7_Стадии проектирования БД: определение цели и функций, логическое проектирование, объектно-ориентированное программирование, размещение проекта на сервере и разработка кода доступа к данным, тестирование.

На фазе проектирования архитектуры системы строится предметная модель. Этот процесс включает в себя:

- детальное описание функционирования системы;
- дальнейший анализ используемых данных и построение логической модели данных для последующего проектирования базы данных;
- определение структуры пользовательского интерфейса, спецификации форм и порядка их появления;
- уточнение диаграмм потоков данных и списка событий, выделение среди процессов нижнего уровня интерактивных и неинтерактивных, определение для них миниспецификаций.

Результатами проектирования архитектуры системы являются:

- модель процессов (диаграммы архитектуры системы (SAD) и миниспецификации на структурированном языке. *Миниспецификация* – точная запись логики последовательно выполняемых преобразований входных данных элементарного процесса в выходные);
- модель данных (ERD и подсхемы ERD);
- модель пользовательского интерфейса (классификация процессов на интерактивные и неинтерактивные функции, диаграмма последовательности форм (FSD - Form Sequence Diagram), показывающая, какие формы появляются в приложении и в каком порядке. На FSD фиксируется набор и структура вызовов экранных форм. Диаграммы FSD образуют иерархию, на вершине которой находится главная форма приложения, реализующего подсистему.

- На фазе детального проектирования строится модульная модель. Под модульной моделью понимается реальная модель проектируемой прикладной системы. Процесс ее построения включает в себя:
  - уточнение модели базы данных для последующей генерации SQL-предложений;
  - уточнение структуры пользовательского интерфейса;
  - построение структурных схем, отражающих логику работы пользовательского интерфейса и модель бизнес-логики (Structure Charts Diagram - SCD) и привязка их к формам.
- Результатами детального проектирования являются:
  - модель процессов (структурные схемы интерактивных и неинтерактивных функций);
  - модель данных (определение в ERD всех необходимых параметров для приложений);
  - модель пользовательского интерфейса (диаграмма последовательности форм (FSD), показывающая, какие формы появляются в приложении и в каком порядке, взаимосвязь между каждой формой и определенной структурной схемой, взаимосвязь между каждой формой и одной или более сущностями в ERD).

- На фазе реализации строится реализационная модель. Процесс ее построения включает в себя:
  - генерацию SQL-предложений, определяющих структуру целевой БД (таблицы, индексы, ограничения целостности);
  - уточнение структурных схем (SCD) и диаграмм последовательности форм (FSD) с последующей генерацией кода приложений.
- На основе анализа потоков данных и взаимодействия процессов с хранилищами данных осуществляется окончательное выделение подсистем (предварительное выделение подсистем должно было быть сделано на этапе формулировки требований в техническом задании).
- При выделении подсистем необходимо руководствоваться принципом функциональной связанности и принципом минимизации информационной зависимости. При группировке процессов и данных в подсистемы необходимо учитывать требования к конфигурированию продукта, если они были сформулированы на этапе анализа.

Объектно-ориентированный подход (Гради Буч**)**

- Бучем, Рамбо и Якобсоном (компания Rational Software) в 1997 году была предложена графическая нотация UML, которая группой OMG (Object Management Group) и международным институтом стандартизации ISO к настоящему времени признана промышленным стандартом и реализована во многих CASE-системах.
- *Объект* определяется встроенными в него данными и функциями (методами). Функционирование системы представляется как выполнение действий над объектами. Например, объект «Банк» связан с объектом «Клиент» через объект «Документ», над которым оба выполняют действие «Обработка».
- *Класс* – одно из основных понятий ООП - семейство объектов, обладающих одинаковыми свойствами. Спецификация класса содержит описание представления объектов класса и выполняемых над ними операций.
- *Атрибут* – компонент внутреннего состояния объекта (свойство, реализованное как параметр функции).
- *Метод* – описание выполнения одной из операций над объектом (свойство, реализованное как функция).

- При моделировании объектно-ориентированных систем используются два подхода к делению реальности:
  - на классы и объекты. В графическом представлении для объекта принято использовать тот же символ, что и для его класса, а название объекта подчеркивать.
  - на интерфейс и его реализацию. Интерфейс декларирует контракт, а реализация представляет конкретное воплощение этого контракта и обязуется точно следовать объявленной семантике интерфейса.

Язык моделирования UML (Unified Modeling Language)

- UML предназначен для спецификации, визуализации, документирования, анализа и проектирования программных систем. UML в сконцентрированном виде представляет несколько концепций моделирования:
  - визуальный язык моделирования для разработки моделей, а также для накопления важной информации и обмен ею;
  - поддержка спецификаций, независимых от языка программирования и процесса разработки;
  - возможность организации репозитариев (хранилищ) моделей.

- Средства ООП, включающие в себя как технологии объектно-ориентированного проектирования, так и языки объектно-ориентированного программирования, дают возможность создавать эффективное ПО с меньшими усилиями. Всеми объектно-ориентированными инструментами поддерживаются следующие характеристики подхода:
  - *инкапсуляция,*
  - *полиморфизм,*
  - *наследование.*
- При этом в терминах объектов высокого уровня можно описать многие специальные проблемы, сократив общение между специалистом-прикладником и программистом.

Инкапсуляция

- возможность описать в виде класса данные и манипулирующий ими программный код. Можно представить себе объект как суперданные, которые содержат и данные, и присущую им функциональность (как один файл с БД в MS Access).
- возможность ограничить доступ к содержимому объектов, т.к. при этом оказывается закрытым доступ к внутреннему представлению данных. Вместо обращения к данным объекта можно вызвать один из методов объекта. Существенно то, что объект сам определяет, как ему реагировать на вызов того или иного метода - разные объекты могут по-разному реагировать на вызов одного и того же метода.

Полиморфизм и Наследование

- **Полиморфизм** - операция может выполняться над объектами более чем одного типа. Классический пример – графический редактор, который рисует различные геометрические фигуры и для каждого графического объекта определяет свой метод его рисования. В ООП функции используют одно и то же имя для аналогичных действий (например, для поворота графического изображения на экране) с различными типами объектов. При этом каждый объект «знает», как интерпретировать такую функцию в соответствии со своими особенностями.
- **Наследование** - создание новых объектов из уже существующих, которые не только унаследуют функции и свойства своих предшественников, но и могут добавить собственные.

Диаграммы UML

- UML выделяет девять типов диаграмм. При рассмотрении **статических** аспектов системы используются:
  - диаграммы классов;
  - диаграммы объектов;
  - диаграммы компонентов;
  - диаграммы развертывания.
- Для работы с **динамическими** частями системы применяются:
  - диаграммы прецедентов;
  - диаграммы последовательности;
  - диаграммы кооперации;
  - диаграммы состояний;
  - диаграммы деятельности.

Диаграммы UML

- Их деление на этапах проектирования следующее:
  - основные диаграммы
    - *прецедентов или вариантов использования (a)*,
    - *классов (bk),*
    - *объектов (bo),*
  - поведенческие диаграммы
    - *состояний (c)*,
    - *деятельности (d),*
  - диаграммы взаимодействия объектов системы
    - *последовательности (e),*
    - *кооперации (f),*
  - диаграммы физической реализации системы
    - *компонентов (j),*
    - *размещения (h)*.

Диаграммы вариантов использования (прецедентов)

- *Варианты использования (а)* (Айвар Якобсон, 1992 г.) - описание функциональности системы. Ключевыми элементами являются Актеры (**Actors**), взаимодействующие с системой с помощью вариантов использования (**Use Cases**). Актером является сущность, взаимодействующая с системой как извне, так и изнутри (человек, оборудование, другая ИС). Вариант использования описывает множество возможных последовательностей действий с участием Актеров:

1. Процесс построения моделей с использованием UML начинается с анализа предметной области, определения понятий и работ. На основании получаемых в процессе анализа *глаголов* и *существительных* строится диаграмма вариантов использования (в комментариях определяются объекты для Актеров и типы действий для вариантов использования).
2. Затем строятся *диаграммы классов (bк)*. Атрибуты и методы классов определяются ответом на вопрос «С чем имеешь дело?».
3. После этого создаются *диаграммы состояний (c)* и происходит возврат на новый виток спирали проектирования – к диаграммам вариантов использования, где происходит повторный анализ с учетом появившихся обратных связей.

Актеры модели "Предприятие по сборке и продаже компьютеров"

Актер	Краткое описание
Менеджер по работе с клиентами	Сотрудник, который общается с заказчиком и работает с заказом
Менеджер по снабжению	Сотрудник, который занимается закупкой необходимых комплектующих
Инженер по сборке настольных компьютеров	Сотрудник, который занимается сборкой настольных компьютеров
Инженер по сборке ноутбуков	Сотрудник, который занимается сборкой ноутбуков
Инженер по тестированию	Сотрудник, который занимается тестированием собранных компьютеров
Завскладом	Сотрудник, который заведует складом комплектующих

Функции системы:

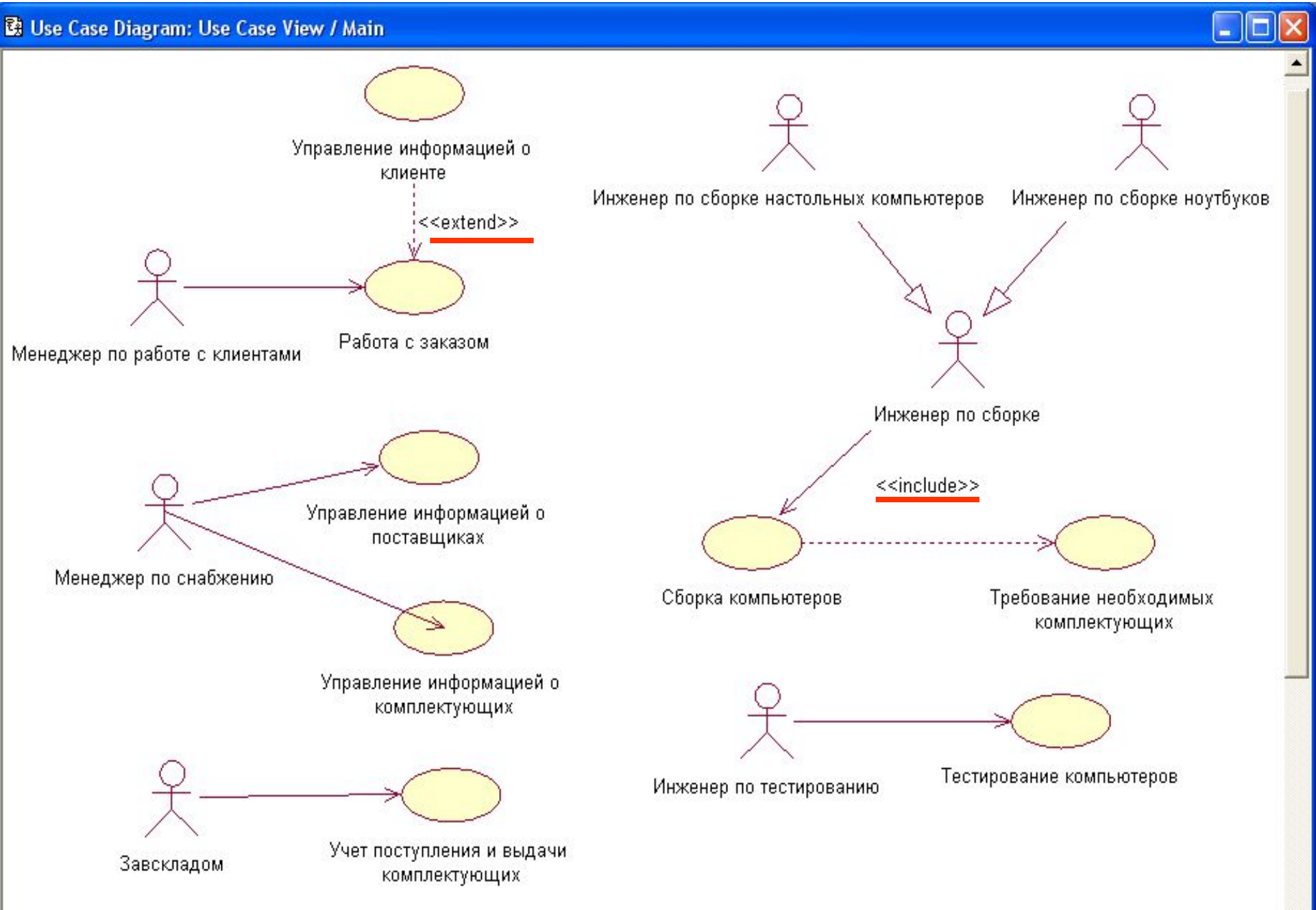
- **Менеджер по работе с клиентами** использует систему для оформления, редактирования заказов и управления информацией о клиентах предприятия;
- **Менеджер по снабжению** использует систему для просмотра перечня необходимых для закупки комплектующих и ведения информации о снабжении;
- **Инженер по сборке ПК** использует систему для просмотра нарядов на сборку ПК, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- **Инженер по сборке ноутбуков** использует систему для просмотра нарядов на сборку ноутбуков, для заказа комплектующих со склада и отметки о ходе выполнения работы;
- **Инженер по тестированию** использует систему для просмотра нарядов на тестирование собранной продукции и отметки о ходе выполнения работы;
- **Завскладом** использует систему для учета поступления и выдачи комплектующих.

Прецеденты

Прецедент	Краткое описание
Работа с заказом	Запускается менеджером.по работе с клиентами. Позволяет вносить, изменять, удалять или просматривать заказ.
Управление информацией о клиенте	Запускается менеджером по работе с клиентами. Позволяет добавлять, изменять или удалять клиентов, а также просматривать информацию о клиентах.
Управление информацией о поставщиках	Запускается менеджером по снабжению. Позволяет добавлять, изменять или удалять поставщиков, а также просматривать информацию о поставщиках.
Управление информацией о комплектующих	Запускается менеджером по снабжению. Позволяет просматривать информацию о комплектующих, производить анализ их расходования, прогнозировать необходимое их количество и делать заказ.
Сборка компьютеров	Запускается инженером по сборке. Позволяет просматривать наряды на сборку компьютеров и делать отметки о ходе выполнения работы.
Требование необходимых комплектующих	Запускается инженером по сборке. Предназначено для затребования необходимых комплектующие со склада.
Тестирование компьютеров	Запускается инженером по тестированию. Позволяет просмотреть список компьютеров, подлежащих тестированию и сделать отметки о ходе выполнения работ.
Учет поступления и выдачи комплектующих	Запускается завскладом. Позволяет вести учет поступления и выдачи запчастей и комплектующих.

