

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА



Институт радиоэлектроники и информационных технологий
Кафедра вычислительные системы и технологии

Лабораторная работа № 2
Изучение операторов языка T-SQL
и их применение на тестовой БД

ОТЧЕТ

по лабораторной работе

по дисциплине

Базы данных

РУКОВОДИТЕЛЬ:

Мисевич П.В.

СТУДЕНТ:

Сапожников В.О.
19-В-1

Работа защищена «__» _____

С оценкой _____

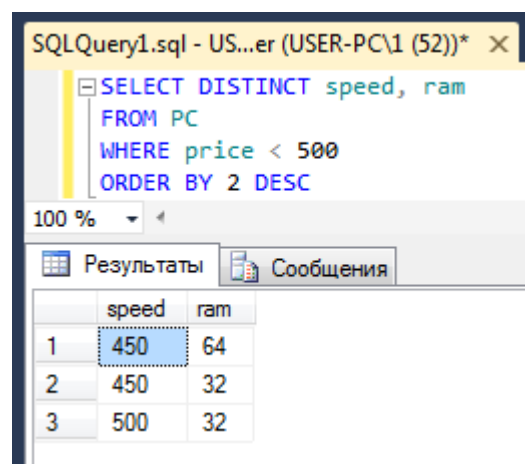
Нижний Новгород 2021

Цель работы: изучение операторов языка T-SQL и их применение на тестовой БД.

1. Оператор SELECT

Оператор **SELECT** осуществляет выборку данных и имеет наиболее сложную структуру среди всех операторов языка SQL. Помимо обычной выборки при помощи оператора **SELECT** и дополнительных функций можно: исключить повторения – **DISTINCT**, выполнить сортировку по любому кол-ву полей – **ORDER BY**, выполнить выборку согласно предикату – **WHERE**, сгруппировать данные – **GROUP BY**.

Пример: выполним запрос по столбцам speed и ram, исключив повторения, где цена больше 500 условных единиц и упорядочим по второму столбцу. Так же выполним запрос всех данных из таблицы для сравнения.



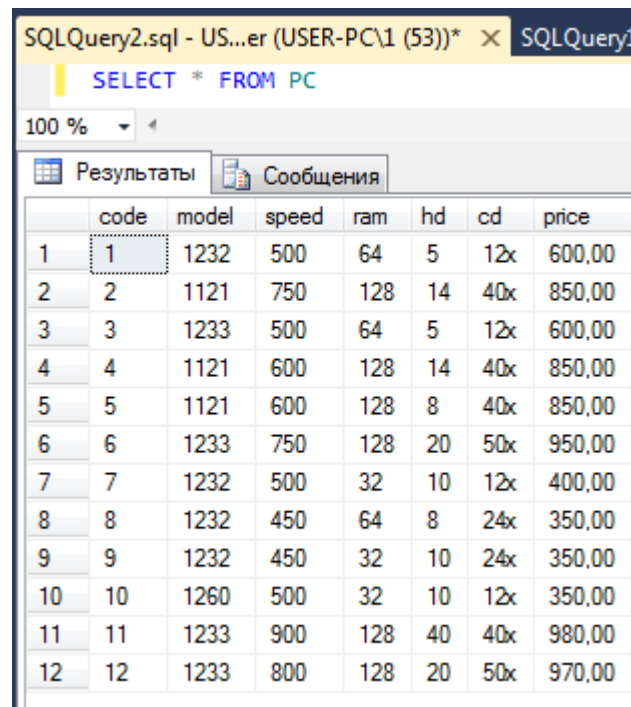
SQLQuery1.sql - US...er (USER-PC\1 (52))* X

```
SELECT DISTINCT speed, ram
FROM PC
WHERE price < 500
ORDER BY 2 DESC
```

100 %

Результаты Сообщения

	speed	ram
1	450	64
2	450	32
3	500	32



SQLQuery2.sql - US...er (USER-PC\1 (53))* X SQLQuery1

```
SELECT * FROM PC
```

100 %

Результаты Сообщения

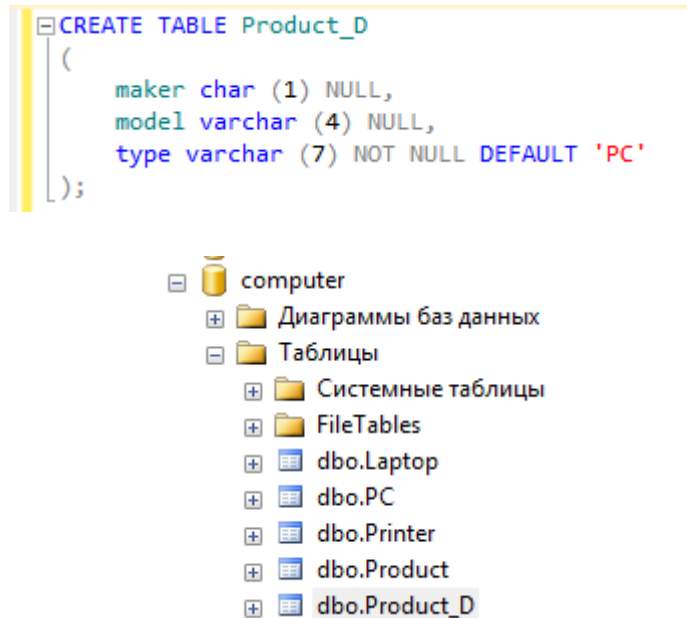
	code	model	speed	ram	hd	cd	price
1	1	1232	500	64	5	12x	600,00
2	2	1121	750	128	14	40x	850,00
3	3	1233	500	64	5	12x	600,00
4	4	1121	600	128	14	40x	850,00
5	5	1121	600	128	8	40x	850,00
6	6	1233	750	128	20	50x	950,00
7	7	1232	500	32	10	12x	400,00
8	8	1232	450	64	8	24x	350,00
9	9	1232	450	32	10	24x	350,00
10	10	1260	500	32	10	12x	350,00
11	11	1233	900	128	40	40x	980,00
12	12	1233	800	128	20	50x	970,00

