Мамонов Антон ЗИСиП-19-1

Создаем базу данных

```
postgres=# create database test;
CREATE DATABASE
postgres=# \c test
Вы подключены к базе данных "test" как пользователь "postgres".
test=# \?
╟с·шх
  \copyright             єёыютш  шёяюы№чютрэш  ш ЁрёяЁюёЄЁрэхэш  PostgreSQL
  \crosstabview [╤╥╦┴╬┼╓█] тV яюыэшЄ№ чряЁюё ш тVтхёЄш ЁхчеыN№ЄрЄ т яхЁхъЁ╕ёЄэюь тшфх
```

Создаем таблицу courses

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

Помощь с созданием

```
test=# \help create table
Команда:     CREATE TABLE
Описание:    создать таблицу
Синтаксис:
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UN
  { имя_столбца тип_данных [ COLLATE правило_сортиров
    | ограничение_таблицы
    | LIKE исходная_таблица [ параметр_порождения ...
    [, ... ]
] )
```

Заполняем данными

```
^Ctest=# INSERT INTO courses(c_no, title, hours)
test-# VALUES ('CS301', 'Базы данных', 30),
test-# ('CS305', 'Сети ЭВМ', 60);
INSERT 0 2
```

Создаем таблицу students

```
test=# CREATE TABLE students(
test(# s_id integer PRIMARY KEY,
test(# name text,
test(# start_year integer
test(# );
CREATE TABLE
```

Заполняем данными

```
test=# INSERT INTO students(s_id, name, start_year)
test-# VALUES (1451, 'Анна', 2014),
test-# (1432, 'Виктор', 2014),
test-# (1556, 'Нина', 2015);
INSERT 0 3
```

Создаем таблицу exams

```
test=# CREATE TABLE exams(
test(# s_id integer REFERENCES students(s_id),
test(# c_no text REFERENCES courses(c_no),
test(# score integer,
test(# CONSTRAINT pk PRIMARY KEY(s_id, c_no)
test(# );
CREATE TABLE
```

Заполняем данными

```
test=# INSERT INTO exams(s_id, c_no, score)
test-# VALUES (1451, 'CS301', 5),
test-# (1556, 'CS301', 5),
test-# (1451, 'CS305', 5),
test-# (1432, 'CS305', 4);
INSERT 0 4
```

Запрос из таблицы courses

```
test=# SELECT title AS course_title, hours
test-# FROM courses;
 course_title | hours
--------------+-------
 Базы данных  |    30
 Сети ЭВМ     |    60
(2 строки)
```

```
test=# SELECT * FROM courses;
 c_no  |    title     | hours
-------+--------------+-------
 CS301 | Базы данных  |    30
 CS305 | Сети ЭВМ     |    60
(2 строки)
```

Выбираем все из таблицы courses

Выбираем начало года из таблицы students. Имеются повторения.

```
test=# SELECT start_year FROM students;
 start_year
------------
       2014
       2014
       2015
(3 строки)
```

```
test=# SELECT DISTINCT start_year FROM students;
 start_year
------------
       2014
       2015
(2 строки)
```

DISTINCT разобрался с повторениями

2+2 = ?

```
test=# SELECT 2+2 AS result;
 result
--------
      4
(1 строка)
```

```
test=# SELECT * FROM courses WHERE hours > 45;
 c_no  |  title    | hours
-------+----------+-------
 CS305 | Сети ЭВМ |    60
(1 строка)
```

Select с условием

Выбираем все из таблиц courses и exams. Выглядит не круто

```
test=# SELECT * FROM courses, exams;
 c_no  |    title     | hours | s_id | c_no  | score
-------+--------------+-------+------+-------+-------
 CS301 | Базы данных  |    30 | 1451 | CS301 |     5
 CS305 | Сети ЭВМ     |    60 | 1451 | CS301 |     5
 CS301 | Базы данных  |    30 | 1556 | CS301 |     5
 CS305 | Сети ЭВМ     |    60 | 1556 | CS301 |     5
 CS301 | Базы данных  |    30 | 1451 | CS305 |     5
 CS305 | Сети ЭВМ     |    60 | 1451 | CS305 |     5
 CS301 | Базы данных  |    30 | 1432 | CS305 |     4
 CS305 | Сети ЭВМ     |    60 | 1432 | CS305 |     4
```

```
test=# SELECT courses.title, exams.s_id, exams.score
test-# FROM courses, exams
test-# WHERE courses.c_no = exams.c_no;
    title     | s_id | score
--------------+------+-------
 Базы данных | 1451 |     5
 Базы данных | 1556 |     5
 Сети ЭВМ    | 1451 |     5
 Сети ЭВМ    | 1432 |     4
(4 строки)
```

