Санкт-Петербургский Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №4**

**Дисциплина**: Базы данных

**Тема**: Язык SQL-DML

Выполнил студент гр. 43501/1 Дао Суан Хоа

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2015 г.

Санкт-Петербург

2015

1. **Цель работы**

Познакомить студентов с языком создания запросов управления данными SQL-DML.

1. **Программа работы**
2. Изучите SQL-DML
3. Выполните все запросы из списка стандартных запросов. Продемонстрируйте результаты преподавателю.
4. Получите у преподавателя и реализуйте SQL-запросы в соответствии с **индивидуальным** заданием. Продемонстрируйте результаты преподавателю.
5. Выполненные запросы SELECT сохраните в БД в виде представлений, запросы INSERT, UPDATE или DELETE -- в виде ХП. Выложите скрипт в Subversion.
6. **Язык SQL**

Язык SQL (Structured Query Language) -- язык структурированных запросов. Он позволяет формировать весьма сложные запросы к базам данных. В SQL определены два подмножества языка:

SQL-DDL (Data Definition Language) -- язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.

SQL-DML (Data Manipulation Language) -- язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

1. **Выполнение работы**  
   1) Выборка всех данных из каждой таблицы

CONNECT 'D:/XUANHOA.FDB' USER 'SYSDBA' PASSWORD 'masterkey';

CREATE view CLIENTS\_VIEW AS SELECT \* FROM CLIENTS;

CREATE view COACH\_VIEW AS SELECT \* FROM COACH;

CREATE view EXTRA\_VIEW AS SELECT \* FROM EXTRA\_SEV;

CREATE view SALE\_VIEW AS SELECT \* FROM SALE;

CREATE view SEASON\_VIEW AS SELECT \* FROM SEASON\_TICKET;

CREATE view SPORT\_VIEW AS SELECT \* FROM SPORT;

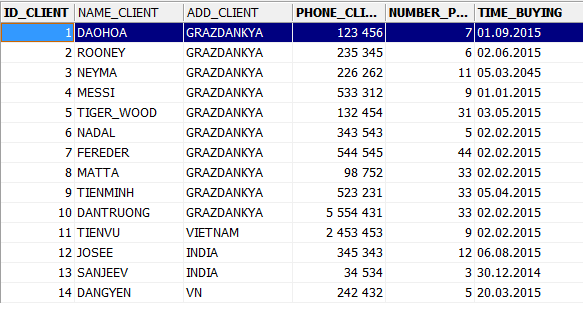
CREATE view STADIUM\_VIEW AS SELECT \* FROM STADIUM;

CREATE view SEASONTICKET\_TYPE\_VIEW AS SELECT \* FROM seasonticket\_type;

CREATE view sport\_tickettype\_VIEW AS SELECT \* FROM sport\_tickettype;

CREATE VIEW CLASS\_VIWE AS select \* FROM CLASS;

Результат для CLIENTS\_VIEW:



* 1. Выборка данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN

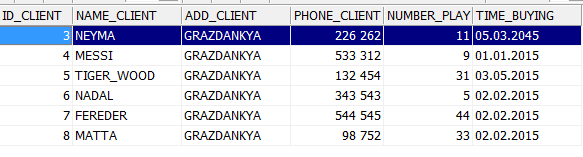
CREATE VIEW MESSI AS SELECT \* FROM CLIENTS WHERE NAME\_CLIENT LIKE 'MESSI';

(Из таблицы CLIENTS показать клиентов с фамилией MESSI )



CREATE VIEW WHERE\_VIEW AS SELECT \* FROM CLIENTS WHERE ID\_CLIENT BETWEEN 3 AND 8;

(Из таблицы CLIENTS показать клиентов с ID между 3 и 8: )



CREATE VIEW IN\_VIEW AS SELECT \* FROM CLIENTS WHERE NUMBER\_PLAY IN (5);

(Из таблицы CLIENTS показать клиентов с номер игры 5)

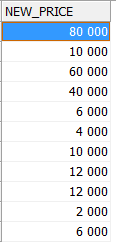


* 1. Создайте в запросе вычисляемое поле

create view NEW\_PRICE AS select

TYPE\_TICKET.PRICE \* 2 as NEW\_PRICE

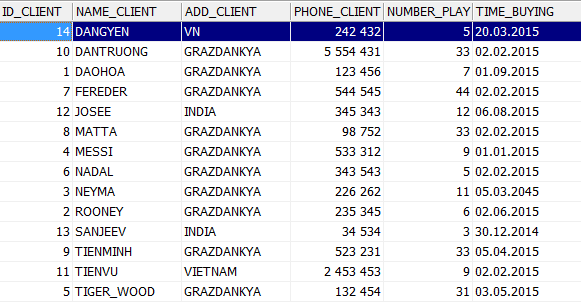
from TYPE\_TICKET ;



