САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Базы данных

Лекция 4
Основы SQL.
Операторы DDL

Агафонов Антон Александрович
д.т.н., доцент кафедры ГИиИБ

Самара

План лекции

- Основы языка SQL:
  - История возникновения
  - Основные возможности
  - Типы данных
  - Операторы
  - Базовые функции

- Операции определения данных (DDL)

Требование к языку

Требование к языку реляционных БД:

- Создание базы данных, таблиц и других объектов БД
- Выполнение основных операций редактирования данных в таблицах (вставка, модификация и удаление)
- Выполнение запросов пользователя к данным, преобразующих хранящиеся в таблицах данные в выходные отношения
- Работа с трехзначной логикой: `True / False / Unknown (NULL)`
- Декларативный тип языка

Стандарт SQL

SQL (Structured Query Language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

- Американский национальный институт стандартов (ANSI)
- Международная организация стандартизации (ISO)

Набор операций SQL:

- создание в базе данных новой таблицы;
- добавление в таблицу новых записей;
- изменение записей;
- удаление записей;
- выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
- изменение структур таблиц;
- описание и управление хранимыми объектами.

Стандарт SQL

Год	Название	Изменения
1986	SQL-86	Первый вариант стандарта, принятый институтом ANSI и одобренный ISO в 1987 году.
1989	SQL-89	Немного доработанный вариант предыдущего стандарта.
1992	SQL-92	Значительные изменения (ISO 9075).
1999	SQL:1999 / SQL-3	Добавлена поддержка регулярных выражений, рекурсивных запросов, поддержка триггеров, базовые процедурные расширения, нескалярные типы данных и некоторые объектно-ориентированные возможности.
2003	SQL:2003	Введены расширения для работы с XML-данными, оконные функции, генераторы последовательностей и основанные на них типы данных.
2006	SQL:2006	Функциональность работы с XML-данными значительно расширена. Появилась возможность совместно использовать в запросах SQL и XQuery.
2008	SQL:2008	Улучшены возможности оконных функций, устранены некоторые неоднозначности стандарта SQL:2003.
2011	SQL:2011	Реализована поддержка хронологических баз данных (PERIOD FOR), поддержка конструкции FETCH.
2016	SQL:2016	Защита на уровне строк, полиморфные табличные функции, JSON.

Достоинства SQL

- Независимость от конкретной СУБД
  Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую.

- Наличие стандартов
  Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка.

- Декларативность
  С помощью SQL описывается только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса.

Недостатки SQL

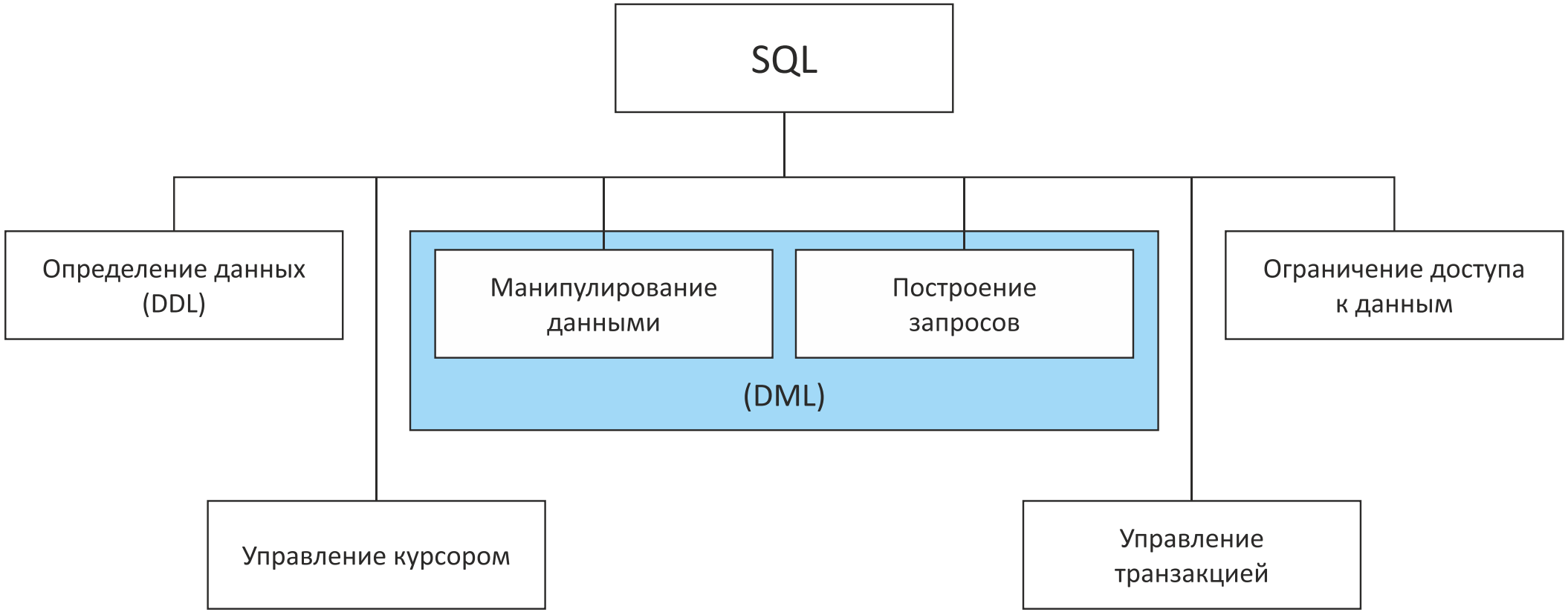
- Несоответствие реляционной модели данных:
  - допущение строк-дубликатов в таблицах и результатах выборок, что в рамках реляционной модели данных невозможно и недопустимо;
  - поддержка неопределённых значений (NULL), создающая фактически многозначную логику;
  - значимость порядка столбцов, возможность ссылок на столбцы по номерам (в реляционной модели столбцы должны быть равноправны);
  - допущение столбцов без имени, дублирующихся имён столбцов.

