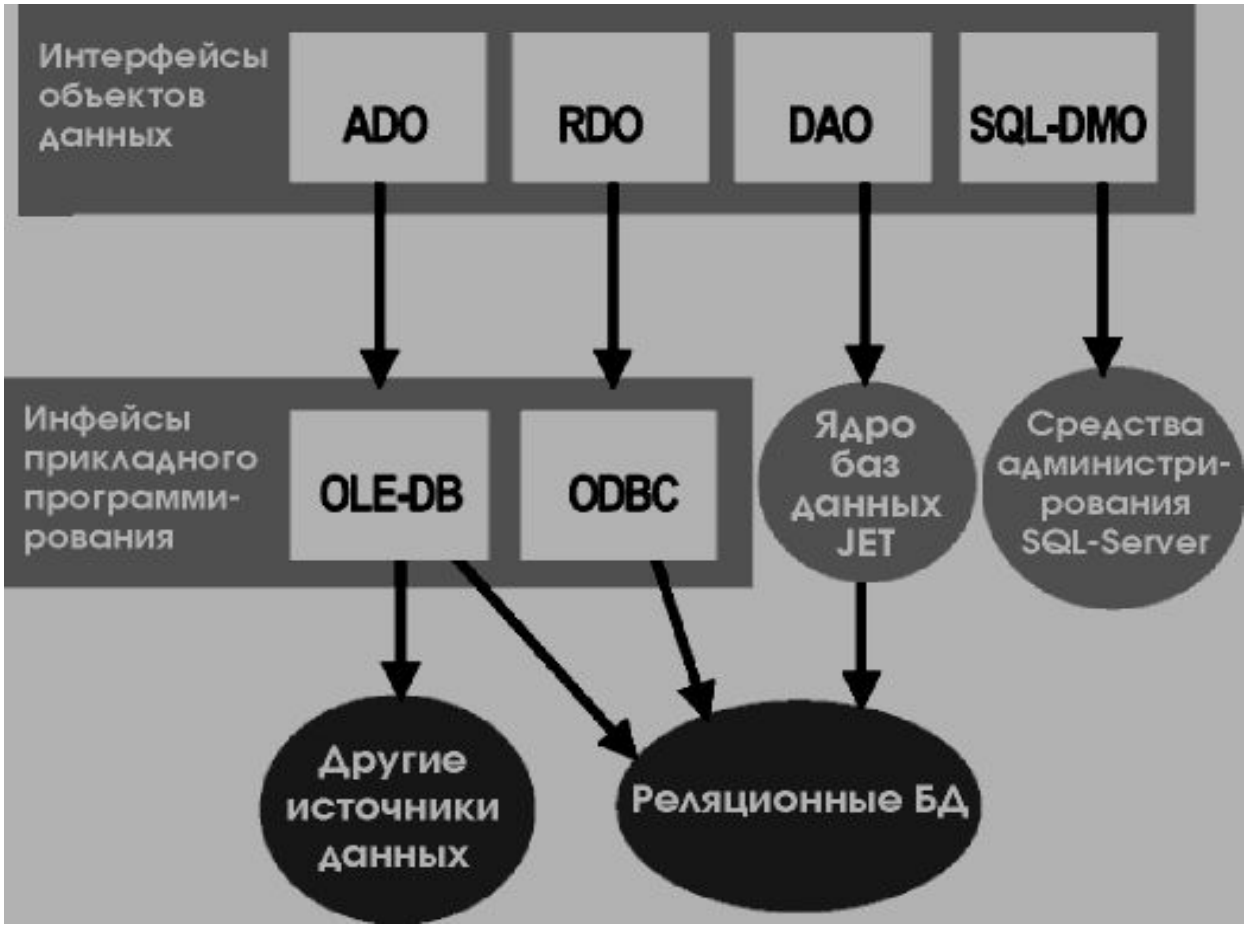
Технологии доступа к данным

Эволюция способов работы с данными

- Эволюция структур данных породила системы упорядочивания, структуризации и управления данными – *серверы баз данных*.
- Первоначально наиважнейшей характеристикой серверов БД считалась их низкая стоимость. На этом этапе развития архитектура приложений состояла из двух компонентов: клиента и сервера. Последний разрабатывался специально для установленного сервера БД, требования которого к операционной системе были весьма специфичными.
- Современный подход, предложенный разработчиками Microsoft, позволил выполнять однотипные манипуляции над разнообразнейшими наборами данных, вне зависимости от их внутренней структуры и типа сервера БД. Этот подход вошел в историю под названием ***UDAS (Universal Data Access Strategy)***.
- Microsoft ADO – это современный интерфейс промежуточного уровня, позволяющий создавать приложения, работающие со всевозможными, разнообразными как по своей структуре, так и по своему происхождению (серверу) базами данных единообразно, максимально просто и оптимизированно.

Интерфейсы баз данных для разработчиков



Основные термины

- Независимо от того, как определяется понятие архитектуры "клиент-сервер", в основе этого понятия лежит распределенная модель вычислений. В самом общем случае под *клиентом* и *сервером* понимаются два взаимодействующих процесса, из которых один является поставщиком некоторого сервиса для другого.
- *Сервер* – логический процесс, который обеспечивает некоторый сервис по запросу от клиента. Обычно сервер не только выполняет запрос, но и управляет очередностью запросов, буферами обмена, извещает своих клиентов о выполнении запроса и т. д.
- *Клиент* – процесс, который запрашивает обслуживание от сервера. Процесс не является клиентом по каким-то параметрам своей структуры, он является клиентом только по отношению к серверу.
- При взаимодействии клиента и сервера инициатором диалога с сервером, как правило, является клиент, сервер сам не инициирует совместную работу. Это не исключает, однако, того, что сервер может извещать клиентов о каких-то зарегистрированных им событиях. Инициирование взаимодействия, запрос на обслуживание, восприятие результатов от сервера, обработка ошибок – это обязанности клиента.

Отличие архитектуры "клиент-сервер" от архитектуры "файл-сервер"

1. Сетевое многопользовательское приложение строится по принципу *файл-серверной архитектуры*. Данные в виде одного или нескольких файлов размещаются на файловом сервере. Файловый сервер принимает запросы, поступающие по сети от компьютеров-клиентов, и передает им требуемые данные.

   Однако обработка этих данных выполняется на компьютерах-клиентах. На каждом из компьютеров запускается полная копия процессора обработки данных Jet Engine. Любая копия Jet независимо управляет файлами MDB, содержащими данные. Единственная связь между этими независимыми действиями — файл блокировок (файл, который имеет имя, совпадающее с именем файла приложения, но с расширением Idb), который обязательно создается для каждого файла базы данных с расширением mdb. При этом каждая копия Jet выполняет изменения индексов, работу с системными таблицами и другие функции, входящие в компетенцию СУБД.
2. В *архитектуре "клиент-сервер"* сервер базы данных не только обеспечивает доступ к общим данным, но и берет на себя всю обработку этих данных. Клиент посылает на сервер запросы на чтение или изменение данных, которые формулируются на языке SQL. Сервер сам выполняет все необходимые изменения или выборки, контролируя при этом целостность и согласованность данных, и результаты в виде набора записей или кода возврата посылает на компьютер клиента.

