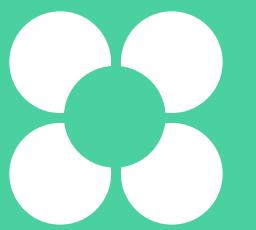


# SQL

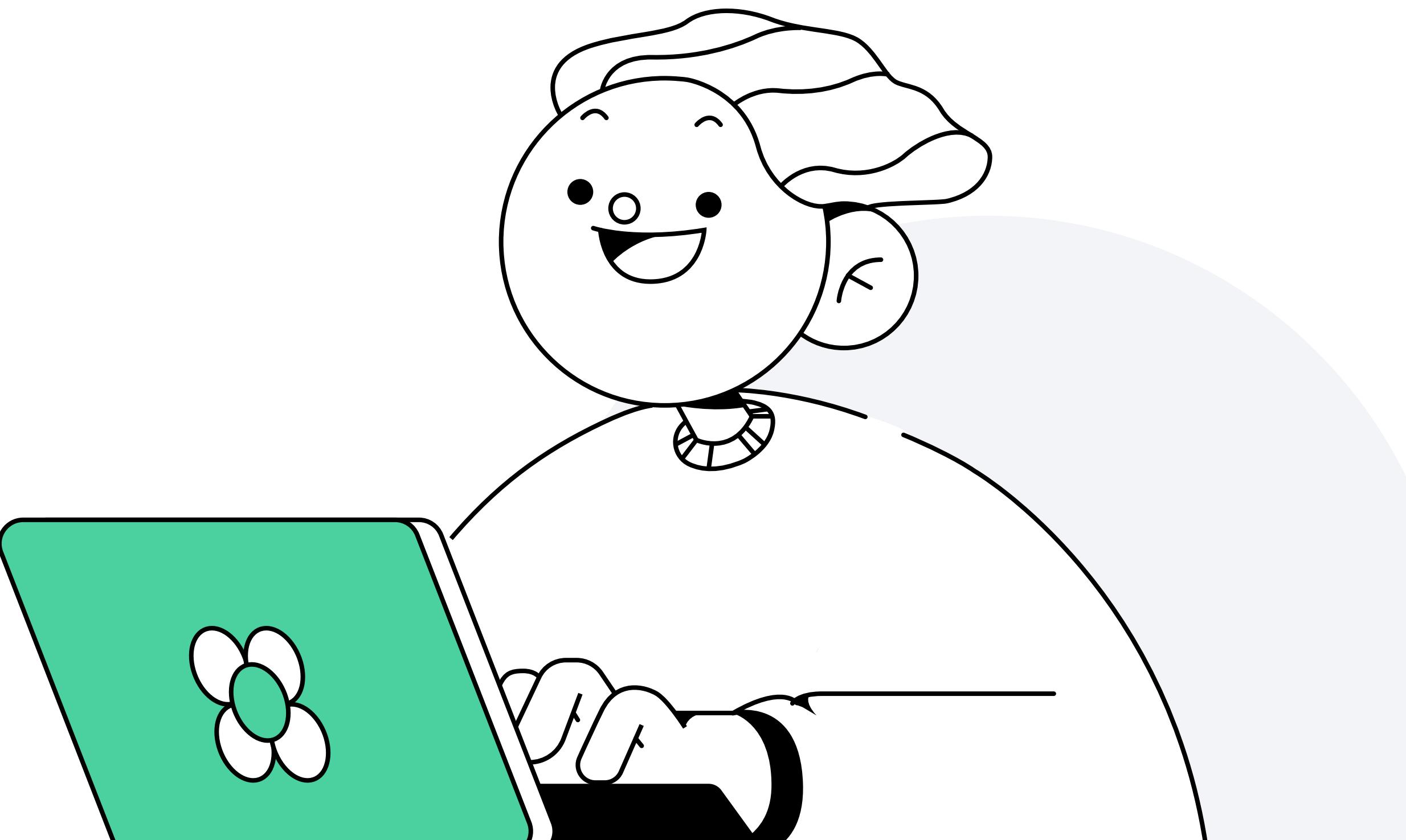
Сергей Андрюнин

DevOps-инженер, Центр биометрических технологий



# План лекции

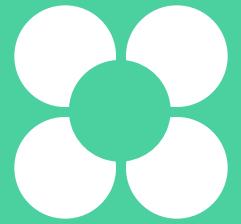
- 1 SQL-операторы
- 2 Ключи, индексы, EXPLAIN
- 3 Сложные выборки данных



