**министерство образования и науки Российской Федерации**

Старооскольский технологический институт им. А.А. УГАРОВА

(филиал) федерального государственного автономного образовательного учреждения

высшего образования

«Национальный исследовательский технологический университет «МИСиС»

**ОСКОЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ**

УТВЕРЖДено

НМС опк

пРОТОКОЛ №1

ОТ «01» сентября 2016 г.

**ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ**

***Методические указания для студентов очной формы обучения для выполнения курсовой работы***

Специальность 09.02.04 Информационные системы (по отраслям)

Старый Оскол 2016г

|  |  |
| --- | --- |
| *Рассмотрены на заседании П(Ц)К 09.02.04*  *Протокол №*  *от «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2016г.*  *Председатель*  *Назарова О.И.* | *Методические указания составлены в соответствии с рабочей программой по дисциплине*  *Основы проектирования баз данных*  *Специальности*  *09.02.04 Информационные системы (по отраслям)*  *Зам .директора по М Р*  *к.п.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Степанова* |

***Составитель: Артюхина Д.Д., Коренькова Т.Н.***

***Рецензенты:***

*внутренний: Горюнова М.В.* - преподаватель ОПК СТИ НИТУ МИСиС

*внешний:* Анпилов А.Э. – инженер ООО «КМАЭМ»

**СОДЕРЖАНИЕ**

|  |  |  |
| --- | --- | --- |
| ВВЕДЕНИЕ | | 4 |
| 1 Организация проведения курсовой работы | | 5 |
|  | 1.1 Цель курсовой работы | 5 |
|  | 1.2 Задание на курсовую работу | 5 |
|  | 1.3 Этапы выполнения курсовой работы | 6 |
|  | 1.4 Содержание курсовой работы | 7 |
| 2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ | | 11 |
|  | 2.1 Построение концептуальной модели | 11 |
|  | 2.2 Этап инфологического проектирования: теоретические  сведения | 13 |
|  | 2.3 Этап логического проектирования: теоретические сведения | 17 |
|  | 2.4 Этап физического проектирования: теоретические сведения | 21 |
|  | 2.5 Создание соединение с базой данных | 31 |
|  | 2.6 Использование форм для ввода и редактирования данных | 34 |
|  | 2.7 Реализация запросов | 43 |
|  | 2.8 Реализация отчетов | 57 |
| СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ | | 71 |

**ВВЕДЕНИЕ**

Базы данных являются составной частью информационной системы (ИС), осуществляющей хранение и обработку данных в широком смысле, включая ввод и корректировку данных, удаление данных, выполнение различных запросов, арифметические и логические преобразования данных, составление отчетов, выдачу информации в различных формах на экране и в бумажном виде.

Проектирование баз данных представляет собой достаточно сложный процесс, основывающийся на многих научных дисциплинах таких, как системный анализ, концептуальное моделирование, проектирование баз данных, теории нормализации, проектировании, реляционной алгебры и языков запросов, и многих других, включенных в общую теорию информатики.

В предлагаемом учебном пособии рассматриваются вопросы концептуального моделирования, теории проектирования баз данных на основе проектирования, операций реляционной алгебры и простейших запросов.

По структуре материал можно разбить на две части. В первой части дается теоретическое введение, поскольку перечисленные вопросы разбросаны по различным литературным источникам, либо изложены в труднодоступной для студентов форме. Вторая часть посвящена непосредственно проектированию баз данных. Представленный уровень достаточный для понимания основных принципов проектирования баз данных и приобретения соответствующих навыков.

**1 Организация проведения курсовой работы**

**1.1 Цель курсовой работы**

Целью курсовой работы является освоение методов проектирования баз данных и работы с базами данных в среде конкретной СУБД.

**1.2 Задание на курсовую работу**

Задание содержит название конкретной предметной области, для которой необходимо:

- сформулировать цель проектирования базы данных;

- описать возможного пользователя базы данных;

- определить круг запросов и задач, которые предполагается решать с использованием созданной базы данных;

- построить концептуальную модель;

- сформулировать требования к базе данных;

- построить реляционную модель и выполнить её нормализацию;

- осуществить выбор СУБД и технических средств;

- создать базу данных в среде выбранной СУБД;

- рассчитать объём базы данных и/или привести формулы для расчёта объёма;

- разработать приложение для реализации запросов и решения задач;

- оценить базу данных с точки зрения возможностей её дальнейшего развития

Результатом выполнения курсовой работы должны быть законченное программное приложение и оформленная курсовая работа.

Разрабатываемое программное приложение должно:

1. заносить информацию в созданную базу данных;
2. выполнять необходимые действия по модификации и удалению информации в базе данных; при этом все операции по занесению, модификации и удалению данных должны выполняться в терминах предметной области, а не базы данных;
3. поддерживать целостность базы данных, не допуская появления некорректных данных;
4. выполнять все действия над базой данных в рамках транзакций;
5. содержать достаточное количество данных, позволяющих показать результаты выполнения запросов;
6. выполнять запросы из текста технического задания;
7. контролировать все вводимые данные;

**1.4 Содержание курсовой работы**

Представляемая курсовая работа должна содержать:

**Введение**

1. **Теоретическая часть** 
   1. **Анализ предметной области**
      1. Состав информационной системы
      2. Перечень функций и задач системы

Перечень функций и задач системы в целом и каждой подсистемы в отдельности

* + 1. Описание работы подсистем

Подробное описание работы каждой подсистемы, взаимоотношение их между собой, описание отдельных сценариев работ;

* + 1. Диаграммы работ и диаграммы потоков данных

Диаграммы работ и диаграммы потоков данных для всей информационной системы в целом и для входящих в нее подсистем.

**2. Практическая часть**

**2.1. Построение инфологической модели базы данных**

Для построения инфологической модели предметной области рекомендуется придерживаться следующей последовательности действий.

2.1.1. Каталог задач и запросов предметной области

Описать предметную область с указанием всех задач и запросов предметной области (описание должно быть кратким, но достаточным для принятия решений по проекту базы данных).

2.1.2. Состав и содержание информации предметной области

Определить состав и содержание информации, используемой в данной предметной области, а именно:

- составить перечень задач и запросов, указать входные и выходные данные;

- построить схему входной и выходной информации;

- определить возможные будущие изменения информационных потребностей пользователей;

- установить уточнённые требования к информационным потребностям пользователей.

2.1.3. Основные информационные объекты-сущности предметной области

Выявить сущности и задачи сущностей, а именно:

- определить атрибуты каждой сущности и требования к ним;

- определить ключ каждой сущности;

- разработать, если необходимо, классификаторы и кодификаторы сущностей;

- определить требования к сущностям

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название сущности | Атрибуты сущности | Тип данных атрибутов | Задачи сущности | Требования к сущности | Ограничение доступа | Классификация сущности |
|  |  |  |  |  |  |  |
|  |  |

2.1.4. Отношения между сущностями

Выявить связи между сущностями, а именно:

- структурные связи для выявления классов и подклассов сущностей;

- функциональные связи типа 1:1, 1: m, n:m, n-арные;

- если необходимо, определить атрибуты связей.

2.1.5. ER-диаграмма модели базы данных

2.1.6. Анализ модели с учётом информационных потребностей пользователей

По результатам этапа концептуального проектирования необходимо сформулировать требования, которые должны учитываться на этапе логического проектирования базы данных. Требования должны носить конкретный характер.

Требования могут содержать:

2.1.6.1. Требования к эксплуатационным характеристикам базы данных

* *Функциональное назначение программы (о*писание главных функций разрабатываемой программы)
* *Эксплуатационное назначение программы (*описание основных эксплуатационных характеристик разрабатываемой программы)

Конечными пользователями программы могут являться как сотрудники того или иного предприятия/отдела, для которого разрабатывается программный продукт (полное право доступа к информации), так и лица, не имеющие отношения к данному предприятию/отделу (частичное предоставление информации).

Требования к функциональным характеристикам

* Требования к составу выполняемых функций
* *Требования к организации входных данных*

Входные данные программы должны быть организованы в виде вводимого в специальную форму текста или файла, соответствующего определенному шаблону. Данные, вводимые  вручную, проверяются на корректность после попытки сохранения; данные, вводимые из файла, проверяются в ходе анализа и размещения данных.

Файлы указанного формата должны размещаться (храниться) на локальных или съемных носителях, отформатированных согласно требованиям операционной системы. Каждый день происходит резервирование полученной информации на отдельный носитель, для возможности восстановления информации в случае ошибки программы или поломки оборудования.

* *Требования к организации выходных данных*

Выходные данные программы должны быть организованы в виде отчетов или таблиц. Отчеты делятся на несколько групп по предназначению определенной группе пользователей. Доступ к таблицам зависит и от принадлежности пользователя к определенной группе пользователя с теми или иными правами.

Файлы указанного формата должны храниться на локальных или съемных носителях, отформатированных согласно требованиям операционной системы. Отчеты формируются в режиме реального времени и передаются пользователю. Отчеты, являются временными и стираются по завершению работы программы, могут быть сформированы заново при следующем запуске компьютера. При желании любой отчет можно сохранить отдельно.

* *Требования к временным характеристикам*

Требования к временным характеристикам зависит от выполняемой задачи. При формировании отчета временные рамки увеличиваются пропорционально обрабатываемым данным.

* Требования к надежности
* *Требования к обеспечению надежного (устойчивого) функционирования программы*

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1)      организацией бесперебойного питания технических средств;

2)      выполнением рекомендаций  Министерства труда и социального  
развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении  
межотраслевых типовых норм времени на работы по сервисному обслуживанию  
ПЭВМ и оргтехники и сопровождению программных средств»;

3)      выполнением требований ГОСТ 51188-98. Защита информации.  
Испытания программных средств на наличие компьютерных вирусов;

4)      необходимым уровнем квалификации сотрудников профильных подразделений.

* *Время восстановления после отказа*

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени, необходимого на перезагрузку операционной системы и запуск программы, при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Обеспечивается копиями (обеспечивается программой) необходимой информации и хранении дистрибутивов на отдельном компьютере (обеспечивается стороной-заказчиком).

* *Отказы из-за некорректных действий оператора*

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

* Требования к составу и параметрам технических средств

В состав технических средств должен входить персональный компьютер. В случае работы системы в сети все компьютеры должны быть подобны. Так же необходимы кабеля для создания сети, сетевые карты на каждом компьютере и маршрутизатор. При предоставлении возможности поступления информации через сеть Интернет, один из компьютеров в сети, не являющийся сервером, должен иметь модем.

* Требования к информационной и программной совместимости
* *Требования к информационным структурам и методам решения*

Пользовательский интерфейс должен быть интуитивно понятным и содержать подсказки. Должен существовать программный доступ из пользовательского интерфейса к созданию копий базы данных в XML формате. Отчеты должны содержать лишь интересующую информацию. Программа-анализатор должна выполнять запрос за наименее короткое время.

* *Требования к исходным кодам и языкам программирования*

Исходные коды программы должны быть реализованы на языке программирования Delphi. Описать взаимодействие с СУБД.

* *Требования к программным средствам, используемым программой*

Системные программные средства, используемые программой, должны быть представлены локализованной версией операционной системы Windows.

Основой для системы должна стать база данных, в которой будет храниться вся информация.

*Подсистема администрирования (по желанию)*.

 Подсистема администрирования предназначена для управления настроек системы. Управление осуществляется администратором. Управление должно учитывать настройку следующих параметров:

-         регистрация групп пользователей,

-         регистрация пользователей (с настройкой пароля),

-         предоставление различных прав различным группам пользователей,

-         сетевые параметры,

-         настройка параметров источника базы данных,

-         системные параметры.

Для удобства администрирования данная подсистема должна иметь свой интерфейс (где видны все настройки и графы), который предоставляется пользователю в том случае, если последний идентифицирован как администратор. Интерфейс должен иметь инструменты настройки вышеперечисленных параметров, а так же модули ввода, обработки и поиска информации.

*Подсистемы учета*

 Данные подсистемы должны содержать следующие модули:

-          модуль ввода информации,

-          модуль поиска информации (по заданным параметрам),

-          модуль создания отчетов.

 Модуль ввода информации для подсистем учета.

Данный модуль должен осуществлять внесения новых данных в базы, так же модуль должен выполнять следующие функции:

-         обеспечение удобный ввод, соответствующий подсистеме данных,

-         улучшение качество ввода за счет ограничений на значение, типизированные форматы данных, значения по умолчанию, списки выбора значения, и т.п.,

-         обеспечение ввода критериев поиска из списка имеющихся параметров,

-         обеспечение ввода информации из файлов.

*Модуль поиска информации для подсистем учета*

Модули поиска информации всех подсистем учета должны обеспечивать выборку информации из базы данных по заданным критериям и выполнять следующие функции:

-         обеспечение задания критериев поиска,

-         создание запросов по заданным критериям поиска.

-         обеспечение удобного предоставления найденной информации для пользователя.

*Модуль создания отчетов*

-         данный модуль должен обеспечивать выборку информации по заданным параметрам и выполнять следующие функции:

-         создание соответственного электронного файла-документа с отчетом

-         вывод документа на печать

-         возможность рассылки данного документа на почту или на другой компьютер в сети.

* *Требования к защите информации и программы*

В Системе должен быть обеспечен надлежащий уровень защиты информации в соответствии с законом о защите персональной информации и программного комплекса в целом от несанкционированного доступа – «Об информации, информатизации и защите информации» РФ N 24-ФЗ от 20.02.95.

2.1.6.2. Обоснование выбора типа СУБД

**2.2. Логическое проектирование модели базы данных**

2.2.1. Описание состава отношений базы данных и набора атрибутов каждого отношения

*Пример:*

*В состав отношений данной базы данных входят следующие объекты:*

*Название таблицы 1 – описание таблицы 1.*

*Скрин таблицы 1 (в режиме конструктора)*

2.2.2. Первичные и внешние ключи отношений

*Описание всех первичных и внешних ключей*

|  |  |  |  |
| --- | --- | --- | --- |
| *№ п/п* | *Сущность* | *Первичные ключи* | *Вторичные ключи* |
|  |  |  |  |

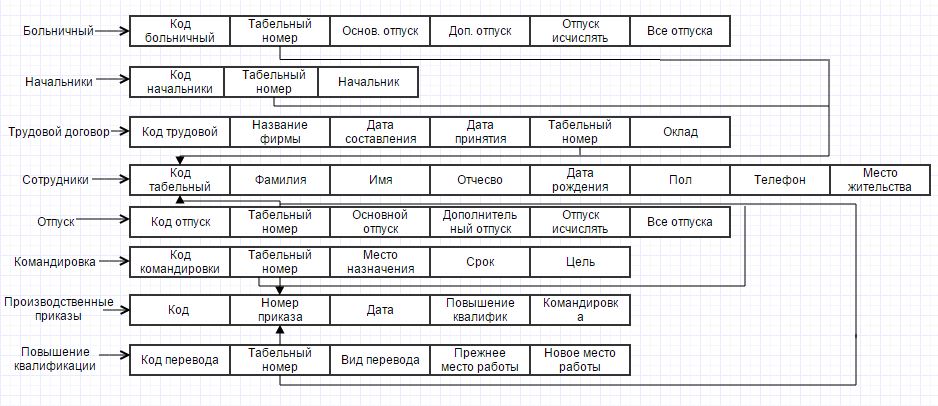
2.2.3. Нормализация отношений базы данных

2.2.4. Ограничения целостности данных в базе данных

2.2.5. Вертикальная диаграмма связей между внешними и первичными ключами

*Пример:*

*На рисунке № представлена вертикальная диаграмма связей между внешними и первичными ключами.*

**

*Вертикальная диаграмма*

**2.3. Разработка хранимых процедур и триггеров**

2.3.1. Описание разработанных хранимых процедур и триггеров

Описание разработанных хранимых процедур и триггеров для обеспечения ограничений целостности и выполнения законченных фрагментов действий. В основном здесь являются макросы.

