Адаптивная оптимизация запросов

Олег Иванов

Postgres Professional

HighLoad++

Профессиональная конференция разработчиков высоконагруженных систем

Что такое оптимизация запросов?

Как современные реляционные СУБД оптимизируют запросы?

Что такое адаптивная оптимизация запросов?

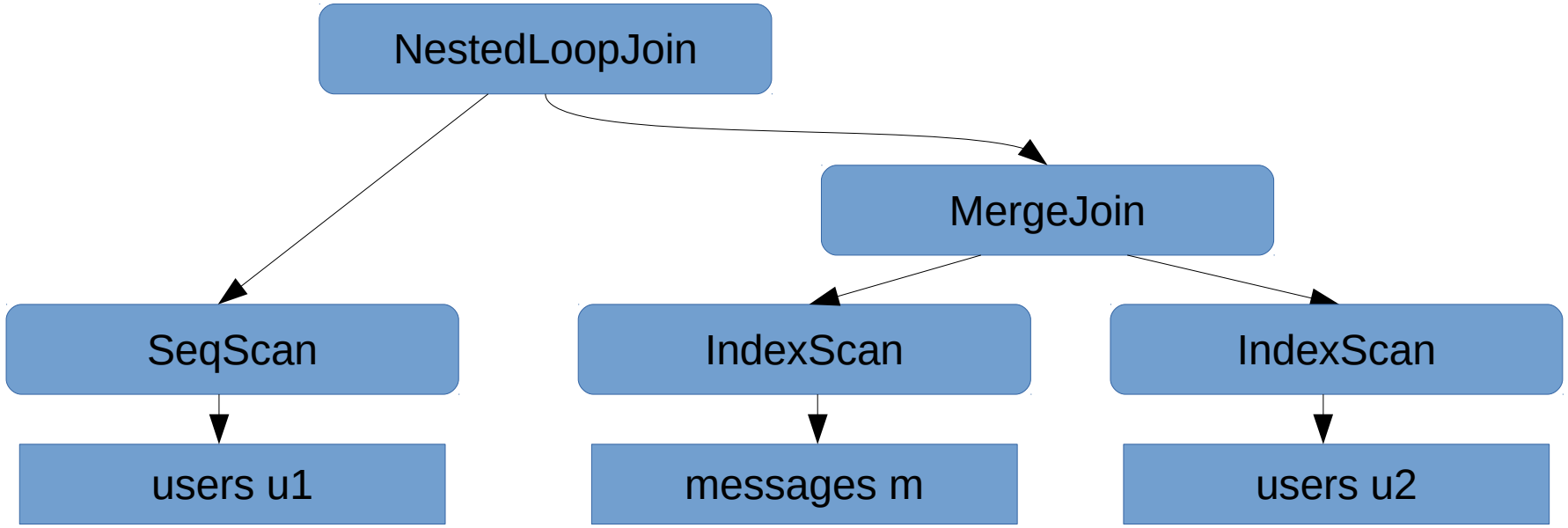
Машинное обучение, метод ближайших соседей.

Как использовать машинное обучение для адаптивной оптимизации запросов?

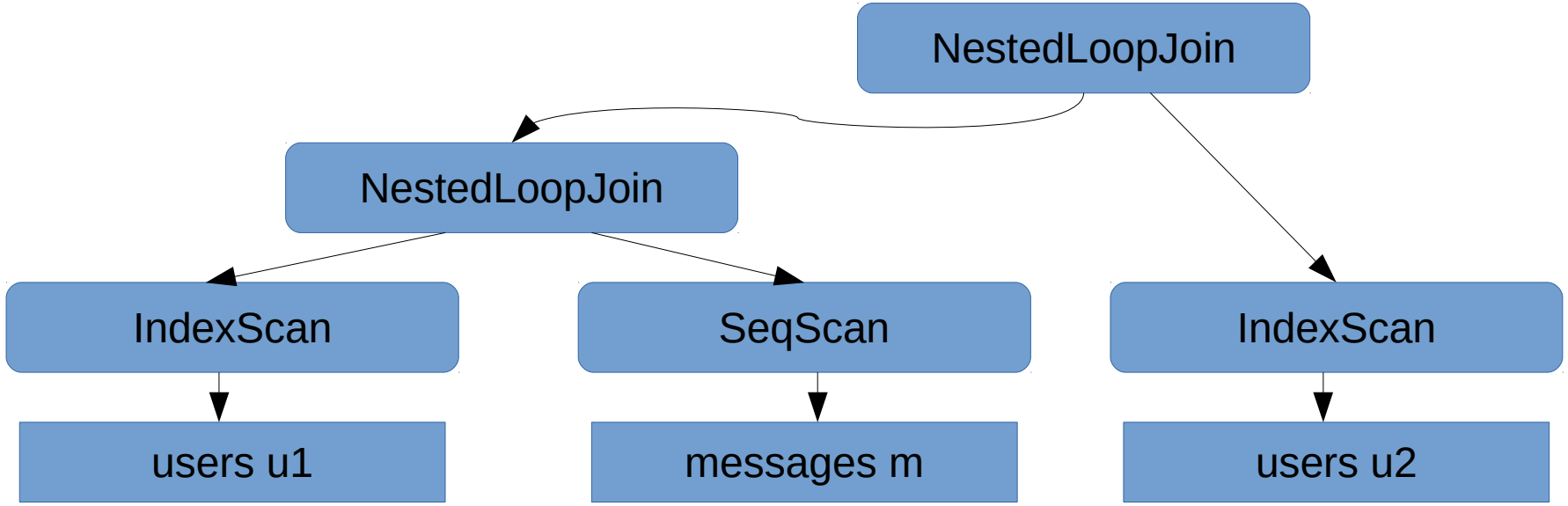
Насколько это улучшает производительность СУБД?

Что такое оптимизация запросов?

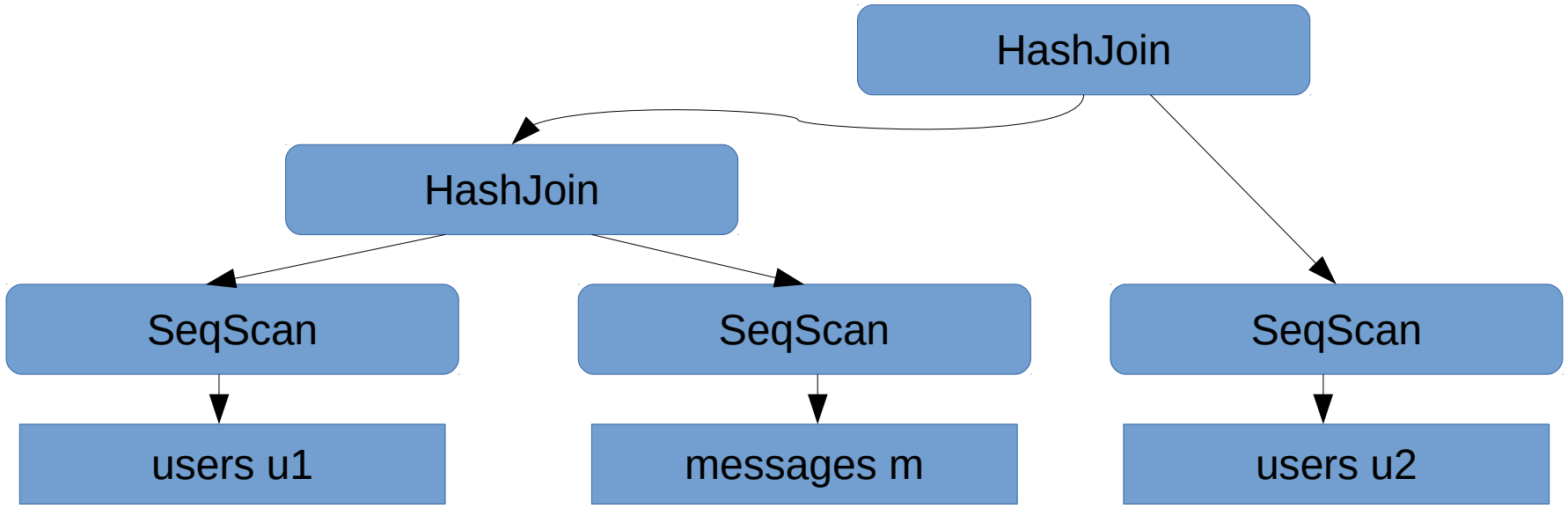
```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



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



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



```
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)
```

Стоимость выполнения плана

```
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)
```

Стоимость выполнения вершины плана

Мощность (cardinality) вершины плана

Стоимостная оптимизация запросов

System R (1974)

Среди всех планов выполнения запроса
выбираем план с наименьшей стоимостью

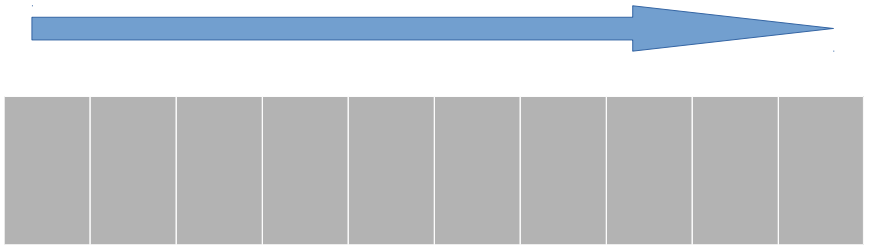
PostgreSQL

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

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

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

SeqScan



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

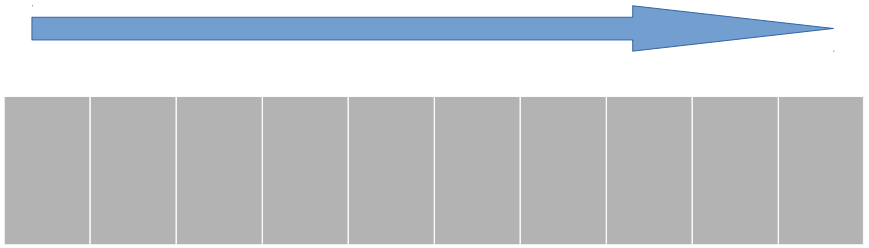
$$n_s = N_{pages}$$

$$n_o = N_{tuples}$$

IndexScan

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

SeqScan

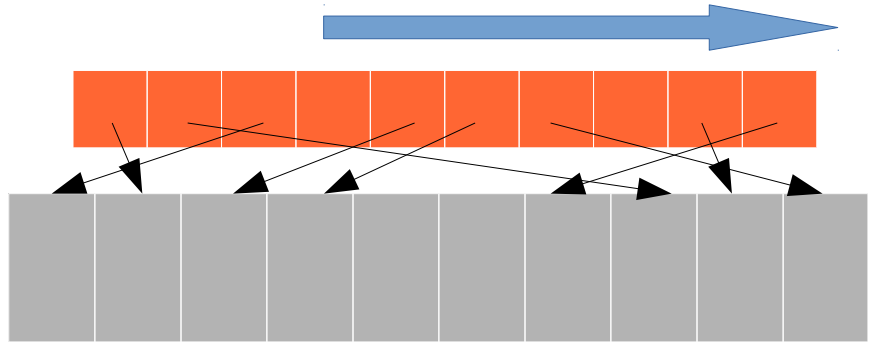


$$Cost = n_s c_s + n_o \cdot c_o$$

$$n_s = N_{pages}$$

$$n_o = N_{tuples}$$

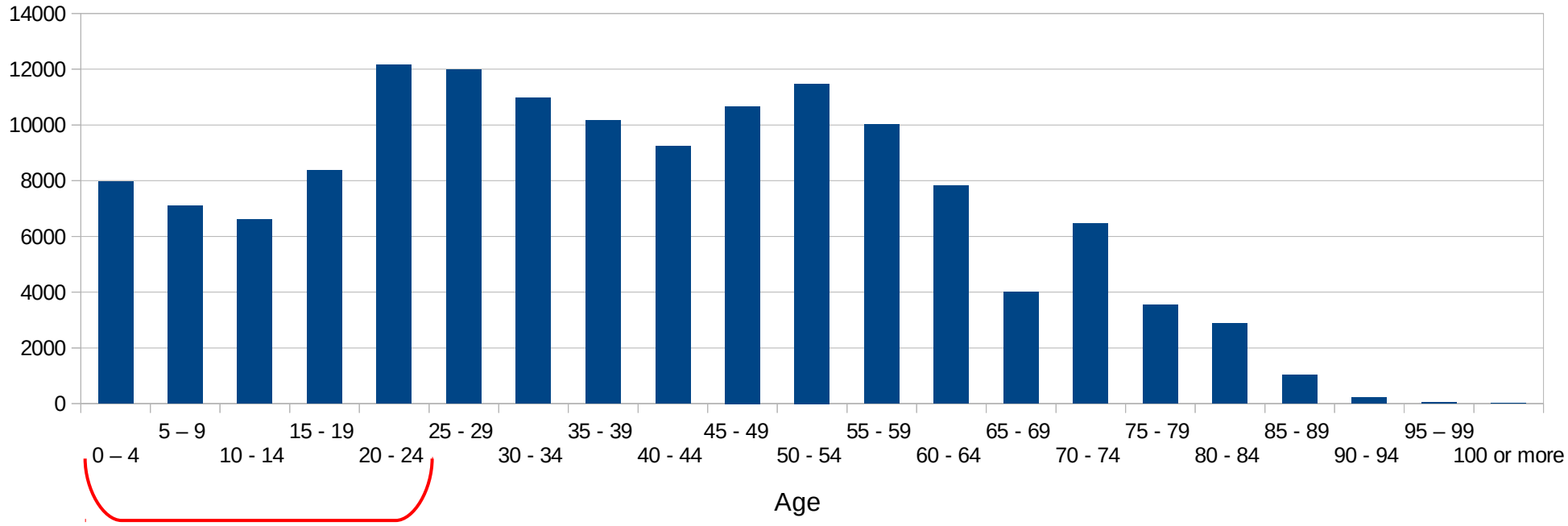
IndexScan



$$Cost = n_r \cdot c_r$$

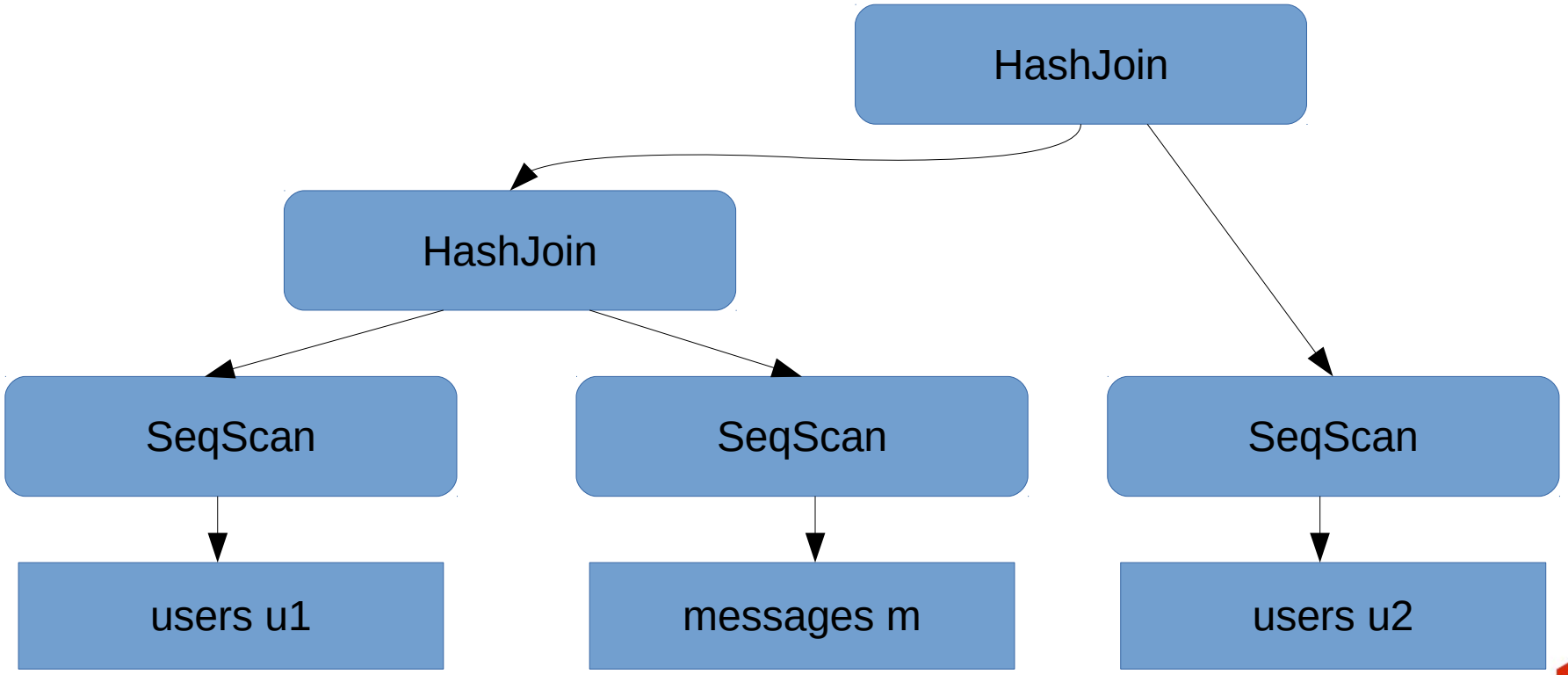
$$n_r = Cardinality$$

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

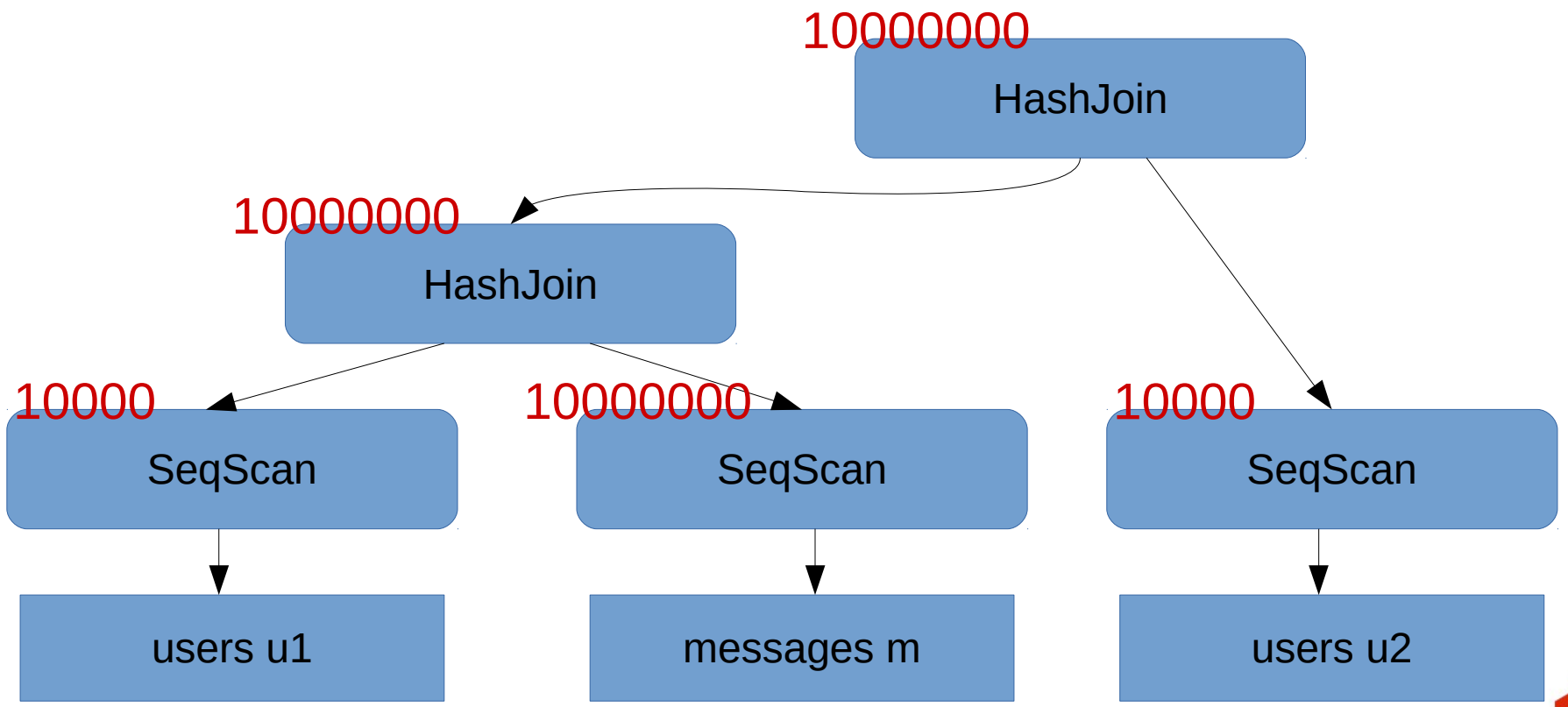


Cardinality

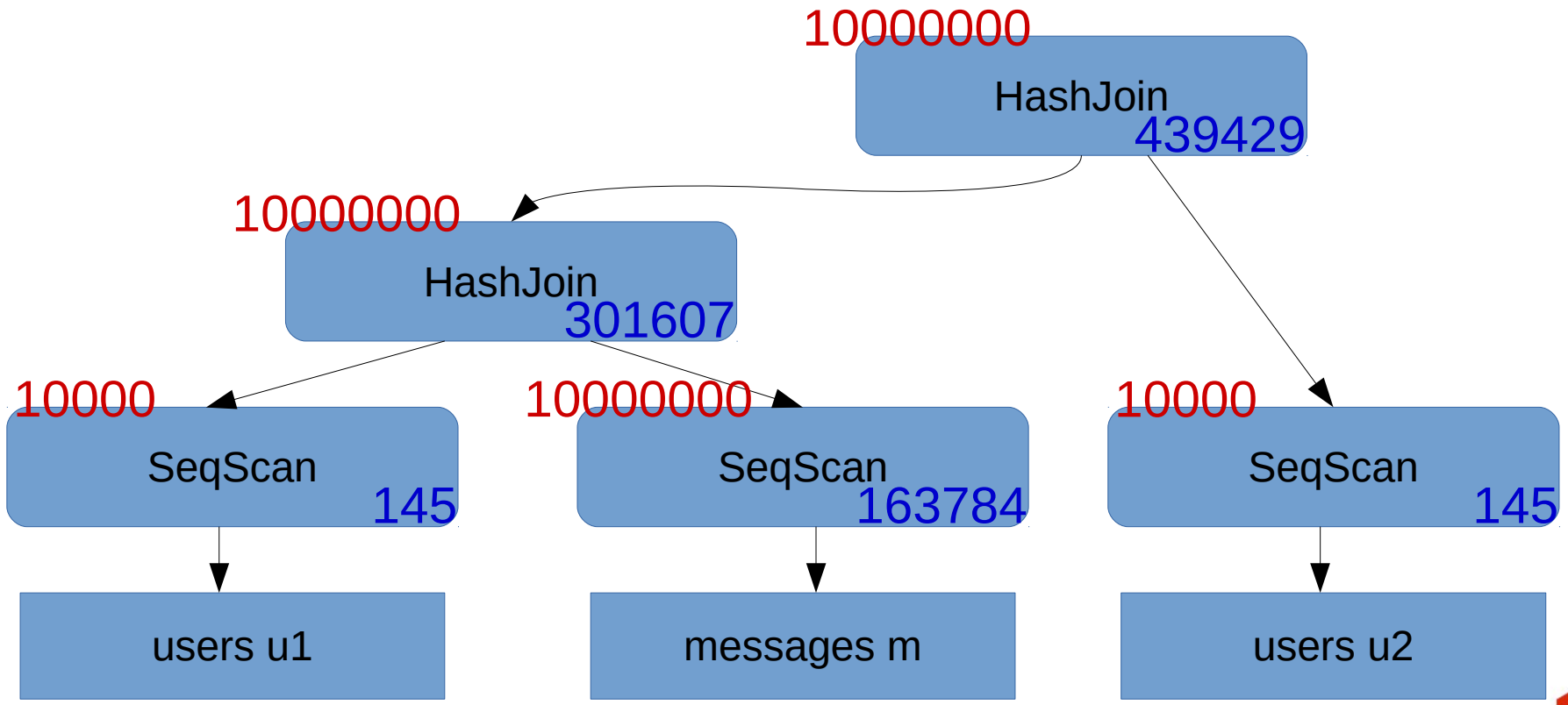
```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```



