**Лабораторная работа 4**

**Создание базы данных средствами SQL**

# **Теоретическая часть**

## 

**Создание таблиц**

Выполняется с помощью оператора CREATE TABLE. Оператор создает пустую таблицу. Значения в таблицу вводятся с помощью команды INSERT INTO. Оператор CREATE TABLE определяет имя таблицы и множество поименованных столбцов в указанном порядке. Для каждого столбца определен тип. Синтаксис команды следующий:

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype,

....

);

При создании или изменении структуры таблицы могут быть определены ограничения на вводимые значения. В этом случае будет отвергаться любое значение, которое не соответствует заданным ограничениям.

Можно задать ограничение, действующее на один столбец или на группу столбцов. Ограничение на один столбец добавляется после определения типа данных столбца и действует только на значения этого столбца. Ограничение на группу столбцов называется ограничением на таблицу. Оно размещается после определения последнего столбца и действует на столбцы, которые указываются в скобках после этого ограничения.

Существуют следующие типы ограничений:

1. Ограничение NOT NULL запрещает использование в указанном столбце NULL-значений. Такое ограничение может быть указано только как ограничение на столбец.

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(20) NOT NULL,

    LastName CHARACTER VARYING(20) NOT NULL,

    Age INTEGER

);

2. Ограничение UNIQUE. Ограничение разрешает использовать только уникальные значения для указанных столбцов.

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(20),

    LastName CHARACTER VARYING(20),

    Email CHARACTER VARYING(30) UNIQUE,

    Phone CHARACTER VARYING(30) UNIQUE,

    Age INTEGER

);

В данном случае столбцы, которые представляют электронный адрес и телефон, будут иметь уникальные значения. И мы не сможем добавить в таблицу две строки, у которых значения для этих столбцов будет совпадать.

Также мы можем определить этот атрибут на уровне таблицы:

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(20),

    LastName CHARACTER VARYING(20),

    Email CHARACTER VARYING(30),

    Phone CHARACTER VARYING(30),

    Age INTEGER,

    UNIQUE(Email),

    UNIQUE(Phone)

);

3. Ограничение первичных ключей PRIMARY KEY.

Простой первичный ключ задается как ограничение на конкретный столбец.

Пример.

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(30),

    LastName CHARACTER VARYING(30),

    Email CHARACTER VARYING(30),

    Age INTEGER

)

Первичный ключ уникально идентифицирует строку в таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом SERIAL, они могут представлять любой другой тип.

Установка первичного ключа на уровне таблицы:

CREATE TABLE Customers

(

    Id SERIAL,

    FirstName CHARACTER VARYING(30),

    LastName CHARACTER VARYING(30),

    Email CHARACTER VARYING(30),

    Age INTEGER,

    PRIMARY KEY(Id)

);

Составные первичные ключи описываются как ограничения на таблицу, то есть после определения всех столбцов.

CREATE TABLE OrderLines

(

    OrderId INTEGER,

    ProductId INTEGER,

    Quantity INTEGER,

    Price MONEY,

    PRIMARY KEY(OrderId, ProductId)

);

1. Ограничение на проверку CHECK. Задает множество возможных значений атрибута. Оно записывается как ограничение на столбец или таблицу. Для одного столбца может быть задано несколько ограничений.

Пример.

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(20),

    LastName CHARACTER VARYING(20),

    Age INTEGER DEFAULT 18 CHECK(Age >0 AND Age < 100),

    Email CHARACTER VARYING(30) UNIQUE CHECK(Email !=''),

    Phone CHARACTER VARYING(20) UNIQUE CHECK(Phone !='')

);

1. Ограничение DEFAULT. Позволяет задать значение по умолчанию. Ограничение задается только как ограничение на столбец.

Пример.

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    FirstName CHARACTER VARYING(20),

    LastName CHARACTER VARYING(20),

    Age INTEGER DEFAULT 18

);

1. Ограничение внешнего ключа FOREIGN KEY.

Для связи между таблицами применяются внешние ключи. Внешний ключ устанавливается для столбца из зависимой, подчиненной таблицы (referencing table), и указывает на один из столбцов из главной таблицы (referenced table). Как правило, внешний ключ указывает на первичный ключ из связанной главной таблицы.

