# Visual Basic - работа с базами данных.

Конспект лекций, прочитанный на кафедре ФМЭГ

ФТ-факультета НТУ «ХПИ»

к.ф.-м.н. доцентом

Шкалето Владимиром Ивановичем

# Введение

**Database** (База данных) - это место, где хранится информация. Точнее, это место, где хранится упорядоченная информация, предназначенная для использования с определенной целью. Информация в базе данных должна быть упорядочена так, чтобы программы и приложения обеспечивали конечным пользователям удобную и продуктивную работу с ней.

# Сводка важнейших терминов

- **База данных** - это совокупность таблиц;
- **Таблица** хранит упорядоченную, взаимосвязанную информацию. Каждая таблица состоит из множества записей;
- каждая **запись** содержит информацию об одной сущности (например, об издательстве или счете). Информация в каждой записи разбита на поля;
- **поле** - это минимальная единица хранения в базе данных. Оно может содержать текст, числа, даты и даже фотографии;

# Сводка важнейших терминов

- чтобы **организовать** информацию в базе данных, используются индексы (один или несколько).
- **Индекс** содержит ссылки на записи, хранящиеся в отсортированном виде, и позволяет быстро найти нужные записи;
- в каждой таблице могут быть определены первичные и внешние **ключи**, дающие возможность организовать связи между таблицами.

# Составляющие VB-программы для работы с базами данных

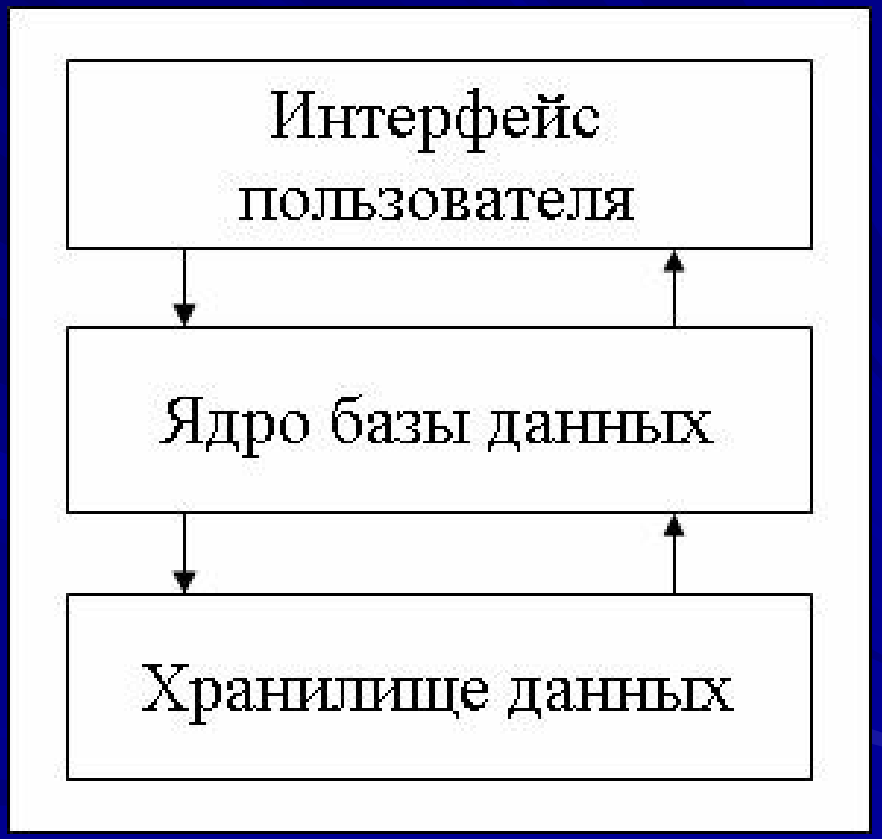
В программе на **Visual Basic** 6 для работы с базой данных можно выделить три составляющих:

- пользовательский интерфейс,
- ядро базы данных и
- хранилище данных.

Ниже на рисунке показано, как эти части соотносятся друг с другом. Рассмотрим указанные компоненты.

# Пользовательский интерфейс

- **Пользовательский интерфейс User interface** - это часть программы, с которой взаимодействует пользователь (будет разрабатываться интерфейс на **Visual Basic**).



# Пользовательский интерфейс

- Обычно он состоит из **screen forms** (экранных форм) для отображения и модификации данных. Помимо этих форм, имеются **data modules** (модули данных) и **classes** (классы), содержащие VB-код. В коде вы сообщаете базе данных о необходимости выполнять различные операции:

  **добавить, удалить или обновить записи.**

Также при помощи кода выполняются запросы для извлечения из базы информации, требуемой для составления отчетов и построения графиков.

