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 Чинь Тхань Нам

(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**
2. Выборка всех данных из каждой таблицы

connect 'd:/library\_films.fdb' user 'SYSDBA' password 'masterkey';

create view viewershow as select \* from viewer;

create view requestshow as select \* from request;

create view genreshow as select \* from genre;

create view film\_companyshow as select \* from film\_company;

create view film\_personshow as select \* from film\_person;

create view personshow as select \* from person;

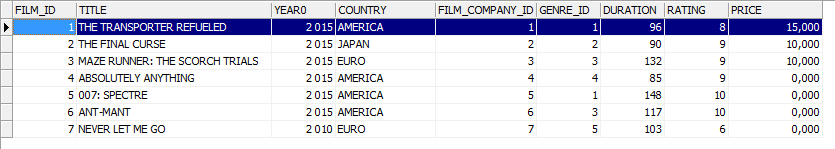
create view charactorshow as select \* from charactor;

create view biletshow as select \* from bilet;

create view cinema1show as select \* from cinema1;

create view filmshow as select \* from film;

Выборка данных из таблицы FILM:



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

create view selectantmant as select \* from film where title like '%ant-mant%';

create view selectduring as select \* from film where duration

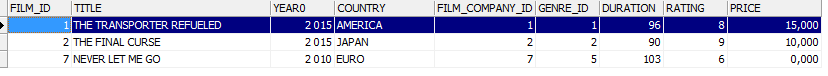
between 90 and 110;

create view selectgerne as select \* from film where genre\_id in (2,3);

Были созданы следующие выборки:

* Из таблицы film показать фильмы с называние ant-mant
* Из таблицы film показать продолжительности от 90 до 110 минут
* Из таблицы film показать жанры с значениями id=2,3;

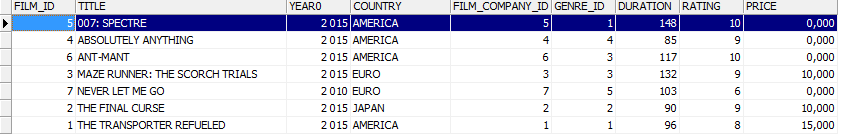
Пример выборки с тренерами с опытом работы от 3 до 10 лет:



1. Выборка всех данных с сортировкой по нескольким полям

create view selectsortfilms as select \* from film order by title asc, price asc;

сотировка фильмов по называниям и ценам.



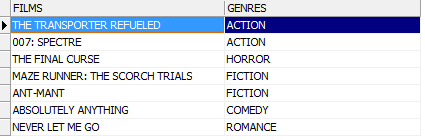
1. Выборка данных из связанных таблиц

create view selectfilm\_genre as select film.title as films,

genre.title as genres from film, genre where genre.genre\_id = film.genre\_id;

Выборки, отображающие:

* Фильмы и жанры:

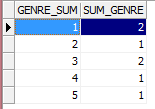


1. Запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

create view selectgenre as select film.genre\_id as genre\_sum,

COUNT(film.genre\_id) as sum\_genre from film group by film.genre\_id;

Определение количества фильмов по жанру:

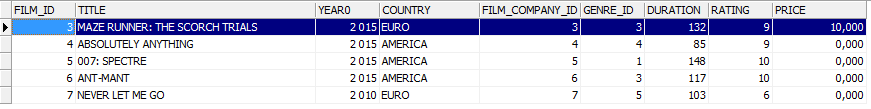


1. Использование вложенного запроса

create view selectUnusedfilm as select \* from film where film\_id

not in (select film\_id from request);

Отображение фильмы, которые не были куплены:



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

connect 'd:/library\_films.fdb' user 'SYSDBA' password 'masterkey';

create procedure insert\_genre (i int, n varchar(32))

as begin

insert into genre (genre\_id,title)

values (:i,:n);

end;

create procedure insert\_filmcom ( i int, t varchar(32), y int, p varchar(255))

as begin

insert into film\_company (film\_company\_id, title, year\_establish,positioning)

values (:i,:t,:y,:p);

end;

create procedure insert\_viewer (v int, n varchar(16), b date, a varchar(255),nu varchar(16))

as begin

insert into viewer (viewer\_id,name, birthday,address,number\_telephone)

values (:v,:n,:b,:a,:nu);

end;

