

## Лабораторная работа 6

### ОПЕРАТОРЫ МОДИФИКАЦИИ ДАННЫХ

При работе с базой данных необходимо добавлять, изменять и удалять данные. Для выполнения этих операций используются следующие операторы модификации данных:

- **INSERT** (вставка новых строк)
- **UPDATE** (изменение значения столбцов)
- **MERGE** (слияние строк)
- **DELETE** (удаление строк)

Если при выполнении операторов модификации данных будут нарушены ограничения ссылочной целостности, то операторы не выполняются, а выводится сообщение об ошибке.

Эти операторы не осуществляют вывод измененных данных, отображается только число модифицированных строк. Для вывода данных, которые были изменены, следует добавить предложение

**RETURNING \* | [список столбцов]**

После выполнения операторов **INSERT**, **UPDATE**, **MERGE** будут выведены данные получившиеся в результате модификации, а при выполнении оператора **DELETE** будут выведены строки, которые были удалены.

#### 7.1 Оператор INSERT

Оператор **INSERT** используется для добавления (вставки) новых строк в таблицу. Можно вставить одну строку или несколько строк, полученных в результате выполнения оператора **SELECT**.

Оператор **INSERT** для добавления одной строки имеет следующий формат:

```
INSERT {имя таблицы} [{список столбцов}]
VALUES ({список значений});
```

[список столбцов] нужно указывать в том случае, если список значений не совпадает со списком столбцов таблицы.

Добавление данных о новом товаре.

```
INSERT INTO Products
VALUES (88, 'ASUS X540LB', 4, 1800);
```

При выполнении оператора

```
INSERT INTO Products
VALUES (89, 'ASUS X555LB', 1800)
```

ВОЗНИКНЕТ ОШИБКА

SQL Error [23514]: ОШИБКА: новая строка в отношении "products" нарушает ограничение-проверку "product\_r"  
 Подробности: Ошибочная строка содержит (89, ASUS X555LB, 1800, null).

Причиной ошибки является попытка присвоить столбцу rating\_p значение 1800, а для этого столбца установлено ограничение  $1 \leq \text{rating\_p} \leq 5$ . Эта ошибка не возникнет, если указать столбцы, которым присваиваются значения.

```
INSERT INTO Products(product_id,product_name,price)
VALUES (89, 'ASUS X555LB', 1800)
```

Столбцу rating\_p, который отсутствует в списке, будет присвоено значение NULL.

### **Вставка значений заданных по умолчанию**

При создании таблицы, для каждого столбца, можно задать значение по умолчанию (DEFAULT), например:

- для столбца `order_date` в таблице `Orders` это текущая дата, возвращаемая функцией `CURRENT_DATE`;
- для столбца `status` в таблице `Orders` задано значение по умолчанию `Pending` (в ожидании);

Для того чтобы столбцу, при вставке новых строк, было присвоено значение по умолчанию, нужно в списке значений указать служебное слово `DEFAULT`.

**Запрос 7.1.** Ввести данные о новом заказе, присвоив столбцам `order_date` и `status` значения по умолчанию.

```
INSERT INTO Orders (order_id, customer_id, salesman_id,
order_date, status)
VALUES (105, 18, 175, DEFAULT, DEFAULT)
returning *
```

order_id	customer_id	status	salesman_id	order_date
105	18	Pending	175	2023-06-10

Значение по умолчанию будет присвоено и в том случае, если `DEFAULT` будет отсутствовать в предложении `VALUES`. Например:

```
INSERT INTO Orders (order_id, customer_id, salesman_id)
VALUES (105, 18, 175);
```

Если указать значение `DEFAULT` для столбца, у которого не задано значение по умолчанию, то ему будет присвоено значение `NULL`.

### Вставка нескольких строк

Можно вставить в таблицу несколько строк, сформированных в результате выполнения оператора `SELECT`. В этом случае оператор `INSERT` должен иметь следующий формат:

```
INSERT {имя таблицы} [{список столбцов}]
SELECT [{список столбцов}] {текст запроса};
```

Списки столбцов после имени таблицы и после SELECT должны совпадать. Если список столбцов, которые возвращает запрос, точно соответствует списку столбцов таблицы, то список столбцов после элемента имя таблицы можно не указывать.

