База Данных

Лекция 5

SQL
Язык структурированных запросов

Этапы проектирования базы данных

- Анализ предметной области
- Концептуальный дизайн
- Логический дизайн
- Физический дизайн

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.

- Произносится как «**Эскуэль/ЭсКьюЭль**», реже «**СиКуЭль/СиКьюЭль**», но чаще всего можно услышать жаргонное «**Сиквэл/Сиквел**».

- Команды SQL НЕ чувствительны к регистру. Тем не менее, все ключевые слова (команды) SQL принято писать заглавными буквами.
- Большинство систем баз данных требуют точку с запятой (;) в конце каждого оператора SQL. Точка с запятой — это стандартный способ разделения каждого оператора SQL, который позволяет выполнять более одного оператора SQL за один вызов сервера.

Для чего нужен SQL на конкретном примере

Чтобы непрофессионалу понять, **что значит SQL для ИТ-отрасли**, приведём простой пример.

Представьте таблицу с информацией о студентах: *имена, возраст, предмет обучения и так далее*. В ней есть определённое количество **строк** и **столбцов**. Один из рядов содержит **успеваемость студентов**.

Как только все данные будут внесены в таблицу, каждая из записей попадает в разные категории (**столбцы** или «**аттрибуты**»). Это и есть **организованная база данных**. Вся организованная внутри неё информация, которой можно управлять, называется **Database Schema** (схема данных).

Если вы захотите **выдать стипендии** учащимся, которые получают **оценку 90%** или выше, то выполняется **запрос данных в SQL**, что простыми словами значит «*попросить базу данных предоставить информацию о студентах, получающих 90% и более баллов*».

Команда будет иметь синтаксический вид:

```
SELECT * FROM Student WHERE Percentage>=90;
```

Когда количество данных мало (скажем, **10 студентов**), то можно всё легко посчитать и написать **на клочке бумаге**. Но когда объём данных увеличивается **до тысяч записей**, становится **нужен SQL** — он помогает управлять огромными данными эффективно, то есть быстро получать расчёты на их основе.

SQL-операторы

Работать с данными помогают операторы — определенные слова или символы, которые используются для выполнения конкретной операции.

В языке SQL традиционно выделяются

- группа операторов определения данных (Data Definition Language — **DDL**),
- группа операторов манипулирования данными (Data Manipulation Language — **DML**) и
- группа операторов, управляющих привилегиями доступа к объектам базы данных (Data Control Language — **DCL**).