Отличие архитектуры "клиент-сервер" от архитектуры "файл-сервер"

Недостатки архитектуры с файловым сервером:

1. Данные хранятся в одном месте, а обрабатываются в другом. Это означает, что их нужно передавать по сети, что приводит к очень высоким нагрузкам на сеть и, вследствие этого, резкому снижению производительности приложения при увеличении числа одновременно работающих клиентов.
2. Децентрализованное решение проблем целостности и согласованности данных и одновременного доступа к данным. Такое решение снижает надежность приложения.

Архитектура "клиент-сервер" позволяет устранить все указанные недостатки. Кроме того, она позволяет оптимальным образом распределить вычислительную нагрузку между клиентом и сервером, что также влияет на многие характеристики системы: стоимость, производительность, поддержку

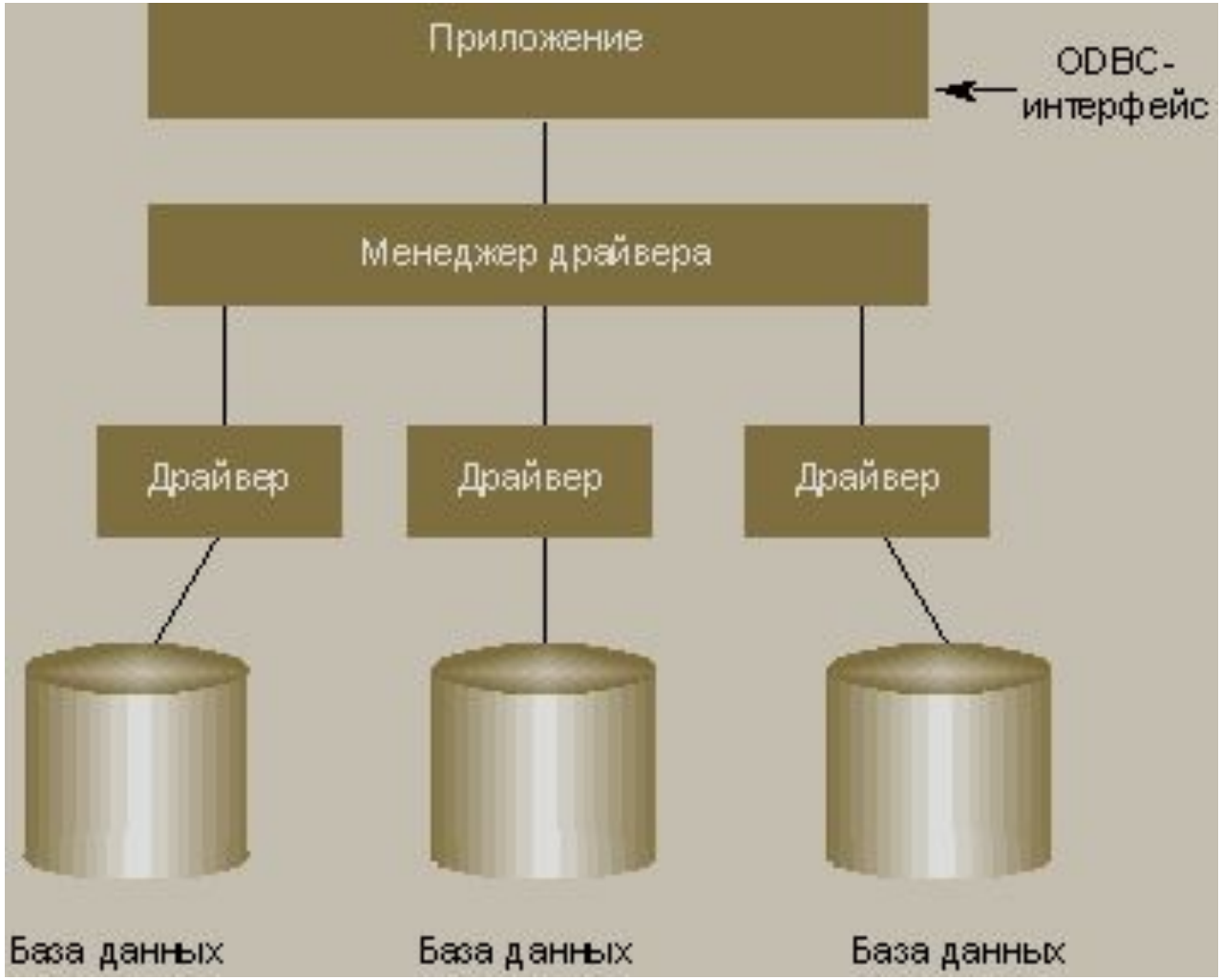
ODBC (Open Database Connectivity)

- В большинстве систем проектирования баз данных приложения основываются на одном типе баз данных. В таких простых схемах разработчик приложения может программировать напрямую, используя системный интерфейс базы данных. Хотя подобный подход обеспечивает быстрый и эффективный доступ к данным, могут возникать проблемы, когда задача расширяется, и разработчику приходится дорабатывать программу. При данном подходе это означает, что каждая готовая программа должна иметь различные версии с поддержкой всевозможных типов баз данных. Если компании расширяются или объединяются одна с другой, приложение должно получить доступ к базам данных, основанным на различных платформах.
- **ODBC (Open Database Connectivity)** – стандартный способ доступа к реляционным данным. Этот компонент универсального механизма доступа к данным оставлен с целью обеспечения совместимости с прежними версиями программного обеспечения. ODBC использует язык SQL как стандарт для доступа к данным.
- Этот интерфейс очень удобен: одно приложение может обращаться к различным базам данных SQL через общий набор команд. Таким образом, разработчик может создавать и распространять приложения, не привязываясь к конкретной базе данных.

ODBC

- Интерфейс Microsoft Open Database Connectivity признан в качестве промышленного стандарта и является компонентом архитектуры *Microsoft WOSA* (Windows Open Services Architecture - открытая архитектура служб среды Windows).
- ODBC дает возможность приложениям обращаться к данным из различных систем управления базами данных. Он обеспечивает максимально широкие возможности для взаимодействия. Например, приложение может обращаться к содержимому различных СУБД, используя единый интерфейс. Более того, то же приложение не будет зависеть ни от одной СУБД, к которым оно производит доступ.
- Пользователи приложения могут добавлять программные компоненты - драйверы, образующие интерфейс между приложением и конкретной СУБД. В комплекте Windows NT Server 4.0 и Windows 2000 Server поставляются ODBC-драйверы для подключения ко всем популярным базам данных.
- Как показано на рисунке 2, менеджер драйверов является промежуточным звеном между приложением и базами данных.
- Интерфейс ODBC содержит набор функций, который управляет каждым инструментом базы данных. Если приложению нужно сменить используемую базу, разработчик просто заменяет один драйвер другим, и приложение может работать как обычно, без необходимости модификации кода программы.
- ODBC использует низкоуровневый интерфейс, поэтому программисты на C и C++ могут задействовать все преимущества технологии ODBC.

