МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**БАЗЫ ДАННЫХ И**

**СИСТЕМЫ УПРАВЛЕНИЯ**

**БАЗАМИ ДАННЫХ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №2

Выполнил Учащийся группы ПО-455 М.Г.Верхов

Шифр 15

2023

**6 Опишите таблицы, создание, модификацию, удаление таблиц.**

**Создание таблицы**

Для создания таблиц используется команда CREATE TABLE. Эта команды применяет ряд операторов, которые определяют столбцы таблицы и их атрибуты. Общий формальный синтаксис команды CREATE TABLE:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE название\_таблицы  (название\_столбца1 тип\_данных атрибуты\_столбца1,   название\_столбца2 тип\_данных атрибуты\_столбца2,   ................................................   название\_столбцаN тип\_данных атрибуты\_столбцаN,   атрибуты\_уровня\_таблицы  ) |

После команды CREATE TABLE идет название таблицы. Имя таблицы выполняет роль ее идентификатора в базе данных, поэтому оно должно быть уникальным. Затем в скобках перечисляются названия столбцов, их типы данных и атрибуты. В самом конце можно определить атрибуты для всей таблицы. Атрибуты столбцов, а также атрибуты таблицы указывать необязательно.

Создадим простейшую таблицу. Для этого выполним следующий скрипт:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE DATABASE productsdb;    USE productsdb;    CREATE TABLE Customers  (      Id INT,      Age INT,      FirstName VARCHAR(20),      LastName VARCHAR(20)  ); |

Таблица не может создаваться сама по себе. Она всегда создается в определенной базе данных. Вначале здесь создается база данных productsdb. И затем, чтобы указать, что все дальнейшие операции, в том числе создание таблицы, будут производиться с этой базой данных, применяется команда USE.

Далее собственно идет создание таблицы, которая называется Customers. Она определяет четыре столбца: Id, Age, FirstName, LastName. Первые два столбца представляют идентификатор клиента и его возраст и имеют тип INT, то есть будут хранить числовые значения. Следующие столбцы представляют имя и фамилию клиента и имеют тип VARCHAR(20), то есть представляют строку длиной не более 20 символов. В данном случае для каждого столбца определены имя и тип данных, при этом атрибуты столбцов и таблицы в целом отсутствуют.

И в результате выполнения этой команды будет создана база данных productsdb, в которой будет создана таблица Customers.

**Переименование таблиц**

Если после создания таблицы мы захотим ее переименовать, то для этого нужно использовать команду RENAME TABLE, которая имеет следующий синтаксис:

|  |  |
| --- | --- |
| 1 | RENAME TABLE старое\_название TO новое\_название; |

Например, переименуем таблицу Customers в Clients:

|  |  |
| --- | --- |
| 1 | RENAME TABLE Customers TO Clients; |

**Полное удаление данных**

Для полного удаления данных, очистки таблицы применяется команда TRUNCATE TABLE. Например, очистим таблицу Clients:

|  |  |
| --- | --- |
| 1 | TRUNCATE TABLE Clients; |

**Удаление таблиц**

Для удаления таблицы из БД применяется команда DROP TABLE, после которой указывается название удаляемой таблицы. Например, удалим таблицу Clients:

|  |  |
| --- | --- |
| 1 | DROP TABLE Clients; |

## Изменение таблиц и столбцов

Последнее обновление: 11.05.2018

Если таблица уже была ранее создана, и ее необходимо изменить, то для этого применяется команда ALTER TABLE. Ее сокращенный формальный синтаксис:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ALTER TABLE название\_таблицы  { ADD название\_столбца тип\_данных\_столбца [атрибуты\_столбца] |    DROP COLUMN название\_столбца |    MODIFY COLUMN название\_столбца тип\_данных\_столбца [атрибуты\_столбца] |    ALTER COLUMN название\_столбца SET DEFAULT значение\_по\_умолчанию |    ADD [CONSTRAINT] определение\_ограничения |    DROP [CONSTRAINT] имя\_ограничения} |

Вообще данная команда поддерживает гораздо больше опций и возможностей. Все их можно посмотреть в документации. Рассмотрим лишь основные сценарии, с которыми мы можем столкнуться.

### Добавление нового столбца

Добавим в таблицу Customers новый столбец Address:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Customers  ADD Address VARCHAR(50) NULL; |

В данном случае столбец Address имеет тип VARCHAR и для него определен атрибут NULL.

### Удаление столбца

Удалим столбец Address из таблицы Customers:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Customers  DROP COLUMN Address; |

### Изменение значения по умолчанию

Установим в таблице Customers для столбца Age значение по умолчанию 22:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Customers  ALTER COLUMN Age SET DEFAULT 22; |

### Изменение типа столбца

Изменим в таблице Customers тип данных у столбца FirstName на CHAR(100) и установим для него атрибут NULL:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Customers  MODIFY COLUMN FirstName CHAR(100) NULL; |

### Добавление и удаление внешнего ключа

Пусть изначально в базе данных будут добавлены две таблицы, никак не связанные:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | CREATE TABLE Customers  (      Id INT PRIMARY KEY AUTO\_INCREMENT,      Age INT,      FirstName VARCHAR(20) NOT NULL,      LastName VARCHAR(20) NOT NULL  );  CREATE TABLE Orders  (      Id INT PRIMARY KEY AUTO\_INCREMENT,      CustomerId INT,      CreatedAt Date  ); |

