Санкт-Петербургский государственный политехнический университет

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

Отчет по лабораторной работе

По дисциплине «Базы данных»

«Язык SQL-DML»

Работу выполнил студент группы № 43501/4 Смирнов Л.А.

Работу принял преподаватель Мяснов А.А.

Санкт-Петербург

2015

Цель работы:

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

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)

Выполнение работы:

Выполним запросы из списка стандартных запросов, приведенных ниже.

1) Сделаем выборку всех данных из каждой таблицы

Этот запрос реализуется с помощью операции SELECT. Если мы хотим выбрать определенный атрибут, то синтаксис следующий: SELECT столбец FROM название таблицы;
Если мы хотим выбрать все данные из таблицы, то синтаксис следующий: SELECT * FROM название таблицы
Пример: SELECT * FROM Activities

ID_FILE	ID_ACTIV...	ACTIVITY_DATE
100	2	11.09.2013
100	5	23.04.2013
100	6	25.09.2013
101	2	24.09.2013
101	3	11.09.2013
102	2	11.09.2013
103	1	11.09.2013
103	2	11.09.2013
103	5	11.09.2013
103	4	11.09.2013
104	1	11.09.2013
105	2	11.09.2013
106	3	11.09.2013
107	4	11.09.2013
108	5	11.09.2013
109	2	11.09.2013
110	2	11.09.2013

2) Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее трех разных примеров).

В том случае, когда нас интересуют не все записи, а только те, котрые удовлетворяют некому условию, это условие можно указать после ключевого слова WHERE.

```
SELECT * FROM Recruit WHERE Birth_date>='1990-01-01' AND Birth_date <='1991-01-01'; - выведет всех призывников, у кого дата рождения в данных временных рамках.
```

Логическая операция проверки на вхождение в список:

```
SELECT * FROM Activities WHERE Id_activity IN (3,4); - выведет личные дела всех призывников, кто прошел медицинский осмотр и прибыл на сборы.

SELECT * FROM Recruit WHERE Name like "Исаков*"; - выведет всех «Исаковых» из таблицы призывников.
```

3) Создайте в запросе вычисляемое поле

```
SELECT Registr_date, id_file - 105 FROM Personal_file where id_file > 105 ; - выведет все дела с датами регистрации после дела с номером №105
```

4) Создайте выборку всех данных с сортировкой по нескольким полям

```
Select * from Activities order by id_activity; - выведет результат с отсортированным полем активности призывника
```

5) Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

```
select p1.Id_file, p2.Name from personal_file as p1, recruit as p2 where p1.id_recruit = p2.id_recruit; - делает выборку из таблицы личного дела и призывника. Выводит номер дела и ФИО призывника
```

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

```
select p1.id_recruit, p2.activity_date, p3.kind_activity from Personal_file as p1, activities as p2, kind_activity as p3 where p1.id_file = p2.id_file and p2.id_activity = p3.id_activity; - выводит склеенными поля призывников, активность призывника и дату активности.

select p1.id_file, p2.Name, p3.Name from personal_file as p1, recruit as p2, academic_place as p3 where p1.id_recruit = p2.id_recruit and p2.id_academic = p3.id_academic; - выводит склеенными поля поля ид_личного дела, ФИО призывника и названия учебного заведения.
```

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

```
select personal_file.id_file, count(activities.id_activity) from personal_file, activities where personal_file.id_file = activities.id_file group by id_file; - сгруппировать номера личных дел и количеств активностей призывника
```

8) Придумайте и реализуйте пример использования вложенного запроса

```
select id_file from personal_file where id_recruit in (select p1.id_recruit from recruit as p1, academic_place as p2 where p1.Id_academic = p2.Id_academic and p2.Name = 'SPBAU'); - выводит номера дел призывников, которые учатся в СПБАУ
```

9) С помощью оператора INSERT добавьте в каждую таблицу по одной записи

```
Insert into recruit() values();
```

10) С помощью оператора UPDATE измените значение нескольких полей у всех записей отвечающих заданному условию

```
update recruit set Registr_place = 'Spb, Zvezdnaya street, 14' where
id_recruit = 12; - обновит место регистрации призывника с id = 12
```

Индивидуальное задание:

1) Вывести 10 образовательных учреждений, которые выдают наибольшее количество отсрочек.
2) Вывести 5 призывников, которые провели наибольшее количество дней в мероприятиях не связанных с призывом.
3) Удалить неиспользуемые причины отсрочек.

1) Вывести 10 образовательных учреждений, которые выдают наибольшее количество отсрочек.

```
select p1.name, count(p1.id_academic) from academic_place as p1, recruit as
p2
where p1.id_academic = p2.id_academic and p2.id_recruit in
(
    select p1.id_recruit from Personal_file as p1, delays as
    p2, reasons as p3 where  p2.id_reason = p3.id_reason
    and p3.reason_text = 'Delay University' and p1.id_file = p2.id_file
) group by p1.name order by count(p1.id_academic) DESC rows 1 to 10 ;
```

ДОПУСТИЛ ОШИБКУ ПРИ СОРТИРОВКЕ: БЫЛО – сортировка по алфавиту, стало – сортировка в порядке убывания по количеству выданных отсрочек

Последовательность действий: для примера использован вложенный select. Сначала во вложенном SELECT находятся Id_призывников, у которых указана отсрочка от армии предоставлена от университета. Далее во внешнем SELECT мы ведем вывод названий университетов и количество упоминаний по id_academic.
Вывод:

NAME	COUNT
SPBAU	2
SPBBGTU	1
SPBGU	1
SPBPU	1

ПРИ ДОБАВЛЕНИИ 100000 записей в каждую из таблиц:
ВРЕМЯ ЗАТРАЧЕННОЕ НА ОБРАБОТКУ ЗАПРОСА -------- 2,48 с

2) Вывести 5 призывников, которые провели наибольшее количество дней в мероприятиях не связанных с призывом.

```
select p3.name, sum(p1.activity_date_off - p1.activity_date)  from activities
as p1, personal_file as p2, recruit as p3

where p2.id_file = p1.id_file and p1.activity_date_off is not NULL and
p2.id_recruit = p3.id_recruit and p1.id_activity in

(

    select p1.id_activity from kind_activity as p1

    where  p1.kind_activity not like 'Military%'

)

group by p3.name order by sum(p1.activity_date_off - p1.activity_date) DESC,
p3.name asc rows 1 to 5;
```

ПРИ ДОБАВЛЕНИИ 100000 записей в каждую из таблиц:
ВРЕМЯ ЗАТРАЧЕННОЕ НА ОБРАБОТКУ ЗАПРОСА -------- 6,54 с

Последовательность действий: на вывод выводятся имя призывника и количество дней, в течении которых призывник провел на любых видах мероприятий, не связанных с призывом. На рисунке ниже мы видим, какие бывают мероприятия (таблица KIND_ACTIVITY)

ID_ACTIV...	KIND_ACTIVITY
1	Medical certificate
2	Medical examination
3	Get delay
4	Gathering 2015-01-01
5	Military draft
6	Military out

Допустим, что в данной базе данных все активности связанные с призывом начинаются со слова Military*. Таким образом, появляется возможность искать и считать у каждого призывника те активности, которые не связаны с призывом. Далее ведется поиск по таблице активностей и находятся id_личных дел, которые впоследствии проверяются на равенство в таблице recruit, где и лежат фамилии призывников.

Также стоит заметить, что производится сортировка по убыванию и сортировка по количеству записей, основанная на приведенном выше задании.

Вывод при добавлении 100000 рандомных записей в таблицу:

NAME	DAYS
65,Gv=sv8>z?x8Ub<|G"onjuL:HM+9+|Cp;0a` f(<Y4s0C~ry=q?GcPZD/Yl~AkB	9 102
p)6'Q7!aOL[(Zon(&UDDLgwedJJxtp{/f>(t0TbBV9~1#@/'~]o9m767=d!f:_y)6`	8 561
1X^w.W,9y-8;t=1eaWxoUG^g@CO1JR;`Q`maFRV3Y$=bF=MN<kW}K"/8@g[	7 716
NS !VuNLKgAhI/`-^ahP`5?Fnx{5VS`g_ >]7tInNp?+?s~3cX^b<cb_{jFRH}	7 700
y	7 225

3) Удалить неиспользуемые причины отсрочек.

До того как мы удалим неиспользуемые записи, посмотрим на состояние таблиц Reasons и Delay с помощью запроса

```
SELECT *
FROM  Delay  FULL JOIN Reasons
ON Delay.id_reason = Reasons.id_reason
```

ID_FILE	ID_REASON	DELAY_DATE_TO	ID_REASON1	REASON_TEXT
100	1	12.10.2014	1	Delay University
102	1	12.10.2014	1	Delay University
105	1	12.10.2014	1	Delay University
107	1	12.10.2014	1	Delay University
110	1	12.10.2014	1	Delay University
101	2	12.10.2014	2	Delay Health
106	2	12.10.2014	2	Delay Health
111	2	12.10.2014	2	Delay Health
114	3	12.10.2014	3	Delay Family Condition
103	4	12.10.2014	4	Delay Work
108	4	12.10.2014	4	Delay Work
112	4	12.10.2014	4	Delay Work
104	5	12.10.2014	5	Delay Children
109	5	12.10.2014	5	Delay Children
113	5	12.10.2014	5	Delay Children
<null>	<null>	<null>	6	For delete 1
<null>	<null>	<null>	7	For delete 2

Видим, что есть две причины отсрочки, которые не используются в таблице Reasons. Поэтому стоит их удалить следующим запросом:

```
delete from Reasons where Reasons.id_reason not in (select Delays.id_reason from Delays)
```

Состояние таблицы Reasons после удаления не используемых:

ID_REAS...	REASON_TEXT
1	Delay University
2	Delay Health
3	Delay Family Condition
4	Delay Work
5	Delay Children

Выполненные запросы SELECT сохранены в БД в виде представлений, запросы INSERT, UPDATE или DELETE -- в виде ХП, которые будут показаны лично преподавателю.

Выводы: в данной лабораторной работе мы познакомились с операторами манипулирования данными. С помощью одного из основных операторов SECECT можно извлечь любые данные из таблиц и получать ответы на различные вопросы. Разнообразие логических операторов, группировок, сортировок данных в запросе дает пользователю свободу в написании запросов.

Также мы познакомились с такими объектами базы данных как представления и хранимые процедуры. Запросы представлений, фактически, запрос запроса. Использование представлений удобно для получения значений из постоянно меняющегося содержания базы данных. Хранимые процедуры используется, например, если в базе данных нужно обработать какой-то сложный запрос. Незачем в это вовлекать какое-то внешнее приложение, ведь всю функциональность можно спрятать в самой базе данных. Помимо осмысленности хранимые процедуры выполняются намного быстрее относительно запросов над базой данных из приложения