- Сложность

- Отступления от стандартов
  Появление диалектов языка SQL, специфичных для каждой конкретной СУБД.

- Сложность работы с иерархическими структурами

Возможности SQL



Возможности SQL

- **Определение данных** (DDL, Data Definition Language) – решение вопросов создания и удаления базы данных и ее объектов.
  `CREATE, ALTER, DROP`

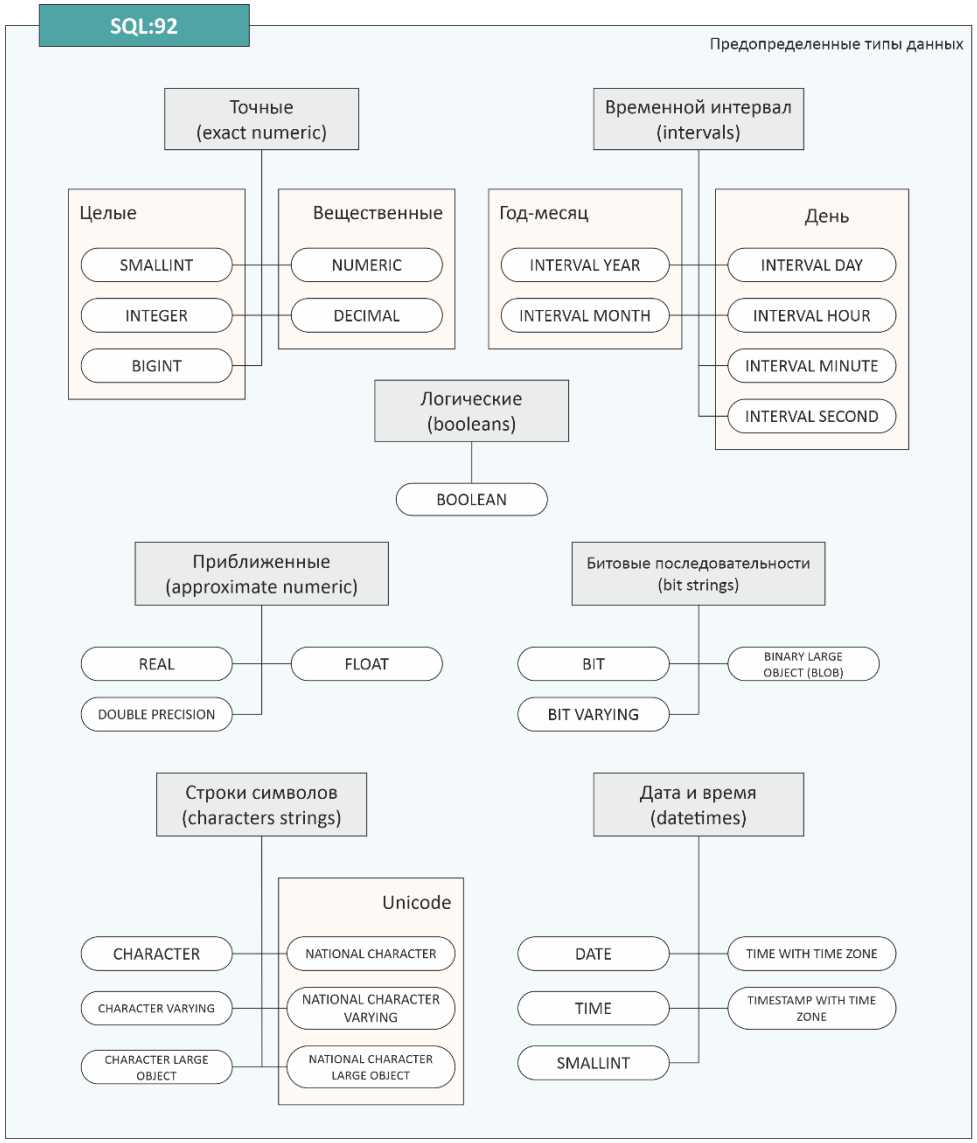
- **Манипулирование данными** (DML, Data Manipulation Language) – проведение операций вставки, редактирования и удаления данных из таблиц БД:
  - модификация данных, команды `INSERT, UPDATE, DELETE`
  - построение запросов – извлечение данных из одной или нескольких таблиц, команда `SELECT`

