

# Реляционные базы данных: Работа с данными. DDL, DML, DCL, TCL.



Роман  
Гордиенко



**Роман Гордиенко**

Backend Developer, Factory5



# План занятия

1. [SQL – историческая справка.](#)
2. [MySQL по CAP и PACELC, архитектура.](#)
3. [Логический порядок обработки инструкции SELECT.](#)
4. [DDL.](#)
5. [DML.](#)
6. [DCL.](#)
7. [TCL.](#)
8. [Итоги.](#)
9. [Домашнее задание.](#)



# SQL — историческая справка



## SQL – историческая справка

В начале 1970-х годов в одной из исследовательских лабораторий компании IBM была разработана экспериментальная реляционная СУБД IBM System R, для которой затем был создан специальный язык SEQUEL, позволявший относительно просто управлять данными в этой СУБД.

Источник



## SQL — историческая справка

Аббревиатура SEQUEL расшифровывалась как Structured English QUery Language — «структурированный английский язык запросов». Позже язык SEQUEL был переименован в SQL.

Когда в 1986 году первый стандарт языка SQL был принят ANSI (American National Standards Institute), официальным произношением стало [ˌes kjuːˈel] — эс-кью-эл.

Источник



## SQL — историческая справка

Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования.

Разработкой языка запросов занимались Дональд Чэмбэрлин (Donald D. Chamberlin) и Рэй Бойс (Ray Boyce). Пэт Селинджер (Pat Selinger) занималась разработкой стоимостного оптимизатора (cost-based optimizer), Рэймонд Лори (Raymond Lorie) занимался компилятором запросов.



## SQL – историческая справка

SEQUEL был не единственным языком подобного назначения. В Калифорнийском Университете Беркли была разработана некоммерческая **СУБД Ingres** (являвшаяся дальним прародителем популярной сейчас некоммерческой СУБД PostgreSQL), которая являлась реляционной СУБД, но использовала свой собственный **язык QUEL**, который, не выдержал конкуренции по количеству поддерживающих его СУБД по сравнению с языком SQL.

Источник





## SQL – историческая справка

Первыми СУБД, поддерживающими новый язык SQL, стали в 1979 году Oracle V2 для машин VAX от компании Relational Software (впоследствии ставшей компанией Oracle) и System/38 от IBM, основанная на System/R.

Источник

# SQL — историческая справка

В 1986 году ANSI представил свою первую версию стандарта, описанную в документе ANSI X3.135-1986 под названием «Database Language SQL». Неофициально этот стандарт SQL-86 получил название SQL1.

В 1987 году была завершена работа над версией стандарта ISO 9075-1987 под тем же названием. Разработка этого стандарта велась под патронажем Технического Комитета TC97. Именно его подразделение, именуемое как Подкомитет SC21, курировало разработку стандарта, что стало залогом идентичности стандартов ISO и ANSI для SQL1 (SQL-86).

Источник

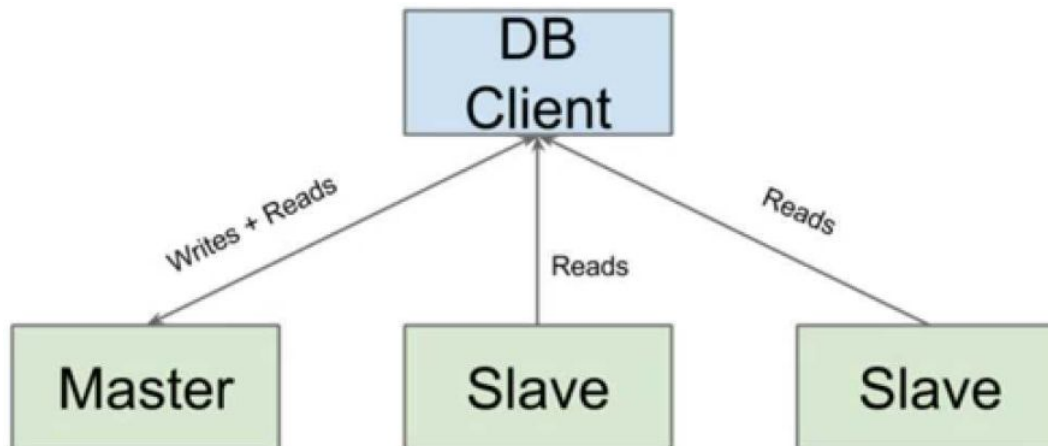


# MySQL по CAP и PACELC, архитектура

# MySQL по CAP и PACELC

Теоремы CAP и PACELC предназначены для распределенных баз данных. По умолчанию MySQL имеет репликацию master-slave.

