Лабораторная работа №2

Создание объектов базы данных в СУБД

Цель работы: изучить основные возможности языка SQL для создания структуры базы данных. Научиться создавать базы данных, таблицы, связи, ограничения, а также создавать, изменять и удалять данные.

Основные теоретические сведения

Управление созданием и использованием баз данных является основной задачей *системы управления базами данных (СУБД)*. Базы данных, которыми управляет СУБД, физически располагаются на *сервере баз данных*. Для получения доступа к серверу баз данных необходимо знать имя или IP-адрес сервера, имя пользователя и пароль.

Взаимодействие с сервером баз данных для создания и модификации структуры базы данных выполняется с помощью специальных команд *языка описания данных (DDL — Data Definition Language)*. В СУБД это SQL-запросы, предназначенные для создания, изменения и удаления объектов баз данных, а также самих баз данных. Создание, изменение и удаление данных в таблицах базы данных выполняется командами *языка манипулирования данными (DML — Data Manipulation Language)*, которые также являются частью языка SQL.

Для создания и запуска SQL-запросов необходима специальная программа, предназначенная для взаимодействия с сервером баз данных под управлением конкретной СУБД. Существуют различные программы сторонних производителей, большинство из которых являются платными и содержат дополнительные возможности. Однако имеется возможность использования некоторых полнофункциональных программ бесплатно в учебных целях (DataGrip от JeTBrains).

Для создания базы данных используется следующий запрос:

```
CREATE DATABASE database_name
```

где *database_name* — имя базы данных.

Удаление созданной ранее базы данных выполняется с помощью запроса:

```
DROP DATABASE database_name
```

Изменение имени существующей базы данных:

```
ALTER DATABASE database_name
MODIFY NAME = new_database_name
```

где *new_database_name* — новое имя базы данных.

На одном сервере баз данных может быть создано несколько баз данных. При выполнении запросов к объектам базы данных требуется указать, к какой базе данных происходит обращение. Чтобы избежать этого, можно задать базу данных по умолчанию: `USE database_name`.

При выполнении следующих запросов, если при обращении к объекту данных не указано имя базы данных, будет использоваться база данных по умолчанию.

Схема базы данных — это логический контейнер, который включает в себя другие объекты базы данных. Имена объектов базы данных должны быть уникальными в рамках схемы. Схемы данных удобно использовать в больших базах данных для удобного логического разбиения всей базы данных на отдельные части, а также для разграничения доступа к объектам базы данных для различных пользователей.

С помощью следующего запроса можно создать новую схему данных:

```
CREATE SCHEMA schema_name
```

где *schema_name* — имя схемы.

Удаление созданной ранее схемы данных выполняется с помощью запроса:

```
DROP SCHEMA schema_name
```

Любая таблица базы данных содержит один или несколько столбцов, а также может содержать *ограничения целостности*. Имена столбцов должны быть уникальными для таблицы, а имена ограничений уникальны для текущей схемы базы данных.

Для создания таблицы в базе данных используется следующий запрос:

```
CREATE TABLE table_name (column[, column, ...])
```

где *table_name* — имя таблицы в полном или сокращённом виде, *column* — описание столбца.

Полный формат имени таблицы имеет вид:

```
[[database_name.]schema_name.]t_name
```

где *t_name* — имя таблицы. Если не указано имя базы данных или схемы данных, используются значения по умолчанию.

Описание столбца состоит из нескольких секций, разделённых пробелами, которые обычно располагаются в следующем порядке:

1) имя столбца, обязательно;
2) тип данных, обязательно;
3) допускает ли столбец пустые значения: если допускает — значение *NULL*, если не допускает — значение *NOT NULL*. Если значение не указано, применяется *NULL*;
4) *UNIQUE* — если указано, значения столбца являются уникальными;
5) *PRIMARY KEY* — если указано, столбец является первичным ключом. *PRIMARY KEY* включает в себя ограничения *NOT NULL* и *UNIQUE*;
6) *IDENTITY (seed, increment)* — если указано, столбец является автоинкрементным. Параметр *seed* задаёт значение, присваиваемое самой первой строке, добавляемой в таблицу, параметр *increment* — значение приращения, которое прибавляется к значению идентификатора предыдущей добавленной строки. Если значения параметров не указаны, применяются значения по умолчанию: *(1,1)*. Применяется к суррогатным ключам целочисленного типа;
7) *DEFAULT value* — если указано, параметр *value* задаёт значение по умолчанию, которое используется, если при добавлении строки не указано значение данного столбца;
8) *CHECK (condition)* — если указано, параметр *condition* задаёт условие, которое проверяется при добавлении и изменении строки. Если условие не выполняется, действие со строкой завершается ошибкой. Условие является логическим выражением, которое может содержать константные значения, имена столбцов таблицы, стандартные и пользовательские функции;
9) связь — если указано, столбец является внешним ключом по отношению к другой таблице.

В SQL определены различные основные типы данных для столбцов, например: *CHAR(n)* — строковые типы фиксированной длины, *INT(size)* — целочисленный тип. Диапазон значений от -2147483648 до 2147483647. DATE — дата, формат даты YYYY-MM-DD, TIME — время, формат HH-MI-SS и другие.

