ЯЗЫК МАНИПУЛИРОВАНИЯ ДАННЫМИ (DML - Data Manipulation Language)

Язык манипулирования данными (**DML**) включает операторы, управляющие содержанием таблиц базы данных и извлекающими информацию из этих таблиц.

DML включает в себя четыре основные команды:

- **INSERT** – вставляет данные в таблицу;
- **UPDATE** – обновляет данные таблицы;
- **DELETE** – удаляет данные из таблицы;
- **SELECT** – возвращает (выбирает) данные из объекта базы данных.

КОМАНДА INSERT

Оператор **INSERT** предназначен для добавления строк в таблицу или представление данных.

Синтаксис команды:

INSERT INTO имя_таблицы/представления (столбец1, столбец2,...)

VALUES(значение1, значение2,...);

INSERT INTO имя_таблицы/представления (столбец1, столбец2,...) SELECT...;

Имена в списке столбцов могут быть перечислены в любом порядке. В столбцы, не указанные в списке, заносится пустое значение. Все столбцы с признаком NOT NULL должны быть указаны, и иметь предназначающиеся для них значения. В предложении VALUES перечисляются конкретные значения столбцов в добавляемой строке. Каждый указанный столбец должен иметь соответствующее ему значение в предложении VALUES. Типы данных значения и столбца должен быть совместимы или преобразуемы. Значения типа CHAR, VARCHAR2 и DATE надо заключать в апострофы (‘абв’). Чтобы добавить строки из другой таблицы, следует использовать подзапрос. Оператор SELECT в этом подзапросе должен извлекать значения для каждого перечисленного столбца.

Рассмотрим примеры вставки данных в таблицы. В первом примере мы производим вставку данных во всех столбцы таблицы и в том порядке, в каком она была создана, поэтому здесь мы можем опустить спецификацию столбцов для вставки.

```
INSERT INTO s_fiz_lic VALUES (1,'Иванов','Иван','Иванович','ПВС Западного округа','3310 124568');
INSERT INTO s_fiz_lic VALUES (2,'Петров','Петр','Петрович', 'ПВС Центрального округа','3250 145868');
INSERT INTO s_fiz_lic VALUES (3,'Сидоров','Иван','Иванович', 'ПВС Западного округа','7810 124879');
INSERT INTO dolj VALUES (1, ‘Слесарь’);
INSERT INTO dolj VALUES (2, ‘Программист’);
INSERT INTO dolj VALUES (3, ‘Бухгалтер’);
```

В следующем примере в последние два столбца вставляем NULL значения, для того чтобы указать, что у нас отсутствуют данные.

```
INSERT INTO s_fiz_lic VALUES (4, ’Иванов’,
’Иван’,’Иванович’,NULL,NULL);
```

В данном примере мы производим аналогичную вставку, но со спецификацией столбцов, которая определяет в каком порядке вставлять данные.

```
INSERT INTO s_fiz_lic (kod,im,otch,fam, p_vidan,p_ser_nom) VALUES (5, 'Иван', 'Иванович', 'Иванов', 'ПВС Западного округа', '3310 124568');
```

Рассмотрим пример вставки данных с использованием запроса. В этом примере мы создаем дублирующие данные таблицы «s_fiz_lic», при этом для всех выбранных строк из таблицы, мы увеличиваем значение поля «kod» на 5. В результате в таблице мы получим дубликаты строк физизческих лиц, но с разными значениями в поле «kod». В последней строке мы производим фиксацию транзакции.

```
INSERT INTO s_fiz_lic (SELECT kod+5,im,otch,fam, p_vidan,p_ser_nom FROM s_fiz_lic);
COMMIT;
```

В следующем примере мы вставляем данные из таблицы «s_fiz_lic» в таблицу «s_fiz_lic3».

```
INSERT INTO s_fiz_lic3 (kod, fio) (SELECT kod, UPPER(fam||' '||im||' '||otch) FROM s_fiz_lic);
COMMIT;
```

КОМАНДА UPDATE

Оператор **UPDATE** заменяет значения одного или нескольких указанных столбцов на значения выражений или результат запроса.

Синтаксис команды:

UPDATE таблица/имя_предст_данных SET

столбец_имя = выражение

столбец_имя = (SELECT_с_одним_результатом)

(столбец_имя, столбец_имя,...) = оператор_SELECT

WHERE_предложение;

Оператор SELECT в этом запросе должен возвращать как минимум одну строку и обеспечивать значения для каждого столбца, стоящего слева от знака =. Этот оператор SELECT не может содержать фразы INTO. Для определения набора строк, подлежащих обновлению, используется предложение WHERE. В нем указываются условия, которым должна отвечать обновляемая строка. Если предложение WHERE опустить, то будут обновлены все строки.