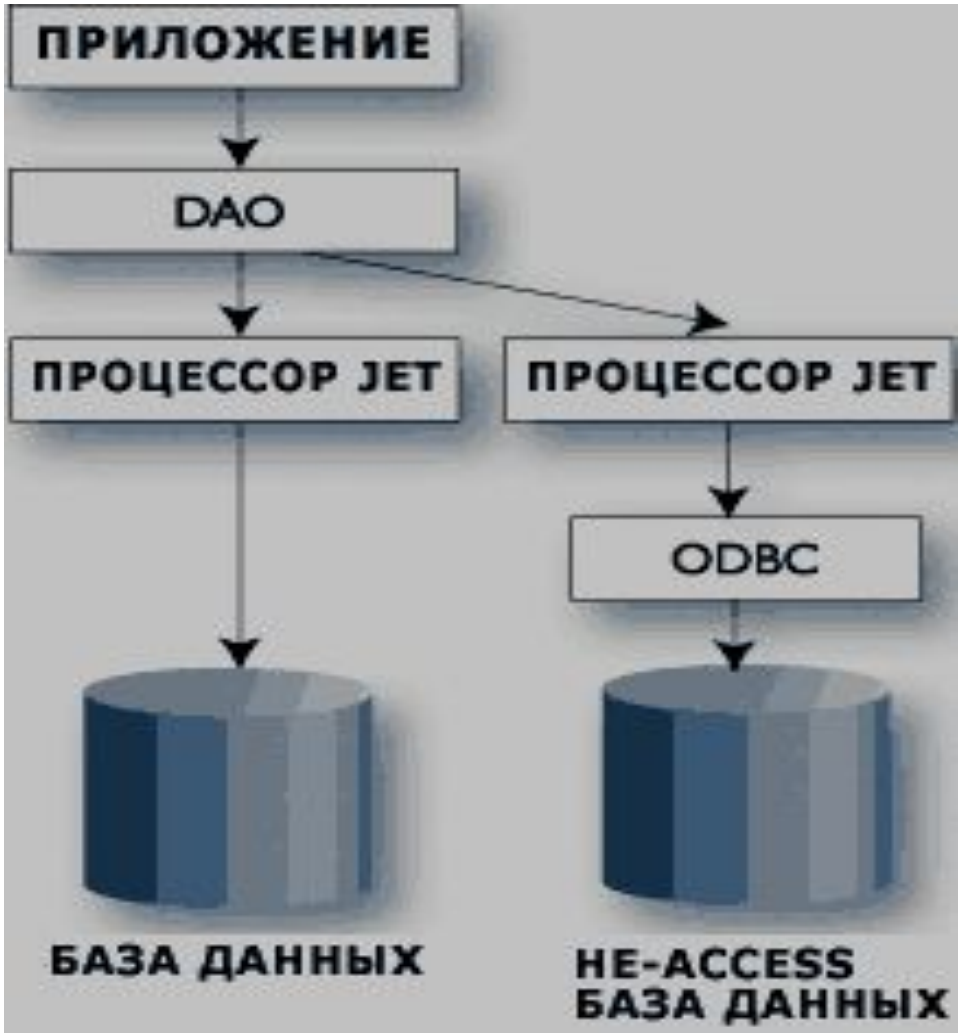
Архитектура ODBC



DAO (Data Access Objects)

- DAO базируется на технологии баз данных Microsoft Jet – процессоре баз данных, предназначенном для Microsoft Access.
- JET был первым объектно-ориентированным интерфейсом для связи с Access. Приложения, использующие Access, могут задействовать DAO для прямого доступа к данным.
- Поскольку DAO создавалась сразу же вслед за Access, применение этой технологии – самый быстрый и наиболее эффективный способ доступа к базам данных Access.
- DAO может работать и с отличными от Access базами данных, такими, как SQL Server и Oracle. DAO использует ODBC, но, поскольку метод DAO спроектирован специально для взаимодействия с JET, JET транслирует запросы между DAO и ODBC. Этот дополнительный шаг трансляции и является причиной замедления работы с базами данных, отличными от Access.

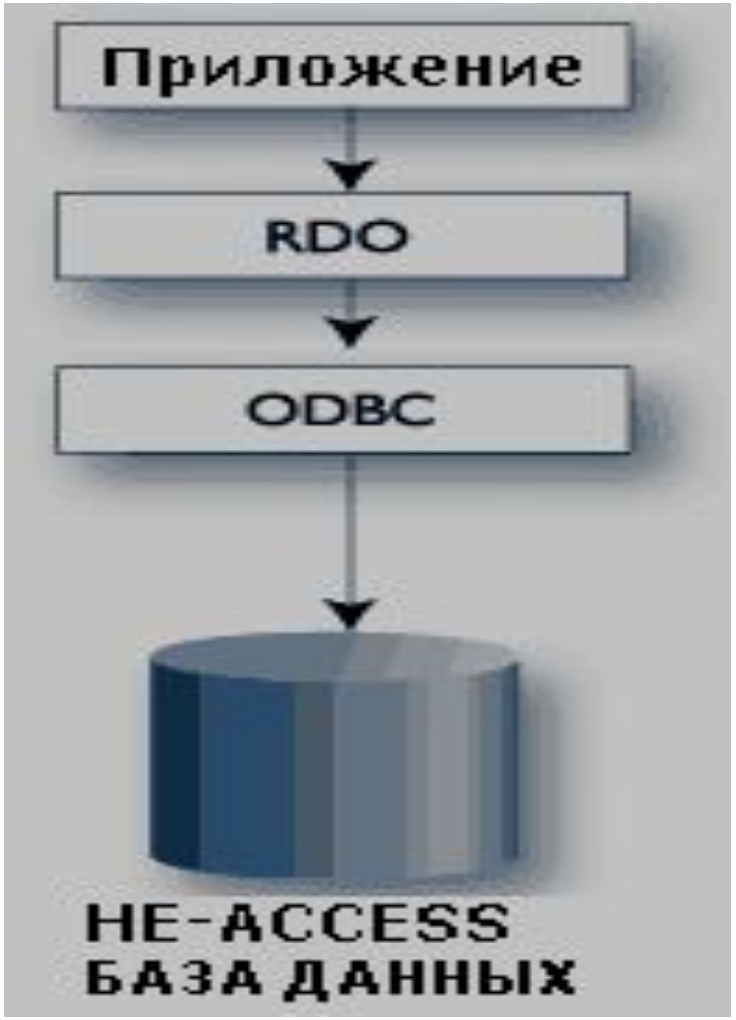
Использование DAO



RDO (Remote Data Objects)

Для ускорения работы с базами данных, отличными от Access, разработчики Microsoft создали RDO. На рисунке 4 показано, что RDO обращается к ODBC API напрямую, минуя JET.

RDO – тонкая надстройка над ODBC, обеспечивающая соединение с БД, создание результирующих наборов данных, курсоров и выполнения хранимых процедур. Технология RDO была создана для работы с такими СУБД, как, например, SQL Server и Oracle.