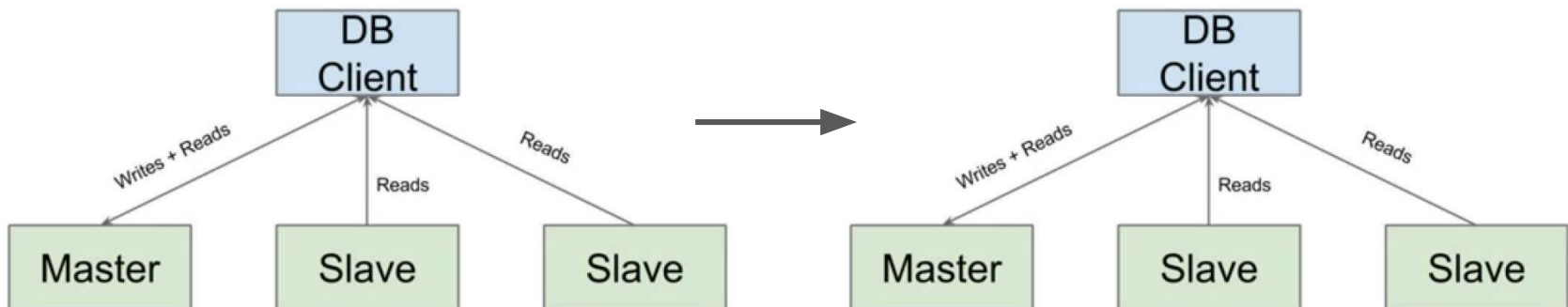
Таким образом, MySQL имеет просто первичные и вторичные узлы, соответственно для обработки данных используется только master и следовательно нарушается толерантность к разделению.



# MySQL по CAP и PACELC

Получаем, что MySQL по теореме CAP является CA системой. Но при этом MySQL можно сконфигурировать в CP систему, сформировав несколько распределенных master-slave узлов.

По теореме PACELC MySQL относится к PC/EC системам, так как данные в системе должны быть согласованы в ущерб доступности данных и времени ответа кластера.





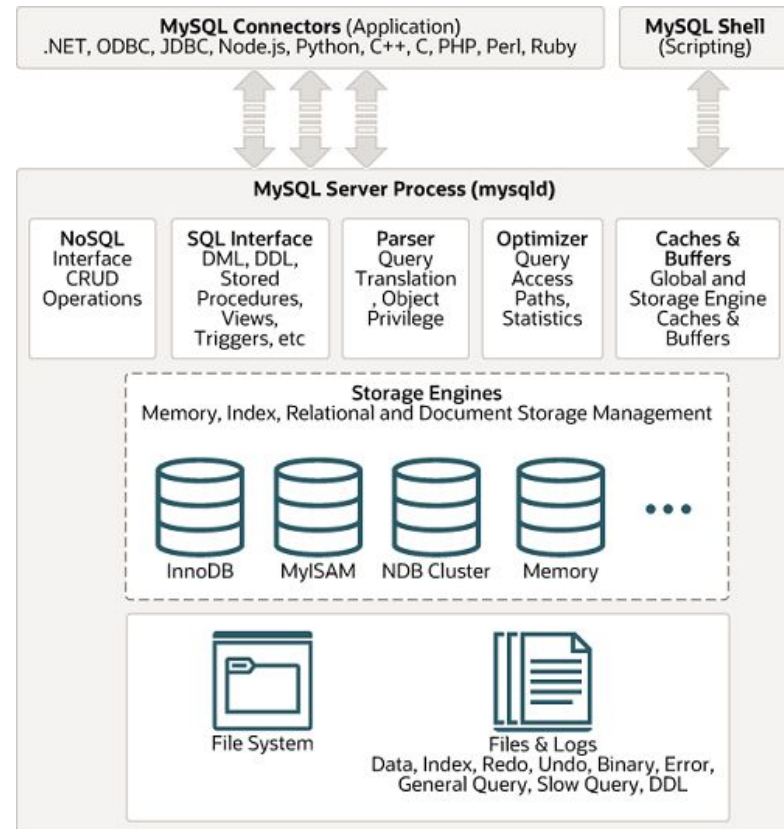
# MySQL архитектура

MySQL — это открыто распространяемая СУБД и функционирует по модели клиент-сервер.

**Отличительной особенностью MySQL** является возможность выбора подсистемы хранения, в которой обработка запросов и другие серверные задачи отделены от хранения и извлечения данных. Подобное разделение задач позволяет выбирать способ хранения данных, а также настраивать производительность, ключевые характеристики и так далее.