2. Модификация данных.

2.1. Оператор CREATE TABLE

Для создания таблиц в БД используется оператор **CREATE TABLE**. При создании таблиц мы указываем столбцы и их типы, а также возможные ограничения и/или значения по умолчанию.

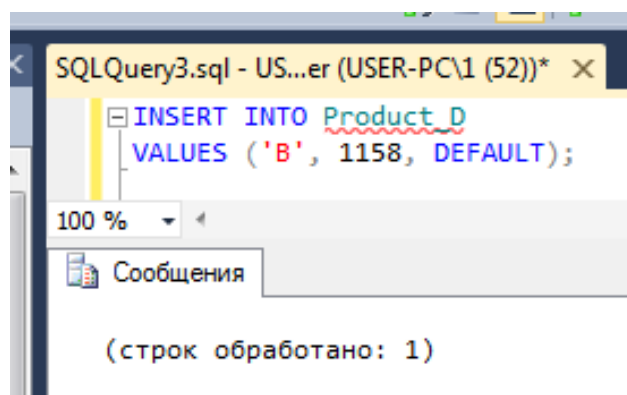
Пример: создадим таблицу **Product_D** с тремя столбцами, для последнего столбца зададим значение по умолчанию.



2.2. Оператор INSERT

При помощи оператора **INSERT** мы можем вставлять записи в таблицу. Важно, что значения передаваемые в качестве параметров записи, должен совпадать по типу с параметрами, указанными при создании таблицы (**CREATE TABLE**)

Пример: вставим запись в таблицу **Product_D**, последний параметр укажем по умолчанию.



SQLQuery4.sql - US...er (USER-PC\1 (53))*

```
SELECT * FROM dbo.Product_D
```

100 %

Результаты Сообщения

	maker	model	type
1	B	1158	PC

2.3.Оператор UPDATE

При помощи оператора **UPDATE** могут быть заданы значения для любого количества столбцов. Однако в одном и том же операторе **UPDATE** можно вносить изменения в каждый столбец таблицы только один раз. При отсутствии предложения **WHERE** будут обновлены все строки таблицы.

Пример: в таблице **Laptop** заменим все значения столбца **speed** на максимальное значение столбца **speed**. Стоит отметить, что внутри команды **UPDATE** мы использовали подзапрос с выборкой максимального значения **speed**.

SQLQuery4.sql - US...er (USER-PC\1 (53))* SQLQuery3.sql -

```
SELECT * FROM Laptop
```

100 %

Результаты Сообщения

	code	model	speed	ram	hd	price	screen
1	1	1298	350	32	4	700,00	11
2	2	1321	500	64	8	970,00	12
3	3	1750	750	128	12	1200,00	14
4	4	1298	600	64	10	1050,00	15
5	5	1752	750	128	10	1150,00	14
6	6	1298	450	64	10	950,00	12

SQLQuery4.sql - US...er (USER-PC\1 (53))* SQLQ

```
UPDATE Laptop
SET speed = (SELECT MAX(speed)
FROM Laptop
);
```

100 %

Сообщения

(строк обработано: 6)

SQLQuery4.sql - US...er (USER-PC\1 (53))* × SQLQuery3.sql

```
SELECT * FROM Laptop
```

100 %

Результаты Сообщения

	code	model	speed	ram	hd	price	screen
1	1	1298	750	32	4	700,00	11
2	2	1321	750	64	8	970,00	12
3	3	1750	750	128	12	1200,00	14
4	4	1298	750	64	10	1050,00	15
5	5	1752	750	128	10	1150,00	14
6	6	1298	750	64	10	950,00	12

2.4.Оператор DELETE

Оператор **DELETE** позволяет удалять записи, удовлетворяющие предикату.

Пример: из таблицы **Laptop** удаляем записи о ноутбуках с диагональю меньше 12 условных единиц.