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

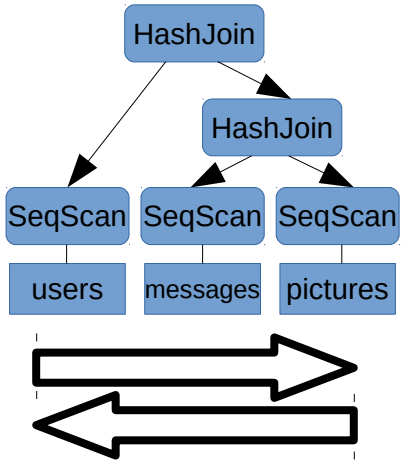


```
SELECT *
FROM users AS u1, messages AS m, users AS u2
WHERE u1.id = m.sender_id AND m.receiver_id = u2.id;
```

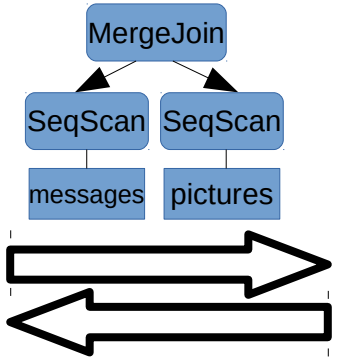


Метод оптимизации

Полный перебор



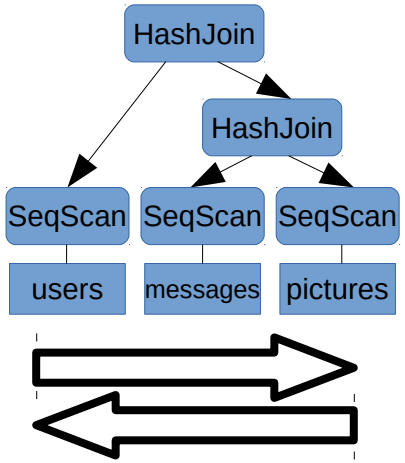
Cost = 439429



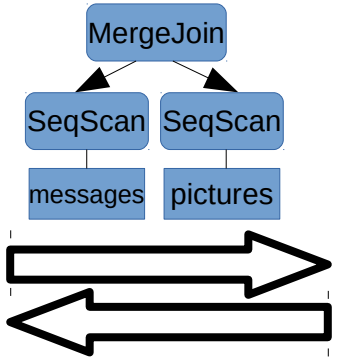
Cost = 304528

Оценка стоимости плана

Метод оптимизации



Cost = 439429



Cost = 304528

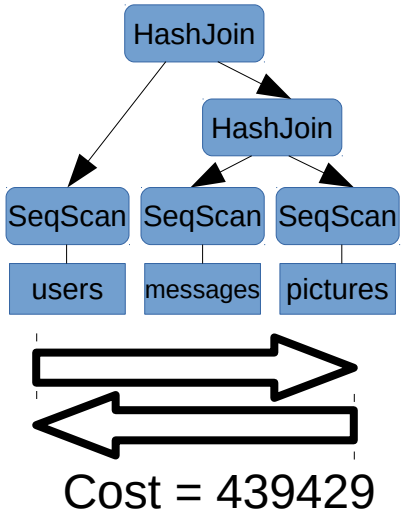
Оценка стоимости плана

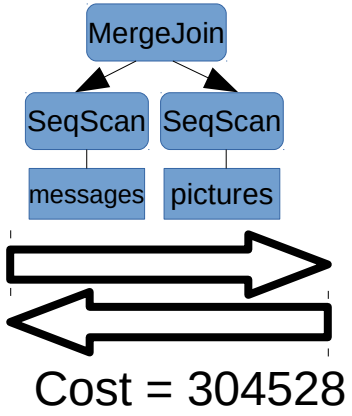
Метод оптимизации

Динамическое программирование

или

Генетический алгоритм





Оценка стоимости плана

Динамическое программирование

по подмножествам

- System R
- Временная сложность: 3^n
- Объем потребляемой памяти: 2^n
- Всегда находит самый дешевый план

Генетический алгоритм

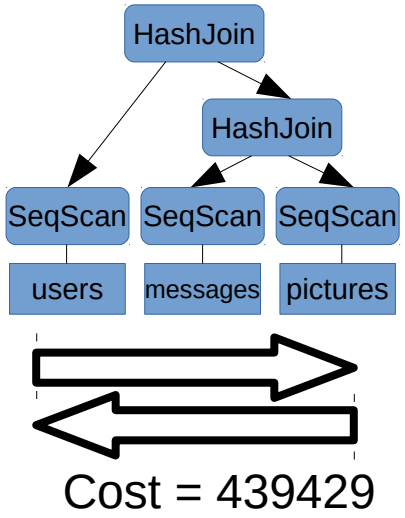
- PostgreSQL
- Общий, гибко настраиваемый метод
- Может быть остановлен на любой итерации
- Нет гарантий