# MySQL архитектура

- Клиентские соединения
- Службы и утилиты
- Интерфейс
- Синтаксический анализатор
- Оптимизатор
- Кеши и буферы
- Подключаемые механизмы хранения
- Файловая система





# MySQL механизмы хранения

Самыми популярными встроенными системами хранения в MySQL являются InnoDB и MyISAM.

Менее популярными и востребованными являются Archive, Blackhole, CSV, Federated, Memory, Merge, Example и NDB Cluster.

При этом с версии MySQL 8.0 поддержка механизма MyISAM прекращена, использование возможно, но дальнейшая судьба неизвестна.



---

# MySQL. InnoDB.

Это механизм хранения по умолчанию для **MySQL 5.5** и выше:

- придерживается требований ACID;
- поддерживает ограничения ссылочной целостности FOREIGN KEY;
- поддерживает функции фиксации, отката и восстановления после сбоя для защиты данных;
- поддерживает блокировку на уровне строк — это «согласованное чтение без блокировки» повышает производительность при использовании в многопользовательской среде;
- хранит данные в кластерных индексах, что уменьшает количество операций ввода-вывода для запросов на основе первичных ключей.

---

# MySQL. MyISAM.

Это механизм хранения по умолчанию для **MySQL 5.3** и ниже:

- управляет нетранзакционными таблицами;
- обеспечивает высокоскоростное хранение и извлечение;
- поддерживает полнотекстовый поиск;
- данные хранятся в кроссплатформенном формате, что позволяет переносить базы с сервера непосредственным копированием файлов, минуя промежуточные формы;
- допускается индексирование текстовых столбцов, в том числе и переменной длины и т.д.


# MySQL сервер. Docker. IDE. Dump.

Для того чтобы скачать и установить последнюю версию MySQL Community Server необходимо перейти по [ссылке](#).

Также можно поднять чистый [контейнер](#) с MySQL.

Для удобной работы с СУБД можно скачать и установить [DBeaver](#) Community Edition.

[Ссылка](#) на дамп файл с учебной базой данных.



# Логический порядок обработки инструкции **SELECT**



# Логический порядок

SQL, как любой язык программирования имеет свою структуру и синтаксис.

Прежде чем перейти к работе с этим языком, нужно запомнить, что структура запросов имеет свой порядок, но «под капотом» интерпретатор будет выполнять запрос в определенной логической последовательности, и от этого будут зависеть области видимости и с какими данными, что будет происходить.

---

# Логический порядок

1. FROM.
2. ON.
3. JOIN.
4. WHERE.
5. GROUP BY.
6. WITH CUBE или WITH ROLLUP.
7. HAVING.
8. SELECT.
9. OVER.
10. DISTINCT.
11. ORDER BY.
12. В начало.



**DDL**

---

# DDL

**Data Definition Language (DDL)** – это группа операторов определения данных. С помощью этих операторов определяется структура базы данных и происходит работа с объектами базы данных.

В эту группу входят следующие операторы:

- **CREATE** – используется для создания объектов базы данных;
- **ALTER** – используется для изменения объектов базы данных;
- **DROP** – используется для удаления объектов базы данных.



---

# DDL. CREATE

- Создание базы данных:

```
CREATE DATABASE `dvd-rental`;
```

- Создание таблицы:

```
CREATE TABLE customer(id SERIAL PRIMARY KEY, fio CHAR(50) UNIQUE, dob DATE);
```

- Создание представления:

```
CREATE VIEW underage AS SELECT fio FROM customer WHERE TIMESTAMPDIFF(YEAR, dob, CURDATE()) >= 18;
```

- Создание индекса:

```
CREATE UNIQUE INDEX dob_idx ON customer(dob);
```

---

# DDL. ALTER

- Изменить имя базы данных:

```
ALTER DATABASE old_name MODIFY NAME = new_name;
```

\* синтаксис T-SQL, не работает в MySQL

- Изменение имени таблицы:

```
ALTER TABLE customer RENAME TO persons;
```

- Изменение столбцов в таблице:

```
ALTER TABLE persons ADD password CHAR(20) NOT NULL;
```

- Изменение типа данных атрибута:

```
ALTER TABLE persons MODIFY fio VARCHAR(100);
```

---

# DDL. DROP

Удалить базу данных:

```
DROP DATABASE `dvd-rental`;
```

Удалить таблицу:

```
DROP TABLE persons;
```

Удалить индекс:

```
DROP INDEX dob_idx ON persons;
```

Удалить представление:

```
DROP VIEW underage;
```



# DML

---

# DML

**Data Manipulation Language (DML)** – это группа операторов для манипуляции данными. С помощью этих операторов можно добавлять, изменять, удалять и получать данные из базы данных.