На языке UML Актеры представляются в виде значков фигур, а варианты использования - в виде овалов.

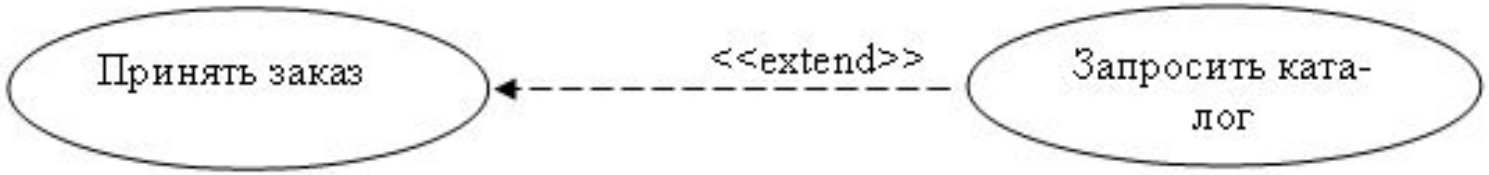
Главная диаграмма прецедентов



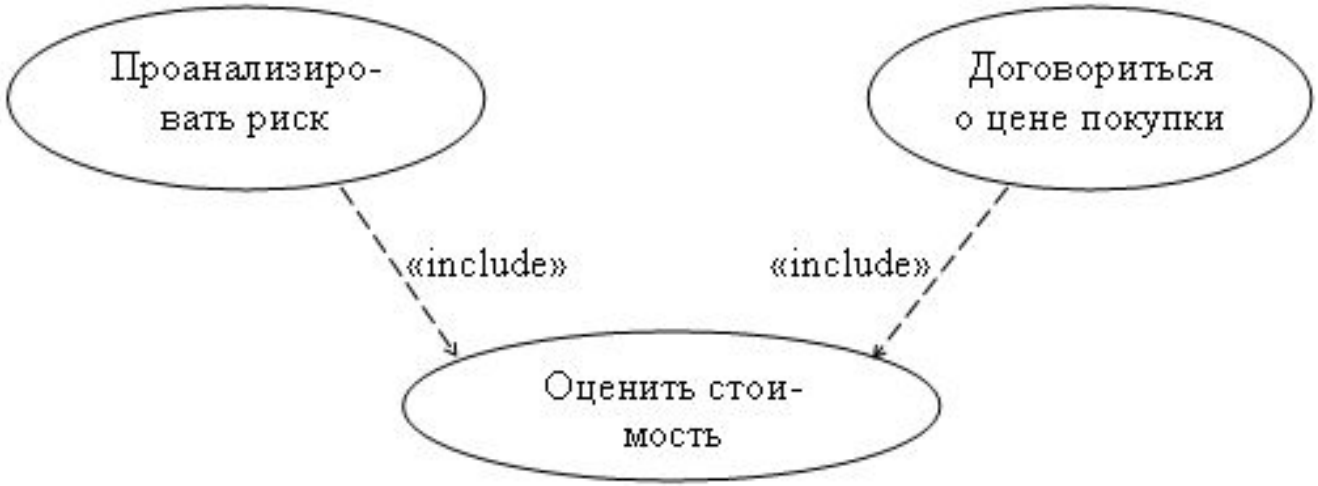
Отношения между прецедентами

Между собой **варианты использования (а)** не обмениваются сообщениями и могут находиться в отношениях *расширения* (extend), *включения* (include) и *обобщения* (generalization).

В отношении *расширения* (extend), варианта использования актера КЛИЕНТ вносится дополнительная последовательность действий, начиная с указанной *точки расширения:*



В отношении *включения* (include) один вариант использования включается в базовый вариант использования, начиная с *точки включения*. Выполнение включения длится до полного его завершения. После этого продолжается выполнение базового варианта, начиная с операции, следующей за точкой включения.

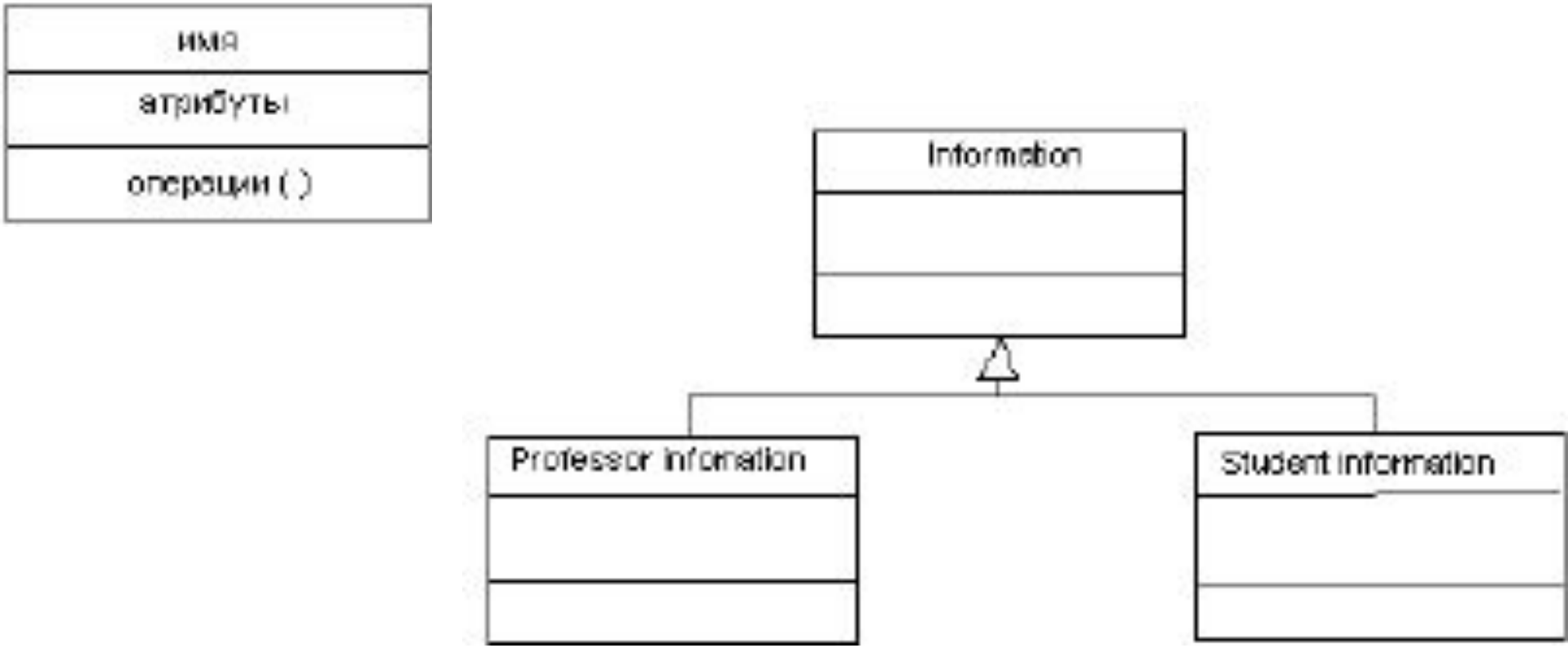



Установить предельные цены
Менеджер по продажам
Обновить счета
Проанализировать риск
Система учета
«использует»
Договориться о цене
«использует»
Оценка стоимости
Оптовый торговец
Заключить сделку
Действующее лицо
Продавец
«расширяет»
Вариант использования
Лимиты превышены

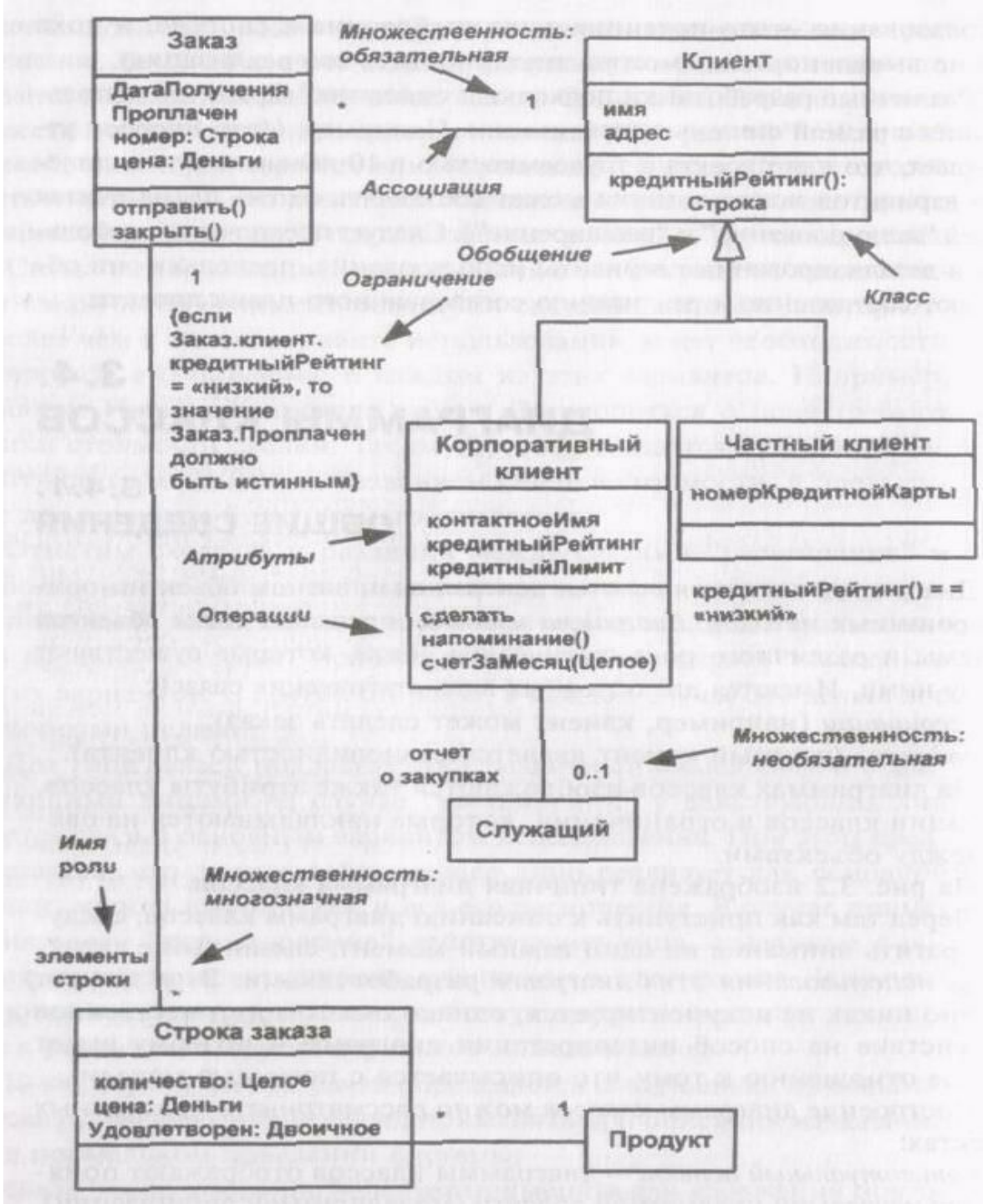
Диаграммы объектов и классов

- *Диаграмма объектов (bo)* показывает, какие существуют объекты и связи между ними, снимок потока событий
- *Диаграммы классов (bk)* показывают, какие существуют классы и связи между ними в структуре системы. Класс обозначается прямоугольником, в котором указываются имя класса, его атрибуты и операции.

Наследование на языке UML называют обобщением и изображают в виде стрелки от подкласса к суперклассу. Циклы запрещаются.



