**Язык определения данных DDL**

Язык определения данных (Data Definition Language, DDL) позволяет создавать и изменять структуру объектов *базы данных*, например, создавать и удалять *таблицы*. Основными командами языка DDL являются следующие: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX.

*Физическая модель реляционной базы данных* содержит *базовые таблицы*. Для определения и создания таблиц в *SQL-92* предусмотрена *команда* *CREATE* *TABLE*, которая определяет *имя таблицы*, имена и физический порядок колонок для нее, тип каждой колонки, а также некоторые указания для *СУБД*, такие как *определение* первичного или внешнего ключа, требования на запрет неопределенных значений в колонке таблицы и т.п. Полный *формат команды* CREATE TABLE для каждой *СУБД* приводится в соответствующем документе с названием типа "Справочное руководство по *SQL* для СУБДѕ".

Базовый синтаксис оператора *создания таблицы* имеет следующий вид:

<определение\_таблицы> ::=

 CREATE TABLE имя\_таблицы

 (имя\_столбца тип\_данных

 [NULL | NOT NULL ] [,...n])

**Пример.** Создать *таблицу* Товар:

CREATE TABLE Товар

(Название VARCHAR(50) NOT NULL,

 Цена MONEY NOT NULL,

 Тип VARCHAR(50) NOT NULL,

 Сорт VARCHAR(50),

 ГородТовара VARCHAR(50))

Базовый синтаксис оператора *изменения таблицы* имеет следующий вид

ALTER TABLE имя\_таблицы

 {[ADD [COLUMN] имя\_столбца тип\_данных [

 NULL | NOT NULL ]]

 | [DROP [COLUMN] имя\_столбца]}

**Пример.** Добавить в *таблицу*  Клиент поле для номера расчетного счета.

ALTER TABLE Клиент ADD Рас\_счет CHAR(20)

**Ограничения**

|  |
| --- |
| Таблица 9.1. Ограничения на объекты реляционной базы данных |
|  | Ограничение | Описание |
| **1** | **CHECK** | гарантирует, что значения находятся в границах специфицированного интервала, задаваемого предикатом |
| **2** | **DEFAULT** | помещает значение по умолчанию в колонку. Гарантирует, что колонка всегда имеет значение |
| **3** | **FOREIN KEY** | гарантирует, что значения существует как значение в колонке первичного ключа другой таблицы. Обеспечивает процедуры удаления дочерних строк при удалении связанных с ней родительских |
| **4** | **NOT NULL** | гарантирует, что колонка всегда содержит значение |
| **5** | **PRIMARY KEY** | гарантирует, что колонка всегда содержит значение и оно уникально в таблице |
| **6** | **UNIQUE** | гарантирует, что значение будет уникальным в таблице |

**Добавление CHECK-ограничения в спецификацию колонки**

Ограничение CHECK позволяет выполнять проверку содержимого колонки относительно некоторых условий и списка значений. Она налагается с помощью предложения CHECK. Для добавления этого ограничения нужно после объявления столбца в спецификации колонки определить синтаксическую конструкцию CHECK (предикат). Согласно требованиям стандарта с помощью ключевого слова VALUE в предикате вы ссылаетесь на значение колонки. Но практически во всех диалектах для этой цели используется имя колонки.

**Пример.** В учебной базе данных в таблице EMPLOYEE для сотрудников может указываться признак пола: 0 - мужской, 1 - женский. Бизнес-правило предметной области для значений этого поля может быть сформулировано так:

Лицо, принимаемое на работу, может иметь один из двух допустимых признаков пола.

Тогда *спецификация колонки* может выглядеть так:

SEX int NOT NULL CHECK (SEX=0 OR SEX=1),

**Использование опции DEFAULT**

Опция DEFAULT заставляет СУБД размещать значение по умолчанию в колонке, когда кортеж вставляется в таблицу и никакого значения колонки не представлено. Чтобы указать значение по умолчанию, нужно в *спецификацию колонки* добавить ключевое слово "DEFAULT" и после него указать любое значение, являющееся достоверным экземпляром типа данных колонки.