# Пользовательский интерфейс

Также при помощи кода выполняются запросы для извлечения из базы информации, требуемой для составления отчетов и построения графиков.

Пользователь не обращается напрямую к базе данных (или хранилищу). Запросы, которые формулируются с помощью **Visual Basic**, адресованы не физическому файлу (каким может быть база данных, электронная таблица или текстовый файл).

# Ядро базы данных

Вместо этого посылается команда ядру базы данных. И именно оно выполняет указанные операции и возвращает результаты VB-программе.

База данных (например, Access) состоит из таблиц, в которых хранится информация. База данных может содержать прекрасно организованные, важные сведения.

# Ядро базы данных

На нижнем уровне база данных - это просто файл. Она не умеет манипулировать информацией.

Для этого существует **database engine** (ядро базы данных). Это программа, которая и отвечает за управление информацией в базе данных.

# Ядро базы данных

Когда нужно найти какую-либо информацию, то отправляется запрос ядру базы данных, а оно уже анализирует, какую информацию требуется извлечь, и находит ее в файле базы данных. Именно ядро базы данных манипулирует хранящейся в базе информацией. Поэтому при программировании Access на самом деле программируется **Jet** - ядро базы данных Access.

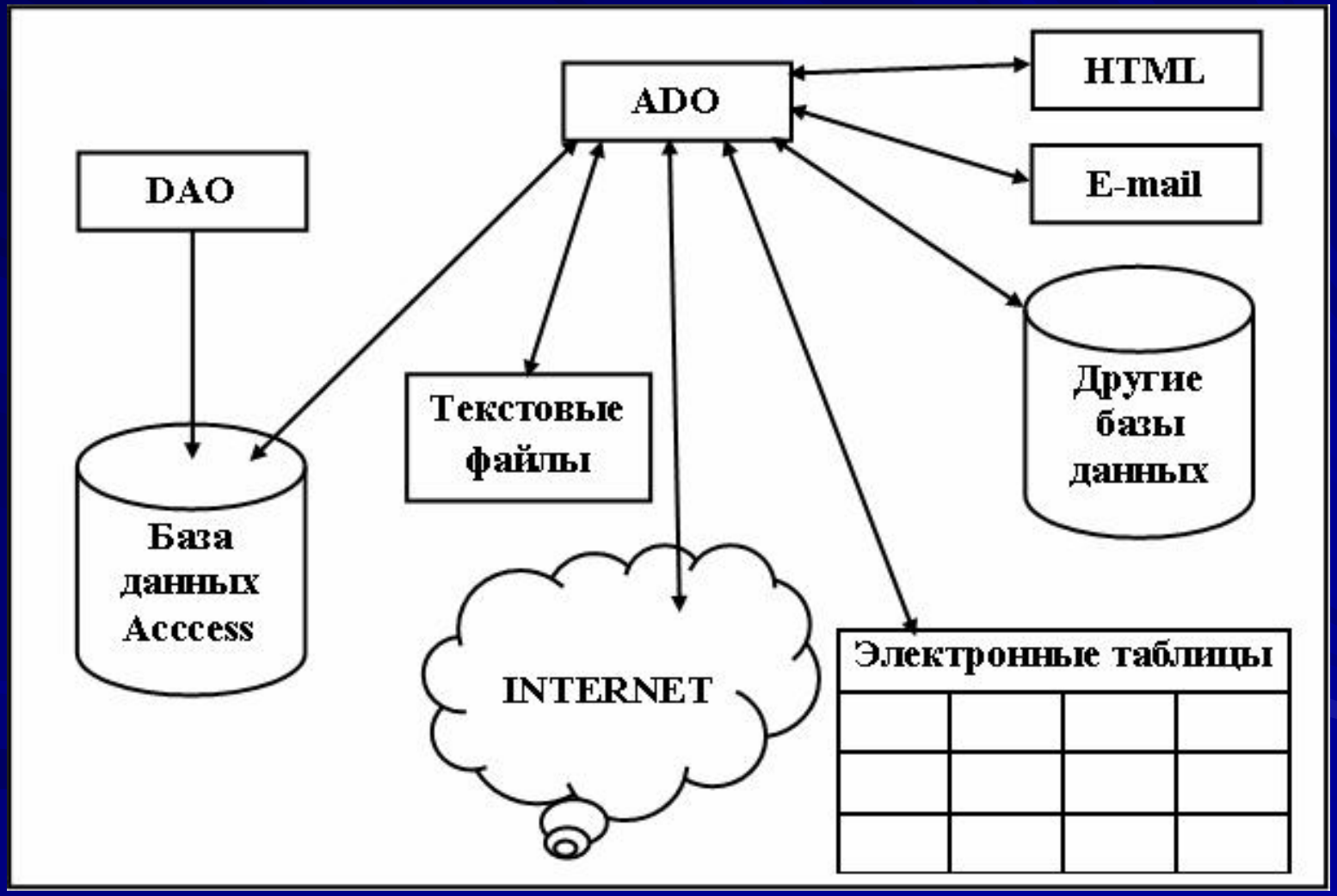
# Хранилище данных

Под **data store** (хранилищем данных) понимается один или несколько файлов на локальном жестком диске или где-то в сети, которые содержат собственно данные, упорядоченные в виде таблиц. Приложения на Visual Basic могут одновременно работать с информацией, хранящейся в нескольких базах данных разных форматов. Однако, в каком бы формате данные ни были записаны, само хранилище пассивно. Оно хранит данные, но ничего с ними не делает.

# Технологии DAO и ADO

Для программирования **Jet** с целью получения данных лучше использовать **Data Access Objects** (**DAO** - объекты доступа к данным). Модель **DAO** позволяет выполнять практически любые операции с базой данных. Намного большими возможностями обладает технология **ActiveX Data Objects** (**ADO** - объекты данных **ActiveX**). В будущем мир постепенно перейдет на технологию доступа к данным **ADO**

# Технологии DAO и ADO



# Некоторые плюсы и минусы DAO и ADO

Причины, по которым стоит использовать DAO:

- если вы модифицируете существующее DAO-приложение, не отбрасывайте признанную технологию;
- если вы разрабатываете небольшую программу, которая будет обращаться к данным, хранящимся на локальной машине или на локальном сервере, DAO - лучший вариант;