Допускается запись комментариев



Диаграммы взаимодействий (Interaction diagram)

- Диаграмма последовательностей (Sequence) - диаграмма акцентирующая внимание на временной упорядоченности сообщений. Представляет таблицу, объекты в которой располагаются вдоль оси X, а сообщения в порядке возрастания времени вдоль оси Y.
- Диаграмма кооперации (Collaboration) – диаграмма с акцентами на структурной организации объектов, принимающих и отправляющих сообщения. Представляет ориентированный граф с объектами в качестве вершин и сообщениями в

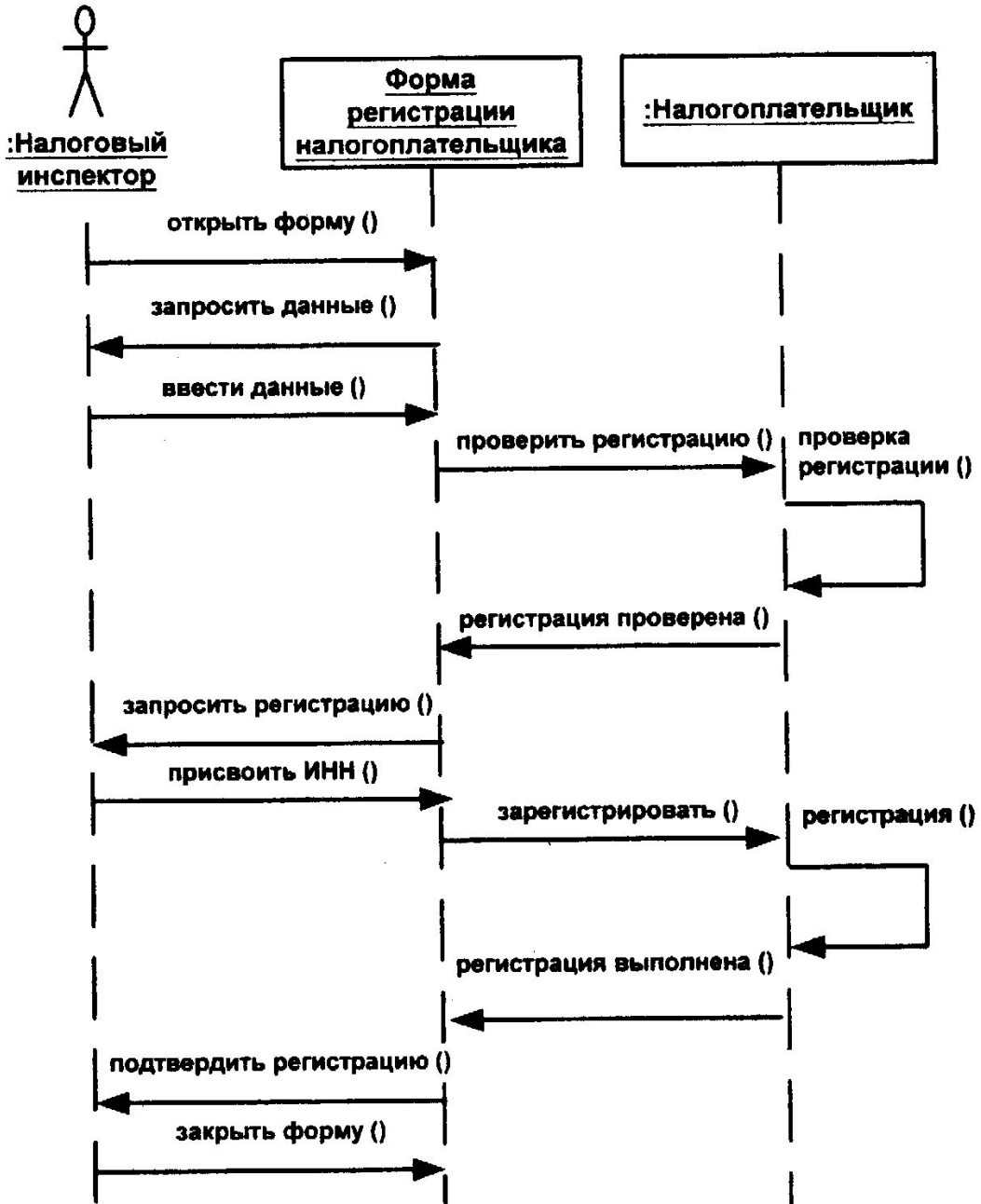
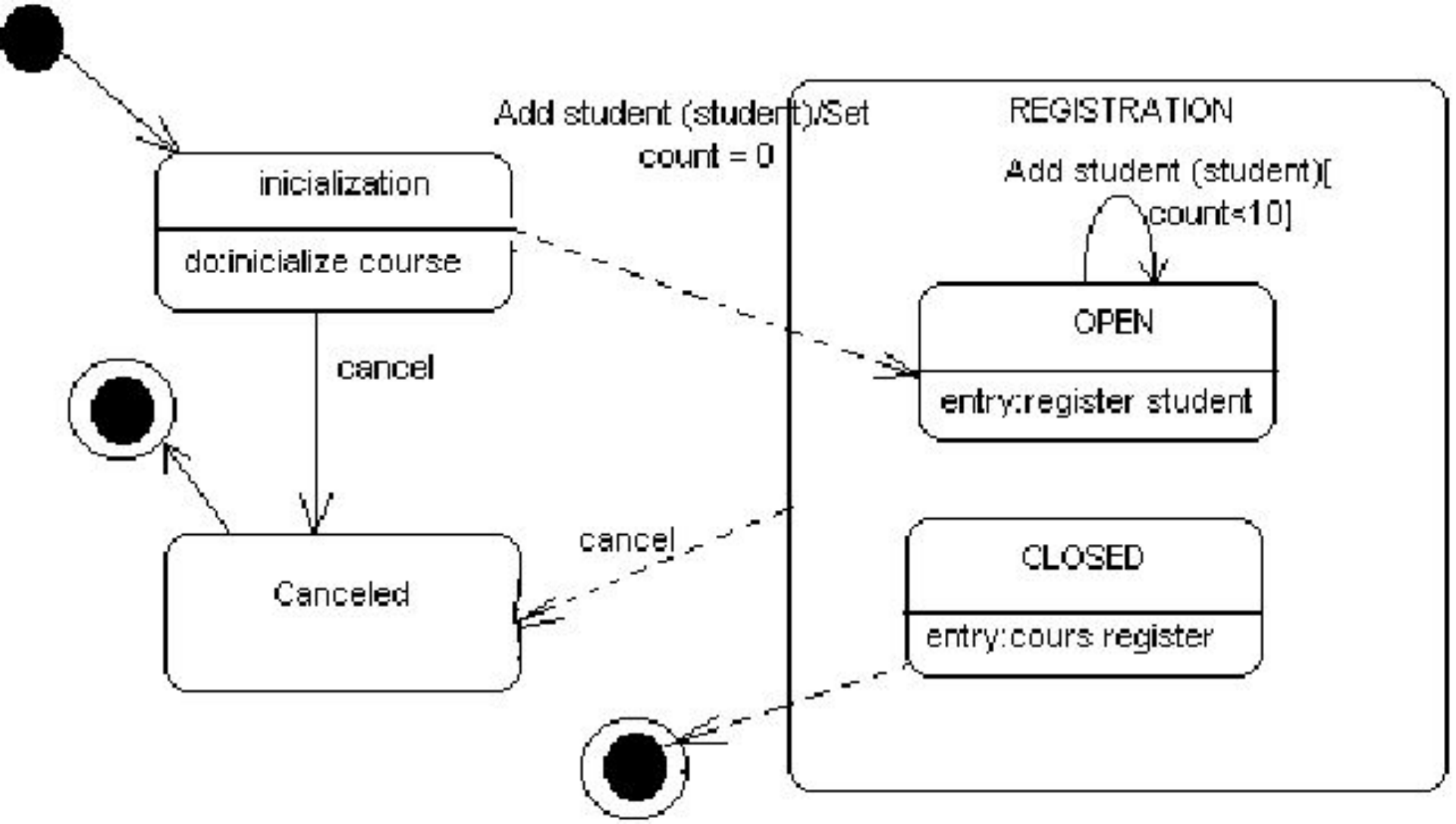


Диаграмма состояний

Показывает состояния класса; события, которые влекут переход из одного состояния в другое; действия, которые происходят при изменении состояния. Начальное состояние обязательно присутствует на диаграмме и только одно, присутствие конечного состояния не обязательно и может быть несколько конечных состояний (диаграмма



Диаграммы деятельности и компонентов

- Диаграмма деятельности - блок-схема, которая описывает последовательность выполнения операций во времени.
- Диаграмма компонентов - изображено множество компонентов и зависимости между ними. Компонент - это физически заменяемая часть системы, совместимая с одним набором интерфейсов и обеспечивающая реализацию какого-либо другого интерфейса. Компонент изображается в виде прямоугольника с вкладками. К имени компонента обычно добавляется расширение имени файла. Отношения между компонентами изображаются стрелкой, выходящей из зависимого модуля

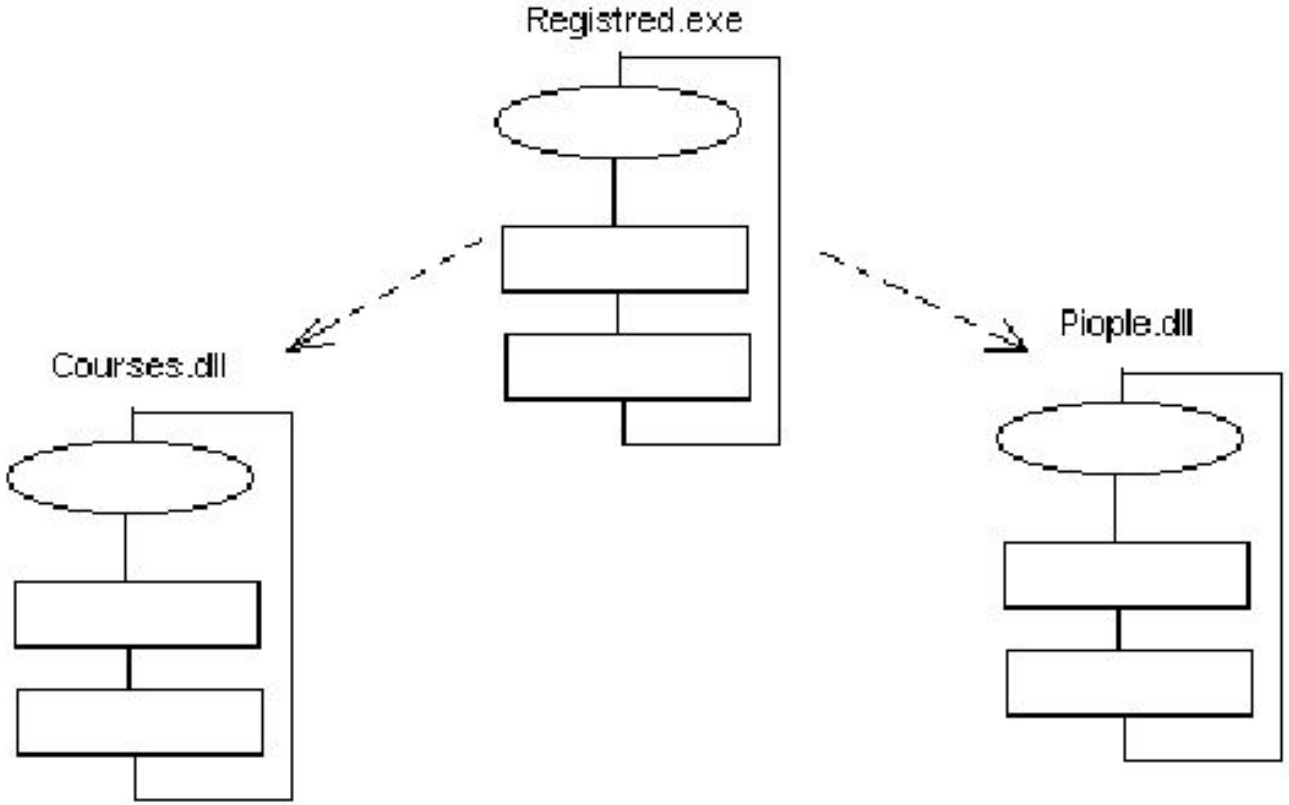
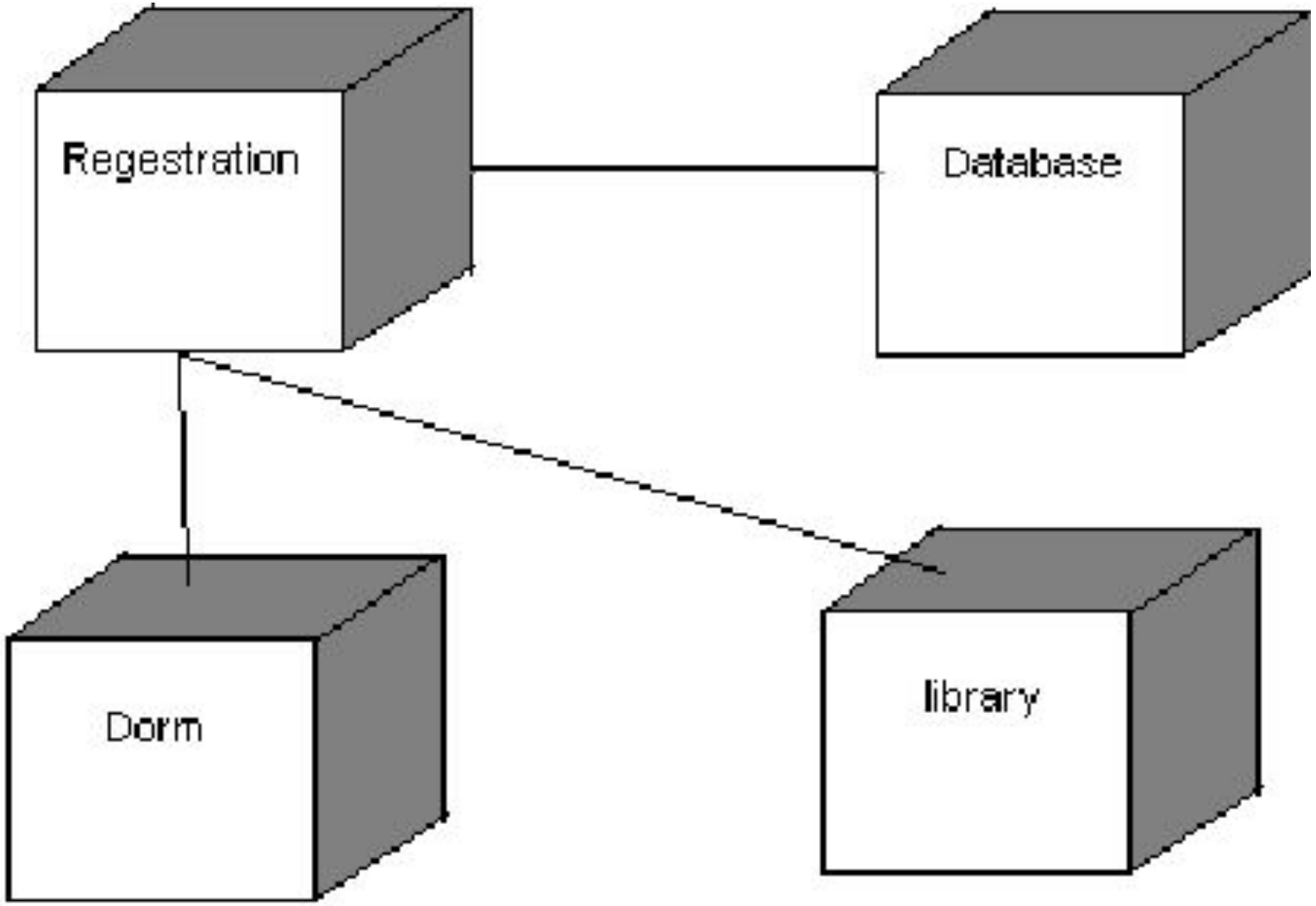


Диаграмма развертывания (размещения)

- Представлена конфигурация обрабатывающих узлов и размещенные на них компоненты.