2.3.2. Защита от несанкционированного доступа к данным

**2.4. Описание разработанного программного обеспечения**

**2.5. Руководство пользователя**

**3. Охрана труда**

**Заключение**

**2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

**2.1 Построение концептуальной модели**

Для построения концептуальной модели предметной области рекомендуется придерживаться следующей последовательности действий.

1. Описать предметную область (описание должно быть кратким, но достаточным для принятия решений по проекту базы данных).

2. Определить состав и содержание информации, используемой в данной предметной области, в том числе:

- составить перечень задач и запросов, указать входные и выходные данные;

- определить частоту решения задач и используемые при этом бизнес-правила;

- определить возможные будущие изменения информационных потребностей пользователей;

- установить уточнённые требования к информационным потребностям пользователей.

3. Выявить сущности, в том числе:

- определить атрибуты каждой сущности и требования к ним;

- определить ключ каждой сущности;

- разработать, если необходимо, классификаторы и кодификаторы сущностей;

- определить требования к сущностям, вытекающие из бизнес-правил предметной области.

4. Выявить связи между сущностями, в том числе:

- структурные связи для выявления классов и подклассов сущностей;

- функциональные связи типа 1:1, 1: m, n:m, n-арные;

- если необходимо, определить атрибуты связей.

5. Представить концептуальную модель в виде концептуальной схемы.

6. Проанализировать модель с учётом информационных потребностей пользователей.

Результаты проделанной работы рекомендуется оформить в виде следующих документов:

-каталог задач и запросов предметной области;

- альбом форм входных и выходных документов (разместить в приложении к курсовой работе);

- граф информационных связей задач и запросов;

- таблица сущностей;

- таблица атрибутов;

- таблица связей;

- таблица атрибутов связей.

В таблице сущностей могут быть отражены следующие сведения:

- наименование сущности;

- условное обозначение;

- первичный ключ;

- количество экземпляров сущностей на момент обследования моделируемой предметной области;

- динамика изменения количества экземпляров за определённый период, например, в процентах;

- частота коррекции;

- перечень задачи и запросов, в которых используется данная сущность;

- активность, то есть минимальное количество экземпляров сущности, выбираемое при однократном обращении к ней;

- ограничение на доступность.

В таблице атрибутов для каждой сущности могут быть приведены следующие сведения:

- наименование атрибута;

- условное обозначение;

- признак ключа и тип значения (атомарное или множественное);

- формат (тип и длина);

- диапазон значений;

- возможность принимать неопределённое значение;

- ограничение на доступность (если отличается от ограничений для сущности);

- метод контроля достоверности.

В таблице для связей могут быть приведены следующие сведения:

- наименование связи;

- условное обозначение;

- тип связи;

- характеристика динамики (динамическая или статическая);

- характеристика мощности связи, то есть количество экземпляров сущностей, участвующих в связи (варианты: 0, 1 или много; 1 или много; 0 или 1; точное число);

- перечень атрибутов связи.

Описание атрибутов связей аналогично таблице атрибутов, рассмотренной выше.

На все данные, полученные в результате анализа предметной области и построения концептуальной модели должны быть ссылки в последующих разделах курсовой работы.

По результатам этапа концептуального проектирования необходимо сформулировать требования, которые должны учитываться на этапе логического проектирования базы данных. Требования должны носить конкретный характер.

Требования могут содержать:

- требования к эксплуатационным характеристикам базы данных;

- тип СУБД;

- требования к разрабатываемому программному обеспечению;

- описание ролей пользователей и др.

Для осуществления логического этапа проектирования необходимо выполнить следующие действия.

1. Построить реляционную модель предметной области одним из известных методов, в том числе:

- описать последовательность принимаемых решений в соответствии с выбранным методом проектирования;

- дать математическое описание полученной модели с использованием аппарата теории множеств и математической логики.

2. Выбрать СУБД и обосновать этот выбор (при выборе учесть результаты концептуального этапа и требования к логическому этапу проектирования).

3. Выбрать технические средства и обосновать это выбор (при выборе необходимо учитывать результаты предыдущих этапов проектирования, в том числе, количественные характеристики).

В процессе создания базы данных в среде конкретной СУБД необходимо:

- создать таблицы;

- определить свойства полей;

- задать ключи;

- создать необходимые индексы;

- создать связи;

- определить правила ссылочной целостности;

- создать формы для работы с таблицами;

- реализовать запросы;

- создать формы для запросов;

- спроектировать отчёты;

- создать необходимые кнопочные формы;

- создать представления;

- написать необходимый программный код;

- заполнить базы данных и продемонстрировать работу приложения.

Все перечисленные действия должны быть обоснованы.

Результаты проделанной работы должны быть представлены в виде курсовой работы и приложения, работу которого необходимо продемонстрировать непосредственно на компьютере.

**2.2 Этап инфологического проектирования:**

**теоретические сведения**

Этап инфологического проектирования базы данных связан с анализом и описанием разнообразных информационных требований пользователей. Результатом этапа инфологического проектирования является представление информационных требований в виде целостной концептуальной инфологической модели предметной области. Инфологическую модель представить в виде диаграммы, используя средства моделирования "сущность-связь". Построение диаграммы производится на основе следующих проектных шагов:

- определение информационных элементов каждого информационного требования;

- определение сущностей;

- определение атрибутов сущностей;

- определение ключевых атрибутов (выбор ключей);

- определение связей между сущностями;

- построение концептуальной инфологической модели.

*Концептуальное моделирование — изучает объекты в общем виде, т.е. и в статике, и динамике с помощью представления информации об объекте в форме достаточной для полной характеристики объекта с точки зрения поставленных целей или сформулированных задач. Объект необходимо описать данными,* достаточными для решения требуемых задач. При этом и структура данных должна наглядно и полно отображать особенности объекта. В концептуальном моделировании кроме этого требуется, чтобы эта информация была представлена согласно теории проектирования баз данных. Последнее условие вытекает из разнородности типов данных и из-за разнообразия описания объекта, его свойств, внутренних и внешних взаимосвязей и взаимодействий между внутренними составляющими объекта и между взаимосвязью объекта с внешней средой. Причем структура данных должна быть независима от решаемых задач, языка программирования, типа компьютеров и т.д.

В концептуальном моделировании различают три этапа: сбор информации об объекте, концептуальный анализ и синтез концептуальной модели профессиональной среды.

Рассмотрим этапы концептуального моделирования согласно схеме сплошного жизненного цикла автоматизированной информационной системы.

Пусть выбран объект автоматизации. Например, это может быть рабочее место начальника вокзала или вагонного депо, начальника сортировочной станции, кассира билетных касс, диспетчера электродепо, начальника отдела кадров дороги или отделения. В любом случае проект ИС должен ориентироваться на конкретного специалиста. Тогда под словом «объект» будем понимать производственную среду вместе с документацией, материальной частью и профессиональными работниками, выполняющими свои должностные обязанности. Последние определяют функции, которые выполняет работник в общей структуре производственного процесса. Реализация этих функций приводит к некоторой цели, например, обеспечению жизнедеятельности транспортного процесса.

Любой проект ИС также должен иметь определенную цель. Совершенно необязательно, что непосредственно от внедрения ИС должен быть экономический эффект, хотя последний желателен. Поскольку базы данных являются составной частью информационной системы или автоматизированных рабочих мест, то они, как правило, ориентированы на конкретную профессиональную среду либо на технологическую цепочку, связывающую несколько видов профессиональной деятельности. Поэтому и цели разработки баз данных также связаны с целями соответствующих служб или отделов, но при этом проектирование баз данных имеет самостоятельные цели.

Целями в ИС могут быть:

- увеличение оперативности обработки текущей информации служб;

- увеличение оперативности выдачи отчетности и статистики информации;

- обеспечение жизнедеятельности некоторой службы.

Последнюю цель прокомментируем. С ростом потоков информации, а точнее необходимостью ее обработки в реальном режиме времени, т.е. с ограниченным ресурсом времени, служба может не справиться с этой обработкой и тогда технологический процесс перевозок может быть неуправляем: в лучшем случае он затормозится, в худшем случае может создаться предаварийная ситуация. Можно перечислить и другие цели, но, как правило, они приведут к вышеперечисленным (оперативность и жизнеобеспечение).

Если объект автоматизации выбран, должностные обязанности профессионала известны, т.е. функции перечислены, цель автоматизации сформулирована, то можно перейти к формированию и формализации функций, выполняемых ИС. Обычно при этом разрабатывают функциональную структуру ИС. Различают два подхода к разработке функциональной структуры:

- от задач профессиональной области (от объекта);

- от задач, типизированных в СУБД (Система управления базой данных).

Первый подход основывается на конкретных задачах профессиональной среды, с перечислением порядка выполнения и описанием требуемой входной и выходной информации.

Второй подход основывается на сформированных функциях СУБД: создание и редактирование баз данных, сортировка, фильтрация, поиск, оформление отчетов. Такие типовые задачи СУБД будут отличаться входной и выходной информацией, отображающей профессиональную сферу.

Возможно и сочетание этих подходов.

Формирование целей, выбор задач требуют знания всех технологических операций, выполняемых на исследуемом объекте. Рекомендуется максимально использовать знания специалиста или группы экспертов информатизируемой профессиональной среды, применяя методы и подходы системного анализа.

***Концептуальный анализ профессиональной среды***

Основная цель концептуального анализа заключается в исследовании профессиональной среды как объекта моделирования в виде некоторой системы данных. Для этого необходимо как можно подробнее рассмотреть следующие этапы:

- выделение профессиональной среды;

- выбор количественных и качественных характеристик (свойств, атрибутов) функциональной среды;

- сбор информации на основе выбранных характеристик;

- анализ данных.

Профессиональная среда определяется должностными инструкциями специалиста и перечислением его обязанностей, среди которых выделяются те, которые могут быть хотя бы частично выполнены при помощи компьютера. Безусловно, есть и такие обязанности, которые нельзя формализовать, чтобы затем перепоручить компьютеру. Обычно формализации поддаются наиболее рутинные, «бумажные» и вычислительные работы, такие как хранение и поиск информации, составление планов, разработка графиков предупредительных ремонтов и осмотров, составление списка наличия запчастей на складе или оформление заказа на недостающие запчасти, расчет заработной платы с изменяющимися коэффициентами тарифных ставок. Желательно обязанности специалиста перечислить в максимально возможном объеме, упорядочивая их по мере необходимости автоматизации. Затем обязанности рассматриваются в отдельности и описываются как можно подробнее, вплоть до элементов выполнения. Обязанности специалиста можно подразделить на три рода деятельности:

- преобразование материи, т.е. технологические операции над предметами труда, например, производство деталей, ремонт электроприборов, строительство железнодорожного полотна и т.д.;

- преобразование энергии: обеспечение железной дороги постоянным и переменным током, контроль и распределение электроэнергии, управление теплосетями и т.д.;

- преобразование информации, например, сбор информации о поездах, поступающих на сортировочную станции и выходящих с нее и выдача документов на переформирование составов;

- перечисленные процессы сопровождаются оформлением соответствующих справок, документов, отчетов, журналов и т.д., порождая соответствующий информационный поток.

Эти процессы необходимо разделить для описания исходной и обработанной информации, что позволит проследить динамику преобразования информации, а также источников и потребителей этой информации за пределами разрабатываемой профессиональной среды. Целесообразно, по мере возможности, включить в проект ИС задачи коллег-специалистов, работающих в этой технологической цепочке, или в родственных областях.

Например, составление графика работы поездных бригад, с одной стороны, связано с расписанием поездов, а с другой, - с бухгалтерией, которая согласно выполненной работе начисляет заработную плату.

Рекомендуется разделение задач проводить с учетом требования системного анализа, исходя из следующих принципов: целостности и многоаспектности. Принцип целостности предполагает относительную самостоятельность объекта или системы в выполнении некоторой функции или нескольких функций. При этом свойства всего объекта являются не только суммой свойств элементов, но и обусловлены эффектами взаимосвязей и взаимодействия этих элементов, т.е. целое (система) качественно выше, чем просто набор элементов. Принцип многоаспектности предусматривает анализ объекта с различных точек зрении, например, по функциональному назначению, профессиональному уровню, внутреннему содержанию и структуре. При этом объект может быть достаточно сложен для описания.

Путь имеется N - элементов. Тогда описание свойств этих элементов, их взаимосвязей и взаимодействий по 2-м, 3-м и т.д. элементам будет описываться следующей величиной: n = 2N-1.

Для анализа таких сложных объектов используют декомпозиции объекта, придерживаясь критерия: связи и взаимодействия между подсистемами являются несущественными по сравнению со связями и взаимодействиями элементов внутри подсистем. Степень детализации может быть сколь угодно подробная и зависит от решаемых задач или от предлагаемой структуры базы данных. Схема декомпозиции может быть иерархической, например, на основе классификации. Системы могут быть наделены внутренними признаками или внешними. Внутренние неизменны для однородных систем. Например, электродвигатели нетрудно представить в виде набора элементов: статор, ротор, электромагнитное поле и т.д.

Для систем с внешними признаками напротив характерна изменчивость. Например, товарные вагоны имеют ряд элементов, качественные и количественные характеристики которых значительно различаются по диапазону в пределах некоторого класса. Грузовые вагоны составляют подкласс, но они различаются на платформы, цистерны и рефрижераторы. В результате декомпозиции мы представляем профессиональную среду в виде подобластей (фрагментов). Напомним, что эти фрагменты выполняют одну из функций и в некотором смысле они самостоятельны. Для удобства будем называть эти фрагменты объектами моделирования. Следовательно, профессиональная среда будет представлена объектами, между которыми существует взаимосвязь и взаимодействие. Эти объекты при описании их знаками или символами будем называть *денотатами*, а смысловое понятие *денотата - концептом*.

