Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Базы данных

Тема: Язык SQL-DML

Выполнил студент гр. 43501/3 Крылов И.С.

Преподаватель Мяснов А.В

«_____» _____2018 г.

Санкт-Петербург
2018

1 Цель работы.

Познакомитьсяс языком создания запросов управления данными SQL-DML.

2 Программа работы

1. Изучение SQL-DML.
2. Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
3. Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием. Демонстрация результатов преподавателю.
4. Сохранение в БД выполненных запросов SELECT в виде представлений, запросов INSERT, UPDATE или DELETE – в виде ХП. Выкладывание скрипта в GitLab.

3 Список стандартных запросов

- Сделайте выборку всех данных из каждой таблицы
- Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее 3-х разных примеров)
- Создайте в запросе вычисляемое поле
- Сделайте выборку всех данных с сортировкой по нескольким полям
- Создайте запрос, вычисляющий несколько совокупных характеристик таблиц
- Сделайте выборку данных из связанных таблиц (не менее двух примеров)
- Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки
- Придумайте и реализуйте пример использования вложенного запроса
- С помощью оператора INSERT добавьте в каждую таблицу по одной записи

- С помощью оператора UPDATE измените значения нескольких полей у всех записей, отвечающих заданному условию
- С помощью оператора DELETE удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики
- С помощью оператора DELETE удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

4 Ход работы

База была заполненна тестовыми данными с помощью генератора.

4.1 Выполнение стандартных запросов

Для того, чтобы сделать выборку всех данных из каждой таблицы воспользуемся скриптом:

```
select * from cpu;
```

Получим данные из таблиц *developer* и *tournament* при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN:

```
SELECT * FROM developer WHERE year_of_foundation BETWEEN 1954 AND
    1970;
SELECT * FROM tournament WHERE xprize IN('4');
SELECT * FROM tournament WHERE rules LIKE 'S%';
```

Используем в запросе вычисляемое поле, чтобы определить сколько турниров с призом = 4:

```
SELECT COUNT (*) FROM tournament WHERE xprize = '4';
```

Организуем выборку данных с сотрировкой по нескольким полям. Сортировать данные будем из таблицы *tournament* по полям *tournament_name* и *tournament_xprize*:

```
select * from tournament order by name, xprize desc;
```

Таким образом получим турниры по убыванию призового фонда.

Для вычисления совокупных характеристик таблицы разработаем sql-скрипт:

```
SELECT avg(price), max(price), min(price) FROM game;
SELECT avg(price), max(price), min(price) FROM dlc;
```

Сделаем выборку данных из связаных таблиц.

```
select * from game join dlc on (dlc.id_dlc = game.id_dlc);
select * from minimal_requirements join os on os.id_os =
   minimal_requirements.id_os;
```

Выведем память gpu по возрастанию частоты значения в БД:

```
SELECT memory, COUNT(*) AS mem FROM gpu GROUP BY memory
HAVING COUNT(*) > 2
ORDER BY mem;
```

Вложенный запрос, который выводит 64 разрядные системы и процессоры с ОП 2Гб :

```
select * from optimal_requirements where id_os in (select id_os
   from os where bits > 20);
select * from minimal_requirements where id_cpu in (select id_cpu
    from cpu WHERE ram IN('2'));
```

Был разработан скрипт, который добавляет в таблицу по одной записи:

```
INSERT INTO cpu VALUES ((SELECT MAX(id_cpu) FROM cpu)+1,'I am','
   krylov ivan', 2,2,8);
```

Создан скрипт, который изменяет значение в базе данных:

```
UPDATE cpu SET ram = 0 WHERE cores = 8;
```

Удалим игру с наибольшей стомостью:

```
delete from game where price = (select max(price) from game);
delete from game where rating = (select min(rating) from game);
```

Удалим системные требования, не привязанные к игре:

```
delete from system_requirements where id_system_requirements not
   in (select id_system_requirements from game);
```

5 Запросы по индивидуальному заданию

5.1 Задание

Реализовать запросы к БД:

1. Вывести процессоры, производительности которых хватает для запуска более половины игр из всего перечня. Производительность процессора измерять как произведение тактовой частоты на количество ядер.

2. Вывести игры, для которых суммарная стоимость дополнительного контента более, чем в 3 раза превышает стоимость игры.

5.2 Процессоры

Был разработан sql-скрипт, который выводит процессоры с производительностью больше чем требуется для половины имеющихся игр.

```
select * from cpu where cpu.cores * cpu.frequency > (select mult
   from (with summary as (select cpu.cores * cpu.frequency as
   mult from game join system_requirements using (
   id_system_requirements) join minimal_requirements using (
   id_minimal_requirements) join cpu using (id_cpu)) select mult,
    ROW_NUMBER () OVER (order by mult) from summary) x where
   ROW_NUMBER = (select count(*) from game) / 2 + 1);
```

Для каждой игры считается необходимая мощность процессора, после чего составляется упорядоченная по требуемой мощности для игры таблица, берётся медианное значение и сравнивается для каждого из процессоров.

5.3 Доп контент

Был разработан sql-скрипт.

```
with summary as (select id_game as id_g, game.name as g_name,
   game.price as g_pr, SUM(dlc.price) as summ from game join dlc
   using (id_game) group by id_game) select id_g, g_name, g_pr,
   summ from summary where (g_pr < summ * 3);
```

Для каждой игры была посчитана общая сумма дополнительного контента, после чего это значение было сравнено с о стоимостью самой игры.

6 Представления и хранимые процедуры

Из запросов сделаных в предидущих пунктах сформируем представления:

```
create view select_all as
 select * from cpu

create view select_between as SELECT * FROM developer WHERE
   year_of_foundation BETWEEN 1954 AND 1970;
create view select_in as SELECT * FROM tournament WHERE xprize IN
   ('4');
create view select_like as SELECT * FROM tournament WHERE rules
   LIKE 'S%';

create view select_count as SELECT COUNT (*) FROM tournament
   WHERE xprize = '4';

create view select_order as select * from tournament order by
   name, xprize desc

create view select_amm as SELECT avg(price), max(price), min(
   price) FROM game;
```

```
13
14 create view select_many as select * from game join dlc on (dlc.
   id_dlc = game.id_dlc);
```

Оформить запросы на изменение базы данных, как хранимые процедуры не получится. Это связано с тем, что хранимые процедуры доступны только начиная с PostgreSQL11. Поэтому будем использовать функции:

```
1 create function updateFirstUserName () returns void as
2 $$
3 begin
4 update cpu
5 set ram = 0
6 where cores = 0;
7 end ;
8 $$
9 language plpgsql ;
```

7 Вывод

В ходе лабораторной работы были улучшены навыки работы с sql-dml и улучшено общее понимание функционирования баз данных. К основным запросам sql-dml относятся SELECT, INSERT, UPDATE, DELETE. В данной работе наиболее подробно рассмотрена работа с SELECT. В запросов этого типа были установлены различные ограничения, вложенные конструкции, объединения. По результатам использования sql-dml можно сказать, что это мощный инструмент, который позволяет легко и элегантно решить поставленную задачу.