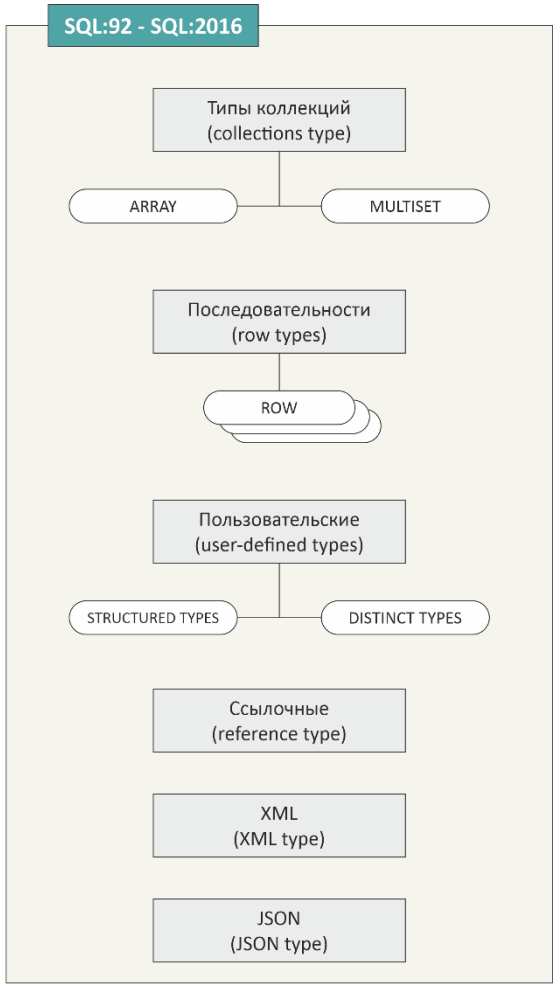
- **Ограничение доступа к данным** – определение набора прав пользователей при работе с объектами БД.
  `GRANT, REVOKE, DENY`

- **Управление курсором** – построчная обработки данных.
  `DECLARE CURSOR, OPEN CURSOR, FETCH CURSOR, CLOSE CURSOR`

- **Управление транзакцией** – определение уровня изоляции транзакции, запуск, фиксация или возвращение транзакции в исходное состояние.
  `SET TRANSACTION, BEGIN TRANSACTION, COMMIT, ROLLBACK`

Типы данных


SQL:92
Предопределенные типы данных
Точные (exact numeric)
Временной интервал (intervals)
Целые
SMALLINT
INTEGER
BIGINT
Вещественные
NUMERIC
DECIMAL
Год-месяц
INTERVAL YEAR
INTERVAL MONTH
День
INTERVAL DAY
INTERVAL HOUR
INTERVAL MINUTE
INTERVAL SECOND
Логические (booleans)
BOOLEAN
Приближенные (approximate numeric)
REAL
FLOAT
DOUBLE PRECISION
Битовые последовательности (bit strings)
BIT
BINARY LARGE OBJECT (BLOB)
BIT VARYING
Строки символов (characters strings)
CHARACTER
CHARACTER VARYING
CHARACTER LARGE OBJECT
Unicode
NATIONAL CHARACTER
NATIONAL CHARACTER VARYING
NATIONAL CHARACTER LARGE OBJECT
Дата и время (datetimes)
DATE
TIME
SMALLINT
TIME WITH TIME ZONE
TIMESTAMP WITH TIME ZONE