# SQL-операторы

Сергей Андрюнин

DevOps-инженер, Центр биометрических технологий



# Операторы определения данных

## CREATE

- Создание БД

```
CREATE DATABASE zawod;
```

- Создание таблицы

```
CREATE TABLE kadry (nomerceh INT, tabnom SERIAL , fio CHAR(20) UNIQUE);
```

- Создание псевдотаблицы

```
CREATE VIEW poor AS SELECT tabnom, fio FROM kadry WHERE tabnom < 120;
```

- Создание индекса

```
CREATE UNIQUE INDEX indkdtb ON kadry (tabnom);
```

- Создание синонима имени таблицы

```
CREATE SYNONYM t1 FOR zawod.kadry;
```

# Операторы определения данных

## ALTER

- Изменить имя БД  
ALTER DATABASE zawod MODIFY NAME = factory;
- Изменение имени таблицы  
ALTER TABLE kadry RENAME TO persons;
- Изменение столбцов в таблице  
ALTER TABLE kadry ADD (dolzhnost CHAR(20) BEFORE fio), DROP(tabnom);
- Упорядочивание таблицы по индексу  
ALTER INDEX indkdtb TO CLUSTER;

# Операторы определения данных

## DROP

- Удалить БД  
DROP DATABASE zawod;
- Удалить таблицу  
DROP TABLE kadry;
- Удалить индекс  
DROP INDEX indkdtb;
- Удалить синоним  
DROP SYNONYM t1;
- Удалить псевдотаблицу  
DROP VIEW poor;

# Операторы манипулирования данными

## SELECT

- Вывести все записи таблицы

```
SELECT * FROM persons;
```

- Вывести количество записей в таблице

```
SELECT COUNT(*) FROM persons;
```

- Вывести определённые столбцы из таблицы

```
SELECT fio, tabnom from persons;
```

- Вывести данные по условию

```
SELECT fio, tabnom from persons where tabnom>100;
```

# Операторы манипулирования данными

## SELECT

- Вывести только уникальные значения  
SELECT DISTINCT fio, tabnom from persons;
- Вывести упорядоченные данные по признаку: ASC – по возрастанию, DESC – по убыванию  
SELECT fio, tabnom from persons ORDER BY tabnom ASC;
- Вывести сгруппированные значения по признаку  
SELECT COUNT(\*) from persons GROUP BY fio;

# Операторы манипулирования данными

## INSERT

- Вставка данных в таблицу

```
INSERT INTO persons VALUES (1, 123, "Пупкин");
```

- Вставка данных в таблицу с указанием столбца

```
INSERT INTO persons (nomerseh, tabno, fio) VALUES (1, 123, "Пупкин");
```

# Операторы манипулирования данными

## UPDATE

- Изменение поля в конкретной строке

```
UPDATE persons SET fio = 'Alfred Schmidt' WHERE tabno = 1;
```

- Изменение поля во всей таблице

```
UPDATE persons SET fio = 'Alfred Schmidt';
```



Важно

Если не указывать конкретную строку (через оператор WHERE),  
изменения затронут всю таблицу

# Операторы манипулирования данными

## DELETE

- Удалить все данные в таблице  
`DELETE FROM persons;`
- Удалить конкретную строку (или набор строк)  
`DELETE FROM persons WHERE tabno=1;`



