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:/435011/FILMBASE.FDB' USER 'SYSDBA' PASSWORD 'masterkey';

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

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

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

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

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

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

CREATE view AWARD\_ID\_VIEW AS SELECT \* FROM AWARD\_ID;

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

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

CREATE VIEW GENRE\_VIEW AS select \* FROM GENRE;

CREATE view MADE\_IN\_VIEW AS SELECT \* FROM MADE\_IN;

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

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

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

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

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

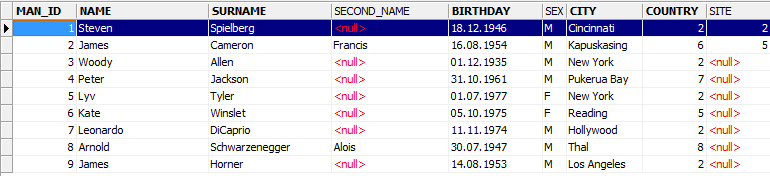
CREATE view SESSION\_VIEW AS SELECT \* FROM SESSION;

CREATE view RENTAL\_COMPANY\_VIEW AS SELECT \* FROM RENTAL\_COMPANY;

CREATE view RENTAL\_VIEW AS SELECT \* FROM RETAL;

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

Результат для MAN\_VIEW:

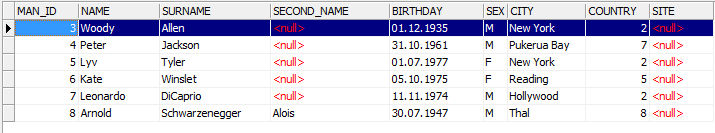


* 1. Выборка данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN

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



CREATE VIEW WHERE\_VIEW AS SELECT \* FROM MAN WHERE MAN\_ID BETWEEN 3 AND 8;

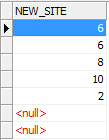


CREATE VIEW IN\_VIEW AS SELECT \* FROM MAN WHERE COUNTRY IN (5);



* 1. Создайте в запросе вычисляемое поле

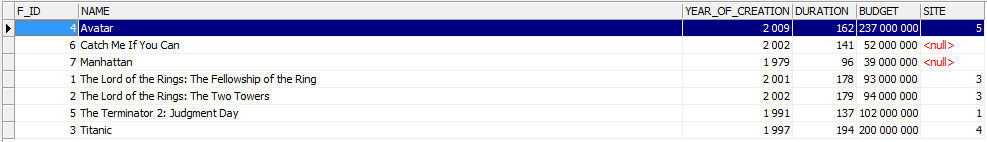
create view NEW\_SITE AS select FILM.SITE \* 2 as NEW\_SITE from FILM ;



* 1. Выборка всех данных с сортировкой по нескольким полям

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

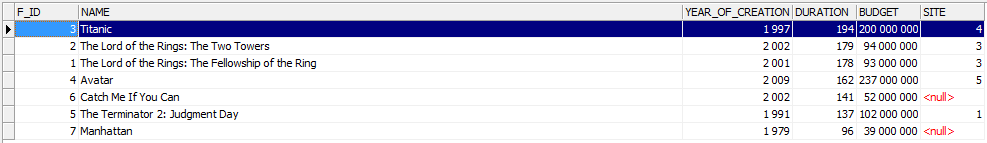
Клиеты, отсортированные по имени :



CREATE OR ALTER VIEW CAL\_VIEW1 AS select \* FROM FILM ORDER BY DURATION

desc;

(Обратная сортировка)



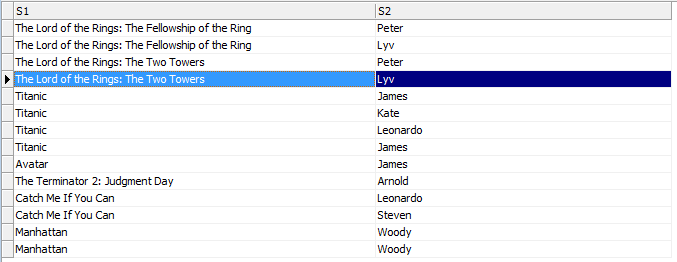
* 1. Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

create view MAX\_BUDGET as select max(FILM.BUDGET) as MAXBUDGET from FILM ;



* 1. Выборка данных из связанных таблиц

CREATE OR ALTER VIEW RELATE\_VIEW AS SELECT FILM.NAME as S1,MAN.NAME AS S2 FROM FILM,MAN,PART WHERE MAN.MAN\_ID = PART.MAN\_ID AND FILM.F\_ID=PART.FILM\_ID;

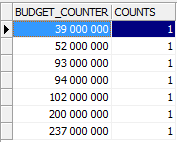


* 1. Запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

create view SELECT\_BUDGET as select FILM.BUDGET as BUDGET\_COUNTER,

COUNT(FILM.BUDGET) as COUNTS

from FILM group by FILM.BUDGET;



