

# ЛЕКЦИЯ №7 БАЗЫ ДАННЫХ

Москва, 2019

# Разработка базы данных в MSAccess

## Типы данных Microsoft Access

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Текстовый        | Текст или комбинация текста и чисел, например, адреса, а также числа, не требующие вычислений, например, номера телефонов, инвентарные номера или почтовые индексы. Сохраняет до 255 знаков. Свойство "Размер поля" ( <code>FieldSize</code> ) определяет максимальное количество знаков, которые можно ввести в поле                                                                                                                         |
| Поле MEMO        | Предназначено для ввода текстовой информации, по объему превышающей 255 символов. Такое поле может содержать до 65 535 символов. Этот тип данных отличается от типа Текстовый ( <code>Text</code> ) тем, что в таблице даются не сами данные, а ссылки на блоки данных, хранящиеся отдельно. За счет этого ускоряется обработка таблиц (сортировка, поиск и т. п.). Поле типа <code>MEMO</code> не может быть ключевым или проиндексированным |
| Числовой         | Данные, используемые для математических вычислений, за исключением финансовых расчетов (для них следует использовать тип " <code>Денежный</code> "). Сохраняет 1, 2, 4 или 8 байтов. Конкретный тип числового поля определяется значением свойства Размер поля ( <code>FieldSize</code> )                                                                                                                                                     |
| Дата/время       | Значения дат и времени. Сохраняет 8 байтов                                                                                                                                                                                                                                                                                                                                                                                                    |
| Денежный         | Используется для денежных значений и для предотвращения округления во время вычислений. Сохраняет 8 байтов                                                                                                                                                                                                                                                                                                                                    |
| Счетчик          | Автоматическая вставка уникальных последовательных (увеличивающихся на 1) или случайных чисел при добавлении записи. Сохраняет 4 байта                                                                                                                                                                                                                                                                                                        |
| Логический       | Данные, принимающие только одно из двух возможных значений, таких, как " <code>Да/Нет</code> ", " <code>Истина/Ложь</code> ", " <code>Вкл/Выкл</code> ". Значения <code>Null</code> не допускаются. Сохраняет 1 бит                                                                                                                                                                                                                           |
| Поле объекта OLE | Объекты OLE <sup>1</sup> (такие, как документы Microsoft Word, электронные таблицы Microsoft Excel, рисунки, звукозапись или другие данные в двоичном формате), созданные в других программах, использующих протокол OLE. Сохраняет до 1 Гигабайта (ограничивается объемом диска)                                                                                                                                                             |

Применение определенного типа данных позволяет избежать ошибок в работе с таблицами - в поле с форматом даты невозможно ввести значение суммы, а в поле с денежным форматом невозможно ввести дату.