Так делать правильнее

Запросы с помощью JOIN

```
test=# SELECT students.name, exams.score
test-# FROM students
test-# JOIN exams
test-# ON students.s_id = exams.s_id
test-# AND exams.c_no = 'CS305';
  name   | score
--------+-------
 Анна    |     5
 Виктор |     4
(2 строки)
```

```
test=# SELECT students.name, exams.score
test-# FROM students
test-# LEFT JOIN exams
test-# ON students.s_id = exams.s_id
test-# AND exams.c_no = 'CS305';
  name   | score
--------+-------
 Анна    |     5
 Виктор |     4
 Нина    |
(3 строки)
```

Запросы с помощью LEFT JOIN

Меняем and на where, Нина уже не будет в выборке

```
test=# SELECT students.name, exams.score
test-# FROM students
test-# LEFT JOIN exams ON students.s_id = exams.s_id
test-# WHERE exams.c_no = 'CS305';
  name   | score
--------+-------
 Анна    |     5
 Виктор |     4
(2 строки)
```

Выборка с подзапросом в блоке select

```
test=# SELECT name,
test-# (SELECT score
test(# FROM exams
test(# WHERE exams.s_id = students.s_id
test(# AND exams.c_no = 'CS305')
test-# FROM students;
  name   | score
--------+-------
 Анна    |     5
 Виктор |     4
 Нина    |
(3 строки)
```

Выборка с подзапросом в блоке where

```
test=# SELECT name, start_year
test-# FROM students
test-# WHERE s_id IN (SELECT s_id
test(# FROM exams
test(# WHERE c_no = 'CS305');
  name   | start_year
--------+------------
 Анна    |       2014
 Виктор |       2014
(2 строки)
```

```
test=# SELECT *
test-# FROM exams
test-# WHERE (SELECT start_year
test(# FROM students
test(# WHERE students.s_id = exams.s_id) > 2014;
 s_id | c_no  | score
------+-------+-------
 1556 | CS301 |     5
(1 строка)
```

Выборка с подзапросом, который возвращает произвольное количество строк

```
test=# SELECT name, start_year
test-# FROM students
test-# WHERE s_id NOT IN (SELECT s_id
test(# FROM exams
test(# WHERE score < 5);
 name | start_year
------+------------
 Анна |       2014
 Нина |       2015
(2 строки)
```

Выборка с подзапросом и NOT IN, который проверяет, что значение не входит в подзапрос

```
test=# SELECT name, start_year
test-# FROM students
test-# WHERE NOT EXISTS (SELECT s_id
test(# FROM exams
test(# WHERE exams.s_id = students.s_id
test(# AND score < 5);
 name | start_year
------+------------
 Анна |       2014
 Нина |       2015
(2 строки)
```

Подзапрос с EXISTS на проверку, выдал ли подзапрос хоть одну строку

```
test=# SELECT s.name, ce.score
test-# FROM students s
test-# JOIN (SELECT exams.*
test(# FROM courses, exams
test(# WHERE courses.c_no = exams.c_no
test(# AND courses.title = 'Базы данных') ce
test-# ON s.s_id = ce.s_id;
 name | score
------+-------
 Анна |     5
 Нина |     5
(2 строки)
```

Вводим псевдонимы

```
test=# SELECT s.name, e.score
test-# FROM students s, courses c, exams e
test-# WHERE c.c_no = e.c_no
test-# AND c.title = 'Базы данных'
test-# AND s.s_id = e.s_id;
 name | score
------+-------
 Анна |     5
 Нина |     5
(2 строки)
```