SQL:92 - SQL:2016
Типы коллекций (collections type)
ARRAY
MULTISET
Последовательности (row types)
ROW
Пользовательские (user-defined types)
STRUCTURED TYPES
DISTINCT TYPES
Ссылочные (reference type)
XML (XML type)
JSON (JSON type)

Точные числовые типы

Спецификация	Описание
NUMERIC [(n[,m])]	Точное число, описываемое аргументами n и m
DECIMAL [(n[,m])] или DEC [(n[,m])]	В отличие от NUMERIC, способно хранить число дальше с большей точностью, чем определено в аргументе m. Поэтому говорят, что NUMERIC задает реальное значение точности, а DECIMAL – минимальное значение точности
BIGINT	Тип данных, предназначен для хранения больших целых чисел (обычно 64 бит). Типы данных BIGINT, INTEGER и SMALLINT являются частным случаем типа данных NUMERIC, у которого масштаб установлен в 0, а точность определена возможностями СУБД
INTEGER или INT	Тип данных, предназначен для хранения целых чисел (обычно 32 бит)
SMALLINT	Тип данных, предназначен для хранения малых целых чисел (обычно 16 бит)

Целочисленные типы в MySQL

Тип	Размер	Минимальное со знаком	Минимальное без знака	Максимальное со знаком	Максимальное без знака
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-8388608	0	8388607	16777215
INT	4	-2147483648	0	2147483647	4294967295
BIGINT	8	-2^{63}	0	$2^{63}-1$	$2^{64}-1$

Приближенные числовые типы

Спецификация	Описание
REAL	Точность и предел значений зависят от СУБД. Как правило, занимает в памяти 6 байт и в состоянии хранить число в интервале от –3,4E – 38 до +3,4E + 38 с точностью до 7 цифр после запятой.
FLOAT [(n)]	Аргументом n определяется минимальное значение точности. Обычно подразумевается хранение чисел с одинарной точностью, 4 байта.
DOUBLE PRECISION	Точность определяется версией СУБД, превышает точность REAL (обычно, 8 байт). Тип данных способен хранить число в интервале от 1,7E – 308 до +1,7E – 308.

Таблица истинности AND

AND	False	True	Unknown
False	False	False	False
True	False	True	Unknown
Unknown	False	Unknown	Unknown

Таблица истинности OR

OR	False	True	Unknown
False	False	True	Unknown
True	True	True	True
Unknown	Unknown	True	Unknown

Таблица истинности NOT

Not	
False	True
True	False
Unknown	Unknown

Строковые типы данных

Спецификация	Описание
CHARACTER [n], или сокращенно CHAR[n]	Тип данных предназначен для создания текстовой строки фиксированной длины, дополненная пробелами. Количество символов в строке определяется в квадратных скобках после указания типа данных. Если в поле типа CHAR помещается текстовое значение меньшего размера, чем размерность поля, то оставшиеся позиции символов заполнятся пробелами.
CHARACTER VARYNG [n], или сокращенно VARCHAR[n]	Текстовая строка переменной длины. Максимальный размер строки определяется в квадратных скобках.
NCHAR[n], NCHAR VARYING [n]	Строки национального символьного набора (NATIONAL). При использовании следует указать спецификацию набора символов, воспользовавшись командой CHARACTER SET.
(NATIONAL) CHARACTER LARGE OBJECT [n], или CLOB[n]	Предназначен для определения столбцов таблиц, хранящих большие группы символов.

Битовые последовательности

Спецификация	Описание
`BIT [n]`	Битовая последовательность фиксированной длины. Аргумент n устанавливает длину последовательности в битах. Если аргумент отсутствует (или установлен в 1), то тип данных используется для создания полей логического типа (Да/Нет). Особенность типа данных фиксированной длины в том, что попытка записать в поле этого типа значения меньшей длины, чем указано в аргументе n, приведет к ошибке.
`BIT VARYNG [n]`	Битовая последовательность переменной длины. Максимальное значение битовой последовательности указывается в аргументе n.
`BINARY LARGE OBJECT [n]`, или сокращенно `BLOB [n]`	Тип данных предназначен для хранения больших объектов. Например, файлов мультимедиа и изображений.