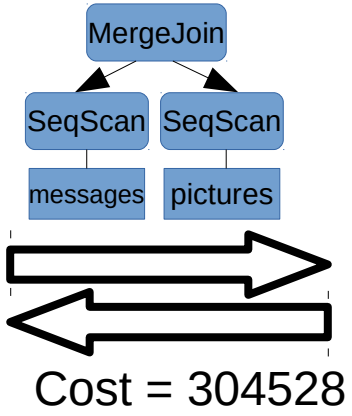
Метод оптимизации

Динамическое программирование

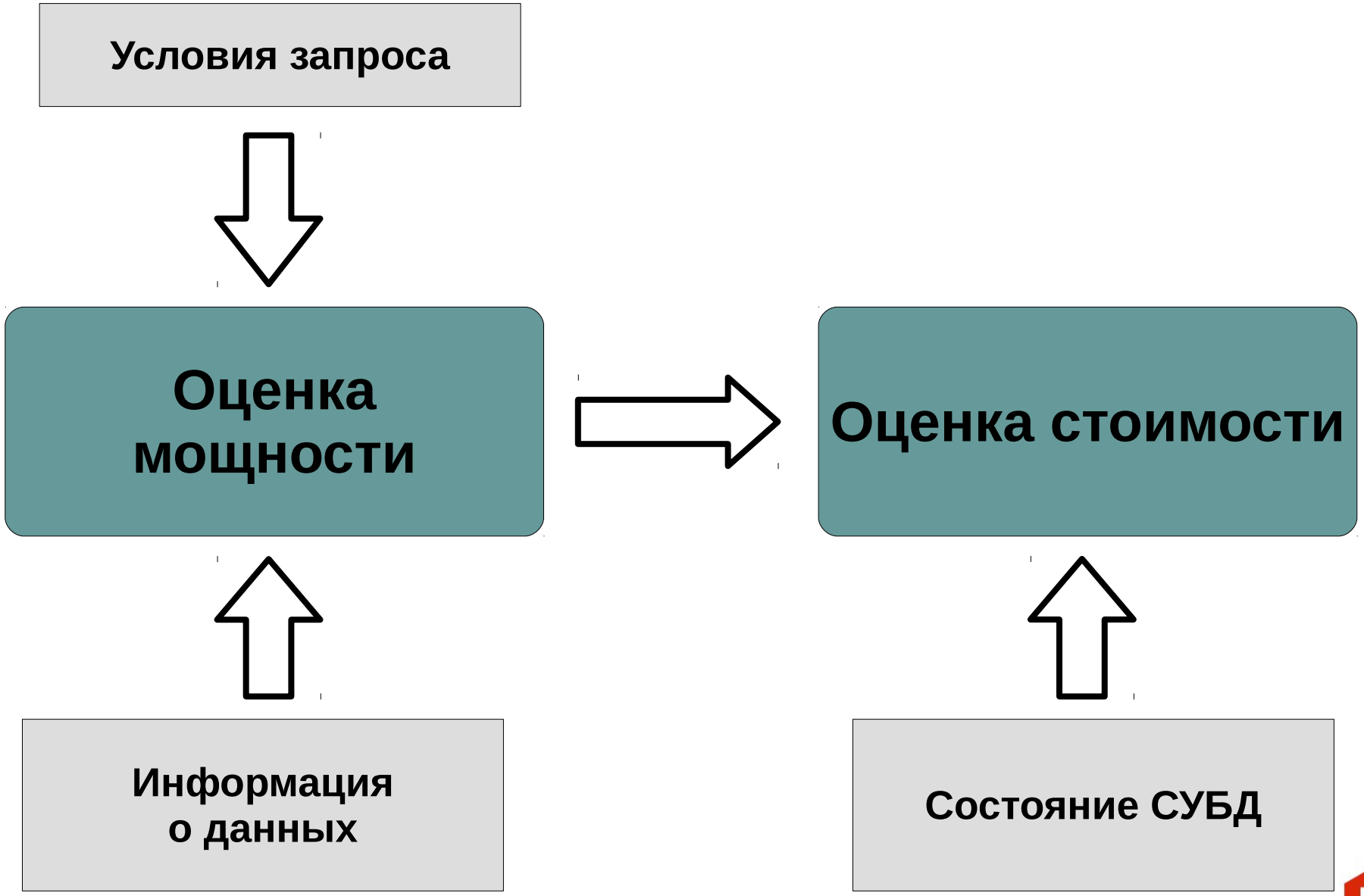
или

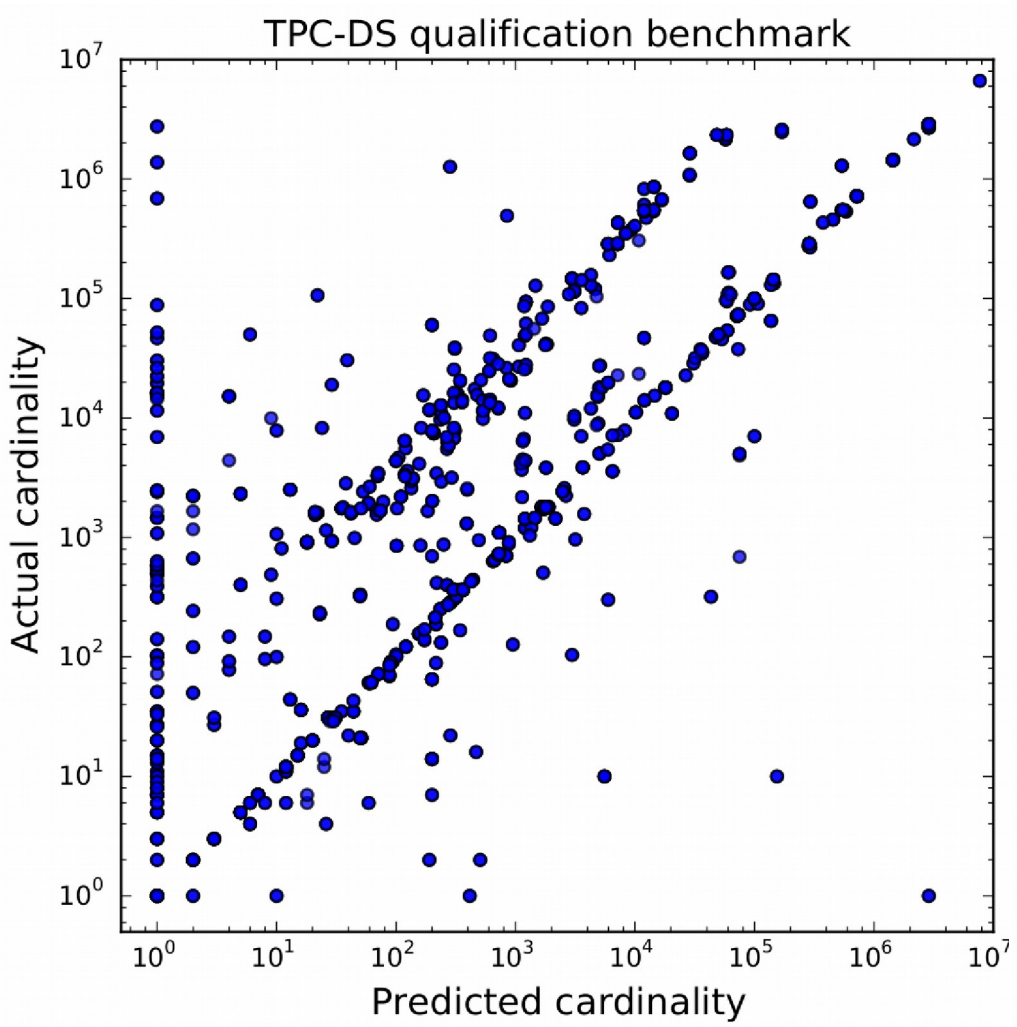
Генетический алгоритм

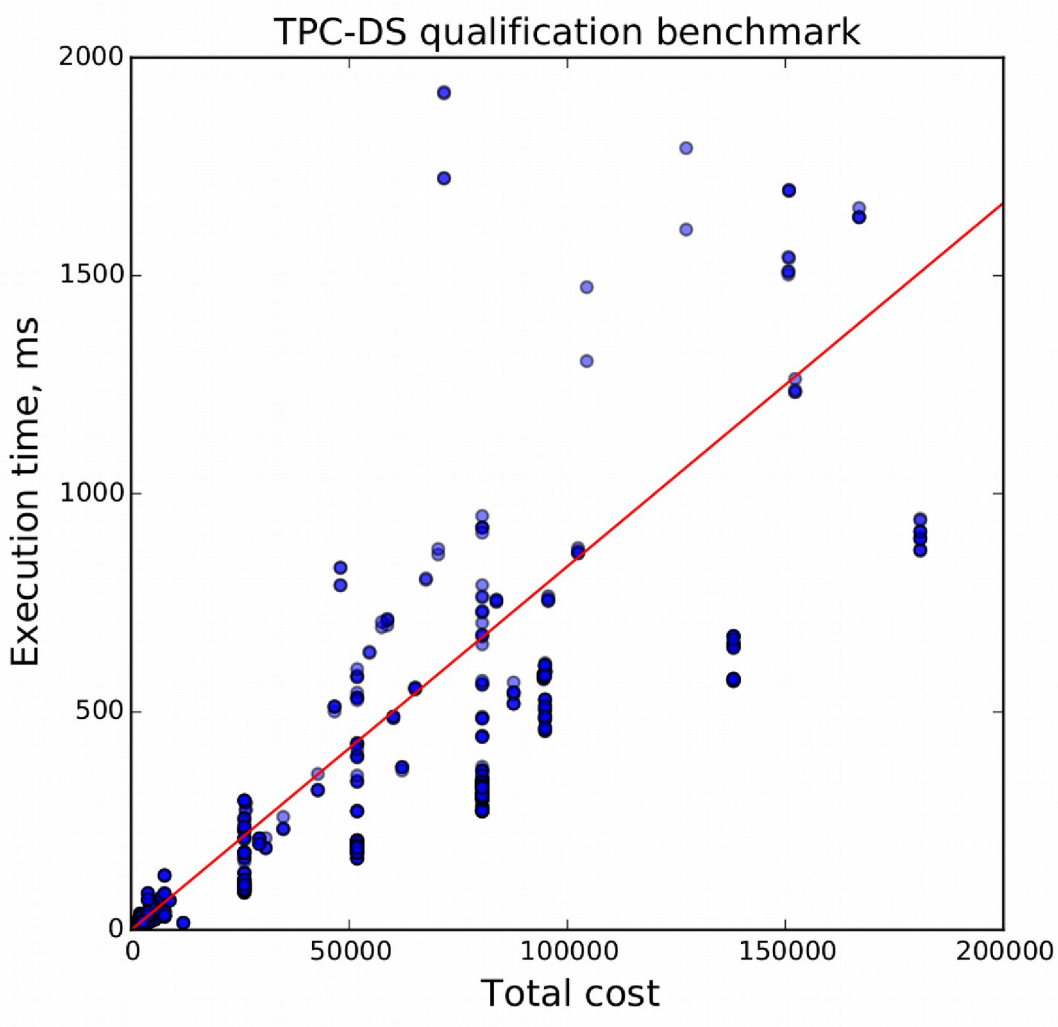




Оценка стоимости плана

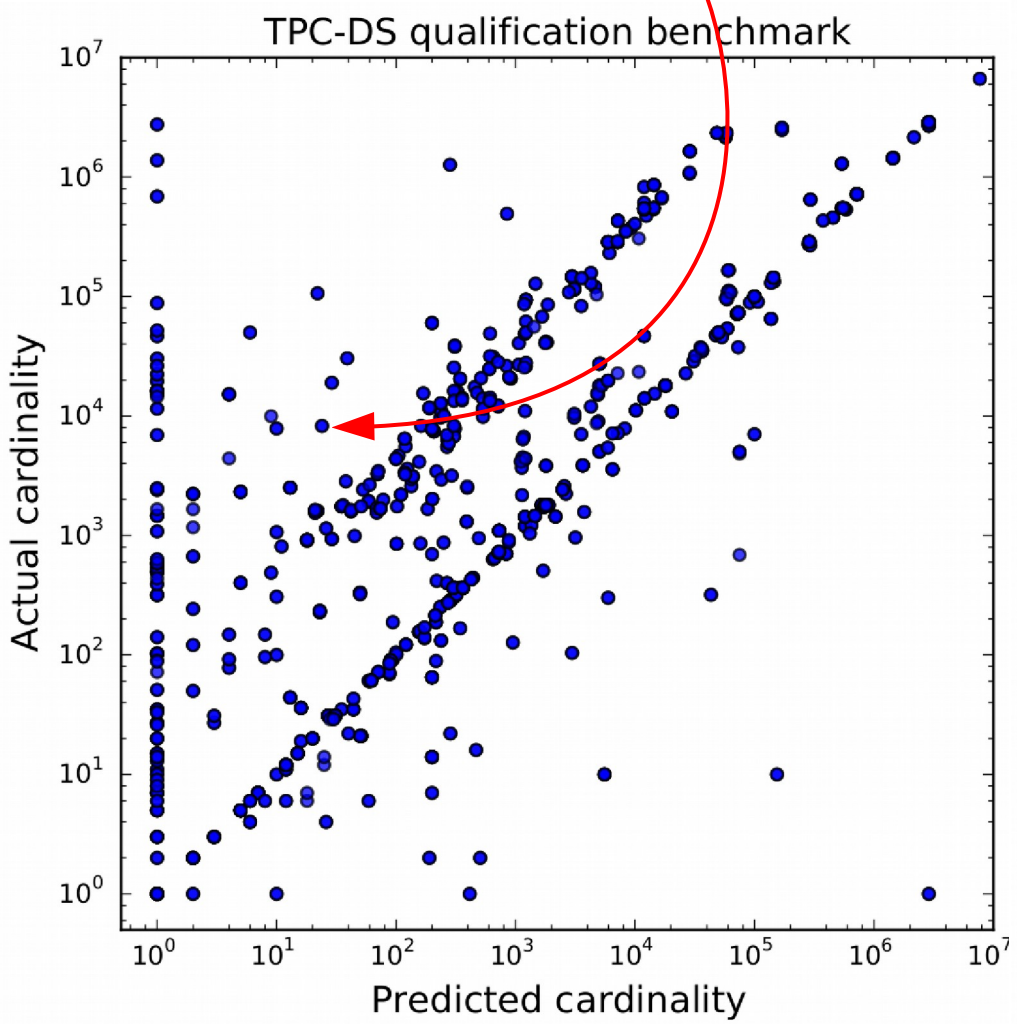




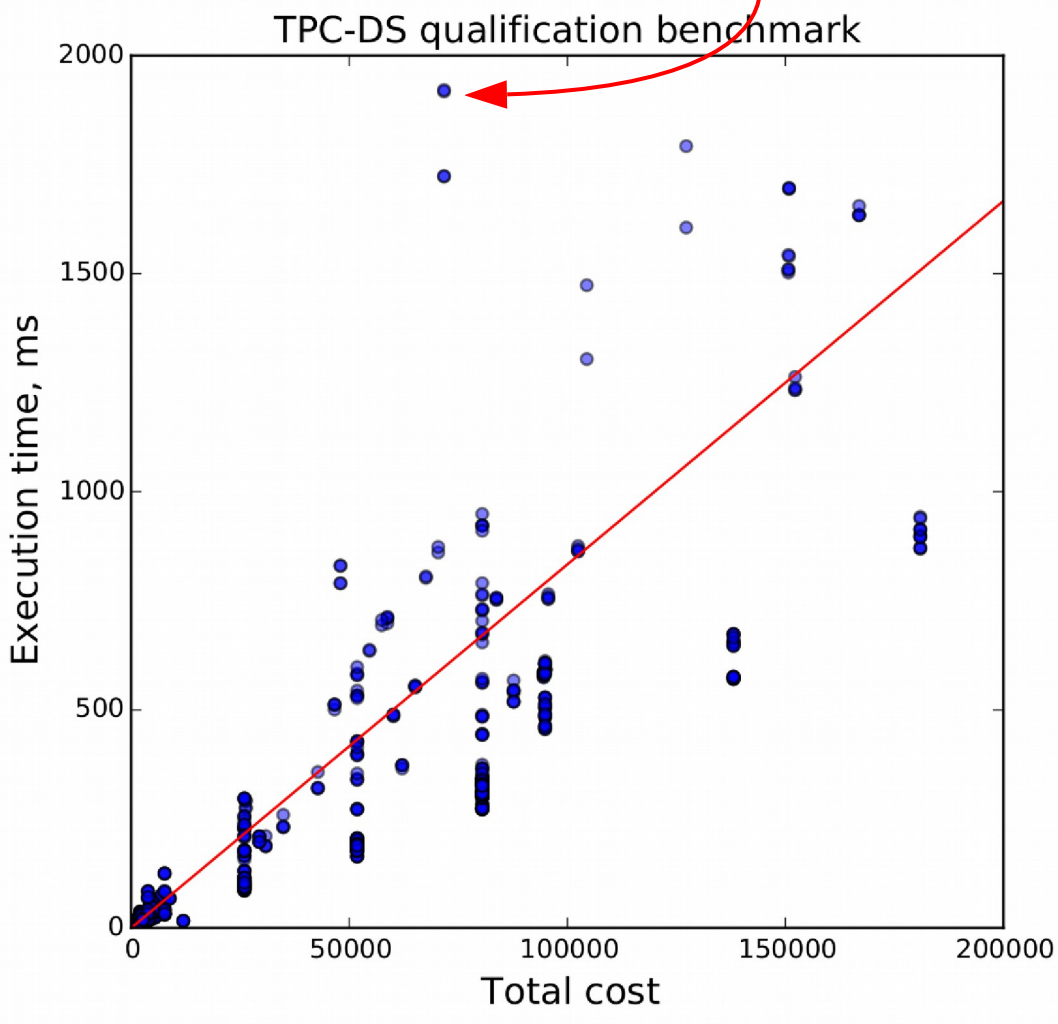


Dataset:
The TPC Benchmark™DS (TPC-DS)
http://www.tpc.org/tpcds/

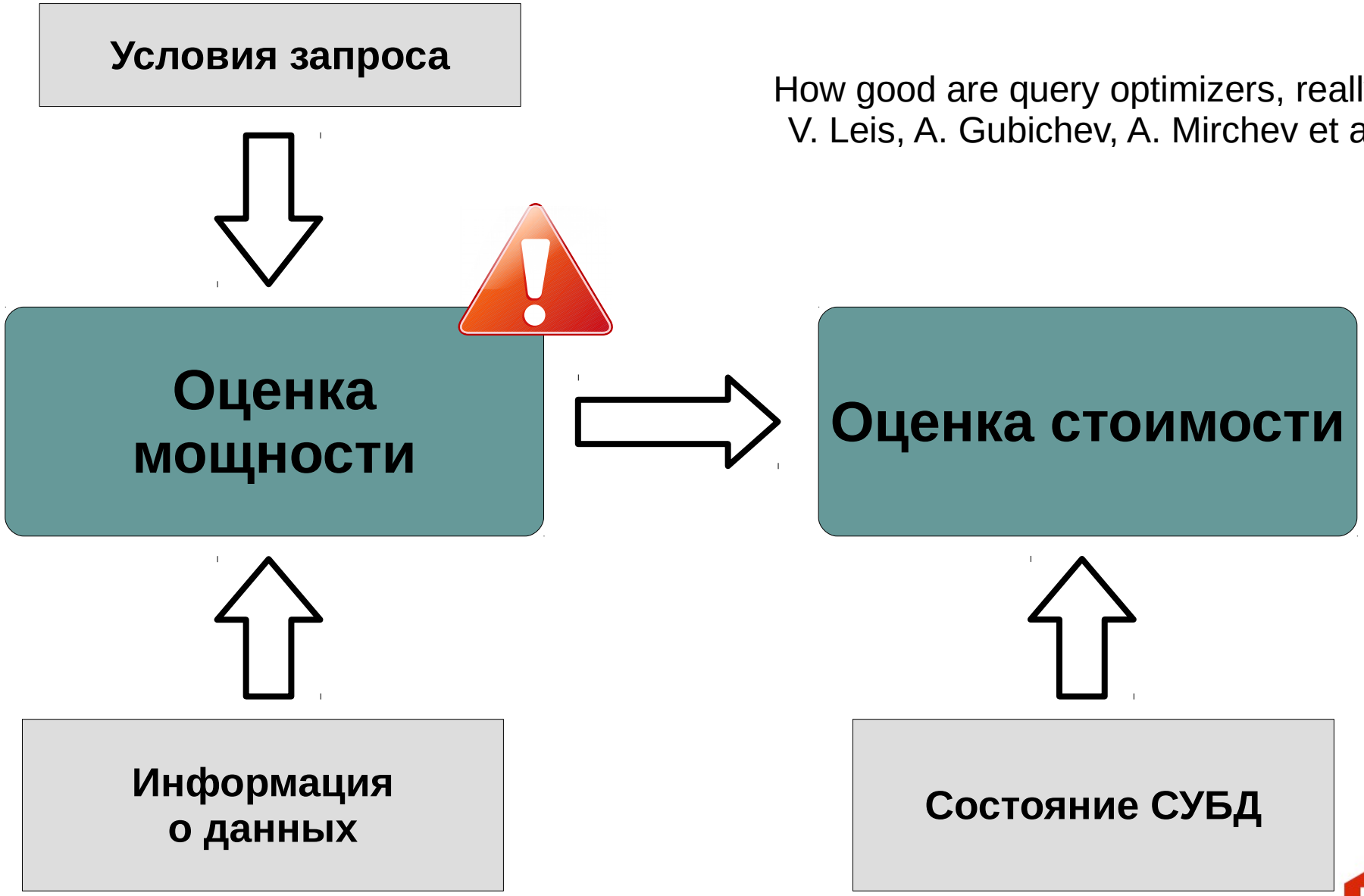
Ошибка в 300 раз



Ошибка в 4 раза

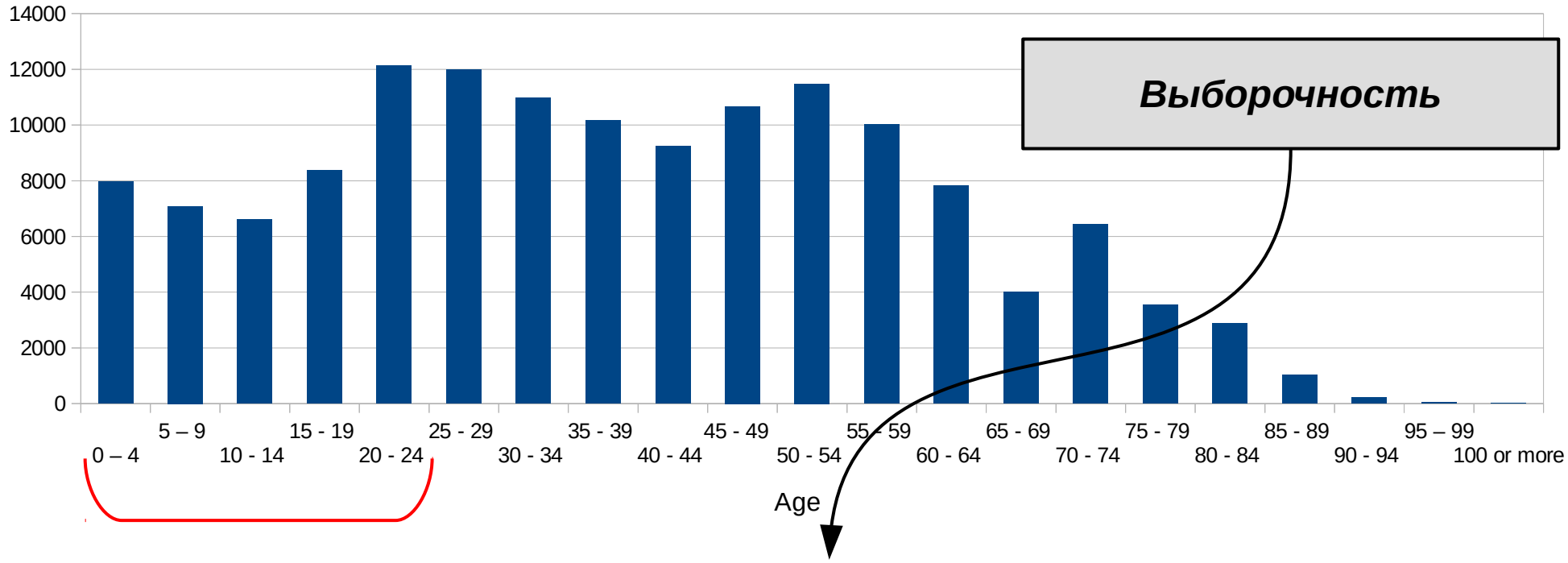


Dataset:
The TPC Benchmark™DS (TPC-DS)
http://www.tpc.org/tpcds/



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

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



$$Selectivity \simeq 0.3$$

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

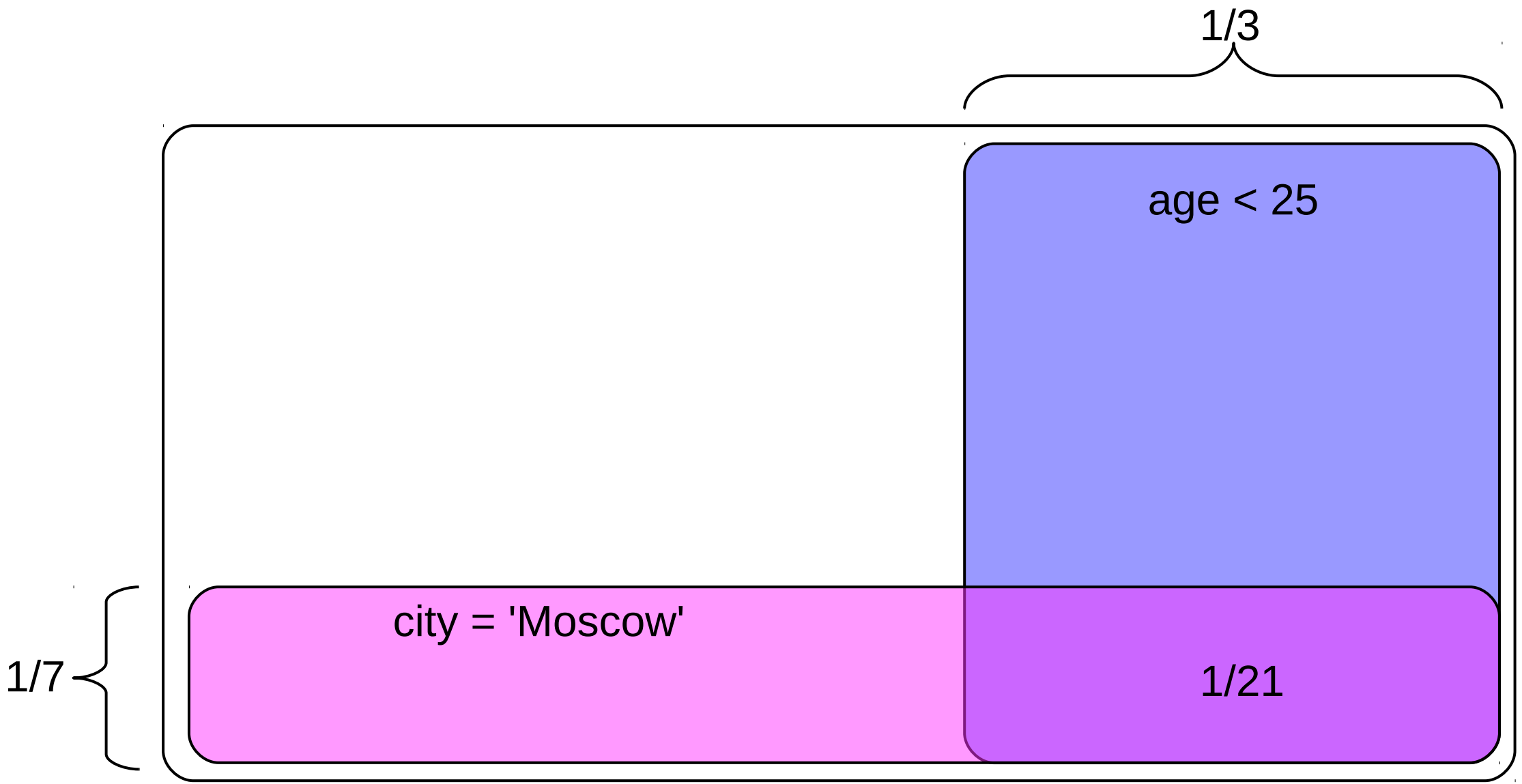
```
SELECT * FROM users
WHERE age < 25 AND city = 'Moscow';
```

Известны только выборочности отдельных условий

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

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

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



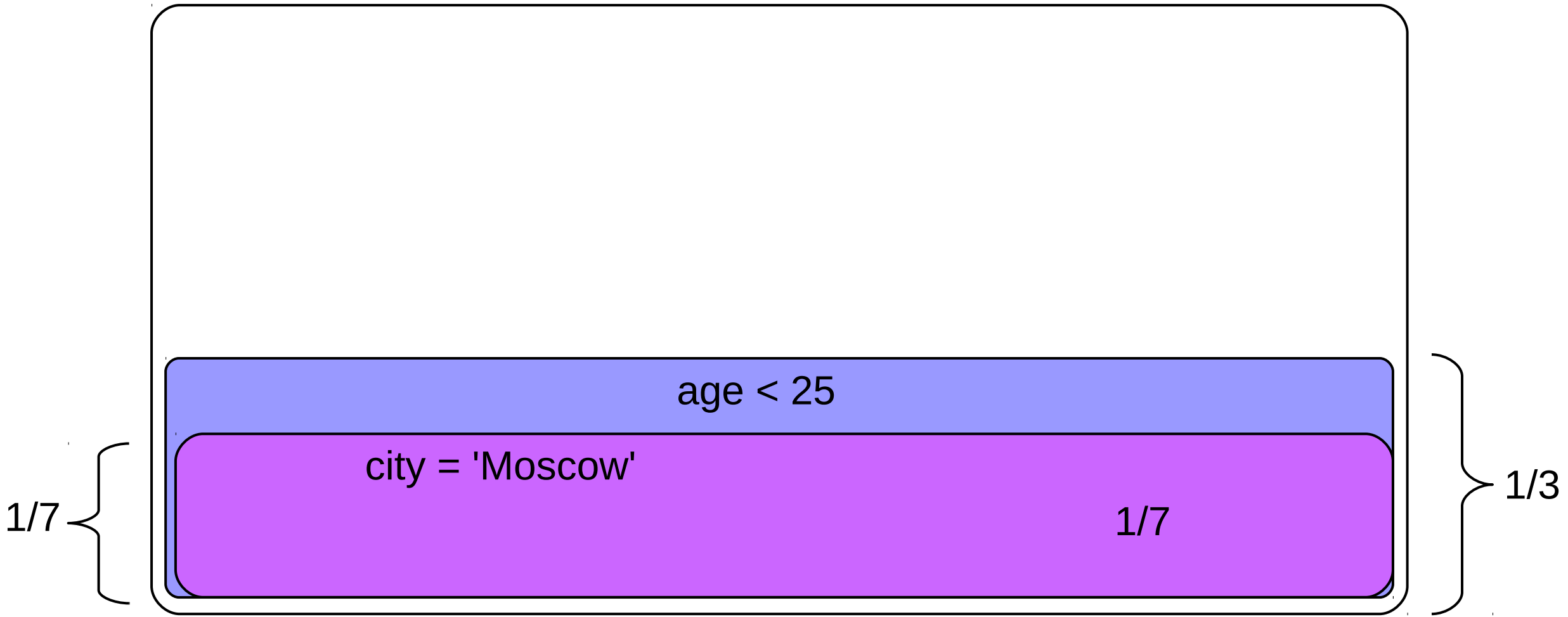
```
SELECT * FROM users
WHERE age < 25 AND city = 'Moscow';
```

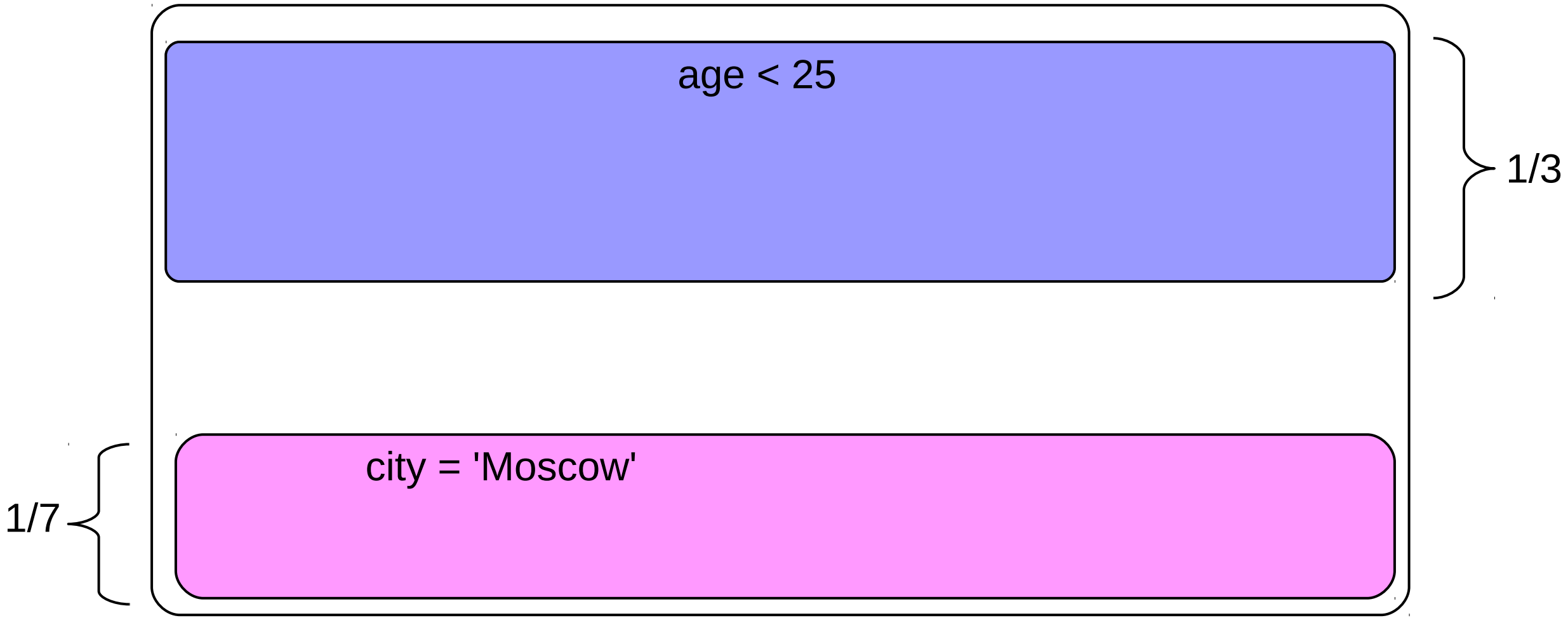
Известны только выборочности отдельных условий