SQL DDL

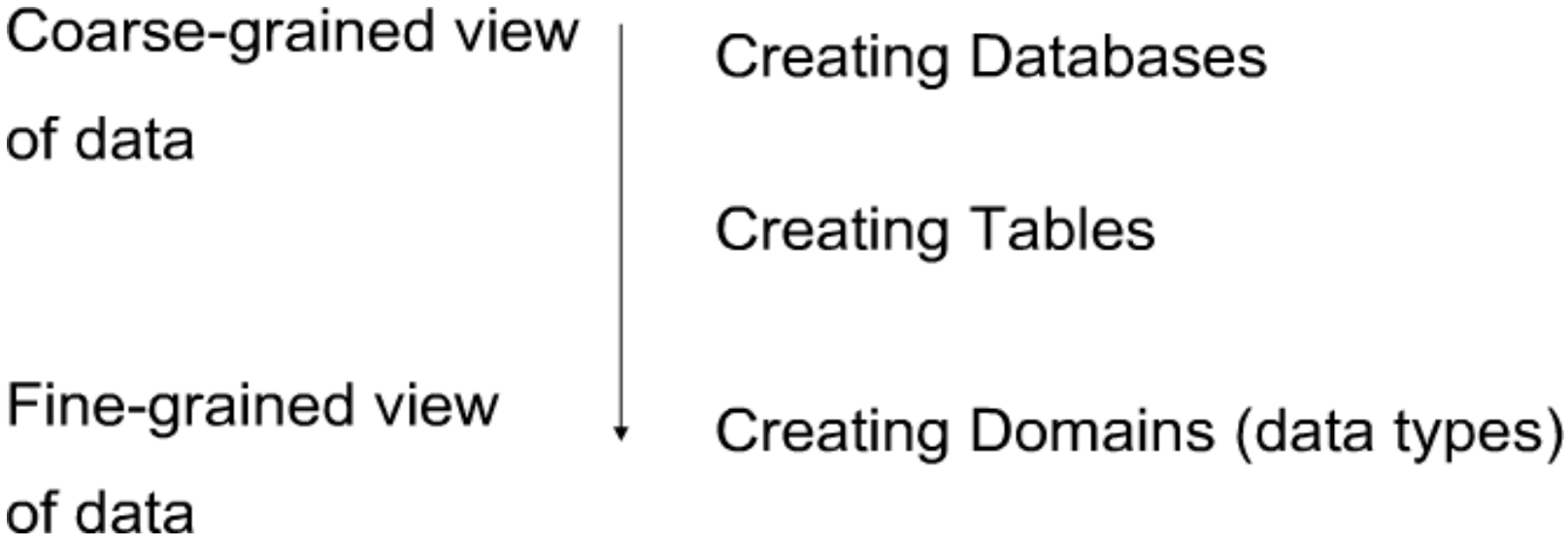
К операторам языка определения данных (DDL) относятся команды для создания, изменения и удаления таблиц, представлений и других объектов базы данных:

CREATE DATABASE — создает базу данных.
CREATE TABLE — создает новую таблицу внутри базы данных.
DROP DATABASE — удаляет базу данных.
DROP TABLE — удаляет таблицу в базе данных.
ALTER TABLE — изменяет таблицу в базе данных.

Примечание: будет использоваться диалект SQL, поддерживаемый PostgreSQL.

Представление SQL DDL сверху вниз

- На "вершине" создается база данных
- Далее по иерархии создается набор таблиц
- В нижней части иерархии создаются типы данных



Создание базы данных

В PostgreSQL есть команда **CREATEDB**, которая создает базу данных.

Синтаксис:
CREATE DATABASE dbname;

где dbname – название создаваемой Базы данных

Создание таблицы (CREATE TABLE)

Оператор CREATE TABLE позволяет создать пустую таблицу в текущей базе данных.

Синтаксис:

```
CREATE TABLE table_name
( column1_name datatype,
column2_name datatype,
.....
columnN_name datatype,
PRIMARY KEY(column_name(s))
);
```

Создание таблицы (CREATE TABLE): пример

```
CREATE TABLE Groups
(
    group_id  int,
    group_name varchar(15),
    PRIMARY KEY (group_id)
);
```

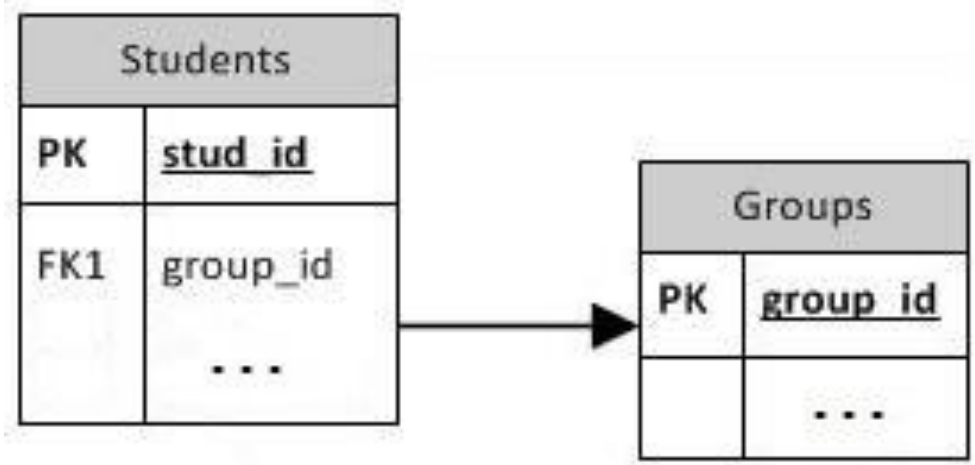
или

```
CREATE TABLE Groups
(
    group_id int PRIMARY KEY ,
    group_name varchar(15)
);
```