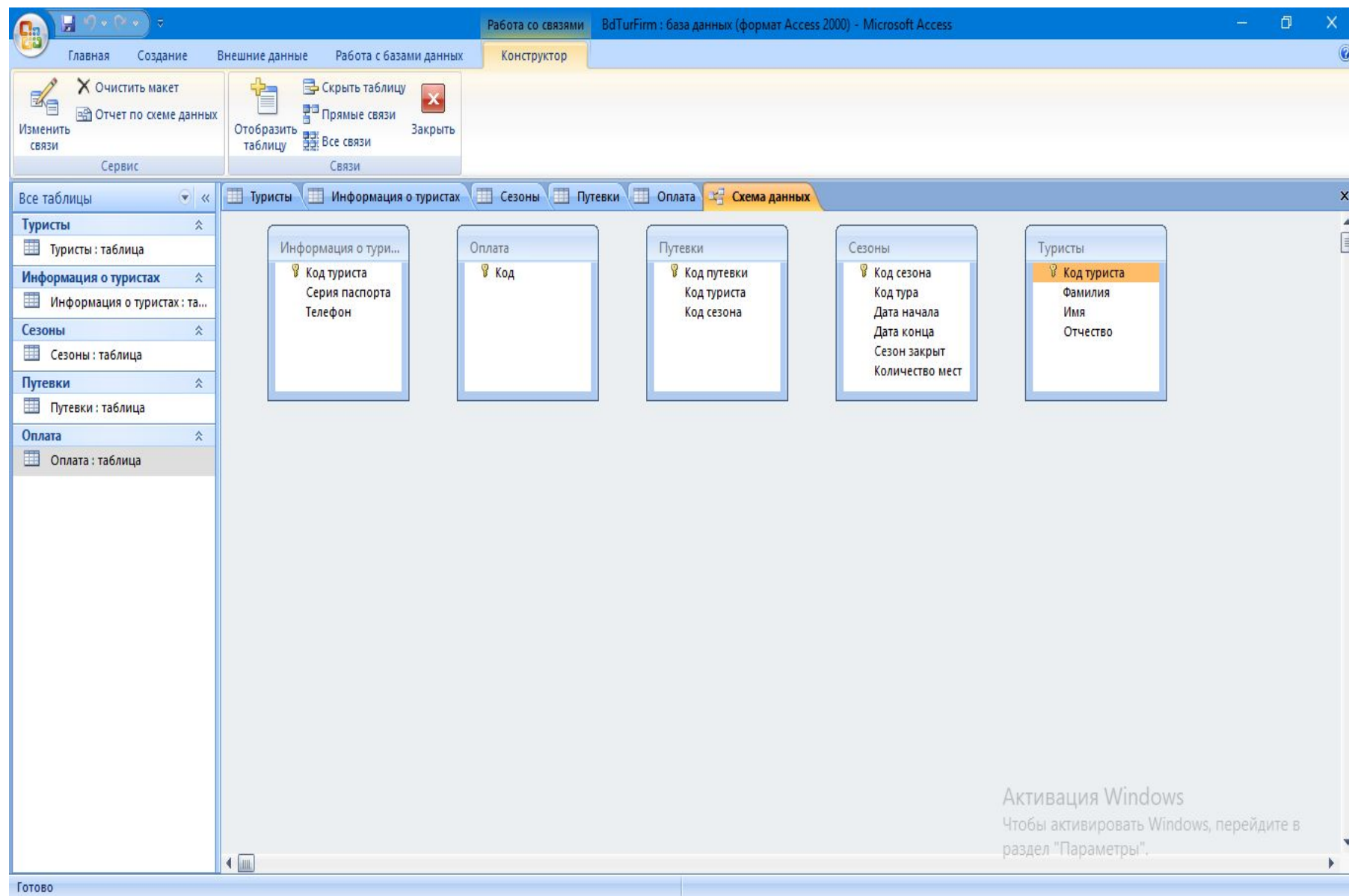
## Разработка базы данных в MSAccess управления туристической фирмой.

|   |                          |                   |              |   |                |            |                                                                                                                                   |
|---|--------------------------|-------------------|--------------|---|----------------|------------|-----------------------------------------------------------------------------------------------------------------------------------|
| 1 | Туристы                  | Код туриста       | Счетчик      |   | Имя поля       | Тип данных | Содержит основные сведения о туристе                                                                                              |
|   |                          |                   |              | 🔑 | Код туриста    | Счетчик    |                                                                                                                                   |
|   |                          | Фамилия           | Текстовый    |   | Фамилия        | Текстовый  |                                                                                                                                   |
|   |                          | Имя               | Текстовый    |   | Имя            | Текстовый  |                                                                                                                                   |
|   |                          | Отчество          | Текстовый    |   | Отчество       | Текстовый  |                                                                                                                                   |
| 2 | Информация<br>о туристах | Код туриста       | Числовой     |   | Имя поля       | Тип данных | Содержит дополнительные сведения о туристе,<br>которые были вынесены в отдельную таблицу - для<br>избегания повторяющихся записей |
|   |                          |                   |              | 🔑 | Код туриста    | Числовой   |                                                                                                                                   |
|   |                          | Серия<br>паспорта | Текстовый    |   | Серия паспорта | Текстовый  |                                                                                                                                   |
|   |                          |                   |              |   | Город          | Текстовый  |                                                                                                                                   |
|   |                          | Город             | Текстовый    |   | Страна         | Текстовый  |                                                                                                                                   |
|   |                          | Страна            | Текстовый    |   | Телефон        | Текстовый  |                                                                                                                                   |
|   |                          | Телефон           | Текстовый    |   | Индекс         | Числовой   |                                                                                                                                   |
|   |                          | Индекс            | Числовой     |   |                |            |                                                                                                                                   |
| 3 | Туры                     | Код тура          | Счетчик      |   | Имя поля       | Тип данных | Содержит общие сведения о странах для туров                                                                                       |
|   |                          |                   |              | 🔑 | Код тура       | Счетчик    |                                                                                                                                   |
|   |                          | Название          | Текстовый    |   | Название       | Текстовый  |                                                                                                                                   |
|   |                          | Цена              | Денежный     |   | Цена           | Денежный   |                                                                                                                                   |
|   |                          | Информация        | Поле<br>МЕМО |   | Информация     | Поле МЕМО  |                                                                                                                                   |

