Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №3**

**Дисциплина**: Базы данных

# SQL-DML.

Выполнил студент гр. 43501/3 Муравьев Ф.Э.

Преподаватель: Мяснов А. В.

Санкт-Петербург

2015

**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.

## 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)

**3. Выполнение работы**

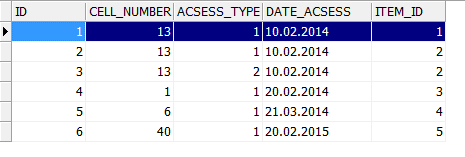
Выполнение стандартных запросов:

1. Выборка всех данных из каждой таблицы:

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

|  |
| --- |
| create view Cells\_Select as select \* from cells;  create view Clients\_Select as select \* from clients;  create view Contracts\_Select as select \* from contracts;  create view Groups\_Select as select \* from groups;  create view Groups\_Clients\_Select as select \* from groups\_clients;  create view History\_Cell\_Select as select \* from history\_cell;  create view Items\_Select as select \* from items;  create view Repository\_Select as select \* from repository; |

Полученная выборка данных из таблицы history\_cell:

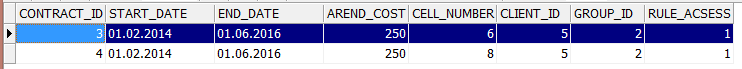


1. Выборка данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN:

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

|  |
| --- |
| create view selectContracts250 as select \* from contracts where arend\_cost like 250;  create view selectItemsPrice as select \* from items where cost between 5000 and 50000;  create view selectAcess0\_1 as select \* from history\_cell where acsess\_type in (0, 1); |

Полученные результаты для оператора like(выборка контрактов с арендной платой 250):

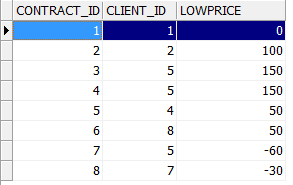


1. Создайте в запросе вычисляемое поле

**Листинг 3. Создание вычисляемого поля.**

|  |
| --- |
| create view CalcField as select Contract\_ID, Client\_ID, (arend\_cost-100) AS LowPrice FROM contracts; |

Полученные результаты:

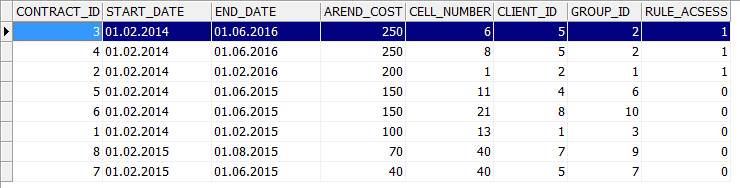


1. Сделайте выборку всех данных с сортировкой по нескольким полям

**Листинг 4. Выборка с сортировкой.**

|  |
| --- |
| create view SortContracts as select \* from contracts order by arend\_cost desc, start\_date asc; |

Полученные результаты сортировки(контракты отсортированные по арендной плате и дате окончания контракта):



1. Создайте запрос, вычисляющий несколько совокупных характеристик таблиц

**Листинг 5.**

|  |
| --- |
| create view Max\_And\_Sred as select max(end\_date) as "max\_date", avg(arend\_cost) as "avg" from contracts; |

Полученные результаты(максимальная дата окончания контракта, средняя цена за аренду):

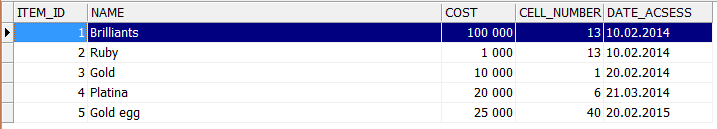


1. Сделайте выборку данных из связанных таблиц (не менее двух примеров)

**Листинг 6.**

|  |
| --- |
| create view selectAllHistItems as select history\_cell.item\_ID, name, cost, cell\_number, date\_acsess from history\_cell, items where history\_cell.item\_id=items.item\_id and history\_cell.acsess\_type=1;  /\*Выборка всей истории о сдаче предметов на хранение\*/  create view ContractsView as select contract\_ID, name, cell\_number, arend\_cost from contracts, clients where contracts.client\_id=clients.client\_ID and contracts.rule\_acsess=0;  /\*Выборка всех контрактов оформленных на одного человека\*/ |

Полученные результаты для выборки всей истории:

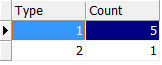


1. Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

**Листинг 7.**

|  |
| --- |
| create view His\_Ac\_Items as select acsess\_type as "Type", COUNT(acsess\_type) as "Count" from history\_cell group by acsess\_type;  /\*определение количества различных доступов к предметам в ячейке(0- осмотр, 1- добавление, 2- изъятие)\*/ |

Результаты запроса:

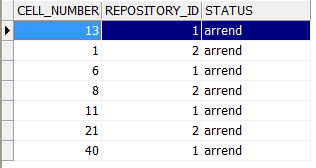


1. Придумайте и реализуйте пример использования вложенного запроса

**Листинг 8.**

|  |
| --- |
| create view View\_Cell\_arrend as select \* from cells where cell\_number in (select cell\_number from contracts);  /\*определение ячеек, на которые оформлены заказы\*/ |

Полученные результаты:



1. С помощью оператора INSERT добавьте в каждую таблицу по одной записи

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

|  |
| --- |
| create procedure insert\_items (item\_ID integer, contracts\_ID integer, name varchar(50), cost integer)  as  begin  insert into items(item\_ID, contract\_ID, name, cost) values (:item\_ID, :contracts\_ID, :name, :cost);  end;  create procedure insert\_history\_cell (ID integer, cell\_number integer, acsess\_type integer, date\_acsess date, item\_id integer)  as  begin  insert into history\_cell(ID, cell\_number, acsess\_type, date\_acsess, item\_ID) values (:ID, :cell\_number, :acsess\_type, :date\_acsess, :item\_ID);  end;  create procedure insert\_cells(cell\_number integer, repository\_ID integer, status varchar(15))  as  begin  insert into cells(cell\_number, repository\_ID, status) values (:cell\_number, :repository\_ID, :status);  end;  create procedure insert\_repository(repository\_ID integer, adress varchar(100), director varchar(32))  as  begin  insert into repository(repository\_ID, adress, director) values (:repository\_ID, :adress, :director);  end;  create procedure insert\_contract(contract\_ID integer, start\_date date, end\_date date, arend\_cost integer, cell\_number integer, client\_ID integer, group\_ID integer, rule\_acsess integer)  as  begin  insert into contracts(contract\_ID, start\_date, end\_date, arend\_cost, cell\_number, client\_ID, group\_ID, rule\_acsess) values(:contract\_ID, :start\_date, :end\_date, :arend\_cost, :cell\_number, :client\_ID, :group\_ID, :rule\_acsess);  end;  create procedure insert\_client(client\_ID integer, name varchar(50), adress varchar(100), born\_date date)  as  begin  insert into clients(client\_ID, name, adress, born\_date) values(:client\_ID, :name, :adress, :born\_date);  end;  create procedure insert\_group(group\_ID integer, date\_create date)  as  begin  insert into groups(group\_ID, date\_create) values(:group\_ID, :date\_create);  end;  create procedure insert\_group\_client(ID integer, client\_ID integer, group\_ID integer)  as  begin  insert into groups\_clients(ID, client\_ID, group\_ID) values(:ID, :client\_ID, :group\_ID);  end; |

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

**Листинг 10. Обновление записей**

|  |
| --- |
| create procedure update\_free\_cells(date\_end date) as  begin  update cells set status='free' where cells.cell\_number in (select cell\_number from contracts where contracts.end\_date<:date\_end);  end;  /\*освобождение ячеек, на которые истекли контракты\*/ |

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

**Листинг 11. Удаление записей.**

|  |
| --- |
| create procedure del\_history\_about\_max\_cost(acsess\_t integer) as  begin  delete from history\_cell where acsess\_type = :acsess\_t and item\_id = (select item\_id from items where cost = (select max(cost) from items));  end;  /\*удаление записей о предмете с максимальной стоимостью из таблицы history\_cell с заданным типом доступа\*/ |

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

**Листинг 12. Удаление записей.**

|  |
| --- |
| create procedure del\_unused\_items as  begin  delete from items where item\_id not in(select contract\_id from contracts);  end;  /\*удаление записи о предмете, по которому не оформлен по контракт\*/ |

**Индивидуальное задание:**

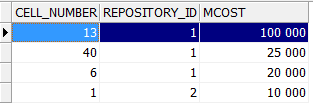
*Реализовать следующие запросы:*

1. Отобразить 10 ячеек, в которых на текущий момент находятся ценности с максимальной суммарной стоимостью.
2. Отобразить количество заполненных ячеек в каждом хранилище.
3. Удалить данные о ценностях по истекшим контрактам.
4. **Отобразить 10 ячеек, в которых на текущий момент находятся ценности с максимальной суммарной стоимостью.**

**Листинг 13. 10 ячеек с max суммарной стоимостью.**

|  |
| --- |
| create view top10cells as select first 10 cells.cell\_number, repository\_id, sum(items.cost) as MCost from cells, items, history\_cell  where (history\_cell.cell\_number=cells.cell\_number)and(history\_cell.item\_id=items.item\_id)and(history\_cell.acsess\_type=1)and(history\_cell.item\_id not in (select item\_id from history\_cell where item\_id=2)) group by cells.cell\_number, cells.repository\_id order by MCost desc; |

Результат запроса:



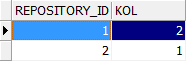
При выполнении на большом объеме данных(100000 записей в таблице) время выполнения скрипта – 2s 510ms.

1. **Отобразить количество заполненных ячеек в каждом хранилище.**

**Листинг 14. Количество заполненных ячеек в каждом хранилище**

|  |
| --- |
| create view ind\_arend\_cells as select repository.repository\_id, count(contracts.contract\_id) as kol from cells, repository, contracts  where (repository.repository\_id=cells.repository\_id)and(cells.cell\_number=contracts.cell\_number)and(contracts.end\_date>current\_date) group by repository.repository\_id; |

Результат запроса:

****

При выполнении на большом объеме данных(100000 записей в таблице) время выполнения скрипта – 320ms.

1. **Удалить данные о ценностях по истекшим контрактам.**

**Листинг 15. Удаление данных об истекших ценностях**

|  |
| --- |
| create procedure delete\_ditems as  begin  delete from history\_cell where history\_cell.item\_id in (select items.item\_id from items, contracts where(contracts.contract\_id=items.contract\_id)and(contracts.end\_date<current\_date));  delete from items where item\_id in (select items.item\_id from items, contracts where(contracts.contract\_id=items.contract\_id)and(contracts.end\_date<current\_date));  end; |

При выполнении на большом объеме данных(100000 записей в таблице) время выполнения скрипта – 781ms.

**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.