Санкт-Петербургский Политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт о лабораторной работе №3**

**Дисциплина**: Базы данных

**Тема**: Язык SQL-DML

Выполнила студентка гр. 43501/1 Ракова В.В.

(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:\Litpub\ONE.fdb' user 'SYSDBA' password 'masterkey';  create view AuthorSel as select \* from author;  create view Author\_bookSel as select \* from author\_book;  create view BookSel as select \* from BOOK;  create view EditionSel as select \* from edition;  create view GenreSel as select \* from GENRE;  create view Genre\_bookSel as select \* from genre\_book;  create view PrizeSel as select \* from prize;  create view Rating\_salarySel as select \* from rating\_salary;  create view RedactorSel as select \* from redactor;  create view Redactor\_bookSel as select \* from redactor\_book;  create view ResellerSel as select \* from reseller;  create view Reseller\_editionSel as select \* from reseller\_edition; |

Выборка данных из таблицы AUTHOR:



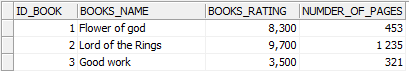
1. Выборка данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN

|  |
| --- |
| create view selectIvanov as select \* from AUTHOR where AUTORS\_NAME like '% Ivanov %';  create view selectBookPage as select \* from BOOK where NUMDER\_OF\_PAGES  between 300 and 1500;  create view selectRedBook as select \* from redactor\_book where "ID\_REDACTOR" in (2,3); |

Были созданы следующие выборки:

1. Из таблицы AUTHOR показать авторов с фамилией Ivanov
2. Из таблицы BOOK показать книги с количеством страниц от 300 до 1500
3. Из таблицы REDACTOR\_BOOK показать книги у которых редакторы с id = 2 или 3

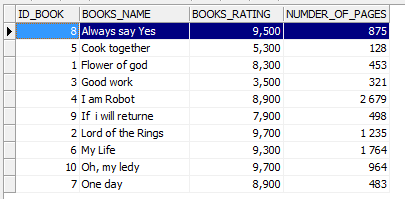
выборки с книгами с количеством страниц от 300 до 1500:



1. Выборка всех данных с сортировкой по нескольким полям

create view selectSortBooks as select \* from BOOK order by BOOKS\_NAME asc, BOOKS\_RATING asc;

Книги отсортированные по названию и рейтингу:



1. Выборка данных из связанных таблиц

create view selectBookAuthor as select BOOK.books\_name as Books\_name,

AUTHOR.autors\_name as authors\_name from BOOK, AUTHOR where AUTHOR.id\_author = BOOK.id\_book;

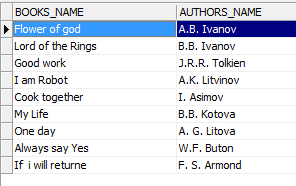
create view selectRedactorBook as select BOOK.books\_name as books\_name,

REDACTOR.redactor\_name as redactor\_name, BOOK.books\_rating as rating from BOOK,

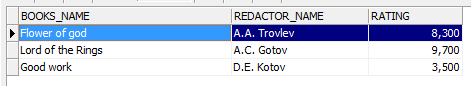
REDACTOR where REDACTOR.id\_redactor = BOOK.id\_book;

Выборки, отображающие:

* Авторов и книги:



* Редакторов и книги с рейтингом:

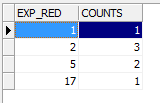


1. Запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

create view selectCountExperienceRedactor as select REDACTOR.experience\_red as exp\_red,

COUNT(redactor.experience\_red) as Counts from REDACTOR group by REDACTOR.experience\_red;

Определение количества редакторов по стажу работы:

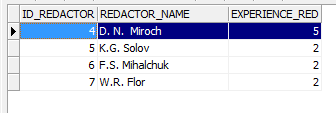


1. Использование вложенного запроса

create view selectfreeRedactor as select \* from REDACTOR where ID\_REDACTOR

not in (select ID\_REDACTOR from REDACTOR\_BOOK);

Отображение редакторов, не курирующих ни одной книги:



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

|  |
| --- |
| create procedure insert\_author ( i int, n varchar(255), d date)  as begin  insert into AUTHOR(id\_author, autors\_name, date\_of\_born)  values (:i, :n, :d);  end;  create procedure insert\_author\_book ( ic int, ia int, ib int )  as begin  insert into AUTHOR\_BOOK (id\_comab, id\_author, id\_book)  values (:ic, :ia, :ib);  end;  create procedure insert\_book ( i int, n varchar(255), r float, p int)  as begin  insert into BOOK (id\_book, books\_name, books\_rating, numder\_of\_pages)  values (:i, :n, :r, :p);  end;  create procedure insert\_edition ( i int, n int, b int, ni int)  as begin  insert into EDITION (id\_edition, number\_of\_copy, id\_book, number\_of\_issue)  values (:i, :n, :b, :ni);  end;  create procedure insert\_genre ( i int, n varchar(255))  as begin  insert into GENRE (id\_genre, genre\_name)  values (:i, :n);  end;  create procedure insert\_genre\_book ( ic int, ig int, ib int)  as begin  insert into genre\_book (id\_comgb, id\_genre, id\_book)  values (:ic, :ig, :ib);  end;  create procedure insert\_prize ( i int, id int, s int, ia int)  as begin  insert into PRIZE (id\_prize, id\_edition, prize\_sum, id\_author)  values (:i, :id, :s, :ia);  end;  create procedure insert\_rating\_salary ( i int, r float, e int, s int , p varchar(255))  as begin  insert into rating\_salary (id\_author, rating\_author, experience\_author, salary\_author, prize\_was)  values (:i, :r, :e, :s, :p);  end;  create procedure insert\_redactor ( i int, n varchar(255), e int)  as begin  insert into REDACTOR (id\_redactor, redactor\_name, experience\_red)  values (:i, :n, :e);  end;  create procedure insert\_redactor\_book( i int, r int, b int)  as begin  insert into redactor\_book (id\_comrb, id\_redactor, id\_book)  values (:i, :r, :b);  end;  create procedure insert\_reseller ( i int, n varchar(255), d varchar(255), p int)  as begin  insert into RESELLER (id\_reseller, reseller\_name, director\_name, phone\_number)  values (:i, :n, :d, :p);  end;  create procedure insert\_reseller\_edition ( i int, e int, n int)  as begin  insert into reseller\_edition (id\_reseller, id\_edition, number\_partcopy)  values (:i, :e, :n);  end; |

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

create procedure update\_sal (sal int) as

begin

update rating\_salary set SALARY\_AUTHOR = :sal where rating\_salary.rating\_author > 4.0;

end;

Добавление или изменение зарплаты авторов если их рейтинг выше 4.0

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

create procedure del\_worseadition (i int) as

begin

delete from EDITION where ID\_BOOK = :i and

NUMBER\_OF\_COPY = (select MIN(NUMBER\_OF\_COPY) from EDITION where ID\_BOOK = :i);

end;

Данная процедура удаляет тираж заданной книги с наименьшим объемом выпуска.

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

create procedure del\_author as

begin

delete from AUTHOR where ID\_AUTHOR not in (select ID\_AUTHOR from AUTHOR\_BOOK);

end;

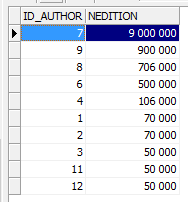
Данная процедура удаляет авторов , не написавших ни одной книги.

11) Реализовать следующие запросы:

1. Вывести 10 наиболее популярных (по объему тиражей) авторов.
2. Вывести книги с максимальным количеством авторов.
3. Удалить неиспользуемые жанры.

1.Вывести 10 наиболее популярных (по объему тиражей) авторов.

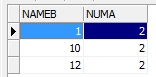
|  |
| --- |
| create view top10autor1 as select first 10  author.id\_author ,  SUM(edition.number\_of\_copy) as Nedition  from book, author , edition, author\_book  where book.id\_book = author\_book.id\_book AND  author.id\_author =author\_book.id\_author AND  book.id\_book = edition.id\_book  group by author.id\_author  order by Nedition desc; |



В результате получили 10 самых популярных авторов.

2. Вывести книги с максимальным количеством авторов.

|  |
| --- |
| create view booksmaxauthors1 as select first 3  book.id\_book as NameB ,  COUNT(author.id\_author) as NumA  from book, author , author\_book  where book.id\_book = author\_book.id\_book AND  author.id\_author =author\_book.id\_author  group by book.id\_book  order by NumA desc ; |



3.Удалить неиспользуемые жанры.

|  |
| --- |
| create procedure del\_genre as  begin  delete from GENRE where ID\_GENRE not in (select ID\_GENRE from genre\_book);  end; |

**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.