Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчет по лабораторной работе №4**

по дисциплине «Базы данных»

Язык SQL-DML

Выполнил

студент гр. 43501/3 Е.А. Никитин

Преподаватель А.В. Мяснов

«\_\_»\_\_\_\_\_\_\_\_\_2015г.

Санкт-Петербург

2015

**Цели работы**

Изучение SQL-DML, ознакомление с основными методами создания запросов

**Программа работы**

1. Изучите SQL-DML
2. Выполните все запросы из списка стандартных запросов. Продемонстрируйте результаты преподавателю.
3. Получите у преподавателя и реализуйте SQL-запросы в соответствии с **индивидуальным** заданием. Продемонстрируйте результаты преподавателю.
4. 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:

1. **SQL** — [формальный](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной [реляционной базе данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей системой управления базами данных ([СУБД](https://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94)).

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. Выполнить все запросы из списка стандартных запросов.

Сделайте выборку всех данных из каждой таблицы

create view car\_typeV as select \* from car\_type;

create view ordersV as select \* from orders;

create view passengerV as select \* from passenger;

create view placev as select \* from place;

create view place\_typeV as select \* from place\_type;

create view route\_stationV as select \* from route\_station;

create view sheduleV as select \* from shedule;

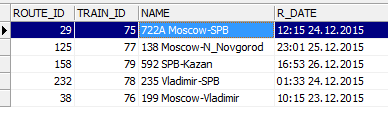
create view stationV as select \* from station;

create view ticketV as select \* from ticket;

create view trainV as select \* from train;

create view t\_orderV as select \* from t\_order;

commit;



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

create view stationVUSL as select \* from station where st\_id like 1;

create view passengerVUSL as select \* from passenger where

pas\_id between 36 and 38;

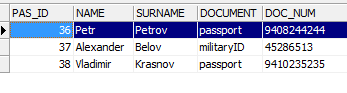
create view trainVUSL as select \* from train where train\_id in (75, 76);

commit;

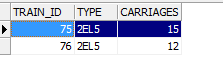
Выборка данных из таблицы station при условии, что st\_id=1

C:\Users\v\Desktop\ss+(2015-12-12+at+06.20.10).png

Выборка данных из таблицы passenger при соответствии значений pas\_id от 36 до 38.



Выборка данных из таблицы train при соответствии значений train\_id 75 или 76.



* Создайте в запросе вычисляемое поле

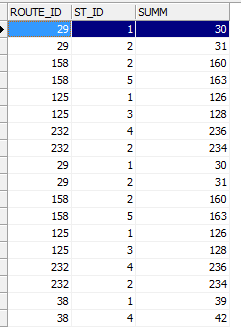
Для примера сложим значения полей route\_id и st\_id таблицы route\_station:

create view vich\_pole as

select route\_id, st\_id, route\_id+st\_id as summ from route\_station;

commit;

Результат:



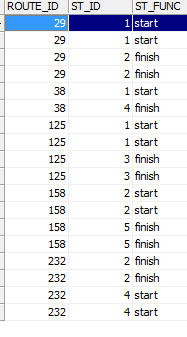
* Сделайте выборку всех данных с сортировкой по нескольким полям

create view v\_sort as select \* from route\_station

order by route\_id asc, st\_id asc;

commit;

Результат:



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

create view svyaz\_tables as select route\_station.st\_func as func,

station.name as name, route\_station.num\_in\_route as num

from route\_station, station

where route\_station.st\_id=station.st\_id;

create view svyaz\_tables1 as select place.carriage as carriage,

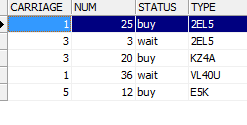
place.num\_in\_car as num, ticket.status, train."TYPE" from ticket, place, train

where place.place\_id=ticket.place\_id and train.train\_id=place.train\_id;

commit;

Результаты:





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

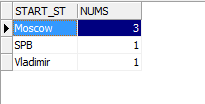
Расчет количества билетов по начальной станции.

create view grouping as select ticket.start\_st as start\_st,

count(ticket.start\_st) as nums from ticket group by ticket.start\_st;

commit;

Результат:



* Придумайте и реализуйте пример использования вложенного запроса

Вывод мест в поезде с id=75, на которые еще не оформлены билеты:

create view vlozh as select

place\_id as place\_id, num\_in\_car as num\_in\_car from place

where place.place\_id not in (select place\_id from ticket) and place.train\_id=75;

commit;

http://puu.sh/lSuVp/9fb182a2ad.png

* С помощью оператора INSERT добавьте в каждую таблицу по одной записи

Создание процедур:

create procedure ins\_station (i INTEGER, n char(255)) as begin

insert into station (st\_id, name) values (:i, :n);

end;

create procedure ins\_passenger (i INTEGER, n char(255), s char(255), d char(255),

num char(255)) as begin

insert into passenger (pas\_id, name, surname, document, doc\_num)

values (:i, :n, :s, :d, :num);

end;

create procedure ins\_train (i INTEGER, t char(255), c INTEGER) as begin

insert into train (train\_id, "TYPE", carriages) values (:i, :t, :c);

end;

create procedure ins\_place (i INTEGER, ti integer, c integer,

num INTEGER, ct integer, pt integer) as begin

insert into place (place\_id, train\_id, carriage, num\_in\_car, car\_type\_id,

place\_type\_id) values (:i, :ti, :c, :num, :ct, :pt);

