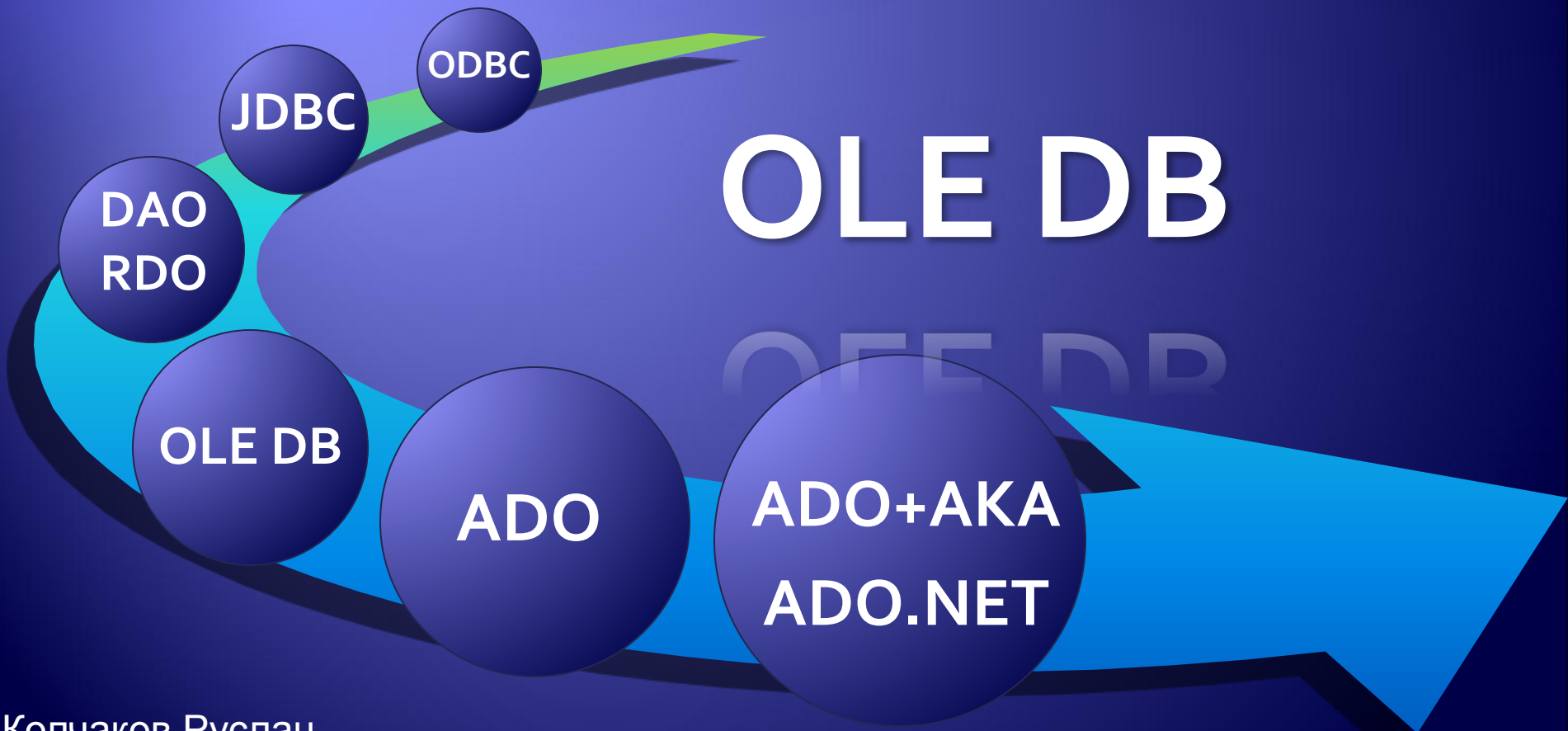
# Технологии доступа к базам данных




Колчаков Руслан

# Недостатки ODBC


- 1. Стандарт ODBC предназначен для реляционных баз данных, но в электронном мире существует достаточно много не реляционной информации, такой как сообщения электронной почты, документы, Web-страницы,  картинки, аудиофайлы и т.д. Эти данные не могут хранится в формате реляционных СУБД.

# СУБД

- Система управления базами данных (СУБД) – комплекс программных средств для создания баз данных, хранения и поиска в них необходимой информации.