Дата и время

Спецификация	Описание
DATE	Тип данных включает три поля: YEAR (год) – от 0001 до 9999; MONTH (месяц) – от 01 до 12; DAY (день) – от 01 до 31. Формат записи: «yyyy–mm–dd»
TIME [(n)]	Тип данных включает три поля: HOUR (часы), MINUTE (минуты), SECOND (секунды). Если аргумент точности (n) не определен, то полное число позиций (вместе с разделителями) равно 8. Формат записи: «hh:mm:ss».
TIMESTAMP [(n)]	Метка даты-времени, представляющая собой комбинацию типов данных DATE и TIME.
TIME WITH TIME ZONE	Тип данных аналогичен TIME плюс два дополнительных значения, характеризующих смещение от Гринвичского меридиана в часах TIMEZONE_HOUR и минутах TIMEZONE_MINUTE.
TIMESTAMP WITH TIME ZONE	Метка даты-времени плюс смещение от Гринвича.

Непредопределенные типы данных

- Массив: `тип_данных ARRAY [n];`
  Например, `INT ARRAY[10];`
- Множество и мультимножество: `тип_данных MULTISET;`
- Последовательность
- Пользовательский тип
- XML, JSON

Операторы

- Операция присваивания: «=», реже «:=»

- Арифметические операторы:
  - «+» сложения
  - «–» вычитания
  - «*» умножения
  - «/» деления
  - «DIV» целочисленного деления
  - «%» (MOD) остатка от деления

- Логические операторы:
  - AND – логическое умножение («И»)
  - OR – логическое сложение («ИЛИ»)
  - NOT – логическое отрицание («НЕ»)
  - XOR – исключающее «ИЛИ»

Операторы

- Операторы сравнения

Оператор	Операция
<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
=	Проверка равенства
<=>	Проверка равенства с учетом NULL
!=, <>	Проверка неравенства

Операторы

- Оператор проверки на неопределенность NULL

```
<значение> IS [NOT] NULL

SELECT NULL IS NULL, 1 IS NULL;
->1  0
```

- Функция проверки на неопределенность ISNULL

```
SELECT ISNULL(NULL), ISNULL(1);
->1  0
```

Встроенные функции

Функция	Описание
BIT_LENGTH(битовая строка)	Возвращает длину строки в битах
CAST(значение AS тип данных)	Функция преобразования исходного значения к новому типу данных
CHAR_LENGTH(символьная строка)	Возвращает длину строки в символах
CURRENT_DATE	Возвращает текущую дату
CURRENT_TIME(точность)	Возвращает текущее время с указанной точностью
CURRENT_TIMESTAMP(точность)	Возвращает текущую дату и время с указанной точностью
LOWER(строка)	Преобразование текстовой строки к нижнему регистру
POSITION(подстрока IN строка)	Возвращает позицию, с которой начинается вхождение подстроки в строку
SUBSTRING(строка FROM n FOR длина)	Возвращает часть строки, начиная с n-го символа с указанной длиной
TRANSLATE(строка USING функция)	Преобразование строки с использованием указанной функции
TRIM(LEADING | TRAILING | BOTH символ FROM строка)	Удаление из строки всех первых (LEADING), последних (TRAILING) или первых и последних (BOTH) символов
UPPER(строка)	Преобразование текстовой строки к верхнему регистру

Операторы определения данных (DDL)

Операторы определения данных (Data Definition Language, DDL):

- **CREATE** создаёт объект базы данных (саму базу, таблицу, представление, пользователя и т.д.)
- **ALTER** изменяет объект
- **DROP** удаляет объект

Создание базы данных

- Создание БД

```
CREATE DATABASE [IF NOT EXISTS] <имя_базы_даных>;

CREATE DATABASE <имя_базы_даных>;    # операция возвратит ошибку, если БД уже существует
CREATE DATABASE IF NOT EXISTS <имя_базы_даных>;  # попытка создания БД,
                                                 если БД с таким именем не существует
```

Пример:

```
CREATE DATABASE productsdb;
```

- Стандарт SQL создания БД

```
CREATE SCHEMA <имя_схемы> [AUTHORIZATION имя владельца]
    [DEFAULT CHARTERSET набор символов по умолчанию]
    [PATH <символьный путь к файлам БД> ]
    [ <дополнительные инструкции>... ]
```

Выбор и удаление базы данных

- Выбор БД

```
USE <имя_базы_даных>;
```

- Удаление БД

```
DROP DATABASE [IF EXISTS] <имя_базы_даных>;
```

```
DROP DATABASE <имя_базы_даных>;    # операция возвратит ошибку, если БД не существует
DROP DATABASE IF EXISTS <имя_базы_даных>;  # попытка удаления БД,
                                           если БД с таким именем существует
```