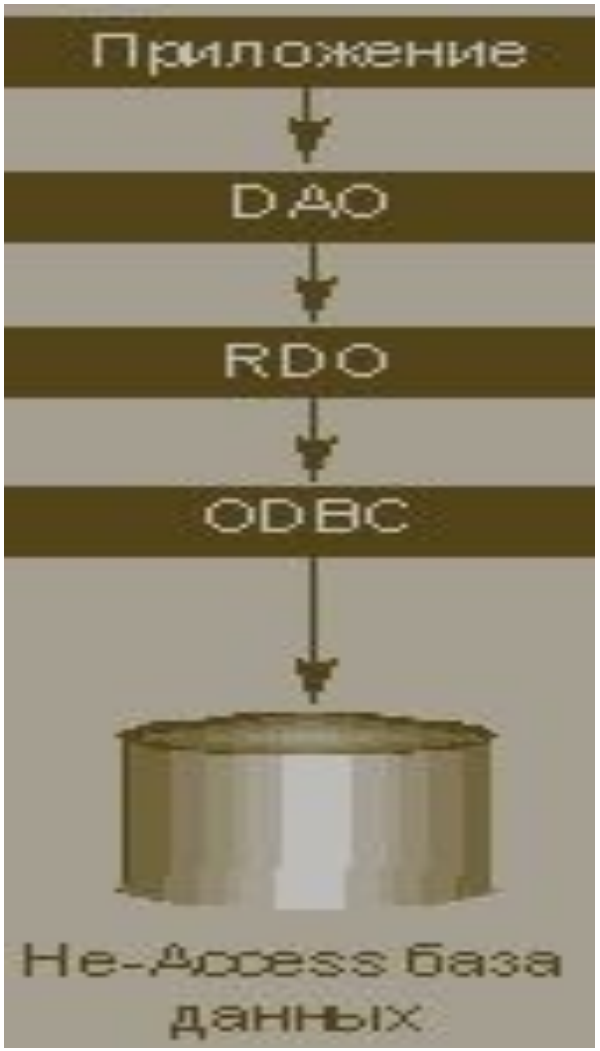


ODBCDirect

ODBCDirect – компонент DAO, реализующий прямой доступ к ODBC-совместимым источникам данных посредством DAO-доступа к объектам удаленных данных.

ODBCDirect осуществляет доступ к удаленным источникам данных через ODBC, не пользуясь средствами ядра Jet. Это позволяет избежать дополнительных затрат времени, вызываемых взаимодействием клиентского и серверного ядра БД.

На рисунке 5 показано, как DAO-приложение, используя ODBCDirect, обращается к базе данных, минуя проблемы, которые вызывает JET.



OLE DB

- **OLE DB** представляет собой программный интерфейс для доступа к различным источникам данных, таким как реляционные и нереляционные данные, текстовые, графические и географические данные, архивы электронных писем, файловая система, бизнес-объекты.
- В спецификации OLE DB определен набор COM-интерфейсов (*Component Object Model* – компонентная модель объектов Microsoft, являющаяся составной частью 32-разрядных версий Windows), инкапсулирующих различные сервисы управления данными и предоставляющих однотипный доступ к перечисленным выше данным. Эти интерфейсы могут быть использованы в приложениях, предоставляющих доступ к данным.
- Технология OLE DB построена на ODBC и расширяет ее до компонентной архитектуры, которая обеспечивает высокоуровневый интерфейс доступа к данным. Эта архитектура предоставляет постоянный доступ к SQL-данным, не SQL-данным и неструктурированным источникам данных по локальным сетям и Internet.

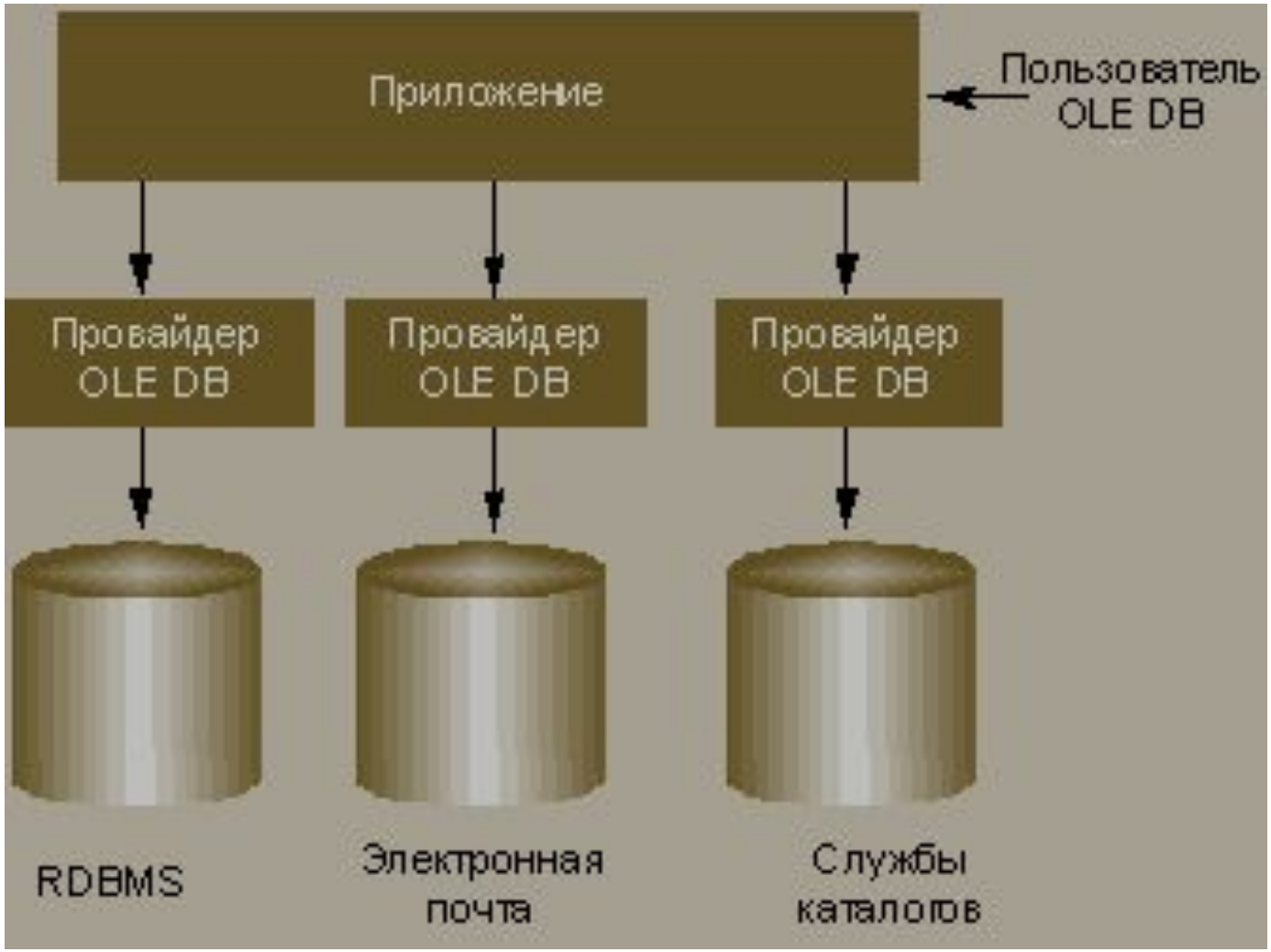
OLE DB

- Для доступа к SQL-данным OLE DB использует ODBC.
- OLE DB обеспечивает связывание для программистов на C и C++, а также программистов, использующих языки с C-подобными вызовами функций. Такие языки, как VB и VBScript, не поддерживают тип данных «указатель» (адресных переменных). Следовательно, они не могут использовать связывание в стиле C и прямое обращение к OLE DB.
- Эта технология позволяет манипулировать с данными независимо от их формата, типа или расположения. OLE DB поддерживает SQL запросы, транзакции и "курсоры" и множество других мощных техник для работы с базами данных. Курсор представляет собой указатель на текущую запись в массиве записей (recordset).
- Существуют различные способы позиционирования курсора. Например, forward-only, когда курсор может двигаться по массиву данных только от первой записи к последней. Это позволяет увеличить скорость доступа к данным, когда нужно быстро прочитать или перебрать все их содержимое.