# Реляционная модель

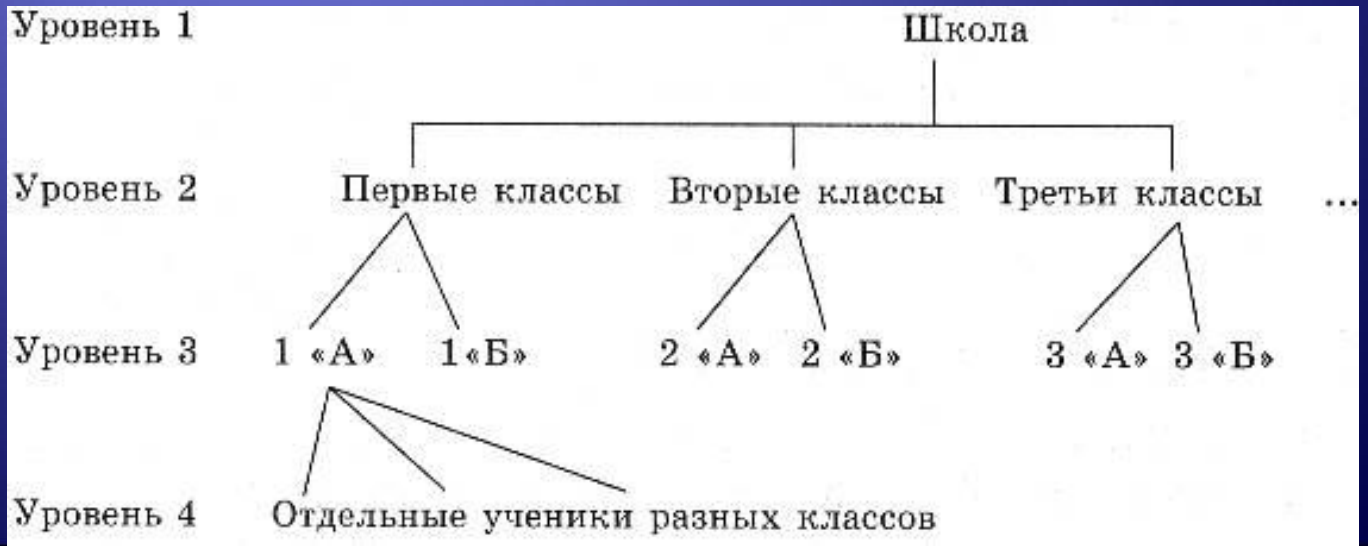
- Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных
- все ячейки в столбце таблицы однородные, то есть все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.)
- каждый столбец имеет уникальное имя
- одинаковые строки в таблице отсутствуют
- порядок следования строк и столбцов может быть произвольным

# Иерархическая модель

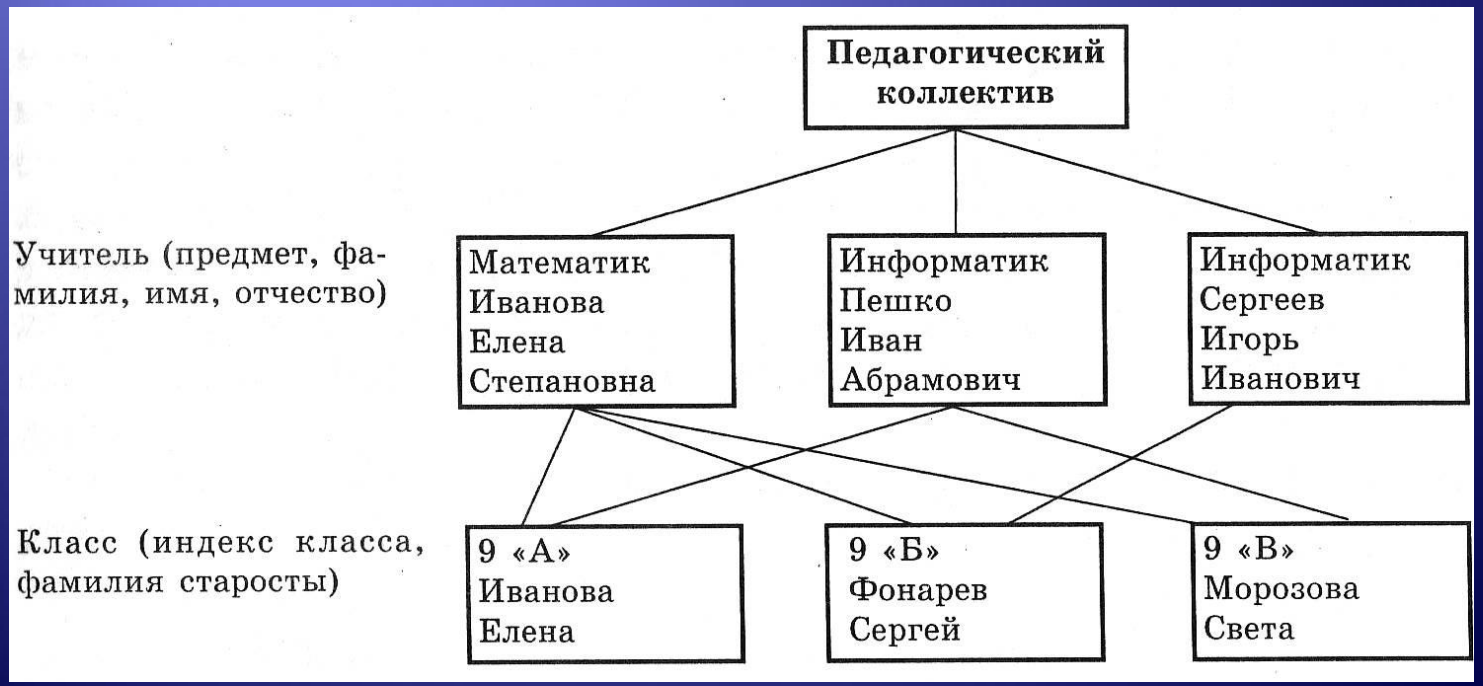
- Иерархическая модель базы данных состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию.

Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй — объекты второго уровня и т. д.



# Сетевая модель

- Сетевая модель аналогична иерархической, за исключением того, что в сетевой модели принята свободная связь между элементами разных уровней.