* 1. Выборка всех данных с сортировкой по нескольким полям

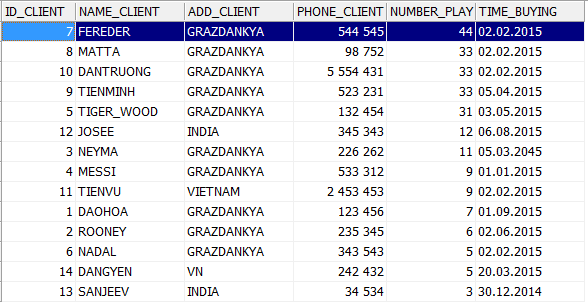
CREATE OR ALTER VIEW CAL\_VIEW AS select \* FROM CLIENTS ORDER BY NAME\_CLIENT ASC;

Клиеты, отсортированные по имени :



CREATE OR ALTER VIEW CAL\_VIEW1 AS select \* FROM CLIENTS ORDER BY number\_play desc;

(Обратная сортировка)



* 1. Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

create view max\_Price11 as select

max(SEASONTICKET\_TYPE.price) as maxPrice from SEASONTICKET\_TYPE ;



* 1. Выборка данных из связанных таблиц

CREATE OR ALTER VIEW RELATE\_VIEW AS

SELECT SEASONTICKET\_TYPE.NAME\_TICKET ,CLIENTS.NAME\_CLIENT

FROM SEASONTICKET\_TYPE,CLIENTS WHERE SEASONTICKET\_TYPE.ID\_SEASONTICKET\_TYPE = CLIENTS.ID\_CLIENT;

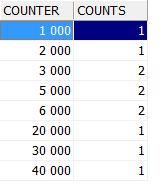


* 1. Запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

create view selectPRICE as select SEASONTICKET\_TYPE.PRICE as COUNTER,

COUNT(SEASONTICKET\_TYPE.PRICE) as Counts

from SEASONTICKET\_TYPE group by SEASONTICKET\_TYPE.PRICE;

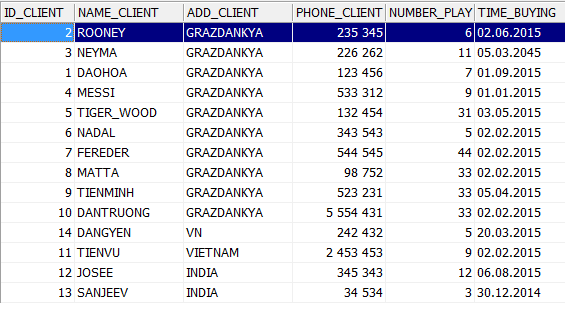


* 1. Использование вложенного запроса

create view NOTSIMPLEJF as SELECT \* FROM CLIENTS

WHERE CLIENTS.ID\_CLIENT IN (SELECT SEASON\_TICKET.ID\_CLIENT FROM SEASON\_TICKET);

Отображение типов абонементов, которые были куплены:



* 1. Использование оператора INSERT для добавления по одной записи в каждую таблицу:

create procedure insert\_SPORT ( i int, d int, p VARCHAR(255))

as begin

insert into SPORT (ID\_SPORT,TYPE\_SPORT)

values (:i, :d, :p);

end;

create procedure insert\_CLIENTS ( i int, d varchar(255), p VARCHAR(255),k int, j int,h varchar(255),)

as begin

insert into CLIENTS (ID\_CLIENT,NAME\_CLIENT,ADD\_CLIENT,PHONE\_CLIENT, NUMBER\_PLAY, TIME\_BUILDING)

values (:i, :d, :p, :k, :j, :h);

end;

create procedure insert\_coach ( i int, d VARCHAR(255), p varchar(255), h varchar(255), t int, k int)

as begin

insert into COACH (ID\_COACH,NAME\_COACH,ADD\_COACH,INF\_COACH,PHONE\_COACH,SPOR T\_ID\_COACH)

values (:i, :d, :p, :h, :t, :k);

end;

create procedure insert\_Stadium ( i int, d VARCHAR(255), p varchar(255), k int, h int)

as begin

insert into STADIUM (ID\_STADIUM,ADD\_STADIUM,NAME\_STADIUM,SIZE\_STADIUM,SPORT \_ID\_STADIUM)

values (:i, :d, :p, :k, :h);

end;

create procedure insert\_class ( i int, d VARCHAR(255), p varchar(255), k int)

as begin

insert into CLASS (ID\_CLASS,NAME\_CLASS,ROOM\_CLASS,SPORT\_ID\_CLASS)

values (:i, :d, :p,:k);

end;

create procedure insert\_sale ( i int, d VARCHAR(255), p date, h varchar(255), k int )

as begin

insert into SALE (ID\_SALE,NAME\_SALE,TIME\_SALE,INF\_SALE,CLIENT\_ID\_SALE)

values (:i, :d, :p, :h, :k);

end;

create procedure insert\_extra ( i int, d VARCHAR(255), p int )

as begin

insert into EXTRA\_SEV (ID\_EXTRA,NAME\_EXTRA,ID\_SEASON)

