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 Чан Ван Кует

(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.

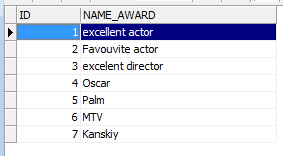
SQL-DML (Data Manipulation Language) -- язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

1. **Выполнение работы**

|  |
| --- |
| connect 'E:/FILMS.FDB' user 'SYSDBA' password 'masterkey';  create view AccountingSel as select \* from award;  create view award\_film\_1as select \* from award\_film;  create view award\_human\_1 as select \* from award\_human;  create view film\_1 as select \* from film;  create view film\_human\_1 as select \* from award;  create view film\_ticket\_1 as select \* from film\_ticket;  create view genres\_1 as select \* from genres;  create view genres\_film\_1 as select \* from genres\_film;  create view human\_1 as select \* from human;  create view role\_1 as select \* from role;  create view studio\_1 as select \* from studio; |

1. Выборка всех данных из каждой таблицы

Выборка данных из таблицы AWARD:



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

|  |
| --- |
| connect 'E:/FILMS.FDB' user 'SYSDBA' password 'masterkey';  create view selectQuyet111 as select \* from HUMAN where NAME like '%Quyet%';  create view HUMAN\_2 as select \* from HUMAN where ID between 3 and 10;  create view selectgenres\_film\_1 as select \* from GENRes\_film where id\_genres in (3); |

Были созданы следующие выборки:

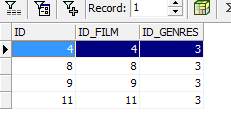
* Из таблицы HUMAN показать люди с фамилией



* Из таблицы HUMAN показать номер ID от 3 до 10



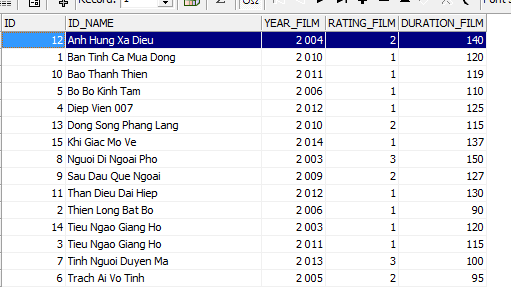
* Из таблицы GENRES\_FILM показать секции, которые расположены с id\_genres\_film = 3



1. Выборка всех данных с сортировкой по нескольким полям

|  |
| --- |
| create view selectFILM\_2 as select \* from film order by id\_name asc, year\_film asc; |

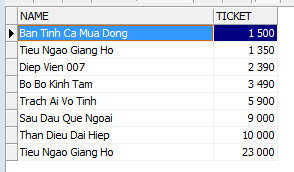
Результат ID\_NAME по алфавиту



1. Выборка данных из связанных таблиц

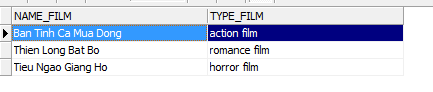
|  |
| --- |
| create view selectFilm\_human\_5  as select FILM.id\_name as name,  film\_ticket.ID\_ticket as ticket from film,film\_ticket where film.id = film\_ticket.ID\_film; |

Выборки, отображающие:



|  |
| --- |
| create view selectfilm\_7  as select film.id\_name as name\_film,  genres.genres as type\_film from film,genres where film.id = genres.id; |

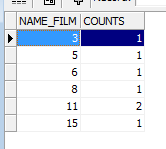
Выборки, отображающие:



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

|  |
| --- |
| create view selectCountawqrd\_film1 as select award\_film.id\_film as name\_film, COUNT(award\_film.id\_film) as Counts from award\_film group by award\_film.id\_film ; |

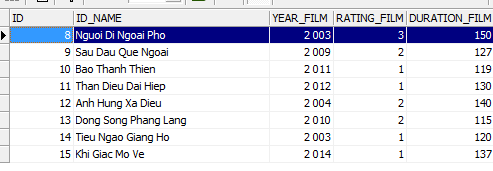
Определение количества:



1. Использование вложенного запроса

|  |
| --- |
| create view selectUnusedFilm as select \* from film where ID  not in (select id from award\_film); |

Результат:



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

connect 'E:/FILMS.FDB' user 'SYSDBA' password 'masterkey';

create procedure address\_studio ( i int, n varchar(32))

as begin

insert into STUDIO (id , studio)

values (:i, :n);

end;

create procedure insert\_film ( i int, n varchar(32) , t int , e int ,f int ,g int ,h inf )

as begin

insert into FILM (id , id\_name , year\_film, rating\_film,duration\_film,studio\_film,genres\_film)

values (:i , :n , :t , :e ,:f , :g , :h);

end;

create procedure insert\_film\_ticket ( i int, t int)

as begin

insert into FILM\_TICKET (id, id\_ticket)

values (:i, :t );

end;

create procedure insert\_award\_film ( i int, :t int, c int)

as begin

insert into AWARD\_FILM (id,id\_film, id\_award)