OLE DB как компонентно-ориентированная технология

- В OLE DB хранилища данных предоставляют интерфейсы, отражающие их изначальный набор возможностей. На основе этих интерфейсов для обеспечения более устойчивых моделей данных могут строиться общие компоненты.
- OLE DB определяет общие характеристики различных провайдеров данных и сервисов, а также определяет общие интерфейсы, которые отражали бы эти характеристики. Например, хотя набор строк (rowset) и может быть получен при помощи множества различных механизмов, результатом всегда будет набор строк с точно определенными интерфейсами, методами и характеристиками. Работа OLE DB с результатами составной таблицы не отличается от работы с результирующим текстовым файлом, содержащим табличные данные. Такое определение общих интерфейсов позволяет компонентам более эффективно расширять индивидуальную функциональность провайдера данных.
- После определения базового набора возможностей всю дополнительную функциональность следует рассматривать как дополнения к базовой. Таким образом, более сложные провайдеры могут предоставлять свои расширенные возможности в дополнение к интерфейсам базового уровня. Более того, отдельные обслуживающие компоненты могут реализовывать эти возможности, строясь на базе более простых провайдеров.

OLE DB

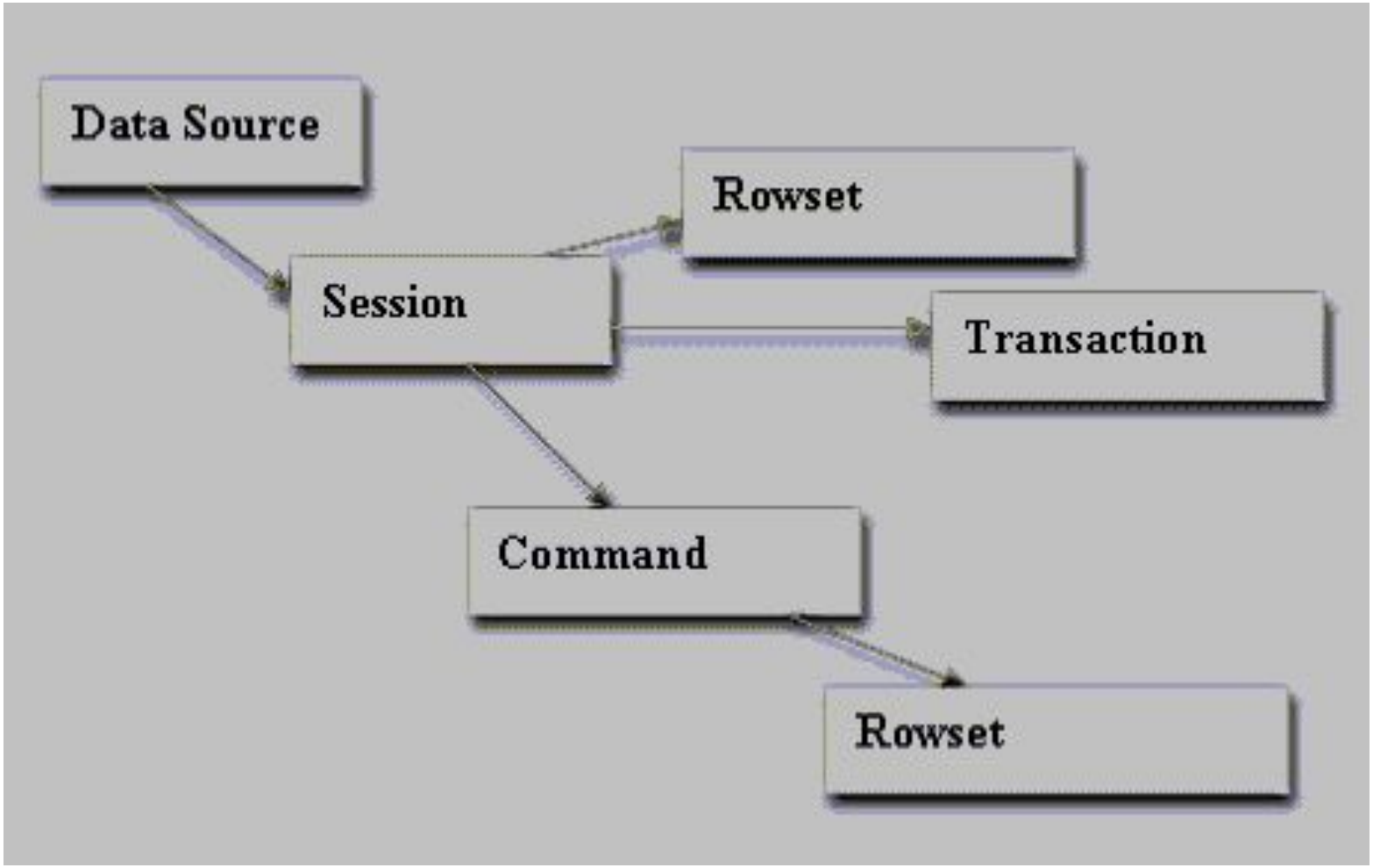

Приложение
Пользователь OLE DB
Провайдер OLE DB
Провайдер OLE DB
Провайдер OLE DB
RDBMS
Электронная почта
Службы каталогов

Компоненты OLE DB

В OLE DB определена иерархия компонентов, каждый из которых является COM-объектом:

1. *Источники данных* (Data Source) – объекты, которые реализуют подключение к источнику данных: Они определяют нужный OLE DB-провайдер, проверяют права доступа потребителя данных и инициируют соединение с источником данных.
2. *Сеансы* (Sessions) – объекты, которые реализуют функции поддержки соединения с источником данных. Они предоставляют контекст для выполнения транзакций и команд. Основная цель сеанса – установить рамки транзакции. Один объект – источник данных может поддерживать несколько сеансов, а значит, и несколько транзакций.
3. *Транзакции* (Transactions) – объекты, которые обеспечивают реализацию механизма транзакций. Они предоставляют методы для того, чтобы начать транзакцию для сеанса или новую транзакцию внутри текущей и подтвердить или отменить транзакцию самого нижнего уровня.
4. *Команды* (Commands) – объекты, которые реализуют выполнение действий с данными (например, запросов). Команды порождаются сеансом, и в одном сеансе можно создать несколько команд.
5. *Наборы рядов* (Rowsets) – объекты, которые предоставляют данные в табличной форме. Они порождаются либо сеансом, либо командой в качестве результата ее выполнения. Непосредственно из сессии можно создать набор рядов, содержащий все данные таблицы. Для реализации такого простого запроса не требуется команды. В остальных же случаях для создания набора рядов используются команды.