*Концепт -* содержание понятия, обозначенного знаком (идентификатором) и отображающего в обобщенной форме предмет или явление действительности посредством фиксации признаков. В качестве последних выступают свойства предметов, явлений и отношений между ними. Пользуясь терминологией семиотики, - науки о знаковых системах, - мы получим триаду терминов, где знак обозначает определенный денотат и отражает концепт - смысл, содержание понятия или реального физического объекта.

Для рассмотрения набора или множества знаков в семиотике вводится понятие знаковой системы, в которой отношения между знаками отображают отношения между обозначенными денотатами. В дальнейшем будем использовать табличные представления данных как наиболее привычные для восприятия человеком, так и наиболее удобные для ввода их в ЭВМ. Тогда строки таблиц будут обозначать конкретные экземпляры объекта - денотаты, а столбцы будут отображать значения признаков (атрибутов). Вся таблица обозначает конкретный объект. Несколько таблиц представляют собой профессиональную среду.

Полностью развернутое описание профессиональной области в таком виде представляет собой одну из форм моделей концептуального моделирования. Такое, особенно табличное, представление может быть усовершенствовано, если ранее при моделировании не учитывались характеристики: *избыточность, противоречивость, вычисляемость*.

*Избыточность* — повторение в таблицах значений записей или атрибутов. Так, в приведенном выше примере, атрибут «Семестр» повторяется в двух последних объектах.

*Противоречивость* данных — обычно возникает вследствие ошибок при оформлении таблиц. Например, количество лекционных часов не может быть больше их общего количества для одного и того же предмета. Противоречивость возникает из-за составления таблиц по разным источникам, или из-за ошибок оператора.

*Вычисляемость* — определение значений некоторых атрибутов по значениям известных. Так общее количество часов на предмет складывается из лекционных часов, часов лабораторных и практических занятий.

Введем еще два понятия, позволяющих перейти от концептуального моделирования к проектированию реляционных баз данных.

Под «отношением» - будем понимать двумерную таблицу данных, где строки являются записями или кортежами, а столбцы доменами. Записи представляют экземпляры объекта.

Отношения подразделяются на «объектные» и «связные». Объектные отношения несут информацию о реальных объектах. Связные отношения несут информацию о связи объектных отношений.

Ключевых атрибутов у объектного отношения может быть один или несколько. В последнем случае говорят о составном ключе. В связном отношении могут быть ключевые атрибуты двух, трех и больше объектных отношений. Но обычно, в силу теории проектирования баз данных, ограничиваются двумя ключами. Однако могут присутствовать и неключевые атрибуты, которые порождаются или сопутствуют связи объектных отношений.

Рассмотрим детально основные шаги формирования ***концептуальной инфологической модели***.

1. Каждое информационное требование представляется в виде идентификационных реквизитов, обозначающих элементы данных, входящие в анализируемое информационное требование. Каждый реквизит отображает конкретное свойство и должен иметь по возможности уникальное имя

2. Анализ смыслового содержания перечня реквизитов используется для выявления основных сущностей рассматриваемой предметной области. Результатом данного шага является определение перечня сущностей заданной предметной области.

3. Каждой сущности должен соответствовать определенный набор атрибутов, характеризующий данную сущность. Набор атрибутов выбирается из определенных ранее информационных элементов по отдельным информационным требованиям.

4. На основе анализа функциональных зависимостей между атрибутами каждой сущности выделяются "первичные ключи" всего отношения, описывающего сущность.

5. Выявить связи между сущностями - объектами и на основании анализа и описания этих связей составить ER-диаграмму, описывающую информационную структуру всей базы данных предметной области

6. Построить концептуальную инфологическую модель, дополнив схему взаимосвязей сущностей раскрывающими каждую сущность атрибутами.

**Инфологическое проектирование базы данных**

1. Поэлементный состав каждого информационного требования в виде перечня идентификационных реквизитов.

Следующим шагом является определение атрибутов каждой сущности и выделение ключевых атрибутов, которые будем обозначать выделением <>.

**2.3 Этап логического проектирования:**

**теоретические сведения**

Концептуальная инфологическая модель данных может быть отображена на логическую модель данных, поддерживаемую конкретной СУБД.

Современные системы управления базами данных наиболее часто поддерживают следующие логические модели: реляционные, сетевые и иерархические. Рассмотрим процесс отображения концептуальной модели на реляционную модель данных.

При проектировании реляционной базы данных 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.

Процесс уменьшения избыточности информации в базе данных называется нормализацией. В теории нормализации баз данных разработаны достаточно формализованные подходы по разбиению данных, обладающих сложной структурой, среди нескольких таблиц. Эти вопросы детально освещаются в специальной литературе. Мы остановимся на некоторых практических аспектах нормализации таблиц, не вдаваясь в их теоретическое обоснование.

Теория нормализации оперирует с пятью нормальными формами таблиц (от первой до пятой включительно). 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. При практическом проектировании баз данных четвертая и пятая формы, как правило, не используются, поэтому мы ограничимся рассмотрением первых трех нормальных форм. Воспользуемся результатами теории при проектировании многотабличной базы данных с эффективной структурой.

При создании базы данных следует иметь ввиду, что

- 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

Основным этапом логического проектирования является отображение концептуальной модели в реляционную, для этого сначала необходимо провести нормализацию полученных на этапе инфологического проектирования отношений.

Процесс нормализации отношений состоит из следующих этапов:

- преобразование отношений в первую нормальную форму (1НФ );

- преобразование отношений во вторую нормальную форму (2НФ),

- преобразование отношений в третью нормальную форму (3НФ).

***Первая нормальная форма таблицы***

Таблица в первой нормальной форме должна удовлетворять следующим требованиям:

1. Таблица не должна иметь повторяющихся записей.

2. В таблице должны отсутствовать повторяющиеся группы полей.

3. Строки должны быть не упорядочены.

4. Столбцы должны быть не упорядочены.

Для удовлетворения условия 1 каждая таблицы должна иметь уникальный индекс.

Для исключения повторяющихся записей можно воспользоваться одним из следующих способов.

1. Добавьте в таблицу новое уникальное ключевое поле Идентификатор, что позволит исключить многозначность. Однако это - далеко не лучший метод. При разработке многотабличных форм и отчетов связь между таблицами часто осуществляется посредством индексов. Ввод нового ключевого поля в таблицу не позволит связать таблицы по значениям совпадающих полей при создании многотабличных форм и отчетов.

2. Второй способ исключения повторяющихся записей состоит в использовании уникального составного индекса, состоящего из соответствующих полей.

После того, как мы разделили повторяющиеся объекты и определили поля, которые образуют уникальный индекс в каждой таблице, считается, что таблица находится в первой нормальной форме. Таким образом, первый шаг при нормализации заключается в образовании двумерных таблиц, содержащих элементы данных в качестве атрибутов. Повторяющиеся группы элементов данных выделяются в отдельный вид отношения. Это и является 1НФ данного отношения.

***Вторая нормальная форма таблицы***

О таблице говорят, что она находится во второй нормальной форме, если:

1. Она удовлетворяет условиям первой нормальной формы.

2. Любое не ключевое поле однозначно идентифицируется полным набором ключевых полей.

Из приведенного выше определения следует, что понятие второй нормальной формы применимо только к таблицам, имеющим составной индекс.

Таким образом, второй шаг нормализации состоит в том, чтобы выделить ключи и зависящие от них атрибуты. Для отношения, находящегося в первой нормальной форме для приведения ко второй нормальной форме необходимо выделить группы не ключевых атрибутов, зависящие от части составного ключа. Эти группы могут образовать отдельные отношения, в которых не ключевые атрибуты будут зависеть только от определенной части составного ключа.

***Третья нормальная форма таблицы***

О таблице говорят, что она находится в третьей нормальной форме, если:

1 Она удовлетворяет условиям второй нормальной формы.

2. Ни одно из не ключевых полей таблицы не идентифицируется с помощь другого не ключевого поля.

Сведение таблицы к третьей нормальной форме предполагает разделение таблицы с целью помещения в отдельную таблицу (или несколько таблиц) столбцов, которые не зависят от полного ключа. В результате такого разбиения каждое из не ключевых полей должно оказаться независимым от какого-либо другого не ключевого поля.

Таким образом, третий шаг нормализации состоит в выявлении возможных транзитивных зависимостей и выделения отдельных отношений для исключения таких зависимостей.

***ER-проектирование баз данных***

Метод возник позже теории нормализации, поэтому свободен от многих недостатков, присущих теории нормализации. При числе атрибутов >20 декомпозиционный подход, основанный на нормальных формах, становится громоздким, и поэтому привлекаются другие методы, например, метод «Сущность - связь» (Еssenсе - сущность, Relation - связь). Он примечателен тем, что функциональные зависимости используются не на начальном этапе проектирования, а на завершающем. Подход оснащен различными формами диаграмм, что позволяет визуализировать различные этапы проектирования баз данных. В настоящее время этот метод поддерживается пакетом автоматизированного проектирования баз данных ERWIN.

Для ER-проектирования необходимо ввести некоторые понятия. Приведем несколько интуитивные определения.

**СУЩНОСТЬ.** Сущность определяется как некоторый объект, представляющий интерес для организации. Этот объект должен иметь экземпляры, отличающиеся друг от друга и допускающие однозначную идентификацию. Единственный определяющий признак, который может помочь в нахождении сущностей, состоит в том, что сущность – это, как правило, существительное. Примерами сущностей могут служить машины, банковские счета, колледжи, служащие и контракты. На рисунке.2 сущностями являются преподаватель и курс.

**СВЯЗЬ.** Связь представляет собой соединение между двумя или более сущностями. При поиске связей в основном следует полагаться на то обстоятельство, что связь обычно выражается глаголом. Типичными примерами связей между двумя сущностями являются: служащие РАБОТАЮТ - В отделах, студенты ИЗУЧАЮТ учебные предметы, рабочие ОБСЛУЖИВАЮТ механизмы.

Тесно связано с предыдущими третье важное понятие - атрибут.

**АТРИБУТ.** Атрибут есть свойство сущности. Например, атрибутами, могущими быть свойствами сущности преподаватель, являются: фамилия, звание, должность. Определения сущности, связи и атрибута не отличаются особой конкретностью, однако являются приемлемыми для использования в тех целях, на которые они рассчитаны. Атрибут, или набор атрибутов, используемый для идентификации экземпляра сущности, называется *ключом сущности*. Каждый экземпляр связи однозначно определяется набором ключей сущностей, соединяемых этой связью и является одним *ключом связи*.

На диаграммах ЕR-типа сущности представляются в виде прямоугольников, а связи – в виде ромбов. Ниже каждой сущности размещается атрибут, или набор атрибутов, являющийся ключом сущности для данной сущности.

**Ключ сущности -** атрибут(ы) для идентификации экземпляра сущности.

**Ключ связности** < атрибуты сущностей > для идентификации связи сущностей.

**Степень связи: 1:1, 1:n, n:1, m:n**, например: 1:1 каждому преподаватель читает только один предмет, 1:n один преподаватель (читает) много курсов, n:1 много преподавателей (читают) один и тот же курс, m:n много преподавателей (читают) много курсов.

*Связь один к многим (1:N).* В данном случае одному экземпляру первой сущности может соответствовать любое число экземпляров второй сущности. Здесь также учитывается класс принадлежностей.

*Связь многие к многим (М:N).* В этом случае каждая из ассоциированных сущностей может быть представлена любым количеством экземпляров.

В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в CASE-системах, поддерживающих автоматизированное проектирование реляционных баз данных.

Как уже отмечалось, более распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD-Entity-Relationship Diagram). С их помощью определяются важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). Получение модели данных часто называют кратко ER-моделью. Из модели «сущность- связь» могут быть порождены такие существующие модели данных, как иерархическая сетевая и реляционная, поэтому она является наиболее общей.

Модель «сущность-связь» была предложена в 1976г. П. Ченом. Очень важным свойством модели «сущность-связь» является возможность ее представления в виде графической схемы, что значительно облегчает анализ предметной области.

Если все экземпляры участвуют в связи, то связи называются обязательными, если не участвуют, то необязательными. Другими словами, если некоторым значениям первой сущности нет соответствующих значений другой сущности, то первая сущность является необязательной. Эта характеристика связи описывает класс принадлежностей входящих в нее сущностей.

Это могут быть:

1. Обязательный класс принадлежностей.

2. Необязательный класс принадлежностей.

Вид зависимости и класс принадлежности определяют кардинальность сущности.

В настоящее время используют 7 правил генерации отношений из диаграмм ER-типа.

**ПРАВИЛО 1.** Если степень бинарной связи равна 1:1 и класс принадлежности обеих сущностей является u1086 обязательным, то требуется только одно связное отношение. Первичным ключом этого отношения может быть ключ любой из двух сущностей.

**ПРАВИЛО 2.** Если степень бинарной связи равна 1:1 и класс принадлежности одной сущности является обязательным, а другой необязательным, то необходимо построение двух отношений. Под необязательную сущность необходимо выделение одного отношения, при этом ключ сущности должен служить первичным ключом для соответствующего отношения. Кроме этого, ключ сущности, для которого класс принадлежности является необязательным, добавляется в качестве атрибута в отношение, выделенное для сущности с обязательным классом принадлежности.

**ПРАВИЛО 3.** Если степень бинарной связи равна 1:1 и класс принадлежности ни одной сущности не является обязательным, то необходимо использовать три отношения: по одному для каждой сущности, ключи которых служат в качестве первичных в соответствующих отношениях, и одного для связи, в котором будет иметься по одному ключу от каждой сущности.

**ПРАВИЛО 4.** Если степень бинарной связи равна 1:N и класс принадлежности обоих сущностей является обязательным, то достаточно использовать два отношения, одно на онарную сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения и одно связное дополнительно. Ключ онарной сущности должен быть представлен как атрибут в отношение, отводимое для n-связной сущности.

**ПРАВИЛО 5.** Если степень бинарной связи равна 1:N и класс принадлежности n-связной сущности является необязательным, то достаточно использовать три отношения, по одному на каждую сущность, при условии, что ключ каждой сущности служит в качестве первичного ключа для соответствующего отношения. Дополнительно ключ одной сущности должен быть представлен как атрибут в отношение, отводимое для n-связной сущности.

**ПРАВИЛО 6.** Если степень бинарной связи равна 1:N и классы принадлежности сущностей являются необязательными, то необходимо формирование трех сущностей: по одному для каждой сущности, причем ключ сущности служит первичным ключом соответствующего отношения, и одного отношения для связи. Связное отношение должно иметь ключи каждой сущности.