## Разработка базы данных в MSAccess

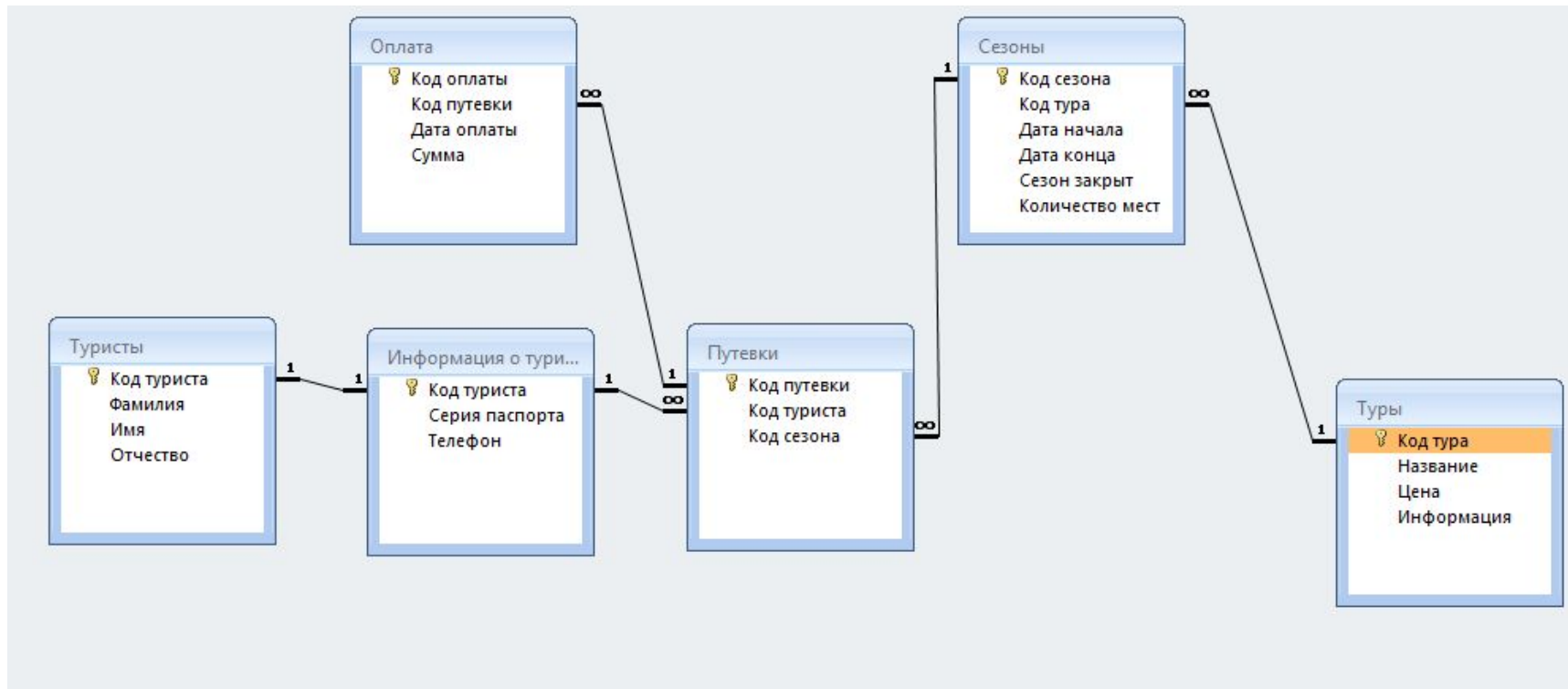
|   |         |                 |            |   |                 |            |                                                                                  |
|---|---------|-----------------|------------|---|-----------------|------------|----------------------------------------------------------------------------------|
| 4 | Сезоны  | Код сезона      | Счетчик    |   | Имя поля        | Тип данных | Содержит сведения о сезонах - некоторые туры доступны лишь в определенный период |
|   |         | Код тура        | Числовой   | ? | Код сезона      | Счетчик    |                                                                                  |
|   |         | Дата начала     | Дата/время |   | Код тура        | Числовой   |                                                                                  |
|   |         | Дата конца      | Дата/время |   | Дата начала     | Дата/время |                                                                                  |
|   |         | Сезон закрыт    | Логический |   | Дата конца      | Дата/время |                                                                                  |
|   |         | Количество мест | Числовой   |   | Сезон закрыт    | Логический |                                                                                  |
|   |         |                 |            |   | Количество мест | Числовой   |                                                                                  |
| 5 | Путевки | Код путевки     | Числовой   |   | Имя поля        | Тип данных | Содержит сведения о путевках, реализованных туристам.                            |
|   |         | Код туриста     | Числовой   | ? | Код путевки     | Числовой   |                                                                                  |
|   |         | Код сезона      | Числовой   |   | Код туриста     | Числовой   |                                                                                  |
|   |         |                 |            |   | Код сезона      | Числовой   |                                                                                  |
| 6 | Оплата  | Код оплаты      | Счетчик    |   | Имя поля        | Тип данных | Содержит сведения об оплате за путевки                                           |
|   |         | Код путевки     | Числовой   | ? | Код оплаты      | Счетчик    |                                                                                  |
|   |         | Дата оплаты     | Дата/время |   | Код путевки     | Числовой   |                                                                                  |
|   |         |                 |            |   | Дата оплаты     | Дата/время |                                                                                  |
|   |         |                 |            |   | Сумма           | Денежный   |                                                                                  |