Этапы создания клиент-серверной БД

Размер БД MS Access не должен превышать 2 Гб. В параметрах БД можно указать автоматическое сжатие БД. Если это действие не эффективно, то можно разделить БД на две - с таблицами и запросами, и, отдельно - с интерфейсом. Так можно создать файл-серверную БД.

Если таблицы и запросы размещены на SQL Server, то средствами MS Access можно создать к ней интерфейс. Во первых, готовая БД MS Access может быть экспортирована на сервер (будет создан проект БД – файл *.adp). Во вторых, можно создать проект, привязавшись к таблицам БД (при этом будет невозможно изменить их структуру).

Создание проекта – самый удобный способ миграции на SQL Server. При этом размер одной только таблицы на сервере может достигать 1 Гб. Используя проект, в среде MS Access можно создавать родные для SQL Server представления, хранимые (сохраненные) процедуры, пользовательские процедуры и триггеры.

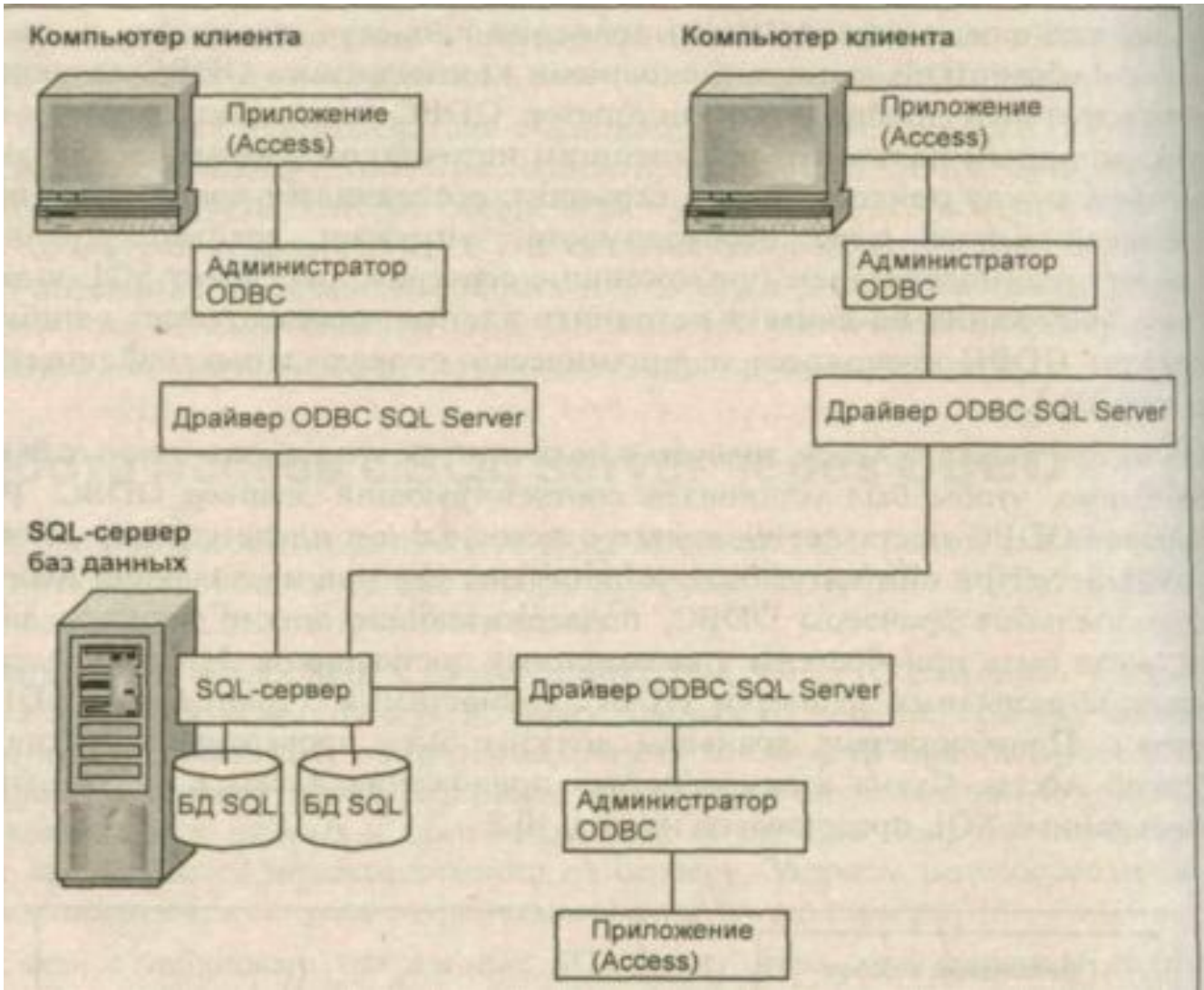
Если средств MS Access недостаточно для автоматизации процессов предприятия, то можно использовать MS Visual Studio для создания интерфейса (Windows Form, Web Form) к серверной БД.

Работа Access с данными на SQL-сервере

ПО архитектуры "клиент-сервер" состоит из двух частей: ПО сервера и ПО клиента. Клиент выполняется на ПК пользователя и посылает запросы серверу. Основная обработка данных производится **сервером БД**, а на ПК возвращаются результаты запроса.

СУБД с архитектурой "клиент-сервер" может включать собственную клиентскую программу. В качестве клиентов сервера БД могут использоваться другие СУБД.

Для взаимодействия приложения MS Access с БД на сервере необходимо создать источник данных на основе интерфейса **ODBC** или на основе интерфейса **OLE DB**.



Преобразование БД MSAccess в формат MS SQL Server

Такое преобразование представляет собой перенос некоторых или всех объектов базы данных из базы данных MS Access (.mdb) в новую или существующую базу данных MS SQL Server или новый проект MS Access (.adp).

Мастер преобразует базу данных MS Access в новую или существующую БД MS SQL Server или в новый проект MS Access путем переноса данных и определений данных в формат MS SQL Server и переноса объектов БД в структуру новой БД. Мастер преобразования в формат MS SQL Server можно использовать тремя способами:

1. Преобразовать только данные или определения данных из формата БД MS Access в формат БД MS SQL Server (операции импорта-экспорта).
2. Преобразовать все объекты БД MS Access в формат проекта MS Access, что позволит создать приложение типа клиент-сервер. Этот подход требует некоторых дополнительных изменений в приложениях и изменений в программах и в сложных запросах.
3. Создать клиентскую БД MS Access для серверной БД MS SQL Server, что позволит создать приложение типа клиент-сервер. Этот подход требует только небольших изменений в приложениях, поскольку программы будут по-прежнему использовать ядро БД MS Jet.

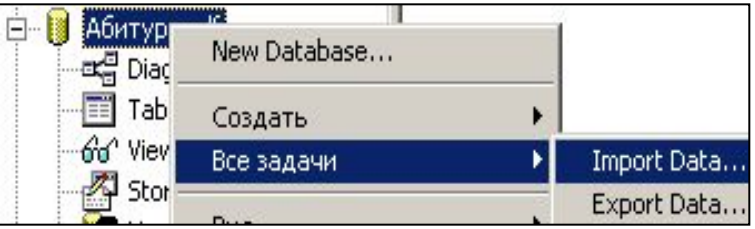
Преобразование БД MS Access в БД MS SQL Server

Перенос объектов из БД Access (.mdb) в новую или существующую БД SQL Server (.mdf) или новый проект Access (.adp):

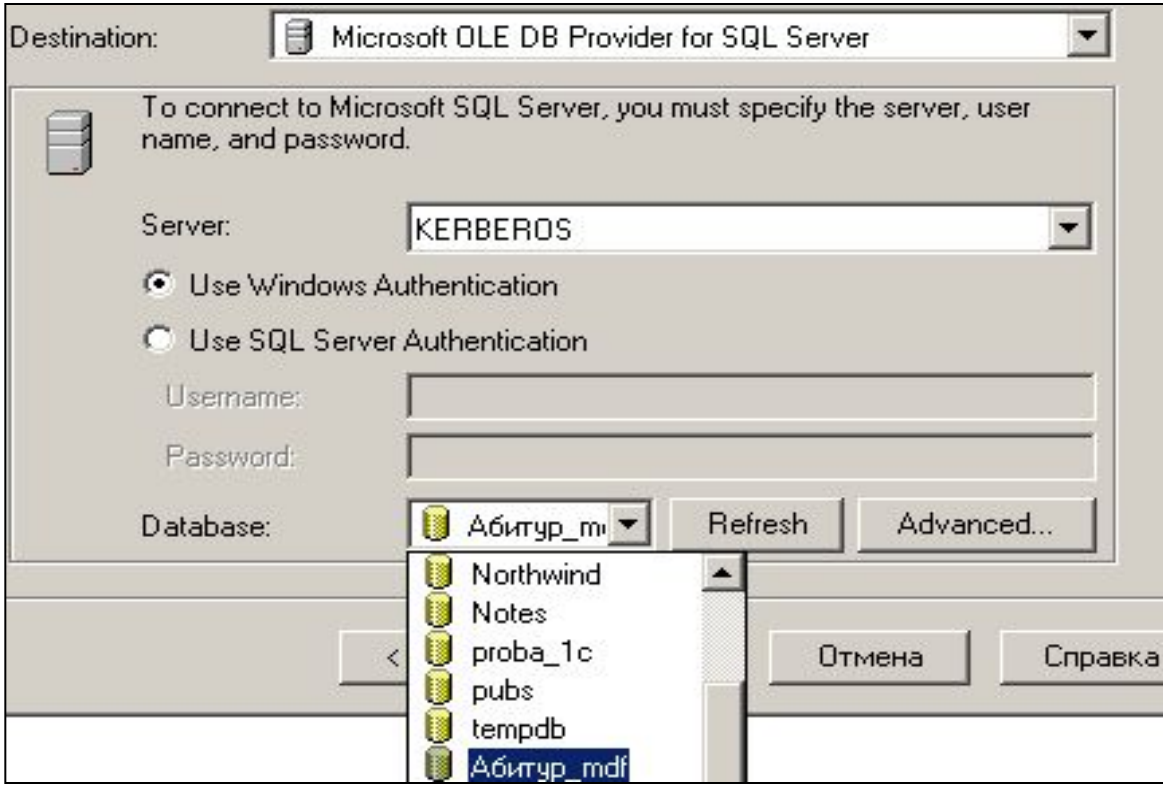
1. **Экспортировать БД из Access или импортировать в SQL Server. Экспортированная таблица не сохраняет определение ключа.**
2. **Преобразовать все объекты БД Access в формат проекта Access, что позволит создать приложение типа клиент/сервер (Сервис – Служебные программы – Преобразовать в формат SQL Server). Проект Access - файл Access, имеющий подключение к БД SQL Server. Файл проекта не содержит таблицы и представления (запросы), он содержит программные или HTML-объекты БД: формы, отчеты, имена и местоположения страниц доступа к данным, макросы и модули. У пользователя имеется возможность добавлять и изменять данные, создавать и изменять таблицы, представления, схемы БД и сохраненные процедуры на SQL Server. Этот подход требует дополнительных изменений в приложениях, программах и сложных запросах.**
3. **Создать клиентскую БД Access для серверной БД SQL Server, что позволит создать приложение типа клиент/сервер. Для этого надо связать БД Access с таблицами БД SQL Server (Файл - Внешние данные - Связь с таблицами). Данные остаются в формате источника, т.е. *Microsoft Jet* (JET = Join Engine Technology). При этом таблицы сервера и локальные таблицы отображаются в окне БД Access. Они могут быть использованы при создании запросов, форм, отчетов привычными диалоговыми средствами Access. Связанные таблицы можно просмотреть и в режиме конструктора, однако изменения структуры таблицы невозможны, хотя допускается изменение значений свойств, управляющих внешним видом полей в Access (формат поля, число десятичных знаков, маска ввода). Для разрыва связи с таблицей на сервере достаточно удалить ее в БД Access. Таблица на сервере удалена не будет.**

Варианты преобразования БД

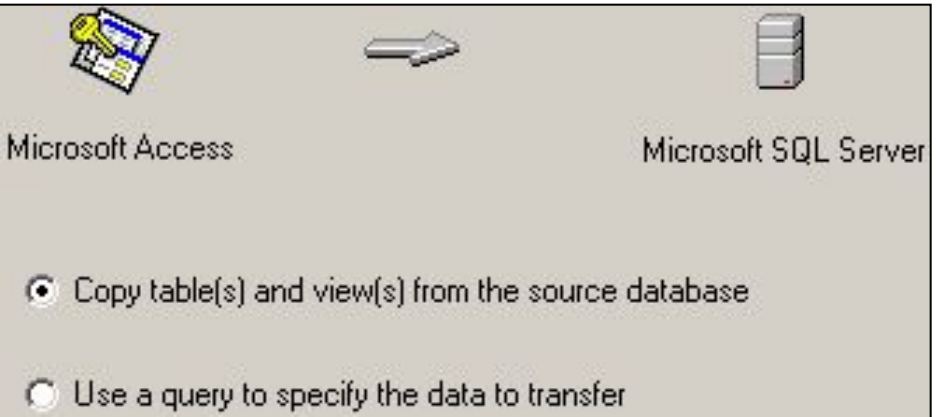
1. В первом случае надо на сервере создать новую БД и выполнить импорт данных из БД Access.



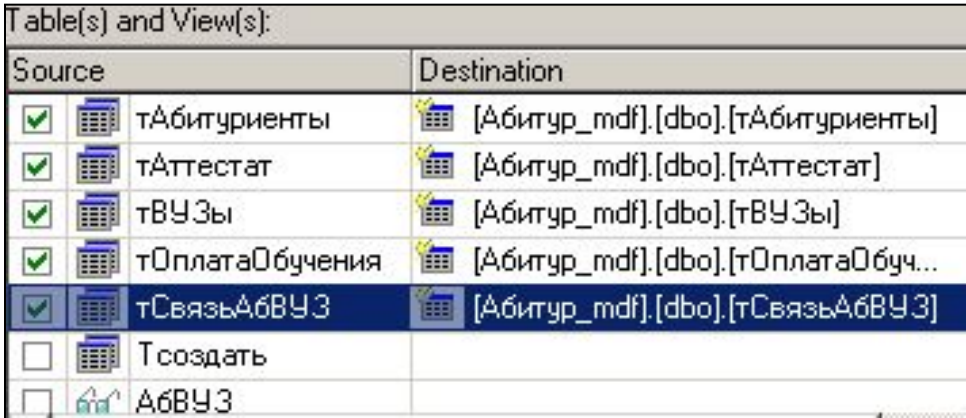
2. В списке драйверов выбрать Access и указать путь к файлу БД на ПК, затем указать БД на сервере для переноса в нее данных из БД ПК.