**ПРАВИЛО 7.** Если степень бинарной связи равна m:n, то для хранения данных необходимы три отношения: по одному для каждой сущности, причем ключ каждой сущности используется в качестве первичного ключа. акая связь порождает пробелы и избыточное повторение информации соответствующего отношения, одного отношения для связи. Последнее отношение должно иметь в числе своих атрибутов ключ каждой сущности. Принцип построения отношений соответствует предыдущему примеру и отличается перечислением в связном отношении всех связей между ключевыми элементами объектных отношений.

**2.4 Этап физического проектирования:**

**теоретические сведения**

На данном этапе необходимо выбрать СУБД для реализации информационной системы. Рассмотрим критерии выбора СУБД при создании информационных систем, которыми следует руководствоваться.

Выбор системы управления баз данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранный программный продукт должен удовлетворять как текущим, так и будущим потребностям предприятия, при этом следует учитывать финансовые затраты на приобретение необходимого оборудования, самой системы, разработку необходимого программного обеспечения на ее основе, а также обучение персонала. Кроме того, необходимо убедиться, что новая СУБД способна принести предприятию реальные выгоды.

Можно выделить несколько групп критериев:

* Моделирование данных
* Особенности архитектуры и функциональные возможности
* Контроль работы системы
* Особенности разработки приложений
* Производительность
* Надежность
* Требования к рабочей среде
* Смешанные критерии

Рассмотрим каждую из этих групп в отдельности.

*Моделирование данных.*

* *Триггеры и хранимые процедуры.* Триггер - программа базы данных, вызываемая всякий раз при вставке, изменении или удалении строки таблицы. Триггеры обеспечивают проверку любых изменений на корректность, прежде чем эти изменения будут приняты. Хранимая процедура – программа, которая хранится на сервере и может вызываться клиентом. Поскольку хранимые процедуры выполняются непосредственно на сервере базы данных, обеспечивается более высокое быстродействие, нежели при выполнении тех же операций средствами клиента БД. В различных программных продуктах для реализации триггеров и хранимых процедур используются различные инструменты.
* *Средства поиска.* Некоторые современные системы имеют встроенные дополнительные средства контекстного поиска.
* *Предусмотренные типы данных.* Здесь следует учесть два фактически независимых критерия: базовые или основные типы данных, заложенные в систему, и наличие возможности расширения типов. В то время как отклонения базовых наборов типов данных у современных систем от некоего стандартного, обычно, невелики, механизмы расширения типов данных в системах того или иного производителя существенно различаются.
* *Реализация языка запросов.* Все современные системы совместимы со стандартным языком доступа к данным SQL-92, однако многие из них реализуют те или иные расширения данного стандарта.

*Особенности архитектуры и функциональные возможности.*

* *Мобильность.* Мобильность – это независимость системы от среды, в которой она работает. Средой в данном случае является как аппаратура, так и программное обеспечение (операционная система).
* *Масштабируемость.* При выборе СУБД необходимо учитывать, сможет ли данная система соответствовать росту информационной системы, причем рост может проявляться в увеличении числа пользователей, объема хранимых данных и объеме обрабатываемой информации.
* *Распределенность.* Основной причиной применения информационных систем на основе баз данных является стремление объединить взгляды на всю информацию организации. Самый простой и надежный подход - централизация хранения и обработки данных на одном сервере. К сожалению, это не всегда возможно и приходится применять распределенные базы данных. Различные системы имеют разные возможности управления распределенными базами данных.
* *Сетевые возможности.* Многие системы позволяют использовать широкий диапазон сетевых протоколов и служб для работы и администрирования.

Контроль работы системы

* *Контроль использования памяти компьютера.* Система может иметь возможность управления использованием, как оперативной памяти, так и дискового пространства. Во втором случае это может выражаться, например, в сжатии баз данных, или удалении избыточных файлов.
* *Автонастройка.* Многие современные системы включают в себя возможности самоконфигурирования, которые, как правило, опираются на результаты работы сервисов с*амодиагностики производительности.* Данная возможность позволяет выявить слабые места конфигурации системы и автоматически настроить ее на максимальную производительность.

Особенности разработки приложений.

* Многие производители СУБД выпускают также средства разработки приложений для своих систем. Как правило, эти средства позволяют наилучшим образом реализовать все возможности сервера, поэтому при анализе СУБД стоит рассмотреть также и возможности средств разработки приложений.
* *Средства проектирования.* Некоторые системы имеют средства автоматического проектирования, как баз данных, так и прикладных программ. Средства проектирования различных производителей могут существенно различаться.
* *Многоязыковая поддержка.* Поддержка большого количества национальных языков расширяет область применения системы и приложений, построенных на ее основе.
* *Возможности разработки Web-приложений.* При разработке различных приложений зачастую возникает необходимость использовать возможности среды Internet. Средства разработки некоторых производителей имеют большой набор инструментов для построения приложений под Web.
* *Поддерживаемые языки программирования.* Широкий спектр используемых языков программирования повышает доступность системы для разработчиков, а также может существенно повлиять на быстродействие и функциональность создаваемых приложений.

Надежность.

Понятие надежности системы имеет много смыслов – это и сохранность информации независящая от любых сбоев, и безотказность работы системы в любых условиях, и обеспечение защиты данных от несанкционированного доступа.

* *Восстановление после сбоев.* При возникновении программных или аппаратных сбоев целостность, да и работоспособность всей системы может быть нарушена. От того, как эффективно спланирован механизм восстановления после сбоев, зависит жизнеспособность системы.
* *Резервное копирование.* В результате аппаратного сбоя может быть частично поврежден или выведен из строя носитель информации и тогда восстановление данных невозможно, если не было предусмотрено резервное копирование базы данных, или ее части. Резервное копирование спасает и в ситуациях, когда происходит логический сбой системы, например при ошибочном удалении таблиц. Существует множество механизмов резервирования данных (хранение одной или более копий всей базы данных, хранение копии ее части, копирование логической структуры и т.д.). Зачастую в систему закладывается возможность использования нескольких таких механизмов.
* *Откат изменений.* При выполнении транзакции применяется простое правило – либо транзакция выполняется полностью, либо не выполняется вообще. Это означает, что в случае сбоев, все результаты недоведенных до конца транзакций должны быть аннулированы. Механизм отката может иметь различное быстродействие и эффективность.
* *Многоуровневая система защиты.* Информационная система организации почти всегда включает в себя секретную информацию, поэтому для предотвращения несанкционированного доступа используется служба идентификации пользователей. Уровень защиты может быть различным. Кроме непосредственной идентификации пользователей при входе в систему может использоваться также механизм шифрования данных при передаче по линиям связи

*Требования к рабочей среде.*

* *Поддерживаемые аппаратные платформы.*
* *Минимальные требования к оборудованию.*
* *Максимальный размер адресуемой памяти.* Поскольку почти все современные системы используют свою файловую систему, немаловажным фактором является то, какой максимальный объем физической памяти они могут использовать.
* *Операционные системы, под управлением которых способна работать СУБД*.

Смешанные критерии.

* *Качество и полнота документации.* К сожалению, не все системы имеют полную и подробную документацию.
* *Локализованность.* Возможность использования национальных языков не во всех системах реализована полностью.
* *Модель формирования стоимости.* Как правило, производители СУБД используют определенные модели формирования стоимости. Например, стоимость одного и того же продукта может существенно изменяться в зависимости от того, сколько пользователей будет с ним работать.
* *Стабильность производителя.*
* *Распространенность СУБД.*

Рассмотрим средства автоматизации конструирования реляционных таблиц, форм просмотра и редактирования данных в СУБД MS Access.

***Создание базы данных***

База данных (БД) в MS Access представляет собой совокупность средств для ввода, хранения, просмотра, выборки и управления информацией. К этим средствам относятся таблицы, формы, отчеты, запросы.

Создание новой базы данных осуществляется командой Файл | Создать . После ввода имени создаваемой базы и нажатия кнопки Создать откроется окно базы данных. Оно состоит из шести вкладок, которые пока пусты. Необходимо далее создать все компоненты, входящие в базу данных. Их перечень соответствует ярлыкам вкладок окна базы данных.

Основным компонентом базы данных являются таблицы. Они хранят всю информацию, помещаемую в БД. В таблицы будет вводится информация, которая может дополняться изменяться и удаляться.

Создание таблицы осуществляется в окне БД. Создать таблицу можно несколькими способами: использовать мастер таблиц, создать таблицу в режиме конструктора таблиц, импортировать таблицу из внешнего файла. В любом случае каждой таблице присваивается определенное имя.

***Создание таблицы в окне конструктора***

Создание таблиц в окне конструктора предоставляет большие возможности. Выбрав Создать таблицу в режиме конструктора откроется окно конструктора. В верхней части окна диалога находится таблица, которая содержит следующие атрибуты: наименование поля, тип данных и описание. Наименование каждого из полей таблицы выбирается произвольно в соответствии с помещаемой в него типом информации.

Наименование поля должно быть уникально может содержать до 64 символов, исключая точку (.), восклицательный знак (!), прямых скобок ([]) и управляющих символов с кодами ASCII – 0-31.

Тип поля определяется типом данных, хранящихся в этом поле. В MS Access допустимы следующие типы: текстовый (до 255 символов), числовой, денежный (8 байт, до 4 знаков после запятой), счетчик (данные не редактируются), дата/время, логический, поле МЕМО (до 64000 символов), поле объекта OLE (размер определяется объемом жесткого диска), мастер подстановок (создает поле, в котором предполагается выбор значений из раскрывающегося списка значений других таблиц).

Каждый из типов поля наделен собственными свойствами, которые отображаются в разделе «Свойства поля» окна конструктора.

Создать структуру таблицы можно следующим образом:

1. В окне конструктора в столбце Имя поля вводится имя поля данных

2. В столбец Тип данных из раскрывающегося списка вводится значение типа данных.

3. Столбец Описание представляет из себя пояснение, которое вы даете своим полям. Это пояснение появляется в строке состояния во время работы с БД.

4. Аналогичным образом вводится описание всех полей таблицы.

5. Завершив ввод структуры, ее надо сохранить, выполнив команду Файл|Сохранить .

***Создание таблицы в режиме таблицы***

Рассмотрим способ создания таблиц, который отличается своей простотой и наглядностью. Приведем последовательность действий, которую предстоит выполнить:

1) Перейдите на вкладку «Таблицы» окна базы данных и нажмите кнопку Создать. 2) В окне диалога «Новая таблица» выберите из списка вариантов значение Режим таблицы и нажмите кнопку ОК.

В результате выполнения этих действий откроется окно диалога «Таблица», содержащее созданную по умолчанию таблицу. Эта таблица содержит 20 столбцов и 30 строк, и этого вполне достаточно для начала. После сохранения этой таблицы, конечно, можно добавить столько строк и столбцов, сколько вам понадобится. 3. Наименования полей таблицы определены по умолчанию, но очень просто присвоить полям новые имена. Для этого нажмите дважды кнопкой мыши на область выбора первого поля (заголовок которого содержит Поле1). Имя поля выделяется и появляется Мигающий курсор. Введите имя первого поля и нажмите клавишу TAB. Аналогично введите остальные имена полей вашей таблицы в следующих столбцах.

4. Теперь заполните несколько строк вашей таблицы, вводя информацию в том виде, в каком она будет вводиться и в будущем. Старайтесь записывать все в одном стиле (например, если первую дату вы записали 10/14/96, то не пишите следующую в виде Ноябрь 3, 1996). Конечно, если MS Access установит неправильный тип данных, вы можете позже его изменить, но и вам желательно стараться вводить все правильно с самого начала.

5. Сохраните таблицу, выполнив команду Файл | Сохранить макет или нажав кнопку Сохранить на панели инструментов, В открывшемся окне диалога «Сохранение» присвойте таблице имя и нажмите кнопку ОК.

6. На запрос о необходимости создания для таблицы первичного ключа нажмите кнопку Да, и MS Access создаст таблицу, удалив лишние строки и столбцы.

7. Теперь убедитесь, что Access выбрал для каждого поля правильные типы данных. Для этого перейдите в окно конструктора таблицы, выполнив команду Вид | Конструктор таблиц. Если вас что-то не устраивает в структуре таблицы, внесите необходимые изменения Модификация структуры таблицы

Для исправления возможных допущенных ошибок MS Access предоставляет необходимые средства. К их числу относятся:

1) Изменение наименования поля и/или его типа .

2) Вставка пропущенного поля

3)Удаление ошибочно введенного поля.

4) Изменение порядка следования полей в таблице

Для модификации структуры таблицы, входящей в базу данных, установите в окне базы данных указатель на модифицируемую таблицу и нажмите кнопку Конструктор.

Изменение наименования поля или его типа осуществляется после установки указателя на данное поле или тип, который надо ввести. Неправильные символы удаляются клавишей Delete. После этого вводятся правильные символы.

Изменение порядка следования полей осуществляется нажатием на область выбора поля и после выделения строки и повторного нажатия левой кнопки мыши перетаскиванием строки (столбца) в нужное место.

Удаление полей таблицы осуществляется клавишей Delete после их выделения.

Добавление нового поля осуществляется командой Вставка | Поле.

***Индексы***

Одним из основных требований, предъявляемых к СУБД, является возможность быстрого поиска требуемых записей среди большого объема информации. Индексы представляют собой наиболее эффективное средство, которое позволяет значительно ускорить поиск данных в таблицах по сравнению с таблицами, не содержащими индексов. В зависимости от количества полей, используемых в индексе, различают простые и составные индексы.

В MS Access допускается создание произвольного количества индексов. Индексы создаются при сохранении макета таблицы и автоматически обновляются при вводе и изменении записей. Можно в любое время добавить новые или удалить ненужные индексы в окне конструктора таблиц.

Требование уникальности индекса в MS Access не является обязательным. Для ускорения поиска требуемой информации могут быть использованы индексы, не являющиеся уникальными

Важной особенностью индексов является то, что можно использовать индексы для создания первичных ключей. Первичный ключ содержит информацию, которая однозначно идентифицирует запись. В этом случае индексы должны быть уникальными. Это означает, что для таблицы, содержащей только одно индексное поле, уникальными должны быть значения этого поля. Для составных индексов величины в каждом из индексных полей могут иметь повторяющиеся значения. Однако индексное выражение должно быть уникальным.

Прежде чем определить первичный ключ в таблице, просмотрите все поля создаваемой вами таблицы. Есть ли хоть одно поле, информация которого была бы уникальна для каждой записи? Даже если использовать полное имя, то есть фамилию, имя и отчество одновременно, оно также не будет неповторимым.

Часто наилучшее решение этой проблемы заключается в том, чтобы каждой записи в таблице поставить в соответствие идентификационный номер. Это и делает MS Access, когда вы предлагаете ему создать первичный ключ. Он создает поле Код с типом данных Счетчик. Это означает, что каждый раз при создании новой записи значение счетчика увеличивается на 1. Этот номер и является первичным ключом для каждой новой записи.

