SQL. Structured Query Language

- SQL – это широко распространенный и стандартизированный язык, который используется для работы с реляционными базами данных, поддерживается большинством производителей СУБД.
- Первые разработки появились в 70 годах.
- Structured English Query Language (1983) – первый стандарт появился.
- Стандарты: 86, 89, 92, 1999.

Группы операторов языка SQL:

- DDL (Язык Определения Данных);
- DML (Язык Манипулирования Данными);
- DCL (Язык Управления Данными).

SQL DDL (Язык Определения Данных)

Основные операторы

- CREATE <OBJECT>[OPTIONS] – создает объекты;
- ALTER <OBJECT>[OPTIONS] – изменяет объекты;
- DROP <OBJECT>[OPTIONS] – удаляет объекты;

DML (Язык Манипулирования Данными)

- SELECT – оператор выборки;
- INSERT – оператор вставки данных;
- UPDATE - оператор изменения данных;
- DELETE – оператор удаления данных;
- TRUNCATE - оператор удаления данных;
- COMMIT – зафиксирует транзакцию;
- ROLLBACK – откат транзакции.

DCL (Язык Управления Данными)

операторы, которые управляют правами пользователя:

- GRANT - используется для назначения привилегий пользователям.
- REVOKE - осуществляется отмена привилегий.

POSTGRESQL

История

- СУБД POSTGRES - разрабатывался под руководством Майкла Стоунбрейкера (Michael Stonebraker), профессора Калифорнийского университета в Беркли.
- До этого Майкл Стоунбрейкер уже возглавлял разработку INGRES : POSTGRES возник, как результат продолжения работы.
- Первая версия СУБД была выпущена в 1989 году.
- В 1994 году Беркли Эндрю и Джоли Чену взялись за его дальнейшее развитие. Проект получил название Postgres95.
- К 1996 году получило новое развитие, связь с языком SQL и получило название PostgreSQL.

Установка PostgreSQL

- Скачайте с сайта:

www.postgrespro.ru/

postgresql-13.2-1-windows-x64

PostgreSQL_9.6.12_64bit_Setup – установочник PostgreSQL
.

- **PgAdmin3_1.22.1_X86bit_Setup** — графическое средство для PostgreSQL. Программа упрощает основные задачи администрирования, отображает объекты баз данных, позволяет выполнять запросы SQL.

- Приглашение имеет вид : postgres=#.
- «Postgres» здесь — имя базы данных, к которой вы сейчас подключены. Один сервер

  PostgreSQL может одновременно обслуживать несколько баз данных, но одновременно вы работаете только с одной из них.
- При неправильным отображением русских букв в терминале :
- Вести команду chcp 1251.
- В свойствах окна измените на «Lucida Console».

Полезные консольные команды psql

- \? Справка по командам psql.
- \h Справка по SQL: список доступных команд или синтаксис конкретной команды.
- \x Переключает обычный табличный вывод (столбцы и строки) на расширенный (каждый столбец на отдельной строке) и обратно. Удобно для просмотра нескольких «широких» строк.
- \l Список баз данных.
- \dt Список таблиц.
- \q Завершить сеанс работы.

Создание новой базы данных с именем test

- postgres=# **CREATE DATABASE test;**

Переключение на созданную базу:

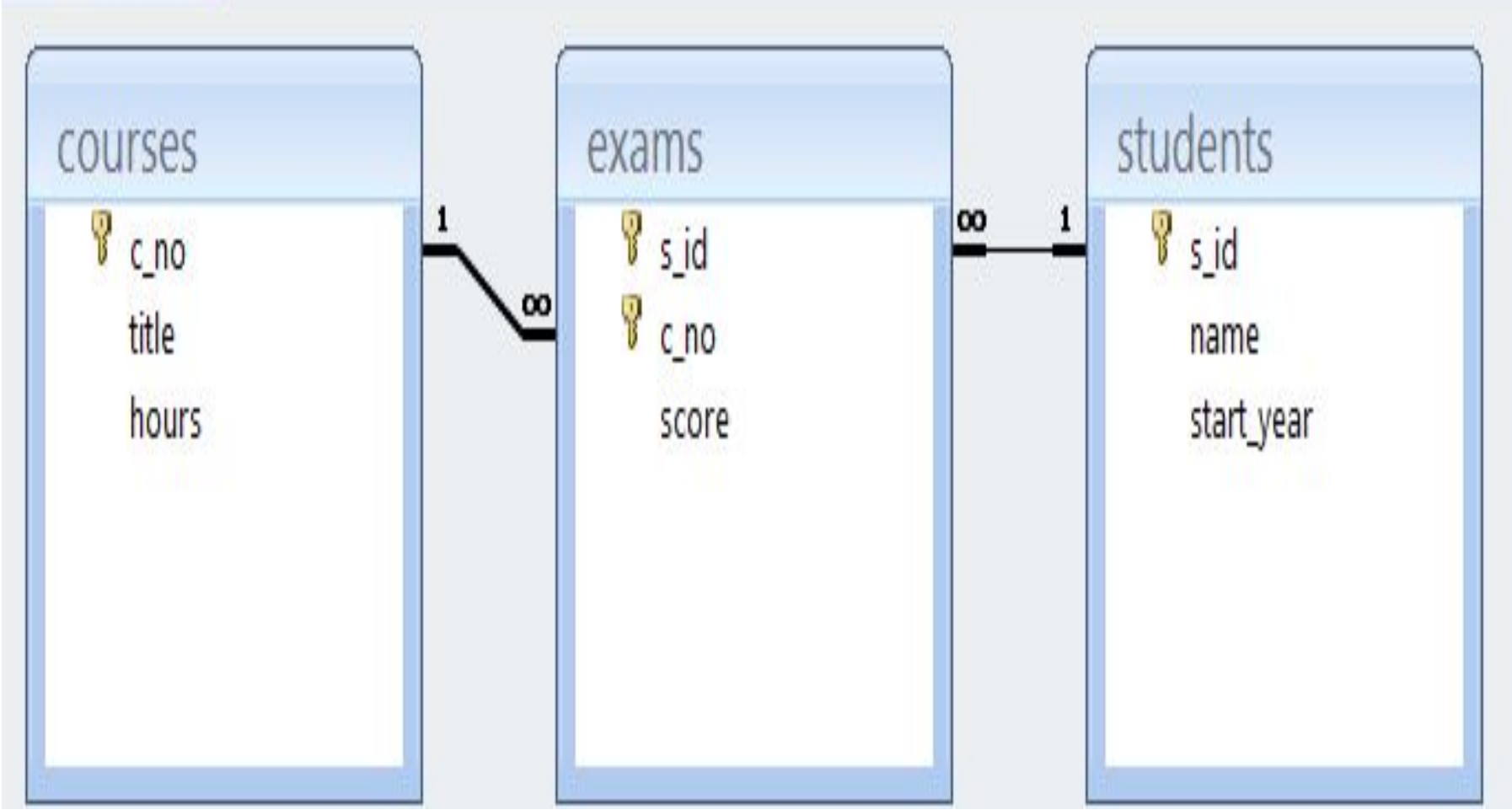
- postgres=# **\c test**
- приглашение сменилось на test=#
- test=# **\? –** полный список команд

Типы данных

- integer — целые числа;
- text — текстовые строки;
- boolean — логический тип, принимающий значения true (истина) или false (ложь);
- Date - дата.
- *неопределенное значение null* «значение неизвестно» или «значение не задано».