# Недостатки ODBC

- 2. Для выполнения самых простых задач программисту приходится вызывать десятки сложных функций, т.к. доступ к драйверам ODBC осуществлялся посредством API. Тоесть

для доступа к данным при помощи ODBC любая программа вызывает API-функции, причем в определённой последовательности:

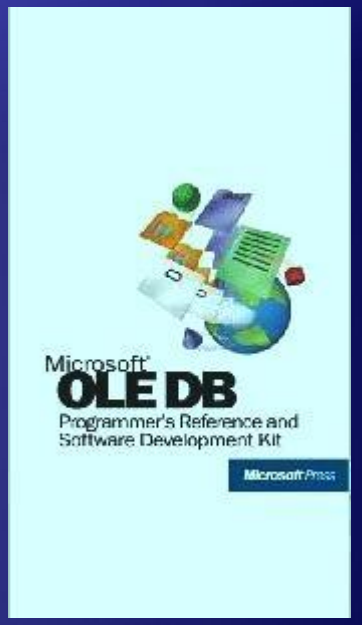
- подключение к источнику данных;
- инициализация и настройка параметров SQL-запроса/оператора;
- формирование и выполнение запроса/оператора;
- получение результатов;
- отключение от источника данных.

# API

- Интерфейс прикладного программирования (Application Programming Interface, **API**) — набор готовых констант, структур и функций, используемых при программировании пользовательских приложений и обеспечивающих правильное взаимодействие между пользовательским приложением и операционной системой.

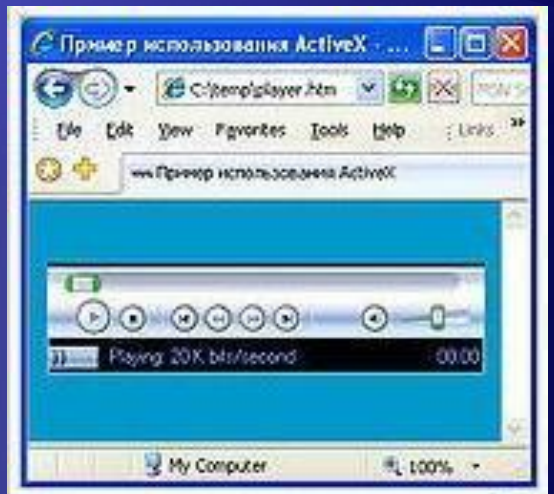
# Введение OLE DB

- Технология OLE DB пришла на смену ODBC, с тем, чтобы устранить её недостатки.
- В начале 1990-х корпорация Microsoft представила OLE DB – объектно-ориентированный интерфейс для доступа к базам данных .



# OLE

- OLE (Object Linking and Embedding) — технология связывания и внедрения объектов в другие документы и объекты, разработанные корпорацией Microsoft.
- OLE используется при обработке составных документов (compound documents), может быть использована при передаче данных между различными несвязанными между собой системами посредством интерфейса переноса (drag-and-drop), а также при выполнении операций с буфером обмена.
- OLE-объекты являются COM-объектами и поддерживают все требуемые для таких объектов интерфейсы.




Использование ActiveX — последнего продукта в эволюции OLE. Проигрывание музыкального файла.

# COM

- Стандарт **COM** был разработан в 1993 году корпорацией Microsoft как основа для развития технологии OLE.
- Объектная Модель Компонентов (Component Object Model, **COM**) — это технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих распределённых компонентов, каждый из которых может использоваться во многих программах одновременно.
- Стандарт **COM** воплощает в себе идеи полиморфизма(1) и инкапсуляции(2) объектно-ориентированного программирования.


1-Полиморфизм — взаимозаменяемость объектов с одинаковым интерфейсом.
2-Инкапсуляция — свойство языка программирования, позволяющее объединить данные и код в объект и скрыть реализацию объекта от пользователя.

# Цели создания OLE DB

- 1. Создание объектных интерфейсов для элементов функциональности СУБД – запрос, обновление, управление транзакциями.
- 2. Увеличение гибкости:

  - дать потребителям данных возможность использовать только те объекты, которые им нужны.

  - дать поставщикам данных возможность открывать доступ к элементам функциональности СУБД.

  - обеспечить возможность доступа к функциональности с помощью множества различных интерфейсов.

  - сделать эти интерфейсы стандартизированными и расширяемыми.

- 3. Создание объектных интерфейсов для любых типов данных: реляционных баз данных (через ODBC или собственные интерфейсы СУБД), нереляционных баз данных, систем обработки файлов (VSAM и др.),  электронной почты и т.п.
- 4. Реализация стратегии, при которой данные не должны преобразовываться в другие форматы или перемещаться из того места, где они находятся.