***Создание индекса для одного поля***

Для создания простого индекса используется свойство поля Индексированное поле, позволяющее ускорить выполнение поиска и сортировки записей по одному полю таблицы. Не допускается создание индексов для полей MEMO и полей объектов OLE.

***Создание составного индекса***

Индексы, содержащие несколько полей, следует определять в окне индексов.

1. В окне конструктора откройте таблицу, для которой вы создаете составной индекс. Для этого в окне базы данных установите указатель на данную таблицу и нажмите кнопку Конструктор.

2. Нажмите кнопку Индексы на панели инструментов. На экране откроется окно диалога «Индексы»

3, В открывшемся окне диалога введите имя индекса в поле столбца Индекс в первой пустой строке. В качестве имени индекса можно использовать имя одного из полей, включенных в индекс, или любое допустимое имя.

4. В столбце Имя поля той же строки нажмите кнопку раскрытия списка и выберите первое поле индекса.

5. В столбце Имя поля следующей строки выберите имя следующего поля индекса. (В этой строке поле столбца Индекс следует оставить пустым). Определите таким же образом остальные поля индекса. Индекс может включать до 10 полей.

6. Закончив выбор полей для индекса, нажмите кнопку закрытия окна, расположенную в строке заголовка окна диалога.

По умолчанию задается порядок сортировки По возрастанию. Для сортировки конкретного поля по убыванию выберите в столбце Порядок сортировки строки с выбранным полем значение По убыванию.

***Установка первичного ключа***

Когда MS Access напоминает вам об отсутствии первичного ключа и предлагает его создать, нажмите Да. Access добавит поле код в первой строке описания структуры таблицы.

Если вы хотите установить первичный ключ самостоятельно, нажмите на поле, которое будет в дальнейшем использоваться в качестве первичного ключа. Затем нажмите правую кнопку мыши и выберите пункт всплывающего Первичный ключ. В области выбора поля, которое будет использоваться как ключ, появится маленькая пиктограмма с изображением первичного ключа. Для установки первичного ключа также можно использовать кнопку Ключевое поле на панели инструментов.

Определение отношений между таблицами В MS Access вы можете устанавливать постоянные отношения между таблицами, которые будут поддерживаться при создании форм, отчетов и запросов.

Устанавливая связи между двумя таблицами, вы выбираете поле, которое содержит одну и ту же информацию. Чаще всего вы будете связывать первичный ключ одной таблицы с совпадающими полями другой таблицы.

Поля, с помощью которых устанавливается связь между двумя таблицами, могут иметь различные имена, но удобнее использовать совпадающие имена.

Из существующих типов связей тип «один-ко-многим» наиболее важен, поэтому самое пристальное внимание следует уделить ему. В отношении «один-ко-многим» главной таблицей является таблица, которая содержит первичный ключ и составляет часть «один» в отношении «один-ко-многим».

Внешний ключ — это поле (или поля), содержащее такой же тип информации в таблице со стороны «много» в отношении «один-ко-многим», которую называют подчиненной таблицей.

***Создание связей между таблицами в MS Access***

Создание связей между таблицами в MS Access осуществляется в окне диалога «Схема данных». Для определения связей между таблицами необходимо выполнить следующие действия:

1. Откройте окно диалога «Схема данных», выполнив команду Сервис | Схема данных или нажав кнопку Схема данных на панели инструментов. На экране откроется окно диалога «Схема данных».

Перед определением связей между таблицами необходимо предварительно закрыть все открытые таблицы. Не допускается создание или удаление связей между открытыми таблицами.

2. Добавьте в это окно диалога последовательно две связываемые таблицы. Для этого выполните команду Связи | Показать таблицу или нажмите кнопку Добавить таблицу на панели инструментов. На экране откроется окно диалога «Добавление таблицы»

3. В списке таблиц выделите первую добавляемую таблицу и нажмите кнопку Добавить. Затем выберите вторую добавляемую таблицу и также нажмите кнопку Добавить. Затем нажмите кнопку Закрыть для закрытия окна диалога «Добавление таблицы». В окне диалога «Схема данных» появились две связываемые таблицы.

4. Для связывания таблиц выберите поле в первой связываемой таблице и переместите его с помощью мыши на соответствующее поле второй таблицы. Для связывания сразу нескольких полей выберите эти поля при нажатой клавише Ctrl и переместите во вторую таблицу группу выделенных полей.

В большинстве случаев связывают ключевое поле (представленное в списке полей полужирным шрифтом) одной таблицы с соответствующим ему полем внешнего ключа (часто имеющим то же имя) во второй таблице. Связанные поля не обязательно должны иметь одинаковые имена, однако, они должны иметь одинаковые типы данных (из этого правила существует два исключения) и иметь содержимое одного типа. Кроме того, связываемые поля типа Числовой должны иметь одинаковые значения свойства Размер поля. Исключениями из этого правила являются поля счетчика с последовательной нумерацией, которые могут связываться с числовыми полями размера Длинное целое, а также поля счетчика с размером Код репликации, связываемые с полями типа Числовой, для которых также задан размер Код репликации. 5. На экране откроется окно диалога «Связи» . В данном окне диалога проверьте правильность имен связываемых полей, находящихся в столбцах. При необходимости выберите другие имена полей. Затем нажмите кнопку Создать. Вы вернетесь в окно диалога «Схема данных».

Тип создаваемой связи зависит от полей, которые были указаны при определении связи;

- Отношение «один-ко-многим» -создается в том случае, когда только одно из полей является ключевым или имеет уникальный индекс.

- Отношение «один- к-одному» -создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы. - Связь с отношением «многие-ко-многим» -фактически представляет две связи с отношением «один-ко-многим» через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, которые являются полями внешнего ключа в двух других таблицах.

В окне диалога «Схема данных» при переносе поля, не являющегося ключевым или не имеющего уникального индекса, на другое поле, которое также не является ключевым или не имеет уникального индекса, создается неопределенное отношение. В запросах, содержащих таблицы с неопределенным отношением, MS Access по умолчанию создает линию объединения между таблицами, но условия целостности данных при этом не поддерживаются, и нет гарантии уникальности записей в любой из таблиц.

В окне диалога «Схема данных» можно также выполнять следующие действия:

1) Изменить структуру таблицы. 2) Изменить существующую связь. 3) Удалить связь. 4) Удалить таблицу из окна диалога «Схема данных». 5) Вывести на экран все существующие связи или связи только для конкретной таблицы. 6) Определить связи для запросов, не задавая условия целостности данных.

***Связывание двух полей одной таблицы***

Иногда возникает необходимость в определении поля с подстановкой значений из той же таблицы. Для связывания поля таблицы с другим полем той же таблицы дважды добавьте эту таблицу в окно диалога «Схема данных» и создайте требуемую связь, соединив поля линией связи.

***Создание между таблицами отношения «многие-ко-многим»***

Рассмотрим создание между таблицами отношения «многие-ко-многим». В MS Access отношение «многие-ко-многим» представляет две связи с отношением «один-ко-многим» через третью таблицу, ключ которой состоит, по крайней мере, из двух полей, являющихся полями внешнего ключа в двух других таблицах. Рассмотрим создание такой связи:

1) Создайте таблицы, между которыми требуется определить связь с отношением «многие-ко-многим».

2) Создайте третью (связующую) таблицу с полями, описание которых совпадает с описанием ключевых полей в каждой из двух связываемых таблиц. В этой таблице ключевые поля выполняют роль внешнего ключа. Другие поля в связующую таблицу можно добавлять без ограничений.

3) Определите в связующей таблице ключ, содержащий все ключевые поля двух связываемых таблиц.

4) Определите связи с отношением «один-ко-многим» между каждой из двух таблиц и связующей таблицей.

***Изменение структуры таблицы в окне диалога «Схема данных»***

При создании связи в окне диалога «Схема данных» может возникнуть необходимость в изменении структуры таблицы. При этом вы можете не покидать окна диалога, а внести нужные изменения в структуру таблицы непосредственно в окне диалога:

1. Находясь в окне диалога «Схема данных», установите указатель мыши на модифицируемую таблицу.

2. Нажмите правую кнопку мыши и выберите из контекстного меню команду Конструктор таблиц.

3. Внесите в структуру таблицы необходимые изменения.

4. Закончив внесение изменений, нажмите кнопку закрытия окна в строке заголовка окна диалога. В ответ на запрос о сохранении изменений выберите Да для сохранения изменений и возвращения в окно диалога «Схема данных». Изменение существующей связи

Прежде чем приступить к изменению связей между таблицами, закройте все открытые таблицы. Access не допускает изменение связей между открытыми таблицами. Затем выполните следующую последовательность действий:

1. Находясь в окне базы данных, нажмите кнопку Схема данных на панели инструментов,

2. Если таблицы, связи между которыми требуется изменить, не отображаются в окне диалога «Схема данных», нажмите кнопку Добавить таблицу на панели инструментов, установите указатель на имя нужной таблицы и дважды нажмите кнопку мыши. После этого нажмите кнопку Закрыть,

3. Установите указатель на линию связи, которую требуется изменить, и дважды нажмите кнопку мыши.

4. В открывшемся окне диалога «Связи» внесите нужные изменения и нажмите кнопку ОК.

***Удаление связи***

Нажмите кнопку Схема данных на панели инструментов, установите указатель на линию связи, которую требуется удалить, и выделите ее, нажав кнопку мыши. Нажмите клавишу Delete. Когда Access предложит вам подтвердить удаление связи, нажмите кнопку Да. Удаление таблицы из макета схемы данных

1. Откройте окно диалога «Схема данных». 2. Выберите таблицу, которую требуется удалить из данного окна, и нажмите клавишу Delete. Таблица будет удалена из макета схемы данных вместе с определенными для нее связями. Данная операция изменяет только макет в окне диалога «Схема данных». И таблица, и ее связи будут по-прежнему сохраняться в базе данных.

***Определение условий целостности данных***

Целостность данных является одним из самых важных требований, предъявляемых к базам данных. Для задания условий целостности данных служат установленные между таблицами отношения. Условиями целостности данных называют набор правил, используемых в МS Access для поддержания связей между записями в связанных таблицах. Эти правила делают невозможным случайное удаление или изменение связанных данных Условия целостности данных выполняются при следующих условиях:

1) Связанное поле главной таблицы является ключевым полем или имеет уникальный индекс.

2) Связанные поля имеют один тип данных.

3) Обе таблицы принадлежат одной базе данных MS Access.

Если таблицы являются присоединенными таблицами, то они должны быть таблицами МS Access. Невозможно определить условия целостности данных для присоединенных таблиц из баз данных других форматов.

При определении условия целостности данных действуют следующие ограничения:

- Невозможно ввести в поле внешнего ключа связанной таблицы значение, не содержащееся в ключевом поле главной таблицы. Однако возможен ввод в поле внешнего ключа пустых значений, показывающих, что записи не являются связанными. - Не допускается удаление записи из главной таблицы, если существуют связанные с ней записи в подчиненной таблице.

- Невозможно изменить значение ключевого поля в главной таблице, если имеются записи, связанные с этой записью. Например, невозможно удалить код сотрудника в таблице Сотрудники, если в таблице Заказы имеются заказы, относящиеся к данному сотруднику.

Определение целостности данных предполагает выполнение следующих действий:

1) В окне диалога «Схема данных» два раза мышью на линии связи между двумя таблицами. Откроется окно диалога «Связь».

2) Установите флажок Обеспечение целостности данных и нажмите ОК.

Для того чтобы преодолеть ограничения на удаление или изменение связанных записей, сохраняя при этом целостность данных, следует включить режимы каскадного обновления и каскадного удаления. При установленном флажке Каскадное обновление связанных полей изменение значения в ключевом поле главной таблицы приводит к автоматическому обновлению соответствующих значений во всех связанных записях. При установленном флажке Каскадное удаление связанных записей удаление записи в главной таблице приводит к автоматическому удалению связанных записей в подчиненной таблице.

***Использование каскадных операций***

Обратимся к окну диалога «Связи». При установке опции Обеспечение целостности данных вам стали доступны опции Каскадное обновление связанных полей и Каскадное удаление связанных нолей. При выборе этих опций, MS Access выполняет изменения в связанных таблицах таким образом, чтобы сохранить целостность данных, даже если вы изменяете значения ключевых полей или удаляете запись в главной таблице.

Каскадные изменения Может возникнуть необходимость внести изменения в ключевое поле записи, связанной по типу «один-ко-многим». Если вы не установили опцию Каскадное обновление связанных полей, то MS Access не позволит этого сделать. Вместо изменения MS Access выдаст вам предупреждение о том, что вы нарушите целостность данных, если попытаетесь осуществить это изменение. Это является нарушением потому что это поле является первичным ключом в связи «один-ко-многим», и изменяемое значение появляется, по крайней мере, в одной записи в поле связанной с ним таблицы. Если же вы установили Каскадное обновление связанных полей, проблем не возникнет. Вы можете ввести изменения. MS Access выполнит все необходимые изменения в связанных таблицах автоматически.

***Каскадные удаления***

При попытке удалить запись в ключевом поле записи, связанной по типу «один-ко-многим» при не установленной опции Каскадное удаление связанных полей, Access выдаст сообщение об ошибке.

Однако, если выбрана опция Каскадное удаление связанных полей, вы можете удалить запись из главной таблицы. Связанные с нею записи в подчиненных таблицах будут также автоматически удалены, соблюдая ,таким образом, правила целостности данных. Другими словами, если выбрана опция Каскадное удаление связанных полей то все связанные поля из подчиненной таблицы удаляются, как только удаляется запись из главной таблицы.

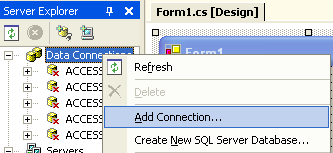
**2.5 Создание соединение с базой данных**

При создании клиент - серверного приложения необходимо создать подключение к базе данных клиентского приложения. В среде *Visual Studio* подключение к базе данных можно организовать следующими способами:

**1 способ**. Создадим подключение к базе данных Access в обозревателе серверов.

Чтобы создать подключение в обозревателе серверов

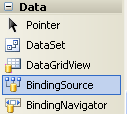
1. В обозревателе серверов (Server explorer) щелкните правой кнопкой мыши Подключения данных (Data Connection) и выберите команду Добавить подключение (Add connection).



*Рисунок 3 – Контекстное меню создания подключения*

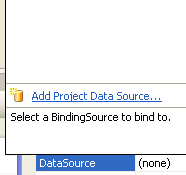
1. В появившемся окне выберитеMicrosoft Access Database File. Нажмите ОК.
2. В следующем окне нажмите кнопку Browse для выбора базы данных, к которой необходимо подключиться.
3. Если необходимо введите Имя пользователя, используемый для проверки подлинности при входе в источник данных и введите Пароль, используемый для проверки подлинности при входе в источник данных.
4. Нажмите кнопку OK.