полный список типов данных :
postgrespro.ru/doc/datatype.html

Схема данных



Пример создания таблицу дисциплин

```
CREATE TABLE courses(
 c_no text PRIMARY KEY,
title text,
hours integer
 );
```

Полный список ограничений целостности:

postgrespro.ru/doc/ddl-constraints.html

Наполнение таблиц

```
INSERT INTO courses(c_no, title,hours)
VALUES ('CS301', 'Базы данных', 30),
('CS305', 'Сети ЭВМ', 60);
```

Создание таблицы students

- test=# **CREATE TABLE students (**

s_id integer PRIMARY KEY,

name text,

start_year integer);

Заполнение:

- test=# **INSERT INTO students(s_id, name, start_year)**

VALUES (1451, 'Анна', 2014),

(1432, 'Виктор', 2014),

(1556, 'Нина', 2015);

Создание внешнего ключа

- test=# **CREATE TABLE exams(**

s_id integer REFERENCES students(s_id),
c_no text REFERENCES courses(c_no),
score integer,
CONSTRAINT pk PRIMARY KEY(s_id, c_no)
);

- test=# **INSERT INTO exams(s_id, c_no, score)**

VALUES (1451, 'CS301', 5),
(1556, 'CS301', 5),
(1451, 'CS305', 5),
(1432, 'CS305', 4);

Общая форма команды CREATE TABLE

```
CREATE TABLE "имя_таблицы"
( имя_поля тип_данных [ограничения_целостности],
  имя_поля тип_данных [ограничения_целостности],
...
  имя_поля тип_данных [ограничения_целостности],
  [ограничение_целостности],
  [первичный_ключ],
  [внешний_ключ]
);
```

Создание ограничений

NOT NULL – (NULL - неопределенность) –не содержит неопределенное значение.

CHECK задаётся выражение, возвращающее булевский результат, по которому определяется, будет ли успешна операция добавления или изменения для конкретного значения.

```
CREATE TABLE aircrafts
( aircraft_code char( 3 )  NOT NULL,
model text  NOT NULL,
range integer  NOT NULL,
 CHECK ( range > 0 ),
PRIMARY KEY ( aircraft_code ));

CREATE TABLE progress
( ...
mark numeric( 1 ),
CONSTRAINT valid_mark CHECK ( mark >= 3 AND mark <= 5 ),
...
);
```

- Ограничение уникальности **UNIQUE** – все значения столбца должны быть уникальными.

```
CREATE TABLE students
( record_book numeric( 5 ) UNIQUE,
... );

CREATE TABLE students
( record_book numeric( 5 ),
name text NOT NULL,
...
CONSTRAINT unique_record_book UNIQUE (
   record_book ),
…);
```

Первичный ключ.

```
CREATE TABLE students
( record_book numeric( 5 ) PRIMARY KEY,
…);
```

```
CREATE TABLE students
( record_book numeric( 5 ),
…
PRIMARY KEY ( record_book ));
```

Внешний ключ.

```
CREATE TABLE progress
( record_book numeric( 5 ),
…
FOREIGN KEY ( record_book )
REFERENCES students ( record_book )
);
```

Удаление таблицы

- **DROP TABLE имя таблицы;**

Выборка данных. Простые запросы.Оператор SELECT

SELECT имя_поля1, имя_поля2 ...

FROM имя_таблицы;

Вывод два столбца из таблицы courses:

SELECT title AS course_title, hours

FROM courses;

- Конструкция **AS** позволяет переименовать столбец, если это необходимо.

Оператор **SELECT**

- Чтобы вывести все столбцы, необходимо указать символ звездочки:

SELECT * FROM courses;

- Чтобы результирующей строке убрать дублирующие строки, после select надо добавить слово distinct:

  SELECT DISTINCT start_year FROM students;

- Подробно в документации:

Заданий условий

Условие фильтрации записывается во фразе **WHERE:**

```
SELECT * FROM courses
    WHERE hours > 45;
```

- Условие должно иметь логический тип. Оно может содержать отношения =, <> (или !=), >, >=, <, <=, Like (NOT Like) .
- может объединять опреаторов с **and, or, not** и круглых скобок — как в обычных языках программирования.
- Можно использовать шаблоны **%, _**.
- Between …. And….