Условия полагаются независимыми:

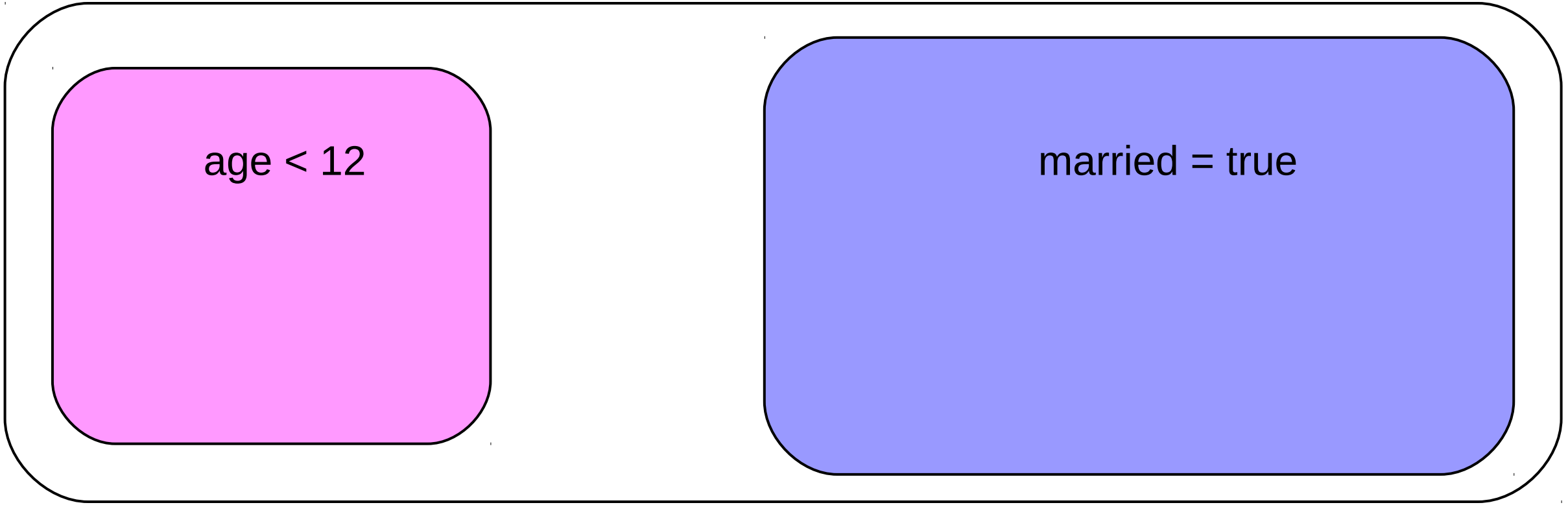
$$Selectivity_{age,city} = Selectivity_{age} \cdot Selectivity_{city}$$

За исключением $Selectivity_{25<age\ \mathrm{AND}\ age<57} = Selectivity_{25<age<57}$

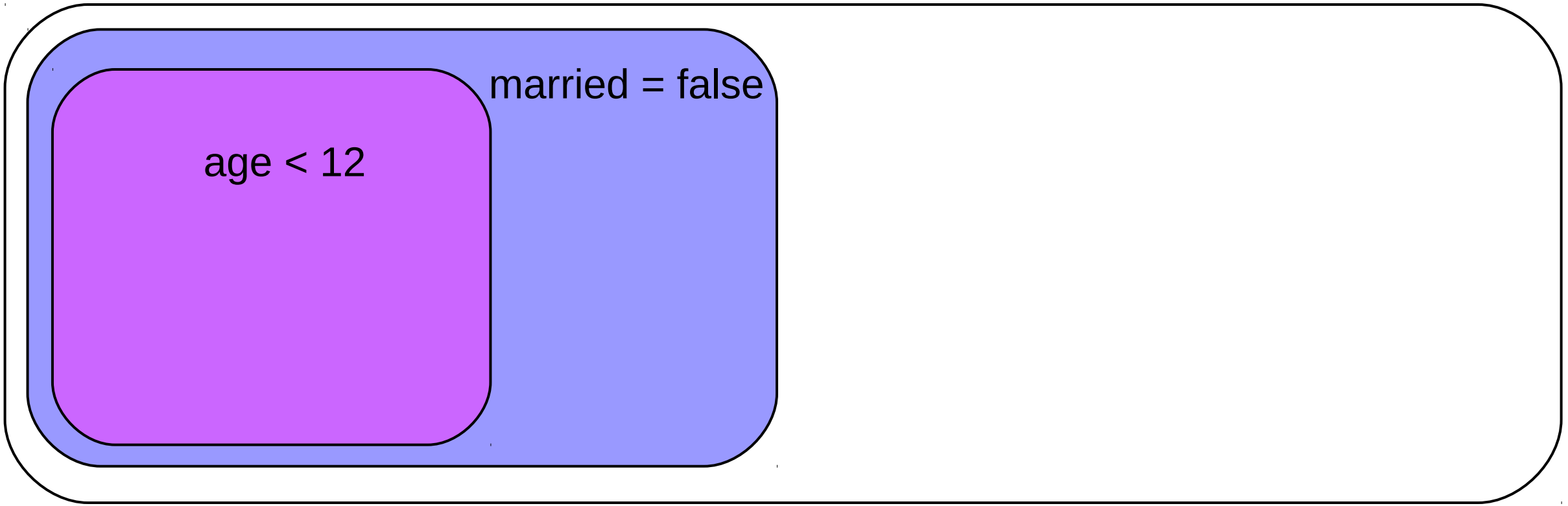




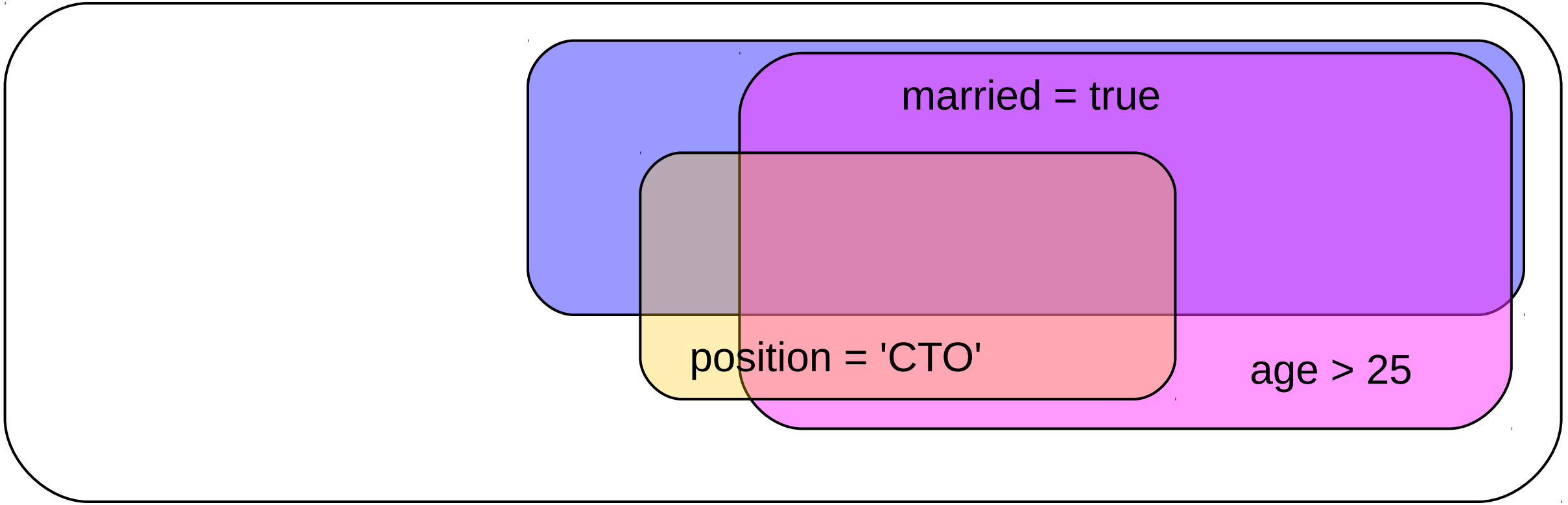
```
SELECT * FROM users
WHERE age < 12 AND married = true;
```



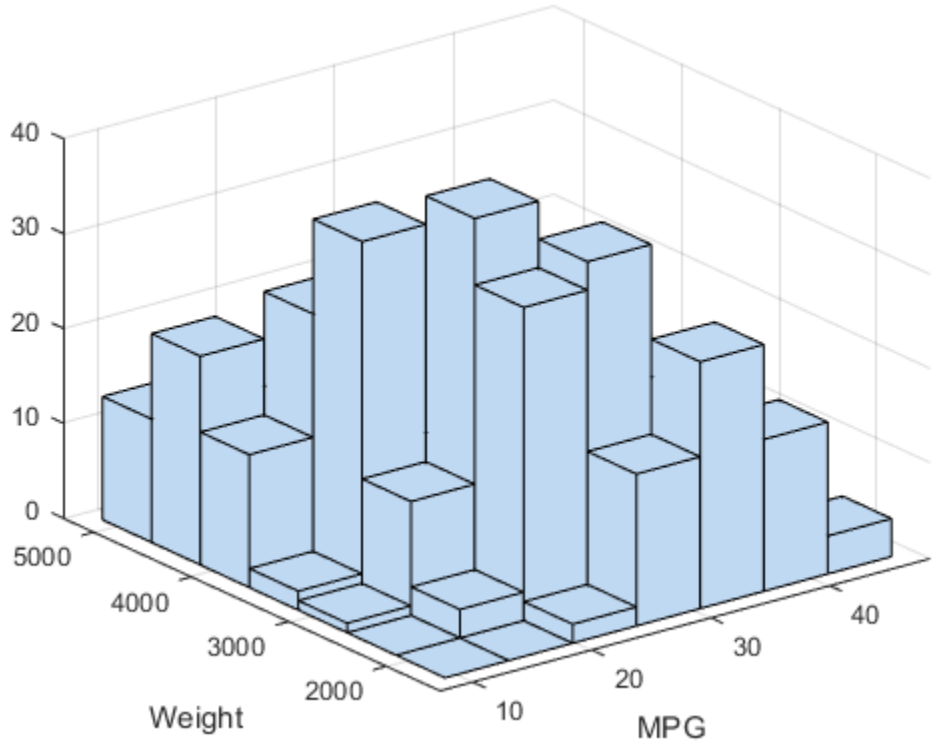
```
SELECT * FROM users
WHERE age < 12 AND married = false;
```



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



Многомерные гистограммы



Что такое адаптивная оптимизация?

SQL запрос

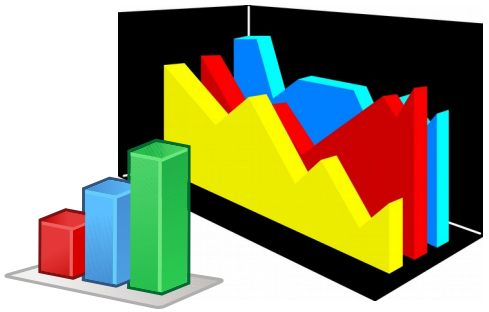
Оптимизация запроса

Выполнение запроса

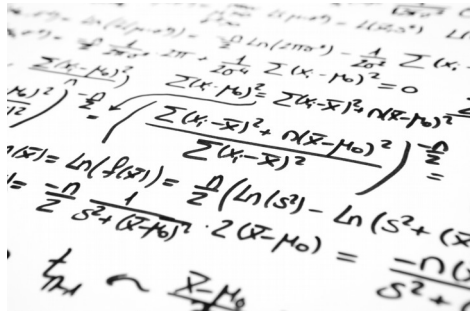
Результат

Обратная связь

Статическая информация



Гистограммы



Модели стоимости

Статистика выполнения

Что такое адаптивная оптимизация?