3. В следующем окне подтвердить копирование из БД таблиц и запросов



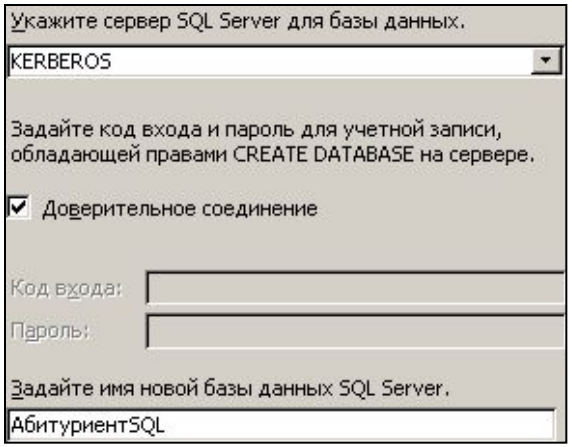
4. В следующем окне перечислить необходимые таблицы и запросы



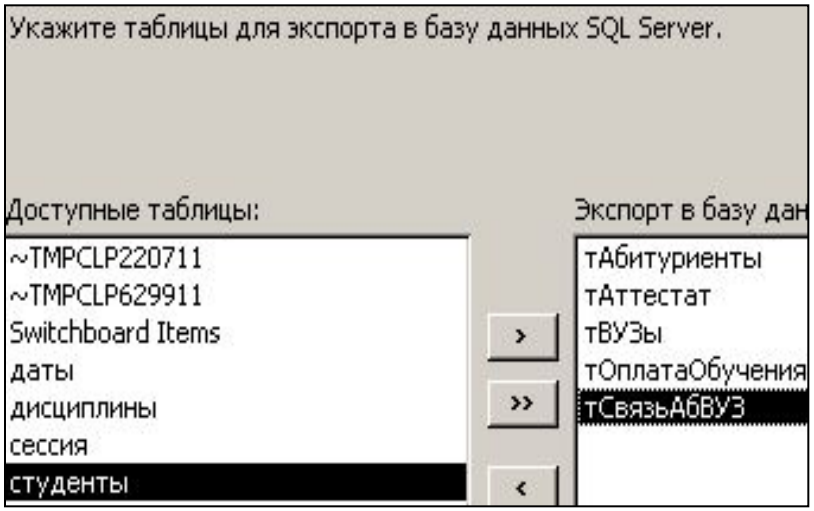
Создание проекта Access

БД открыть монопольно. Выполнить команду Сервис– Служебные программы – Мастер преобразования в формат SQL Server –

Создать БД

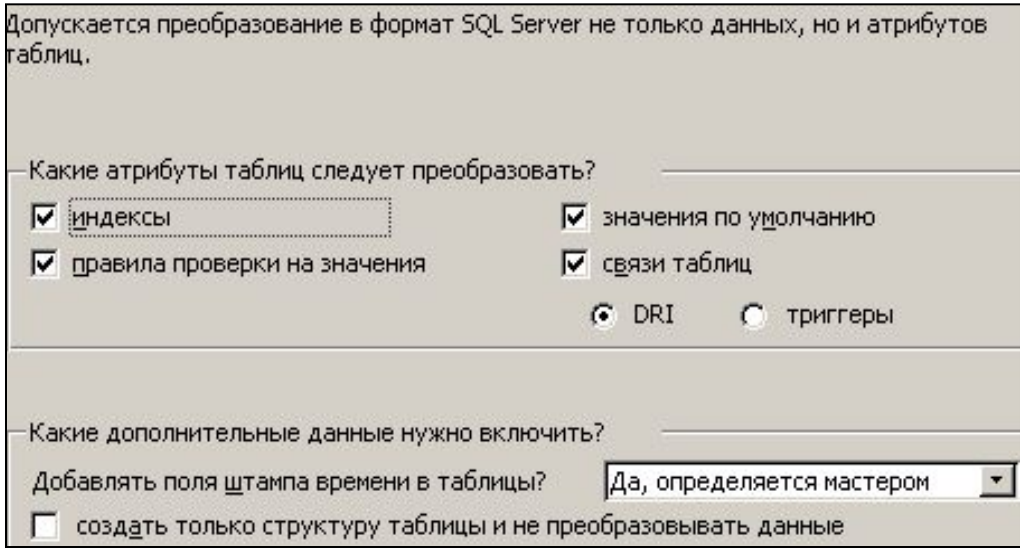


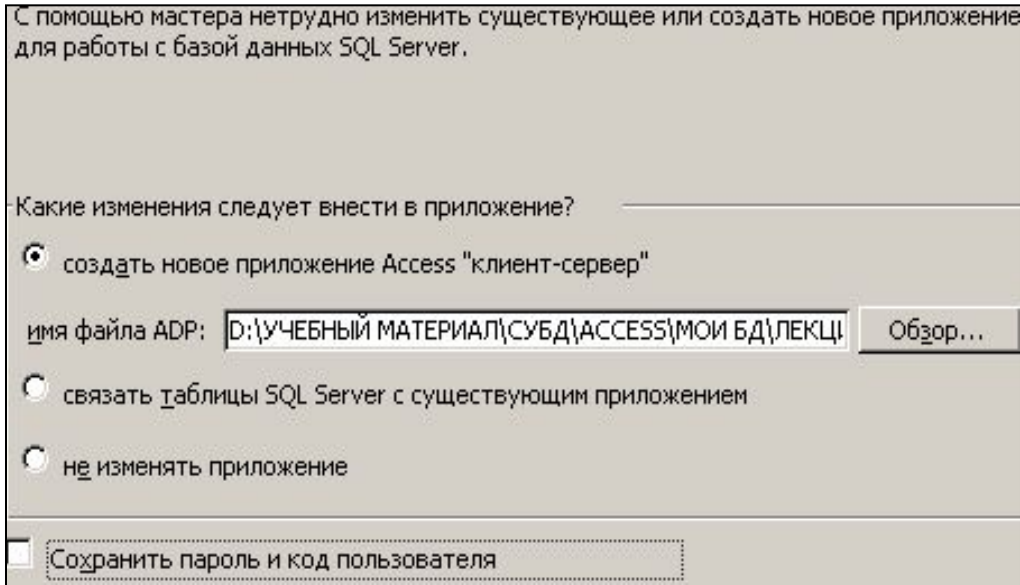
1. Доверительное соединение - имя и пароль WinNT (**sa** без пароля).



4. Приложение создается на ПК, БД экспортируется на SQL Server.

3. Декларативная целостность данных (**DRI**) позволяет связывать главные и подчиненные таблицы.





Состав проекта (.adp)

На SQL Server создается БД АбитуриентSQL.mdf.

На ПК создается файл АбитуриентCS.adp с формами, отчетами, страницами доступа, макросами и модулями. Это позволяет работать с БД SQL Server из среды MS Access.

Соответствие объектов

БД Access	Проект Access
Таблица	Таблица
Запрос на выборку (Select Query)	Представление
Запрос на действие (Action Query)	Сохраненнная процедура
Схема данных	Диаграмма
Форма	Форма
Отчет	Отчет
Страница доступа	Страница доступа
Макрос	Макрос
Модуль	Модуль

Соответствие основных типов данных

БД Access	Проект Access
Текст	Varchar, Nvarchar, Char, Nchar
Memo	Text, Ntext
Числовой	Int, Smallint, Real, Float
Дата/время	DateTime, SmallDateTime
Денежный	Money, SmallMoney
Счетчик	Int (идентификация - да)
Логический	Bit
OLE	Image
Гиперссылка	Нет эквивалента

тАбитуриенты : таблица

	Имя столбца	Тип данных	Длина	Разрешить Null	Описание
🔑	КодАБ	int	4		
	Фамилия	nvarchar	50	✓	
	Имя	nvarchar	50	✓	
	Отчество	nvarchar	50	✓	
	Паспорт	nvarchar	50	✓	
	Индекс	int	4	✓	
	Город	nvarchar	50	✓	
	Адрес	nvarchar	50	✓	
	[Средний балл]	real	4	✓	
	Фото	nvarchar	50	✓	
	Примечания	ntext	16	✓	
	upsize_ts	timestamp	8	✓	

Связывание таблиц в проекте Access с помощью мастера связывания таблиц (ADP)

В проекте Microsoft Access (.adp), подключенном к БД MS SQL Server, мастер связывания таблиц позволяет связать одну или несколько таблиц в другой БД SQL, БД Microsoft Access и в других источниках данных OLE DB или источниках данных ODBC.

Если имеется локальная установка Microsoft SQL Server или Microsoft SQL Server 2000 Desktop Engine, мастер связывания таблиц позволяет связывать данные из следующих приложений:

1. БД Microsoft Access .mdb (все версии),
2. проекты Microsoft Access .adp (все версии),
3. dBASE (версии 3, 4 и 5),
4. Paradox (версии 3.x, 4.x, 5.x и 7.x),
5. Microsoft Excel (версия 3.0 и более поздние),
6. текстовые файлы с разделителями (использующие системный разделитель, установленный в окне **Язык и стандарты** панели управления Windows)
7. HTML.

Примечание. Без локальной установки SQL Server допускается только связывание таблиц SQL Server.

Защита БД Access от несанкционированного доступа

Способы защиты БД Access:

- защита паролем;
- защита на уровне пользователей;
- Защита программного кода VBA.

Администрирование БД, защищенных с помощью пароля, сводится к изменению пароля защиты (когда это необходимо).

Если для проекта используется файловый сервер, то каталогу с БД пользователям должен быть предоставлен **доступ на чтение**, а программный код должен быть защищен паролем. Локальная копия файла проекта должна быть защищена в файловой системе ПК.

В проектах на SQL Server используется система защиты БД SQL Server.

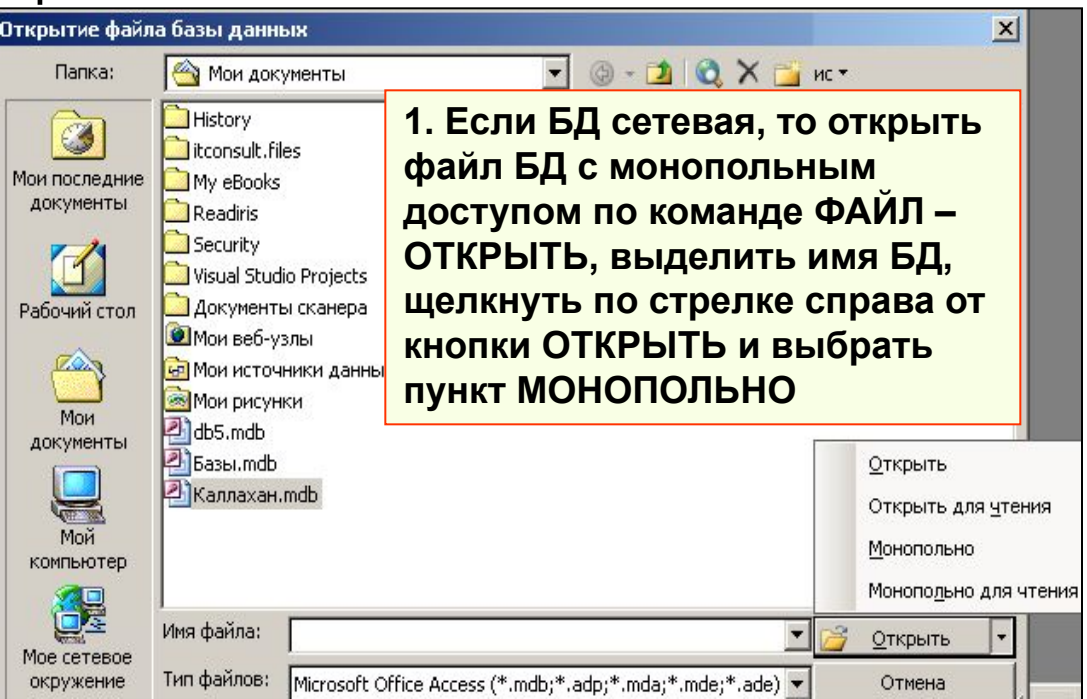
Можно преобразовать файл с БД в формат MDE или ADE (*без права просмотра программного кода*) или использовать параметры запуска для ограничения доступа к программам VBA и некоторым параметрам среды Access.

Можно также скрыть некоторые объекты от пользователей с помощью диалогового окна Параметры (Options).

Способы защиты БД MS Access

1. защита БД (mdb-файла) паролем и шифрованием;
2. сокрытием объектов БД с помощью настройки параметров запуска;
3. защита паролем программы на языке VBA;
4. защита кода VBA путем создания файла без исходного кода (MDE-файл для mdb-файла или ADE-файл для adp-файла проекта);
5. защита БД и ее объектов на уровне пользователей.

Для многопользовательских приложений рекомендуется следующая схема защиты: серверную часть БД защищают на уровне пользователей, в клиентской части настраивают параметры запуска и создают из нее MDE-файл.



1.1. Защита БД паролем

Создать предварительно копию БД.

Команда СЕРВИС – ЗАЩИТА – ЗАДАТЬ ПАРОЛЬ БД (*проверить клавишу CAPS LOCK*). Снять пароль можно после открытия БД по паролю (с монопольным доступом) по команде СЕРВИС - ЗАЩИТА – УДАЛИТЬ ПАРОЛЬ БД.

1. 2. Шифрование БД

При шифровании БД сжимается и делается недоступной для чтения служебных программ или текстовых редакторов.