Создание таблицы (CREATE TABLE): пример связи таблиц с помощью FK

```
CREATE TABLE Groups
(
      group_id  int,
      group_name
      varchar(15),
      PRIMARY KEY (group_id)
      );

CREATE TABLE Students
(
stud_id int,
first_name    varchar(20),
last_name    varchar(20),
group_id int,
PRIMARY KEY (stud_id),
FOREIGN KEY (group_id) REFERENCES Groups(group_id)
);
```



FK в Students.

PK в Groups

Создание таблицы (CREATE TABLE): пример связи таблиц с помощью FK

```
CREATE TABLE Students
(
    stud_id int PRIMARY KEY,
    first_name varchar(20),
    last_name varchar(20),
    group_id int,
    FOREIGN KEY (group_id) REFERENCES Groups(group_id)
);
```

or

```
CREATE TABLE Students
(
    stud_id int PRIMARY KEY,
    first_name varchar(20),
    last_name varchar(20),
    group_id int REFERENCES Groups(group_id)
);
```

Ограничения (constraints)

Ограничения в SQL определяются для таблиц и столбцов и позволяют задавать правила на добавление, изменение или удаление данных.

- **PRIMARY KEY** — ограничение, которое определяет уникальность идентификатора записи в таблице.
- **FOREIGN KEY** — ограничение, которое задает связь между таблицами через внешний ключ.
- **CHECK** — ограничение, которое проверяет соответствие значения в столбце определенному условию.
- **NOT NULL** — ограничение, которое обязывает значение в столбце быть непустым.
- **UNIQUE** — ограничение, которое запрещает дублирование значений в столбце.

Ограничение CHECK

- Ограничение CHECK гарантирует, что все значения в столбце удовлетворяют определенному условию.
- CHECK включает условие для проверки значения, введенного в запись. Если условие оценивается как ложное, запись нарушает ограничение и не вводится в таблицу.

- Пример:
- CREATE TABLE Students(
- stud_id int PRIMARY KEY,
- first_name varchar(20),
- last_name varchar(20),
  gpa int **CHECK (gpa <= 4)**
-);

Ограничение NOT NULL

- По умолчанию столбец может содержать значения NULL. Если вы не хотите, чтобы столбец имел значение NULL, необходимо определить ограничение NOT NULL.
- NOT NULL гарантирует наличие значений во всех строках данного столбца.
- PK по умолчанию имеют ограничение NOT NULL.

- Пример:
- CREATE TABLE Students(
  stud_id int PRIMARY KEY, first_name varchar(20),
  last_name varchar(20) **NOT NULL**
-);

Ограничения Unique

- Ограничения Unique гарантируют, что значения в столбцах уникальны.
- РК по умолчанию являются уникальными.

- Пример:
- CREATE TABLE Groups(
  group_id int PRIMARY KEY,
- group_name varchar(15) **UNIQUE**
-);
- Или
- CREATE TABLE Groups(
- group_id int PRIMARY KEY,
- group_name varchar(15),
- **UNIQUE (group_name)**
-);

Типы данных

При создании таблицы для каждого столбца необходимо указываете тип данных, т. е. какие данные будут храниться в полях таблицы.