Важно

Если не указывать конкретную строку (через оператор WHERE),  
изменения затронут всю таблицу

# Операторы манипулирования данными

- Синтаксис создания записи

`INSERT INTO table_name VALUES (value1, value2, value3, ...);`

`INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);`

- Синтаксис получения записи

`SELECT * FROM table_name;`

`SELECT column1, column2, ... FROM table_name;`

- Синтаксис обновления записи

`UPDATE table_name SET column1 = value1, ... WHERE condition;`

- Синтаксис удаления записи

`DELETE FROM table_name WHERE condition;`

# Операторы доступа к данным

## GRANT

Синтаксис выдачи прав

```
GRANT privilege_name ON object_name to {user_name | public | role_name};
```

Пример

```
GRANT ALL ON customer TO iwanow, petrow;
```

```
GRANT UPDATE(fname, lname, company, city),SELECT ON customer TO PUBLIC;
```

# Создание пользователя

POSTGRESQL

```
CREATE USER testuser WITH PASSWORD 'myPassword';
vi /var/lib/pgsql/9.6/data/pg_hba.conf
```

```
# IPv4 local connections:
```

```
host all testuser x.x.x.x/y md5
```

MySQL

```
CREATE USER 'some_user'@'somehost.somedomain' IDENTIFIED BY 'some_password';
FLUSH PRIVILEGES;
```

# Операторы доступа к данным

## REVOKE

Синтаксис отзыва прав

```
REVOKE privilege_name ON object_name FROM {user_name | public | role_name};
```

Пример

```
REVOKE ALL ON customer FROM iwanow, petrow;
```

```
REVOKE UPDATE(fname, lname, company, city),SELECT ON customer FROM PUBLIC;
```

# Операторы доступа к данным

## DENY

Синтаксис запрета

```
DENY privilege_name ON object_name TO {user_name | public | role_name};
```

Пример

```
DENY ALL ON customer TO iwanow, petrow;
```

```
DENY UPDATE(fname, lname, company, city),SELECT ON customer TO PUBLIC;
```

# Операторы управления транзакциями

Пример транзакции для PostgreSQL

```
# начало транзакции
BEGIN;

# обновляем данные
UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';

# ставим точку сохранения
SAVEPOINT my_savepoint;

# обновляем данные
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';

# допустили ошибку, возвращаемся к my_savepoint
ROLLBACK TO my_savepoint;

# теперь правильно обновляем данные
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally';

# завершаем транзакцию
COMMIT;
```

# Операторы управления транзакциями

Пример транзакции для MySQL

```
# начало транзакции
START TRANSACTION;

# обновляем данные
UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';

# ставим точку сохранения
SAVEPOINT my_savepoint;

# обновляем данные
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';

# допустили ошибку, возвращаемся к my_savepoint
ROLLBACK TO my_savepoint;

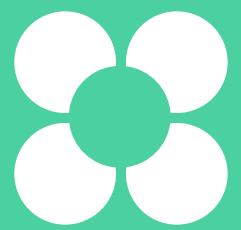
# теперь правильно обновляем данные
UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally';

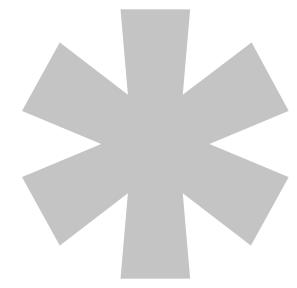
# завершаем транзакцию
COMMIT;
```