# Разработка базы данных в MSAccess



## Разработка базы данных в MSAccess

Отношение "один-ко-многим" далее появится у нас между таблицами "Информация о туристах" и "Путевки" - один турист может приобрести несколько путевок



## Типы данных MSSQL

| Тип данных Microsoft SQL   | Описание типа данных Microsoft SQL                                                                                                                                                       |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>nvarchar</code>      | Тип данных для хранения текста до 4000 символов                                                                                                                                          |
| <code>ntext</code>         | Тип данных для хранения символов в кодировке Unicode до $2^{30} - 1$ (1 073 741 823) символов                                                                                            |
| <code>int</code>           | Численные значения (целые) в диапазоне от $-2^{31}$ (-2 147 483 648) до $2^{31} - 1$ (+2 147 483 647)                                                                                    |
| <code>smalldatetime</code> | Дата и время от 1 января 1900 г. до 6 июня 2079 года с точностью до одной минуты                                                                                                         |
| <code>money</code>         | <i>Денежный тип</i> данных значения которого лежат в диапазоне от $-2^{63}$ (-922 337 203 685 477.5808) до $2^{63} - 1$ (+922 337 203 685 477.5807), с точностью до одной десятитысячной |
| <code>int</code>           | См. 3                                                                                                                                                                                    |
| <code>bit</code>           | Переменная, способная принимать только два значения - 0 или 1                                                                                                                            |
| <code>image</code>         | Переменная для хранения массива байтов от 0 до $2^{31}-1$ (2 147 483 647) байт                                                                                                           |
| <code>ntext</code>         | См. 2                                                                                                                                                                                    |
| <code>nvarchar</code>      | См. 1                                                                                                                                                                                    |

# Агрегатные функции

## Агрегатные функции

При статистическом анализе баз данных необходимо получать такую информацию, как общее количество записей, наибольшее и наименьшее значения заданного поля записи, усредненное значение поля. Это делается с помощью запросов, содержащих так называемые агрегатные функции. Агрегатные функции производят одиночное значение для всей группы таблицы. Имеется список этих функций.

- `count` извлекает количество записей данного поля.
- `sum` извлекает арифметическую сумму всех выбранных значений данного поля.
- `avg` извлекает арифметическое среднее (усреднение) всех выбранных значений данного поля.
- `max` извлекает наибольшее из всех выбранных значений данного поля.
- `min` извлекает наименьшее из всех выбранных значений данного поля.

Для определения общего числа записей в таблице Products используем запрос

```
select count (*) from Products;
```

результатом которого будет следующее (рис. 1.42):

|   | (No column name) |
|---|------------------|
| 1 | 77               |

# Оператор сравнения like

## Оператор сравнения like

Оператор сравнения `like` нужен для поиска записей по заданному шаблону. Это одна из наиболее часто встречаемых задач - например, поиск клиента с известной фамилией в базе данных.