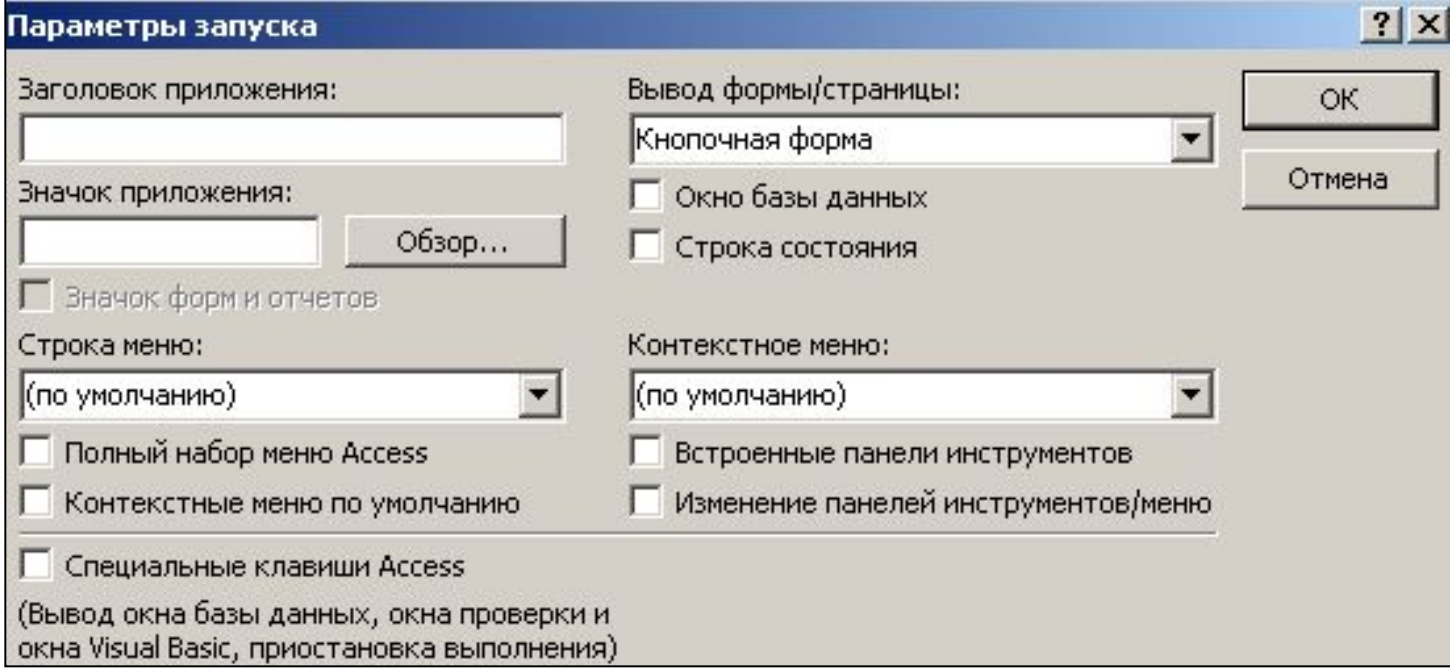
- **Чтобы зашифровать или дешифровать БД, нужно ее закрыть, выбрать пункт меню Сервис - Защита - Шифровать/дешифровать (Tools - Security - Encrypt/Decript Database). Указать шифруемый и результирующий файлы. Если БД защищена паролем, необходимо его указать при шифровании.**

Примечания:

- **В результате редактирования БД прежняя информация становится недоступной, занимая место в файле *.mdb. Поэтому необходимо периодически сжимать БД для оптимизации ее работы. Эта операция выполняется по команде СЕРВИС – СЛУЖЕБНЫЕ ПРОГРАММЫ – СЖАТЬ И ВОССТАНОВИТЬ БД.**
- **Можно настроить автоматическое сжатие БД при ее закрытии по команде СЕРВИС – ПАРАМЕТРЫ, вкладка ОБЩИЕ, флажок СЖИМАТЬ ПРИ ЗАКРЫТИИ.**
- **Если удалить все данные в таблицах, то после сжатия БД поля типа СЧЕТЧИК вновь будут заполняться значениями, начиная с 1.**
- **Для оптимизации работы БД необходимо также не использовать, по возможности, объекты типа OLE и MEMO, не использовать рисунки в качестве фона формы, сокращать длину текстовых полей, тип ДЛИННОЕ ЦЕЛОЕ менять на ЦЕЛОЕ. Значения, принятые по умолчанию для типов полей, можно менять по команде СЕРВИС – ПАРАМЕТРЫ, вкладка ТАБЛИЦЫ и ЗАПРОСЫ**

2. Сокрытие конструктора приложения

Предварительно создайте копию БД !



1. Команда СЕРВИС – ПАРАМЕТРЫ ЗАПУСКА. Убрать отметки на всех параметрах. В интерфейсе будет скрыт конструктор, стандартное и контекстное меню, панели инструментов MS Access. Заодно теперь отсутствует и возможность просмотра макросов, модулей и программного кода, встроенного на событиях форм и отчетов!

2. Параметры запуска и действия, которые определены в макросе с именем AutoExec или процедуре обработки события Open, можно обойти, удерживать клавишу SHIFT при открытии БД. Отключить клавишу SHIFT можно с помощью кода VBA процедуры Form_Load загружаемой формы, который задает для свойства AllowBypassKey БД значение False.

```
Private Sub Form_Load()
    Call SetBypassProperty
End Sub
```

```
Sub SetBypassProperty()                ‘ MSDN, топик AllowBypassKey Property
   Const DB_Boolean As Long = 1
   ChangeProperty "AllowBypassKey", DB_Boolean, False
End Sub

Function ChangeProperty(strPropName As String, varPropType As Variant, varPropValue As
Variant) As Integer
   Dim dbs As Object, prp As Variant
   Const conPropNotFoundError = 3270
   Set dbs = CurrentDb
On Error GoTo Change_Err
   dbs.Properties(strPropName) = varPropValue
   ChangeProperty = True
   Exit Function
Change_Err:
   If Err = conPropNotFoundError Then   ' Свойство не найдено
      Set prp = dbs.CreateProperty(strPropName, varPropType, varPropValue)
      dbs.Properties.Append prp
      Resume Next                      ‘ Продолжить с кода, вызвавшего ошибку
   Else                             ' Неизвестная ошибка
      ChangeProperty = False
      Exit Function
   End If
End Function
```

3. Защита кода VBA

Предварительно создайте копию БД ! При появлении ошибки в работе программного кода, откроется окно с сообщением об ошибке и редактор VBA. Необходимо поставить пароль на программный код. В редакторе VBA команда TOOLS – PROPERTIES, вкладка PROTECTIONS, установить флажок LOCK PROJECT FOR VIEWING, задать и продублировать пароль на код VBA (*проверить клавишу CAPS LOCK*).

4. Создание приложения без кода VBA (MDE-файл)

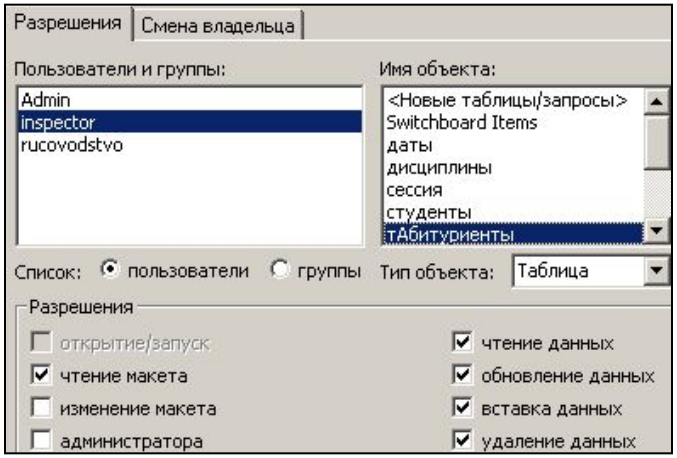
Предварительно создайте копию БД ! Закрыть код VBA на формах и отчетах можно преобразовав БД в формат *.mde по команде СЕРВИС – СЛУЖЕБНЫЕ ПРОГРАММЫ – СОЗДАТЬ MDE-ФАЙЛ. При этом модули будут скомпилированы и удалены, а БД будет сжата. Программы VBA будут по-прежнему выполняться, но их нельзя будет просматривать или изменять. Если БД ссылается на другую БД, необходимо сохранить все БД, входящие в цепочку ссылок, как MDE-файлы. Однако таблицы, запросы, страницы доступа к данным и макросы можно импортировать из MDE-файла и экспортировать в БД, не являющиеся MDE-файлами.

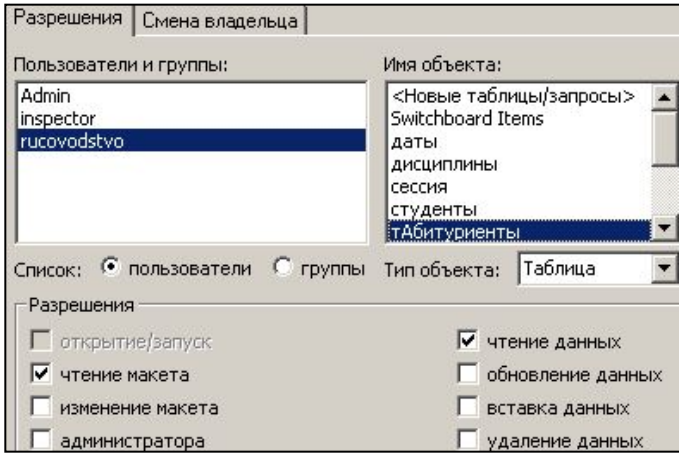
5. Защита БД и ее объектов на уровне пользователей.

Группа пользователей сети, совместно использующих одну или несколько баз данных *Access*, называется рабочей группой. При применении защиты на уровне пользователя каждый пользователь должен иметь свое имя и пароль. Администратор БД может снять пароль, а пользователь откроет приложение с пустым паролем и введет себе пароль. Информация о пользователях и их паролях хранится в зашифрованном виде в файле рабочей группы. Стандартный файл рабочей группы System.mdw создается при установке Access и хранится в папке (1049) с Office или в папке (1033) с личными настройками пользователя Windows.

Перед защитой БД нужно разблокировать код VBA.

Желательно создать новый файл рабочей группы по команде СЕРВИС – ЗАЩИТА – АДМИНИСТРАТОР РАБОЧИХ ГРУПП – СОЗДАТЬ. По умолчанию в файле есть группы Admins и Users и пользователь Admin, который входит в обе группы. Пользователю Admin необходимо задать пароль на вкладке СЕРВИС – ЗАЩИТА – ПОЛЬЗОВАТЕЛИ И ГРУППЫ – ИЗМЕНЕНИЕ ПАРОЛЯ, а на вкладке ПОЛЬЗОВАТЕЛИ создать пользователей, например, “inspector” и “rucovodstvo” (пароли они укажут сами при первом входе в БД). На вкладке СЕРВИС – ЗАЩИТА – РАЗРЕШЕНИЯ дать разрешения на доступ к объектам БД.





Пользователя Admin надо оставить только в группе Users, введя вместо него в группу Admins нового пользователя.

Установить защиту можно с помощью МАСТЕРА ЗАЩИТЫ. Открытие защищенной МАСТЕРОМ БД выполняется только через помещенный на РАБОЧИЙ СТОЛ ярлык. При этом Access запускается в новой рабочей группе, созданной Мастером защиты, а открытие БД в стандартной рабочей группе невозможно.

Текущий пользователь:

MsgBox Application.Current User или MsgBox CurrentUser

Защита страниц доступа к данным

Страница доступа к данным представляет собой сочетание ярлыка, хранящегося в файле БД, и соответствующего файла на языке HTML, расположенного в файловой системе компьютера. Для защиты страницы доступа к данным и данных, к которым она обращается, требуется:

• защитить БД с помощью соответствующего файла рабочей группы (изменить подключение к странице можно в окне DATA LINK PROPERTIES - вкладка ALL, свойтво JET OLEDB:SYSTEM DATABASE, указать путь UNC к файлу рабочей группы в поле PROPERTY VALUE);

• запретить несанкционированный доступ со стороны программ *Visual Basic* с помощью параметров защиты *Microsoft Internet Explorer* и трехуровневого доступа к данным.

Разделение баз данных

Создать резервную копию базы данных !

Файл с таблицами можно отделить от файлов с остальными объектами и разместить на файловом сервере. При этом возможна коллективная обработка одних и тех же данных с разных рабочих мест. В файле оставшихся объектов БД будут находиться ссылки на файл с таблицами:

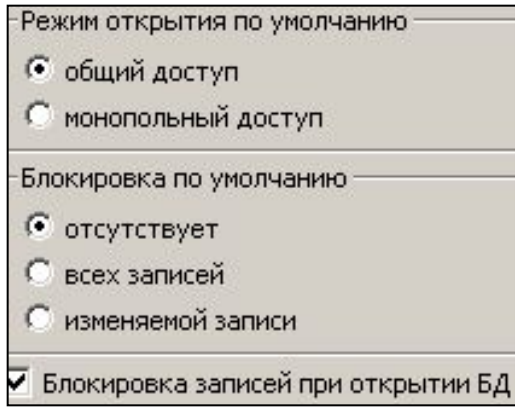
• СЕРВИС - СЛУЖЕБНЫЕ ПРОГРАММЫ - РАЗДЕЛЕНИЕ БД - РАЗДЕЛИТЬ;

• указать имя файла с таблицами, нажать кнопку РАЗДЕЛЕНИЕ;

В случае перемещения или переименования связанных таблиц, Диспетчер связанных таблиц помогает восстановить связи.

Коллективная работа

В Access существуют стандартные механизмы блокировки для поддержания общего доступа к данным в сети и разрешения конфликтов, возникающих при сохранении изменений, отслеживая состояние записи во время ее правки с помощью маркера в области выделения этой записи:



- — запись является текущей и к настоящему моменту не изменена;
- — запись изменена, но изменения не сохранены, и другие пользователи не видят изменений;
- — запись заблокирована другим пользователем, ее нельзя изменить.