PostgreSQL поддерживает широкий набор типов данных. Есть основные категории:

- Числовой
- Символьный
- Дата/время
- Логический

Числовые типы

Имя	Размер	Описание	Диапазон
smallint	2 байта	целое в небольшом диапазоне	-32768 .. +32767
integer	4 байта	типичный выбор для це- лых чисел	-2147483648 .. +2147483647
bigint	8 байт	целое в большом диапазо- не	-9223372036854775808 .. 9223372036854775807
decimal	переменный	вещественное число с указанной точностью	до 131072 цифр до деся- тичной точки и до 16383 — после
numeric	переменный	вещественное число с указанной точностью	до 131072 цифр до деся- тичной точки и до 16383 — после
real	4 байта	вещественное число с переменной точностью	точность в пределах 6 де- сятичных цифр
double precision	8 байт	вещественное число с переменной точностью	точность в пределах 15 де- сятичных цифр
smallserial	2 байта	небольшое целое с автоувеличением	1 .. 32767
serial	4 байта	целое с автоувеличением	1 .. 2147483647
bigserial	8 байт	большое целое с автоувеличением	1 .. 9223372036854775807

Целочисленные типы

Типы **smallint, integer и bigint** хранят целые числа, то есть числа без дробной части, имеющие разные допустимые диапазоны. Попытка сохранить значение, выходящее за рамки диапазона, приведёт к ошибке.

Чаще всего используется тип integer, как наиболее сбалансированный выбор ширины диапазона, размера и быстродействия.

В SQL определены только типы integer (или int), smallint и bigint.

Числа с произвольной точностью

Тип **numeric** позволяет хранить числа с очень большим количеством цифр. Он особенно рекомендуется для хранения денежных сумм и других величин, где важна точность.
Для столбца типа numeric можно настроить и максимальную точность, и максимальный масштаб.
Столбец типа numeric объявляется следующим образом:

NUMERIC(точность, масштаб)

Например, число 23.5141 имеет точность 6 и масштаб 4.
Целочисленные значения можно считать числами с масштабом 0.

Символьные типы

- Символьные строки – это последовательности печатаемых символов

Имя	Описание
character varying(n), varchar(n)	строка ограниченной переменной длины
character(n), char(n), bpchar(n)	строка фиксированной длины, дополненная пробелами
text	строка неограниченной переменной длины

- Для char(n) и varchar(n) n — это длина строки.

- Все символьные строки в SQL начинаются и заканчиваются одинарными кавычками. Например, ‘string’ является допустимой строкой SQL.

Типы даты/времени

Имя	Описание	Пример	Наименьшее значение	Наибольшее значение
timestamp	дата и время (без часового пояса)	2020-01-01 08:00:00	4713 до н.э.	294276 н.э.
timestamptz	дата и время (с часовым поя- сом)	2020-01-01 08:00:00+6	4713 до н.э.	294276 н.э.
date	дата (без вре- мени суток)	2020-01-01	4713 до н.э.	5874897 н.э.
time	время суток (без даты)	08:00:00	00:00:00	24:00:00
time with time zone	время дня (без даты), с часо- вым поясом	08:00:00+6	00:00:00+12	23:59:59-12
interval	временной ин- тервал	1 12:10:10 (read as 1 day, 12 hours, 10 minutes, 10 seconds)	-178000000 лет	178000000 лет

Значения даты и времени в SQL также начинаются и заканчиваются одинарными кавычками.

Логический тип данных

- PostgreSQL (и большинство других диалектов SQL) поддерживают тип данных boolean.

Имя	**Размер**	**Описание**
boolean	1 байт	состояние: истина или ложь

- Тип boolean может иметь следующие состояния: «true», «false» и третье состояние, «unknown», которое представляется SQL- значением NULL.

Состояние	**Формы представления состояния**
true	TRUE, ‘t’, ‘true’, ‘y’, ‘yes’, ‘1’
false	FALSE, ‘f’, ‘false’, ‘n’, ‘no’, ‘0’

ALTER TABLE Изменение таблицы

- Когда вы создаете таблицу и понимаете, что допустили ошибку, или требования приложения меняются, вы можете удалить таблицу и создать ее заново. Но это неудобный вариант, если таблица уже заполнена данными или если на таблицу ссылаются другие объекты базы данных (например, ограничение внешнего ключа). Поэтому PostgreSQL предоставляет семейство команд для внесения изменений в существующие таблицы.