end;

create procedure ins\_ticket (tn INTEGER, i integer, status char(255),

sta char(255), sto char(255)) as begin

insert into ticket (t\_num, place\_id, status, start\_st, stop\_st)

values (:tn, :i, :status, :sta, :sto);

end;

create procedure ins\_shedule (ri INTEGER, ti integer, n char(255),rd char(255))

as begin

insert into shedule (route\_id, train\_id, name, r\_date)

values (:ri, :ti, :n, :rd);

end;

create procedure ins\_r\_station (si INTEGER, ri integer, f char(255),

stime char(255), num integer)

as begin

insert into route\_station (route\_id, st\_id, st\_func, stop\_time, num\_in\_route)

values (:si, :ri, :f, :stime, :num);

end;

create procedure ins\_t\_order (oi INTEGER, tnum integer, tsnum integer)

as begin

insert into t\_order (order\_id, t\_num, tickets\_num)

values (:oi, :tnum, :tsnum);

end;

create procedure ins\_orders (i INTEGER, oi integer)

as begin

insert into orders (pas\_id, order\_id)

values (:i, :oi);

end;

Использование процедур для заполнения:

execute procedure ins\_passenger

(34, 'Sergey', 'Smirnov', 'passport', '9809999999');

execute procedure ins\_station (100002, 'Tver');

execute procedure ins\_train (74, 'E5K', 12);

execute procedure ins\_place (78, 74, 2, 24, 0, 0);

execute procedure ins\_ticket (88, 78, 'buy', 'Moscow', 'Tver');

execute procedure ins\_shedule (177, 74, '654 Moscow-Tver', '27.12.2015');

execute procedure ins\_r\_station (177, 1, 'start', '-', 1);

execute procedure ins\_t\_order (7546, 88, 1);

execute procedure ins\_orders (34, 7546);

commit;

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

Создадим процедуру, которая изменяет имя и фамилию пассажиров, чей id от 55 до 63:

create procedure udpdt (n char(255), s char(255)) as begin

update passenger set name = :n, surname = :s

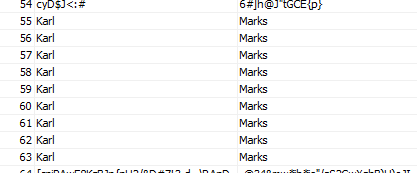
where passenger.pas\_id between 55 and 63;

end;

Используем процедуру:

execute procedure udpdt ('Karl', 'Marks');

commit;



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

Удалим запись с входным параметром-именем n и минимальным id из таблицы passenger:

create procedure del1 (n char(255)) as begin

delete from passenger where name= :n and pas\_id=(select MIN(pas\_id)

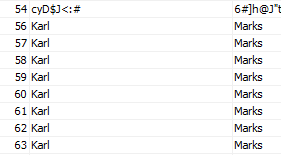
from passenger where name= :n);

end;

Использование процедуры:

execute procedure del1('Karl');

commit;



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

Удалим пользователей, у которых нет заказов. Сейчас в таблице passenger 100000 записей.

create procedure del2 as begin

delete from passenger where pas\_id

not in (select pas\_id from orders);

end;

Используем процедуру:

execute procedure del2;

commit;



Задания, выданные преподавателем:

1. Удалить неиспользуемые станции.

Сейчас в таблице station 100001 запись. Удалим не использующиеся:

create procedure del\_station as begin

delete from station where st\_id

not in (select st\_id from route\_station);

end;

Используем процедуру:

execute procedure del\_station;

commit;

Результат:



1. Отобразить 10 наиболее популярных маршрутов.

create view top10routes as select first 10

shedule.name as name, place.train\_id as train\_id, count(place.train\_id) as nums

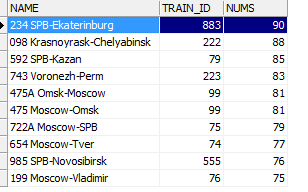
from shedule, place

where shedule.train\_id=place.train\_id group by shedule.name, place.train\_id

order by nums desc;

commit;

Результат:



1. Отобразить 5 станций, на которых больше всего входят и выходят пассажиры.

create view top5stations

as

select first 5 T1.st as st\_name, T1.num + T2.num\_start as summ from

(select ticket.stop\_st as st, count(all ticket.stop\_st) as num

from ticket

group by st order by num desc)T1,

(select ticket.start\_st as st\_start, count(all ticket.start\_st) as num\_start

from ticket

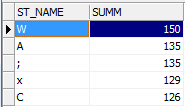
group by st\_start order by num\_start desc)T2

where T1.st = T2.st\_start

order by summ desc;

commit;

Результат:



Запросы проверены на большом объеме записей (100000).

Вывод.

В ходе лабораторной работы был изучен язык SQL-DML и основные методы создания запросов, использование условий, логических операций.

Также был приобретен опыт работы с хранимыми процедурами, которые позволяют использовать скрипты написанные ранее командой вызова процедуры, что удобно, если процедуру нужно выполнить много раз (не нужно открывать скрипт и выполнять вручную).

Были изучены представления, которые можно сформировать в удобном для себя виде и которые будут хранить нужные на данный момент данные. Представление является отображением данных из основных таблиц, сформированным в нужном виде.