Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №4**

«SQL-DML»

Работу выполнил студент группы № 43501/3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Анисимов А.А.

Работу принял преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мяснов А.В.

Санкт-Петербург

2016

**Цели работы**

Знакомство 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.
5. Список стандартных запросов
6. Сделайте выборку всех данных из каждой таблицы
7. Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее 3-х разных примеров)
8. Создайте в запросе вычисляемое поле
9. Сделайте выборку всех данных с сортировкой по нескольким полям
10. Создайте запрос, вычисляющий несколько совокупных характеристик таблиц
11. Сделайте выборку данных из связанных таблиц (не менее двух примеров)
12. Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки
13. Придумайте и реализуйте пример использования вложенного запроса
14. С помощью оператора INSERT добавьте в каждую таблицу по одной записи
15. С помощью оператора UPDATE измените значения нескольких полей у всех записей, отвечающих заданному условию
16. С помощью оператора DELETE удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики
17. С помощью оператора DELETE удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

**3. Выполнение работы**

Выполнение стандартных запросов:

Выборка всех данных из каждой таблицы:

**Листинг 1. Представления для выборки всех данных из каждой таблицы**

|  |
| --- |
| CREATE VIEW ClientComplite AS SELECT \* FROM client;  CREATE VIEW GroupsComplite AS SELECT \* FROM groups;  CREATE VIEW AccessTypeComplite AS SELECT \* FROM access\_type;  CREATE VIEW DepBoxComplite AS SELECT \* FROM deposit\_box;  CREATE VIEW GrMemberComplite AS SELECT \* FROM group\_member;  CREATE VIEW ContractComplite AS SELECT \* FROM contract;  CREATE VIEW BoxHistComplite AS SELECT \* FROM box\_history; |

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

**Листинг 2. Выборка с использованием логических операций.**

|  |
| --- |
| CREATE VIEW NamesWithAn AS SELECT \* FROM client WHERE client\_lastname LIKE 'An%';  CREATE VIEW GroupsEndsOn23 AS SELECT \* FROM groups WHERE group\_id LIKE '%23';  CREATE VIEW BoxValueEndsOn223 AS SELECT \* FROM deposit\_box WHERE depb\_value LIKE '%223';  CREATE VIEW SpecificPassportRange AS SELECT \* FROM client WHERE client\_passport BETWEEN 100000 AND 100200;  CREATE VIEW SpecificGroupMemberRange AS SELECT \* FROM group\_member WHERE client\_id BETWEEN 99000 AND 100000;  CREATE VIEW SpecificBoxHistoryRange AS SELECT \* FROM box\_history WHERE history\_previousValue BETWEEN 9700 AND 10000;  CREATE VIEW AllAlAndVic AS SELECT \* FROM client WHERE client\_name IN ('Alexander', 'Victor');  CREATE VIEW SpesificPreviousValue AS SELECT \* FROM box\_history WHERE history\_previousValue IN (10981.538, 8269.8896);  CREATE VIEW RangeOfDebBox AS SELECT \* FROM deposit\_box WHERE depb\_id IN (7, 75555); |

Создайте в запросе вычисляемое поле

**Листинг 3. Создание вычисляемого поля.**

|  |
| --- |
| CREATE VIEW CalcField AS SELECT client\_id, client\_name AS whoisit\_calc\_f FROM client WHERE client\_id BETWEEN 2 AND 30; |

Сделайте выборку всех данных с сортировкой по нескольким полям

**Листинг 4. Выборка с сортировкой.**

|  |
| --- |
| CREATE VIEW MultSelect AS SELECT \* FROM client ORDER BY client\_lastname ASC, client\_dob DESC; |

Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

**Листинг 5.**

|  |
| --- |
| CREATE VIEW CalcMultCharct AS SELECT MAX(contract\_cost) AS HighestPrice, AVG(accesstype\_id) AS AverageTypeOfAccess FROM contract; |