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

end;

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

as begin

insert into ROLE (id, name)

values (:i, :n);

end;

create procedure insert\_film\_human ( i int, t int, f int ,d int)

as begin

insert into FILM\_HUMAN (id ,id\_human,id\_film,id\_role)

values (:i, :n,:f,:d);

end;

create procedure insert\_genres\_film ( i int, d int, f int)

as begin

insert into GENRES\_FILM (id , id\_film, id\_genres)

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

end;

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

as begin

insert into GENRES (id, genres)

values (:i, :n);

end;

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

as begin

insert into HUMAN (id,name)

values (:i, :n);

end;

create procedure insert\_award\_human( i int, s int ,t int)

as begin

insert into AWARD\_HUMAN (id, id\_human,id\_award)

values (:i, :s ,:t);

end;

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

as begin

insert into AWARD (id , name\_award)

values (:i, :n);

end;

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

|  |
| --- |
| create procedure update\_disc (disc int) as  begin  update FILM set RATING\_FILM = :disc where FILM.ID > 10;  end; |

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

create procedure del\_exptrainer (i int) as

begin

delete from FILM where ID = :i and

DURATION\_FILM = (select MIN(DURATION\_FILM) from FILM where

ID = :);

end;

.

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

create procedure del\_types as

begin

delete from FILM where ID not in (select ID from AWARD);

end;

Выполнение индивидуального задания:

Реализовать следующие запросы:

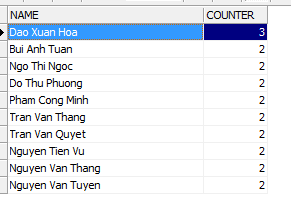
1. Вывести 10 наиболее популярных (по сборам) режиссеров.
2. Вывести фильмы с максимальным количеством актеров.
3. Удалить неиспользуемые студии.

**Вывести 10 наиболее популярных (по сборам) режиссеров**.

|  |
| --- |
| create view top10ridector as select first 10 human.name ,  COUNT(FILM\_HUMAN.id\_human ) as counter from HUMAN , film\_human  where FILM\_HUMAN.id\_human = human.id group by  HUMAN.name order by counter desc; |

Результат работы скрипта. 10 наиболее популярных (по сборам) режиссеров.

:



Время выполнения: 63ms

1. **Вывести фильмы с максимальным количеством актеров**

|  |
| --- |
| create view top2 as select first 2 film.id\_name ,  COUNT(FILM\_HUMAN.id\_film ) as counter from FILM , FILM\_HUMAN  where FILM\_HUMAN.id\_film = film.id group by  film.id\_name order by counter desc; |

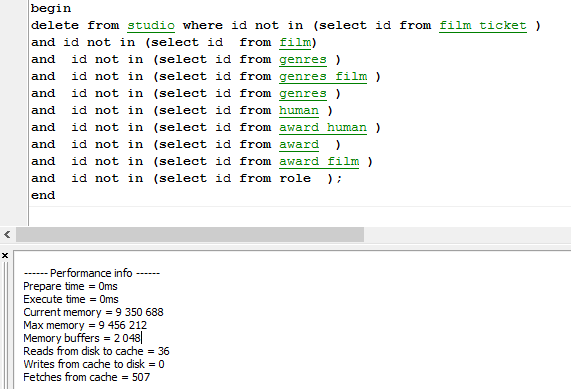


Время выполнения: 47 ms

1. **Удалить неиспользуемые студии.**

|  |
| --- |
| connect 'E:/FILMS.FDB' user 'SYSDBA' password 'masterkey';  create procedure del\_studio1 as  begin  delete from studio where id not in (select id from film\_ticket )  and id not in (select id from film)  and id not in (select id from genres )  and id not in (select id from genres\_film )  and id not in (select id from genres )  and id not in (select id from human )  and id not in (select id from award\_human )  and id not in (select id from award )  and id not in (select id from award\_film )  and id not in (select id from role );  end ; |

Результат выполнения процедуры:



1. **Вывод**

В результате выполнения работы был изучен язык управления данными SQL-DML. Это семейство компьютерных языков применимо для получения информации из баз данных, вставки, удаления и изменения данных при работе с базами данных. Так как язык стандартизирован, выполнять эти операции относительно удобно для решения задач различной сложности.

В ходе выполнения лабораторной работы 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.