SQLQuery4.sql - US...er (USER-PC\1 (53))* × SQLQuery3.sql

```
SELECT * FROM Laptop
```

100 %

Результаты Сообщения

	code	model	speed	ram	hd	price	screen
1	1	1298	750	32	4	700,00	11
2	2	1321	750	64	8	970,00	12
3	3	1750	750	128	12	1200,00	14
4	4	1298	750	64	10	1050,00	15
5	5	1752	750	128	10	1150,00	14
6	6	1298	750	64	10	950,00	12

SQLQuery4.sql - US...er (USER-PC\1 (53))* SQLQuery3.sql - US...er (USER-PC\1 (52))* ×

```
DELETE FROM Laptop
WHERE screen < 12;
```

100 %

Сообщения

(строк обработано: 1)

SQLQuery4.sql - US...er (USER-PC\1 (53))* X SQLQuery3.sql -

```
SELECT * FROM Laptop
```

100 %

Результаты Сообщения

	code	model	speed	ram	hd	price	screen
1	2	1321	750	64	8	970,00	12
2	3	1750	750	128	12	1200,00	14
3	4	1298	750	64	10	1050,00	15
4	5	1752	750	128	10	1150,00	14
5	6	1298	750	64	10	950,00	12

3. Работа с датами

Для работы с датами в реляционных базах данных используется тип Datetime. Использование отдельного типа для дат обуславливается наличием специальных операций для взаимодействия с данными такого типа.

Пример: найдем время в полете рейса 1123 из таблицы **Trip** базы данных **aero**.

SQLQuery6.sql - US...ro (USER-PC\1 (53))* SQLQuery5.sql - US...ro (USER-PC\1 (52))* X

```

SELECT CASE
  WHEN time_dep >= time_arr
  THEN time_arr - time_dep + 1440
  ELSE time_arr - time_dep
  END dur
FROM (SELECT DATEPART(hh, time_out)*60 + DATEPART(mi, time_out) time_dep,
  DATEPART(hh, time_in)*60 + DATEPART(mi, time_in) time_arr
  FROM Trip
  WHERE trip_no = 1123
) tm;

```

100 %

Результаты Сообщения

	dur
1	680

SQLQuery6.sql - US...ro (USER-PC\1 (53))* × SQLQuery5.sql - US...ro (USER-PC\1 (52))*

SELECT * FROM Trip

100 %

Результаты Сообщения

	trip_no	ID_comp	plane	town_from	town_to	time_out	time_in
1	1100	4	Boeing	Rostov	Paris	1900-01-01 14:30:00.000	1900-01-01 17:50:00.000
2	1101	4	Boeing	Paris	Rostov	1900-01-01 08:12:00.000	1900-01-01 11:45:00.000
3	1123	3	TU-154	Rostov	Vladivostok	1900-01-01 16:20:00.000	1900-01-01 03:40:00.000
4	1124	3	TU-154	Vladivostok	Rostov	1900-01-01 09:00:00.000	1900-01-01 19:50:00.000
5	1145	2	IL-86	Moscow	Rostov	1900-01-01 09:35:00.000	1900-01-01 11:23:00.000
6	1146	2	IL-86	Rostov	Moscow	1900-01-01 17:55:00.000	1900-01-01 20:01:00.000
7	1181	1	TU-134	Rostov	Moscow	1900-01-01 06:12:00.000	1900-01-01 08:01:00.000
8	1182	1	TU-134	Moscow	Rostov	1900-01-01 12:35:00.000	1900-01-01 14:30:00.000
9	1187	1	TU-134	Rostov	Moscow	1900-01-01 15:42:00.000	1900-01-01 17:39:00.000
10	1188	1	TU-134	Moscow	Rostov	1900-01-01 22:50:00.000	1900-01-01 00:48:00.000
11	1195	1	TU-154	Rostov	Moscow	1900-01-01 23:30:00.000	1900-01-01 01:11:00.000
12	1196	1	TU-154	Moscow	Rostov	1900-01-01 04:00:00.000	1900-01-01 05:45:00.000
13	7771	5	Boeing	London	Singapore	1900-01-01 01:00:00.000	1900-01-01 11:00:00.000
14	7772	5	Boeing	Singapore	London	1900-01-01 12:00:00.000	1900-01-01 02:00:00.000
15	7773	5	Boeing	London	Singapore	1900-01-01 03:00:00.000	1900-01-01 13:00:00.000
16	7774	5	Boeing	Singapore	London	1900-01-01 14:00:00.000	1900-01-01 06:00:00.000
17	7775	5	Boeing	London	Singapore	1900-01-01 09:00:00.000	1900-01-01 20:00:00.000
18	7776	5	Boeing	Singapore	London	1900-01-01 18:00:00.000	1900-01-01 08:00:00.000
19	7777	5	Boeing	London	Singapore	1900-01-01 18:00:00.000	1900-01-01 06:00:00.000
20	7778	5	Boeing	Singapore	London	1900-01-01 22:00:00.000	1900-01-01 12:00:00.000
21	8881	5	Boeing	London	Paris	1900-01-01 03:00:00.000	1900-01-01 04:00:00.000

4. Вывод

Операции рассмотрены в ходе данной работы являются основными и покрывают большинство запросов пользователей при работе с БД.

Для работы со временем необходимо использовать особый тип – Datetime. Для работы с данным типом предусмотрено множество инструментов, например функция between.