# Ключи, индексы, EXPLAIN

Сергей Андрюнин

DevOps-инженер, Центр биометрических технологий



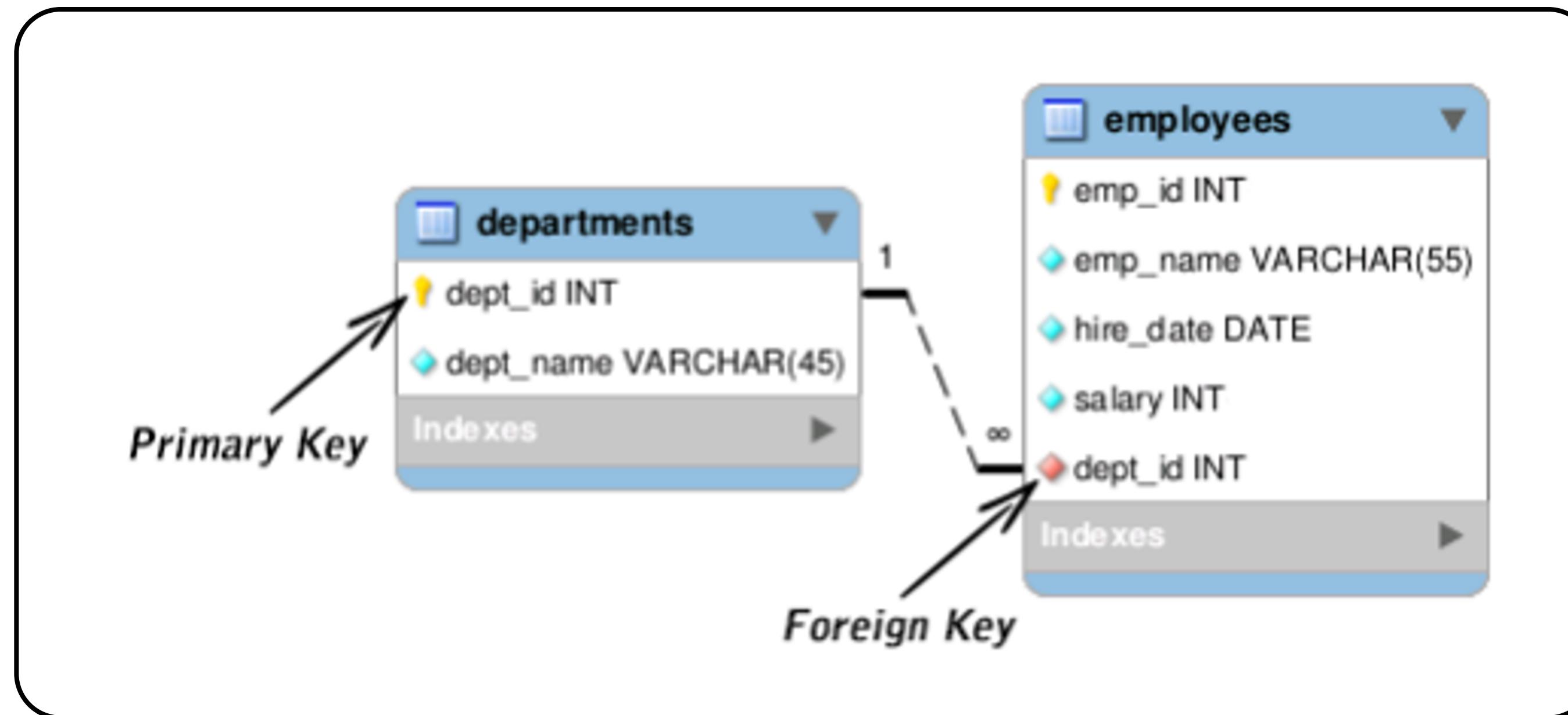


# Ключи

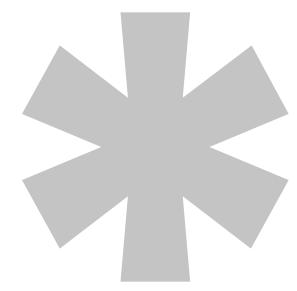
это некие сущности, созданные для установления определённых ограничений, которые поддерживают целостность и доступность данных в таблицах баз данных

# Первичные и внешние ключи

- Первичный ключ, или primary key, означает, что в таблице значение колонки primary key не может повторяться
- Внешний ключ, или foreign key, устанавливает взаимосвязь между данными в разных таблицах