После выполнения выше изложенных шагов, перетащите из обозревателя серверов на поле формы или конструктора объект BindingSource.



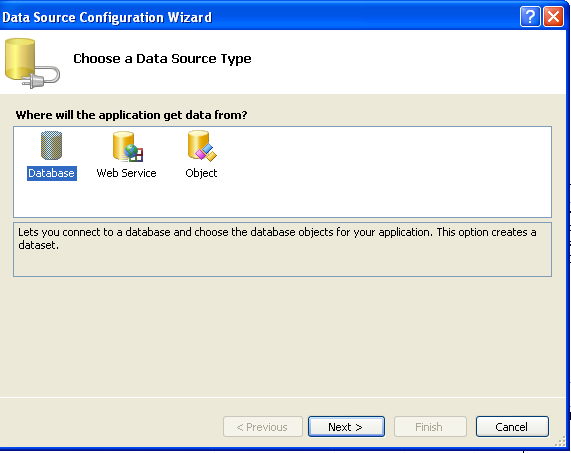
*Рисунок 4- Контекстное меню добавления в форму объекта BindingSource*

В нижней части формы появится соответствующий значок, щелкните правой кнопкой мыши по данному значку, выберите строку Properties. В открывшемся окне щелкните в строке DataSource .



*Рисунок 5 - Контекстное меню добавления источника данных*

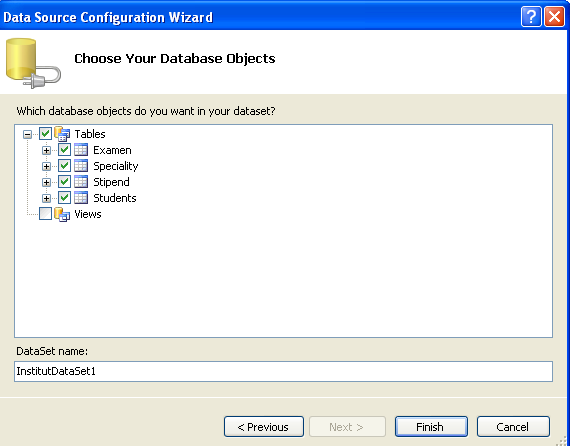
В открывшемся окне выберите **Add Project data Source.**



*Рисунок 6 – Диалоговое окно подключения к базе данных*

Затем выберите **Database** нажмите Next, укажите в строке путь к используемой базе данных, обычно он записан по умолчанию и нажмите Next.

Далее необходимо выбрать объекты базы данных, которые войдут в объект DataSet.



*Рисунок 7- Диалоговое окно выбора таблиц подключаемой базы данных*

И нажмите Fi*n*ish.

### В *Visual Studio* существует объект *Connection*. Объект *Connection* служит для соединения с базой данных, нуждается в строке соединения для указания пути к СУБД и входа в систему. Свойства класса *Connection* показаны в таблице. *OleDbConnection*, *SqlConnection*, *OdbcConnection* – наследники класса *Connection*, специфические для провайдеров *OleDb*, *MS* *SQL*, *ODBC* соответственно.

Все свойства, кроме ConnectionString, только для чтения.

### ***Таблица 1 - Свойства объекта Connection***

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| *DataSource* | Путь к базе данных в файловой системе при использовании *Oledb*, имя экземпляра базы сервера при использовании *SqlConnection* |
| *Database* | Возвращает имя базы данных, используемой в объекте *Connection* после открытия |
| *State* | Возвращает текущее состояние соединения. Возможные значения – *Broken* – «Соединение с источником данных разорвано», значение 16  *Closed* – « Соединение закрыто », значение 0  *Connecting* – « Идет процесс подключения », значение 2  *Executing* – « Соединение находится в процессе выполнения команды 4 , *Fetching* – «Объект соединения занят выборкой данных» значение 8  Open – «Соединение открыто» значение 1. |
| *ConnectionString* | Строка соединения с СУБД |

**2 способ**. Создадим подключение программно. Для этого перейдите в программный код формы Form1.cs. На панели **Solution Explorer** щелкните правой клавишей на **Form1.cs**, выберите **View Code**.

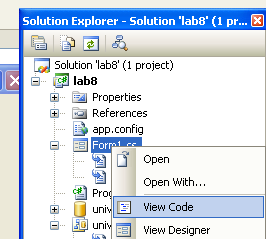


Рисунок 8- Контекстное меню открытия программного кода формы

Для подключение к базе данных Access необходимо добавить пространство имен

using System.Data.OleDb;

Затем в классе формы объявить строковую переменную, содержащую информацию о провайдере и пути к базе данных в нашем примере D:\Institut.mdb.

namespace primer\_soedineniya

{

public partial class Form1 : Form

{

string InstConnectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + @"D:\Institut.mdb";

OleDbConnection con;

public Form1()

{

InitializeComponent();

con = new OleDbConnection(InstConnectString);

con.Open(); // открываем соединение

}}}

**2.6 Использование форм для ввода**

**и редактирования данных**

Форма позволяет объединить поля в группы по определенным признакам. Это облегчает восприятие информации.

Для создания формы в MS Access на начальном этапе необходимо выполнить следующие действия:

1. Открыть окно БД.

2. Перейти на вкладку Формы.

3. Нажать кнопку Создать. Выбрать нужный вариант. Ок.

Можно воспользоваться мастером форм и проделать следующие операции:

1. Открыть окно БД. 2. В окне перейти на вкладку «Таблицы». 3. Указать на таблицу, для которой создается экранная форма. 4. Выполнить команду Вставка|Автоформа или нажать кнопку Новый объект и из списка выбрать опцию Автоформа. На экране появится готовая форма.

***Создание формы в конструкторе форм***

В данном разделе обсудим создание формы с помощью конструктора форм. Процесс по созданию формы может включать в себя все или часть из приведенных ниже процедур:

- Размещение текста.

- Размещение полей.

- Создание управляющих кнопок. - Размещение линий, прямоугольников и рисунков.

- Установка цвета объектов формы.

- Перемещение объектов формы.

Наиболее частым источником ошибок в базе данных является ввод пользователем неправильных данных. Тщательно разработав форму, которую пользователи будут использовать для ввода, редактирования или просмотра данных, можно предотвратить возникновение большого количества ошибок. При создании форм учитывайте следующее

- Если пользователи привыкли к использованию стандартных бланков, формы должны выглядеть так же, как эти бланки. Необходимость каждый раз искать местонахождение информации никогда не способствовала безошибочному вводу данных.

- Для группировки элементов управления используйте линии и прямоугольники. При этом пользователь будет вводить близкие по смыслу данные (такие, как вся идентификационная информация по товару или полный домашний адрес) вместе.

- Не концентрируйте элементы управления в какой-либо части формы. Это затрудняет чтение информации. Пользователь должен наглядно видеть, с каким элементом управления он работает в данное время.

- Пояснительный текст формы должен быть максимально информативным и иметь минимальную длину. Используйте условия правильности ввода данных, что поможет предотвратить ввод неверных данных.

- Используйте маски ввода для ввода стандартизированной информации. Например, если вам известно точное количество символов, необходимых для ввода номера телефона, используйте маску ввода.

- Для облегчения восприятия чисел используйте форматы ввода.

***Настройка формы***

Для создания формы войдите в окно конструктора форм. Для этого, находясь на вкладке «Формы» БД нажмите кнопку Создать и в открывшемся окне диалога «Новая форма» нажмите кнопку ОК. На экране откроется окно конструктора форм.

Первое, что нужно сделать — определить свойства самой формы как объекта. Каждая форма имеет свойства, определяющие расположение ее в основном окне MS Access, размер, заголовок, стиль и некоторые другие параметры.

1. Для определения или изменения стиля формы, находясь в конструкторе форм, выполните команду Формат|Автоформат или нажмите кнопку Автоформат на панели инструментов На экране откроется окно диалога «Автоформат» . Выберите из списка стили и посмотрите на их внешний вид в окне просмотра После того как вы нашли нужный стиль, нажмите кнопку ОК.

2. Для задания размеров формы используйте мышь.

3. Установите указатель мыши в нижний правый угол формы.

4. При этом курсор примет вид двунаправленной стрелки.

5. Нажмите кнопку мыши и установите требуемый размер формы.

6. Для использования формы только для просмотра, установите для свойства Разрешить изменение значение Нет. Данная установка не позволит пользователю редактировать содержимое элементов управления, связанных с таблицей или запросом.

7. Если установить для свойства Разрешить добавление и Разрешить добавление значения Нет, то становятся недоступными команды Вставка|Запись, Записи|Ввод данных и Правка|Удалить запись.

***Размещение текстовой информации***

Размещение текста в экранной форме осуществляется с помощью инструмента Надпись, который находится на панели элементов. Под текстом понимается любая текстовая информация: заголовки, поясняющая информация. Для размещения текста в форме выполните следующие действия:

1. Выберите инструмент Надпись на панели элементов. Если данная панель отсутствует на экране, для ее отображения выполните команду Вид |Панели инструментов и в списке панелей инструментов установите опцию Панель элементов или нажмите кнопку Панель элементов на панели инструментов.

2. Установите указатель мыши на место предполагаемого расположения текстового объекта и введите текст.

3. Закончив ввод текста, нажмите клавишу Enter.

4. Выделите созданный объект.

5. Используя панель инструментов «Форматирование» или окно свойств созданного объекта, задайте для него тип шрифта, размер, цвет шрифта, цвет рамки, тип и цвет фона и другие параметры оформления. Все свойства вы можете определить в окне свойств. Некоторые, наиболее часто используемые, можно задавать и с помощью панели «Форматирование»

***Размещение полей ввода***

Следующим шагом в создании формы является добавление в нее полей различных типов. Наиболее простым типом поля является поле ввода. Для размещения поля ввода в форме выполните следующие действия:

1. Выберите инструмент Поле на панели элементов.

2. Нажмите мышью место, в котором вы предполагаете разместить поле. В форме появится связанный объект, состоящий из поля ввода и его надписи. Выделите поле ввода и откройте для него окно свойств.

3. Чтобы связать созданное поле с полем таблицы или запроса выберите свойство Данные вкладки «Данные». В поле ввода свойства воспользуйтесь кнопкой раскрытия списка и выберите из списка всех полей открытой таблицы поле, которое хотите добавить в форму. Если вы хотите связать поле с выражением, нажмите кнопку Построить. Создайте необходимую формулу с помощью построителя выражений.

4. Используя панель инструментов форматирования или окно свойств поля ввода, задайте для него тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры.

5. Если вы создаете поле, информация из которого должна быть доступна только для чтения, необходимо установить значение свойства Доступ равным Нет.

6. Свойство Всплывающая подсказка вкладки «Другие» позволяет создать краткое пояснение к полю, которое будет появляться на экране, когда указатель установлен на поле и удерживается на нем некоторое время.

7. Для определения значения поля по умолчанию задайте свойство Значение по умолчанию.

8. Выделите надпись к полю ввода и откройте для него окно свойств.

9. Чтобы задать текст надписи, выберите свойство Подпись вкладки «Макет» и в поле ввода свойства введите текст надписи к полю.

10. Используя панель инструментов «Форматирование» или окно свойств, задайте для надписи тип шрифта, размер, цвет шрифта, цвет рамки, тип, цвет фона и другие параметры.'

***Скрытие поля***

Используя окно свойств поля, вы можете сделать поле невидимым (и поле ввода и надпись к нему), так что его не будет видно в режиме просмотра формы. Аналогичная возможность существует в режиме просмотра таблиц, в котором для скрытия столбца таблицы используется команда Формат | Скрыть столбцы. В форме для скрытия поля и надписи к нему необходимо для свойства Вывод на экран вкладки «Макет» поля задать значение Нет.

Отображение сообщений в строке состояния При создании таблиц упоминалось, что информация, помещенная в столбец «Описание», в режиме просмотра формы выводится в строке состояний, когда курсор находится в данном поле. Если этот столбец пуст, то сообщение не выводится.

Вы можете изменить сообщение в форме с помощью свойства Текст строки состояния вкладки «Другие». Для этого откройте окно свойств поля и введите в поле ввода данного свойства текст, который будет появляться в строке состояния.

***Изменение формата отображения дат и чисел***

Для задания формата отображения дат и/или чисел в форме используется свойство Формат поля вкладки «Макет».

***Размещение списка и раскрывающегося списка***

При вводе информации в таблицу во многих случаях удобнее выбирать повторяющееся значение из списка, чем каждый раз вводить одно из двух-трех значений. Кроме того, выбор из списка позволяет быть уверенным, что введенное значение является допустимым.

Элементы панели элементов Список и Поле со списком предназначены для отображения на экране элементов списка. Выбор одного из этих двух типов объектов определяется требованиями пользователя и наличием свободного места в форме.

При небольшом числе элементов списка удобнее использовать Список, который всегда раскрыт на экране. Когда список большой, используйте элемент Поле со списком. Он занимает меньше места на экране, поскольку список раскрывается на экране только при нажатии кнопки раскрытия списка. Список и Поле со списком состоят из строк данных. Строки содержат один или несколько столбцов, определяемых с помощью следующих средств:

- Список фиксированных значений.

- Список полей.

- Значения поля таблицы или запроса.

Тип источника данных определяется свойством Тип источника на вкладке «Данные».

Объекты типа списка и поля со списком имеют дополнительные свойства, которые отсутствовали у ранее рассмотренных объектов Надпись и Поле.

Для размещения списка, поля со списком, группы параметров и ряда других элементов управления вы можете использовать мастера. Для этого вам необходимо установить режим использования мастера и выбрать соответствующий инструмент на панели элементов. После указания места расположения элемента запускается соответствующий мастер.

Размещение флажка Для индикации состояния, которое может иметь только одно из двух допустимых значений, используются флажки. Они могут использоваться по одному или группами. Установленный флажок будет соответствовать значению «постоянный», а снятый — значению «временный».

Рассмотрим последовательность ваших действий при создании флажка для редактирования поля , которое имеет тип Логический.

1. Откройте форму в режиме конструктора и выберите инструмент Флажок на панели элементов.

2. Нажмите мышью место предполагаемого размещения элемента в форме.

3. Откройте окно свойств размещенного в форме флажка.

4. Чтобы связать созданное поле с полем таблицы, выберите свойство Данные вкладки «Данные». В поле ввода значения свойства воспользуйтесь кнопкой раскрытия списка и выберите поле из списка полей таблицы.

5. Выделите надпись созданного флажка и в его окне свойств скорректируйте свойство Подпись вкладки «Макет». Просмотрите форму в режиме формы.