* 1. Использование вложенного запроса

create view NOTSIMPLEJF as SELECT \* FROM MAN

WHERE MAN.MAN\_ID IN (SELECT PART.MAN\_ID FROM PART);

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

create procedure insert\_MAN ( a int, b VARCHAR(255),c VARCHAR(255),d VARCHAR(255),e date,f CHAR(1),g VARCHAR(255),h int,i int)

as begin

insert into MAN (MAN\_ID,NAME,SURNAME,SECONDNAME,BIRTHDAY,SEX,CITY,COUNTRY,SITE)

values (:a, :b, :c, :d, :e, :f, :g, :h, :i);

end;

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

as begin

insert into FILM (F\_ID, NAME, YEAR\_OF\_CRETION, DURATION, BUDGET, SITE)

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

end;

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

as begin

insert into AWARD (AWARD\_ID,YEAR\_AW,AWARD,RESULT,CATEGORY)

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

end;

create procedure insert\_SOUNDTRACK ( i int, d VARCHAR(255), p int, k int, h time)

as begin

insert into SOUNTRACK (SOUND\_ID, NAME, AUTHOR, FILMS, DURATION)

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

end;

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

as begin

insert into RATING (R\_ID,RATING,E\_MAIL,FILM\_ID)

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

end;

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

as begin

insert into CINEMA (CIN\_ID, NAME, FILM\_ID, SESSION\_ID, PRICE)

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

end;

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

as begin

insert into RENTAL (FILM\_ID,COM\_ID,PRICE)

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

end;

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

as begin

insert into RENTAL\_COMPANY (COM\_ID,NAME,ADDRESS)

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

end;

create procedure insert\_WEBSITE ( i int, d varchar(255), p varchar(255) )

as begin

insert into WEBSITE (S\_ID,NAME,URL)

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

end;

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

as begin

insert into COMPANY (COM\_ID,NAME,YEAR\_OF\_FOUNDATION)

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

end;

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

as begin

insert into PREMIUM (FILM\_ID,MAN\_ID,AWARD\_ID)

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

end;

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

as begin

insert into CAREER (ID,NAME)

values (:i, :d);

end;

create procedure insert\_PRODUCTION ( i int, d int)

as begin

insert into PRODUCTION (FILM\_ID,COM\_ID)

values (:i, :d);

end;

create procedure insert\_OWNER ( i int, d int)

as begin

insert into OWNER (MAN\_ID,COM\_ID)

values (:i, :d);

end;

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

as begin

insert into COUNTRY (C\_ID,NAME)

values (:i, :d);

end;

create procedure insert\_MADE\_IN ( i int, d int)

as begin

insert into MADE\_IN (FILM\_ID,COUNTRY\_ID)

values (:i, :d);

end;

create procedure insert\_STYLE ( i int, d int)

as begin

insert into STYLE (FILM\_ID,GENRE\_ID)

values (:i, :d);

end;

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

as begin

insert into GENRE (ID,NAME)

values (:i, :d);

end;

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

as begin

insert into SESSION (SESSION\_ID,NAME)

values (:i, :d);

end;

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

create procedure update\_BUDGET as

begin

update FILM set FILM.BUDGET = FILM.BUDGET + 5 000 000 where SEASONTICKET\_TYPE.PRICE < 100 000 000;

end;

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

create procedure del\_BUDGET (i int) as

begin

delete from FILM

where FILM.BUDGET = (select max(FILM.BUDGET) from FILM);

end;

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

create procedure del\_FILM as

begin

delete from FILM where F\_ID

not in (select STYLE. FILM\_ID from STYLE);

end;

Удаление фильм , которые не имеет ни какой вид.

**Выполнение индивидуального задания:**

Выполнить следующие запросы:

1. Вывести рейтинг актеров по наградам.
2. Вывести рейтинг стран по бюджетам фильмов.
3. Удалить неиспользуемые жанры.

* Вывести рейтинг актеров по наградам.

create view rat\_ac as select man.name as NAME,count(premium.man\_id) as counter\_pre

from man,premium,part

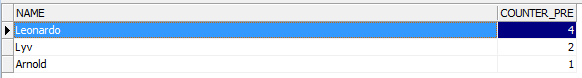
where part.role\_id = 3 and man.man\_id=part.man\_id

and premium.man\_id=man.man\_id

group by man.name

order by counter\_pre desc;

результат:



* Вывести рейтинг стран по бюджетам фильмов.

create view rating\_country as select country.name as name,sum(film.budget) as counter\_budget

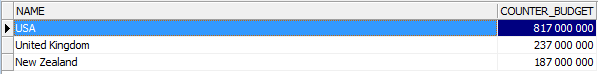
from film,made\_in,country

where country.c\_id = made\_in.country\_id and made\_in.film\_id = film.f\_id

group by country.name

order by counter\_budget desc;

результат:



* Удалить неиспользуемые жанры.

create procedure del\_genre as

begin

delete from GENRE where ID not in (select GENRE\_ID from STYLE);

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.