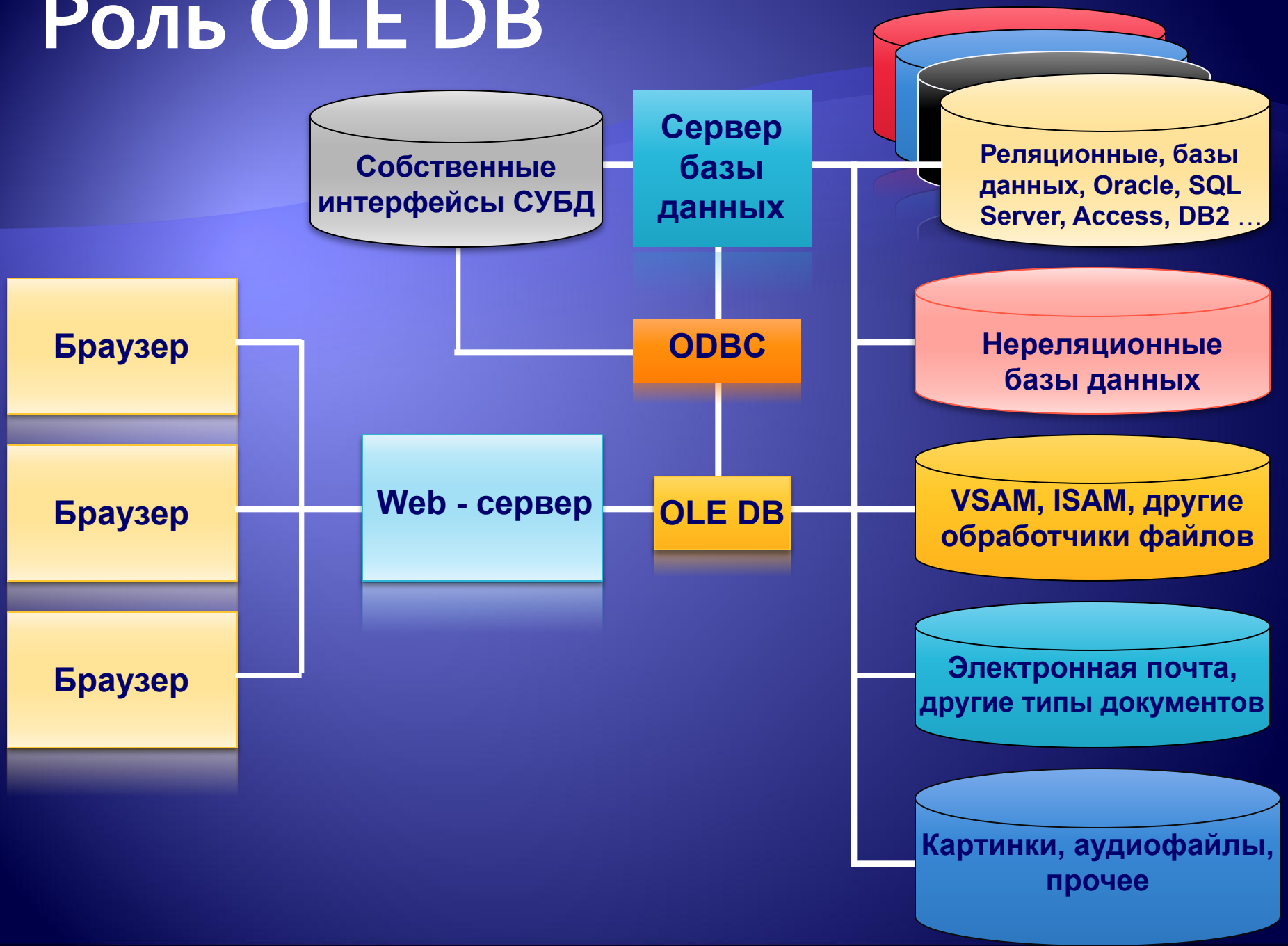
# Основные понятия OLE DB


- OLE DB представляет собой набор COM-интерфейсов, которые предоставляют приложению-клиенту унифицированный
- доступ к различным источникам данных.
- А именно, OLE DB разбивает всю совокупность возможностей и функций СУБД на отдельные фрагменты – COM-объекты с узкой специализацией. Одни объекты выполняют запросы, другие производят обновление, третьи создают структурные элементы базы данных – таблицы, индексы и представления, четвёртые занимаются управлением транзакциями – например, устанавливают оптимистическую блокировку.

- OLE DB – это метод доступа к любым данным, вне зависимости от типа данных и места их расположения. В качестве данных могут выступать  реляционные и не реляционные базы данных, картинки, аудиофайлы, электронная почта и другие источники данных.
- Средства, предоставляющие доступ к источнику данных с использованием технологии OLE DB, называются OLE DB провайдерами.
- Программы-клиенты, использующие для доступа OLE DB провайдеры, называются потребителями данных.

- В случае, если существует только ODBC-драйвер для доступа к конкретному источнику данных, то для применения технологии OLE DB можно использовать OLE DB провайдер, предназначенный для доступа к ODBC-источнику данных.

# Роль OLE DB


Собственные интерфейсы СУБД
Сервер базы данных
Реляционные, базы данных, Oracle, SQL Server, Access, DB2 …
Браузер
Браузер
Браузер
ODBC
Нереляционные базы данных
Web - сервер
OLE DB
VSAM, ISAM, другие обработчики файлов
Электронная почта, другие типы документов
Картинки, аудиофайлы, прочее

# Объектная модель OLE DB

- Спецификация OLE DB определяет набор интерфейсов базового уровня, которые должны реализовываться любыми OLE DB провайдерами.
- В базовую модель OLE DB входят следующие объекты:

  - объект DataSource (источник данных), используемый для соединения с источником данных и создания одного или нескольких сеансов. Этот объект управляет соединением, использует информацию о полномочиях и аутентификациях(1) пользователя.