Для создания объектов логического типа используются элементы Переключатель и Выключатель панели элементов, соответственно. Создание кнопок управления

Кнопки используются в формах для выполнения определенного действия или ряда действий. Например, можно создать в форме кнопку, открывающую другую форму, или создать набор кнопок для перемещения по записям таблицы, если вас не устраивают стандартные средства перемещения, предусмотренные в форме. Для того чтобы кнопка выполняла какое-либо действие, необходимо создать макрос или процедуру обработки события и связать их со свойством кнопки Нажатие кнопки.

В MS Access предусмотрено создание более 30 разных кнопок, что избавляет пользователя от необходимости самостоятельно разрабатывать макросы. Достаточно лишь воспользоваться мастером по созданию кнопки. Так для создания кнопки, открывающей другую форму нужно выполнить следующие действия:

1. Установите режим использования мастера на панели элементов и выберите инструмент Кнопка на панели элементов.

2. Установите указатель мыши на место в форме, в котором вы предполагаете расположить кнопку и нажмите кнопку мыши. Запускается мастер создания кнопки конструктора форм.

3. В первом окне диалога расположены два списка: Категории и Действия. При перемещении по списку Категории список Действия обновляется. Рассмотрите внимательно содержимое этих списков. Список Категории содержит наборов действий, а список Действия — действия, которые будут выполняться при нажатии на данную кнопку. Выберите нужное значение из списка Действия и нажмите кнопку Далее.

4. Во втором окне диалога мастера из списка форм базы данных выберите форму, которая будет открываться при нажатии на кнопку.

5. На следующем шаге определяется тип отображаемой информации на кнопке: текстовая или графическая. Если вы выбрали текстовую информацию, то в поле ввода рядом с опцией Текст введите текст, отображаемый на кнопке. При размещении графической информации установите опцию Рисунок и нажмите кнопку Обзор для открытия окна диалога «Выбор рисунка», в котором выберите графическое изображение. После чего переходите к следующему шагу.

6. На заключительном шаге работы мастера задается имя создаваемого объекта Введите имя, затем нажмите кнопку Готово. Кнопка, при нажатии на которую открывается другая экранная форма, создана.

***Добавление рисунка или другого объекта в форму***

Способ вставки рисунка или объекта зависит от того, какой объект предполагается создать: присоединенный или свободный. Присоединенный объект хранится в таблице. При переходе к новой записи в форме или отчете отображается другой объект. Например, этот способ удобен для хранения фотографий сотрудников фирмы. Свободный объект является частью структуры формы. При переходе к новой записи объект не изменяется. Например, таким способом в форме или отчете сохраняют эмблему фирмы, созданную в приложении MS Paint.

***Размещение графического изображения***

В экранные формы можно вставлять различные графические изображения, позволяющие облегчать восприятие информации. Для этого используется инструмент Рисунок панели элементов. При размещении данного элемента в форме открывается окно диалога «Выбор рисунка» . Используя список Тип файла, укажите тип используемого изображения. Выбрав имя вставляемого рисунка, нажмите кнопку ОК.

***Размещение объекта типа OLE***

Примером использования данного типа объектов является расположение в форме поля Фотография таблицы Сотрудники. В данном поле таблицы хранятся фотографии всех сотрудников фирмы. Для присоединенного объекта в форме выполните следующие действия:

1. Для добавления графического поля типа OLE в форму выберите инструмент Присоединенная рамка объекта на панели элементов.

2. Нажмите мышью место, где вы хотите добавить поле. Удерживая кнопку мыши в нажатом состоянии, переместите указатель по диагонали так, чтобы получилась рамка требуемого размера.

3. Откройте окно свойств созданного объекта.

4. Чтобы связать созданное поле с полем таблицы, выберите свойство Данные. В поле ввода свойства воспользуйтесь кнопкой раскрытия списка и из списка полей открытой таблицы Сотрудники выберите поле типа OLE Фотография.

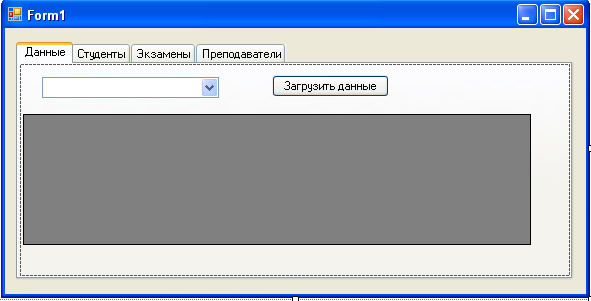
5. Просмотрите форму в режиме формы. Если рисунок не помещается в рамке целиком, вернитесь в режим конструктора и увеличьте размер поля.

***Использование линий и прямоугольников***

Линии и прямоугольники в экранной форме применяются дня улучшения внешнего вида формы и восприятия информации, а также для объединения объектов в логические группы. Для добавления в экранную форму линий используется инструмент Линия панели элементов. Чтобы нарисовать вертикальную или горизонтальную линию на экране выберите данный инструмент, нажмите мышью то место, где должна начинаться линия, и переместите указатель до получения линии нужной длины. Настройка параметров линии осуществляется с помощью ее свойств.

Для добавления в экранную форму прямоугольников используется инструмент Прямоугольник панели элементов. Чтобы нарисовать прямо- угольник выберите данный инструмент, нажмите мышью место расположения одного из углов прямоугольника, и переместите указатель до получения прямоугольника нужного размера.

Для создания формы в Visual Studio например следующего вида:

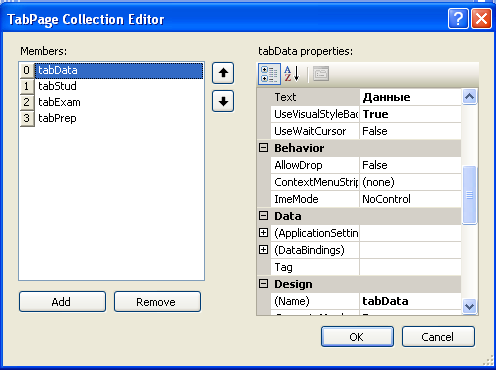


*Рисунок 9 – Окно создаваемой формы*

1) Необходимо перейти в дизайнер формы. Form1.cs [Design].

2) Добавить с панели инструментов **ToolBox** объект **TabControl** и расположить его на форме.

3) В окне свойств (Properties) перейдите на строчку **TabPages**  нажмите на  . В окне tabPagesCollection Editor нажмите на кнопку ADD для добавления вкладок, измените свойства Name и Text. Для вкладки Данные Name = tabData, Text = Данные, на второй вкладке Студенты Name = tubStud, Text = Студенты, на третьей вкладке Name = tubExam, Text = Экзамены, на четвертой вкладке Name = tubPrep, Text = Преподаватели.



*Рисунок 10 – Диалоговое окно мастера создания вкладок*

На вкладке **Данные** добавьте следующие объекты с панели инструментов, изменив их свойства:

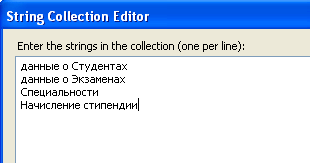
******Этот объект используется для представления выводимых данных из таблицы

|  |  |
| --- | --- |
| Name | datGrid |

Объект ComboBox

|  |  |
| --- | --- |
| Name | cmbTable |

Выберите свойство Items (Collection) и введите следующие пункты списка:

**

*Рисунок 11 – Диалоговое окно создания пунктов меню*

Добавьте кнопку (Button)

Name= butLoad, Text = «Загрузить данные».

Затем перейдите на вкладку **Студенты** и добавьте следующие объекты с панели инструментов, изменив их свойства:

******

|  |  |
| --- | --- |
| Name | datGridStud |

На вкладку Экзамены добавьте две кнопки:

***Кнопка Добавить запись***

|  |  |
| --- | --- |
| Name | butInsExam |
| Text | Добавить запись |

***Кнопка Загрузить данные***

|  |  |
| --- | --- |
| Name | butUpdExam |
| Text | Загрузить данные |

Реализуем следующие функции приложения:

1. Выбрав пункт меню и нажав кнопку Загрузить данные, в объекте dataGridView отображаются данные выбранной таблицы.

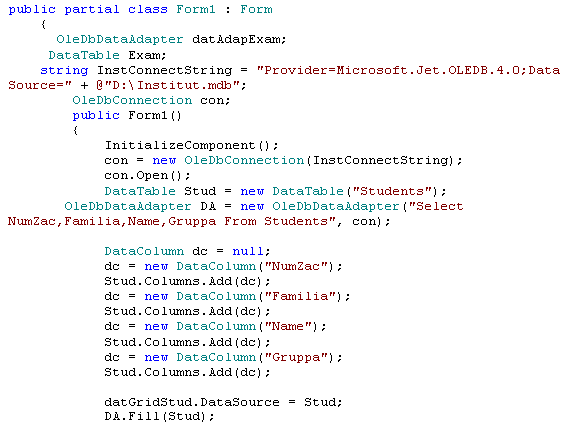
Дважды щелкните по кнопке «Загрузить данные» и в обработчике введите следующий код. А также добавьте объявление переменной OleDbDataAdapter datAdapTable;

перед public Form1().



2. Возможность отображения в объекте **datGridStud** не всех данных из таблицы Студенты, а только №зачетки, фамилии, имени и группы.

Измените программный код следующим образом:



Таким образом, для отображения выборочных данных в клиентском приложении необходимо:

1. Создать соединение. OleDbConnection con = new OleDbConnection(InstConnectString);
2. Открыть соединение.
3. Создать объект DataTable для помещения в него данных таблицы Студенты.
4. Создать адаптер данных, так как база данных создана в Access, используется OleDBProvider, поэтому OleDbDataAdapter.Указать какие данные поместить в адаптер, например как в данном случае в виде SQL –запроса и указать соединение.
5. Создать схему представления данных с помощью объектов DataColumn.
6. Заполнить объект Stud данными из адаптера DA.
7. Указать объекту dataGridView источник данных для отображения, в данном случае это объект DataTable Stud.

**2.7 Реализация запросов**

Практическое использование базы данных основано на выборке из исходных таблиц определенного сорта информации, удовлетворяющей определенным критериям. Критерии могут определяться сочетанием ряда условий. Для решения задач, связанных с выборкой данных, предназначены мастера и конструкторы запросов. Под запросом обычно понимается вопрос, сформулированный к БД.

Языки запросов важны и с точки зрения приложений СУБД, например, в АРМ. Объяснение этому достаточно естественно. Разработанные АРМ имеют, как правило, четко определенное меню, т.е. отработанные формы запросов, оформленные в систему выбора (да/нет). Поэтому необходимо применять языки запросов для реализации диалога в командах, что требует квалификации программиста. Языки запросов снимают и эту проблему, но порождаются две других.

Во-первых, для формулировки запросов необходимо знать язык запросов. Если в QBE этому обучиться достаточно просто, благодаря развитому экранному представлению примера запроса в среде WINDIWS и QBE, то в SQL, представляющий своеобразный язык, требуется определенная подготовка, может быть даже сравнимая с изучением языка одной из СУБД.

Во-вторых, и QBE, и SQL требует дополнительных ресурсов от ЭВМ, что не всегда реально в медленно обновляющемся парке ЭВМ. И, наконец, еще одно замечание о включении языков запросов QBE и SQL в уже действующие APМы или СУБД. А, следует ли включать такие мощные средства из-за 10-20 нестандартных запросов, поступающих, например, в течение полгода обслуживания СУБД или АРМов. Может сложиться ситуация, что разработчик сформулирует вывод: «Язык запросов необходимо включить в АРМ, но ни оперативной, ни дисковой памяти не хватит». В этом случае, надеясь, что разработчик знает систему команд СУБД, он может сам написать транслятор для языка запросов, основанного на элементах реляционной алгебры.

К базе данных БД могут быть предъявлены различные вопросы на поиск информации, удовлетворяющие тем или иным условиям, или ограничениям. В результате поиска из набора файлов, хранящих объектные или связные отношения, будет получено отношение, которое может быть сохранено в виде файла, либо выведено пользователю на экран дисплея или принтер.

Формулировки вопросов могут быть представлены разнообразными формами запросов, простыми и сложными, к одному файлу или к нескольким и т. д. Однако все запросы могут быть структурированы, а следовательно, формализованы, что и представляет возможность их реализации в языковой (программной среде).

Рассмотрим наиболее простые типы запросов.

*Обычный запрос*. А(Е) = ? Каково значение атрибута А объекта Е ( в понятие атрибут входит имя атрибута и значения поля (столбца) в реляционном отношение. Объектом называем строку (запись) в отношении. Аналогичное можно распространить на плоский файл). Пример. Определить заработок у доцента института. Здесь значение атрибута А - заработок, объект - доцент. Доцент в свою очередь является значением атрибута должность (доцент, профессор, старший преподаватель, преподаватель, ассистент).

*Инвертированный запрос*. А(?)= V. Какой объект Е имеет значение атрибута А равное V . Пример. Какие должности преподавательского состава имеют заработок (=) равный V.

*Атрибутный запрос*. ?(Е)=V (<=, =, ≠, >=). Какой атрибут объекта Е имеет значение V. Пример. Найти имя атрибута имеющего значение 10000р у должности доцента.

*Запрос на поиск информации для заданного объекта* ? (Е) = ?(<=, =, ≠, >=) Выбрать всю информацию атрибута и их значение для объекта Е. Пример. Выдать все атрибуты и их значения известных для профессии «доцент».

*Запрос на поиск значений заданного атрибута для всех объектов*. А(?) = ? Для заданного атрибута А найти все значения для всех объектов. Выдать величину зарплаты для всех должностей.

*Запрос на поиск всех объектов и всех атрибутов имеющих значение* ?(?) =V (<=, =, ≠, >=). Найти все объекты и все атрибуты, для которых задано значение V. Пример. Найти все атрибуты и все объекты, для которых значение неизвестного атрибута равно 100000р.

*Сложный запрос на одном отношении*: Запрос основывается на перечисленных типах, усложнением структуры запроса, логическими операндами «или» = «or», «и» = «AND» и вычислительными операциями. Пример. Перечислить кафедры, на которых работают и доценты, и профессора. Перечислить кафедры, на которых имеется должность с окладом условно 10000 р. и 15000р. Перечислить фамилии работающих на кафедре «Физика» в должности доцента.

*Сложный запрос на 2-х и больше отношениях*. Запрос требует обработки 2-х и более отношений. Может иметь логические операнды и вычислительные операции и требовать дополнительных операций над отношениями: проекция, выбор, объединение, соединение и т.д., что более подробно рассмотрено ниже.

Из анализа перечисленных типов запросов можно сделать следующий вывод. Для формализации запросов необходимо иметь:

— имена отношений (имена файлов);

— имена атрибутов (имена полей);

— значения атрибутов (значения полей);

— правила формирования условий (<=, =, ≠, >=);

— правила логических операций (AND, OR);

— правила реляционной алгебры для отношений.