- Команда **ALTER TABLE** используется для изменения структуры существующей таблицы.

ALTER TABLE Изменение таблицы

Синтаксис:

```
ALTER TABLE table_name ...;
```

Возможные модификации:

- Добавление / удаление столбцов
- Добавление / удаление ограничений
- Изменение типов данных столбцов
- Переименование столбцов / таблиц и т.д.

ADD COLUMN Добавить столбец

Синтаксис:

ALTER TABLE table_name ADD COLUMN column_name datatype [constraint];

Предположим, мы хотим добавить столбец адрес в существующую таблицу Студенты.

ALTER TABLE Students ADD COLUMN adress varchar(25);

Добавить столбец с ограничением:

ALTER TABLE Students ADD COLUMN adress varchar(25) NOT NULL;

DROP COLUMN Удаление столбца

- Удаление столбца: оператор DROP COLUMN используется с командой ALTER

- Синтаксис следующий

  ALTER TABLE table_name DROP COLUMN column_name;

- Итак, чтобы удалить столбец адрес, пишем:

  ALTER TABLE Students DROP COLUMN adress;

RENAME column/table Переименование столбцов/таблицы

Переименование столбца :
ALTER TABLE table_name RENAME COLUMN old_column_name TO new_column_name ;

Переименование столбца адрес в таблоице *Students* на контакт:
ALTER TABLE Students RENAME COLUMN adres TO contact;

Переименование таблицы:
ALTER TABLE old_table_name RENAME TO new_table_name ;

Переименование таблицы *Students* в таблицу *Students_info*:
ALTER TABLE Students RENAME TO Students_info;

TYPE Изменение Типа данных

Чтобы изменить ТИП данных столбца :
ALTER TABLE table_name ALTER COLUMN column_name TYPE new_datatype;

Предположим, чтобы изменить тип данных на varchar для столбца phone_number. :
ALTER TABLE Students ALTER COLUMN phone_number TYPE varchar(15);

Примечание: обратитесь к документации, чтобы уточнить возможность изменения типов данных.

ADD FOREIGN KEY Добавить внешний ключ

SQL DDL также позволяет нам добавлять ограничения к таблицам с помощью команды ALTER TABLE. Мы можем добавить ограничения PK, FK, UNIQUE, NOT NULL, CHEK.

Синтаксис:

```
ALTER TABLE child_table_name ADD FOREIGN KEY
(FK_column_name) REFERENCES parent_table_name
(PK_column_name);
```

Чтобы добавить связь FK для таблиц «Студенты» и «Группы» (на этом этапе столбец group_id в таблице «Студенты» уже должен существовать):

```
ALTER TABLE Students ADD FOREIGN KEY (group_id)
REFERENCES Groups (group_id);
```

SET and DROP NOT NULL - Добавлять и удалять NOT NULL

- Чтобы добавить ограничение NOT NULL к столбцу (на этом этапе столбец в таблице уже должен существовать):
- ALTER TABLE имя_таблицы ALTER COLUMN имя_столбца SET NOT NULL;

- Удаление ограничения NOT NULL
- ALTER TABLE имя_таблицы ALTER COLUMN имя_столбца DROP NOT NULL;

DROP TABLE - Удаление таблиц

Команда DROP TABLE удаляет таблицу из базы данных.