Пример:

```
DROP DATABASE productsdb;
```

- Стандарт SQL удаления БД

```
DROP SCHEMA <имя_схемы> [CASCADE|RESTRICT];
```

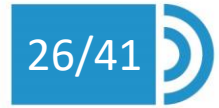

Создание таблицы

```
CREATE TABLE <название_таблицы> (
  название_столбца1  <тип_данных>  <атрибуты_столбца1>,
  название_столбца2  <тип_данных>  <атрибуты_столбца2>,
    ...
  название_столбцаN  <тип_данных>  <атрибуты_столбцаN>,
  атрибуты_уровня_таблицы
)
```

Пример:

```
CREATE DATABASE productsdb;
USE productsdb;

CREATE TABLE customers (
    id INT,
    age INT,
    first_name VARCHAR(20),
    last_name VARCHAR(20)
);
```

Атрибуты столбцов

Ключ	Описание
NOT NULL	Запрет на вставку в столбец неопределенного значения NULL.
UNIQUE	Значение столбца должно быть уникальным.
AUTO_INCREMENT	Значение столбца будет автоматически увеличиваться при добавлении новой строки.
DEFAULT	Определяет значение по умолчанию для столбца.
PRIMARY KEY	Признак первичного ключа. Значение поля должно быть уникальным, оно не может содержать NULL, в таблице это ограничение может использоваться только один раз.
CHECK	Ограничение-проверка на допустимое значение. В скобках за оператором CHECK указывается предикат, проверяющий допустимость значения.
FOREIGN KEY	Ограничение внешнего ключа для таблицы. Ограничения внешнего ключа требуют, чтобы все значения, присутствующие во внешнем ключе, соответствовали значениям родительского ключа (обеспечение ссылочной целостности)

Пример задания атрибутов

```sql
CREATE TABLE customers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    age INT DEFAULT 18 CHECK(age > 0 AND age < 100),
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    email VARCHAR(30) NULL,
    phone VARCHAR(20) UNIQUE,
    CHECK ((email != '') AND (phone != ''))
);
```


Первичный ключ

Первичный ключ на уровне столбца

```
CREATE TABLE customers (
    id INT PRIMARY KEY,
    age INT,
    first_name VARCHAR(20),
    last_name VARCHAR(20)
);
```

Составной первичный ключ

```
CREATE TABLE orders (
    order_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10, 2),
    PRIMARY KEY (order_id, product_id)
);
```

Первичный ключ на уровне таблицы

```
CREATE TABLE customers (
    id INT,
    age INT,
    first_name VARCHAR(20),
    last_name VARCHAR(20),
    PRIMARY KEY (id)
);
```

Установка имени ограничения

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT,
    age INT,
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    email VARCHAR(30),
    phone VARCHAR(20) NOT NULL,
    CONSTRAINT customers_pk PRIMARY KEY(id),
    CONSTRAINT customer_phone_unique UNIQUE(phone),
    CONSTRAINT customer_age_check CHECK(age > 0 AND age < 100)
);
```


Внешние ключи

```
[CONSTRAINT <имя_ограничения>]
FOREIGN KEY (<столбец1>, <столбец2>, ..., <столбецN>)
REFERENCES <главная_таблица> (<столбец_главной_таблицы1>,
       <столбец_главной_таблицы2>, ..., <столбец_главной_таблицыN>)
[ON DELETE <действие>]
[ON UPDATE <действие>]
```

Внешние ключи

Главная таблица

```
CREATE TABLE customers (
    id INT PRIMARY KEY AUTO_INCREMENT,
    age INT,
    first_name VARCHAR(20),
    last_name VARCHAR(20)
);
```

Зависимая таблица

```
CREATE TABLE orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    created_at DATE,
    FOREIGN KEY (customer_id)
        REFERENCES customers (id)
);
```

Именованный внешний ключ

```
CREATE TABLE orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    created_at DATE,
    CONSTRAINT orders_customers_fk FOREIGN KEY (customer_id) REFERENCES customers (id)
);
```

ON DELETE / ON UPDATE

Ключ	Описание
CASCADE	Изменение значения первичного ключа приводит к автоматическому изменению соответствующих значений внешнего ключа.
SET NULL	При изменении значения или удалении первичного ключа все значения в связанных с внешним ключом колонках устанавливаются в NULL. В результате в дочерней таблице появляются «брошенные» строки, утерявшие связь с соответствующей записью из главной таблицы.
RESTRICT	Отклоняет удаление или изменение строк в главной таблице при наличии связанных строк в зависимой таблице.
NO ACTION	То же самое, что и RESTRICT.
SET DEFAULT	При удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT.

Каскадное удаление

