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»

«SQL-DML»

Работу выполнили студенты группы №43501/4 Н.С. Шаляпин\_\_\_\_\_\_\_\_\_

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

Санкт-Петербург

2015

1. Цель работы

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

1. Язык SQL

Язык SQL (Structured Query Language) - язык структурированных запросов. Он поз­воляет формировать весьма сложные запросы к базам данных, 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.
* В SQL-DML определены команды «select» («выбрать»), «insert» («вставить»), «update» («обновить»), и «delete» («удалить»).

1. 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.

1. Ход работы

Был самостоятельно изучен язык SQL-DML;

Были выполнены все запросы из списка всех стандартных запросов:

* Представления для выборки данных из каждой таблицы.

|  |
| --- |
| CREATE view MotorSelect as select \* from motors;  CREATE view MarksSelect as select \* from marks;  CREATE view Complete\_setSelect as select \* from complete\_set;  CREATE view Model\_carSelect as select \* from model\_car;  CREATE view CustomersSelect as select \* from customers;  CREATE view Additional\_optionsSelect as select \* from additional\_options;  CREATE view TradeSelect as select \* from trade;  CREATE view ServiceSelect as select \* from service;  CREATE view TransmissionSelect as select \* from transmission; |

Пример выборки данных из таблицы доп. опции:

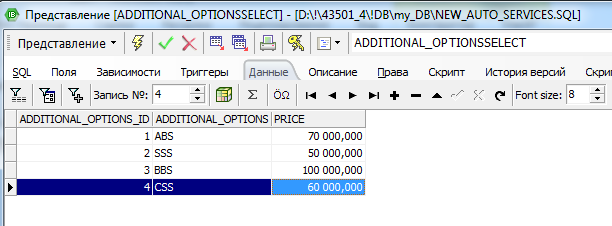


Рис. 1: Выборка данных из таблицы доп. опции.

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

* Выборка с использованием логических операций

|  |
| --- |
| create view selectIvanov as select \* from customers where customers like 'Ivanov % ' ;  create view selectOptPrice as select \* from additional\_options where price between 10000 and 80000;  create view selectMotorType as select \* from motors where "TYPE\_OF\_MOTOR" in ('V6', 'V8 '); |

Пример использования оператора BETWEEN:

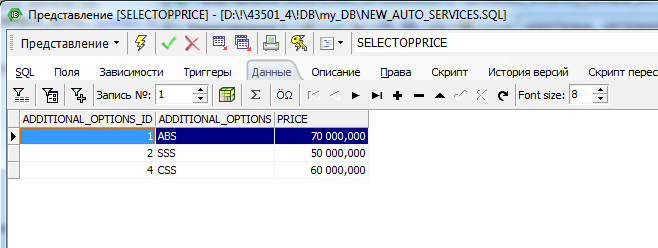


Рис. 2: Выборка всех опций с ценой от 10000 до 80000.

* Выборка с сортировкой

|  |
| --- |
| /\* все двигатели отсортированы по названию, типу, объему и мощности. \*/  create view selectMotrSort as select \* from motors order by motor asc , type\_of\_motor asc ,  volume\_motor asc, power asc; |

Пример использования сортировки:

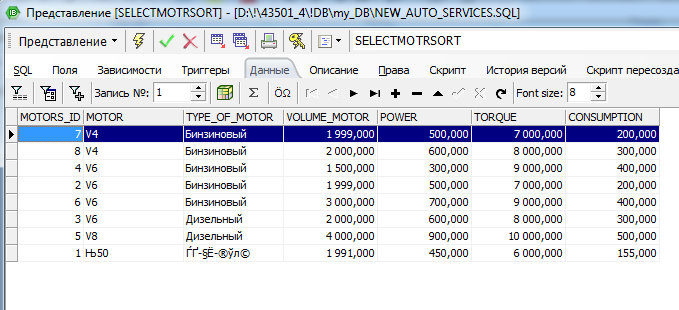
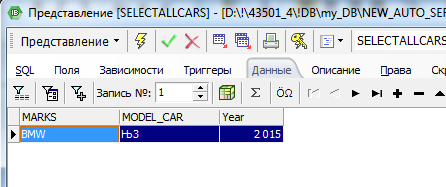


Рис. 3: Двигатели отсортированные по названию, типу, объему и мощности.