SQL запрос

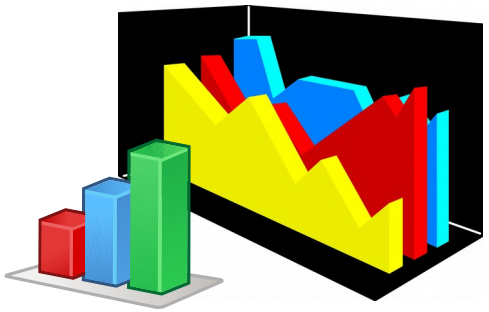
Оптимизация запроса

Выполнение запроса

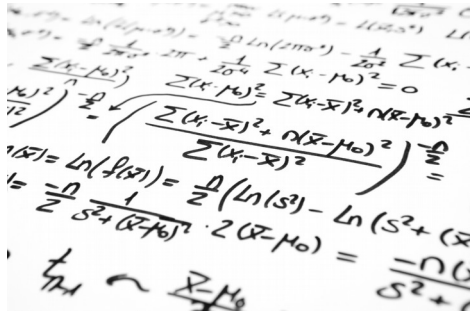
Результат

Обратная связь

Статическая информация скорректированная

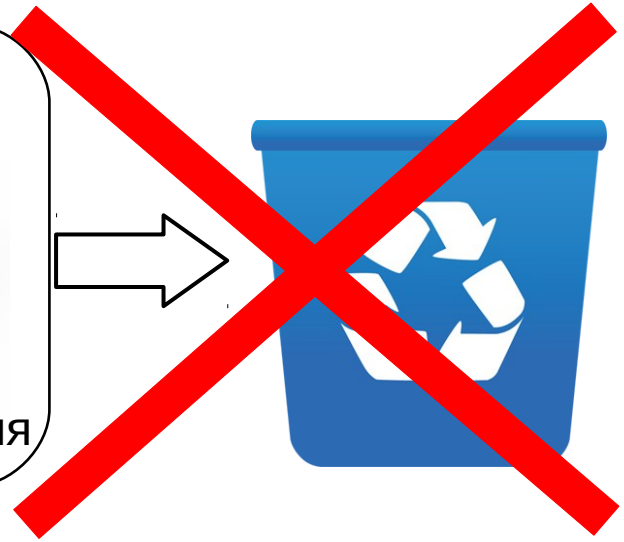


Гистограммы

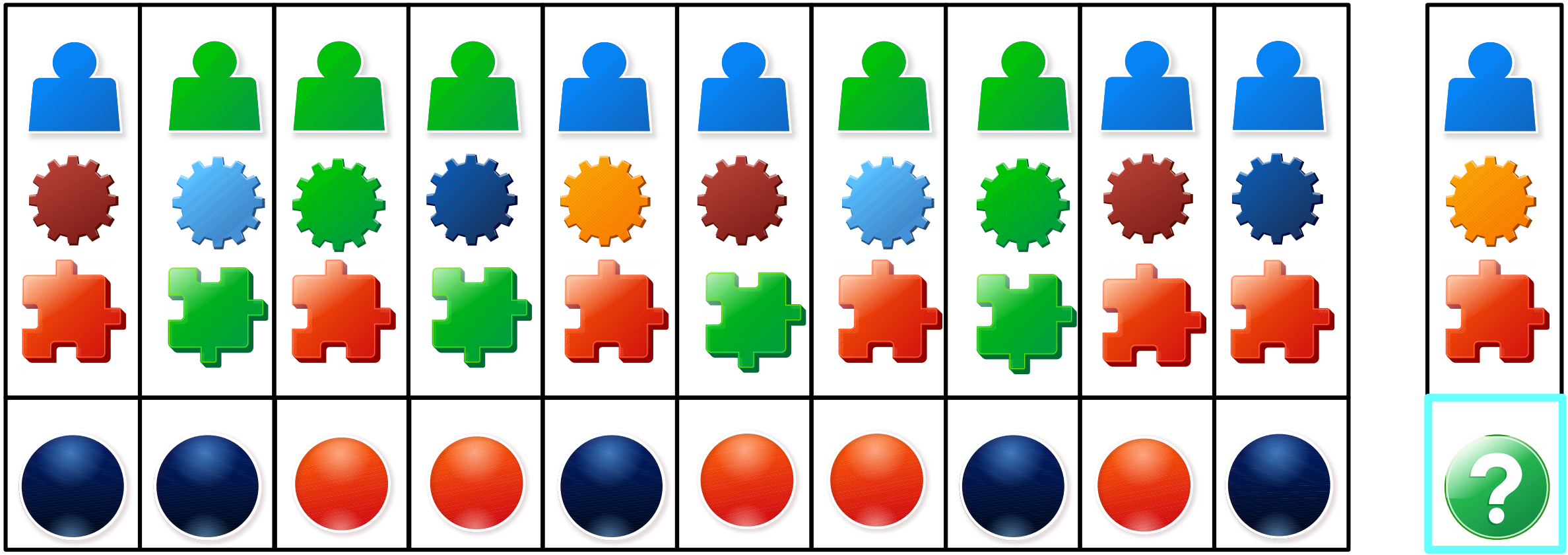


Модели стоимости

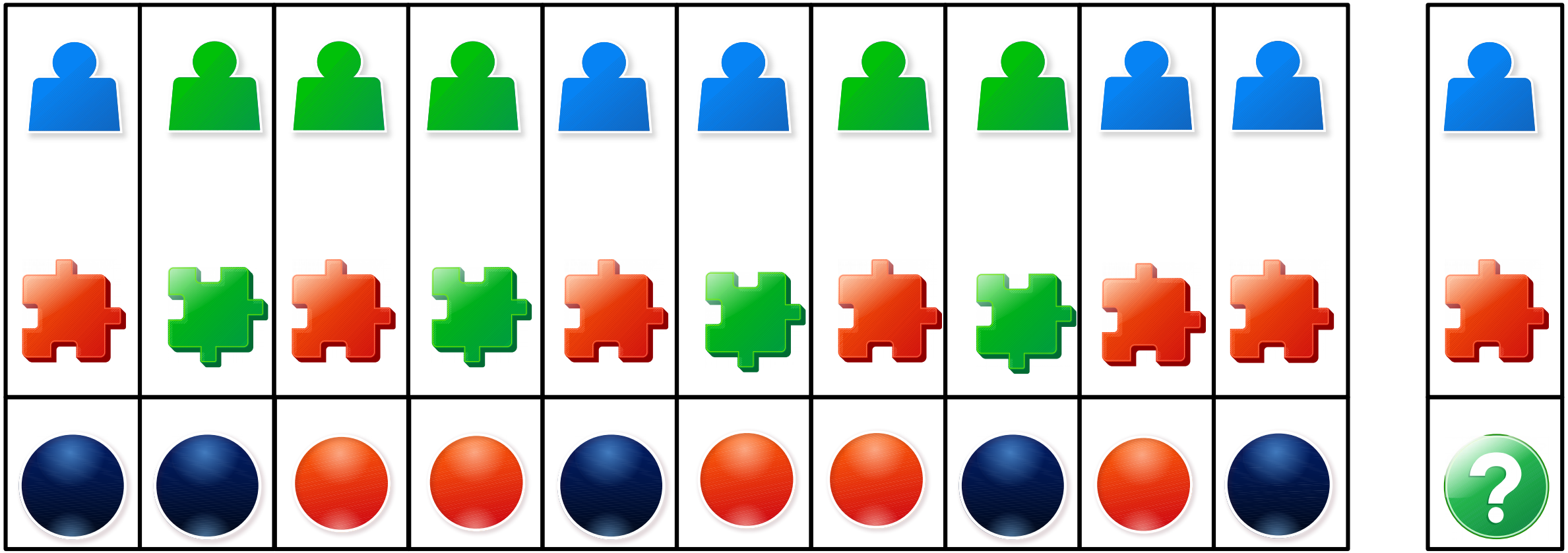
Статистика выполнения



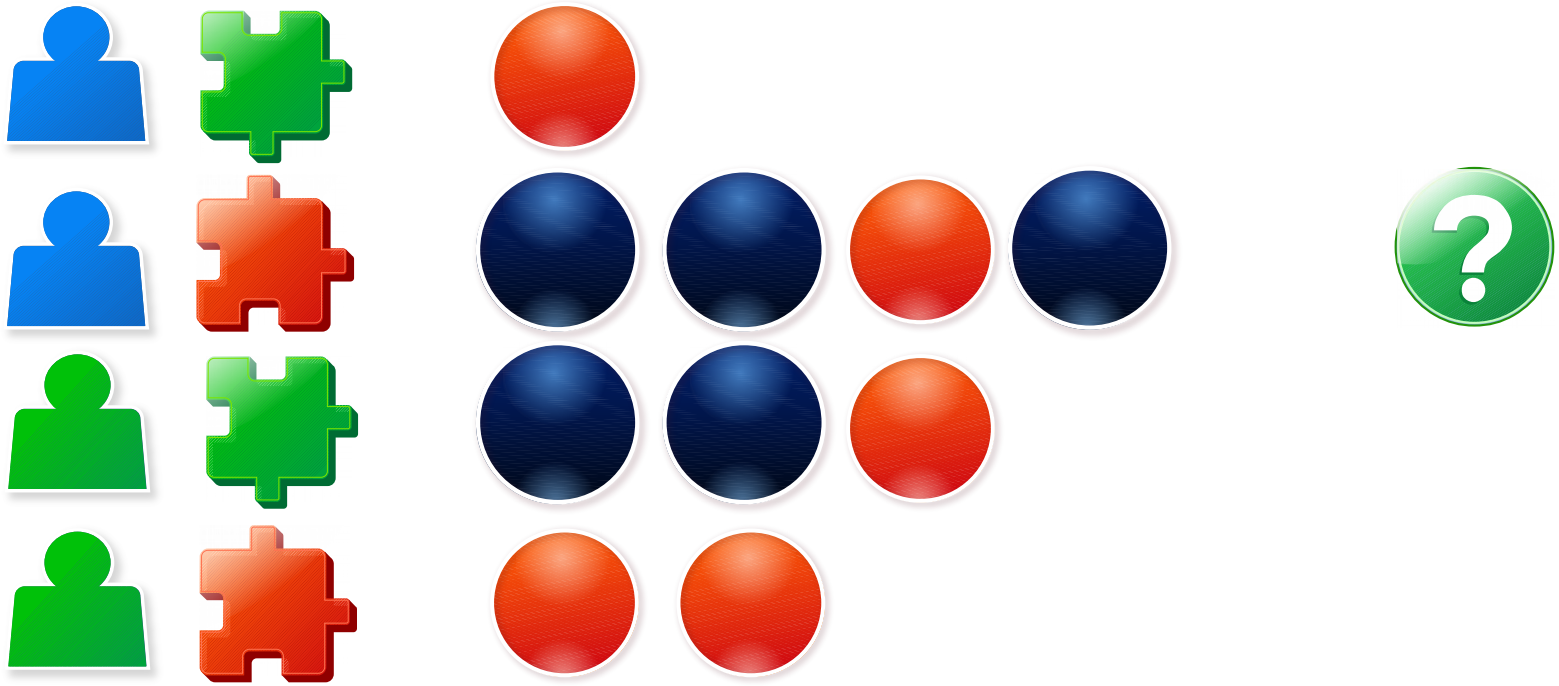
Машинное обучение

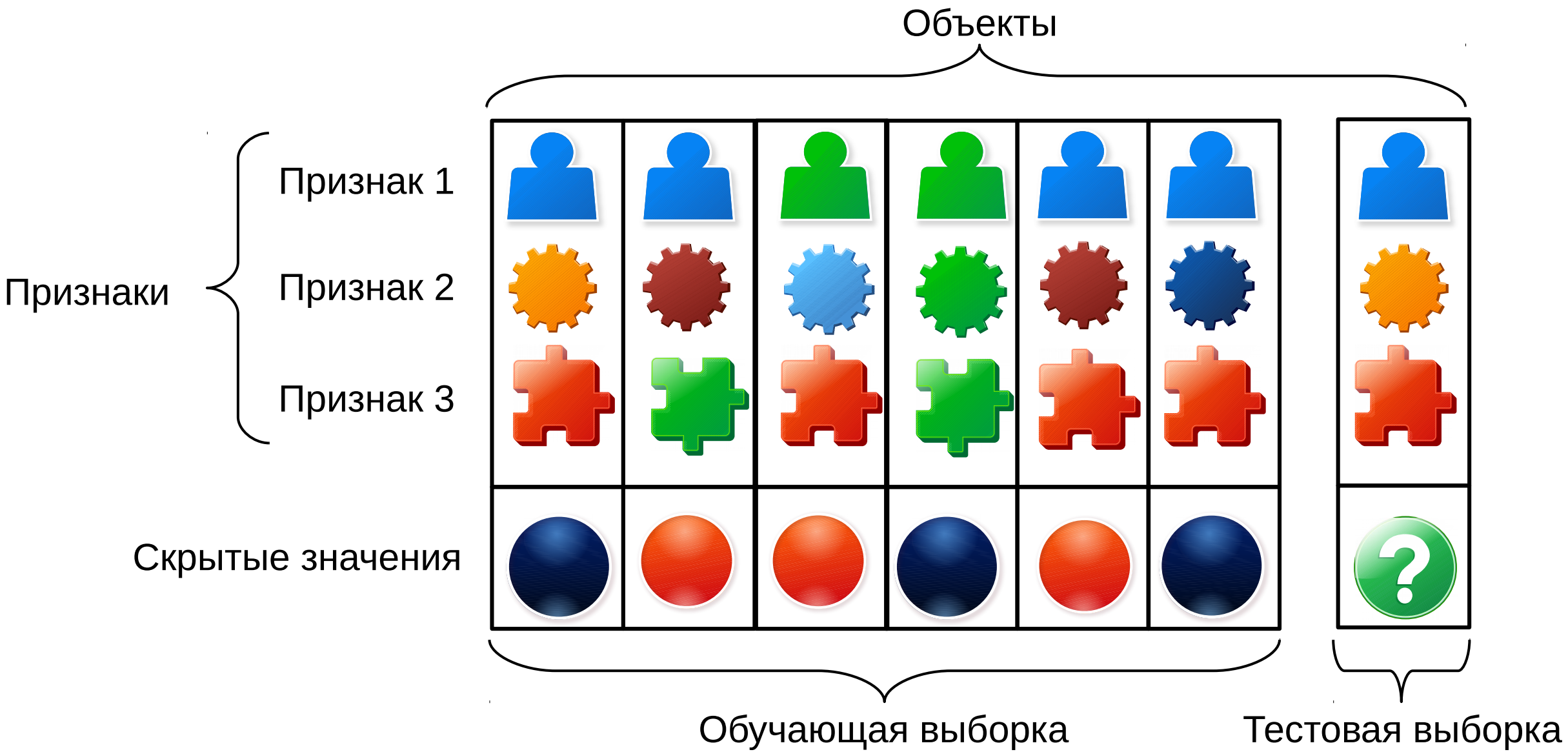


Машинное обучение

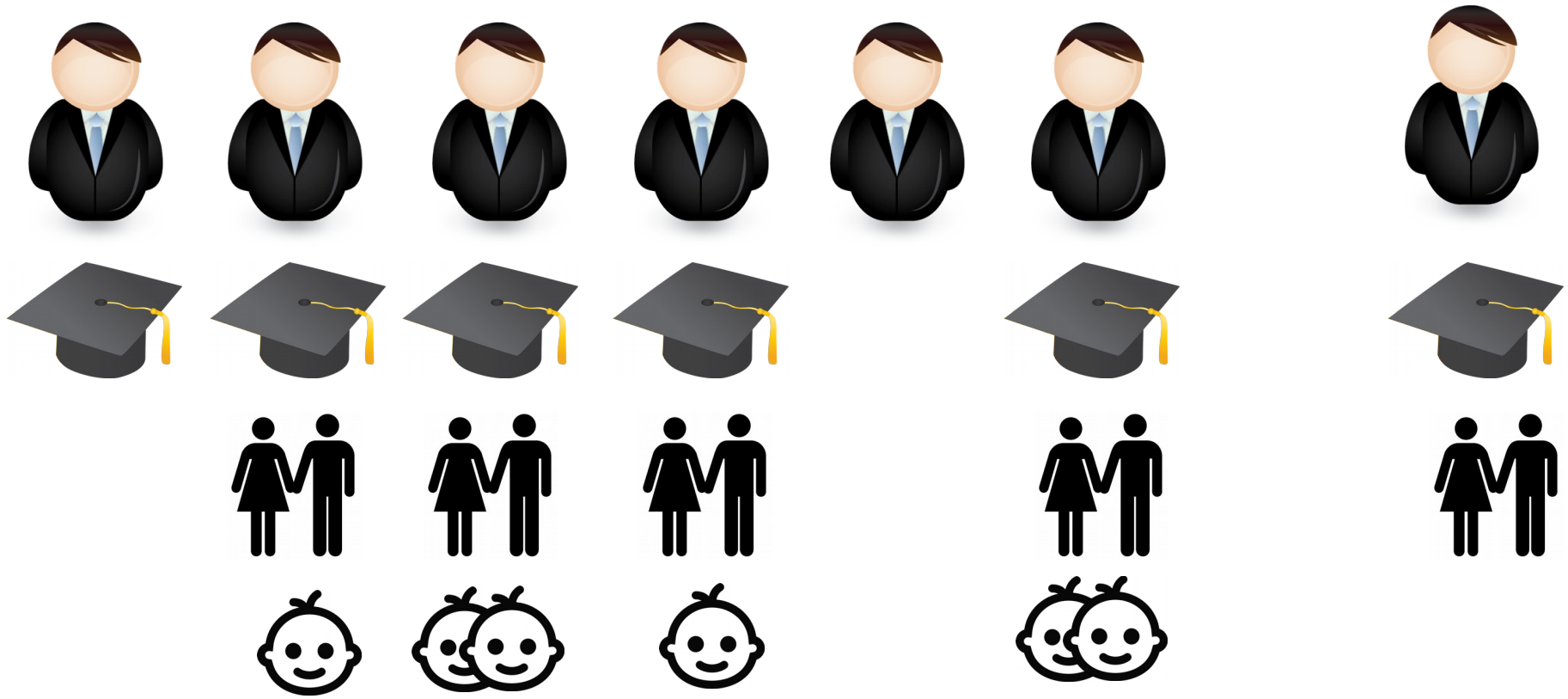


Машинное обучение



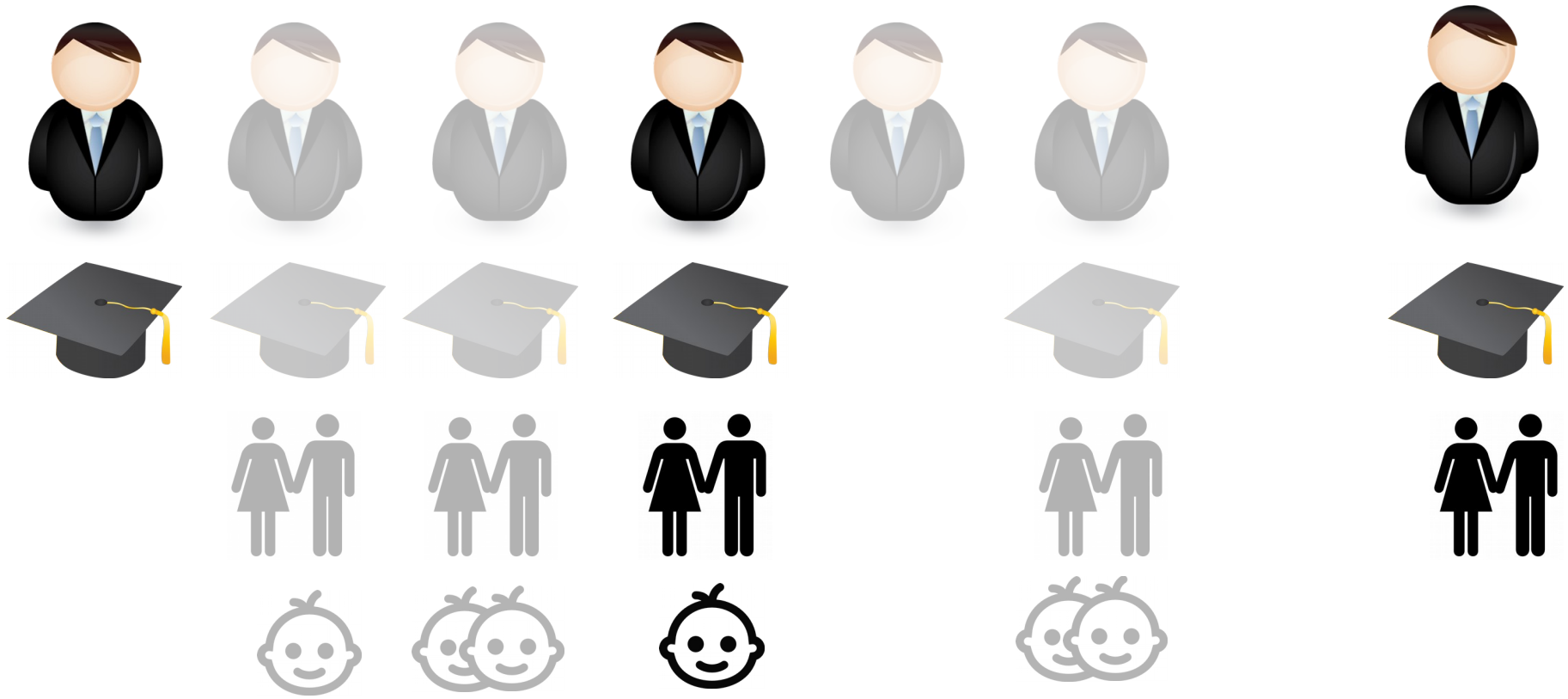


Метод k ближайших соседей



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

Метод k ближайших соседей

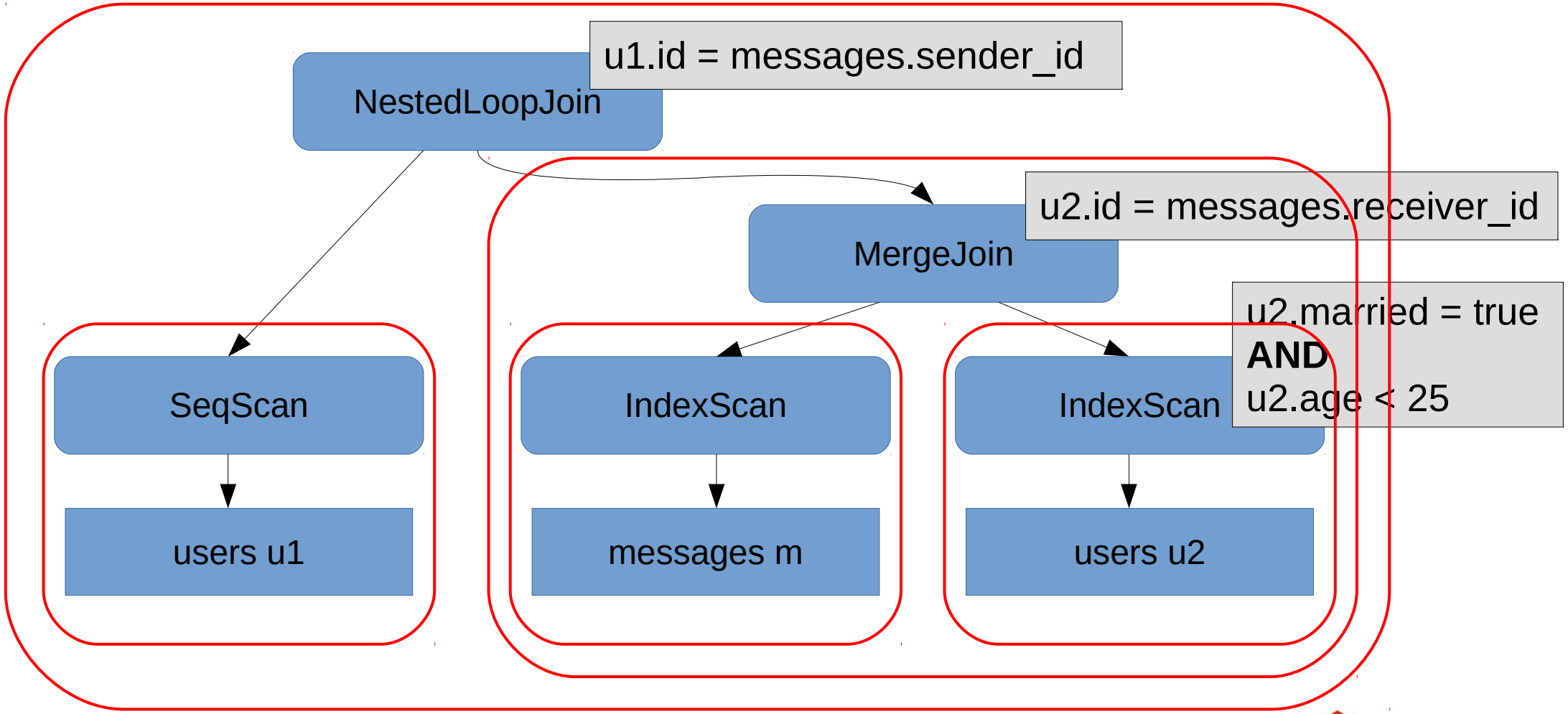


25	47	55	32	22	45	28
50	120	100	80	30	90	?

Градиентный метод k ближайших соседей

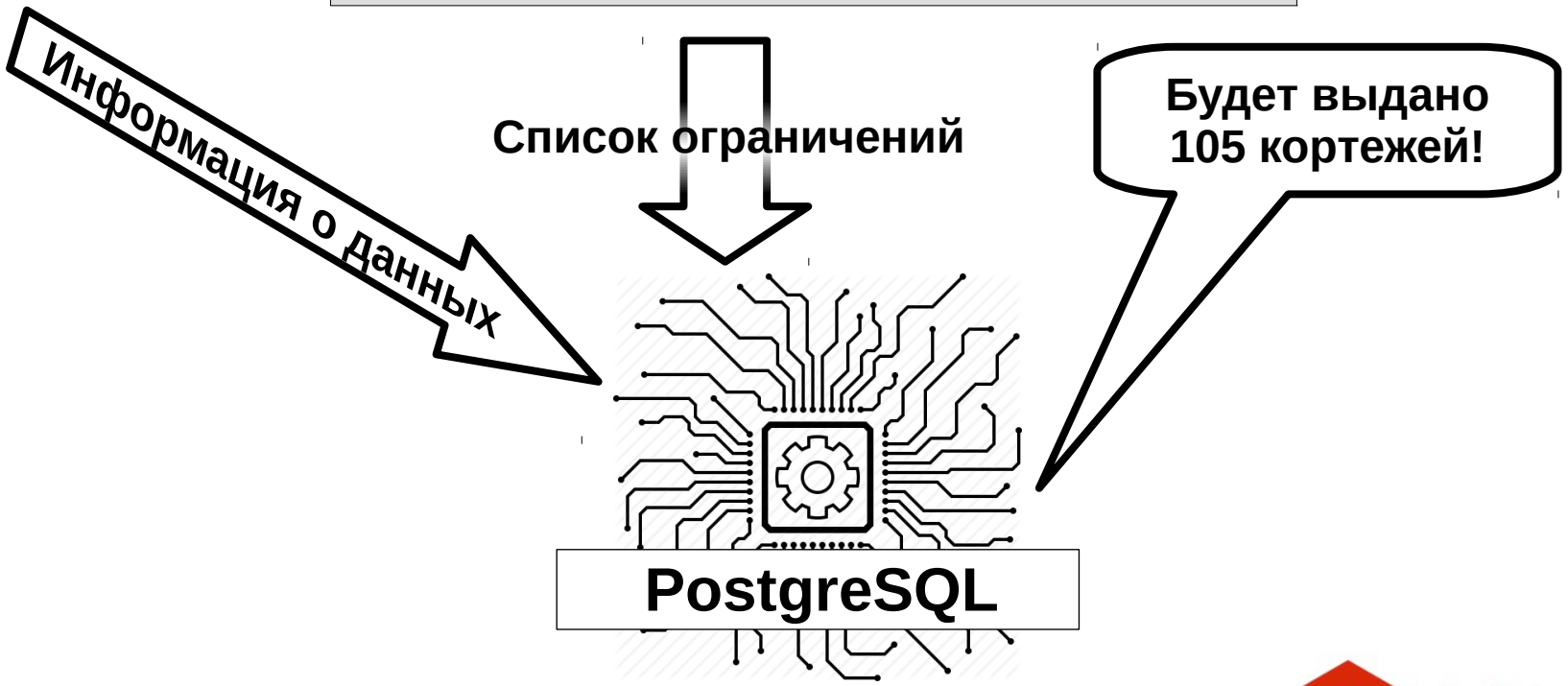
Как применить машинное обучение для адаптивной оптимизации запросов?