# Некоторые плюсы и минусы DAO и ADO

- безусловно, ADO еще недостаточно разработана и испытана, чтобы переносить на нее существующие DAO-приложения. ADO пока не поддерживает разграничение прав пользователей, групп и т.д. К тому же, DAO в большей степени гарантирует защищенность базы данных;

# Некоторые плюсы и минусы DAO и ADO

- программа Microsoft Access сама использует **Jet** и **DAO**, а Access распространяется в составе пакета Microsoft Office Suite, пользующегося огромной популярностью, так что рынок для DAO-приложений сохранится достаточно долго.

# Некоторые плюсы и минусы DAO и ADO

- Еще одно преимущество испытанной технологии - DAO **data control** (Элемент управления данными DAO), который позволяет VB-программам легко и быстро обращаться к базам данных. Он является внутренним элементом, поскольку встроен в язык **Visual Basic** 6.

# Некоторые плюсы и минусы DAO и ADO

С другой стороны, в некоторых ситуациях следует предпочесть ADO:

- если проект находится еще в стадии проектирования и вы собираетесь публиковать данные в Internet или использовать в качестве источника данных не Access;
- технология ADO проще в использовании, чем DAO;

# Некоторые плюсы и минусы DAO и ADO

- если проект находится еще в стадии проектирования и вы собираетесь публиковать данные в Internet или использовать в качестве источника данных не Access;
- технология ADO проще в использовании, чем DAO;

# Некоторые плюсы и минусы DAO и ADO

- ADO мощнее DAO и с одинаковой легкостью обеспечивает доступ к данным из гораздо большего числа источников;
- если вы используете DAO для разработки приложений в архитектуре клиент-сервер и не хотите ограничиваться только ядром **Jet**, вероятно, стоит перейти на ADO уже сейчас.

# Элемент управления данными ADO

**ADO data control** (ADO-элемент управления данными) - это компонент, который можно использовать в VB-программах для подключения к базе данных. Он помещается в форму и соединяется с базой. Данные отображаются в форме другими элементами управления, которые связываются с базой посредством элемента управления данными **ADO**.

# Элемент управления данными ADO

- Самый простой способ сделать все это – воспользоваться входящим в Visual Basic Мастером форм данных.

Основой функционирования Мастера является ADO-элемент управления данными. В составе Visual Basic есть несколько элементов управления, и все они позволяют легко организовать доступ практически к любому источнику информации.