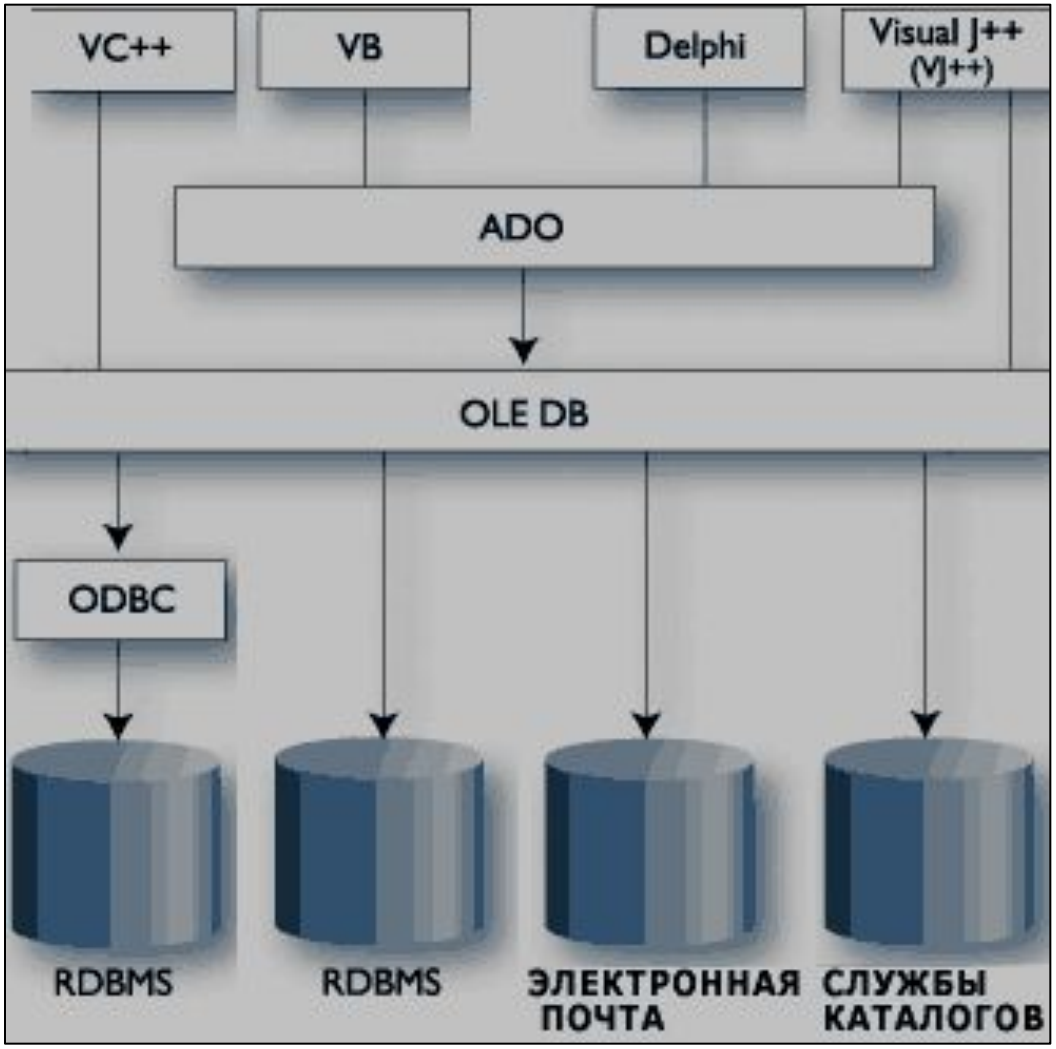
Компоненты OLE DB



ADO (ActiveX Data Objects)

- **ADO** – технология стандартного обращения к реляционным данным от Microsoft. ADO представляет собой высокоуровневый программный интерфейс для доступа к OLE DB-интерфейсам. Он позволяет манипулировать данными с помощью любых OLE DB-провайдеров, как входящих в состав MDAC (Microsoft Data Access Components) некоторых других продуктов Microsoft, так и произведенных сторонними производителями. ADO содержит набор объектов, используемых для соединения с источником данных, для чтения, добавления, удаления и модификации данных.
- ADO работает с объектами DAO и RDO, а также поддерживает более простые модели, чем DAO и RDO (хотя с избыточной функциональностью, так что можно выполнить операцию несколькими способами).
- Объектная иерархия в ADO более однородная, чем в DAO. ADO содержит несколько встроенных объектов, которые упрощают доступ к данным из информационных хранилищ.
- На рисунке 8 показано несколько способов, с помощью которых приложение связывается с базой данных. Например, VB-программист может использовать ADO для соединения приложения с провайдером OLE DB. Если база данных не поддерживает OLE DB, приложение может задействовать ODBC. Программист на Visual C++ может применять ADO или соединяться напрямую через OLE DB.

Различие маршрутов приложений в ADO



ADO

- ADO - это стратегически важный API-интерфейс доступа к базам данных и информации на платформе Windows. Он обеспечивает единообразный доступ к данным и обслуживает многие потребности разработки, включая создание внешних клиентов баз данных и бизнес-объектов промежуточного уровня, использующих приложения, инструменты, языки и Интернет-браузеры. ADO был разработан как единственный интерфейс с данными, необходимый для разработки одно- и многоуровневых решений типа «клиент-сервер» или Интернет-приложений, управляемых данными.
- Главные достоинства интерфейса ADO - простота использования, высокая скорость, низкие дополнительные затраты при работе с памятью и небольшая потребность в дисковых ресурсах.
- ADO образует удобный интерфейс с OLE DB, обеспечивающей нижний уровень доступа к данным. Этот интерфейс оптимизирован с целью уменьшения сетевого трафика в основных режимах работы и представлен минимальным числом уровней между клиентской частью и источником данных. При этом используется уже знакомый интерфейс COM, доступный во всех ведущих средствах на базе технологии RAD (Rapid Application Development - быстрая разработка приложений).

Объекты ADO

1. Соединение (Connection)

Объекты Соединение, Команда и Набор записей являются ключевыми элементами модели ADO. Объект Соединение позволяет приложениям ADO установить связь с нужным источником данных. После этого объект Соединение можно употреблять для непосредственного исполнения команд SQL, а также он может использоваться объектами Команда и Набор записей. Для установления соединения с источником данных следует применить метод Открыть (Open). Закрыть Соединение поможет метод Закрыть (Close).

2. Команда (Command)

Объект Команда используется для передачи команд в источник данных. Для источников данных типа SQL Server состоят из динамических команд SQL, подготовленных команд SQL или вызовов хранимых процедур. Свойство Текст команды (CommandText) содержит саму команду, а метод Исполнить (Execute) ее запускает.

3. Параметр (Parameter)

Объекты Параметр применяются совместно с объектами Команда. Объекты Параметр специфицируют индивидуальные атрибуты для каждого параметра, используемого объектом Команда. Для создания объекта Параметр разработчики часто пользуются методом Создать параметр (CreateParameter). Коллекция объектов Параметр, связанная с некоторым объектом Команда, содержит все объекты такого типа, которые необходимы для заданного объекта Команда.

4. Набор записей (Recordset)

Разработчики значительно чаще пользуются объектами Набор записей, чем любыми другими объектами ADO. Первоначальной целью создания объекта Набор записей была отправка запросов SQL в SQL Server и последующий возврат результатов обработки запроса клиентскому приложению. Свойство Источник (Source) объекта Набор записей содержит сам запрос или команду SQL. Обычно для исполнения на сервере команды SQL применяют метод Открыть (Open) объекта Набор записей, а затем с помощью запроса наполняют набор данных в клиентском приложении.