**Пример.** Для нашей учебной базы данных мы могли бы определить значение по умолчанию для числовых колонок в таблице EMPLOYEE:

SAL dec(9,2) DEFAULT(0),

COMM dec(9,2) DEFAULT(0),

FINE dec(9,2) DEFAULT(0),

**Добавление NOT NULL ограничения в спецификацию колонки**

NOT NULL ограничение гарантирует, что колонка всегда содержит значения. СУБД не будет разрешать вставлять или обновлять строку таблицы, если в ней существует колонка с ограничением NOT NULL, а данных для этой колонки в добавляемой строке не представлено. Как мы уже видели выше, для колонок первичного ключа это ограничение нужно устанавливать всегда.

**Пример.** Для нашей учебной базы данных действует правило, что сотрудник всегда должен иметь имя и фамилию. Чтобы удовлетворить этому правилу, нужно определить следующую спецификацию колонок ENAME и LNAME в таблице EMPLOYEE:

ENAME char(25) NOT NULL,

LNAME char(10) NOT NULL,

Иногда ограничение NOT NULL используется вместе с опцией DEFAULT, как это было определено в спецификации колонки HIREDATE (дата приема на работу) в таблице EMPLOYEE:

HIREDATE date NOT NULL WITH DEFAULT,

**Добавление ограничения UNIQUE в спецификацию колонки**

Ограничение UNIQUE гарантирует уникальность значения данных в колонке. Оно используется, если нужно следить за тем, чтобы значения колонки, не являющейся первичным ключом, были уникальны в таблице. При этом проверяется уникальность всех значений, отличных от NULL.

Пример. В учебной базе данных в таблице EMPLOYEE используется номер социальной страховки SSECNO, для которого бизнес-правило состоит в том, что для каждой персоны, если она имеет такой номер, он должен быть уникальным. Установить уникальность этих номеров можно следующей спецификацией колонки:

SSECNO char(10) UNIQUE,

Ограничение UNIQUE можно определить также в конце команды CREATE TABLE в следующей синтаксической форме: UNIQUE (SSECNO).

Описание ограничений может быть, как в теле оператора определения объектов базы данных, так и выделено в отдельный оператор CONSTRAINT.

**Пример A: Таблица с полем, значения которого выбираются из значений предопределенного домена**

Создадим таблицу «Регион»

CREATE TABLE REGION (

 REGION REGION\_NAME NOT NULL PRIMARY KEY,

 POPULATION INTEGER NOT NULL);

Определим домен REGION\_NAME, к которому относятся значения поля REGION например, следующим образом:

CREATE DOMAIN REGION\_NAME

 AS VARCHAR(40;

**Пример B: Таблица с предложением UNIQUE как на уровне поля, так и на уровне таблицы**

CREATE TABLE GOODS (

 MODEL SMALLINT NOT NULL UNIQUE,

 NAME CHAR(10) NOT NULL,

 ITEMID INTEGER NOT NULL, CONSTRAINT MOD\_UNIQUE

UNIQUE (NAME, ITEMID));

**Пример C: Таблица с определением первичного ключа, внешнего ключа и конструкции CHECK, а также символьных массивов**

CREATE TABLE JOB (

JOB\_CODE JOBCODE NOT NULL,

JOB\_GRADE JOBGRADE NOT NULL,

JOB\_REGION REGION\_NAME NOT NULL,

JOB\_TITLE VARCHAR(25) NOT NULL,

MIN\_SALARY SALARY NOT NULL,

MAX\_SALARY SALARY NOT NULL,

JOB\_REQ BLOB(400,1),

LANGUAGE\_REQ VARCHAR(15),

PRIMARY KEY (JOB\_CODE, JOB\_GRADE, JOB\_REGION),

FOREIGN KEY (JOB\_REGION) REFERENCES REGION (REGION),

CHECK (MIN\_SALARY < MAX\_SALARY));

Данный пример создает таблицу, содержащую информацию о работах (профессиях). Типы полей основаны на доменах JOBCODE, JOBGRADE, REGION\_NAME и SALARY. Определен массив LANGUAGE\_REQ, состоящий из 5 элементов типа VARCHAR(15). Кроме того, введено поле JOB\_REQ, имеющее тип BLOB с подтипом 1 (текстовый блоб) и размером сегмента 400. Для таблицы определен первичный ключ, состоящий из трех полей JOB\_CODE, JOB\_GRADE и JOB\_REGION. Далее, определен внешний ключ (JOB\_REGION), ссылающийся на поле REGION таблицы REGION. И, наконец, включено предложение CHECK, позволяющее производить проверку соотношения для двух полей и вызывать исключительное состояние при нарушении такого соотношения.