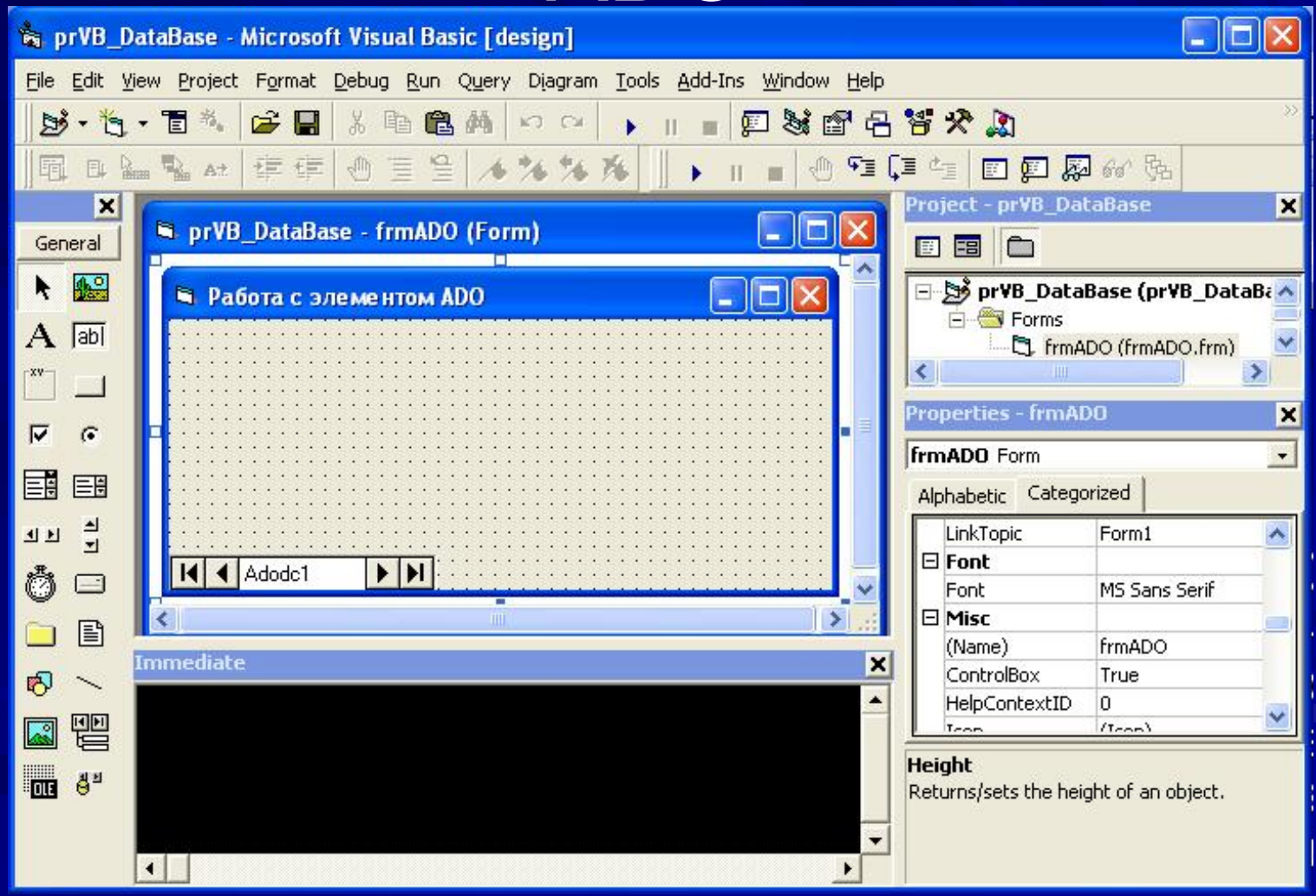
# Элемент управления данными ADO

Но самым удобным из них является ADO-элемент управления данными. С его помощью можно выполнить большинство операций, не написав ни строчки кода. Этот элемент управления дает доступ к данным и позволяет их отображать, манипулировать ими и переходить от одной записи к другой.

# Элемент управления данными ADO

Преимущество нового ADO-элемента управления данными в том, что он является графическим. Этот элемент обладает кнопками навигации, похожими на клавиши видеомагнитофона (первая запись, следующая запись, предыдущая запись и последняя запись), и простым интерфейсом, который позволяет создавать приложения для работы с базами данных без кодирования.