Рассмотрим примеры обновления данных. В первом примере строке с кодом 5, мы устанавливаем новые значения поля «fam», «im», «otch».

```
UPDATE s_fiz_lic SET Fam='Сидоров', im='Петр', otch='Иванович' WHERE kod=5;
```

В следующем примере мы соединяем значения полей «fam», «im», «otch» в верхнем регистре строки с кодом 5 из таблицы «s_fiz_lic3» и присваиваем его полю «fio» таблицы «s_fiz_lic» с кодом 4.

```
UPDATE s_fiz_lic3 SET fio=(SELECT UPPER(fam||' '||im||' '||otch) FROM s_fiz_lic WHERE kod=4) WHERE kod=5;
```

Обновим таблицу «s_fiz_lic3» проставив код должности из таблицы «dolj».

```
UPDATE  s_fiz_lic3 SET kod_dolj=1 WHERE kod IN (1,2,5,6);
UPDATE  s_fiz_lic3 SET kod_dolj=2 WHERE kod IN (3,7);
UPDATE  s_fiz_lic3 SET kod_dolj=3 WHERE kod IN (8);
COMMIT;
```

КОМАНДА DELETE

Оператор **DELETE** производит удаление строк таблицы. Для определения набора строк, подлежащих удалению, используется предложение WHERE. В нем указываются условия, которым должна отвечать удаляемая строка. Если предложение WHERE опустить, то будут удалены все строки.

Синтаксис команды:

DELETE FROM таблица/имя_предст_данных WHERE_предложение;

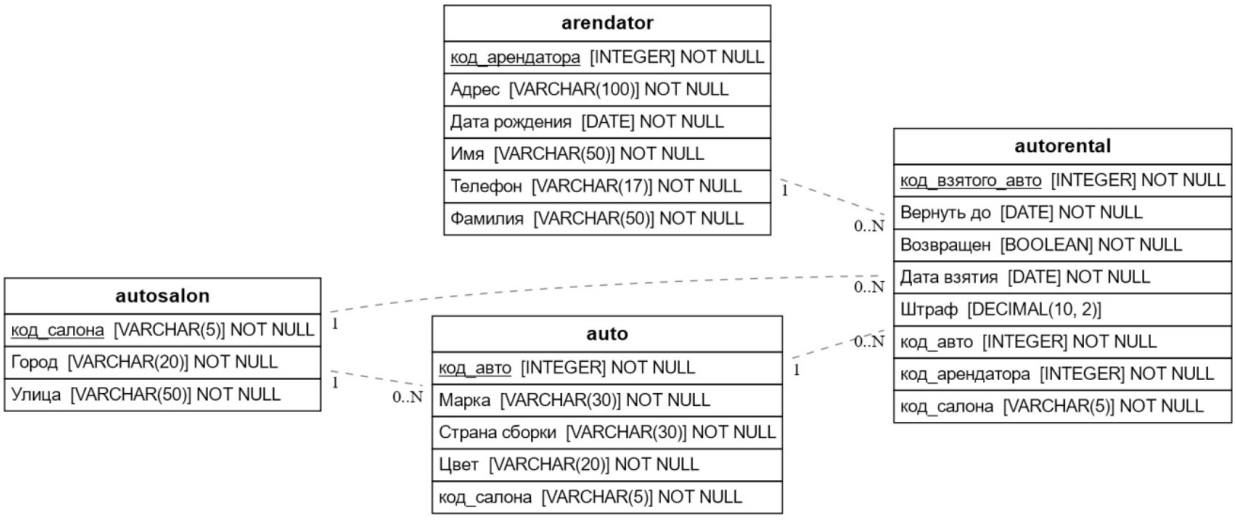
Рассмотрим примеры удаления данных. В первом примере мы производим удаление строки из таблицы с кодом 5.

```
DELETE FROM  s_fiz_lic WHERE kod=5;
```

В следующем примере выполняется удаление всех строк таблицы «s_fiz_lic».

```
DELETE FROM  s_fiz_lic;
```

Задание для аудиторной и самостоятельной работы:



Необходимо реализовать описанную выше схему отношений. При создании таблиц требуется учитывать следующие ограничения:

1) Для всех кодов использовать целочисленные значения от 0 до n, где n количество кортежей в таблице (допускается использование автоинкремента);

2) Код салона — некоторое число, фиксированной длины 5 символов (цифры от 0 до 9). Каждый разряд является значащим, даже если он равен нулю. Недопустимо: значения меньше 5 цифр (например, 324) или больше 5 цифр (например, 111324). Допустимо: значения с ведущими нулями (например, 00324);

3) Город — это строка длины от 3 до 20 символов, начинающаяся с заглавной буквы от “А” до “Я”;

4) Улица — это строка длины от 3 до 20 символов, начинающаяся с заглавной буквы от “А” до “Я” и имеет номер дома. Пример: “Ставропольская д.149”;

5) Атрибуты: город, улица, марка, цвет, дата взятия, вернуть до, имя, фамилия, телефон не могут быть пустыми и должны быть приближены к реальности (недопустимо в качестве имени использовать значения user_name_1 и т.д.);

6) Телефон — строка длины 17, начинающаяся с символа “+”, имеет символ “(“ и “)”, так же два символа “-“. Пример: +7(395)181-15-64;

После создания таблиц необходимо в одну из таблиц добавить 5000 строк, во все остальные таблицы не менее 500 строк. После успешного выполнения задания очистить таблицы и удалить их из базы вместе с ограничениями целостности.

Для выполнения задания, возможно, понадобится функции substr, синтаксис:

Substr(**String**, **Position**, **Substring _length**)

String – рассматриваемая строка, Position – начальная позиция, Substring _length – общее количество символов, выделяемых в подстроке.

Примеры:

Запрос:

```
SELECT SUBSTR('ABCDEFG',3,4) "Substring"
    FROM DUAL;
```

Результат:

```
Substring
---------
CDEF
```

Запрос:

```
SELECT SUBSTR('ABCDEFG',-5,4) "Substring"
    FROM DUAL;
```

Результат:

```
Substring
---------
CDEF
```