Объект – вершина и её поддерево



Гистограммы

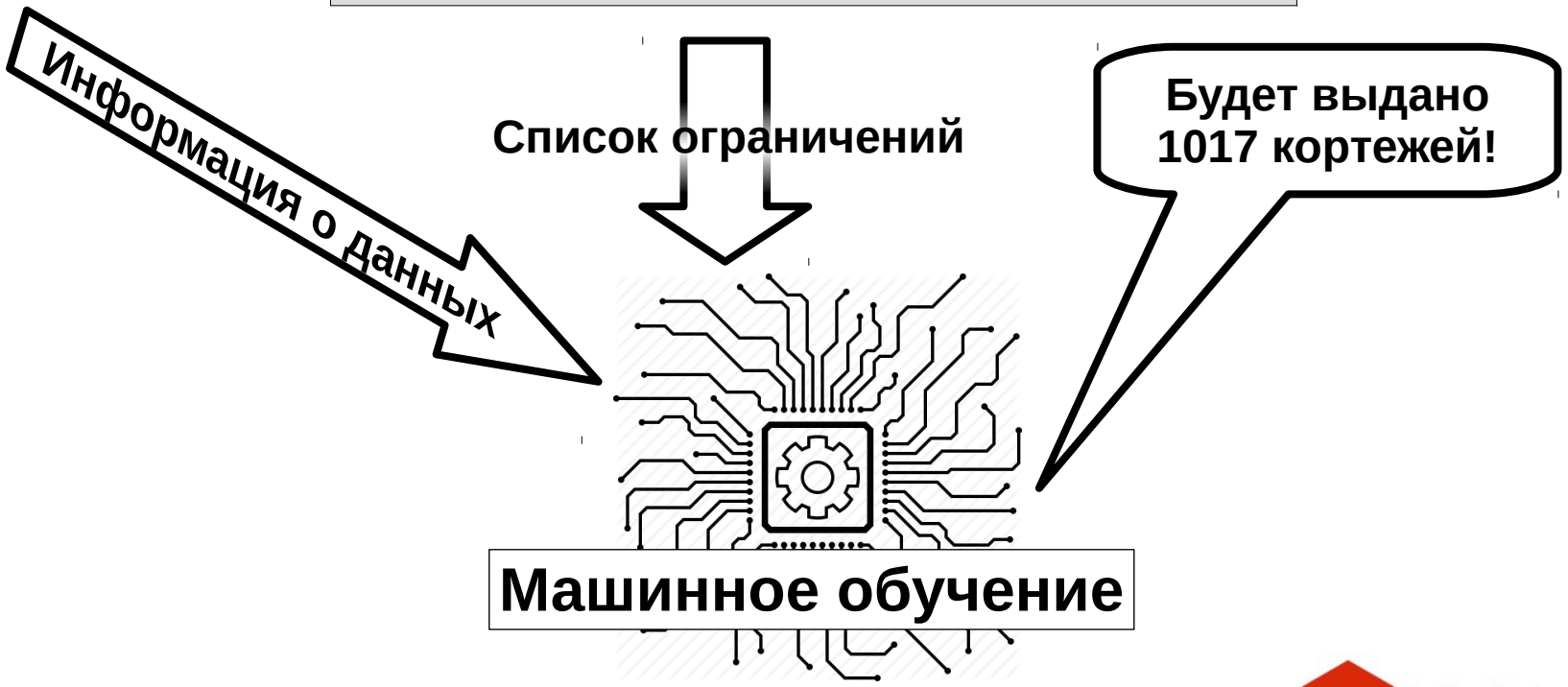
users.id = messages.receiver_id
AND
users.married = true
AND
users.age < 25



Выборочности условий

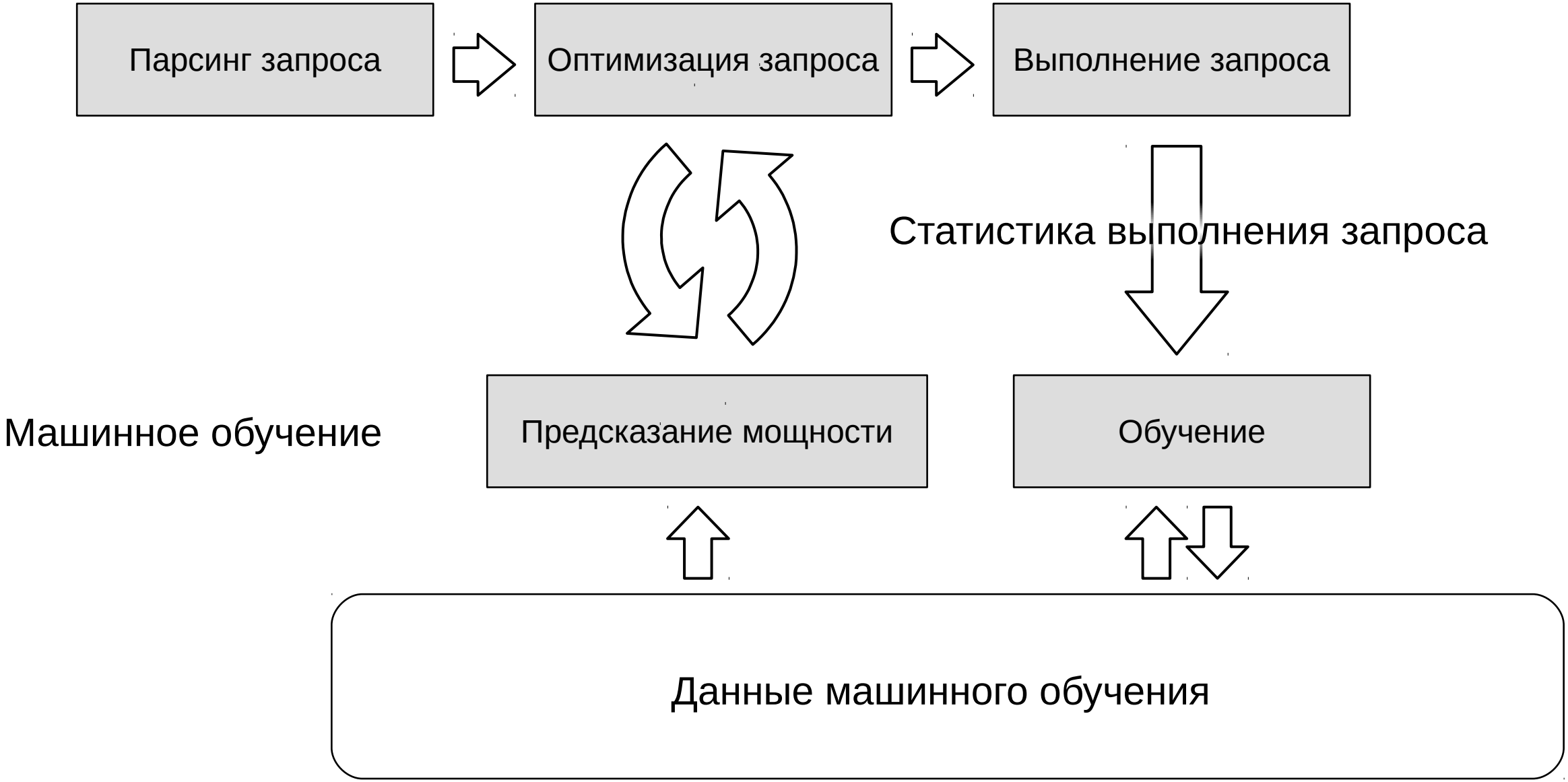
- 0.0001
- 0.73
- 0.23

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



Постановка задачи машинного обучения

		Объект – вершина плана
Признаки	users.id = messages.receiver_id	0.0001
	users.married = **const**	0.73
	users.age < **const**	0.23
Скрытое значение	Мощность вершины	?



- Сойдется ли?
- Как быстро сойдется?
- Куда сойдется?

- Сойдется ли?

Да, за конечное число шагов

- Как быстро сойдется?

- Куда сойдется?

- Сойдется ли?

Да, за конечное число шагов

- Как быстро сойдется?

Неизвестно (на практике за несколько шагов)

- Куда сойдется?

- Сойдется ли?

Да, за конечное число шагов

- Как быстро сойдется?

Неизвестно (на практике за несколько шагов)

- Куда сойдется?

Для выполненных планов предсказания верны

- Сойдется ли?

Да, за конечное число шагов

- Как быстро сойдется?

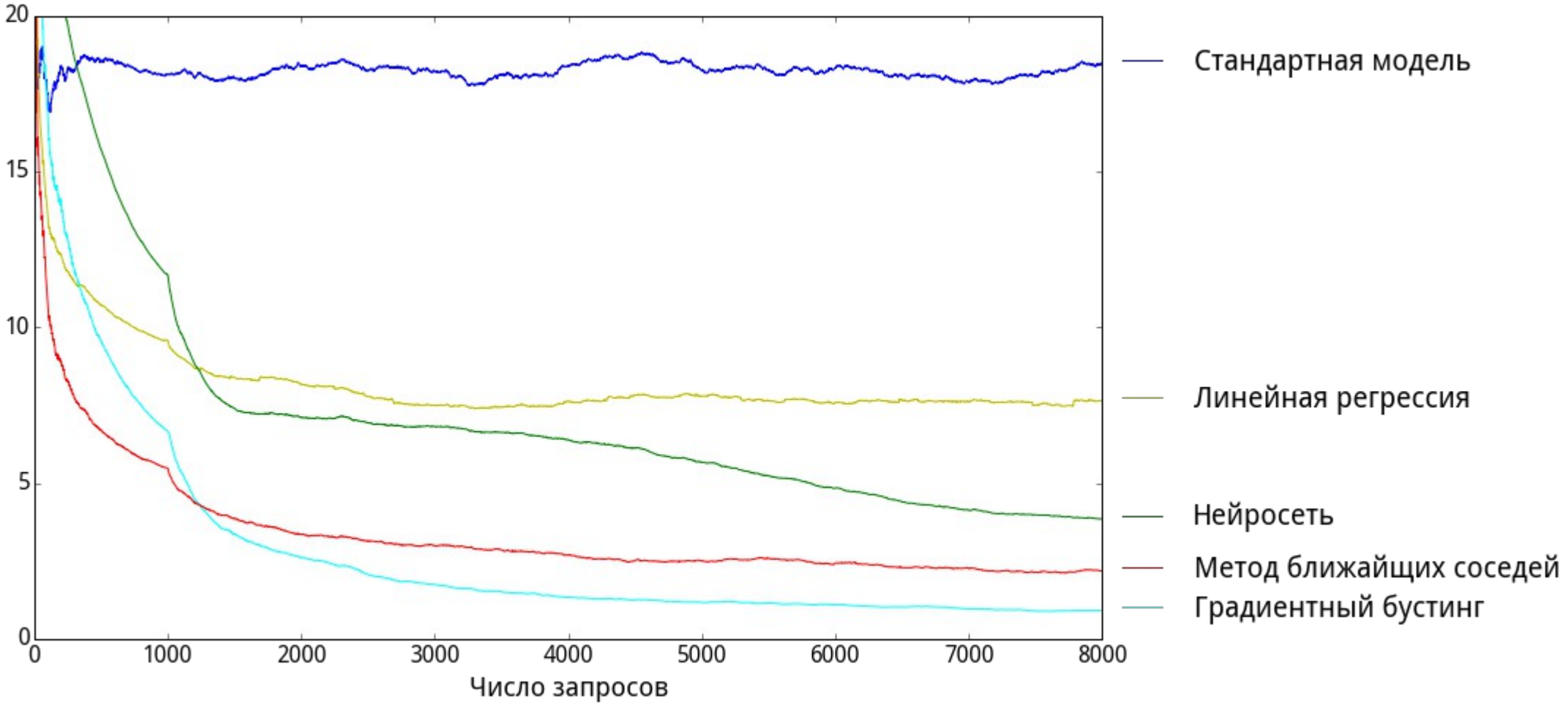
Неизвестно (на практике за несколько шагов)

- Куда сойдется?

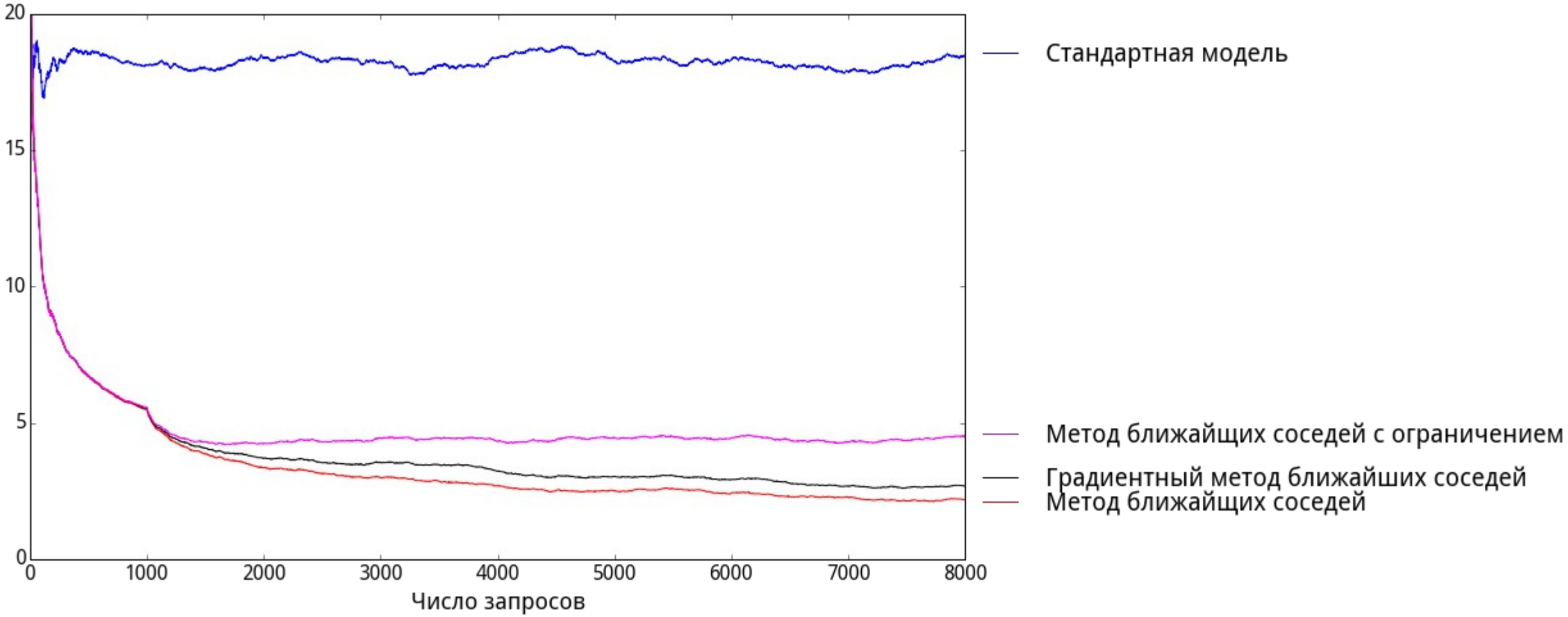
Для выполненных планов предсказания верны
С идеальной моделью стоимости
время выполнения запроса
гарантированно не увеличивается