Примеры

- **SELECT * FROM aircrafts WHERE model LIKE 'Airbus%';**

- **SELECT * FROM aircrafts WHERE range BETWEEN 3000 AND 6000;**

```
SELECT model, aircraft_code, range  FROM aircrafts
WHERE range >= 4000 AND range <= 6000;
```

Удаление строки **DELETE**

DELETE FROM Имя_таблицы

WHERE условие;

Примеры.

```
DELETE FROM aircrafts  WHERE aircraft_code = 'CN1';
DELETE FROM aircrafts WHERE range > 10000 OR
    range < 3000;
DELETE FROM aircrafts;
```

Создание вычисляемых полей

- **SELECT model, range, range / 1.609 AS miles FROM aircrafts;**

Упорядочение данных **ORDER BY**

- По возрастанию (по умолчанию):

SELECT * FROM aircrafts ORDER BY range;

- По убыванию **DESC** :

SELECT * FROM aircrafts ORDER BY range DESC;

- Ограничение число строк (**LIMIT)** :

```
SELECT airport_name, city, longitude
FROM airports
ORDER BY longitude DESC
LIMIT 3;
```

- Для пропуска строк (**OFFSET**):

```
SELECT airport_name, city, longitude
FROM airports
ORDER BY longitude DESC
LIMIT 3 OFFSET 3;
```

Условные выражения

- CASE WHEN condition THEN result
  [WHEN ...]
  [ELSE result]
  END;

```
SELECT model, range,
    CASE WHEN range < 2000 THEN
    ‘Ближнемагистральный'
       WHEN range < 5000 THEN ‘Среднемагистральный‘
        ELSE ' Дальнемагистральный '
    END AS type
FROM aircrafts
ORDER BY model;
```

- model | range | type
- ----------+-------+----------
- Airbus A319-100 | 6700 | **Дальнемагистральный**
- Airbus A320-200 | 5700 | **Дальнемагистральный**
- Airbus A321-200 | 5600 | **Дальнемагистральный**
- Boeing 737-300 | 4200 | **Среднемагистральный**
- Boeing 767-300 | 7900 | **Дальнемагистральный**
- Boeing 777-300 | 11100 | **Дальнемагистральный**
- Bombardier CRJ-200 | 2700 | **Среднемагистральный**
- Cessna 208 Caravan | 1200 | **Ближнемагистральный**

Группировка данных GROUP BY

- Группировка данных – это объединение записей в соответствии со значениями некоторого заданного поля.
- Для группировки результатов выборки совместно с оператором SELECT используется предложение GROUP BY. Данное предложение должно следовать после предложение WHERE, но перед предложением ORDER BY. Как правило, совместно с предложением GROUP BY используются функции агрегирования.

Пример 1.

- Надо подсчитать количество покупок товаров, сделанных каждым из клиентов, используется следующий запрос:

```
SELECT Код_клиента,
SUM(Продано) AS Количество_покупок,
FROM Продажи
GROUP BY  Код_клиента;
```

Пример 2

SELECT aircraft_code, fare_conditions, count(*)
FROM seats
GROUP BY aircraft_code, fare_conditions
ORDER BY aircraft_code, fare_conditions;

aircraft_code | fare_conditions | count

- ---------------+-----------------+-------
  319 | Business | 20
  319 | Economy | 96
  320 | Business | 20
  320 | Economy | 120

HAVING

Чтобы сузить множество группированных записей

- **SELECT departure_city, count(*)**
  FROM routes
  GROUP BY departure_city
  HAVING count(*) >= 15
  ORDER BY count DESC;

СОЕДИНЕНИЯ

- Соединение двух таблиц **на основе равенства** значений атрибутов
- **SELECT a.aircraft_code, a.model, s.seat_no, s.fare_conditions**
  FROM seats AS s, aircrafts AS a
  WHERE s.aircraft_code = a.aircraft_code
  AND a.model ~ '^Cessna'
  ORDER BY s.seat_no;

- **SELECT a.aircraft_code, a.model, s.seat_no, s.fare_conditions**
  FROM seats s
  JOIN aircrafts a
  ON s.aircraft_code = a.aircraft_code
  WHERE a.model ~ '^Cessna'
  ORDER BY s.seat_no;