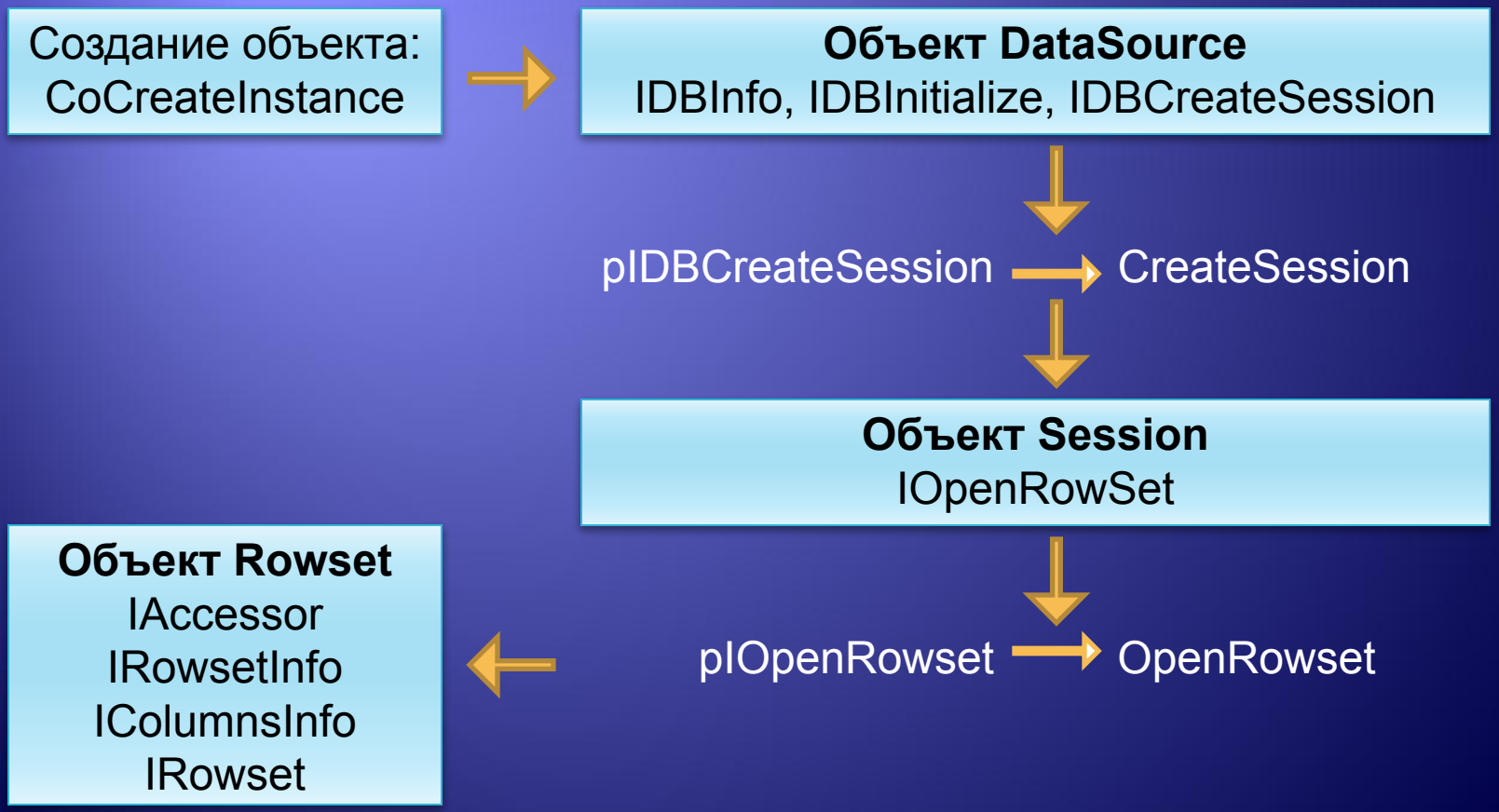
1-**Аутентификация** (Authentication) — процедура проверки соответствия некоего лица и его учетной записи в компьютерной системе. В простейшем случае проверка происходит с помощью пароля.

- объект Session (сеанс) управляет взаимодействием с источником данных – выполняет запросы и создаёт результирующие наборы. Сеанс также может возвращать метаданные(1). В сеансе может создаваться одна или несколько команд.

1-Метаданные:

- информация о данных.
- информация об информации.
- структурированные данные, представляющие собой характеристики описываемых сущностей для целей их идентификации, поиска, оценки, управления ими.
- данные из более общей формальной системы, описывающей заданную систему данных.

- объект Rowset (результирующий набор) представляет собой данные, извлекаемые в результате выполнения команды или создаваемые в сеансе.



- Спецификация OLE DB определяет объект Command (команда), предназначенный для выполнения текстовой команды. В качестве такой команды может выступать и SQL-оператор. При этом выполнение команды может создавать результирующий набор (в случае SQL-оператора – это оператор SELECT).
- Представление (View) определяет подмножество строк и столбцов из набора данных, но само не содержит их. Представления не могут объединять данные из нескольких наборов таблиц.

- OLE DB провайдеры, как и все COM-компоненты, регистрируются в реестре Windows. Для поиска информации о зарегистрированных источниках данных используются специальные объекты, называемые нумераторами. Нумератор - это обычный COM-сервер, позволяющий получить информацию об источниках данных в виде результирующего набора. Для создания такого результирующего набора в объектном типе DataSource специфицирован интерфейс IDBEnumerateSources.
- Для каждого объектного типа спецификация OLE DB определяет набор интерфейсов, который должен обязательно быть реализован для данного объекта. Такие интерфейсы отмечаются как [mandatory]. Интерфейсы, которые могут отсутствовать, отмечаются как [optional].