Создадим таблицу Products\_total, которая должна содержать данные о товарах и общее количество товара, которое было реализовано.

```
CREATE TABLE Products_Total
( product_id INTEGER PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  rating_p INTEGER,
  quantity INTEGER);
```

**Запрос 7.2.** Заполнить данными таблицу Products\_Total

```
INSERT INTO Products_Total
SELECT pr.product_id,pr.product_name,pr.rating_p, SUM(quantity)
As quantity
FROM Products pr JOIN Order_Items oi
ON (pr.product_id=oi.product_id);
```

Можно создать новую таблицу и заполнить ее данными, используя один оператор, который имеет следующий синтаксис:

```
CREATE TABLE имя таблицы As
SELECT .....
```

**Запрос 7.3.** Создать копию таблицы Products, и заполнить ее данными о товарах, которые ни разу не продавались

```
CREATE TABLE Products_Ns As
SELECT * FROM Products
WHERE product_id NOT IN
(SELECT DISTINCT product_id
FROM Order_Items)
```

## Оператор UPDATE

Оператор UPDATE используется для изменения существующих строк в таблице, и имеет следующий синтаксис:

```
UPDATE {имя таблицы}
SET {столбец} = {значение/выражение/запрос}
WHERE {условия};
```

Этот оператор изменяет значения столбцов тех строк, которые удовлетворяют заданным условиям. Следует обратить внимание на то, что новое значение столбца может быть результатом запроса, который возвращает скалярное значение.

**Запрос 4.** Установить новую зарплату равную 8500 для сотрудника 110.

```
UPDATE Employees
SET salary = 8500
WHERE employee_id=110;
```

**Запрос 5.** Увеличить на 10% зарплату сотрудников работающих в отделе 70.

```
UPDATE Employees
SET salary = salary*1.1
WHERE department_id=70;
```

### **Присвоение значений заданных по умолчанию.**

В рассматриваемой предметной области может быть определено следующее правило: если в отделе нет назначенного начальника, то начальником этого отдела является руководитель предприятия Steven King, employee\_id, которого равен 100. Для обеспечения этого правила, значение по умолчанию, столбца manager\_id в таблице Departments, равно 100.

**Запрос 6.** Присвоить значение по умолчанию столбцу manager\_id в таблице Departments для отдела 10.

```
UPDATE departments
SET manager_id = DEFAULT
WHERE department_id = 10;
```

Можно изменить значения нескольких столбцов в одном операторе UPDATE.

**Запрос 7.** Установить сотруднику 122 новую должность, оклад и рейтинг.

```
UPDATE Employees
SET Job_id = 'SA_MAN',
    salary = 10000,
    rating_e = 4
WHERE employee_id = 122;
```

Присваиваемое значение может быть результатом выполнения запроса, который возвращает одну строку. Этот запрос может возвращать значения одного или нескольких столбцов.

**Запрос 7.8.** Сотруднику 122 изменить значение столбца department\_id на значение, которое имеет этот столбец у сотрудника 147.

```
UPDATE Employees
SET department_id =
```

```
(SELECT department_id FROM Employees
WHERE employee_id = 147)
WHERE employee_id = 122;
```

### Обновление строк с использованием коррелированного подзапроса

Создадим копию таблицы Order\_Items, назовем новую таблицу Order\_Items\_Copy.

```
CREATE TABLE Order_Items_Copy As
SELECT * FROM Order_Items;
```

и добавим в эту таблицу новый столбец rating\_p.

```
ALTER TABLE Order_Items_Copy
ADD Column rating_p integer
```

**Запрос 9.** Заполнить столбец **rating\_p** в таблице Order\_Items\_Copy данными, извлекая их из таблицы Products.

```
UPDATE Order_Items_Copy as oic
set rating_p = (SELECT p.rating_p FROM Products p
                WHERE p.product_id=oic.product_id)
returning *;
```

Фрагмент таблицы Order\_Items\_Copy, после выполнения этого оператора.

order_id	item_id	product_id	quantity	unit_price	rating_p
78	5	79	10	2000.00	1
8	1	34	144	150.00	5
32	1	14	86	700.00	4
35	1	76	99	1160.00	1
60	1	15	36	280.00	4
61	1	16	67	730.00	4
67	1	52	85	500.00	2
87	1	1	57	640.00	4

## Оператор MERGE

Данный оператор позволяет сливать строки из одной таблицы в другую таблицу. Если в таблице приемнике, куда осуществляется слияние, существуют строки, для которых выполняется условие слияния, то выполняются операции обновления (UPDATE), в противном случае выполняется операция вставки новых строк. (INSERT).

