Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе №3

По дисциплине «Базы данных»

«Язык SQL-DML»

Работу выполнили студенты группы №43501/4 Климова Д.А. _____

Работу принял преподаватель_____ Мяснов А.В._____

Санкт-Петербург

2015

- ## Цель работы

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

- ## Программа работы

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

Список стандартных запросов

- Сделайте выборку всех данных из каждой таблицы
- Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, `LIKE`, `BETWEEN`, `IN` (не менее 3-х разных примеров)
- Создайте в запросе вычисляемое поле
- Сделайте выборку всех данных с сортировкой по нескольким полям
- Создайте запрос, вычисляющий несколько совокупных характеристик таблиц
- Сделайте выборку данных из связанных таблиц (не менее двух примеров)
- Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки
- Придумайте и реализуйте пример использования вложенного запроса
- С помощью оператора `INSERT` добавьте в каждую таблицу по одной записи
- С помощью оператора `UPDATE` измените значения нескольких полей у всех записей, отвечающих заданному условию
- С помощью оператора `DELETE` удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики

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

- **Ход работы**

- SQL-DML изучен в теории по методическим указаниям приведенным на ресурсе “trac”.
- Создание стандартных запросов.

- Сделайте выборку всех данных из каждой таблицы

```
create view as_books as select* from books;
create view as_authors as select* from authors;
create view as_authors as select* from authorid;
create view as_book_category as select* from book_category;
create view as_category as select* from category;
create view as_clients as select* from clients;
create view as_composition as select* from composition;
create view as_package as select* from package;
create view as_publishers as select* from publishers;
create view as_review as select* from review;
```

- Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее 3-х разных примеров)

```
create view z2_1 as select bookid, clientid, review from review where bookid between 1 and 2 and clientid like '1%';
create view z2_2 as select clientid, delivery_date from package where clientid in (1) and delivery_date between '05.12.2015' and '30.12.2015';
create view z2_3 as select* from authors where booksid between 1 and 4 and authors like '3%';
```

- Создайте в запросе вычисляемое поле

```
create view z5
```

```
as
select package.delivery_date as delivery_date, sum(package.price) as price from
package
where package.delivery_date between '29.12.2015' and '29.01.2016' group by
package.delivery_date;
```

- Сделайте выборку всех данных с сортировкой по нескольким полям

```
create view z4 as select* from books order by bookname, price;
```

- Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

```
create view z3 as select sum(price) as sum_price, count(bookid) as book_amount
from books;
```

- Сделайте выборку данных из связанных таблиц (не менее двух примеров)

```
create view z6_1 as select books.bookname as book, authorid.author as author,
authors.rating as rating from books, authors, authorid where authors.booksid =
books.bookid and authors.author = authorid.authorid;


create view z6_2 as select books.bookname as book, clients.clientname as client,
package.delivery_date as d_date from package, clients, composition, books where
clients.clientid = package.clientid and package.packageid = composition.pa
ckageid and composition.bookid = books.bookid;
```

- Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

```
CREATE OR ALTER VIEW Z7
AS
select package.packageid as packageid , count(composition.bookid) as
book_amount
from package, composition
```

```
where package.packageid = composition.packageid
group by package.packageid
having count (composition.bookid) > 10;
```

- Придумайте и реализуйте пример использования вложенного запроса

```
CREATE OR ALTER VIEW Z8
AS
select books.bookname as book, books.price as price, books.amount,
publishers.publish_year
from books, publishers
where books.bookid = publishers.bookid
and publishers.publish_year =(select max(publishers.publish_year) from
publishers);
```