values (:i, :d, :p);

end;

create procedure insert\_season ( i int, d int, p int )

as begin

insert into SEASON\_TICKET (ID\_SEASON\_TICKET,ID\_SEASONTICKET\_TYPE,ID\_CLIENT)

values (:i, :d, :p);

end;

create procedure insert\_tichket ( i int, d varchar(255), p int,h date )

as begin

insert into SEASONTICKET\_TYPE (ID\_SEASONTICKET\_TYPE,NAME\_TICKET,PRICE,TIME\_TICKET)

values (:i, :d, :p,:h);

end;

create procedure insert\_tichket ( i int, d INT, p int )

as begin

insert into SPORT\_TICKETTYPE (ID\_SPORT\_TICKETTYPE,ID\_SPORT,ID\_TICKETTYPE)

values (:i, :d, :p);

end;

* 1. Помощью оператора UPDATE измените значения нескольких полей у всех записей, отвечающих заданному условию

create procedure update\_PRICE as

begin

update SEASONTICKET\_TYPE set SEASONTICKET\_TYPE.PRICE = SEASONTICKET\_TYPE.PRICE + 500 where SEASONTICKET\_TYPE.PRICE < 2000;

end;

Добавить цены каждого билеты 500$ который имеет цена не больше 2000

11.С помощью оператора DELETE удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

create procedure del\_price (i int) as

begin

delete from TYPE\_TICKET

where SEASONTICKET\_TYPE.PRICE = (select max(SEASONTICKET\_TYPE.PRICE) from SEASONTICKET\_TYPE);

end;

Данная процедура удаляет билет который имеет максимальную цену

* 1. Удаление записей в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос), с помощью оператора DELETE:

create procedure del\_types as

begin

delete from SEASONTICKET\_TYPE where ID\_SEASONTICKET\_TYPE

not in (select ID\_SEASONTICKET\_TYPE from SEASON\_TICKET);

end;

Удаление абонементов, которые никем не куплены.

**Выполнение индивидуального задания:**

Выполнить следующие запросы:

1. Вывести суммарный продажи абонементов каждого типа.
2. Вывести клиентов, которые принесли более заданного количества денег за заданный период.
3. Удалить неиспользуемые типы абонементов.

1/ Вывести суммарный продажи абонементов каждого типа.

CONNECT 'D:/XUANHOA.FDB' USER 'SYSDBA' PASSWORD 'masterkey';

create view SUM\_PRICE AS select

SEASONTICKET\_TYPE.name\_ticket as Name,

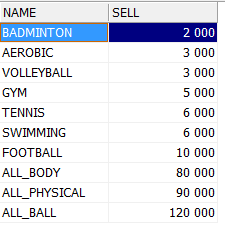
SEASONTICKET\_TYPE.PRICE \* COUNT(SEASON\_TICKET.ID\_SEASONTICKET\_TYPE)

as Sell from SEASONTICKET\_TYPE, SEASON\_TICKET

where SEASON\_TICKET.ID\_SEASONTICKET\_TYPE = SEASONTICKET\_TYPE.ID\_SEASONTICKET\_TYPE group by

SEASONTICKET\_TYPE.name\_ticket,SEASONTICKET\_TYPE.PRICE order by Sell asc;

Результат :



2/ Вывести клиентов, которые принесли более заданного количества денег за заданный период.

CONNECT 'D:/XUANHOA.FDB' USER 'SYSDBA' PASSWORD 'masterkey';

create view buying23 as select

clients.name\_client as id\_client,

sum(seasonticket\_type.price) as BuyingSum

from clients, season\_ticket , seasonticket\_type

where season\_ticket.id\_seasonticket\_type = seasonticket\_type.id\_seasonticket\_type

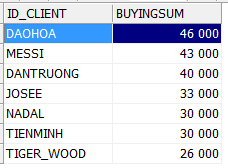
and clients.id\_client = season\_ticket.id\_client

group by clients.name\_client

having sum(seasonticket\_type.price) > 20000

order by BuyingSum desc;

Результат :



3/ Удалить неиспользуемые типы абонементов.

create procedure del\_SEASONTICKET\_TYPE as

begin

delete from SEASONTICKET\_TYPE where ID\_ SEASONTICKET\_TYPE not in (select ID\_ SEASONTICKET\_TYPE from SEASON\_TICKET);

end;

**5/ Вывод**

В результате выполнения работы был изучен язык управления данными SQL-DML. Были выполнены стандартные запросы извлечения данных. Также были выполнены запросы в соответствии с индивидуальным заданием. Были изучены представления и хранимые процедуры, с помощью которых можно спокойно добавлять данные в БД. При выполнении работы проблем не было.

Использовались такие команды языка DML: insert (добавить), update (обновить), delete (удалить), select (выборка данных). Данный язык удобен для написания запросов разной сложности. При обращении к нескольким таблицам в запросе следует выбирать подходящую связь между таблицами для более быстрого выполнения SQL-запроса.