Синтаксис оператора MERGE:

```

MERGE INTO [ONLY] {Таблица приемник}
      USING {Таблица или запрос источник}
      ON {условие слияния}
WHEN MATCHED THEN DELETE|UPDATE
      SET {столбец 1} = {значение 1/выражение 1}
      . . . .
      [{Столбец n} = {значение n/выражение n}]
WHEN NOT MATCHED THEN INSERT
      VALUES ({список столбцов});

```

Если перед именем таблицы содержится служебное слово **ONLY**, то соответствующие строки обновляются или удаляются только в таблице приемнике.

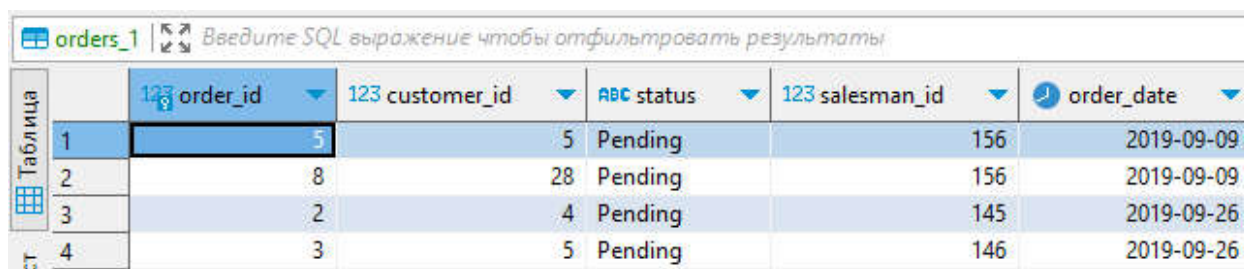
Если **ONLY** не указано, то совпадающие строки также обновляются или удаляются во всех таблицах, наследуемых от таблицы приемнике.

Если этот оператор будет содержать предложение **WHEN MATCHED THEN DELETE**, то в таблице приемнике будут удалены строки, для которых выполняется **условие слияния**, и будут вставлены строки из таблицы источника, для которых это условие не выполняется.

Рассмотрим примеры использования оператора MERGE. На рисунке 7.1. показано содержимое таблицы Orders\_1, которая содержит данные о 4х заказах

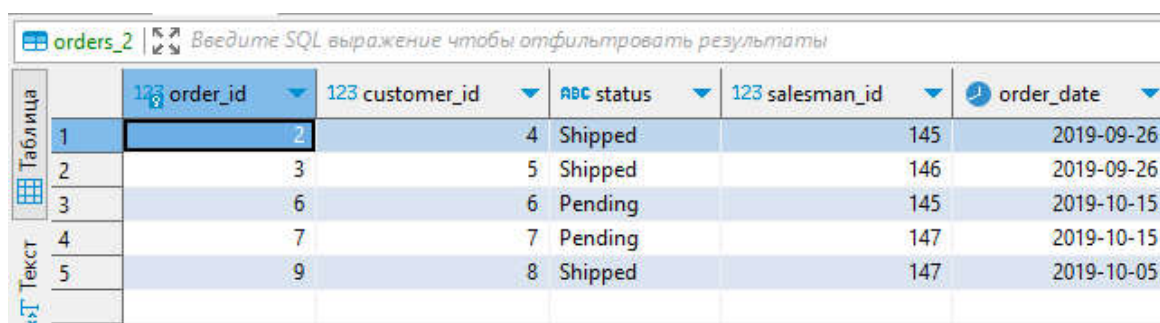


имеющих статус Pending . На рисунке 7.2 показано содержимое таблицы Orders\_2, которая содержит новые данные о заказах. Эта таблица может содержать данные о заказах из таблицы Orders\_1, с измененным значением столбца status, и данные о новых заказах. В результате слияния этих таблиц нужно изменить значение столбца status, и добавить данные о новых заказах.



order_id	customer_id	status	salesman_id	order_date
5	5	Pending	156	2019-09-09
8	28	Pending	156	2019-09-09
2	4	Pending	145	2019-09-26
3	5	Pending	146	2019-09-26

Рис. 1. Содержимое таблицы Orders\_1



order_id	customer_id	status	salesman_id	order_date
2	4	Shipped	145	2019-09-26
3	5	Shipped	146	2019-09-26
6	6	Pending	145	2019-10-15
7	7	Pending	147	2019-10-15
9	8	Shipped	147	2019-10-05