Общий синтаксис установки внешнего ключа на уровне столбца:

REFERENCES главная\_таблица (столбец\_главной\_таблицы)

    [ON DELETE {CASCADE|RESTRICT}]

    [ON UPDATE {CASCADE|RESTRICT}]

Чтобы установить связь между таблицами, после ключевого слова **REFERENCES** указывается имя связанной таблицы и далее в скобках имя столбца из этой таблицы, на который будет указывать внешний ключ. После выражения REFERENCES может идти выражение **ON DELETE** и **ON UPDATE**, которые уточняют поведение при удалении или обновлении данных.

Общий синтаксис установки внешнего ключа на уровне таблицы:

FOREIGN KEY (стобец1, столбец2, ... столбецN)

    REFERENCES главная\_таблица (столбец\_главной\_таблицы1, столбец\_главной\_таблицы2, ... столбец\_главной\_таблицыN)

    [ON DELETE {CASCADE|RESTRICT}]

    [ON UPDATE {CASCADE|RESTRICT}]

Например, определим две таблицы и свяжем их посредством внешнего ключа:

CREATE TABLE Customers

(

    Id SERIAL PRIMARY KEY,

    Age INTEGER,

    FirstName VARCHAR(20) NOT NULL

);

CREATE TABLE Orders

(

    Id SERIAL PRIMARY KEY,

    CustomerId INTEGER,

    Quantity INTEGER,

    FOREIGN KEY (CustomerId) REFERENCES Customers (Id)

);

При задании внешнего ключа можно определить стратегию поддержки ссылочной целостности отдельно для удаления и изменения записей в подчиненной таблице. Стратегия для удаления указывается после слов ON DELETE, а стратегия для изменения - после слов ON UPDATE.

**CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.

**RESTRICT**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.

**NO ACTION**: действие по умолчанию, предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. И генерирует ошибку. В отличие от RESTRICT выполняет отложенную проверку на связанность между таблицами.

**SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение NULL.

**SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты DEFAULT. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение NULL.

Оператор CONSTRAINT. Установка имени ограничений.

С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничений. В качестве ограничений могут использоваться PRIMARY KEY, UNIQUE, CHECK.

Имена ограничений можно задать на уровне столбцов. Они указываются после CONSTRAINT перед атрибутами:

CREATE TABLE Customers

(

    Id SERIAL CONSTRAINT customer\_Id PRIMARY KEY,

    Age INTEGER CONSTRAINT customers\_age\_check CHECK(Age >0 AND Age < 100),

    FirstName CHARACTER VARYING(20) NOT NULL,

    LastName CHARACTER VARYING(20) NOT NULL,

    Email CHARACTER VARYING(30) CONSTRAINT customers\_email\_key UNIQUE,

    Phone CHARACTER VARYING(20) CONSTRAINT customers\_phone\_key UNIQUE

);

## Добавление строк в таблицу

Для добавления строк в существующую таблицу применяется оператор INSERT INTO.

INSERT INTO <таблица> [(<список столбцов>)]

VALUES (<список значений>)

Количество элементов в обоих списках должны быть одинаковым, причем тип каждого значения из списка значений должен соответствовать типу данных столбца, в которое заносится значение.

Пример. Добавить информацию о новой детали в таблицу D.

INSERT INTO STUD (studnum, studname, studgroup, studyear)

VALUES (5, ’Орлов’, 140, 2004)

Если перечислены значения всех столбцов, и они включены в список в том же порядке, в котором следуют в описании таблицы, то список столбцов можно не указывать.

## 

## **Изменение структуры таблицы**

Для модификации структуры и ограничений существующей таблицы используется оператор ALTER TABLE.

Одна инструкция ALTER TABLE может быть применена только к одному столбцу или ограничению.

Синтаксис инструкции ALTER TABLE имеет формы:

1. Добавление столбца к таблице.

ALTER TABLE <имя таблицы>

ADD <имя столбца> <тип>;

1. Добавление ограничения к таблице.

ALTER TABLE <имя таблицы>

ADD CONSTRAINT <имя ограничения> <ограничение>;

1. Изменение свойств столбца.