UNSIGNED ─ дополнительный параметр целочисленного типа. Обычно целое число переходит от отрицательного к положительному. Добавление атрибута UNSIGNED будет перемещать этот диапазон вверх так, чтобы он начинался с нуля вместо отрицательного числа.

Описание связи для столбца внешнего ключа имеет следующий формат:

```
[CONSTRAINT constraint_name] [FOREIGN KEY]
  REFERENCES ref_table_name [(ref_column)]
  [ON DELETE on_delete_action] [ON UPDATE on_update_action]
```

где *constraint_name* — имя связи (если не задано, генерируется автоматически), *ref_table_name* — имя главной таблицы, *ref_column* — имя столбца главной таблицы, участвующего в связи (если не задано, используется столбец первичного ключа главной таблицы), *on_delete_action* — действие, которое выполняется над строками текущей таблицы при попытке удалить соответствующую строку главной таблицы, *on_update_action* — действие, которое выполняется над значениями внешнего ключа записей текущей таблицы при попытке изменить значение первичного ключа соответствующей записи главной таблицы (NO ACTION, SET NULL и другие).

Если требуется задать ограничение *UNIQUE*, *PRIMARY KEY*, *FOREIGN KEY* для нескольких столбцов таблицы, их указывают не в описании столбца, а после описания столбцов таблицы через запятую. Имена столбцов ограничения указываются в круглых скобках через запятую.

Удаление из базы данных созданной ранее таблицы выполняется с помощью запроса:

```
DROP TABLE table_name
```

Удаление таблицы также приводит к удалению всех созданных для неё ограничений.

Изменение существующей таблицы выполняется с помощью следующего запроса:

```
ALTER TABLE table_name operation
```

где *operation* — операция модификации объекта таблицы.

Для добавления данных в таблицу базы данных используется следующий запрос:

```
INSERT [INTO] table_name
[(column_name[, column_name, ...])]
VALUES
(value[, value, ...])[, (value[, value, ...]), ...]
```

где *column_name* — имя столбца, *value* — значение соответствующего поля добавляемой записи.

Для изменения полей существующих записей таблицы базы данных используется следующий запрос:

```
UPDATE table_name
SET column_name = new_value[, column_name = new_value, ...]
[WHERE condition]
```

где *new_value* — новое значение поля, *condition* — условие выборки записей, данные в которых требуется изменить.

Для удаления существующих записей из таблицы базы данных используется следующий запрос:

```
DELETE [FROM] table_name
    [WHERE condition]
```

где *condition* — условие выборки записей, которые требуется удалить. Если условие не задано, удаляются все строки указанной таблицы.

Задание к работе

1. Составить SQL-запросы для создания структуры базы данных, полученной в результате лабораторной работы №1. Указать используемые типы данных, ограничения значений полей; для связей: действия с записями подчинённой таблицы при удалении и изменении соответствующей записи главной таблицы.
2. С помощью SQL-запросов выполнить добавление 3–4 записей в каждую таблицу, изменение и удаление нескольких записей.

Пример выполнения задания

В примере выполнения задания к лабораторной работе №1 для создания структуры базы данных была использована предметная область «Компания, занимающаяся перевозкой грузов». Составим SQL-запрос для создания таблицы «Контактные данные» базы данных.

Таблица «Контактные данные»:

```
CREATE TABLE contact_details(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    fcs VARCHAR (100) NOT NULL,
    phone VARCHAR (50) NOT NULL
    CONSTRAINT phone_check
    CHECK (substring(phone from 1 for 2) LIKE '+7'
    AND "right"(phone, 10) ~ '^[0-9]+$')
    address VARCHAR (100) NOT NULL default 0
);
```

Можно увидеть, что практически на все атрибуты наложено ограничение not null. Такие атрибуты должны по умолчанию иметь значение, отличное от NULL. Атрибут address содержит в себе ограничение default 0. Атрибуты с таким ограничением по умолчанию будут иметь значение 0, если не задать им иное значение. На поле описания у различных типов наложено ограничение по уникальности.

На атрибут phone наложено ограничение. Благодаря ему все вводимые в таблицу номера телефонов должны будут иметь 11 цифр и начинаться с «+7».

С помощью SQL-запросов выполним добавление двух записей в таблицу «Контактные данные», изменение и удаление записей.

Добавление данных в таблицу. При вставке выборки (нескольких записей) в таблицу количество столбцов в выборке должно быть равным числу столбцов в таблице.

```
INSERT INTO public.contact_details(
id, fcs, phone, address)
VALUES (1,'Ivanov Ivan Ivanovich', '12345', 'ul. Esenina, d, 21'),
    (2, 'Petrova Marina Ivanovna', '12388', 'ul. Bulochnaya, d. 11');
```

Изменение записи с условием «id = 2» в таблице. Важно правильно указывать условия в запросе, иначе возникает риск модификации записей, изменение которых не планировалось.

```
UPDATE public.contact_details
SET fcs ='Grozniy Ivan Vasilievich'
    WHERE id = 2;
```

Удаление записи с условием «id = 1» в таблице.

```
DELETE FROM public.contact_details
WHERE id = 1;
```

В процессе выполнения лабораторной работы были получены навыки использования языка SQL для создания структуры базы данных. Изучены и использованы запросы для создания базы данных, таблиц, связей, ограничений, а также создания, изменения и удаления данных в таблицах.