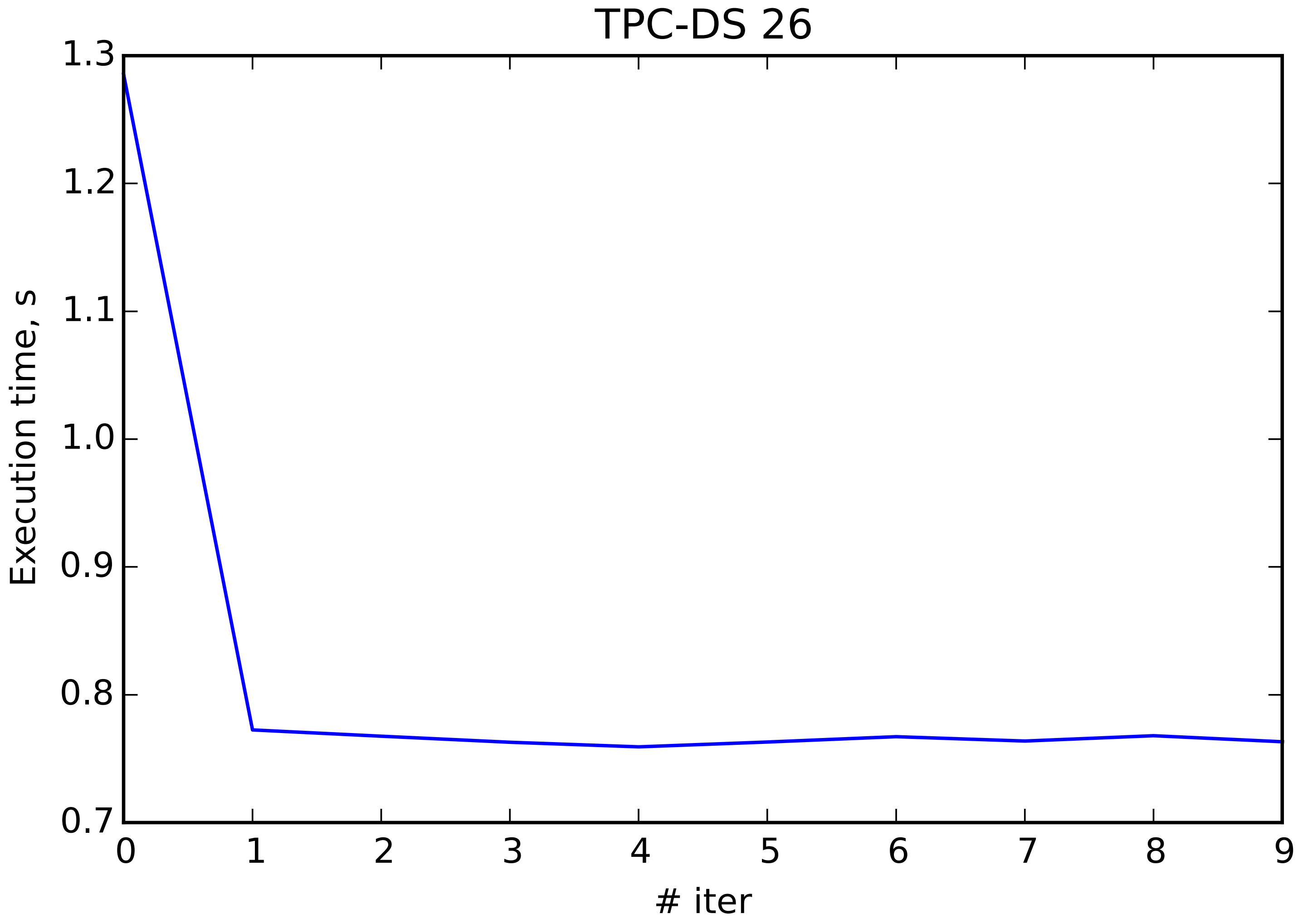
Learning progress



Learning progress



Learning progress



AQO: adaptive query optimization

Current code for vanilla PostgreSQL (extension + patch):
https://github.com/tigvarts/aqo

Available in Postgres Pro Enterprise

AQO: adaptive query optimization

For some queries we don't need AQO.

So we need a mechanism to determine whether the query needs AQO.

AQO: adaptive query optimization

Query type is the set of queries, which differ only in their constants.