ALTER TABLE <имя таблицы>

ALTER COLUMN <имя столбца> <тип> [<ограничение>];

1. Удаление столбца из таблицы.

ALTER TABLE <имя таблицы>

DROP COLUMN <имя столбца>;

1. Удаление ограничения таблицы.

ALTER TABLE <имя таблицы>

DROP CONSTRAINT <имя ограничения>;

## Удаление таблиц

Удаляет таблицу из БД. Удалять можно только пустые таблицы.

DROP TABLE <имя таблицы>

Например,

DROP TABLE PD

Команду DROP TABLE нельзя использовать для удаления таблицы, на которую ссылается ограничение FOREIGN KEY. Сначала следует удалить ссылающееся ограничение FOREIGN KEY или ссылающуюся таблицу.

Например, чтобы удалить таблицу D, надо выполнить команды:

DROP TABLE PD

DROP TABLE D

**Задание на лабораторную работу**

1. Создать таблицы в БД в соответствии со своей тематикой. Количество таблиц – не менее 6.
   1. Таблицы должны содержать все ограничения из методических указаний в соответствии с предметной областью.
   2. Таблицы должны быть связаны с помощью внешних ключей.
2. Наполнить таблицы данными (из своих демонстрационных примеров). Количество записей в каждой таблице – не менее 7.
3. Вывести данные из на экран с помощью запроса.
4. Реализовать не менее двух запросов на изменение структуры таблиц.
5. Реализовать 5 запросов на изменение данных в таблицах.
6. Реализовать 5 запросов на удаление записей из таблиц.

**Отчет должен содержать:**

1. Запросы на создание, изменение таблиц.
2. Запросы на манипулирование данными.
3. Запросы на вывод данных.
4. Данные, полученные в результате вывода. Вывод реализовывать после основных разделов задания (изменение, добавление, удаление данных).

**Рекомендации по выполнению**

- Задания можно выполнять в одном из онлайн-сервисов: <https://sqliteonline.com>, <https://extendsclass.com/sql-online.html>, <https://www.tutorialspoint.com/execute_sql_online.php> или аналогичном, на свой выбор.

- Также, задания можно выполнять офлайн, с помощью предпочитаемого инструмента администрирования БД.

**Справочная информация по типам полей в PostgreSQL**

**Числовые типы данных**

* serial: представляет автоинкрементирующееся числовое значение, которое занимает 4 байта и может хранить числа от 1 до 2147483647. Значение данного типа образуется путем автоинкремента значения предыдущей строки. Поэтому, как правило, данный тип используется для определения идентификаторов строки.
* smallserial: представляет автоинкрементирующееся числовое значение, которое занимает 2 байта и может хранить числа от 1 до 32767. Аналог типа serial для небольших чисел.
* bigserial: представляет автоинкрементирующееся числовое значение, которое занимает 8 байт и может хранить числа от 1 до 9223372036854775807. Аналог типа serial для больших чисел.
* smallint: хранит числа от -32768 до +32767. Занимает 2 байта. Имеет псевдоним int2.
* integer: хранит числа от -2147483648 до +2147483647. Занимает 4 байта. Имеет псевдонимы int и int4.
* bigint: хранит числа от -9223372036854775808 до +9223372036854775807. Занимает 8 байт. Имеет псевдоним int8.
* numeric: хранит числа с фиксированной точностью, которые могут иметь до 131072 знаков в целой части и до 16383 знаков после запятой. Данный тип может принимать два параметра precision и scale: numeric(precision, scale).

Параметр precision указывает на максимальное количество цифр, которые может хранить число.

Параметр scale представляет максимальное количество цифр, которые может содержать число после запятой. Это значение должно находиться в диапазоне от 0 до значения параметра precision. По умолчанию оно равно 0.

Например, для числа 23.5141 precision равно 6, а scale - 4.

* decimal: хранит числа с фиксированной точностью, которые могут иметь до 131072 знаков в целой части и до 16383 знаков в дробной части. То же самое, что и numeric.
* real: хранит числа с плавающей точкой из диапазона от 1E-37 до 1E+37. Занимает 4 байта. Имеет псевдоним float4.
* double precision: хранит числа с плавающей точкой из диапазона от 1E-307 до 1E+308. Занимает 8 байт. Имеет псевдоним float8.