- С помощью оператора `INSERT` добавьте в каждую таблицу по одной записи

```
create procedure in_review(i INTEGER, i2 INTEGER, c CHAR)
        as begin INSERT INTO review VALUES (:i,:i2,:c);
end;
create procedure in_publishers(i INTEGER, c CHAR, i2 INTEGER)
        as begin INSERT INTO publishers VALUES (:i,:c,:i2);
end;
create procedure in_package(i INTEGER, i2 INTEGER, d DATE)
        as begin INSERT INTO package VALUES (:i,:i2,:d);
end;
create procedure in_composition(i INTEGER, i2 INTEGER)
        as begin INSERT INTO composition VALUES (:i,:i2);
end;
create procedure in_clients (i INTEGER, c CHAR,  i2 INTEGER, i3 INTEGER)
        as begin INSERT INTO clients VALUES (:i, :c, :i2, :i3);
end;
create procedure in_category(i INTEGER, c CHAR)
        as begin INSERT INTO category VALUES (:i,:c);
end;
```

```
create procedure in_book_category(i INTEGER, i2 INTEGER)
        as begin INSERT INTO book_category VALUES (:i,:i2);
end;
create procedure in_authorid(i INTEGER, c CHAR)
        as begin INSERT INTO authorid VALUES (:i,:c);
end;
create procedure in_authors(i INTEGER, i2 INTEGER, i3 INTEGER)
        as begin INSERT INTO authors VALUES (:i,:i2,: i3);
end;
create procedure in_books (i INTEGER, c CHAR,  i2 INTEGER, i3 INTEGER)
        as begin INSERT INTO books VALUES (:i, :c, :i2, :i3);
end;
```

```
 insert into review  values (4,2,'review');
 insert into publishers values (5,'publisher',1976);
 insert into authorid values (4,'Author4');
 insert into package values (104,1,'28.06.2016',290,1);
 insert into composition values (104,5);
 insert into clients values(3,'ThirdClient',12, 298);
 insert into category values(4, 'category №4');
 insert into book_category values(6,4);
  insert into books values (9, 'NinthBook', 369,12);
  insert into authors values (9,3,8);
 commit;
```

- С помощью оператора `UPDATE` измените значения нескольких полей у всех записей, отвечающих заданному условию

```
create or alter procedure UPDATE_DELIVERY_DATE (
   D1 date,  D2 date)
as
```

```
begin
  update package
  set package.delivery_date = :d1
  where package.delivery_date = :d2;
end
```

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

```
create or alter procedure IN_DELETE (  I integer)
as
begin
  delete from publishers
  where publishers.publish_year = (select min(publishers.publish_year) from
publishers where publishers.bookid = :i)
  and publishers.bookid = :i;
end
```

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

```
create procedure in_delete_category
        as begin delete from category
            where categoryid not in (select categoryrid from book_category);
end;
```

- **Выполнение индивидуальных заданий**

- Вывести всех авторов, средний рейтинг которых не менее, чем по трем категориям выше заданного.

```
CREATE OR ALTER VIEW AUTHOR_RATING(
   AUTHOR,
```

```
   CAT,
   RATING)
AS
select authorid.author as author, category.categoryid as cat, avg(authors.rating) as
rating
from authorid, authors, book_category, category
where authorid.authorid = authors.author
and authors.booksid = book_category.bookid
and book_category.categoryid = category.categoryid
group by authorid.author, category.categoryid;
```

```
CREATE OR ALTER VIEW AUTHOR_RATING_(
   AUTHOR)
AS
select author_rating.author
from author_rating
where author_rating.rating > 4
group by  author_rating.author
having count(author_rating.cat) > 3;
```

- Вывести среднюю стоимость заказа для каждой категории книг

```
CREATE OR ALTER VIEW AVG_SUM(
   CATEGORY_,
   PRICE)
AS
select category.category  as category_, avg(package.price) as price
from category,  book_category, composition, package
where category.categoryid = book_category.categoryid
and book_category.bookid = composition.bookid
and composition.packageid = package.packageid
group by category.category;
```

- Удалить неиспользуемые категории.

- ## Создание представлений и хранимых процедур

В ходе работы все запросы уже были сохранены в виде представлений, а выполняемые действия в виде процедур.

- ## Выводы

В данной работе изучен язык SQL-DML. Получены навыки работы с запросами. В данной работе мы научились извлекать необходимые данные из таблиц и представлять их в нужной форме. Также мы научились создавать представления и процедуры.

Представления дают нам возможность создавать некий интерфейс для разных типов пользователей. Они позволяют пользователю выполнять все необходимые запросы как запросы к обычным таблицам, но при этом не позволяют ему изменять данные. Таким образом получается некоторая система безопасности.

Хранимые процедуры дают возможность выполнять заранее прописанные действия, возможно, с некоторым изменением параметров. Это удобно, когда при работе необходимо часто выполнять схожие действия, требующие некоторого количества строк кода.