# Элемент управления данными ADO



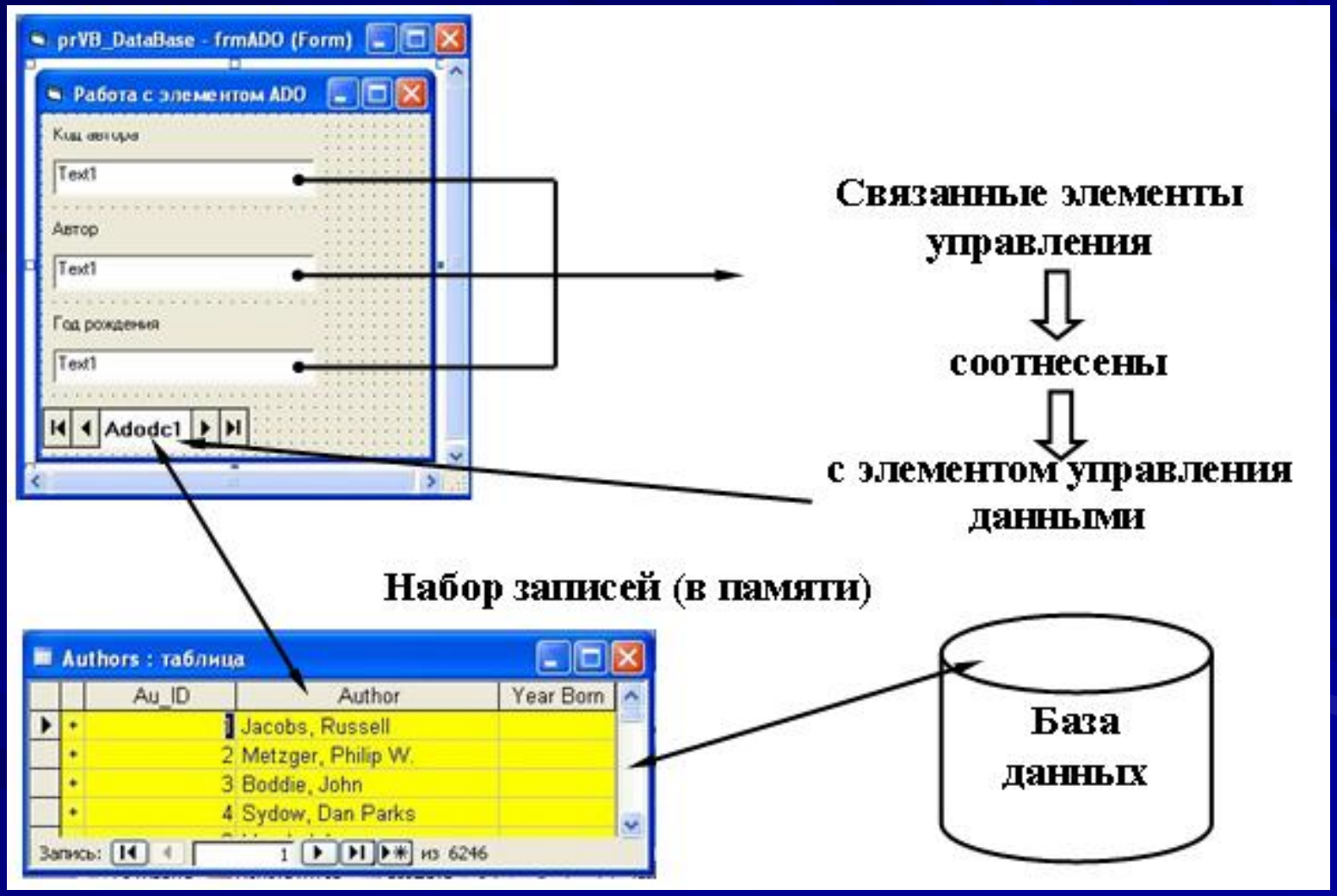
# Набор записей (recordset)

Когда элемент управления данными уже подключен к базе данных, можно воспользоваться любым из информационно-ориентированных элементов управления (их еще называют **bounds controls** (связанные элементы управления)) для отображения записей. Анализируя посланный запрос, ядро базы данных **Jet** строит recordset (набор записей).

# Набор записей (recordset)

**Набор записей** - это рабочая копия сведений, хранящихся в таблицах. Но у пользователя создается ощущение, что он работает напрямую с данными из базы, поскольку, манипулируя набором записей, можно изменять, удалять и добавлять информацию.

# Набор записей (recordset)



# Набор записей (recordset)

Другими словами, информационно-ориентированный элемент управления предоставляет доступ к одному или нескольким полям записи базы с помощью **ADO**-элемента управления данными. Затем, когда пользователь щелкает по одной из кнопок навигации, происходит переход к другой записи.

# Набор записей (recordset)

Во всех связанных элементах управления отображаются данные из полей новой записи. Более того, если пользователь изменяет данные в связанном элементе управления и потом переходит к новой записи, изменения сохраняются в базе автоматически.

# Реляционная база данных

Теория реляционной базы данных разработана в начале 70-х годов Коддом на основе математической теории отношений. В реляционной базе данных все данные хранятся в виде таблиц. Основными понятиями в этой теории являются ***таблица, отношение, строка, столбец, ключи.***

Основой любой базы данных являются таблицы. Таблица состоит из строк и столбцов и имеет уникальное имя в базе данных.