```sql
CREATE TABLE orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    customer_id INT,
    created_at DATE,
    FOREIGN KEY (customer_id) REFERENCES customers (id) ON DELETE CASCADE
);
```

Модификация таблицы

```
ALTER TABLE <название_таблицы>
{
  ADD <название_столбца> <тип_данных_столбца> [<атрибуты_столбца>] |
  DROP COLUMN <название_столбца> RESTRICT | CASCADE |
  MODIFY COLUMN <название_столбца> <тип_данных_столбца> [<атрибуты_столбца>] |
  ALTER COLUMN <название_столбца> SET DEFAULT <значение_по_умолчанию> |
  ADD [CONSTRAINT] <определение_ограничения> |
  DROP [CONSTRAINT] <имя_ограничения> RESTRICT | CASCADE
}
```


Модификация таблицы

Ключ	Описание
ADD [COLUMN]	Добавляет в таблицу новый столбец. Новый столбец определяется так же, как в операторе CREATE TABLE.
ALTER [COLUMN]	Используется для создания или отмены значения по умолчанию для столбца.
DROP [COLUMN]	Удаляет из таблицы столбец. При использовании параметра RESTRICT перед удалением столбца СУБД проверит наличие ссылок на него из других таблиц и представлений. Если таковые имеются, то столбец не будет удален. Наоборот, при использовании параметра CASCADE вместе со столбцом будут удалены все объекты, ссылающиеся на него.
ADD CONSTRAINT	Позволяет добавить к таблице новое ограничение.
DROP CONSTRAINT	Удаляет уже существующие ограничения. Если в предложении задан параметр RESTRICT, то в этот момент столбец не должен использоваться как родительский ключ для внешнего ключа другой таблицы. Если передается параметр CASCADE, то внешние ключи, имеющие ссылки или ограничения FOREIGN KEY, уничтожаются.

Модификация таблицы

- Добавление нового столбца

```
ALTER TABLE customers ADD address VARCHAR(50) NULL;
```

- Удаление столбца

```
ALTER TABLE customers DROP COLUMN address;
```

- Изменение значения по умолчанию

```
ALTER TABLE customers ALTER COLUMN age SET DEFAULT 22;
```

- Изменение типа столбца

```
ALTER TABLE customers MODIFY COLUMN first_name CHAR(100) NULL;
```

Модификация таблицы

- Добавление внешнего ключа

```
ALTER TABLE orders ADD FOREIGN KEY (customer_id) REFERENCES customers (id);

ALTER TABLE orders ADD CONSTRAINT orders_customers_fk
FOREIGN KEY (customer_id) REFERENCES customers (id);
```

- Удаление внешнего ключа

```
ALTER TABLE orders DROP FOREIGN KEY orders_customers_fk;
```

- Добавление первичного ключа

```
ALTER TABLE products ADD PRIMARY KEY (id);
```

- Удаление первичного ключа

```
ALTER TABLE orders DROP PRIMARY KEY;
```


Удаление данных. Удаление таблицы

- Очистка таблицы