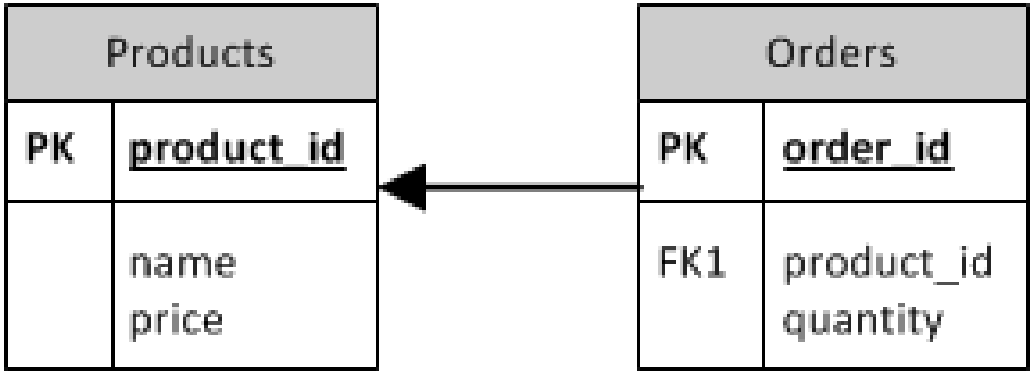
Синтаксис:
DROP TABLE [IF EXISTS] table_name [, ...] [CASCADE | RESTRICT]

IF EXISTS Не выдавать ошибку, если таблица не существует. В ином случае выдается уведомление.
CASCADE Автоматически удалять объекты, связанные с таблицей.
RESTRICT вместо CASCADE определяет поведение по умолчанию: предотвращает удаление объектов, от которых зависят другие объекты.

DROP TABLE удаляет все данные и ограничения для таблицы. Однако, чтобы удалить таблицу, на которую ссылается FK другой таблицы, необходимо указать CASCADE (CASCADE удаляет ограничение FK, а не другую таблицу полностью).

Каскадное Удаление таблиц (CASCADE)

Таблицы: Продукты, заказы (ссылки на продукты)



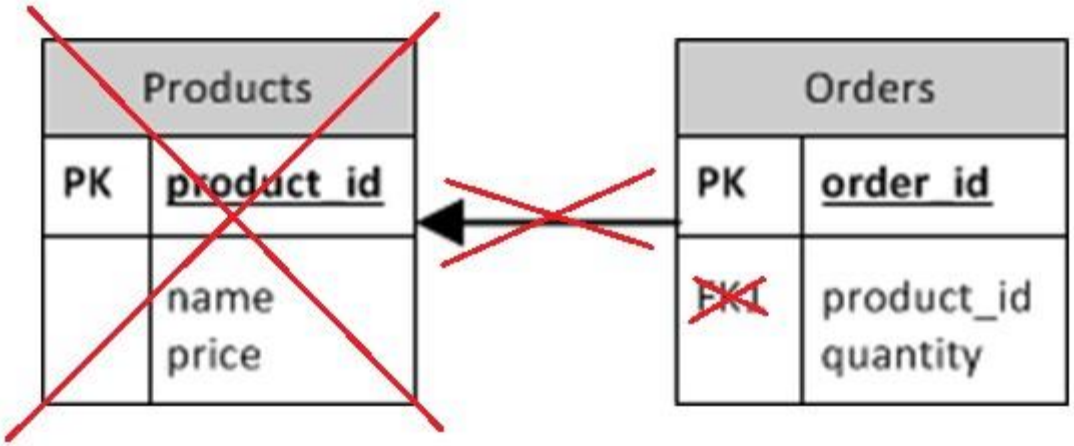

```
DROP TABLE products;
```

ОБРАТИТЕ ВНИМАНИЕ: ограничение orders_product_id_key в таблице Orders зависит от таблицы Products
ОШИБКА: не удается удалить продукты таблицы, потому что от этого зависят другие объекты
ПОДСКАЗКА: Используйте DROP ... CASCADE для удаления зависимых объектов тоже.

Удаление таблиц с каскад (CASCADE)

DROP TABLE products CASCADE;

- В этом случае команда не удаляет таблицу заказов, а только ограничение внешнего ключа.



- Ключевое слово RESTRICT вместо CASCADE определяет поведение по умолчанию: предотвращает удаление объектов, от которых зависят другие объекты.