# Реляционная база данных

База данных содержит множество таблиц, связь между которыми устанавливается с помощью совпадающих полей. В каждой из таблиц содержится информация о каких-либо объектах одного типа (группы).

Между таблицами в базе данных устанавливаются отношения. Имеются четыре типа отношений между таблицами: **один-к-одному, один-ко-многим, много-к-одному, много-ко-многим.**

# Отношение один-к-одному

Отношение **один-к-одному** означает, что каждая запись одной таблицы соответствует только одной записи в другой таблице.

Например, если рассматривать таблицы, одна из которых содержит данные о сотрудниках предприятия, а вторая — профессиональные сведения, то можно сказать, что между этими таблицами существует отношение один-к-одному, поскольку для одного человека, информация о котором содержится в первой таблице, может существовать только одна запись, содержащая профессиональные сведения, во второй таблице.

# Отношение один-ко-многим

Наиболее часто встречающимся типом отношений в базе данных является отношение **один-ко-многим**.

В качестве иллюстрации данного типа отношения можно обратиться к таблицам, содержащим информацию о клиентах предприятия и сделанных ими заказах. В качестве других примеров могут быть рассмотрены отношения между предприятием и работающими на нем сотрудниками. Аналогичный тип отношения существует между компьютером и входящими в него компонентами и т. д.

# Отношение много-к-одному

Отношение **много-к-одному** аналогично рассмотренному ранее типу **один-ко-многим**.

Тип отношения между объектами зависит от вашей точки зрения. Например, если вы будете рассматривать отношение между сделанными заказами и клиентами, то получите отношение много-к-одному.

# Отношение много-ко-многим

Отношение **много-ко-многим** возникает между двумя таблицами в тех случаях, когда:

- одна запись из первой таблицы может быть связана более чем с одной записью из второй таблицы;
- одна запись из второй таблицы может быть связана более чем с одной записью из первой таблицы.

Между таблицами, содержащими данные сведения, существует отношение много-ко-многим, так как на каждый поставляемый товар может быть более одного заказа. Аналогично, каждый заказанный товар может производиться более чем одним предприятием.

# Нормализация данных

При проектировании базы данных вам необходимо решить вопрос о наиболее эффективной структуре данных. Основные цели, которые при этом преследуются:

- обеспечить быстрый доступ к данным в таблицах;
- исключить ненужное повторение данных, которое может являться причиной ошибок при вводе и нерационального использования дискового пространства вашего компьютера;
- обеспечить целостность данных таким образом, чтобы при изменении одних объектов автоматически происходило соответствующее изменение связанных с ними объектов.

# Нормализация данных

Процесс уменьшения избыточности информации в базе данных называется ***нормализацией.*** Для нормализации баз данных разработаны достаточно формализованные подходы по разбиению данных, обладающих сложной структурой, на несколько таблиц. Эти вопросы детально освещаются в специальной литературе. Рассмотрим некоторые практические аспекты нормализации таблиц, не вдаваясь в их теоретическое обоснование.

# Нормализация данных

Теория нормализации оперирует с **пятью нормальными формами таблиц**. Эти формы предназначены для уменьшения избыточной информации от первой до пятой нормальной формы. Поэтому каждая последующая нормальная форма должна удовлетворять требованиям предыдущей формы и некоторым дополнительным условиям.

# Нормализация данных

В качестве примера рассмотрим таблицу, которая содержит следующую информацию о покупателе и сделанных им заказах:

- сведения о покупателях;
- дату заказа и количество заказанного товара;
- дату выполнения заказа и количество проданного товара;
- характеристику проданного товара (наименование, стоимость).

# Нормализация данных

Эту таблицу можно рассматривать как однотабличную базу данных. Основная проблема заключается в том, что в ней содержится значительное количество повторяющейся информации. Например, сведения о каждом покупателе повторяются для каждого сделанного им заказа. Такая структура данных является причиной следующих проблем, возникающих при работе с базой данных.

# Нормализация данных

Вам придется тратить значительное время на ввод повторяющихся данных. Например, для всех заказов, сделанных одним из покупателей, вам придется каждый раз вводить одни и те же данные о покупателе.

- При изменении адреса или телефона покупателя необходимо корректировать все записи, содержащие сведения о заказах этого покупателя.

# Нормализация данных

- Наличие повторяющейся информации приведет к **неоправданному увеличению размера базы данных**. В результате снизится скорость выполнения запросов. Кроме того, повторяющиеся данные нерационально используют дисковое пространство вашего компьютера.
- Любые внештатные ситуации потребуют от вас значительного времени для получения требуемой информации. Например, при многократном вводе повторяющихся данных возрастает вероятность ошибки. При больших размерах таблиц поиск ошибок будет занимать значительное время.