При открытии БД в режиме общего доступа создается файл сведений о блокировке с расширением .LDB. Если к данным открыт общий доступ, то изменяемая запись не блокируется (“оптимистическая блокировка”) и Access предлагает:

– сохранить данный вариант записи, отказавшись от изменений, внесенных другим пользователем;

– скопировать свою, измененную запись в буфер обмена, приняв изменения, внесенные пользователем, чтобы ознакомиться с ними;

– отказаться от изменений в пользу изменений, внесенных другим пользователем.

При общем доступе с блокировкой всех записей (“пессимистическая блокировка”) Access блокирует всю форму или объект в режиме Таблица, включая базовые таблицы, на весь период их открытия.

Если организован общий доступ с блокировкой изменяемой записи, Access препятствует изменению этой записи другими пользователями сети. Пользователь видит маркер заблокированной записи и не может войти в нее. Однако данный режим не влияет на доступ через запросы.
Уровень блокировки задается установкой или сбросом флажка БЛОКИРОКА ЗАПИСЕЙ ПРИ ОТКРЫТИИ БД с последующей перезагрузкой Access. Чтобы установить блокировку страниц, флажок снимается. При создании форм, отчетов, запросов предусмотрены возможности задания параметров режима блокировки. Блокировка записей при открытии БД не относится к запросам на изменение и к программам, выполняющим операции с использованием инструкций SQL. Для того чтобы изменения, производимые одним пользователем, становились видны другим, через определенные интервалы времени предусмотрено автоматическое обновление содержания таблиц, форм и отчетов. Значение периода обновления задается из меню СЕРВИС - ПАРАМЕТРЫ, вкладка ДРУГИЕ, поле ПЕРИОД ОБНОВЛЕНИЯ.

Репликация

Организация независимой работы пользователей с одной БД без общего доступа, но с учетом всех изменений от разных пользователей, называется репликацией. Репликацией создаются копии файлов, между которыми организуется обмен обновляемыми данными или объектами (синхронизация реплик). Каждая реплика может также содержать локальные объекты. Главную роль в наборе реплик играет основная реплика. Только в основной реплике допускается изменение структуры объектов БД и схемы данных. Вводить, обновлять и удалять данные разрешено во всех репликах набора.

Репликации подлежит БД, открытая только одним пользователем и не имеющая пароля:

– Сервис - Репликация. Закрыть БД, при необходимости создать резервную копию;

– в окне РАЗМЕЩЕНИЕ НОВОЙ РЕПЛИКИ можно задать ее приоритет и запретить удаление записей из таблиц реплики;

– если еще не было репликации БД, она преобразуется в основную реплику и будет создана еще одна реплика с указанным именем. Если БД является репликой, будет создана только новая реплика с указанным именем.

Чтобы синхронизировать пару реплик:

– открыть любую реплику, меню СЕРВИС - РЕПЛИКАЦИЯ – СИНХРОНИЗАЦИЯ, задать путь и имя реплики, которую надо синхронизировать с текущей репликой;

– если текущую реплику нужно сделать основной, установить соответствующий флажок и нажать ОК.

В ходе синхронизации могут возникнуть конфликты - подтвердить предложение Access о его разрешении. Для проверки наличия конфликтов надо обратиться к меню СЕРВИС - РЕПЛИКАЦИЯ - УСТРАНИТЬ КОНФЛИКТЫ.

Управление репликами

Любую реплику, кроме основной, можно удалить. Если основная реплика повреждена, потеряна, переименована, удалена или перемещена, следует выбрать в наборе реплику, которую нужно сделать основной, синхронизировать ее со всеми репликами набора и обратиться к меню СЕРВИС - РЕПЛИКАЦИЯ - ВОССТАНОВИТЬ ОСНОВНУЮ РЕПЛИКУ.

В наборе не должно быть больше одной основной реплики. Чтобы одну из реплик в наборе назначить основной, следует убедиться что эта реплика и текущая основная реплика в данный момент не открыты другим пользователем, открыть ее и синхронизировать с текущей основной репликой с установкой флажка СДЕЛАТЬ ОСНОВНОЙ РЕПЛИКОЙ.

Обратное преобразование реплики в не реплицированную БД не выполняется. Для решения этой задачи следует создать новую БД и выполнить импорт в нее из реплики всех объектов, кроме таблиц. Затем создать запросы, выбирающие все записи из исходных таблиц и создающие на их основе новые таблицы. Воссоздать индексы и схему базы данных.

Реплику можно защищать на уровне пользователей, но без пароля.

- Для обработки данных на сервере можно использовать сохраненные процедуры, триггеры, определяемые пользователем функции и инструкции SQL SELECT. Кроме того, можно отсортировать данные на сервере перед их загрузкой.
- **Сохраненная процедура** представляет собой заранее откомпилированную последовательность инструкций SQL и необязательных управляющих инструкций, которая способна принимать параметры и обрабатывается в Microsoft SQL Server как одна программная единица. Сохраненные процедуры располагаются в БД Microsoft SQL Server. Поскольку сохраненные процедуры компилируются на сервере при создании, они выполняются быстрее, чем отдельные инструкции SQL, а выполнение сохраненных процедур на сервере позволяет использовать мощь процессора сервера.
- **Триггеры** — это особый тип сохраненных процедур, автоматически выполняемых при обновлении, вставке или удалении данных. Триггеры используются для проверки бизнес-правил и ограничений, более сложных, чем определяемые посредством проверяемых ограничений, например, можно запретить ввод в поле «Возраст» отрицательных значений или превышающих 110. В отличие от проверяемых ограничений, триггеры могут ссылаться на столбцы других таблиц. Например, триггер может отменить обновления, пытающиеся применить скидку (хранящуюся в таблице скидок) к книгам (хранящимся в таблице названий) стоимостью менее 10 рублей.
- **Определяемые пользователем функции** сочетают лучшие характеристики представлений и сохраненных процедур в одном запросе, который может использоваться как вложенный, поддерживает передачу параметров, сортировку и возвращение значений. Во многих случаях определяемые пользователем функции имеют ряд преимуществ по сравнению с сохраненными процедурами. Они позволяют возвращать таблицу данных или скалярное значение, позволяют скрывать от пользователя логику или подробности создания возвращаемых значений, а также упрощают синтаксис SQL.
- **Сортировка записей** на сервере выполняется с помощью сохраненной процедуры, определяемой пользователем функции или инструкции SQL, указанной в качестве значения свойства **Источник записей (RecordSource)** формы или отчета (которые сохраняются в проекте Microsoft Access, но выполняются на сервере). Сортировка

Различные типы запросов (ADP)

С помощью запросов выполняются операции по извлечению, созданию, изменению или удалению данных в базе данных. Существуют два основных типа запросов.

- Запросами на выборку называют запросы, выполняющие извлечение данных, предназначенных для отображения, с помощью инструкции SQL SELECT.
- Управляющими запросами называют сохраненные процедуры, выполняющие вставку, изменение или удаление данных с помощью инструкций SQL INSERT, UPDATE и DELETE.

▼ Представления

- ▶ Использование представлений
- ▶ Использование индексированных представлений и схемной привязки

▶ Сохраненные процедуры

▶ Определяемые пользователем функции

▼ Запросы с параметрами

- ▶ Запросы с параметрами: общие сведения
- ▶ Использование параметров
- ▶ Сравнение именованных и неименованных параметров
- ▶ Когда можно и когда нельзя использовать именованные параметры

▶ Запросы на изменение

▶ Инструкции SQL SELECT

Доступ к данным

При необходимости доступа к *источнику данных* (БД или другому репозитарию данных) используется установка соединения с ним через объект **Connection**.

1. Для обращения к **открытой БД Jet в Access** используйте свойство **Connection** объекта **Current Project** программы Access.

```
Dim conADOConnection As Connection
Set conADOConnection = CurrentProject.Connection
```

2. Установить подключение с **БД SQL Server** в проекте Access можно с помощью свойства BaseConnectionString объекта Current Project:

```
Dim conADO As New Connection
conADO.ConnectionString = CurrentProject.BaseConnectionString
```

Проекты Access могут взаимодействовать только с БД SQL Server и ни с какими другими средствами доступа OLE DB Provider.

3. Для создания объекта Connection **для других БД** объявите имя переменной для объекта и откройте подключение. Метод **Open** получает в качестве аргумента строку подключения, которая определяет используемое средство доступа OLE DB Provider и источник данных, с которым вы работаете. Или вы можете сначала задать свойства объекта Connection, соответствующего элементам строки подключения, после чего использовать метод Open (объект Connection для одной БД Jet):

Пример 1

```
Dim. conADOConnection As New Connection, strConnect As String
strConnect = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Data\Toys"
conADOConnection.Open strConnect
```

Пример 2

```
Dim conADOConnection As New Connection
With conADOConnection
    .Provider="Microsoft.Jet.OLEDB.4.0;"
    .Properties("Data Source")=”C:\Data\Toys"
    .Open
End With
```

Пример для SQL Server:

```
Dim conADOConnection As New Connection
Dim strConnect As String
strConnect = "Provider=SQLOLEDB; Data Source=Hecate; Initial Catalog = toys;
    User ID = sa; Password = ;"
conADOConnection.Open strConnect
```

Работа с объектами Recordset

- Объект Recordset — это контейнер, содержащий данные, полученные из источника данных. Как и положено контейнеру, один объект Recordset может содержать различные записи в разное время.
- После того как вы объявили переменную для объекта Recordset, в этот момент он существует только"виртуально". Для заполнения пустого контейнера реальными данными прибегните к одному из следующих приемов:
  - собственный метод Open объекта Recordset;
  - метод Execute объекта Command;
  - метод Execute объекта Connection.
- Простейший способ создания объекта Recordset — использование метода Open этого объекта:

```
Dim conman As New Connection
Dim rstMan As Recordset
Dim strSQL As String
    ...(код, используемый для создания объекта подключения conMan)
strSQL = "SELECT * FROM Toys"
Set rstMan.ActiveConnection = conman
rstMan.Open strSQL,, adOpenForwardOnly, adLockReadOnly, adCmdText
```

- Код связывает подключение с объектом Recordset с помощью его свойства ActiveConnection. Параметры, управляющие поведением объекта, указаны в качестве аргументов метода Open.

Создание объектов Recordset с помощью объекта Command

- Инструкции SELECT языка SQL подходят далеко не всегда. В приложениях клиент/сервер эффективность часто диктует необходимость создания объектов Recordset путем выполнения процедур (запросов). Если подобной процедуре требуются определенные параметры, в этой ситуации оказывается удобным использование объекта Command для создания объекта Recordset.
- Прежде всего настройте объект Command, присвоив его свойству ActiveConnection значение, соответствующее необходимому подключению. Затем вы должны определить параметры этого объекта как свойства. После этого вы наполняете объект Recordset записями, пользуясь результатами выполнения метода Execute объекта Command

```
Dim conTest As New Connection
Dim cmdTest As New Command
Dim rstTest As Recordset
Dim strSQL As String
...(код, создающий объект подключения contest  и определяющий строку strSQL)
With cmdTest                    ' Создание объекта Command:
    .ActiveConnection = contest
    .CommandText = strSQL
    .CommandType = adcmdText
End With
rstTest.CursorType = adOpenForwardOnly
rstTest.lockType = adLockReadOnly
Set rstTest = cmdTest.Execute()
```

Создание объектов Recordset с помощью объекта Connection

- Последний прием для создания объектов Recordset заключается в использовании метода Execute объекта Connection. Этот подход позволяет работать с сохраненными процедурами. Однако, если для выполнения процедур необходимы определенные параметры, вам придется включить эти параметры в инструкцию SQL

```
Dim conVert As New Connection
Dim rstVert As Recordset
Dim strSQL As String
...(код, создающий объект подключения conVert и определяющий строку strSQL)
rstVert.CursorType = adOpenForwardOnly
rstVert.lockType = adLockReadOnly
Set rstVert = conVert.Execute()
```

Создание подключения на лету

- Если подключение необходимо только для одного объекта Recordset, предварительное создание подключения не обеспечит никаких преимуществ.

```
Dim rstInPeace As New Recordset
Dim strSQL As String, strConnect As String
strSQL = "SELECT * FROM Bicycles"
rstInPeace.Open strSQL, strConnect, adOpenForwardOnly
```

Проверка записей

- Если при создании объекта Recordset с помощью VBA не возникло проблем, необходимо через значения свойств BOF (Beginning Of File — Метка начала файла) и EOF (End Of File — метка конца файла) проверить - набор содержит какие-то данные. Если значения обоих свойств равны True, набор записей пуст:

```
If rstY.B0F and rstY.EOF Then
    MsgBox "Записи в этом наборе отсутствуют!"
End If
```

Перемещение по наборам записей и нахождение определенных записей

- ADO позволяет перемещаться по набору записей. Для перемещения к первой записи в наборе используется метод **MoveFirst**, для перемещения к последней записи в наборе— метод **MoveLast**, для перемещения к следующей или предыдущей записи в наборе— метод **MoveNext** или **MovePrevious** соответственно.
- Метод **Move** позволяет перемещаться на определенное количество записей в наборе вперед или назад. Например, инструкция rstZ.Move -3 перемещает на три записи назад.
- Если вы захотите вернуться к определенной записи в дальнейшем, создайте для этой записи закладку. При работе с записью присвойте переменной значение свойства **Bookmark** объекта Recordset, как показано ниже:

```
varBookmark1 = rstA.Bookmark
```

- После этого вы можете вернуться к этой закладке в дальнейшем:

```
rstA.Bookmark - varBookinarkl
```

- Метод **Seek**, а также четыре метода Find (**FindFist, FindLast, FindNext и FindPrevious**) позволяют отследить определенную запись, **базируясь на ее содержимом**.
- Поскольку метод Seek обнаруживает целевую запись, используя индексный номер, он работает намного быстрее, чем методы Find, однако, прежде чем вы сможете его использовать, содержимое базы данных должно быть проиндексировано.

Добавление и удаление записей

- Используйте метод **AddNew** для занесения новой записи в набор или перехода к новой записи. Добавление новой записи в объект Recordset

rstIng.AddNew

- После этого можете приступать к заполнению полей записи. Однако можно указать значения полей сразу при создании новой записи:

```
With rstlng
  .AddNew Array ("Имя", "Возраст", "Пол"), Array ("Анна", 42, "Ж")
End With
```

- Как видно из этого примера, вы передаете аргументы методу AddNew в виде пары массивов, первый из которых содержит имена полей, а второй — их значения.
- Для удаления текущей записи предназначен метод **Delete** .