ВНЕШНИЕ СОЕДИНЕНИЯ

- **Левое внешнее соединение(LEFT OUTER JOIN)**

```
SELECT a.aircraft_code AS a_code,
    a.model, r.aircraft_code AS r_code,
    count( r.aircraft_code ) AS num_routes
FROM aircrafts a
LEFT OUTER JOIN routes r ON r.aircraft_code =
  a.aircraft_code
GROUP BY 1, 2, 3
ORDER BY 4 DESC;
```

- **Правое внешнее соединение(RIGHT OUTER JOIN)**
- **Полное внешнее соединение(FULL OUTER JOIN)**

ОПЕРАЦИИ С ВЫБОРКАМИ

- В SELECT предусмотрены средства выполнения операции с выборками, как множествами:
- UNION- для вычисления объединения множества строк из двух выборок;
- INTERSECT – для вычисления пересечения множества строк из двух выборок;
- EXCERT- для вычисления разности множества строк из двух выборок;

UNION

- Вопрос: В какие города можно улететь: либо из Москвы, либо из Санкт-Петербурга?
- **SELECT arrival_city FROM routes**

  WHERE departure_city = ‘Москва'

  UNION

  SELECT arrival_city FROM routes

  WHERE departure_city = ' Санкт-Петербурга '

  ORDER BY arrival_city;

INTERSECT

- Вопрос: В какие города можно улететь: как из Москвы, так из Санкт-Петербурга?
- **SELECT arrival_city FROM routes**

  WHERE departure_city = ‘Москва'

  INTERSECT

  SELECT arrival_city FROM routes

  WHERE departure_city = ' Санкт-Петербурга '

  ORDER BY arrival_city;

EXCEPT

- Вопрос: В какие города можно улететь:из Санкт-Петербурга , но нельзя из Москвы?
- **SELECT arrival_city FROM routes**

  WHERE departure_city = ‘Санкт-Петербурга'

  EXCEPT

  SELECT arrival_city FROM routes

  WHERE departure_city = ' Москва'

  ORDER BY arrival_city;

Подзапросы

- Вложенная команда select, заключенная в круглые скобки, называется ***подзапросом.***
- Подзапросы могут присутствовать в предложениях SELECT, FROM, WHERE и HAVING.

Примеры

- **SELECT name,**
  (SELECT score
  FROM exams
  WHERE exams.s_id = students.s_id
  AND exams.c_no = 'CS305')

FROM students;

- name | score
- −−−−−−−+−−−−−−−
- Анна | 5
- Виктор | 4
- Нина |
- (3 rows)

Использование подзапросов в WHERE

- **SELECT ***

  FROM exams

  WHERE (SELECT start_year

  FROM students

  WHERE students.s_id = exams.s_id) > 2014;

  s_id | c_no | score

- ------+-------+-------

- 1556 | CS301 | 5

- **SELECT name, start_year**
  FROM students
  WHERE s_id in (SELECT s_id
  FROM exams
  WHERE c_no = 'CS305');
- name | start_year
- ––––––––+––––––––––––
- Анна | 2014
- Виктор | 2014

- **SELECT name, start_year**
  FROM students
  WHERE s_id NOT IN (SELECT s_id
  FROM exams
  WHERE score < 5);
- name | start_year
- ------+------------
- Анна | 2014
- Нина | 2015

- **SELECT name, start_year**
  FROM students
  WHERE NOT EXISTS (SELECT s_id
  FROM exams
  WHERE exams.s_id = students.s_id AND score < 5);

Изменение данных

- UPDATE имя_таблицы
- SET имя_поля_1=значение_1
- [,имя_поля_2=значение_2]
- …
- [,имя_поля_N=значение_N]
- [WHERE условие];
- **UPDATE courses**
- **SET hours = hours*2**
- **WHERE c_no = 'CS301';**

Удаление данных

- DELETE FROM Имя_табл [WHERE условие];
- **DELETE FROM exams WHERE score < 5;**