Query type:
SELECT * **FROM** users **WHERE** age > **const** **AND** city = **const**;

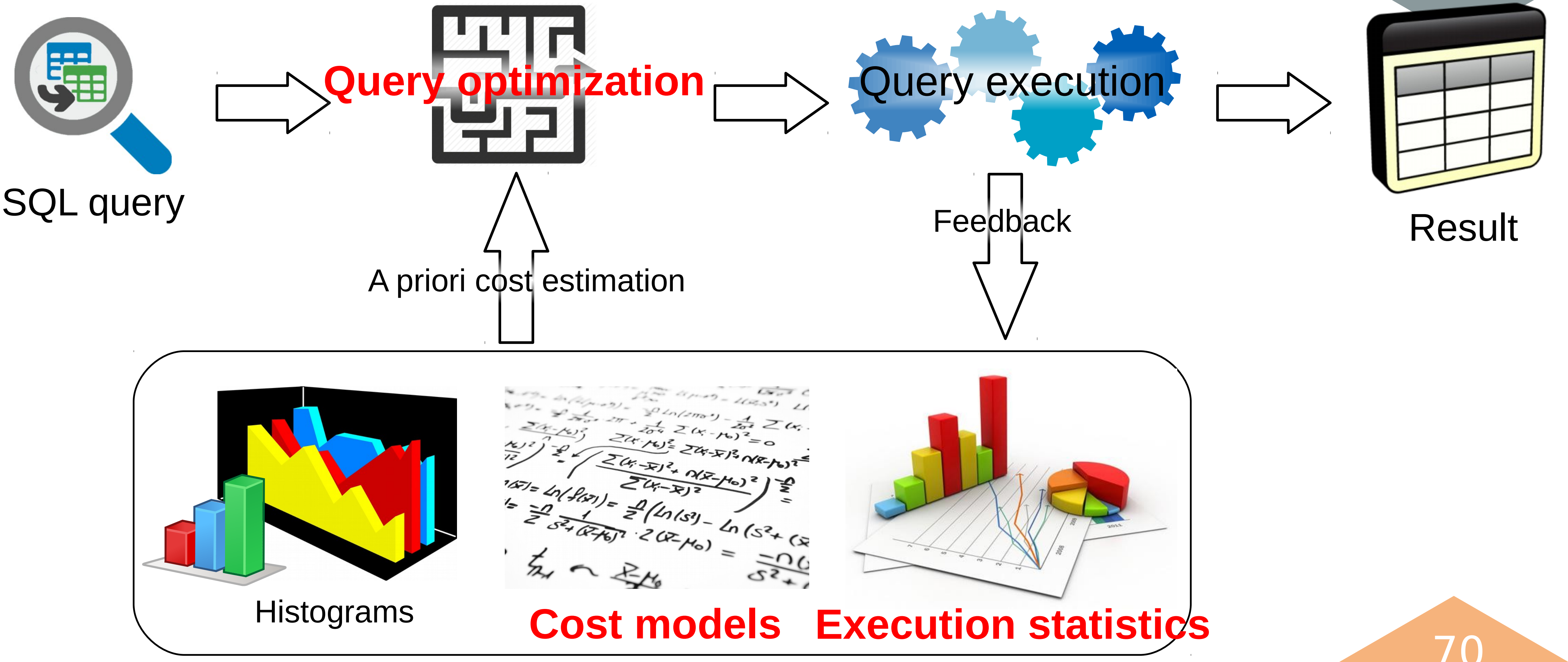
Queries:
SELECT * **FROM** users **WHERE** age > 18 **AND** city = 'Moscow';
SELECT * **FROM** users **WHERE** age > 65 **AND** city = 'Kostroma';
…

AQO: adaptive query optimization

aqo.mode:

- Disabled for all query types
- Enabled for all query types
- Use manual settings for known query types, ignore others
- Use manual settings for known query types, tries to tune others automatically

What is next?



SQL query

Query optimization

Query execution

Result

A priori cost estimation

Feedback

Histograms

Cost models

Execution statistics

Questions

Contacts:

- o.ivanov@postgrespro.ru
- +7 (916) 377-55-63

PROFESSIONAL Postgres

Postgres Professional

http://postgrespro.ru/

+7 495 150 06 91

info@postgrespro.ru

postgrespro.ru