* Выборка данных из связных таблиц.

|  |
| --- |
| /\* выборка всех моделей с именами марок\*/  create view selectAllCars as select Marks.mark as marks, model\_car.model\_car as model\_car,  model\_car.production\_year as "Year" from marks, model\_car where Marks.marks\_id = model\_car.marks\_id;  /\*выборка всех клиентов с совершивших сделку, купив указанную модель\*/  create view selectAllClients as select customers.customers\_id as id, Marks.mark as mark,  model\_car as Model, trade.discount as Discount from customers  , Marks, model\_car, trade where trade. customers\_id = customers.customers\_id and marks.marks\_id = model\_car. marks\_id; |

Пример извлечения всех моделей:



* Запрос, рассчитывающий совокупную характеристику с использованием груп­пировки, с наложением ограничения на результат группировки.

Совокупная характеристика с использованием группировки:

|  |
| --- |
| create view selectMotorsTypes as select motors.type\_of\_motor as "Type",  COUNT (motors.type\_of\_motor) as "Count" from motors group by motors.type\_of\_motor; |

Результат запроса:

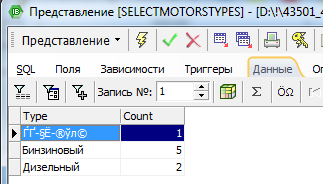


Рис. 5: Определение количества двигателей каждого типа

* Пример использования вложенного запроса

Вложенный запрос

|  |
| --- |
| create view selectUnusedModelCar as select \* from model\_car where model\_car\_id not in  (select model\_car\_id from trade); |

Результат запроса:

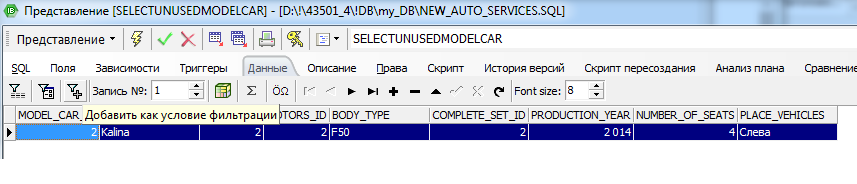


Рис. 6: Определение моделей, которые пи разу не были куплены.

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

Процедуры для добавления записей.

|  |
| --- |
| create procedure ins\_mark (i INTEGER, e varchar(255))  as  begin  insert into marks (marks\_id, mark) values (:i, :e);  end;  create procedure ins\_mark (i INTEGER, e varchar(255))  as  begin  insert into marks (marks\_id, mark) values (:i, :e);  end;  create procedure ins\_motors (i INTEGER, e varchar(16), t VARCHAR (16), v FLOAT, p FLOAT, y FLOAT, c FLOAT)  as  begin  insert into motors (motors\_id , motor , type\_of\_motor , volume\_motor, power , torque, consumption )  values (:i, :e, :t, :v, :p, :y, :c);  end;  create procedure ins\_complete\_set (i INTEGER, e varchar(16), t INTEGER, v INTEGER, p INTEGER)  as  begin  insert into complete\_set (complete\_set\_id , equipment , motors\_id, transmission\_id, additional\_options\_id)  values (:i, :e, :t, :v, :p);  end;  create procedure ins\_model\_car (m INTEGER , r VARCHAR (16) ,d INTEGER ,i INTEGER , b VARCHAR (16), f INTEGER ,p INTEGER, n INTEGER, y VARCHAR (16))  as  begin  insert into model\_car (model\_car\_id , model\_car , marks\_id ,  motors\_id , body\_type , complete\_set\_id,  production\_year , number\_of\_seats , place\_vehicles)  values (:m, :r, :d, :i, :b , :f, :p, :n , :y );  end;  create procedure ins\_customers (i INTEGER, e varchar(100))  as  begin  insert into customers (customers\_id , customers )  values (:i, :e);  end;  create procedure ins\_additional\_options (i INTEGER, e varchar(100),p FLOAT)  as  begin  insert into additional\_options(additional\_options\_id , additional\_options , price )  values (:i, :e, :p);  end;  create procedure ins\_trade (i INTEGER, e INTEGER,p INTEGER, l INTEGER, k float, u float)  as  begin  insert into trade(trade\_id ,customers\_id,model\_car\_id ,additional\_options\_id ,  discount , total\_price )  values (:i, :e, :p, :l, :k, :u );  end;  create procedure ins\_service (i INTEGER, e INTEGER,p varchar(100), l INTEGER)  as  begin  insert into service(service\_id, model\_car\_id,  type\_of\_fault , trade\_id)  values (:i, :e, :p, :l);  end;  create procedure ins\_transmission (i INTEGER, e varchar(255))  as  begin  insert into transmission (transmission\_id , transmission )  values (:i, :e);  end; |