Практические результаты

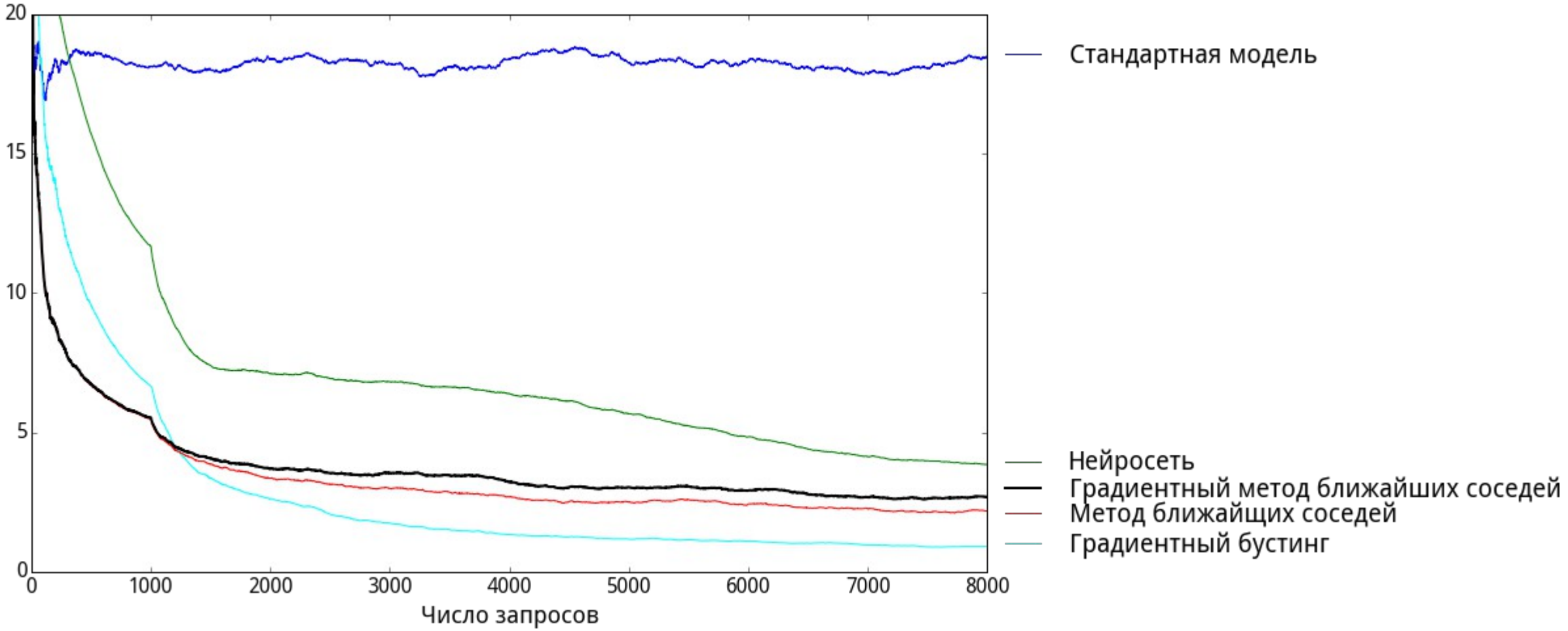
Ошибка предсказания



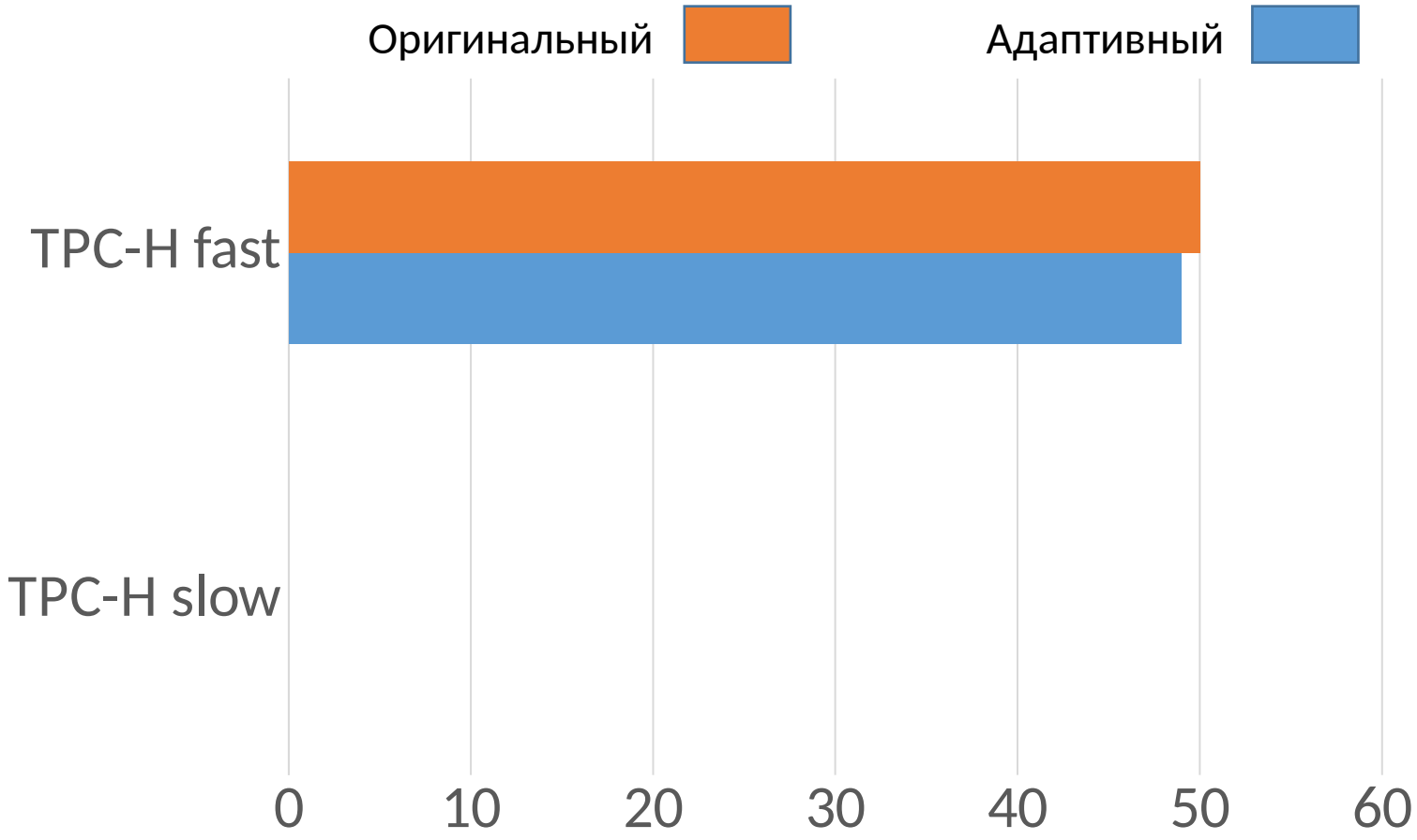
Ошибка предсказания



Ошибка предсказания

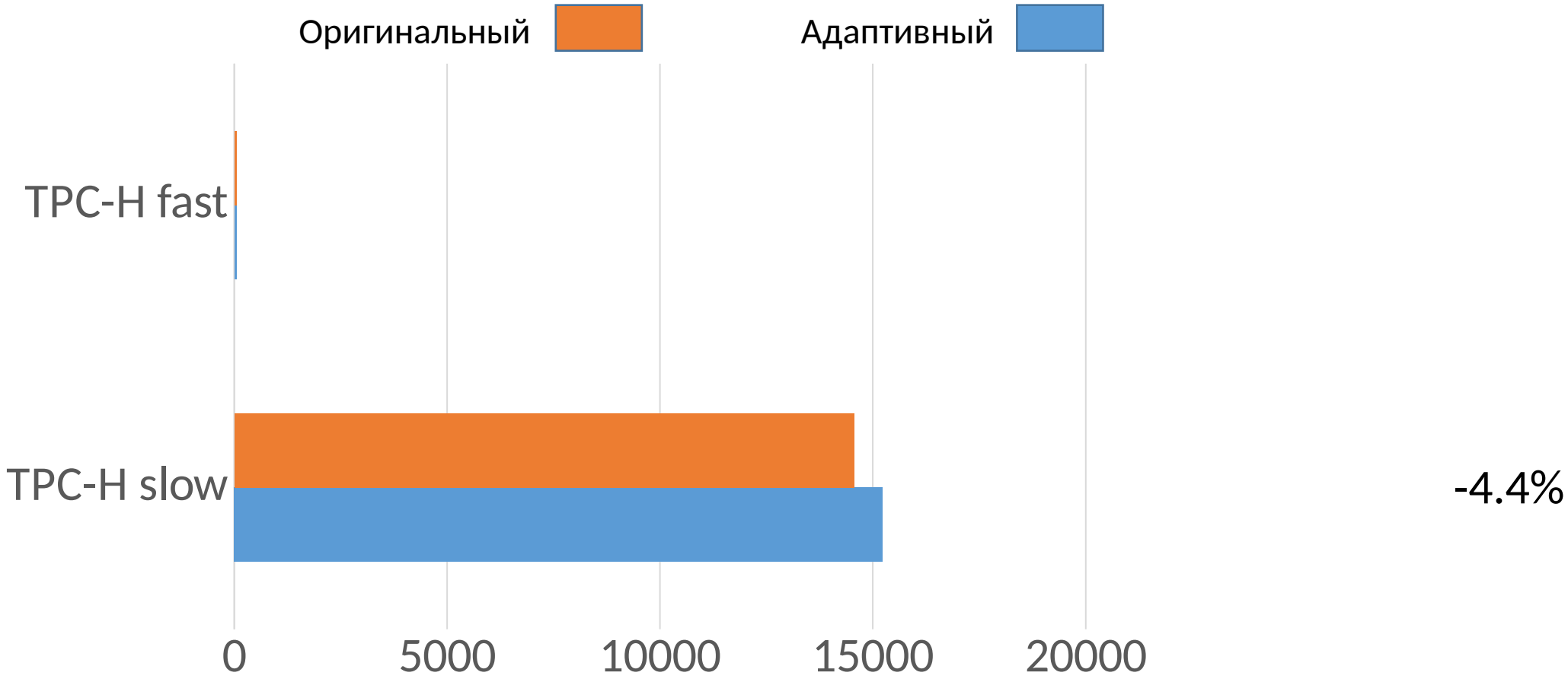


Прирост производительности



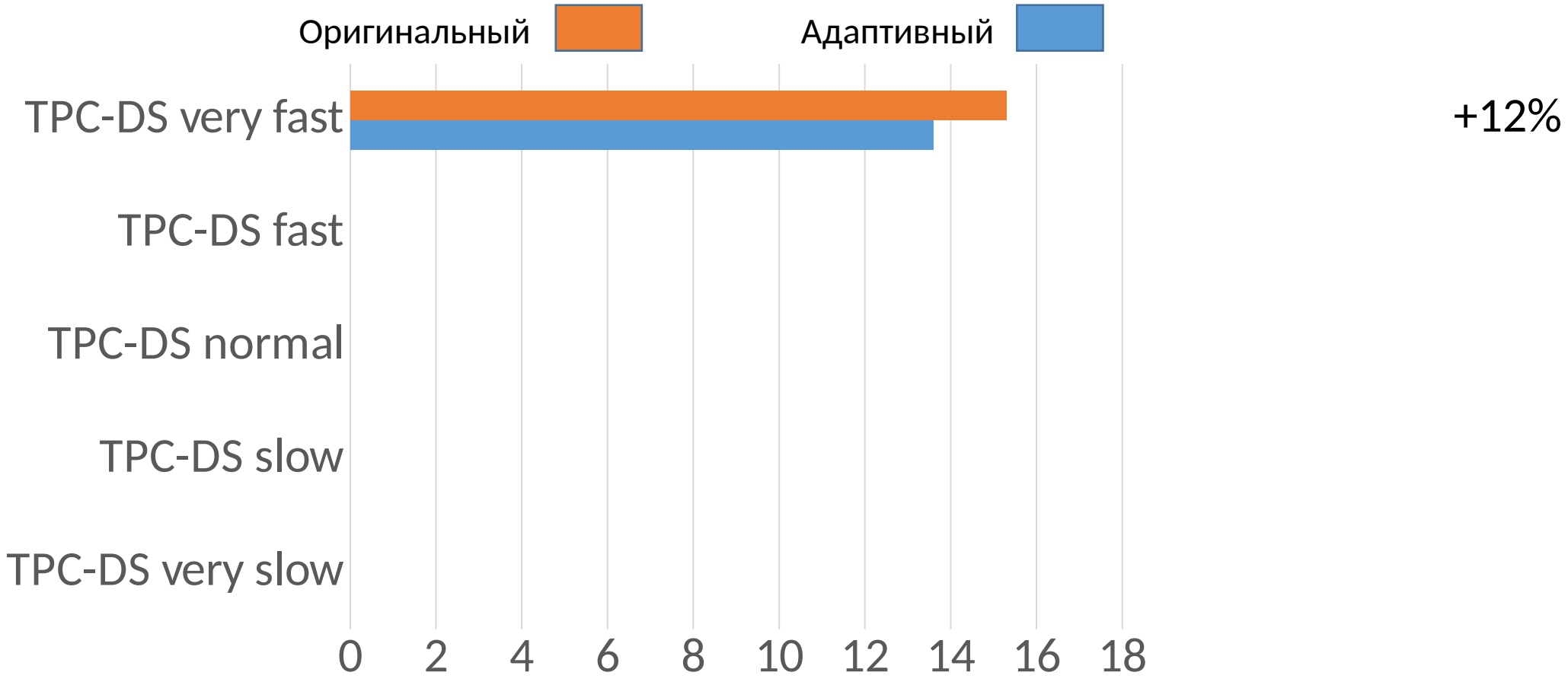
+1.3%

Прирост производительности



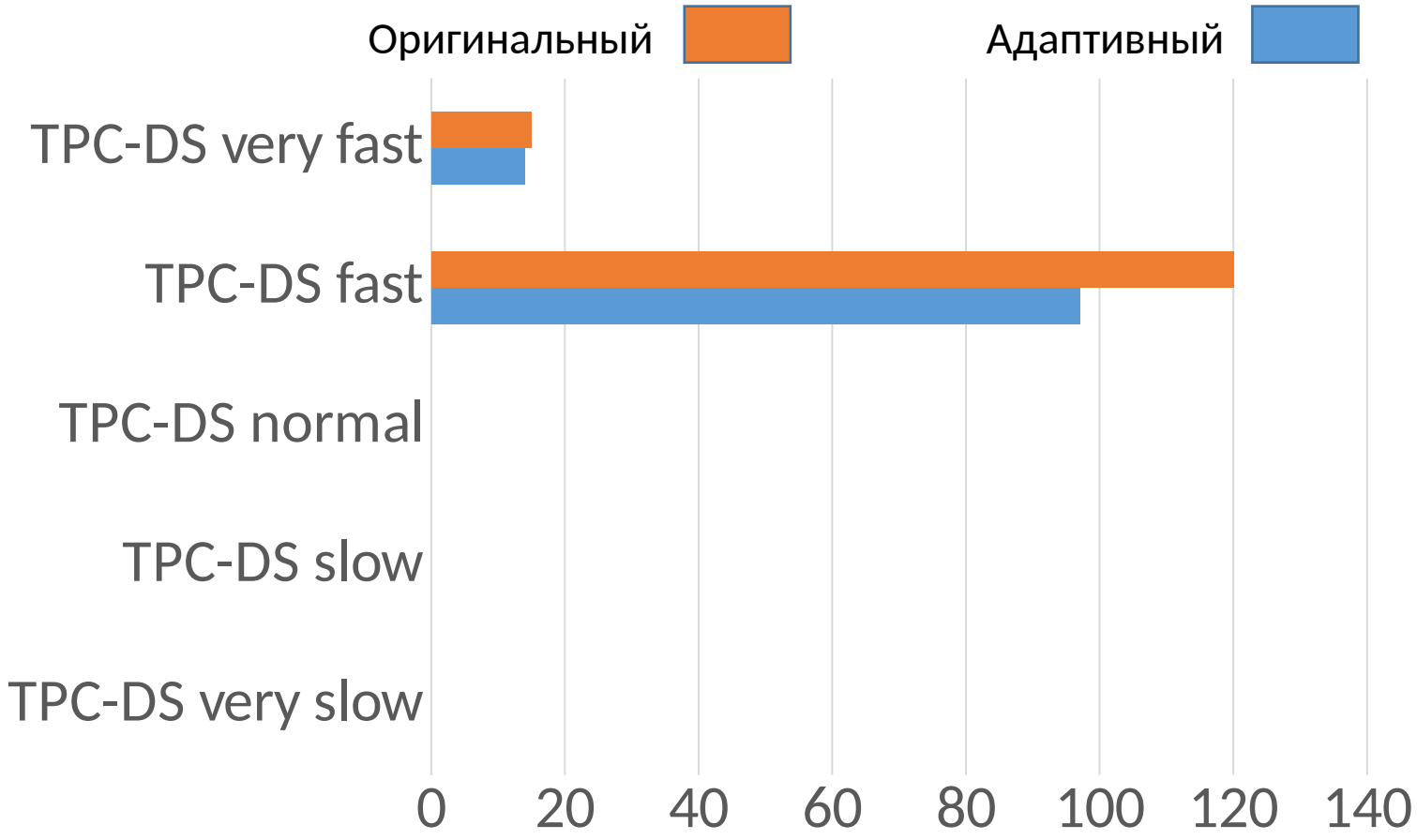
-4.4%

Прирост производительности



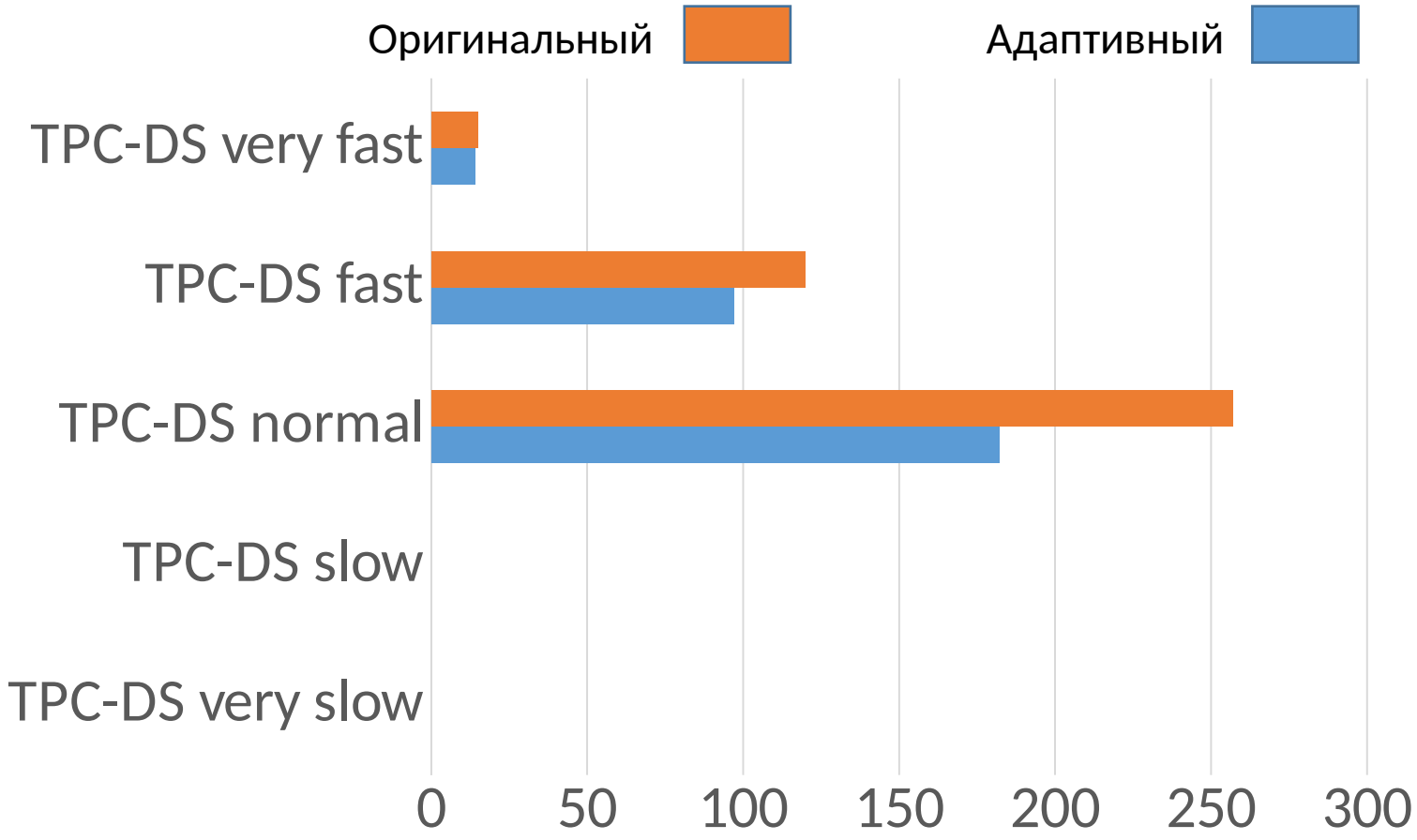
+12%

Прирост производительности



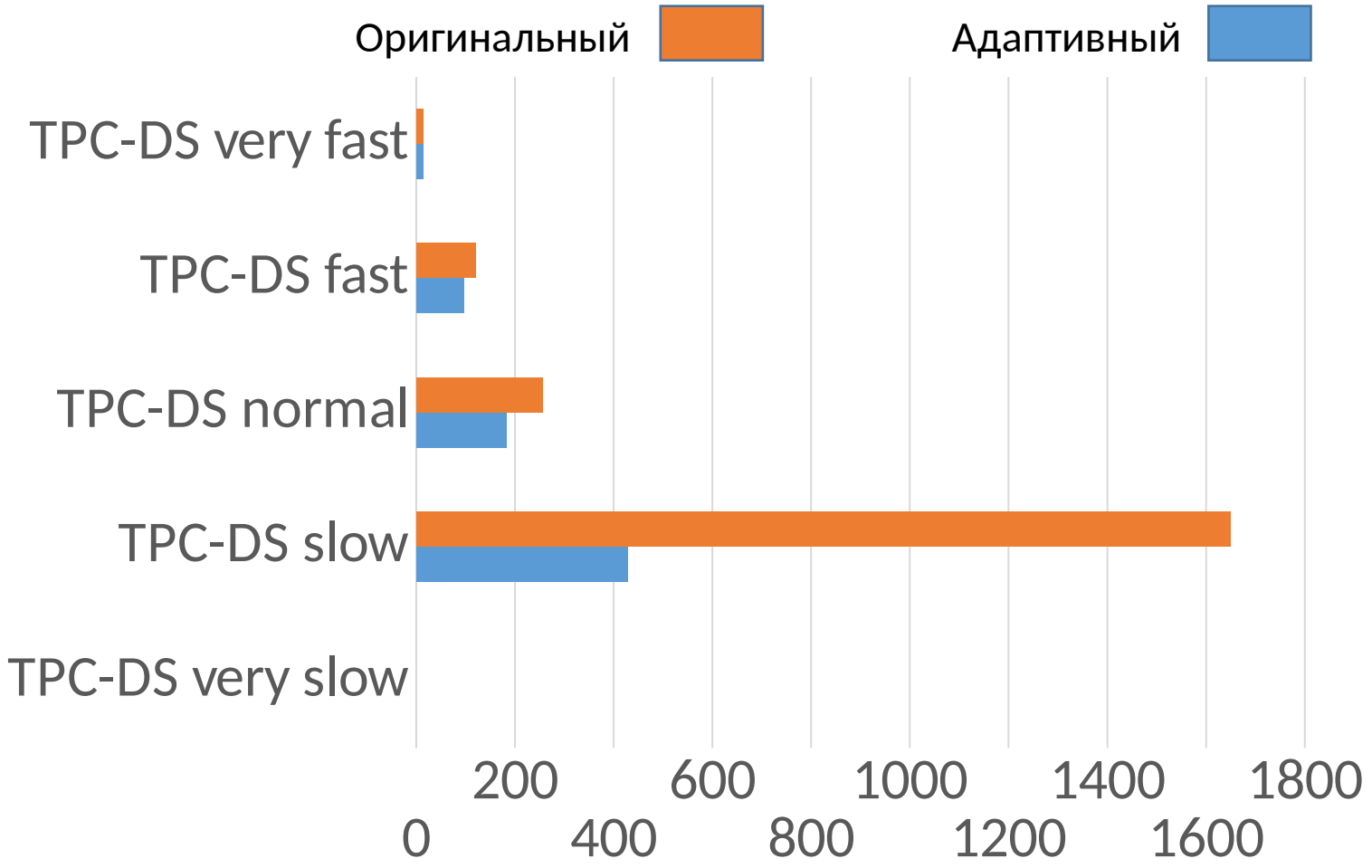
+24%

Прирост производительности



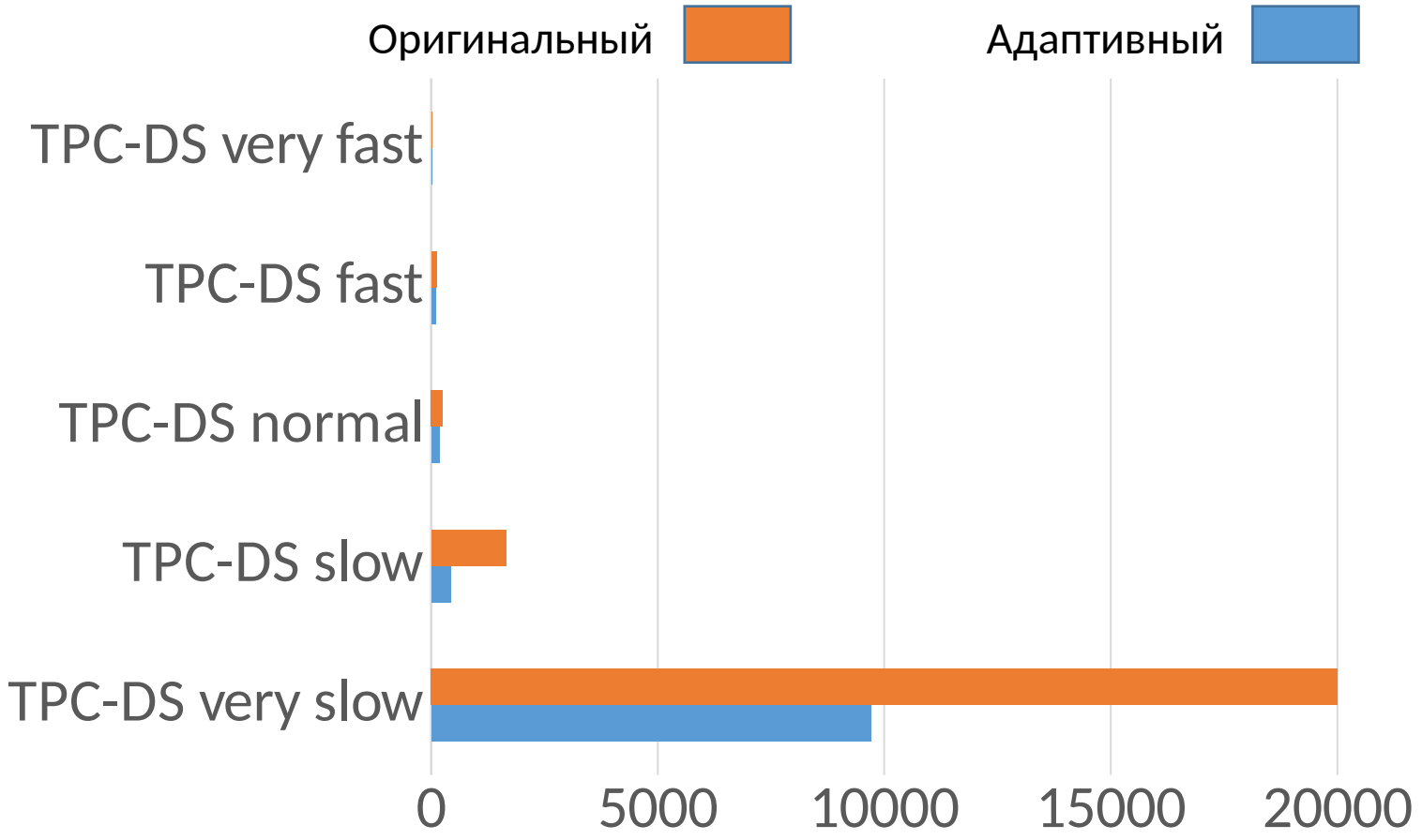
+41%

Прирост производительности



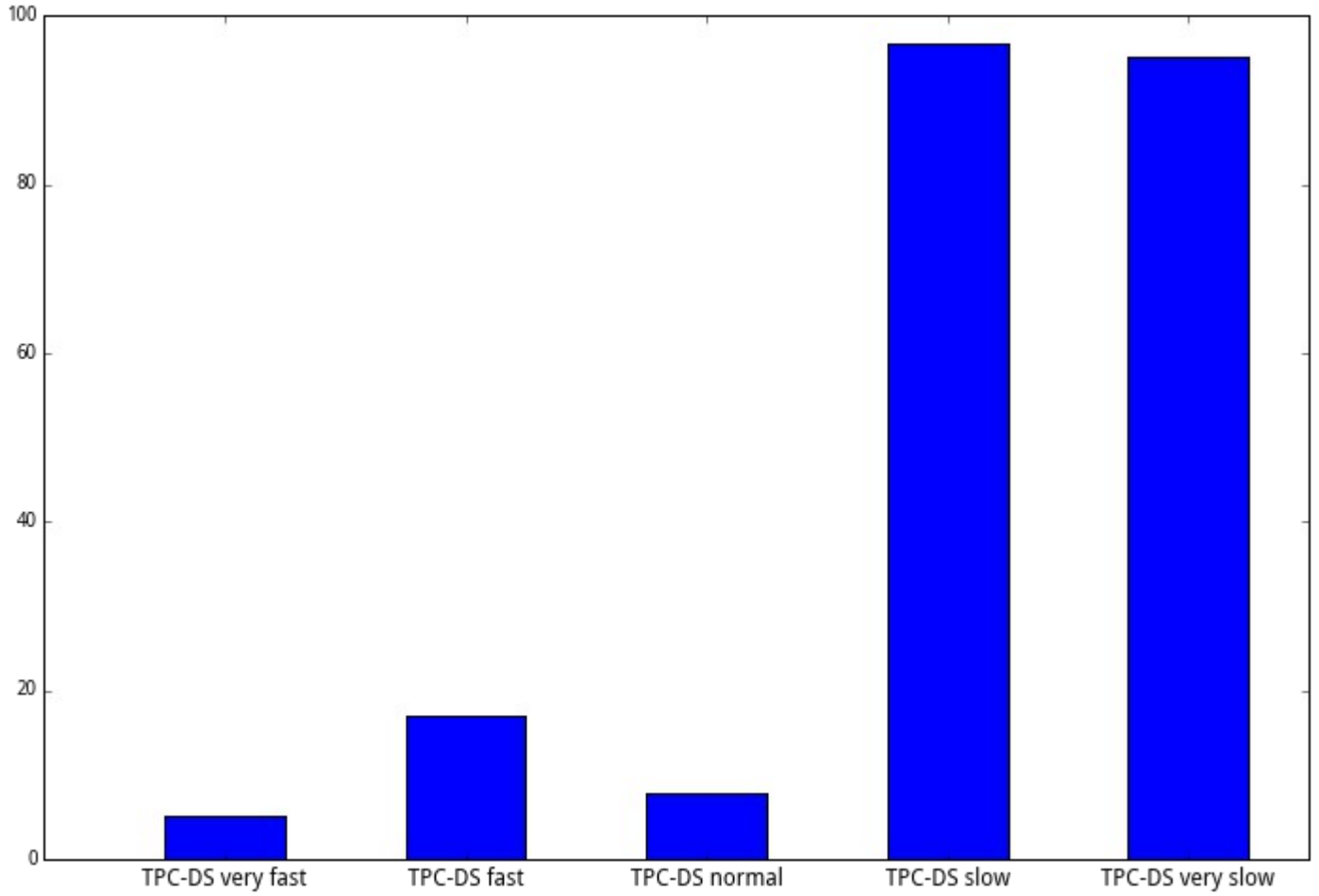
+285%

Прирост производительности