- Для объекта "источник данных" специфицирован следующий набор интерфейсов:

```
CoType TDataSource {
    [mandatory] interface IDBCreateSession;
    [mandatory] interface IDBInitialize;
    [mandatory] interface IDBProperties;
    [mandatory] interface IPersist;
    [optional] interface IConnectionPointContainer;
    [optional] interface IDBAsynchStatus;
    [optional] interface IDBDataSourceAdmin;
    [optional] interface IDBInfo;
    [optional] interface IPersistFile;
    [optional] interface ISupportErrorInfo;
}
```

- Для объекта "сеанс" специфицирован следующий набор интерфейсов:

```
CoType TSession {
    [mandatory] interface IGetDataSource;
    [mandatory] interface IOpenRowset; // Создание набора данных
    [mandatory] interface ISessionProperties;
    [optional] interface IAlterIndex;
    [optional] interface IAlterTable;
    [optional] interface IBindResource;
    [optional] interface ICreateRow;
    [optional] interface IDBCreateCommand;
    [optional] interface IDBSchemaRowset;
    [optional] interface IIndexDefinition;
    [optional] interface ISupportErrorInfo;
    [optional] interface ITableCreation;
    [optional] interface ITableDefinition; // Для создания таблицы
    [optional] interface ITableDefinitionWithConstraints;
    [optional] interface ITransaction;
    [optional] interface ITransactionJoin;
    [optional] interface ITransactionLocal;
    [optional] interface ITransactionObject;
}
```

- Для объекта "результирующий набор" специфицирован следующий набор интерфейсов:

```
CoType TRowset {
    [mandatory] interface IAccessor;
     [mandatory] interface IColumnsInfo;
     [mandatory] interface IConvertType;
     [mandatory] interface IRowset;// Последовательное
     // чтение таблицы
     [mandatory] interface IRowsetInfo;
     [optional] interface IChapteredRowset;
     [optional] interface IColumnsInfo2;
     [optional] interface IColumnsRowset;
     [optional] interface IConnectionPointContainer;
     [optional] interface IDBAsynchStatus;
     [optional] interface IGetRow;
     [optional] interface IRowsetChange; // Для удаления, изменения и добавления
     // строк в набор данных
     [optional] interface IRowsetChapterMember;
     [optional] interface IRowsetCurrentIndex;
     [optional] interface IRowsetFind;
     [optional] interface IRowsetIdentity;
     [optional] interface IRowsetIndex;
     [optional] interface IRowsetLocate; // Прямое
     // позиционирование на запись набора данных
     [optional] interface IRowsetRefresh; // Для
     // обновления данных в созданном наборе данных
     [optional] interface IRowsetScroll; // Поддержка
     // скроллинга по набору данных
     [optional] interface IRowsetUpdate;
     [optional] interface IRowsetView;
     [optional] interface ISupportErrorInfo;
     [optional] interface IRowsetBookmark;
}
```

- Все объекты объектного типа Rowset должны реализовывать следующие интерфейсы:

  1-интерфейс IRowset, используемый для извлечения строк;

  2-интерфейс IAccessor, используемый для определения связывания;

  3-интерфейс IColumnsInfo, предоставляющий информацию о столбцах результирующего набора;

  4-интерфейс IRowsetInfo, предоставляющий информацию о самом результирующем наборе;

  5-интерфейс IConvertType, предоставляющий информацию о преобразовании типов данных, поддерживаемых в результирующем наборе.

# Создание результирующего набора

- При реализации доступа к БД посредством OLE DB провайдера сначала следует создать объект данных и установить соединение с базой данных. Далее необходимо создать объект "сеанс". И только потом можно создавать результирующий набор.
- Результирующий набор может быть создан одним из следующих способов:

  1-для объекта "сеанс" вызывается метод IOpenRowset::OpenRowset, выполняющий непосредственное создание результирующего набора (интерфейс IOpenRowset должен поддерживаться любым провайдером);

- 2-для объекта "сеанс" вызывается метод IDBCreateCommand::CreateCommand, создающий объект Command. Далее для объекта "команда" вызывается метод ICommand::Execute. (при использовании интерфейса IMultipleResults можно работать с несколькими результирующими наборами);

  3-вызывается один из следующих методов IColumnsRowset::GetColumnsRowset, IDBSchemaRowset::GetRowset, IViewRowset::OpenViewRowset или ISourcesRowset::GetSourcesRowset.

  Чтобы результирующий набор, хранимый на сервере, можно было использовать, необходимо выполнить связывание и извлечение данных. Для этого следует определить структуры типа DBBINDING, описывающие столбцы, и создать аксессор. Далее для получения строк результирующего набора можно использовать один из следующих методов:

  1-IRowset::GetNextRows;
  2-IRowsetLocate::GetRowsByBookMarks;
  3-IRowsetLocate::GetRowAt;
  4-IRowsetScroll:: GetRowAtRatio.

  В заключение для записи данных в структуру, определенную аксессором, вызывается метод IRowset::GetData.

  После получения и обработки строк их следует освободить, вызвав метод IRowset::ReleaseRows.

- После просмотра всего результирующего набора следует также освободить аксессор, вызвав метод IRowset::ReleaseAccessor, и освободить сам результирующий набор, вызвав метод IRowset::Release.

Интерфейс IAccessor определяет следующие методы:
1-AddRefAccessor - увеличивает число ссылок на данный аксессор;
2-CreateAccessor - создает аксессор из набора связываний;
3-GetBindings - возвращает связывания, установленные данным аксессором;
4-ReleaseAccessor - освобождает аксессор.