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

Обновление записей

|  |
| --- |
| /\* увеличение цены, доп.опции типа ABS на ргос% \*/  create procedure upd\_additional\_options(proc float) as  begin  update additional\_options set price = :proc \* price where additional\_options.additional\_options = 'ABS';  end; |

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

Удаление записей

|  |
| --- |
| /\* удаление опции с максимальной ценой из сделки c номером i \*/  create procedure del\_max\_opt (i int) as  begin  delete from trade where additional\_options\_id = :i and total\_price = (select MAX(total\_price) from trade where additional\_options\_id = :i);  end; |

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

Удаление записей

|  |
| --- |
| /\* удаление марок, на которые не ссылаются модели \*/  create procedure del\_mark as  begin  delete from marks where marks\_id not in(select marks\_id from model\_car);  end; |

У преподавателя были получены индивидуальные задания:

* Вывести 10 наиболее продаваемых моделей автомобилей с названиями марок, моделей и количеством проданных машин.

Вывод 10 самых популярных моделей

|  |
| --- |
| create view top10models as select first 10 marks.mark as Marki, model\_car.model\_car as Model,  COUNT(trade.trade\_id) as Buyed from model\_car, marks, trade  where model\_car.marks\_id = marks.marks\_id and model\_car.model\_car\_id = trade.model\_car\_id group by  model\_car.model\_car, marks.mark order by Buyed desc; |

Результат запроса:

При выполнении на большом объеме данных (100000 записей в таблице) время выполнения скрипта - 1328 мс.

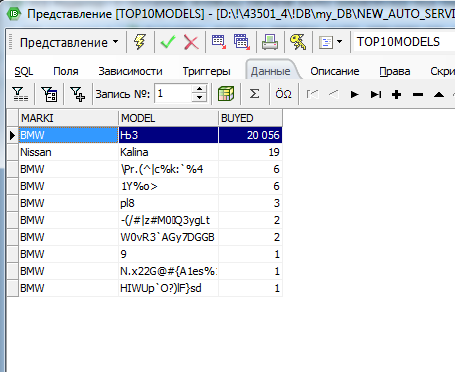


Рис. 7: 10 самых популярных моделей.

* Вывести 5 марок, по которым было наибольшее количество обращений в сервис за выбранный период.

Вывод 5 марок

|  |
| --- |
| create view worst5marks as select first 5 marks.mark as Mark,  COUNT(marks.marks\_id) as Faults from marks, model\_car, service  where model\_car.marks\_id = marks.marks\_id and model\_car.model\_car\_id = service.model\_car\_id and  service.appdate between '2000-01-01' and '2014-01-01' group by marks.mark order by Faults desc; |

Результат запроса:

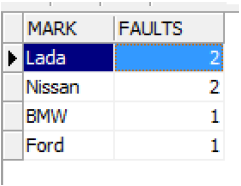


Рис. 8: 5 марок, по которым было наибольшее количество обращений в сервис за выбран­ный период.

При выполнении на большом объеме данных (100000 записей в таблице) время выполнения скрипта - 375 мс.

* Удалить неиспользуемые комплектации.

Удаление комплектаций, которые нигде не используются

|  |
| --- |
| create procedure del\_unused\_compl as  begin  delete from complete\_set where complete\_set.complete\_set\_id not in (select  complete\_set\_id from model\_car);  end; |

При выполнении на большом объеме данных (100000 записей в таблице) время выполнения скрипта - 594 мс.

Вывод

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

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.

Использовались следующие команды языка DML: insert (добавить), update (обновить), delete (удалить), select (выборка данных). Данный язык удобен для написания запросов разной сложности. При обращении к нескольким таблицам в запросе следует выбирать подходящую связь между таблицами для ускорения выполнения SQL-запроса, В ходе работы проблем не возникло.