+115%

Максимальное ускорение



Накладные расходы

экспериментальные

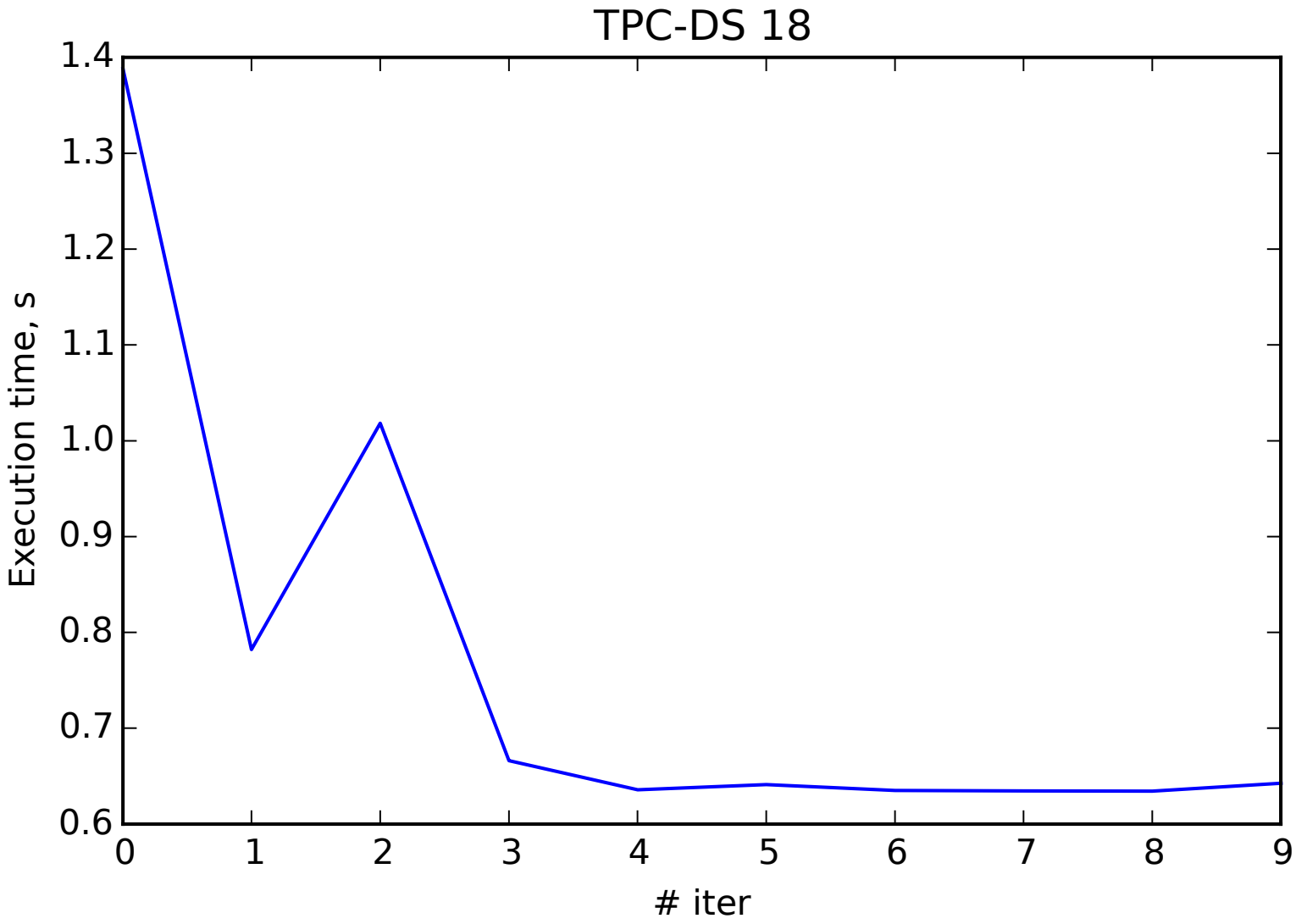
Замедление для генетического алгоритма: не более 2 секунд

Для динамического программирования: не более 30 миллисекунд

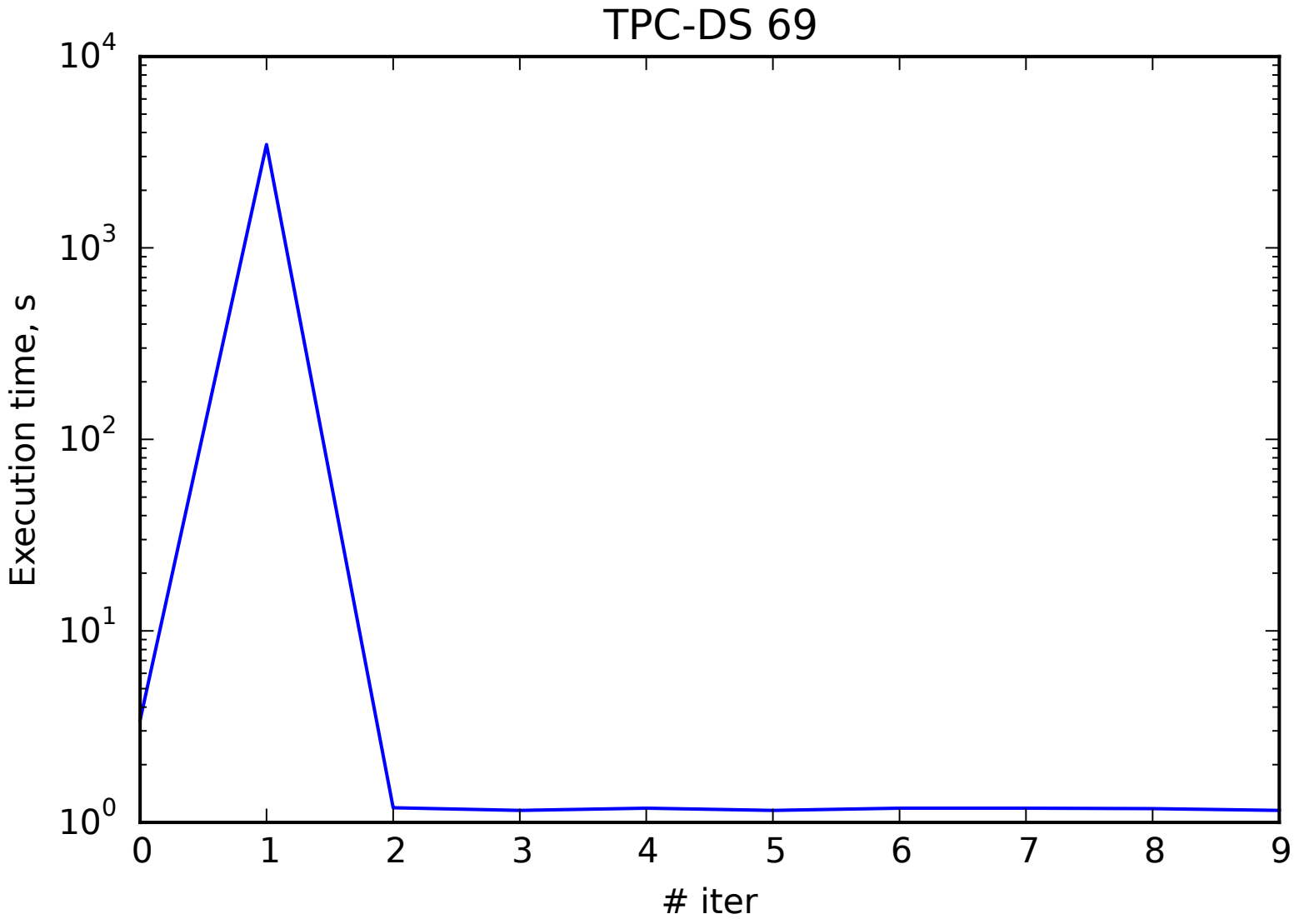
Область применимости

Сложные аналитические запросы
с повторяющейся структурой.

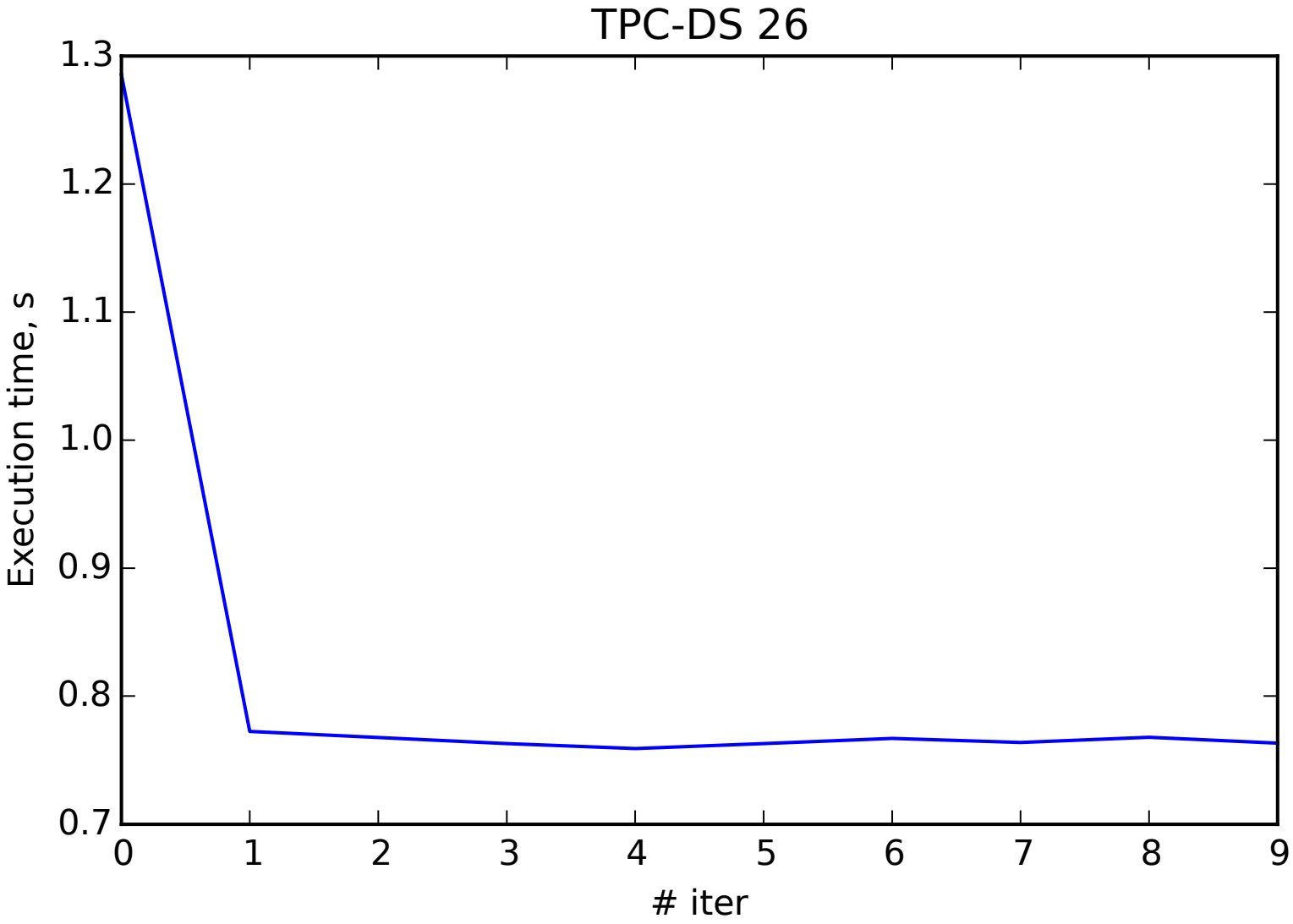
Динамика обучения



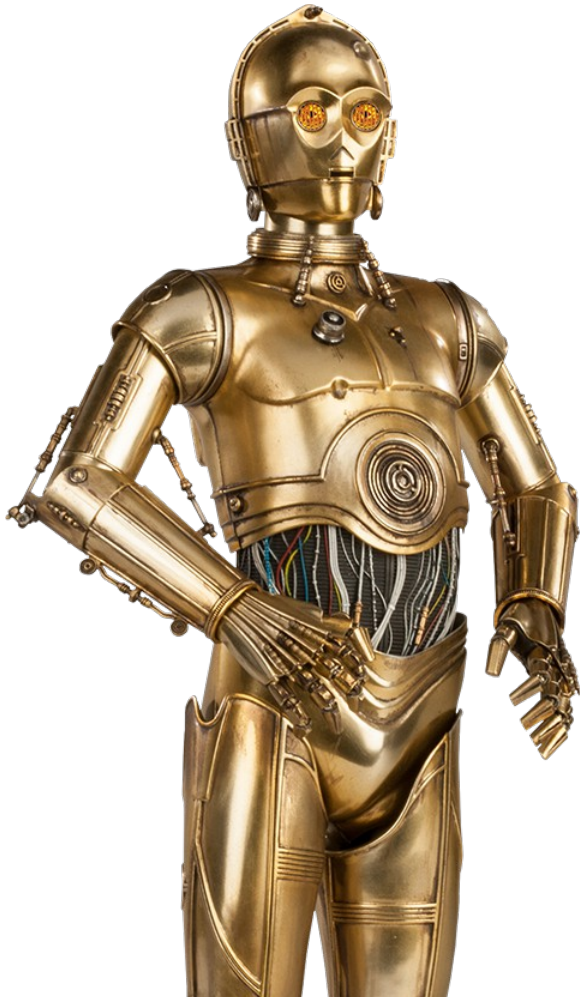
Динамика обучения

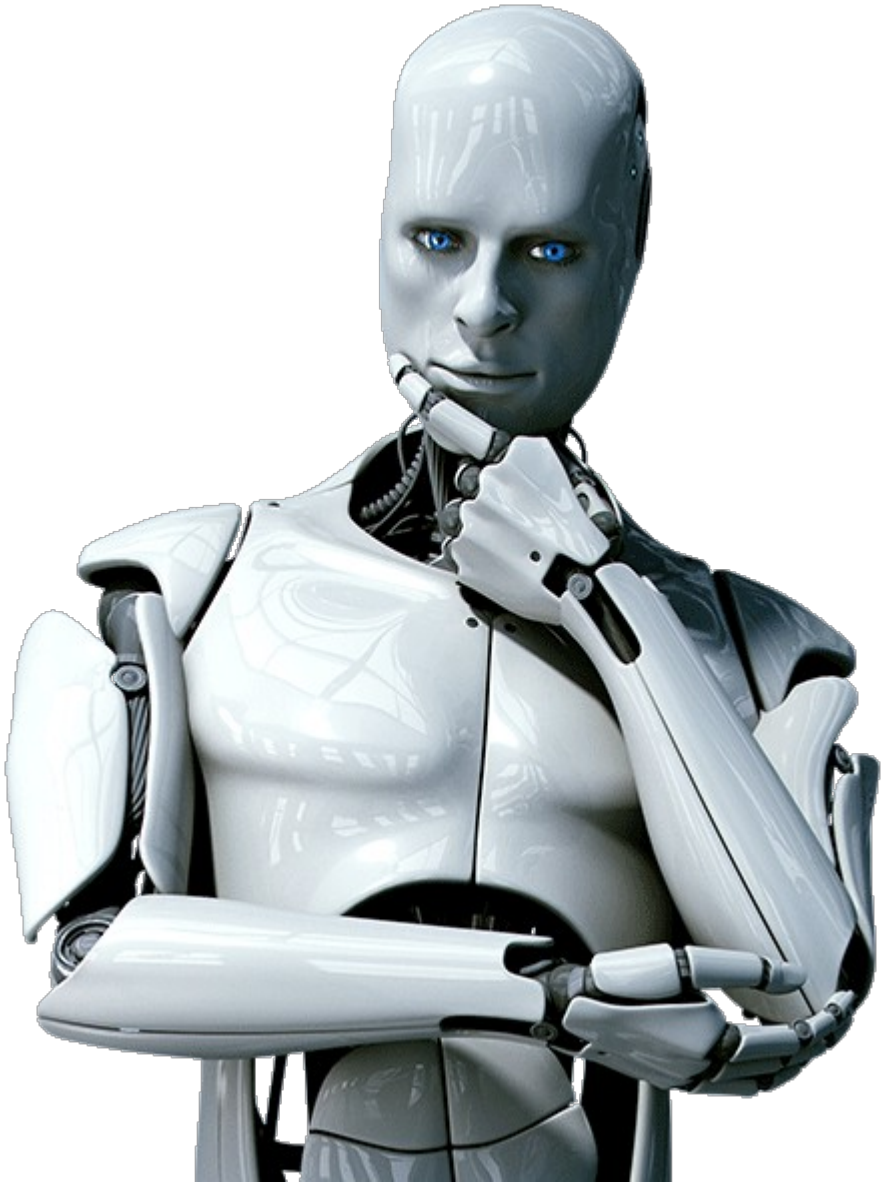


Динамика обучения



Что дальше?





Вопросы?

Текущий код для PostgreSQL:
https://github.com/tigvarts/aqo

Контакты:

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