[Источник](#)



# Транзакция

это набор последовательных операций над БД,  
представляющих логическую единицу

Транзакция применяется полностью или не применяется совсем

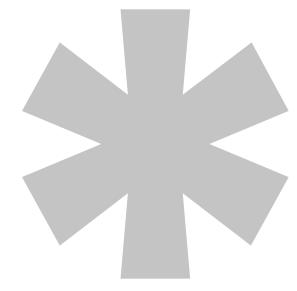
# Транзакции

## Пример транзакции

Необходимо перевести с банковского счёта номер 1 на счёт номер 2 сумму в 10 рублей

Этого можно достичь приведённой последовательностью действий (транзакцией):

- |   |                                      |   |                                      |
|---|--------------------------------------|---|--------------------------------------|
| 1 | Прочесть баланс на счету номер 1     | 4 | Прочесть баланс на счету номер 2     |
| 2 | Уменьшить баланс на 10 рублей        | 5 | Увеличить баланс на 10 рублей        |
| 3 | Сохранить новый баланс счёта номер 1 | 6 | Сохранить новый баланс счёта номер 2 |



# Индексы

это специальные структуры в базах данных, которые позволяют ускорить поиск и сортировку по определённому полю или набору полей в таблице, а также используются для обеспечения уникальности данных

# Индексы

## Общие принципы, связанные с созданием индексов

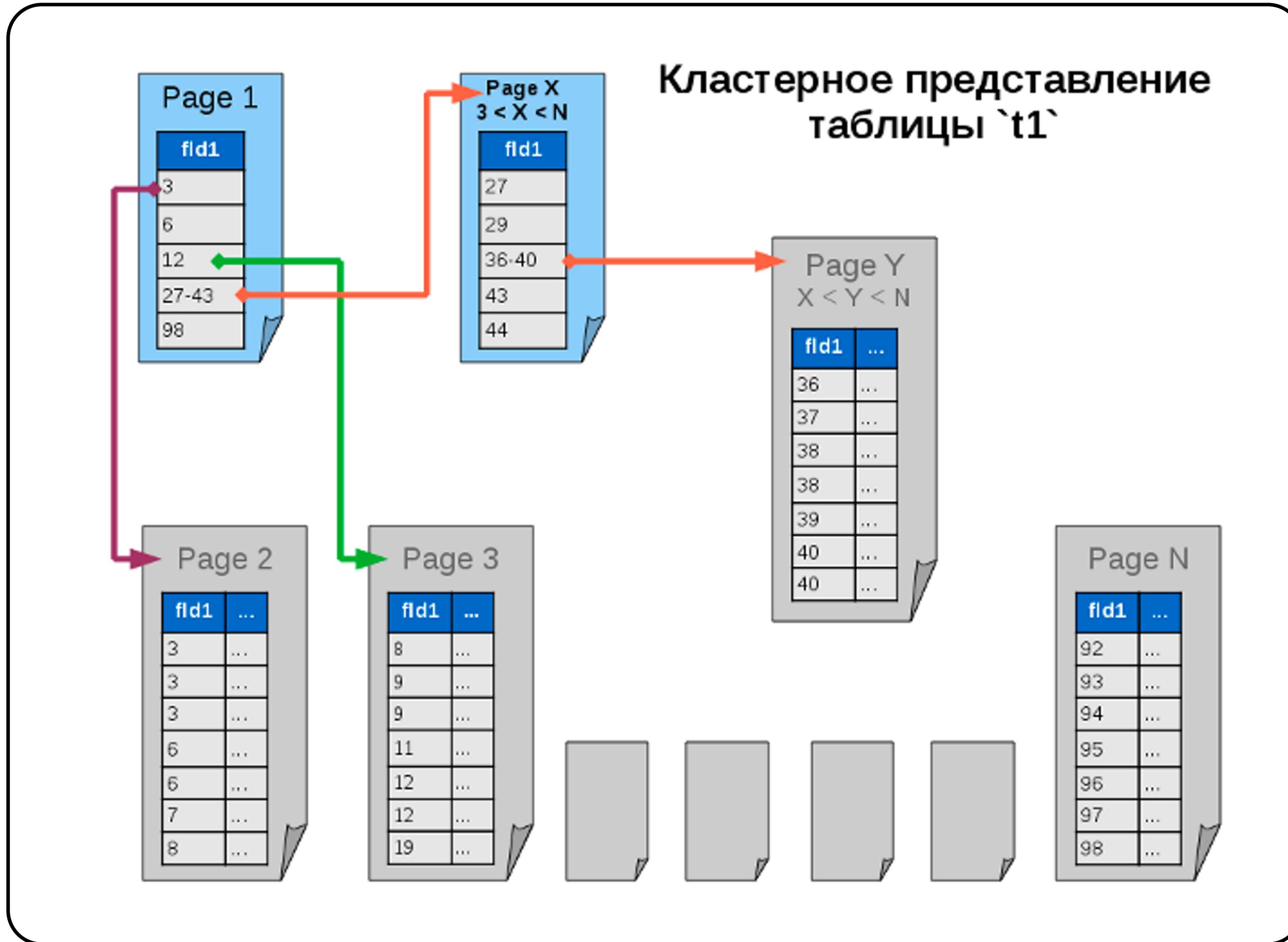
- Индексы необходимо создавать для столбцов, которые используются в JOIN-операциях, по которым часто производятся поиск и операции сортировки
- Для столбцов, на которые наложено ограничение уникальности, индекс создаётся в автоматическом режиме
- Индексы лучше создавать для тех полей, в которых минимальное число повторяющихся значений и данные распределены равномерно
- При внесении изменений в таблицы автоматически изменяются и индексы, наложенные на эту таблицу