Для создания аксессора следует запросить интерфейс IAccessor и выполнить следующий код:

```
HRESULT hr=pIAccessor-> CreateAccessor();
```

Метод CreateAccessor имеет следующее формальное описание:

```
HRESULT CreateAccessor (
DBACCESSORFLAGS dwAccessorFlags, // Свойства
// аксессора и как он используется
DBCOUNTITEM cBindings, // Число связей
// в аксессоре
const DBBINDING rgBindings[], // Описание
// столбца или параметра
DBLENGTH cbRowSize, // Число байтов,
// используемых для одного набора параметров
HACCESSOR *phAccessor, // Указатель
//на созданный аксессор
DBBINDSTATUS rgStatus[]); // Массив значений,
// определяющий статус
// каждого связывания
```

- Каждый столбец формируемого результирующего набора или параметр описывается структурой DBBINDING, которая имеет следующее формальное описание:

```
typedef struct tagDBBINDING {
DBORDINAL iOrdinal; // Порядковый номер
// столбца или параметра (начиная с 1)
DBBYTEOFFSET obValue; // Сдвиг в байтах для
// значения столбца или параметра в буфере
// (указатель на буфер задается при
// создании аксессора)
DBBYTEOFFSET obLength;
DBBYTEOFFSET obStatus;
ITypeInfo *pTypeInfo;
DBOBJECT *pObject;
DBBINDEXT *pBindExt;
DBPART dwPart;
DBMEMOWNER dwMemOwner;
DBPARAMIO eParamIO;
DBLENGTH cbMaxLen;
DWORD dwFlags;
DBTYPE wType;
BYTE bPrecision;
BYTE bScale;
} DBBINDING;
```

- Поле wType определяет тип столбца или параметра, который описывается следующим образом:

```
typedef WORD DBTYPE;

enum DBTYPEENUM {
// Следующие значения точно соответствуют VARENUM
// при автоматизации и не могут быть использованы
// как VARIANT.
DBTYPE_EMPTY = 0, // Значение отсутствует,
// соответствующего типа C нет
DBTYPE_NULL = 1, // Значение равно NULL,
// соответствующего типа C нет
DBTYPE_I2 = 2, // Двухбайтовое целое со знаком,
// соответствует C типу short
DBTYPE_I4 = 3, // Четырехбайтовое целое со знаком,
// соответствует C типу long
DBTYPE_R4 = 4,
DBTYPE_R8 = 5, // Вещественное двойной точности,
// соответствует C типу Double
DBTYPE_CY = 6, // Тип для значения Cyrrency
DBTYPE_DATE = 7, // Тип для значения даты
// (дата хранится в виде вещественного числа:
// целочисленная часть определяет дату,
// а дробная - время)
DBTYPE_BSTR = 8, // Указатель на строку BSTR
DBTYPE_IDISPATCH = 9, // Указатель на интерфейс
// IDispatch
DBTYPE_ERROR = 10, // 32-битовый код ошибки
DBTYPE_BOOL = 11, // Для логического значения
DBTYPE_VARIANT = 12, // Для значения VARIANT
DBTYPE_IUNKNOWN = 13, // Указатель на интерфейс
// IUnknown
```

- DBTYPE_DECIMAL = 14,
  DBTYPE_UI1 = 17, // Однобайтовое беззнаковое целое,
  // соответствует C типу byte
  DBTYPE_ARRAY = 0x2000,
  DBTYPE_BYREF = 0x4000,
  DBTYPE_I1 = 16,
  DBTYPE_UI2 = 18,
  DBTYPE_UI4 = 19,
  // Следующие значения точно соответствуют VARENUM
  // при автоматизации, но не могут быть использованы
  // как VARIANT.
  DBTYPE_I8 = 20,
  DBTYPE_UI8 = 21,
  DBTYPE_GUID = 72, // Для уникального идентификатора GUID
  DBTYPE_VECTOR = 0x1000,
  DBTYPE_FILETIME = 64,
  DBTYPE_RESERVED = 0x8000,
  // Следующие значения недопустимы в VARENUM для OLE.
  DBTYPE_BYTES = 128,
  DBTYPE_STR = 129,
  DBTYPE_WSTR = 130,
  DBTYPE_NUMERIC = 131,
  DBTYPE_UDT = 132,
  DBTYPE_DBDATE = 133,// Для даты, определяемой
  // как структура