Чтение полей данных

- Укажите поле по имени или по его индексному номеру. Свойство **Value** является свойством по умолчанию, поэтому указывать его в коде не обязательно:

```
If rstYGate.Fields("Service visits").Value >10
   MsgBox "Привет!"
End If
strCurrentFieldData = rstYGate.Fields(3).Value
```

- Поскольку коллекция **Fields** является коллекцией по умолчанию объекта Recordset, не обязательно указывать ее по имени. Для обращения к полю укажите его имя, предварительно поставив восклицательный знак и заключив в квадратные скобки имена, содержащие пробелы:

```
rstYGate.Date = #5/15/2001#
With rstYGate
   .intltems =![Oil cans]
End With
```

Изменение данных поля

```
With rstBucket
.Fields(0).Value = "Love"
.MoveNext
End With
```

- Если не хотите перемещаться от текущей записи, можно внести изменения в БД, воспользовавшись методом **Update**:

```
With rstBucket
!Value =8.93
.Update
End With
```

- Как и метод AddNew, метод Update позволяет вам передать значения новых полей с помощью пары массивов, как показано ниже:

```
With rstBucket
.Update Array("Имя", "Звание", "Любимый напиток"),
Array("Лола", "Младший лейтенант", "Кофе")
End With
```

- Если не хотите перемещаться от текущей записи, можно внести изменения в БД, воспользовавшись методом **Update**:

```
With rstBucket
    !Value =8.93
    .Update
End With
```

- Как и метод AddNew, метод Update позволяет вам передать значения новых полей с помощью пары массивов, как показано ниже:

```
With rstBucket
    .Update Array("Имя", "Звание", "Любимый напиток"),
    Array("Лола", "Младший лейтенант", "Кофе")
End With
```

Повторение операций с несколькими записями

- Используйте цикл **Do** для проверки или операций с несколькими записями в наборе, как показано в следующем примере

```
‘ Циклическое перемещение между записями в наборе
With rstlnPeace
    Do Until .EOF
    Debug.Print.Fields(0)
    .MoveNext
    Loop
End With
SetrstlnPeace = Nothing
End Sub
```

Использование объекта Command. Хранимая процедура.

- В ADO объект **Command** представляет команду, такую как инструкция SQL или сохраненная процедура, которая может применяться к источнику данных. Хранимые процедуры — это запросы **и другие операции**, которые хранят в источнике данных (обычно с помощью SQL). В качестве хранимых процедур можно привести запросы Access. Хранимые процедуры выполняются быстрее и более надежны при использовании в сети, чем эквивалентные инструкции SQL.

Настройка объекта Command

- Для настройки объекта Command следует объявить соответствующую переменную и создать объект. После этого можете использовать свойства объекта для связи его с подключением, для определения команд, которые должны выполняться в форме инструкции SQL или имени хранимой процедуры, а также для определения типа операции. После этого можете использовать метод **Execute** объекта Command для действительного выполнения команды (через свойство CommandType=adCmdText можно передать инструкцию SQL**, заключенную в двойные кавычки**, источнику данных):

```
Dim consecrate As Connection
Dim cmdVBA As Command
Dim prmDate
Set cmdVBA = New Command
With cmdVBA
    .ActiveConnection = consecrate
    .CommandText = "UPDATE Bicycles SET OnSale = True WHERE Category = 4;"
    .CommandType = adCmdText
    .Execute
End With
```

Использование параметров команды

- Если выполняете запрос, сохраненный в базе данных Jet/Access, используйте значение adCmdTable свойства CommandType объекта Command, а не adCmdStoredProc, относящееся к SQL Server и другим серверам БД.
- Если для выполнения команды требуются ***входные параметры,*** вы должны определить индивидуальные объекты **Parameter**, добавляя их к коллекции Parameters объекта Command.
- Использование объекта Command для выполнения хранимой процедуры, а не инструкции SQL:

```
Dim consecrate As Connection
Dim cmdVBA As Command
Dim prmDate
Set cmdVBA = New Command
With cmdVBA
        .ActiveConnection = consecrate
        .CommandText = "qryDeleteOldRecords"
        .CommandType = adCmdStoredProc            ' для Jet - adCmdTable
End With
Set prmDate = New Parameter
With prmDate
        .Name = "Date"
        .Value = InputBox "Enter the cut-off date."
        .Type = adDate
        .Direction = adParamInput
End With
With cmdVBA
        .Parameters.Append  prmDate            ' Добавление параметра
        .Execute                          ‘ Выполнение команды
End With
```

Как избежать SQL

- Встроенный конструктор запросов Access — один из подобных инструментов. После того как вы создали и протестировали запрос Access, у вас есть два варианта выбора для добавления объекта Command в ваш код VBA для выполнения запросов. Объект Command может выполнять запрос как хранимую процедуру или же вы можете копировать SQL-код, полученный с помощью конструктора запросов, а затем, заключив в кавычки, вставить его в инструкцию VBA, которая определяет значение свойства CcnmandText объекта Command.
- Конструктор запросов Access не позволяет создавать запросы сложных типов, т.е. **подзапросы**.
- **Хранимые процедуры Access** работают только с БД Jet. Не забывайте о том, что разные БД используют разные версии SQL, поэтому вам придется вносить коррективы в полученный код.

Написание инструкций SELECT

- Простейшая форма инструкции SELECT получает все записи из одной таблицы (* означает, что набор записей содержит все поля из таблицы):

*SELECT * FROM Toys*

- Одна инструкция SELECT может работать с более чем одной таблицей (каждой записи из первой таблицы соответствуют все записи из второй):

*SELECT * FROM Toys, Clerks*

- Для того чтобы правильно связать две таблицы, выполните в инструкции SELECT *соединение. Внутреннее соединение* INNER JOIN, наиболее распространенное, создает запись в наборе, базируясь на соответствии записей в исходных таблицах:

SELECT Toy, Rep
From Toys INNER JOIN Reps On Toys.ID =Reps.ToyID

- Ключевое слово ON определяет поля, содержащие сравниваемые значения. Обычно имена полей после ключевого слова ON разделяются знаком равенства, что свидетельствует о том, что для выбранных записей значения, содержащиеся в указанных полях, должны совпадать.
- Для указания набора полей укажите их имена явным образом, как показано ниже:

SELECT Toy, InStock, OnOrder FROM Toylnventory

- Если имя поля содержит пробелы или знаки пунктуации, заключите его в квадратные скобки:

SELECT Toy, [List Price], [Sale Price] FROM Toylnventory

- Можете назначить другие имена полей (псевдонимы) в наборе записей, используя ключевое слово AS **для каждого поля**, которое вы решили переименовать:

SELECT Toy AS ToyName, InStock AS OnHand, OnOrder FROM Toylnventory

- Если вы извлекаете записи из нескольких таблиц и хотите выбрать поля с одинаковыми именами, укажите перед именем поля имя соответствующей таблицы:

SELECT Toylnventory.Name, Clerks.Name FROM Toylnventory, Clerks

- В инструкции SELECT вычисляемые поля определяются с помощью выражений, базирующихся на операторах и функциях VBA, например, цены на товары, которые получатся после снижения на 10%:

*SELECT Toy, (Price * 0.9) AS SalePrice FROM ToyInventory*

- При определении вычисляемого поля надо включить псевдоним нового поля в наборе записей. Скобки необязательны, однако они помогают подчеркнуть выражение, которое необходимо вычислять.
- **Пример**. Получить набор записей, в котором будут перечислены имена всех клерков прописными буквами, но при этом способ представления имен в самой БД не должен изменяться:

SELECT UCase(Name) AS [Clerk's name] FROM Clerks

- При использовании *агрегатныхфункций* SQL (Count, Avg, Sum, Min, Max, а также несколько статистических функций) инструкция SELECT позволяет получить набор записей, который будет содержать **только одно результирующее значение**, например, количество записей, содержащих определенное значение в поле:

SELECT Count(Recyclable) AS [Can Recycle] From Toys

или среднее значение всех полей:

SELECT Avg(Price) AS [Average Price] FROM Toys

- После этого вы можете передать значение поля этой записи переменной в вашем коде VBA для использования в вычислениях или отображения в форме:

intRecyclableCount = rstRecyclableToys![Can recycle]

Выбор записей с помощью предикатов DISTINCT, DISTINCTROW и TOP

- Используйте *предикаты* **DISTINCT, DISTINCTROW** и **TOP** в инструкции SELECT в качестве инструментов получения определенных наборов записей из баз данных. **Эти слова необходимо указывать сразу после инструкции SELECT**

Предикат	Использование	Пример
DISTINCT	Выбирает только одну запись, если база данных содержит несколько записей с идентичными данными в указанных полях	`SELECT DISTINCT Address FROM Members` Возвращает набор записей, содержащий только одну запись для каждого адреса, даже если в таблице `Members` содержится по несколько записей для каждого адреса
DISTINCTROW	Выбирает все уникальные записи, базируясь на значениях всех полей. Если две записи отличаются всего одним символом, они все равно будут включены в набор; если они полностью идентичны, в набор будет включена только одна из них	`SELECT DISTINCTROW Name, Address FROM Members` Возвращает набор записей, содержащий записи с полями `Name` и `Address`. Набор может содержать дублирующиеся записи, но только в том случае, если отличаются значения других полей

Настройка набора записей: задаем критерии

- Для ограничения набора записей только теми записями, которые удовлетворяют определенным критериям, добавьте к инструкции SELECT ключевое слово WHERE:

*SELECT * FROM Toys WHERE Price <= 20*

SELECT Customer, Date FROM Sales WHERE Date = #10/24/2000#

SELECT Name, Rank, CerealNumber FROM Kids WHERE Rank = 'Queen'

SELECT Name, Age, [Shoe Size] FROM Kids WHERE Age Between 3 And 6

- **Строковые значения заключаются в одинарные, а не двойные кавычки.**
- Ключевое слово WHERE указывается после ключевого слова FROM и содержит выражение, определяющее критерий, которому должны соответствовать записи, чтобы попасть в набор**.** Можете определять диапазоны с помощью конструкции **Between ... And**, которая в VBA отсутствует. Можно объединить выражения, используя логические операторы (And, Or и т.д):

*SELECT * FROM Toys WHERE Price > 20 And Category = 'Action Figures'*

- В коде VBA принято использовать одинарные кавычки для определения строки в инструкции SQL, которая целиком является строкой с точки зрения VBA, а значит, заключается в двойные кавычки:

strSQL = "SELECT Name FROM Kids WHERE Hates ='Brocolli ' "

cmdEr.CommandText = strSQL

- При использовании ключевого слова WHERE бывает необходимо, чтобы часть инструкции SQL основывалась на переменной; например, если вы выполняете запрос, базирующийся на данных, введенных пользователем **в текстовом поле формы**. Добавьте значение переменной к остальной части строки. **Если переменная представляет строковое значение, не забудьте заключить ее в одинарные кавычки**

StrSQL = "SELECT Name FROM Kids WHERE Hates = ' " & frmlnputForm.TextBoxl & " ' "

- *Если переменная представляет данные, а не строку, заключите ее между символами #.*

Группирование записей

- Ключевое слов **GROUP BY** позволяет ва объединять записи, содержащие одинаковые значения в указанных полях, преобразуя их в одну запись в полученном наборе записей. Обычно это ключевое слово используется в том случае, если необходим набор записей, содержащий общие сведения о данных. Например, сколько записей содержится в базе данных для каждого значения указанного поля:

```
SELECT Category, Count(Category) AS [Number of Items]
FROM Toys
GROUP BY Category
```

Дополнительный отбор с помощью ключевого слова HAVING

- Ключевое слово **HAVING** следует после ключевого слова GROUP BY и позволяет определить критерии для сгруппированных записей. Оно работает практически так же, как и ключевое слово WHERE; вы можете использовать его отдельно или в комбинации со словом WHERE для того, чтобы наложить на полученные записи дополнительные ограничения. В этом примере ключевое слово HAVING включает только те категории, которые содержат как минимум пять записей, удовлетворяющих критериям, определенным с помощью ключевого слова WHERE:

```
SELECT Category, Count(Category) As [Number of Items]
FROM Toys
WHERE Price > 100
GROUP BY Category
HAVING Count(Category) > 4
```

Сортировка с помощью ключевого слова ORDER BY

- Используйте оператор **ORDER BY** для сортировки записей, полученных с помощью инструкции SELECT, в соответствии со значениями одного или нескольких полей. Оператор ORDER BY указывается в конце инструкции, как показано ниже:

SELECT Toy, Price , InStock

FROM ToyInventory

ORDER BY Toy

- Можно проводить сортировку по значениям нескольких полей, указав эти поля в необходимом порядке сортировки:
- *SELECT Toy, Price*
- *FROM ToyInventory*
- *ORDER BY Price DESC, Toy*
- По умолчанию сортировка всегда проводится по возрастанию. Для явного указания порядка сортировки используйте ключевое слово **DESC** (descending— убывание) или **ASC** (ascending— возрастание), после которого необходимо указать имя соответствующего поля.

Выполнение групповых обновлений и удалений в SQL

- Инструкции **UPDATE** и **DELETE** позволяют изменять или удалять группу записей в источнике данных с помощью одной команды. Эти инструкции работают непосредственно с исходной БД (повышение цены на 10% для товаров определенной категории):

```
UPDATE Toys
SET Price = Price * 1.1
WHERE Category = ' Trains '
```

- Имя таблицы указывается сразу после слова UPDATE. После этого указывается оператор SET, с помощью которого вы определяете значение одного или нескольких полей в таблице. Необязательный оператор WHERE позволяет задать критерии, ограничивающие записи, к которым будут применяться изменения.
- Пример удаляет записи для всех игрушек, которые отсутствуют на складе и не были заказаны:

```
DELETE FROM Toys
WHERE InStock = 0 And OnOrder = 0
```

- Для удаления значений отдельных полей, а не целых записей, используйте инструкцию UPDATE вместе с оператором SET, определяющим значение поля равным **Null** .
- Инструкции UPDATE и DELETE приводят к необратимым изменениям в базе данных - обязательно создайте резервную копию базы данных.