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

**Листинг 6.**

|  |
| --- |
| CREATE VIEW SelRefClCn AS SELECT client\_name, contract\_cost  FROM client, contract, groups, group\_member  WHERE groups.group\_id=group\_member.group\_id  AND group\_member.client\_id=client.client\_id;  CREATE VIEW SelRefDepBHis AS SELECT deposit\_box.depb\_id, history\_previousValue  FROM box\_history, deposit\_box  WHERE deposit\_box.depb\_id=box\_history.depb\_id  AND box\_history.history\_previousValue < 1000; |

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

**Листинг 7.**

|  |
| --- |
| CREATE VIEW CombWithGrBy AS SELECT client\_name AS Name, COUNT(client\_name) FROM client group by client\_name; |

Придумайте и реализуйте пример использования вложенного запроса

**Листинг 8.**

|  |
| --- |
| CREATE VIEW SubquClName AS SELECT client\_name  FROM client  WHERE client\_id  IN (SELECT contract\_id FROM box\_history WHERE client.client\_id=box\_history.client\_id); |

С помощью оператора INSERT добавьте в каждую таблицу по одной записи

**Листинг 9. Процедуры для добавления записей.**

|  |
| --- |
| SET term # ;  CREATE PROCEDURE insert\_client (client\_id int, client\_name varchar(128), client\_lastname varchar(128),client\_dob timestamp,client\_passport int)  AS  BEGIN  INSERT INTO client (client\_id, client\_name, client\_lastname, client\_dob, client\_passport)  VALUES (:client\_id, :client\_name, :client\_lastname, :client\_dob, :client\_passport);  BEGIN  SUSPEND;  END  END#  SET term ; #  SET term # ;  CREATE PROCEDURE insert\_group (group\_id int)  AS  BEGIN  INSERT INTO groups (group\_id) VALUES (:group\_id);  BEGIN  SUSPEND;  END  END#  SET term ; #  SET term # ;  CREATE PROCEDURE insert\_deposit\_box (depb\_id int, depb\_num int, depb\_value float,depb\_type\_of\_value varchar(128))  AS  BEGIN  INSERT INTO deposit\_box (depb\_id, depb\_num, depb\_value, depb\_type\_of\_value)  VALUES (:depb\_id, :depb\_num, :depb\_value, :depb\_type\_of\_value); BEGIN  SUSPEND;  END  END#  SET term ; #  SET term # ;  CREATE PROCEDURE insert\_group\_member (gm\_id int, group\_id int, client\_id int)  AS  BEGIN  INSERT INTO group\_member (gm\_id, group\_id, client\_id) VALUES (:gm\_id, :group\_id, :client\_id);  BEGIN  SUSPEND;  END  END#  SET term ; #  SET term # ;  CREATE PROCEDURE insert\_contract (contract\_id int, contract\_start timestamp,contract\_end timestamp,contract\_cost float,depb\_id int,accesstype\_id int,group\_id int)  AS  BEGIN  INSERT INTO contract (contract\_id, contract\_start, contract\_end, contract\_cost, depb\_id, accesstype\_id, group\_id) VALUES (:contract\_id, :contract\_start, :contract\_end, :contract\_cost, :depb\_id, :accesstype\_id, :group\_id);  BEGIN  SUSPEND;  END  END#  SET term ; #  SET term # ;  CREATE PROCEDURE insert\_box\_history (history\_id int,client\_id int,history\_previousValue float,history\_currentValue float,depb\_id int,contract\_id int)  AS  BEGIN  INSERT INTO box\_history (history\_id, client\_id, history\_previousValue, history\_currentValue, depb\_id, contract\_id)VALUES (:history\_id, :client\_id, :history\_previousValue, :history\_currentValue, :depb\_id, :contract\_id);  BEGIN  SUSPEND;  END  END#  SET term ; # |

