

Евгений Викторович

УСТАНОВИТЬ:

- PostgreSQL
- PgAdmin

## Лекция 1

Реляционная БД:

- Данные распределены по смыслу по таблицам
- Между таблицами есть отношения

Одна таблица - одна сущность предметной области

**Столбцы** - характеристики сущности (фиксированное кол-во)

**Строки** - непосредственно экземпляры (неограниченное кол-во)

Столбцы обязательно имеют **тип данных**:

- Числовые
- Символьные
- Для работы с датами
- Бинарные
- Логические
- Прочие

*Также можно создавать свои типы данных*

-- комментарий

## Конструкция SELECT

```
-- Основное
select список_столбцов
from имя_таблицы
-- Доп.
where условия_на_выборку
order by список_столбцов -- Сортировка
```

```
-- Example
select id_comp, name from company
```

## Проекция отношений

Необязательно название, можно выражение (в пределах запроса):

```
select Price * ProductCount from Products;
```

Также можно изменить название выходного столбца или задать псевдоним с помощью AS

```
select ProductCount as Title, Price * ProductCount as totalsum
```

## Условие WHERE

```
select * from Products where Price > 100
```

## Функции like и ilike

- like 'text' - с учётом регистра
- ilike 'text' - без учёта регистра
- \_ - один произвольный символ
- % - произвольный набор символов, в том числе пустой

## Пустое значение NULL

Значение в ячейке отсутствует

Для проверки на пустое значение используется IS :

```
is null
is not null -- не пустая
```

Любые операции со значением NULL возвращают NULL !!!

a = NULL - всегда возвращает False

## Сортировка ORDER BY

```
select * from company order by name
```

Для обратной сортировки desc (действует на поле)

```
select * from company order by name, age desc
-- name в правильном порядке, age в обратном
```

## Ограничение вывода `limit`

```
select * from company limit 3
-- будет выведено только 3 строки
```

## Уникальные записи

Для выборки уникальных записей используется ключевое слово `distinct`

```
select distinct place from pass_in_trip order by 1 -- 1 - порядковый номер
столбца в select (то есть place)
```

`order by 1` - плохая практика



## Объединение запросов `union`

```
select name from company
union
select name from passenger
```

Просто `union` удаляет повторяющиеся записи (накладывает `distinct`)

Для вывода всех записей `union all`

```
select name from company
union all
select name from passenger
```

Оба запроса должны выдавать результаты в одинаковой формате (количество и типы полей)

## Практика 2

RgAdmin обращается к серверу только во время запросов

## Лекция 2

## Создание/удаление объектов

- DDL - язык определения данных
- DML - язык обработки данных

Создание БД:

```
create database <name>
```

Удаление БД (необратимый процесс):

```
drop database <name>
```

Эти конструкции выполняются на уровне кластера, все остальные(ниже) на уровне БД

**Схема** - некоторая возможность логически разделить объекты в БД, одноуровневые(схема в схеме невозможна). Схема по умолчанию: `public`

```
create schema <name>
drop schema <name>
```

## Создание/удаление таблиц

Для создания:

```
create table <name_table>
(name_columnn_1, type_data, attributes_column_1,
 имя_столбца_2, тип_данных, атрибуты_столбца_2,
);
```

Пример:

```
create table customers
(
    Id SERIAL PRIMARY KEY,
    FirstName CHARACTER VARYING(30),
    LastName CHARACTER VARYING(30),
    Age INTEGER
);
```

Удаление:

```
drop table <name>
```

# PRIMARY KEY

PRIMARY KEY - **первичный ключ**, уникально идентифицируют строку в таблице  
Варианты создания первичного ключа

```
create table Cusomers
(
    Id SERIAL PRIMARY KEY,
    ...
)
```

```
create table Cusomers
(
    Id SERIAL,
    ...,
    PRIMARY KEY(Id)
)
```

В качестве первичного ключа может быть любой тип данных, главное чтобы было уникально.

SERIAL - создаёт поле int и генерирует последовательные номера (создаёт объект последовательности)

Составной первичный ключ

```
create table Cusomers
(
    Id SERIAL,
    Num INTEGER
    ...,
    PRIMARY KEY (Id, Num)
)
```

UNIQUE - также уникальные значения, но не возможно связывать с другими таблицами

## NOT NULL

Чтобы указать, что обязательно нужны некие значения: NOT NULL

```
create table Customers
(
    ...,
    Name CHARACTER VARYING(52) NOT NULL
)
```

```
...  
)
```

По умолчанию NULL возможен, если не указать обратного

## DEFAULT

DEFAULT - значение по умолчанию, когда при добавлении не определяем значение этого атрибута

```
create table customers  
(  
    Id SERIAL PRIMARY KEY,  
    FirstName CHARACTER VARYING(30),  
    LastName CHARACTER VARYING(30),  
    Age INTEGER DEFAULT 18  
);
```

## CHECK

CHECK - задаёт условие для проверки значений

```
Age INTEGER CHECK(Age > 0 AND Age < 100)  
Email CHARACTER VARYING(30) UNIQUE CHECK(Email != '')
```

CHECK можно наложить на всю таблицу

## CONSTRAINT

CONSTRAINT - задаёт имя для ограничений

## Изменение таблиц

```
ALTER TABLE <name>  
ADD name_column type_date [attr] |  
DROP COLUMN name_column |  
ALTER COLUMN name_column параметры |  
ADD [CONSTRAINT] определение ограничения |  
DROP [CONSTRAINT] имя_ограничения  
;
```

С помощью TYPE можно поменять тип колонки (но не всегда работает)

SET добавляет ограничение

RENAME - переименовать столбец

# Редактирование данных

## Добавление записей в таблицу

```
INSERT INTO имя_таблицы (столбец1, столбец 2, столбец 3) VALUES (значение 1, значение 2, значение 3)
```

Можно не указывать перечень столбцов, но тогда нужно указать значения для всех столбцов в определённом порядке

Также можно добавить несколько строк указывая после VALUES через запятую в скобках сами строки

```
UPDATE имя_таблицы SET столбец1 = значение1, столбец 2 = значение 2 WHERE условие(для каких записей обновить)
```

```
DELETE FROM имя_таблицы WHERE условие (что конкретно удалить)
```

Удаляется строка!