Рис.2. Содержимое таблицы Orders\_2

**Запрос 10.** Выполнить слияние таблиц Orders\_1 и Orders\_2

```

MERGE INTO Orders_1 ord1 USING Orders_2 ord2
ON ord1.order_id = ord2.order_id
WHEN MATCHED THEN
    UPDATE SET status = ord2.status
WHEN NOT MATCHED THEN
    INSERT (order_id, customer_id, status,
            salesman_id, order_date)
VALUES (ord2.order_id, ord2.customer_id, ord2.status,
        ord2.salesman_id, ord2.order_date);

```

orders\_1 Введите SQL выражение чтобы отфильтровать результаты

	123 order_id	123 customer_id	ABC status	123 salesman_id	order_date
1	5	5	Pending	156	2019-09-09
2	8	28	Pending	156	2019-09-09
3	2	4	Shipped	145	2019-09-26
4	3	5	Shipped	146	2019-09-26
5	6	6	Pending	145	2019-10-15
6	7	7	Pending	147	2019-10-15
7	9	8	Shipped	147	2019-10-05

Рисунок 7.3. Содержимое таблицы Orders\_1 после выполнения запроса 10.

Анализируя эти данные можно установить, что в процессе выполнения запроса 10:

- данные о заказах 5 и 8 не изменились.
- изменен статус заказов 2 и 3;
- добавлены данные о заказах 6,7,9;

## Оператор DELETE

Оператор DELETE используется для удаления существующих строк в таблице, и имеет следующий синтаксис:

**DELETE FROM   таблица**  
**[WHERE условия] ;**

При наличии предложения WHERE удаляются только те строки, которые удовлетворяют заданным условиям. Если предложение WHERE отсутствует, то будут удалены все строки.

В запросах, рассматриваемых в этом разделе, будет изменяться содержимое таблиц: Customers, Products, Orders, Orders\_Items. Для того чтобы сохранить содержимое этих таблиц, которое мы будем использовать далее, были созданы копии этих таблиц.

### Запрос 11. Удалить данные о товаре 77.

```
DELETE FROM Products_Copy
WHERE product_id = 77
returning *;
```

product_id	product_name	rating_p	price
77	Logitech G810 Orion Spectrum (920-007750)	1	40.00

Запрос успешно выполнен, это означает, что товар 77 не продавался.

### Запрос 12. Удалить данные о товарах, которые ни разу не продавались.

```
DELETE FROM Products_Copy
WHERE product_id NOT IN
(SELECT DISTINCT product_id
FROM order_items_copy);
```

В этом запросе простой подзапрос извлекает из таблицы **Order\_Items\_Copy** список товаров, которые продавались. Основной запрос последовательно просматривает товары и удаляет те товары, которых нет в этом списке.

## Задание

**Задача 1.** Создайте таблицу **EMP**(employee\_id, first\_name, last\_name, hire\_date, rating\_e, working, layer) и заполните данными о сотрудниках, работающих в отделе 80. Столбцу working присвоить значение, равное количеству полных лет, которые проработал сотрудник. А значение столбца layer зависит от значения столбца rating\_e. Если rating\_e равен 5 то layer= 'A', если rating\_e равен 4 или 3 то layer= 'B', у остальных сотрудников layer= 'C'.

**Задача 2.** Увеличить на 1 rating\_e сотрудников, которые осуществили продажи на сумму более 1000000 и имеют rating\_e < 5.

**Задача 3.** Добавить в таблицу **Employees\_Copy** столбец `emp_sales` и присвоить ему значение общей стоимости продаж осуществленных каждым сотрудником.

**Задача 4.** Выполните слияние таблицы **Orders1** только с теми строками таблицы **Orders2**, в которых заказы находятся в состоянии 'Shipped'.

**Задача 5.** Создайте таблицу **Order\_Items\_New**, которая содержит данные о новых продажах, и заполните ее данными. Выполните слияние таблицы **Order\_Items\_Copy** с таблицей **Order\_Items\_New**. Алгоритм слияния: если в таблице **Order\_Items\_Copy** существует строка, у которой значения столбцов `order_id`, `product_id` совпадают со значениями этих столбцов в добавляемой строке из таблицы **Order\_Items\_New**, то обновить значение столбца `quantity`, в противном случае вставить новую строку.

**Задача 6.** Удалить данные об отмененных заказах (`status = 'Canceled'`), с даты оформления которых прошло более 5 лет.

**Задача 7.** Удалить из таблицы **Order\_Items\_Copy** данные о продаже товаров, которые нарушают правило: рейтинг продавца должен больше или равен рейтингу товара.