Предположим, что в таблице `Customers` требуется найти записи клиентов с фамилиями, начинающимися на букву "C", и содержащие поля `CustomerID`, `ContactName` и `Address`:

```
select CustomerID, ContactName, Address from Customers where ContactName like 'C%';
```

Результатом этого запроса будет таблица (рис. 1.43)

|   | CustomerID | ContactName        | Address                 |
|---|------------|--------------------|-------------------------|
| 1 | BERGS      | Christina Berglund | Berguvsv?gen 8          |
| 2 | FRANR      | Carine Schmitt     | 54, rue Royale          |
| 3 | HILAA      | Carlos Hern?ndez   | Carrera 22 con Ave. ... |
| 4 | LILAS      | Carlos Gonz?lez    | Carrera 52 con Ave. ... |
| 5 | MAISD      | Catherine Dewey    | Rue Joseph-Bens 532     |

# Data management language

## Команды изменения языка DML

Значения могут быть помещены и удалены из полей тремя командами языка DML (Язык Манипулирования Данными):

- `insert` (вставить),
- `update` (изменить),
- `delete` (удалить).

Команда `insert` имеет свои особенности:

- При указании значений конкретных полей вместо использования каких-либо значений можно применить ключевое слово `DEFAULT`
- Вставка пустой строки приводит к добавлению пробела ' ', а не значения `NULL`
- Строки и даты задаются в апострофах.
- Не задавайте данные для столбца, имеющего свойство `IDENTITY`
- Можно задать `NULL` явно, можно задать `DEFAULT` .

Примеры:

```
insert into ClientInfo
(FirstName, LastName, Address, Phone)
values('Petr','Petrov','Chehova 13','1234567');
```

```
update ClientInfo set FirstName = 'Andrey' where FirstName = 'Petr';
```

В этом случае в первой записи поля `FirstName` значение `Petr` изменится на `Andrey`

```
delete from ClientInfo where LastName like 'Petrov';
```

## Подключение к внешней базе MSSQL

Добавить подключение

Введите данные для подключения к выбранному источнику данных или нажмите кнопку "Изменить", чтобы выбрать другой источник данных и (или) поставщик.

Источник данных:  
Microsoft SQL Server (SqlClient) Изменить...

Имя сервера:  
31.31.196.234 Обновить

Вход на сервер

☐ Использовать проверку подлинности Windows

☒ Использовать проверку подлинности SQL Server

Имя пользователя: u0831361\_base

Пароль: ••••••••••••••••

☐ Сохранить пароль

Подключение к базе данных

☒ Выберите или введите имя базы данных:  
u0831361\_base

☐ Прикрепить файл базы данных:  
Обзор...

Логическое имя:

Дополнительно...

Проверить подключение OK Отмена

## ADO NET

### ADO NET

```
graph TD; A[ADO NET] --> B[Требующие подключение: Connection, Transaction, DataAdapter, Command, Parameter, DataReader]; A --> C[Не требующие подключение: DataSet, DataTable, DataRow, DataColumn, Constraint, DataView];
```

Требующие подключение:  
Connection,  
Transaction, DataAdapter, Command,  
Parameter, DataReader

Не требующие подключение:  
DataSet, DataTable, DataRow,  
DataColumn, Constraint, DataView

`DataSet` состоит из объектов типа `DataTable` и объектов `DataRelation`. В коде к ним можно обращаться как к свойствам объекта `DataSet`, т. е. используя *точечную нотацию*. Свойство `Tables` возвращает объект типа `DataTableCollection`, который содержит все объекты `DataTable` используемой базы данных.

### Таблицы и поля (объекты `DataTable` и `DataColumn`)

Объекты `DataTable` используются для представления таблиц в `DataSet`. `DataTable` представляет одну таблицу из базы данных. В свою очередь, `DataTable` составляется из объектов `DataColumn`.

`DataColumn` — это блок для создания схемы `DataTable`. Каждый объект `DataColumn` имеет свойство `DataType`, которое определяет тип данных, содержащихся в каждом объекте `DataColumn`. Например, вы можете ограничить тип данных до целых, строковых и десятичных чисел. Поскольку данные, содержащиеся в `DataTable`, обычно переносятся обратно в исходный источник данных, вы должны согласовывать тип данных с источником.