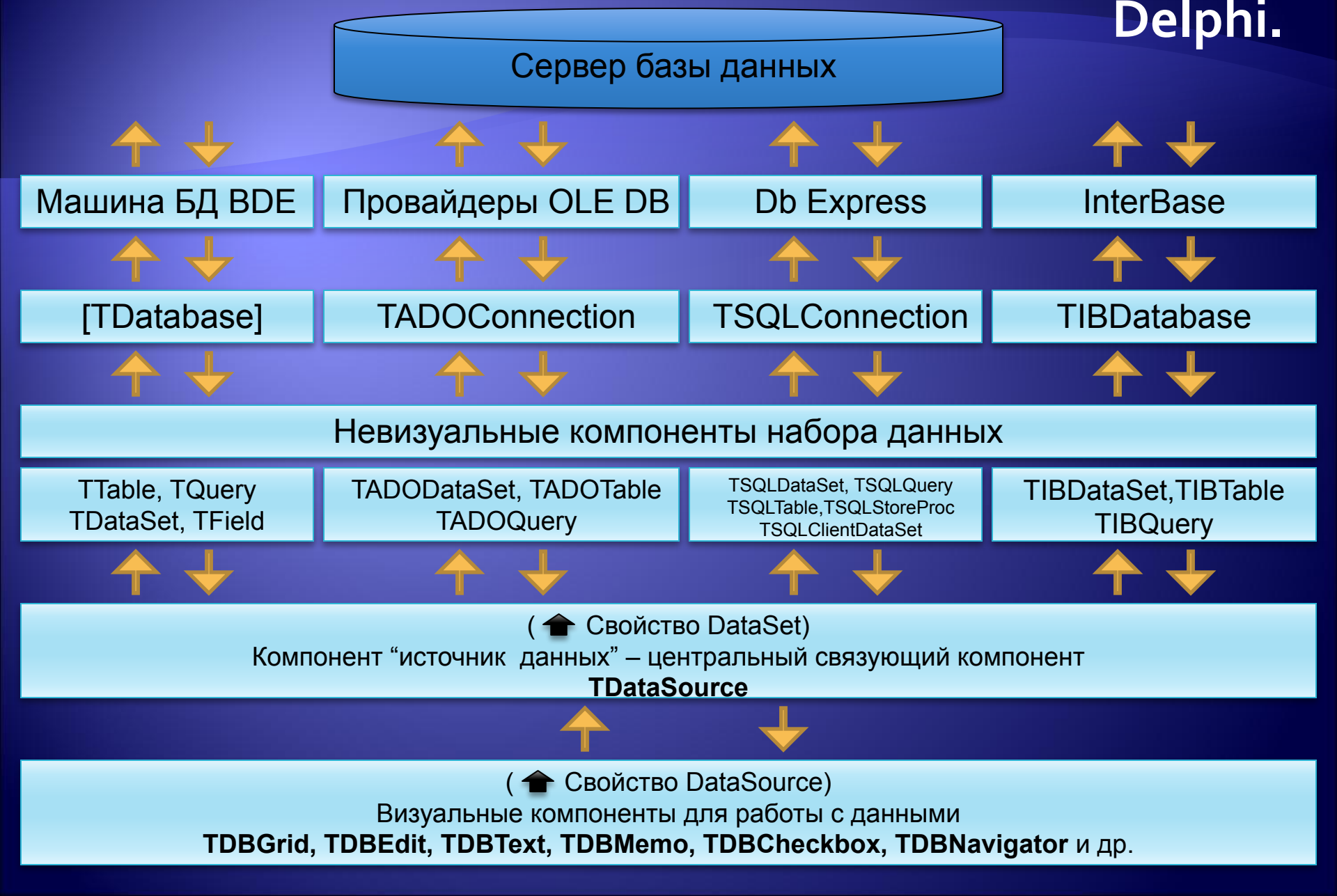
```
// Typedef struct tagDBDATE {
// SHORT year;
// USHORT month;
// USHORT day;
// } DBDATE;
DBTYPE_DBTIME = 134,
DBTYPE_DBTIMESTAMP = 135 // Для даты и времени,

// определяемых как структура
// Typedef struct tagDBTIMESTAMP {
// SHORT year;
// USHORT month;
// USHORT day;
// USHORT hour;
// USHORT minute;
// USHORT second;
// ULONG fraction;
} DBTIMESTAMP;
DBTYPE_HCHAPTER = 136
DBTYPE_PROPVARIANT = 138,
DBTYPE_VARNUMERIC = 139
};
```

# Доступ к OLE DB

- Будучи COM-интерфейсом, OLE DB непосредственно доступен из C++, C# и Java, но недоступен из Visual Basic и сценарных языков. Поэтому чтобы использовать OLE DB из любых языков программирования необходимо использовать объектный интерфейс ADO.

# Реализация доступа к базам данных в среде Delphi.



# Существует ли возможность в Delphi работать напрямую с OLE DB ?

# Доступ к OLE DB в Delphi

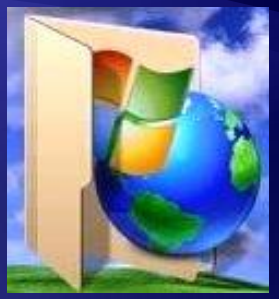

- Работать в Delphi с OLE DB можно и напрямую. Существует несколько низкоуровневых интерфейсов для работы напрямую с OLE DB.
- В частности, OLEDB Direct Components представляет собой набор **VCL** компонентов, которые позволяют получить доступ к базам данных в Delphi и C++ Builder с помощью OLE интерфейсов напрямую.

www.oledbdirect.com

# Visual Component Library


- **Библиотека визуальных компонентов** (Visual Component Library, **VCL**) — объектно-ориентированная библиотека для разработки программного обеспечения, разработанная компанией «Borland» для поддержки принципов визуального программирования. VCL входит в комплект поставки «Delphi», «C++ Builder» и «Borland Developer Studio» и является, частью среды разработки. VCL представляет огромное количество готовых к использованию компонентов для работы в самых разных областях программирования, таких, например, как интерфейс пользователя, работа с базами данных, взаимодействие с операционной системой, программирование сетевых приложений и прочее.

# Список используемой литературы




- С.В. Глушаков, Д.В. Ломотько “Базы данных: учебный курс”, 2001.
- И.Ю. Баженова “Основы проектирования приложений баз данных”, 2006.
- В.В. Корнеев, А.Ф. Гареев “Базы данных, интеллектуальная обработка информации”, 2001.
- Г.С. Иванова “Технология программирования”, 2003.
- Д. Крёнке “Теория и практика построения баз данных”, 2006.
- www.wikipedia.org
- www.securitylab.ru
- www.oledbdirect.com

# СПАСИБО ЗА ВНИМАНИЕ