Объекты ADO

5. Поле (Field)

Каждый объект Поле, используемый вместе с объектом Набор записей, представляет определенный столбец, входящий в состав набора записей. Свойства объекта Поле отражают тип данных столбца, его размер и значение. Коллекция объектов Поле, автоматически формируемая при выполнении метода Открыть (), содержит все объекты Поле для каждого объекта Набор записей.

6. Свойство (Property)

Объект Свойство предоставляет информацию о различных характеристиках объектов Соединение, Команда, Набор записей и Поле. Получить доступ к объектам Свойства можно через любой из этих объектов ADO. Объект Свойства позволяет каждому поставщику OLE DB продемонстрировать уникальные возможности.

7. Ошибка (Error)

Как следует из его названия, объект Ошибка предоставляет пользователям информацию об ошибках, генерируемую средствами ADO в ходе исполнения программ. Несмотря на то, что методы объектов Соединение, Команда и Набор записей могут генерировать сообщения ADO о возникших ошибках, коллекция Ошибки объекта Соединение всегда содержит запись информации об ошибке. Свойство Описание (Description) объекта Ошибка содержит текстовое описание ошибки, а свойство Статус SQL (SQLState) возвращает код ошибки в нотации ODBC или OLE DB, что является гораздо более подробной информацией о ситуации возникновения ошибки.

Пример работы в ADO

- Можно использовать типичный объект – набор строк (Recordset). Объект Recordset представляет собой набор записей (таблицу) и поддерживает типы курсоров *adOpenForwardOnly, adOpenKeyset, adOpenDynamic* и *adOpenStatic*. Курсор может быть как на стороне сервера (по умолчанию), так и на стороне клиента.
- Для доступа к записи ADO требуется просканировать набор строк последовательно. Для доступа к нескольким таблицам необходимо выполнить запрос на объединение JOIN, чтобы получить результат в виде набора строк.
- Хотя объект Recordset поддерживает доступ к данным без соединения с ними, ADO изначально был спроектирован для данных, с которыми установлено соединение. Такой метод доступа заставляет хранить важные ресурсы на стороне сервера.
- Для передачи набора строк следует использовать метод упорядочивания, названный COM marshalling (процесс преобразования типов данных).

Пример работы в ADO

- Для того, чтобы получить доступ к базе, нужно создать объект connection, открыть связь с необходимым провайдером при помощи метода Open и запросить необходимые данные, используя SQL.

- Перемещаться по рекордсету можно при помощи курсора.

- Основные методы и свойства для перемещения в рекордсете:

1. **MoveFirst –** передвигает курсор на первую запись
2. **MoveNext –** передвигает курсор на следующую запись
3. **MoveLast –** передвигает курсор на последнюю запись
4. **RecordCount –** возвращает количество записей
5. **BOF –** возвращает true, если курсор находится в начале рекордсета
6. **EOF –** возращает true, если курсор находится в конце рекордсета
7. **Find –** находит запись по определенному критерию, начиная от положения курсора.

ADO

- Начиная с ADO 2.1, Microsoft добавляет поддержку XML в объектную модель ADO, что позволяет хранить набор строк Recordset как XML-документ. Однако только при появлении ADO 2.5 ряд ограничений XML, который сохранялся в версии ADO 2.1 (например, жесткая иерархия объектов Recordset), был устранен.
- Хотя ADO может преобразовать документ XML в набор Recordset, он в состоянии читать только документы в собственной схеме, известной как *Advanced Data TableGram* (ADTG).
- В поисках механизма доступа к несвязанным данным Microsoft расширяет ADO и вводит службу Remote Data Services (RDS).
- RDS создана после ADO и разрешает передачу объекта Recordset клиенту (например, в Web-браузер) при отсутствии активного соединения. Однако RDS, как и ADO, использует упорядочивание COM marshaling для передачи набора строк от сервера клиенту.

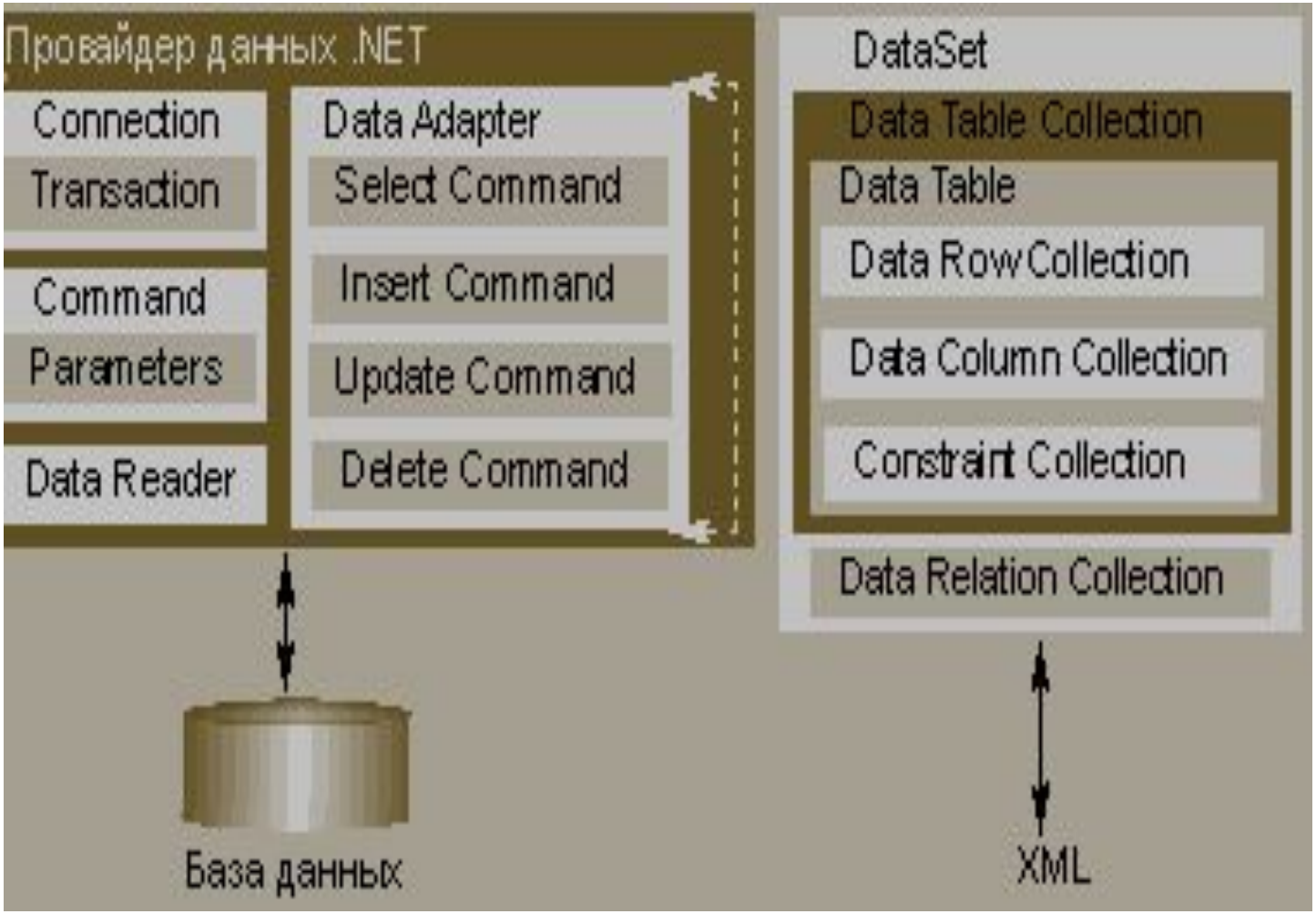
.NET

- Когда Microsoft начала разрабатывать .NET Framework, она имела хорошую возможность пересмотреть модель доступа к данным. Решив не продолжать разработку технологии ADO, специалисты Microsoft приступили к созданию новой структуры доступа к данным, при этом сохранив акроним.
- Microsoft разрабатывает ADO.NET на базе уже зарекомендовавшей себя объектной технологии ADO. Но ADO.NET ориентируется на три важные возможности, которые не поддерживаются ADO:

1. поддержка модели доступа к несвязанным данным, что является ключевым элементом для работы в Web;
2. поддержка тесной интеграции с XML;
3. интеграция с .NET Framework (например, совместимость с базовой библиотекой классов типичной системы).