**Типы для работы с валютой (денежными единицами)**

* Для работы с денежными единицами определен тип money, который может принимать значения в диапазоне от -92233720368547758.08 до +92233720368547758.07 и занимает 8 байт.

**Символьные типы**

* character(n): представляет строку из фиксированного количества символов. С помощью параметра задается задается количество символов в строке. Имеет псевдоним char(n).
* character varying(n): представляет строку из переменной длины. С помощью параметра задается задается максимальное количество символов в строке. Имеет псевдоним varchar(n).
* text: представляет текст произвольной длины.

**Бинарные данные**

* Для хранения бинарных данных определен тип bytea. Он хранит данные в виде бинарных строк, которые представляют последовательность октетов или байт.

**Типы для работы с датами и временем**

* timestamp: хранит дату и время. Занимает 8 байт. Для дат самое нижнее значение - 4713 г до н.э., самое верхнее значение - 294276 г н.э.
* timestamp with time zone: то же самое, что и timestamp, только добавляет данные о часовом поясе.
* date: представляет дату от 4713 г. до н.э. до 5874897 г н.э. Занимает 4 байта.
* time: хранит время с точностью до 1 микросекунды без указания часового пояса. Принимает значения от 00:00:00 до 24:00:00. Занимает 8 байт.
* time with time zone: хранит время с точностью до 1 микросекунды с указанием часового пояса. Принимает значения от 00:00:00+1459 до 24:00:00-1459. Занимает 12 байт.
* interval: представляет временной интервал. Занимает 16 байт.

**Распространенные форматы дат:**

* yyyy-mm-dd - 1999-01-08
* Month dd, yyyy - January 8, 1999
* mm/dd/yyyy - 1/8/1999

**Распространенные форматы времени:**

* hh:mi - 13:21
* hh:mi am/pm - 1:21 pm
* hh:mi:ss - 1:21:34

**Логический тип**

* Тип boolean может хранить одно из двух значений: true или false.

Вместо true можно указывать следующие значения: TRUE, 't', 'true', 'y', 'yes', 'on', '1'.

Вместо false можно указывать следующие значения: FALSE, 'f', 'false', 'n', 'no', 'off', '0'.

**Типы для представления интернет-адресов**

* cidr: интернет-адрес в формате IPv4 и IPv6. Например, 192.168.0.1. Занимает от 7 до 19 байт.
* inet: интернет-адрес в формате cidr/y, где cidr это адрес в формате IPv4 или IPv6, а /y - количество бит в адресе (если этот параметр не указан, то используется 34 для IPv4, 128 для IPv6). Например, 192.168.0.1/24 или 2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128. Занимает от 7 до 19 байт.
* macaddr: хранит MAC-адрес. Занимает 6 байт.
* macaddr8: хранит MAC-адрес в формате EUI-64. Занимает 8 байт.

**Геометрические типы**

* point: представляет точку на плоскости в формате (x,y). Занимает 16 байт.
* line: представляет линию неопределенной длины в формате {A,B,C}. Занимает 32 байта.
* lseg: представляет отрезок в формате ((x1,y1),(x2,y2)). Занимает 32 байта.
* box: представляет прямоугольник в формате ((x1,y1),(x2,y2)). Занимает 32 байта.
* path: представляет набор содиненных точек. В формате ((x1,y1),...) путь является закрытым (первая и последняя точка соединяются линией) и фактически представляет многоугольник. В формате [(x1,y1),...] путь является открытым Занимает 16+16n байт.
* polygon: представляет многоугольник в формате ((x1,y1),...). Занимает 40+16n байт.
* circle: представляет окружность в формате <(x,y),r>. Занимает 24 байта.

**Остальные типы данных**

* json: хранит данные json в текстовом виде.
* jsonb: хранит данные json в бинарном формате.
* uuid: хранит универсальный уникальный идентификатор (UUID), например, a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11. Занимает 32 байта.
* xml: хранит даные в формате XML.