Тот же запрос можно написать без подзапроса

```
test=# SELECT * FROM exams
test-# ORDER BY score, s_id, c_no DESC;
 s_id | c_no  | score
------+-------+-------
 1432 | CS305 |     4
 1451 | CS305 |     5
 1451 | CS301 |     5
 1556 | CS301 |     5
(4 строки)
```

Сортировка ORDER BY

```
test=# SELECT count(*), count(DISTINCT s_id),
test-# avg(score)
test-# FROM exams;
 count | count |        avg
-------+-------+--------------------
     4 |     3 | 4.7500000000000000
(1 строка)
```

Общее количество экзаменов

```
test=# SELECT c_no, count(*),
test-# count(DISTINCT s_id), avg(score)
test-# FROM exams
test-# GROUP BY c_no;
 c_no  | count | count |        avg
-------+-------+-------+--------------------
 CS301 |     2 |     2 | 5.0000000000000000
 CS305 |     2 |     2 | 4.5000000000000000
(2 строки)
```

Группировка GROUP BY по полю c_no

```
test=# SELECT students.name
test-# FROM students, exams
test-# WHERE students.s_id = exams.s_id AND exams.score = 5
test-# GROUP BY students.name
test-# HAVING count(*) > 1;
 name
------
 Анна
(1 строка)
```

GROUP BY и HAVING COUNT вместе

Меняем таблички

```
test=# UPDATE courses
test-# SET hours = hours * 2
test-# WHERE c_no = 'CS301';
UPDATE 1
```

```
test=# DELETE FROM exams WHERE score < 5;
DELETE 1
```

Удаляем строки с условием из таблицы exams

```
test=# CREATE TABLE groups(
test(# g_no text PRIMARY KEY,
test(# monitor integer NOT NULL REFERENCES students(s_id)
test(# );
CREATE TABLE
```

Транзакции

Меняем таблицу students, добавляем новый столбец

```
test=# ALTER TABLE students
test-# ADD g_no text REFERENCES groups(g_no);
ALTER TABLE
```

```
test=# \d students
                    Таблица "public.students"
  Столбец   |   Тип   | Правило сортировки | Допустимость NULL | По умолчанию
------------+---------+--------------------+-------------------+--------------
 s_id       | integer |                    | not null          |
 name       | text    |                    |                   |
 start_year | integer |                    |                   |
 g_no       | text    |                    |                   |
```

Смотрим на табличку

Смотрим, какие таблицы вообще есть в базе данных

```
test=# \d
            Список отношений
 Схема  |   Имя    |   Тип   | Владелец
--------+----------+---------+----------
 public | courses  | таблица | postgres
 public | exams    | таблица | postgres
 public | groups   | таблица | postgres
 public | students | таблица | postgres
(4 строки)
```

Прикалываемся с транзакцией

Выполняем на основной консоли

```
test=# BEGIN;
BEGIN
test=*# INSERT INTO groups(g_no, monitor)
test-*# SELECT 'A-101', s_id
test-*# FROM students
test-*# WHERE name = 'Анна';
INSERT 0 1
test=*#
```

Со второй консоли пытаемся вывести, ничего нет, так как транзакция еще не завершена

```
test=# SELECT * FROM groups;
 g_no | monitor
------+---------
(0 строк)
```

```
UPDATE 3
test=*#
```

Делаем UPDATE в первой консоли

```
test=# SELECT * FROM students;
 s_id |  name  | start_year | g_no
------+--------+------------+------
 1451 | Анна   |       2014 |
 1432 | Виктор |       2014 |
 1556 | Нина   |       2015 |
(3 строки)
```

Теперь во второй консоли все видно

```
test=# SELECT * FROM groups;
 g_no  | monitor
-------+---------
 A-101 |    1451
(1 строка)
```

```
test=# SELECT * FROM students;
 s_id |  name  | start_year | g_no
------+--------+------------+-------
 1451 | Анна   |       2014 | A-101
 1432 | Виктор |       2014 | A-101
 1556 | Нина   |       2015 | A-101
(3 строки)
```

Теперь во второй консоли есть доступ ко всем данным