# Нормализация данных

Таблица, содержащая сведения о сделанных заказах, является **ненормализованной**. Таблица базы данных **в первой нормальной форме должна**, например, удовлетворять следующим требованиям:

- таблица не должна иметь повторяющихся записей;
- в таблице должны отсутствовать повторяющиеся группы полей;
- строки должны быть не упорядочены;
- столбцы должны быть не упорядочены.

# Нормализация данных

Для выполнения первого условия каждая **таблица должна иметь уникальный индекс**. С этой целью в таблицу следует добавить поле, которое будет содержать уникальные значения. Например, в таблицу, содержащую информацию о покупателях, необходимо добавить поле с кодом клиента.

# Нормализация данных

Второе требование постулирует устранение повторяющихся групп. Поскольку каждый покупатель может иметь несколько телефонов и сделать несколько заказов, в каждом из которых в свою очередь может заказать несколько товаров, нам необходимы четыре таблицы. Каждая запись этих таблиц будет содержать следующие данные:

# Нормализация данных

| Таблица | Данные |
|---|---|
| Первая | Сведения о покупателях |
| Вторая | Список телефонов покупателя |
| Третья | Номер и дата заказа покупателя, данные о менеджере |
| Четвертая | Код, наименование, количество заказанного и проданного товара |

# Нормализация данных

Отметим, что отношение между связываемыми таблицами **один-ко-многим**. Теперь первая и вторая таблица нормализованы.

В третьей таблице находится повторяющаяся информация о менеджере, оформлявшем заказ. Эту таблицу можно разделить на две. В одну из них вынесем информацию о менеджерах фирмы, а во вторую — номер и дату заказа покупателя, а также код менеджера, оформлявшего заказ.

# Нормализация данных

Рассмотрим четвертую таблицу, содержащую код, наименование, количество заказанного и проданного товара. Она также содержит повторяющуюся информацию о проданном товаре. Эту информацию базы данных можно разместить в отдельной таблице списка товаров, продаваемых фирмой.

В результате проведенной нормализации вместо одной исходной таблицы мы получили шесть простых таблиц, содержащих неповторяющуюся информацию

# Индексы

Информация в таблицы базы данных вводится в произвольном порядке и в этом же порядке сохраняется на диске. Поиск нужной информации в такой таблице затруднен, особенно если она имеет большое количество записей.

Например, рассмотрим таблицу, содержащую список клиентов фирмы. Чтобы облегчить поиск информации о нужном клиенте в этой таблице, данные необходимо упорядочить в алфавитном порядке по фамилиям. Если фамилии клиента вы не знаете, а проводите поиск по району, в котором он проживает, в этом случае данные можно упорядочить по районам.

# Индексы

Одним из основных требований, к базам, является **возможность быстрого поиска** требуемых записей среди большого объема информации.

**Индексы** представляют собой эффективное средство, которое позволяет ускорить поиск данных в таблицах. Таблица может иметь несколько индексов. В зависимости от количества полей, используемых в индексе, различают ***простые*** (по одному ключу) и ***составные индексы*** (по нескольким полям).

# Индексы

Для каждого значения индекса в **индексном файле** содержится уникальная ссылка, указывающая на местонахождение в таблице записи, соответствующей индексу. Поэтому при поиске записи осуществляется не последовательный просмотр всей таблицы, а прямой доступ к записи на основании упорядоченных значений **индекса**.

# Индексы

Важной **особенностью индексов** является то, что Вы можете использовать их для создания **первичных ключей**. В этом случае индексы должны быть **уникальными**. Это означает, что для таблицы, содержащей только одно индексное поле, уникальным должно быть значение этого поля. Для составных индексов величины в каждом из индексных полей могут иметь повторяющиеся значения. Однако индексное выражение должно быть уникальным.

# Элементы управления связанные с данными

Среди стандартных элементов управления в **Visual Basic** есть такие, которые могут быть связаны с данными. Среди них **поле ввода** (TextBox), **флажок** (CheckBox), **комбинированный список** (ComboBox), **изображение** (Image), **метка** (Label), **список** (ListBox) и **графический список** (PictureBox). Кроме этого, **Visual Basic** имеет несколько дополнительных элементов управления **ActiveX**, которые также могут быть связаны с данными.

# Элементы управления ActiveX

Следующие элементы управления ActiveX поставляются со всеми версиями **Visual Basic** 6:

- **DataList** (Список данных);
- **DataCombo** (Комбинированный список данных);
- **DataGrid** (Таблица данных);
- **Microsoft FlexGrid** (Табличное поле);
- **Hierarchical FlexGrid** (Иерархическая таблица);

# Элементы управления ActiveX

- **Rich TextBox** («Обогащенный» текст);
- **Microsoft Chart** (График);
- **DateTimePicker** (Выбор даты и времени);
- **ImageCombo** (Комбинированный графический список);
- **Month View** (Календарь).