- Архитектура ADO.NET. На Рисунке 9 представлена архитектура ADO.NET. Объект Recordset, который выполняет так много функций в ADO, здесь отсутствует. Вместо него в ADO.NET предусмотрено несколько особых объектов, выполняющих специфические задачи.

Архитектура ADO.NET



.NET

- Поставщик данных .NET состоит из следующих основных компонентов:

1. Connection – объект для связи с источником данных;
2. Command – объект выполняет команды над источником данных;
3. DataReader – читает данные из источника данных в однонаправленном режиме «только чтение». Объект обеспечивает эффективный поиск данных на стороне сервера. Этот объект полезен для Web-приложений, которые используют DataReader для отображения данных на Web-страницах.
4. DataAdapter – читает данные из источника данных и использует их для заполнения объекта DataSet. Основное преимущество DataAdapter состоит в том, что он может работать с любыми источниками данных. Источник данных может быть как базой данных, так и XML-документом.
5. DataSet – поддерживает копии записей из базы данных без соединения. Он сохраняет записи из таблицы (или множества таблиц) в памяти, не поддерживая постоянного соединения с сервером. В памяти DataSet представляет собой двоичный объект. Когда его перемещают или преобразуют, он представляется как DiffGram (формат XML). Поскольку XML - это текстовый формат, записи могут передаваться по Web - в обход ограничений брандмауэров. DataSet также содержит различные объекты, такие, как ограничения, зависимости и представления, которые позволяют работать с таблицами на клиентской стороне, а не только с RecordSet.

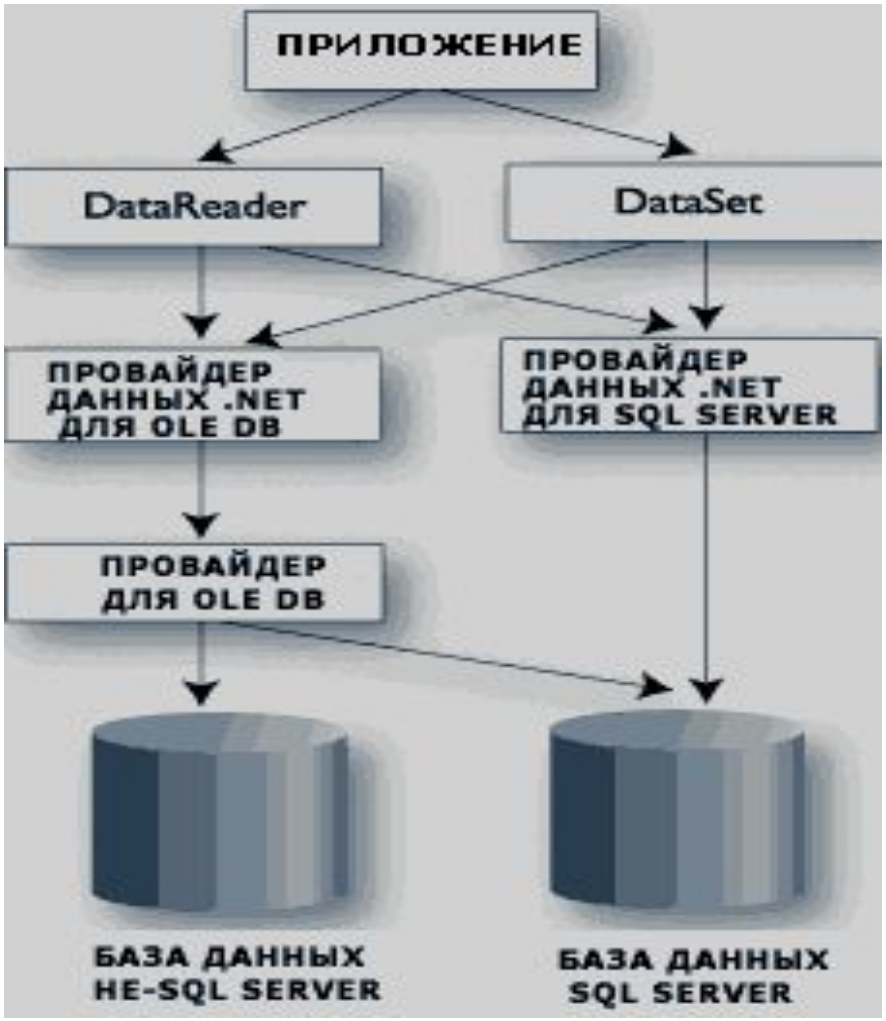
.NET

- Visual Studio .NET содержит два поставщика данных:

1. Поставщик данных SQL Server .NET обеспечивает связь с SQL Server 7.0 и более поздними версиями. Этот метод доступа наиболее эффективен для SQL Server 7.0 и выше, потому что поставщик данных SQL Server .NET связывается напрямую с SQL Server через протокол Tabular Data Stream (TDS).
2. Поставщик данных OLE DB .NET необходим для соединения с отличными от SQL Server базами данных, такими, как Oracle или IBM DB2. Этот поставщик данных использует OLE DB для соответствующих баз данных.

- На рисунке 10 показаны различные пути, по которым приложение может связываться с базой данных через ADO.NET. При выборе пути сначала определяется, какой поставщик данных .NET будет использоваться. Если это SQL Server 7.0 или более поздняя версия, то подключается поставщик данных SQL Server.NET. Если база данных SQL Server 6.5 или отличная от SQL Server (например, Oracle), понадобится поставщик данных OLE DB .NET.
- Далее необходимо определить, какую задачу требуется выполнить. Если надо просто прочитать и отобразить данные из источника данных, объекта Data Reader вполне достаточно.
- Но если предстоит манипулировать данными (например, редактировать или удалять), нужно использовать объект Data Set. Хотя задействовать этот объект следует только в случае необходимости, потому что он работает медленнее, чем Data Reader (Data Set использует Data Reader для заполнения таблиц).

.NET


ПРИЛОЖЕНИЕ
DataReader
DataSet
ПРОВАЙДЕР ДАННЫХ .NET ДЛЯ OLE DB
ПРОВАЙДЕР ДАННЫХ .NET ДЛЯ SQL SERVER
ПРОВАЙДЕР ДЛЯ OLE DB
БАЗА ДАННЫХ НЕ-SQL SERVER
БАЗА ДАННЫХ SQL SERVER