### Объекты `DataRelation`

Объект `DataSet` имеет также свойство `Relations`, возвращающее коллекцию `DataRelationCollection`, которая, в свою очередь, состоит из объектов `DataRelation`. Каждый объект `DataRelation` выражает отношение между двумя таблицами (сами таблицы связаны по какому-либо полю (столбцу); следовательно, эта связь осуществляется через объект `DataColumn`).

# ADO NET

## Строки (объект DataRow)

Коллекция `Rows` объекта `DataTable` возвращает набор строк (записей) заданной таблицы. Эта коллекция используется для изучения результатов запроса к базе данных. Мы можем обращаться к записям таблицы как к элементам простого массива.

## DataAdapter

`DataSet` — это специализированный *объект*, содержащий образ *базы данных*. Для осуществления взаимодействия между `DataSet` и источником данных используется *объект* типа `DataAdapter`. Само название этого объекта — *адаптер*, преобразователь, — указывает на его природу. `DataAdapter` содержит метод `Fill()` для обновления данных из базы и заполнения `DataSet`.

## Объекты DBConnection и DBCommand

*Объект* `DBConnection` осуществляет *связь* с источником данных. Эта *связь* может быть одновременно использована несколькими командными объектами. *Объект* `DBCommand` позволяет послать базе данных команду (как правило, команду *SQL* или хранимую процедуру). Объекты `DBConnection` и `DBCommand` иногда создаются неявно в момент создания объекта `DataSet`, но их также можно создавать явным образом.

## Использование визуальной среды для работы с ADO.NET

Когда мы перетаскиваем на форму элемент управления, *Visual/Studio .NET* автоматически генерирует код, описывающий этот элемент. Если при размещении кнопок, текстовых полей и других элементов управления создавать их программно нецелесообразно, то при работе с элементами данных все как раз наоборот — лучше всего создавать все объекты *ADO.NET* вручную, что обеспечивает большую гибкость приложения. Тем не менее на первых порах использования *ADO.NET* полезно работать с визуальной средой разработки.

Рассмотрим работу с базой данных Microsoft Access xtreme<sup>3</sup>, входящей в состав Microsoft *Visual/Studio .NET*, и работу с базой Microsoft SQL<sup>4</sup> NorthwindCS. В каждой базе есть *таблица* Customer(s)<sup>5</sup>. Наша задача — вывести содержимое двух таблиц *Customer* на две *Windows-* формы.

## Проверка подключения

Поставщики данных:  
OLE DB и SQL Client

Объект connection класса OleDbConnection:

```
OleDbConnection connection = new OleDbConnection();
```

State — свойство, определяющее состояние подключения.

Теперь посмотрим на методы, которые есть у класса:

BeginDbTransaction() — запустить транзакцию базы данных (метод унаследован от класса DbConnection);

BeginTransaction() — перегруженный метод начала транзакции базы данных;

ChangeDatabase() — установить новую базу данных на сервере, к которому мы сейчас подключены;

Close() — закрыть соединение

CreateCommand() — создать объект OleDbCommand

Хорошим тоном является явное закрытие соединения с базой данных

## Проверка подключения

В реальном приложении соединение с базой данных чаще всего делают в кон-  
структоре класса, и если соединение не прошло успешно, то программу  
просто за-  
крывают. Нет смысла запускать программу, работающую с базой данных, без  
воз-  
можности создать реальное соединение с сервером.

### Пул соединений

Открытие и закрытие соединений с базой данных — очень дорогое  
удовольст-  
вие с точки зрения производительности. В момент инициализации  
подключения  
клиенту требуется выполнить достаточно много действий, скрытых от  
конечного  
пользователя.