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

**Листинг 10. Обновление записей**

|  |
| --- |
| SET term # ;  CREATE PROCEDURE update\_name (  client\_id int,  client\_name varchar(128),  client\_new\_name varchar(128))  AS  BEGIN  UPDATE client SET client\_name=:client\_new\_name WHERE client\_name=:client\_name AND client\_id=:client\_id;    BEGIN  SUSPEND;  END  END#  SET term ; # |

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

**Листинг 11. Удаление записей.**

|  |
| --- |
| SET term # ;  CREATE PROCEDURE delete\_max\_cost\_contract  AS  BEGIN  DELETE FROM contract WHERE contract\_cost=(SELECT MAX(contract\_cost) FROM contract);    BEGIN  SUSPEND;  END  END#  SET term ; # |

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

**Листинг 12. Удаление записей.**

|  |
| --- |
| SET term # ;  CREATE PROCEDURE delete\_max\_cost\_contract  AS  BEGIN  DELETE FROM box\_history where history\_id NOT IN (SELECT client\_id FROM client);    BEGIN  SUSPEND;  END  END#  SET term ; # |

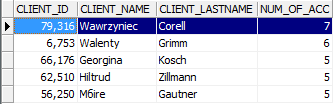
**Индивидуальное задание:**

1. Вывести 5 людей, которые наиболее часто получали доступ к каким-либо ячейкам за заданный промежуток времени.

|  |
| --- |
| SELECT FIRST 5  box\_history.client\_id,  client.client\_name,  client.client\_lastname,  COUNT(box\_history.client\_id) AS Num\_of\_Acc  FROM box\_history, client  WHERE box\_history.time\_of\_access between :start\_from AND :end\_on  AND client.client\_id = box\_history.client\_id  GROUP BY box\_history.client\_id, client.client\_name, client.client\_lastname  ORDER BY Num\_of\_Acc DESC; |

Вместо переменных “:start\_from” и “:end\_on” необходимо подставить временные промежутки. Пример выполнения для промежутка от '03.04.99 11:36:19' до '22.04.2008 11:36:19'

Результат выполнения:



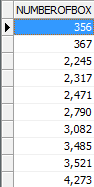
Информация о производительности:



1. Вывести 10 ячеек, стоимость содержимого которых увеличивалась за последний месяц.

|  |
| --- |
| SELECT FIRST 10 depb\_id AS NumberOfBox  FROM box\_history  WHERE EXTRACT (MONTH FROM time\_of\_access) = EXTRACT (MONTH FROM current\_timestamp)  AND EXTRACT (YEAR FROM time\_of\_access) = EXTRACT (YEAR FROM current\_timestamp)  GROUP BY depb\_id, history\_id, HISTORY\_CURRENTVALUE, HISTORY\_PREVIOUSVALUE  HAVING HISTORY\_PREVIOUSVALUE < HISTORY\_CURRENTVALUE; |

Результат выполнения:



Информация о производительности:



1. Удалить типы доступа, которые не используются.

|  |
| --- |
| DELETE FROM access\_type  WHERE NOT EXISTS (SELECT \*  FROM contract  WHERE contract.accesstype\_id = access\_type.accesstype\_id); |

Результат выполнения отсутствуют, т.к. все типы доступа использованы.

Информация о производительности:



**Выводы:**

В результате работы был изучен язык манипулирования данными SQL-DML. Были написаны запросы извлечения данных из БД в соответствии с индивидуальным заданием.

SQL-DDL(Data Definition Language) – язык определения структур и ограничений целостности баз данных. Использовались такие инструменты языка DDL как <create> (создать), <alter>(изменить), <drop> (удалить).

Основой языка SQL является язык реляционных БД. Язык SQL позволяет управлять структурой БД, это обеспечивает гибкость БД к изменяющимся требованиям предметной области.

Были изучены представления и хранимые процедуры SQL. Эти сущности позволяют хранить запросы и скрипты в самой БД. С помощью представлений удобно многократно выполнять сложные запросы к БД. С помощью хранимых процедур удобно реализовывать запросы добавления и удаления данных.