Добавим ограничение внешнего ключа к столбцу CustomerId таблицы Orders:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Orders  ADD FOREIGN KEY(CustomerId) REFERENCES Customers(Id); |

При добавлении ограничений мы можем указать для них имя, используя оператор CONSTRAINT, после которого указывается имя ограничения:

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE Orders  ADD CONSTRAINT orders\_customers\_fk  FOREIGN KEY(CustomerId) REFERENCES Customers(Id); |

В данном случае ограничение внешнего ключа называется orders\_customers\_fk. Затем по этому имени мы можем удалить ограничение:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Orders  DROP FOREIGN KEY orders\_customers\_fk; |

### Добавление и удаление первичного ключа

Добавим в таблицу Products первичный ключ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | CREATE TABLE Products  (      Id INT,      Model VARCHAR(20)  );    ALTER TABLE Products  ADD PRIMARY KEY (Id); |

Теперь удалим первичный ключ:

|  |  |
| --- | --- |
| 1  2 | ALTER TABLE Products  DROP PRIMARY KEY; |

**54 Охарактеризуйте общие особенности трехзвенной архитектуры.**

**Трёху́ровневая архитекту́ра** (*трёхзве́нная архитекту́ра*, [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *three-tier*) — [архитектурная модель](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) [программного комплекса](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), предполагающая наличие в нём трёх типов компонентов (уровней, звеньев): [клиентских приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (с которыми работают [пользователи](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C)), [серверов приложений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (с которыми работают клиентские приложения) и [серверов баз данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (с которыми работают серверы приложений)

## Сервер приложений

|  |
| --- |
|  |
| **Рис. 7. Трехзвенная архитектура с сервером приложений** |

Клиенты (рис. 7) содержат только слой логики представления прикладного ПО, а алгоритмы бизнес-логики и логики доступа к данным перемещены в среднее звено. В этом случае сервер приложений обеспечивает «общее хранилище» бизнес-правил и процедур. Клиенты соединяются с сервером приложений и предоставляют ему данные для обработки. Совместное использование алгоритмов бизнес-логики, общих для всего приложения, обладает важными достоинствами. Помимо «утоньшения» клиента, эта стратегия ведет к созданию системы, в которой будущие обновления прикладного ПО будут производиться, главным образом, на сервере приложений, что упрощает процесс модификаций.

Обычно сервер приложений поддерживает пул ограниченного числа открытых подключений к базам данных и вместо того чтобы делать каждое подключение к базе данных выделенным для определенного клиента (как в двухзвенной архитектуре), подключения многократно используются для выполнения запросов различных клиентов.

Есть и другие преимущества. Во-первых, так как вся «важная» часть прикладной логики теперь централизована в среднем звене, нет необходимости поддерживать сложные механизмы аутентификации на стороне клиентов.

Во-вторых, аппаратная платформа, на которой выполняется сервер приложений, может быть достаточно мощной; это дает дополнительную степень масштабируемости всей прикладной системы.

В-третьих, централизованный доступ к данным в серверах приложений делает всю прикладную систему менее зависящей от конкретной СУБД.

Наконец, сервер приложений обеспечивает эффективную стратегию для интеграции. Придерживаясь того же протокола коммуникации, что и клиент, другое «внешнее» приложение может легко взаимодействовать с «чужим» сервером приложений. Эта конфигурация допускает интеграцию приложений не только на уровне данных, но и на уровне правил бизнес-логики. Это чрезвычайно важно, потому что совместное использование данных разными прикладными программами может вести к логическим противоречиям в базе данных. Типичное решение состоит в том, чтобы копировать одни и те же правила и алгоритмы в несколько прикладных программ. Но тогда очень затрудняются их поддержка и обновление — любое изменение кода должно проводиться во всех прикладных программах, которые его используют. Если же бизнес-правила сосредоточены на сервере приложений и используются совместно, то такой проблемы нет.

**Отличие «многозвенной» архитектуры от «двухзвенной».** Довольно распространена модель работы, когда клиент обращается не непосредственно к серверу БД, а к **промежуточной программе**. Эта программа обычно называется **сервером приложений**. Такую архитектуру называют **«трехзвенной»**, в отличие от **«двухзвенной»** архитектуры клиент-сервер.

**81 «Библиотека компакт-дисков»**

1) Для создания информационной системы спроектируйте базу данных в Microsoft SQL Server по № варианта (задание 61-90), содержащую не менее трех таблиц. Установите отношения между таблицами.

2) Заполните таблицы данными (не менее 10 записей).

3) На языке SQL напишите запросы:

- на вывод некоторых полей из двух таблиц;

- на вывод данных по условию, представляющему выражение:

* 1. типа сравнения;
  2. с логическим оператором «И»;
  3. с логическим оператором «ИЛИ»;

- с вычислениями над полями БД;

- параметрический.

4) Разработать форму на языке C#, содержащую все созданные объекты БД.

5) Реализовать добавление, удаление изменение данных в таблице, а также поиск и фильтрацию данных.