```
TRUNCATE TABLE <имя_таблицы>;
```

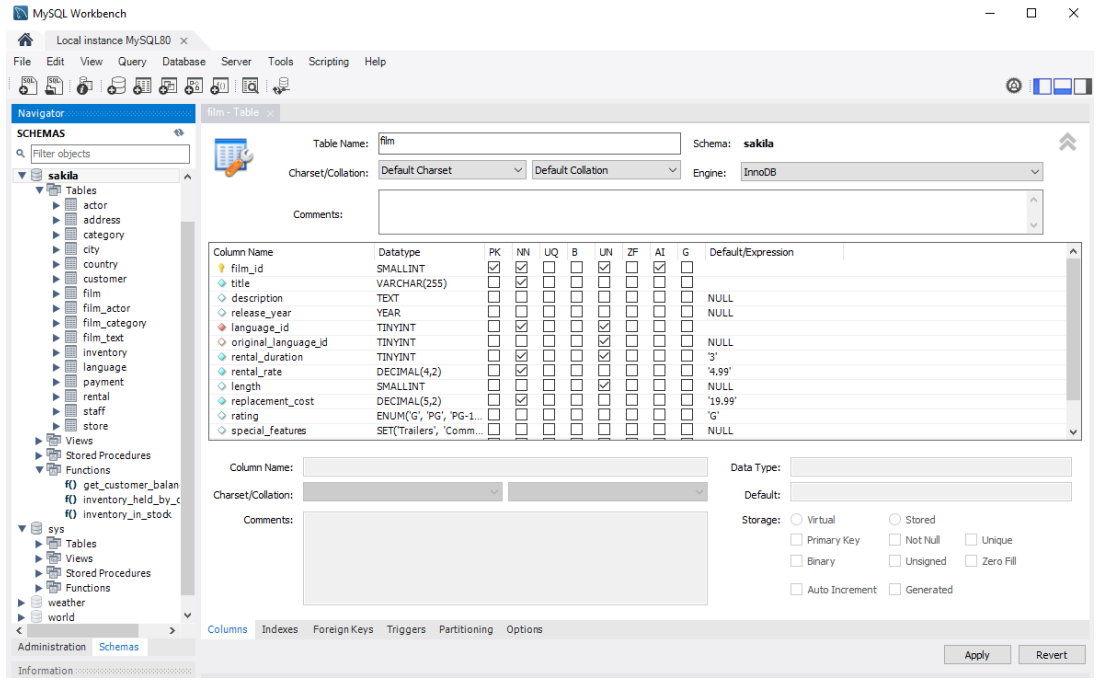
- Удаление таблицы

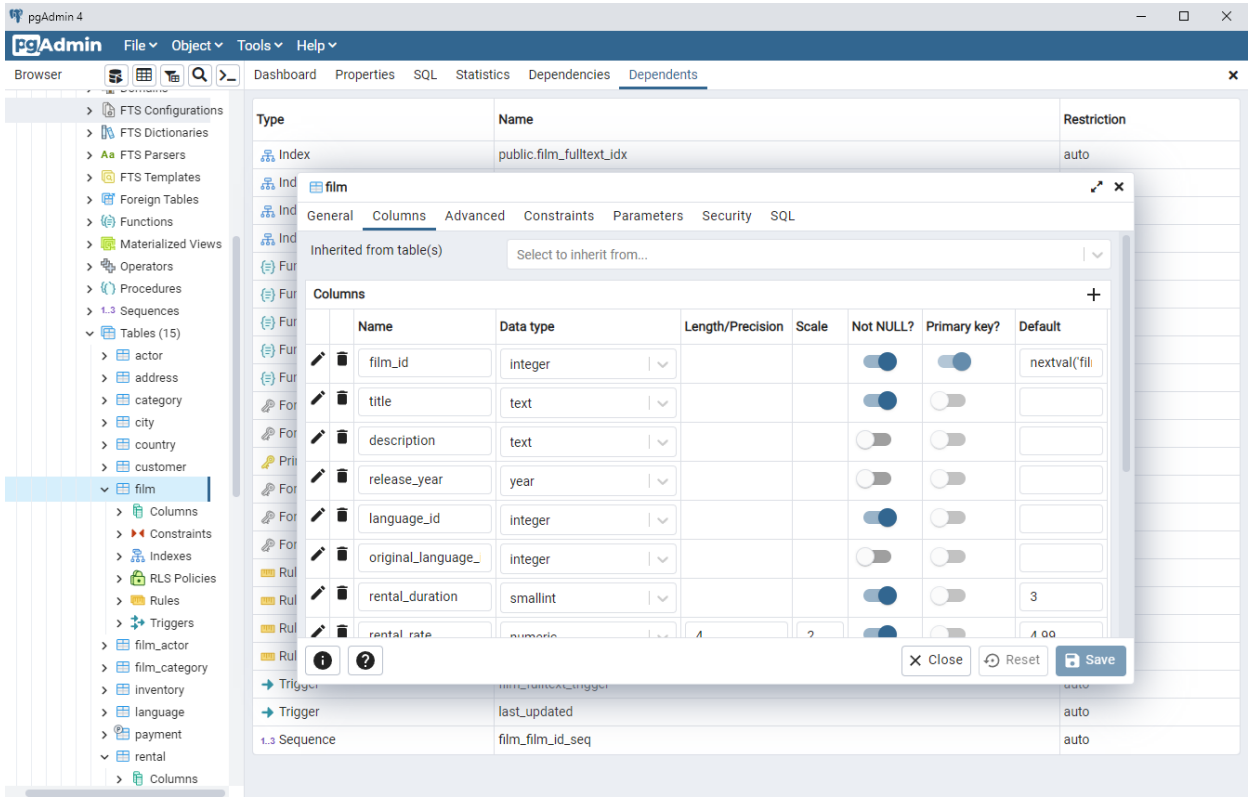
```
DROP TABLE <имя_таблицы>;
```

Пример:

```
DROP TABLE customers;
```

Визуальные средства проектирования





САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
д.т.н., доцент кафедры ГИиИБ