# Мастера в Visual Basic

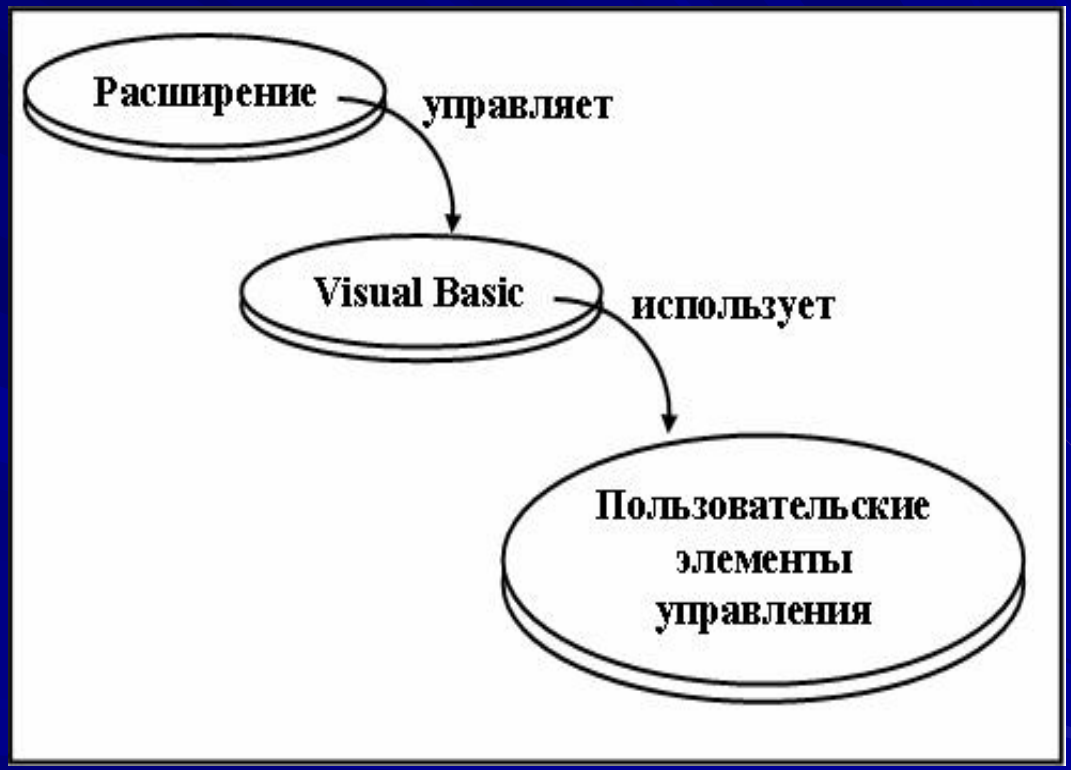
По своей сути, «**Мастера**» - это простые в использовании оболочки для автоматизации различных задач, начиная от простых, но утомительных, и заканчивая самыми сложными. Есть в VB Мастера на любой случай жизни: от создания программ установки дистрибутива программ до построения каркаса приложения или «скелета» программы.

# Расширения

Любой «**Мастер**» не является составной частью среды разработки - **Visual Basic**, но может быть включен в нее для предоставления дополнительных возможностей пользователю. **Integrated Development Environment** (**IDE** - интегрированная среда разработки) **Visual Basic** снабжена фиксированным набором функций. **Add-ins** (расширения) - это инструменты, которые обогащают возможности **IDE**, делая среду более мощной.

# Расширение

- **Wizard (Master, Мастер)** - это особый тип расширения. Как правило, Мастер последовательно «проводит» программиста по этапам некоторого задания, например, построения формы данных.



# Wizard (Master, Мастер)

**Мастер** состоит из набора страниц, по одной для  каждого шага задания, для выполнения которого он предназначен. В левом верхнем углу каждой страницы располагается картинка, а справа от нее - текст; обычно он содержит инструкции для пользователя.

# Источники дополнительных сведений

- Коннэлл Дж. Visual Basic 6. Введение в программирование баз данных: Пер. с англ. - М.: ДМК, 2000. - 720 с: ил. (Серия «Для программистов»).
- Культин Н. Б. Visual Basic. Освой на примерах. — СПб.: БХВ-Петербург, 2004. - 288 с.
- VB6 – книга в формате файлов .html