**Пример D: Таблица с вычисляемым полем**

CREATE TABLE SALARY\_HISTORY (

 EMP\_NO EMPNO NOT NULL,

 CHANGE\_DATE DATE DEFAULT "NOW" NOT NULL,

 UPDATER\_ID VARCHAR(20) NOT NULL,

 OLD\_SALARY SALARY NOT NULL,

 PERC\_CHANGE DOUBLE PRECISION DEFAULT 0 NOT NULL

 CHECK (PERC\_CHANGE BETWEEN -50 AND 50),

 NEW\_SALARY COMPUTED BY

 (OLD\_SALARY + OLD\_SALARY \* PERC\_CHANGE / 100),

PRIMARY KEY (EMP\_NO, CHANGE\_DATE, UPDATER\_ID),

FOREIGN KEY (EMP\_NO) REFERENCES EMPLOYEE (EMP\_NO));

Данный пример создает таблицу, где среди других полей имеется вычисляемое (физически не существующее) поле NEW\_SALARY, значение которого вычисляется по значениям двух других полей (OLD\_SALARY и PERC\_CHANGE).

**Пример создания БД Библиотека**

В БД «b\_library**»** создадим таблицу авторов «**tAuthors»** с такими столбцами: AuthorId, AuthorFirstName, AuthorLastName, AuthorAge:

CREATE TABLE tAuthors (
AuthorId             INT              IDENTITY (1, 1) NOT NULL, PRIMARY KEY,
AuthorFirstName      NVARCHAR (20)    NOT NULL,
AuthorLastName       NVARCHAR (20)    NOT NULL,
AuthorAge            INT               NOT NULL
);

Заполним нашу таблицу таким авторами: Александр Пушкин, Сергей Есенин, Джек Лондон, Шота Руставели и Рабиндранат Тагор. Для этого используем такой SQL запрос:

INSERT tAuthors VALUES
('Александр', 'Пушкин', '37'),
('Сергей', 'Есенин', '30'),
('Джек', 'Лондон', '40'),
('Шота', 'Руставели', '44'),
('Рабиндранат', 'Тагор', '80');

SELECT \* FROM tAuthors;

Воспользуемся нашей предыдущей таблицей «**tAuthors»**и создадим дополнительно еще одну таблицу с книгами этих авторов – «**tBooks»**. В качестве идентификатора авторов книг используем значение AuthorId из «**tAuthors»**, а название книги - BookTitle**.**

CREATE TABLE tBooks (
BookId               INT              IDENTITY (1, 1) NOT NULL, PRIMARY KEY
BookTitle            NVARCHAR (20)    NOT NULL,
AuthorId               INT              NOT NULL, FOREIGN KEY REFERENCES tAuthors (AuthorId)
);

Заполним «**tBooks»**такими книгами:

INSERT tBooks VALUES
('Руслан и Людмила', '1'),
('Кавказский пленник', '1'),
('Евгений Онегин ', '1'),
('Радуница', '2'),
('Преображение', '2'),
('Мартин Иден', '3'),
('Морской волк', '3'),
('Белый Клык', '3');

**Задание на самостоятельное рассмотрение**



**Задание:**

1. Напишите серию операторов CREATE TABLE.
2. Добавьте ограничение CHECK, указывающее, что Производитель должен быть одним из заданного набора (HР, Lenovo…).
3. Добавьте ограничения на размер Частота\_процессора (например, от 0,8 до 2,5 ГГц).
4. Укажите первичные ключи.
5. Добавьте внешние ключи.

Заполните таблицы данными, не менее 7 сотрудников, 3 компьютера, 7 назначений

**Просмотр кода в Access**

Создание таблицы

1. На вкладке **Создание** в группе **макрос и код** нажмите кнопку **Конструктор запросов**.
2. Закройте диалоговое окно **Добавление таблицы**.
3. На вкладке " **Конструктор** " в группе **Тип запроса** нажмите кнопку **Определение данных**.

Скрытые бланке и отображается вкладка объекта SQL представления.

1. Введите следующую инструкцию SQL:

**Создание таблицы автомобилей (имя TEXT(30), год TEXT(4), валюты цена)**

1. На вкладке **Конструктор** в группе **Результаты** нажмите кнопку **Выполнить**.