### 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:**

INSERTINTO 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:**

DELETEFROM таблица/имя\_предст\_данных WHERE\_предложение;

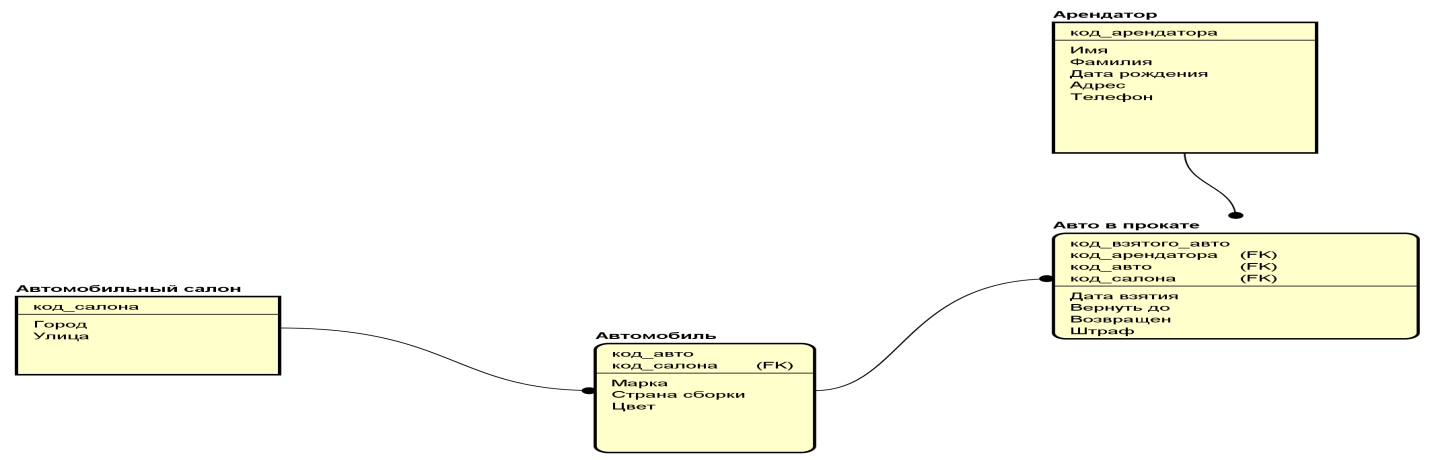
Рассмотрим примеры удаления данных. В первом примере мы производим удаление строки из таблицы с кодом 5.

|  |
| --- |
| DELETE FROM s\_fiz\_lic WHERE kod=5; |

В следующем примере выполянется 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 числа.

2). Код салона – некоторое число, имеющее 5 разрядов, т.е. допустимым значением кода является число 12032, однако числа 324 или 200000 являются недопустимыми значениями этого столбца.

3). Город – это строка, начинающаяся с заглавной буквы от “А” до “Я”.

4). Атрибуты: город, улица, марка, цвет, дата взятия, вернуть до, имя, фамилия, телефон не могут быть пустыми.

5). Телефон – некоторая строка (длины от 12 и до 15 символов), начинающаяся с символа “+”.

После создания таблиц необходимо добавить в каждую из таблиц не менее 5 строк. После успешного выполнения задания очистить таблицы и удалить их из базы вместе с ограничениями целостности.

Частичная реализация указанного задания в командах:

create table auto\_sal

( id\_s NUMBER(5) Primary KEY,

city varchar(25),

street varchar2(25));

alter TABLE auto\_sal add CONSTRAINT chk\_id\_s CHECK (id\_s >=10000 and id\_s<= 99999);

alter table auto\_sal MODIFY (city not null, street not null);

CREATE TABLE automob

( id\_a Number(10),

id\_s Number(5) REFERENCES auto\_sal(id\_s),

mark\_a VARCHAR2(25),

country\_a varchar2(25),

colour varchar2(25));

alter TABLE automob add CONSTRAINT aut\_pk PRIMARY KEY (id\_a, id\_s) ENABLE;

alter TABLE auto\_sal MODIFY (streeat varchar2(50));

INSERT INTO auto\_sal VALUES (10000,'Краснодар','ул. Ставропольская');

INSERT INTO auto\_sal VALUES (10001,'Краснодар','ул. Северная');

INSERT INTO auto\_sal VALUES (10002,'Краснодар','ул. Школьная');

INSERT INTO auto\_sal VALUES (10003,'Краснодар','ул. Красная');

INSERT INTO auto\_sal VALUES (10004,'Краснодар','ул. Уральская');

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