В эту группу входят самые распространенные операторы:

- **SELECT** – осуществляет выборку данных;
- **INSERT** – добавляет новые данные;
- **UPDATE** – изменяет существующие данные;
- **DELETE** – удаляет данные.

# DML. SELECT

Вывести все записи таблицы:

```
SELECT * FROM persons;
```

Вывести количество записей в таблице:

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

Вывести определенные столбцы из таблицы:

```
SELECT fio, dob FROM persons;
```

Вывести данные по условию:

```
SELECT fio, dob FROM persons WHERE id > 25;
```

# DML. INSERT

Вставка данных в таблицу:

```
INSERT INTO persons VALUES (6, «Иванов Иван», «1985-07-12»);
```

Вставка данных в таблицу с указанием столбца:

```
INSERT INTO persons (fio) VALUES («Петров Петр»);
```

Вставка данных в таблицу из другой таблицы:

```
INSERT INTO persons (fio) SELECT fio FROM customers;
```

---

## DML. UPDATE

Изменение значения в конкретной строке:

```
UPDATE persons SET fio = 'Максим Сергеевич' WHERE id = 1;
```

Изменение значений по всему атрибуту:

```
UPDATE persons SET fio = 'Максим Сергеевич';
```

---

**Важно!** Если не указывать конкретную строку (через оператор WHERE) — изменения произойдут во всех строках таблицы.



---

## DML. DELETE

Удалить все данные в таблице:

```
DELETE FROM persons;
```

Удалить конкретную строку (или набор строк):

```
DELETE FROM persons WHERE id IN (1, 3, 7);
```

---

**Важно!** Если не указывать конкретную строку (через оператор WHERE) — будут удалены все данные из таблицы.



**DCL**

# DCL

**Data Control Language (DCL)** – группа операторов определения доступа к данным. Эти операторы нужны для управления разрешениями доступа к данным и выполнения операций над объектами базы данных. Права назначаются на пользователя или на роли.

В данную группу входят следующие операторы:

- **GRANT** – предоставляет пользователю или группе разрешения на определенные операции с объектами;
- **REVOKE** – отзывает выданные разрешения;
- **DENY** – задает запрет, имеющий приоритет над разрешением (Отсутствует в MySQL).

---

## DCL. GRANT

Дать все права пользователю к базе данных и, если его не существует, то создает его (до версии 8.0):

```
GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;
```

Создать пользователя и дать все права к базе данных (после версии 8.0):

```
CREATE USER 'dbuser'@'localhost' IDENTIFIED BY 'password';  
GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'localhost';
```

Дать определенные права пользователю:

```
GRANT SELECT, UPDATE ON `dvd-rental`.* TO 'dbuser'@'localhost'
```

## DCL. REVOKE

Убрать все права пользователя к базе данных:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'dbuser'@'localhost';
```

Удалить пользователя после того, как убрали права:

```
DROP USER 'dbuser'@'localhost';
```

Убрать определенные права у пользователя:

```
REVOKE INSERT, DELETE ON `dvd-rental`.* FROM 'dbuser'@'localhost';
```

## DCL. DENY

В MySQL отсутствует оператор DENY, все права назначаются через GRANT и удаляются через REVOKE.

На примере других СУБД синтаксис будет выглядеть следующим образом:

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



**TCL**

---

# TCL

Transaction Control Language (TCL) – группа операторов для управления транзакциями.

Эта группа состоит из следующих операторов:

- **START TRANSACTION** – служит для определения начала транзакции;
- **COMMIT** – применяет транзакцию;
- **ROLLBACK** – откатывает все изменения, сделанные в контексте текущей транзакции;
- **SAVEPOINT** – устанавливает промежуточную точку сохранения внутри транзакции.



# TCL

## Пример транзакции для MySQL:

```
# начало транзакции
START TRANSACTION;
# обновляем данные
UPDATE accounts SET balance = balance - 1000.00 WHERE id = 10;
# ставим точку сохранения
SAVEPOINT my_savepoint;
# обновляем данные
UPDATE accounts SET balance = balance + 1000.00 WHERE id = 27;
# допустили ошибку, возвращаемся к my_savepoint
ROLLBACK TO my_savepoint;
# теперь правильно обновляем данные
UPDATE accounts SET balance = balance + 1000.00 WHERE id = 37;
# завершаем транзакцию
COMMIT;
```



# Итоги

---

# Итоги

В данной лекции мы:

- Узнали историю появления SQL;
- Разобрали MySQL по CAP и PACELC теоремам, узнали о механизмах хранения;
- Познакомились с DDL, DML, DCL и TCL запросами.





# Домашнее задание

---

## Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера .
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и  
пишите отзыв о лекции!**

**Роман Гордиенко**