create procedure insert\_film (f int,t varchar(255),y int, c country varchar(16),fi int, g int, d int, r int, p int )

as begin

insert into film (FILM\_ID,TITLE,YEAR0,COUNTRY,FILM\_COMPANY\_ID,GENRE\_ID,DURATION,RATING,PRICE)

values (:f,:t,:y,:c,:fi,:g,:d,:r,:p);

end;

create procedure insert\_request (r int,v int,f int, d int)

as begin

insert into request (request\_id, viewer\_id, film\_id, date\_supply\_request)

values (:r, :v, :f, :d);

end;

create procedure insert\_charactor(c int, a blod, d blod, s blod)

as begin

insert into charactor (character\_id, Actor, director, soundman)

values (:c,:a, :d, :s);

end;

create procedure insert\_person ( p int, n varchar(32),b date, p varchar(16), h integer, c int)

as begin

insert into person (person\_id, name, birthday,place\_birth,hight,character\_id)

values (:p,:n,:b, :p,:h, :c);

end;

create procedure insert\_film\_person ( i int, f int, p int)

as begin

insert into film\_person (id, film\_id,person\_id)

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

end;

create procedure insert\_cinema (c int, t varchar(16),a varchar(255))

as begin

insert into cinema1 (cinema\_id,title,address)

values (:c, :t, :a);

end;

create procedure insert\_bilet(b int, f int, c int, p float, d date)

as begin

insert into bilet (bilet\_id, film\_id, cinema\_id, price, date\_reply)

values (:b,:f,:c,:p,:d);

end;

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

create procedure update\_YEAR (DURA int) as

begin

update FILM set RATING = :DURA where FILM.FILM\_ID >8 ;

end;

Добавление или изменение RATING, у фильма куплено больше есть id>8.

1. С помощью оператора DELETE удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики

create procedure del\_price (i int) as

begin

delete from film where film\_id = :i and

price = (select MIN(price) from film where film\_id = :i);

end;

Данная процедура удаляет цену заданной секции с наименьшей ценой.

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

create procedure del\_film as

begin

delete from film where film\_id not in (select film\_id from bilet);

end;

Удаление фильмов, которых не смотрели в кинотеатрах.

Выполнение индивидуального задания:

Выполнить работу согласно [плану](http://tiger.ftk.spbstu.ru/trac/edu-db-2015/wiki/LabDmlSql).

Реализовать следующие запросы:

1. Отобразить список из 5 наиболее продаваемых фильмов через интернет.
2. Отобразить список из 10 наиболее популярных кинотеатров среди всех фильмов за выбранный период.
3. Удалить неиспользуемые жанры.
4. Отобразить 5 наиболее популярных типов абонементов

create view top5types as select first 5 TICKETTYPES.typename, TICKETTYPES.typename as Name,

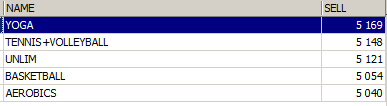
COUNT(SEASONTICKETS.ID\_TYPE) as Sell from TICKETTYPES, SEASONTICKETS

where SEASONTICKETS.ID\_TYPE = tickettypes.ID\_TYPE group by

TICKETTYPES.TYPENAME order by Sell desc;

Для тестирования данного скрипта было сгенирировано 100 000 записей в таблицу SEASONTICKETS.

Результат работы скрипта. 5 самых популярных типов абонементов:



Время выполнения: 530ms

1. Вывести 10 самых непопулярных секций

connect 'C:/SCLUB' user 'SYSDBA' password 'masterkey';

create view unpopular10sectionsver2 as

select

first 10 sectionname as Name,

count (id\_ticket) as Tickets

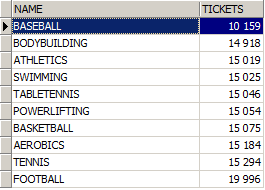
from SECTIONS s

join tickettypestosections t on s.id\_section = t.id\_section

join tickettypes tt on tt.id\_type = t.id\_type

join seasontickets ss on ss.id\_type = tt.id\_type

group by sectionname order by Tickets asc



Время выполнения: 1s 404ms

1. Удалить неиспользуемые типы абонементов

create procedure del\_types as

begin

delete from TICKETTYPES where ID\_TYPE not in (select ID\_TYPE from SEASONTICKETS);

end;

1. **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.