Зачем постоянно держать активное соединение. Есть тайм-аут на процесс  
установки соединения, а если соединение уже установилось, то оно  
может быть активным неделями. Поэтому некоторые пользователи не  
закрывают программы.  
Когда вы уничтожаете объекты класса Connection, то поставщик данных  
ре-

ально не закрывает соединения с базой данных. Объект сохраняется как

## ExecuteScalar

```
OleDbCommand command = connection.CreateCommand();  
command.CommandText = "SELECT COUNT(*) FROM Peoples";  
int number = (int)command.ExecuteScalar();  
MessageBox.Show(number.ToString());
```

Для выполнения SQL-команды в данном примере используется метод `ExecuteScalar()`. Он подходит для тех случаев, когда запрос возвращает только одно значение. Наш запрос возвращает количество записей в таблице

## Транзакции

Begin() — начать транзакцию;

Commit() — сохранить изменения, сделанные внутри транзакции;

Rollback() — отменить изменения, т. е. откатить транзакцию

Для выполнения SQL-команды в данном примере используется метод ExecuteScalar(). Он подходит для тех случаев, когда запрос возвращает только одно значение. Наш запрос возвращает количество записей в таблице

## Наборы данных

Для выполнения запросов, возвращающих наборы данных, используется уже знакомый нам класс `OleDbCommand`. Его метод `ExecuteReader()` выполняет запрос и возвращает объект класса `OleDbDataReader`, через который как раз и можно просмотреть весь набор данных результата.

Чтобы получить очередную строку данных результата, нужно вызвать метод `Read()` класса `OleDbDataReader`. Этот метод возвращает булево значение, которое определяет, прочиталась ли очередная строка. Если мы достигли конца набора данных, то результатом вызова метода будет `false`.

метод `GetValues()`

`GetBoolean()` — возвращает значение поля в виде булева значения;

`GetByte()` — возвращает значение поля в виде байта;

`GetChar()` — возвращает значение поля в виде символа `char`;

`GetDecimal()` — возвращает значение поля в виде числа `Decimal`;

`GetDouble()` — возвращает значение поля в виде числа с плавающей точкой;

`GetString()` — возвращает значение поля в виде строки.

## Запросы с параметрами

```
OleDbCommand command = connection.CreateCommand();  
command.CommandText = "SELECT * FROM Peoples WHERE Фамилия LIKE ?";  
command.Parameters.Add("@lastname", OleDbType.WChar, 50);  
command.Parameters[0].Value = "Смирнов";  
OleDbDataReader reader = command.ExecuteReader();
```

```
command.CommandText =  
"SELECT * FROM Peoples WHERE Фамилия LIKE @lastname";  
command.Parameters.Add("@lastname", OleDbType.WChar, 50);
```

## Отсоединенные данные

Хорошо было бы получить данные, закрыть соединение и спокойно обрабатывать результат. Это особенно важно при формировании отчетов, когда пользователь может просматривать их долгое время. Это также удобно, когда пользователь редактирует набор данных продолжительное время, а потом одной командой заливает на сервер сделанные изменения.

Класс `OleDbDataAdapter` — это класс адаптера, который реализует все необходимые методы для кэширования данных. Но он всего лишь провайдер, который

реализует действия по кэшированию. Хранилищем для кэша может выступать один из классов `DataTable` или `DataSet`.

Первый из них реализует кэш одной таблицы, а второй является набором данных и может состоять из множества таблиц.

Для этого

У класса `DataSet` есть свойство `Tables`, которое является коллекцией из таблиц `DataTable`. Если ваш результат возвращает несколько наборов данных, то логичнее

было бы использовать для хранения данных класс `DataSet`. Мы будем использовать

именно его, как более универсальный

## Отсоединенные данные

Вернемся к классу OleDbDataAdapter. Два основных его метода — Fill() и Update(). Первый метод позволяет скопировать все данные из результата запроса в DataSet, а второй позволяет залить изменения обратно на сервер. При этом класс OleDbDataAdapter сам выполняет указанную команду, поэтому ее не нужно даже самостоятельно открывать.

```
OleDbCommand command = new OleDbCommand("SELECT * FROM Peoples");  
command.Connection = connection;  
OleDbDataAdapter adapter = new OleDbDataAdapter(command);  
DataSet dataset = new DataSet();  
adapter.Fill(dataset);  
connection.Close();
```

После этого соединение с сервером можно закрывать, оно больше не нужно.

Все данные находятся локально в наборе данных DataSet, и вы можете работать с ними.

## Отсоединенные данные

```
private BindingSource bindingSource = new BindingSource();
```

Класс BindingSource представляет собой еще одного посредника, но на  
ЭТОТ

раз между набором данных DataSet и визуальными компонентами. В нашем  
случае визуальным компонентом будет сетка DataGridView, и чтобы сетка  
отобразила данные, нам как раз и нужен посредник в виде BindingSource.

```
AutoGenerateColumns = true
```

в сетке колонки будут сгенерированы автоматически