# Индексы

Индексы можно разделить на следующие подгруппы:

- кластерные
- некластерные

# Индексы

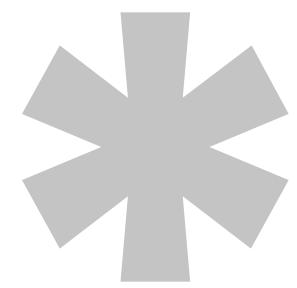


[Источник](#)

# Индексы

Большинство современных СУБД поддерживает несколько полезных алгоритмов индексации.

- **Индексы в В-деревьях**, такие как INDEX, FULLTEXT, PRIMARY KEY и UNIQUE
- **Индексы в R-деревьях**, например индексы для пространственных типов данных
- **Хеш-индексы и инвертированные списки** при использовании индексов FULLTEXT

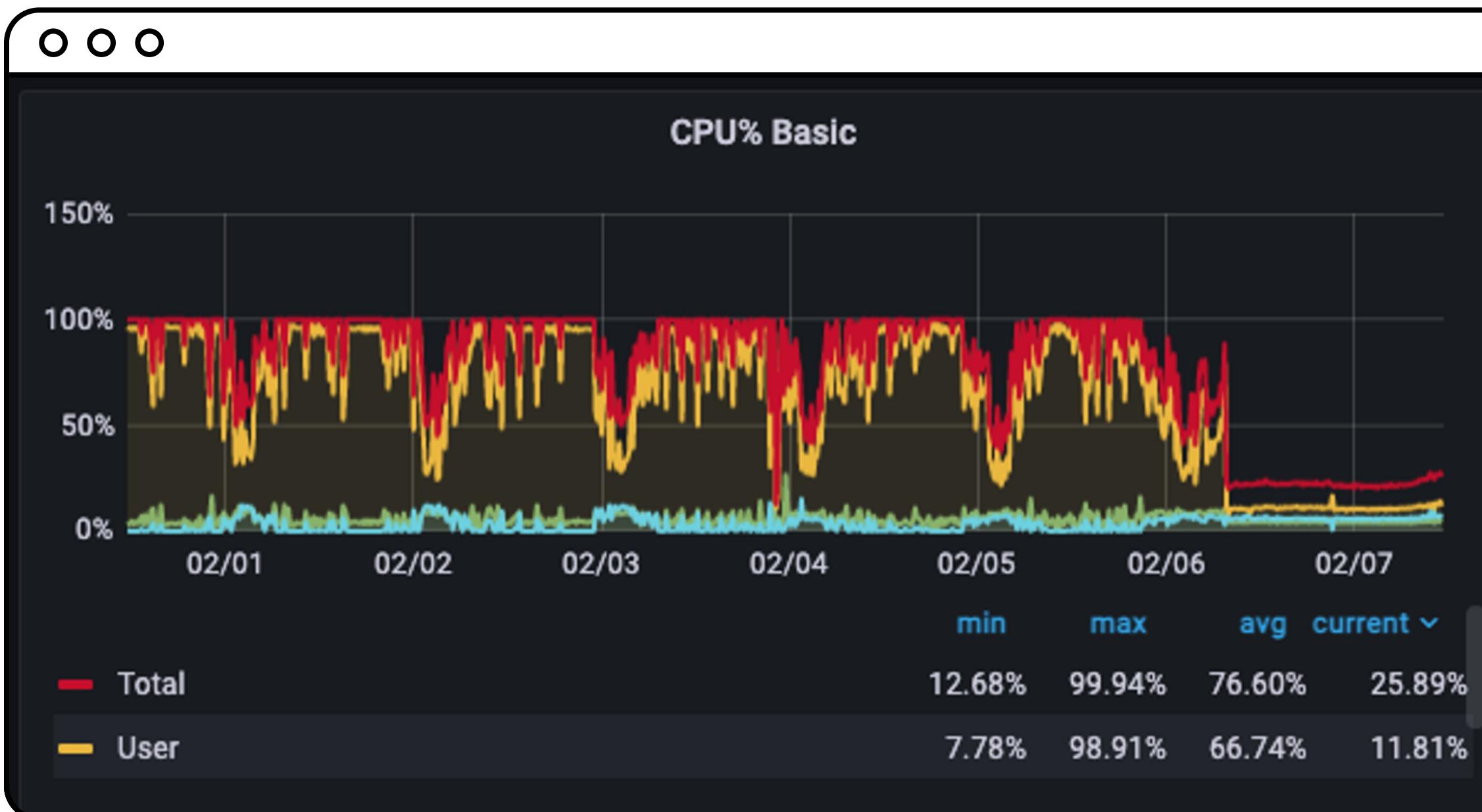


# **EXPLAIN**

это оператор SQL, предоставляющий полную информацию выполнения запроса

# EXPLAIN

short_query	total_time	calls	mean	percentage_cpu
SELECT EXISTS(SELECT \$2 FROM common_request WHERE external_uuid = \$1::uuid)	3879115512.47	40454	95889.54	93.26



# EXPLAIN

Синтаксис применения

EXPLAIN {запрос};

Пример запроса

EXPLAIN SELECT \* FROM table\_name;

```
o o o
*****
1. row ****
id: 1
select_type: SIMPLE
table: categories
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 4
Extra:
1 row in set (0.00 sec)
```

# EXPLAIN

Описание параметров вывода: пример для MySQL

Параметр	Назначение
id	Порядковый номер для каждого SELECT внутри запроса (когда имеется несколько подзапросов)
select_type	Тип запроса SELECT
table	Таблица, к которой относится выводимая строка
type	Тип связи используемых таблиц
possible_keys	Индексы, которые могут быть использованы для нахождения строк в таблице
key	Использованный индекс
key_len	Длина индекса
ref	Столбцы или константы, которые сравниваются с индексом, указанным в поле key
rows	Число записей, обработанных для получения выходных данных
Extra	Содержит дополнительную информацию, относящуюся к плану выполнения запроса