Последнее правило является базовым для формирования сложных структур запросов, особенно для запросов на обработку нескольких отношений (файлов). Следует отметить, что все реляционные СУБД, как правило, в одном файле содержат одно отношение. SQL в одном файле содержит несколько отношений, точнее все, которые обрабатываются в запросе.

Запрос может быть сложным, композиционным, содержащим в себе многоуровневое вложение элементарных запросов с логическими и арифметическими операциями, в этом случае необходимо правило декомпозиции запроса и его формализации.

Рассмотрим способы реализации запросов в Access.

Access реализует метод формирования запроса по образцу. Запрос по образцу – это интерактивное средство для выбора данных из одной или нескольких таблиц. Формирование запроса осуществляется путем заполнения бланка запроса, который располагается в окне конструктора запросов. Такой метод формирования запроса прост для изучения и способствует эффективному использованию возможностей СУБД.

Для создания простейших запросов можно использовать мастер запросов.

***Создание запросов***

Неоспоримым преимуществом мастеров является возможность быстрого получения результатов при минимуме знаний о механизме его получения. Но, к сожалению, мастера запросов не позволяют указать условие выборки, критерий упорядочивания и более сложные, но и более интересные параметры выборки. Применение конструктора запросов в таких случаях позволяет создать сложные запросы в интерактивном режиме.

***Окно конструктора запросов***

Для вызова конструктора запросов перейдите в окне базы данных на вкладку «Запросы» и нажмите кнопку Создать. В окне диалога «Новый запрос» выберите опцию Конструктор и нажмите кнопку ОК. Access предложит вам выбрать таблицу или запрос, на основе которых будет осуществляться выборка. На этом этапе выбор таблицы или запроса не обязателен (вы можете добавить их и позже), но очевидно, что запрос должен выполняться, по крайней мере, на основе какой-либо таблицы или ранее созданного запроса. Выберите таблицу, нажмите кнопку Добавить и закройте окно диалога. На экране появится окно конструктора запросов , а в основном меню — команда Запрос.

Для формирования запроса в окне конструктора запросов необходимо выбрать таблицы, из которых осуществляется выборка, и поля результата запроса, указать критерии для выборки, группировки и упорядочивания данных.

Меню Запрос содержит команды добавления таблиц в окно конструктора запросов и удаления их из нее, команды выбора типа запросов (Выборка, Создание таблицы, Перекрестный, Обновление, Добавление, Удаление), команду запуска запроса и управления запросом (Переключатель режимов, Групповые операции, Добавление Имени таблицы, Свойства). В верхней части окна конструктора запросов находится схема данных запроса. Эта схема очень сильно напоминает схему данных базы данных. В отличие от нее, данная схема содержит список таблиц, включенных в запрос, и отображает связи между ними. В нижней части окна располагается бланк запроса. Каждая строка этого бланка выполняет определенную функцию:

- Поле. В этой строке помещаются те поля, которые вы используете для создания запроса, каждое в своей ячейке таблицы.

- Имя таблицы. Эта строка показывает вам, из какой таблицы (или запроса) выбрано данное поле.

- Сортировка. В этой строке вы указываете тип сортировки информации, возвращаемой в запросе, по возрастанию (от А до Я, от большего к меньшему, от более раннего к более позднему и т.д.) или по убыванию (от Я к А и т.д.).

- Вывод на экран. Если вы хотите, чтобы MS Access показывал информацию, найденную в поле, пометьте эту ячейку, чтобы установить флажок просмотра поля. Если же поле используется только для задания условия выбора данных, которые возвращает ваш запрос, оставьте его пустым.

- Условие отбора. В этой строке (и в строке, расположенной ниже ее) вы вводите ограничения поиска, задавая определенные условия, которые принято называть критерием поиска.

***Создание простого запроса***

Для создания простейших запросов достаточно данных одной таблицы. Рассмотрим создание такого запроса. Вся необходимая информация находится в выбранной вами таблице. Поэтому для создания запроса выполните следующие действия:

1) В окне базы данных перейдите на вкладку «Запросы» и нажмите кнопку Создать. 2) Откроется окно диалога «Добавление таблицы», в котором выберите нужную таблицу и нажмите кнопку Добавить. Закройте окно диалога. 3) На экране открывается окно конструктора запросов, схема данных которого содержит всего одну таблицу, а бланк запроса пуст.

***Добавление полей в бланк запроса***

Для выбора полей, которые должны присутствовать в результирующей таблице, вам необходимо отобразить их в бланке запроса.

В MS Access существует два варианта выбора полей результирующей таблицы. Вы можете воспользоваться наиболее приемлемым способом с вашей точки зрения:

1. Для добавления в таблицу отдельных полей вы можете выбрать поле таблицы на схеме данных и дважды нажать кнопку мыши. Выбранное поле будет вставлено в следующий доступный столбец в строке Поле бланка запроса. В строке Имя таблицы сразу же появится имя таблицы, а позиция Вывод на экран будет помечена.

2. В MS Access широко используется механизм перенести-и-оставить (drag-and-drop). Для использования этого механизма при выборе полей перейдите в таблицу в схеме данных, из которой вам надо выбрать поля. Выделите поля, которые вы собираетесь отобразить в запросе, нажмите кнопку мыши и, не отпуская ее, перенесите выбранные поля в бланк запроса.

Для выделения группы полей используется мышь совместно с клавишами Shift и Ctrl. Для выбора отдельно расположенных полей вместо клавиши Shift используйте клавишу Ctrl. В некоторых случаях необходимо выбрать все поля исходной таблицы. Для этого в MS Access существуют два способа, первый из которых состоит в выборе всех полей таблицы двойным нажатием мыши на строке заголовка и переносе выделенных полей в бланк запроса.

Вы можете использовать звездочку в списке полей таблицы для пометки всех полей в таблице. Для этого нажмите на звездочку в первой строке списка полей и, удерживая кнопку мыши в нажатом состоянии, перенесите ее в бланк запроса. Имя поля в бланке запроса будет содержать имя таблицы, за которым следует точка, а затем символ звездочки (например, клиенты.\*). Это означает, что были выбраны все поля исходной таблицы. В отличие от первого способа, в режиме конструктора вы не увидите каждое поле в отдельном столбце, но после запуска запроса все они будут выбраны.

У метода переноса звездочки есть одно большое преимущество. Если вы добавите в таблицу новое поле после того, как вы создали и запустили запрос, включающий «\*» ваш запрос автоматически включит в себя новое поле. При использовании первого способа запрос выбирает только те поля, которые были перенесены в бланк запроса.

***Удаление полей из бланка запроса***

Команда Правка | Очистить бланк удаляет все поля из бланка запроса.

Для удаления лишнего поля из запроса нажмите на область выбора столбца, а затем клавишу Delete. Поле исчезнет.

Прежде чем удалить поле, удостоверьтесь, что оно не используется в вашем запросе для определения условия выборки. В этом случае для запрета отображения поля в результирующей таблице снимите флажок Вывод на экран.

Изменение порядка полей Порядок полей в бланке запроса определяет порядок появления их в результирующей таблице. Для того чтобы изменить расположение поля в этом списке, выполните следующие действия:

1) Установите указатель мыши на область выбора столбца, который располагается прямо над названием поля. Когда указатель изменит вид на стрелку, нажмите кнопку, чтобы выделить столбец. 2) Нажмите и удерживайте кнопку мыши в этом положении. На конце указателя появится прямоугольник. 3) Перемещайте столбец в требуемом направлении. Толстая вертикальная линия покажет его текущее положение. 4) Отпустите кнопку, когда толстая вертикальная линия окажется в требуемом месте. Поле будет перемещено в новое место.

***Сортировка результатов выборки***

Вы можете сортировать результаты выборки по одному или нескольким полям так же, как вы это делали при сортировке таблицы.

Порядок сортировки записей результирующей таблицы определяется порядком следования полей в бланке запроса и критерием упорядочивания отдельных полей. Чтобы отсортировать данные в отдельном поле, перейдите на строку Сортировка требуемого поля и из раскрывающегося списка выберите значение По возрастанию или По убыванию. Если вы решили отказаться от сортировки по полю, выберите значение Отсутствует. Для сортировки по нескольким полям последовательно переходите на сортируемые поля и установите для них критерий сортировки. В отличие от сортировки в режиме таблицы, вы увидите результат только после выполнения запроса.

***Запуск запроса***

Мы закончили создание простого запроса, и наступило время запустить его на выполнение. Нажмите кнопку Запуск на панели инструментов или выполните команду Запрос| Запуск. Сохранение запроса.

Созданный запрос можно использовать в дальнейшем. Для этого вы Должны присвоить ему имя и сохранить его. Сохранение запроса осуществляется командой Файл| Сохранить как/Экспорт, которое откроет окно диалога ввода имени запроса. MS Access по умолчанию предложит имя запроса в поле ввода Новое имя, но лучше подобрать что-нибудь более значимое.

***Использование критерия выборки записей для ограничения***

***поиска***

Предположим, что вы захотели при формировании запроса ограничиться только вашими знакомыми из Ленинграда. Для этого вы должны задать критерий выборки записей, который определяется следующим образом:

1. В списке полей таблицы найдите нужное поле и дважды нажмите на нем мышью, чтобы добавить в бланк запроса.

2. В строке Условие отбора для только что созданного поля введите его значение и снимите флажок вывода на экран для поля , так как это поле используется только для задания условия выборки записей.

3. Теперь запустите запрос, и результаты появятся в режиме таблицы,

4. Вы можете сохранить этот запрос под новым именем и использовать его в дальнейшем. Изменение внешнего вида результирующей таблицы

Вы можете достаточно легко изменить внешний вид результирующей таблицы, используя те же средства, что и для обычных таблиц. Например, вы можете делать поля невидимыми, зафиксировать их, изменить шрифт, размеры столбцов и строк. Для выполнения этих функций перейдите в режим таблицы и выберите соответствующую команду из меню Формат.

***Использование в запросах вычислимых полей***

В результате выполнения запроса MS Access позволяет вам не только выбирать из таблицы содержащуюся в ней информацию, но также производить вычисления и отображать результат вычисления в результирующей таблице. Таким образом, вы можете получить данные, отсутствующие в исходной таблице.

При выполнении запроса вы можете вычислять значения по одному или нескольким полям исходной таблицы. Кроме того, вы можете использовать вычисляемые поля для объединения нескольких полей исходной таблицы в одно выходное поле. Например, вы можете в выходной таблице объединить фамилию и имя сотрудника.

Результаты вычислений, выводящиеся в поле, не запоминаются в исходной таблице. Вместо этого, вычисления выполняются каждый раз при запуске запроса, поэтому результаты всегда представляют текущее содержимое базы данных. Вы не можете редактировать результаты вычисления.

Для расчетов с использованием формул, определяемых пользователем, требуется создать новое вычисляемое поле прямо в бланке запроса. Вычисляемое поле создается с помощью выражения, которое вводится в пустую ячейку поля в бланке запроса или создается с помощью построителя выражений.

Выражение содержит формулы, которые связываются с помощью операторов. В качестве элементов формулы могут использоваться поля, константы и функции. Для изменения порядка вычислений и группировки данных в выражениях используются круглые скобки. Обратите внимание, что наименование поля в выражении записывается в виде [Имя Таблицы] ! [Поле]. Имена таблицы и поля заключены в квадратные скобки, а между ними находится восклицательный знак. На этом примере видно неоспоримое преимущество использования построителя: вам не обязательно знать все принятые в MS Access соглашения. По выбранному вами значению построитель сам преобразует его в требуемый формат.

После завершения формирования выражения нажмите кнопку ОК, и выражение будет перенесено в строку Поле бланка запроса. Так как каждое поле результирующей таблицы должно иметь имя, Access автоматически задаст имя вычисляемого поля (например. Выражение 1), которое отделяется от выражения двоеточием.

***Построение условий для выбора записей***

На практике в большинстве случаев требуется получить не все записи исходной таблицы, а лишь ту часть, которая удовлетворяет определенным условиям. Простейший критерий для выбора записей предполагает точное совпадение значений поля. Определение такого критерия было рассмотрено ранее, поэтому перейдем к построению более сложных условий выборки записей.

***Точное несовпадение значений одного из полей***

Возможно, потребуется найти в таблице записи, значения которых не удовлетворяют определенному условию. Для установки таких условий используется оператор Not, который печатается перед сравниваемым значением. Выполните следующие действия:

1. В строке Условие отбора поля вставьте перед сравниваемым значением оператор Not или <>.

2. Проверьте установку флажка вывода на экран для контролируемого поля. 3. Для выполнения запроса нажмите кнопку Запуск. На экране появится результирующая таблица, которая содержит записи вне значения контролируемого поля.

***Условие неточного совпадения***

Требование точного задания чисел или последовательности символов в качестве критерия поиска является жестким ограничением. Время от времени возникает неприятная ситуация, при которой вы не помните точного написания условия поиска. Кроме того, вы можете просто не знать, какой регистр (верхний, нижний или их сочетание) был использован при вводе каждой записи. При работе с большими таблицами такая ситуация превращается в серьезную проблему.

В подобных случаях вы можете осуществить выбор записей по условию неточного совпадения значений. Данное условие позволяет найти требуемые записи, зная лишь приблизительное написание значения. В MS Access для этих целей используется оператор Like, который осуществляет сравнение значения поля с заданным образцом. Образец задается сразу же за оператором и может содержать точное значение (например, Like "Иванов") или использовать символы шаблонов для поиска диапазона значений (например, Like "Ив\*"). Приведенная ниже таблица содержит символы шаблонов, которые используются с оператором Like, и количество цифр или символов, которым они соответствуют.

***Символы в образце Соответствие в выражении***

? Любой один знак

\* Нуль или более знаков

# Любая одна цифра [список\_зиаков] Любой один знак в списке\_знаков

[!список\_знаков] Любой один знак, не входящий в список\_знаков

Группа из одного или более символов, заключенная в квадратные скобки, используется для установления совпадения с одним символом выражения и может содержать любые символы, за исключением перечисленных ниже особых символов.

Особыми символами являются открывающая квадратная скобка ([), вопросительный знак (?), знак числа (#) и знак звездочки (\*). Для сравнения с ними необходимо заключить их в квадратные скобки. Закрывающую квадратную скобку нельзя применять в группе для установки соответствия с ней самой, но она может стоять вне группы как отдельный символ.

Помимо простого списка символов, заключенного в квадратные скобки, список\_знаков позволяет задать диапазон символов. Вы можете указать знак (-) для разделения верхней границы диапазона от нижней. Например, использование шаблона [Г-Л] в образце позволяет выбрать все записи, которые в указанной позиции поля содержат одну из заглавных букв в диапазоне от Г до Л. Вы можете использовать даже несколько диапазонов внутри одной пары квадратных скобок без специальных разделителей. Например, [г-лГ-Л] позволяет выбрать как заглавные, так и прописные буквы.

Восклицательный знак в начале списка\_знаков означает, что совпадение наступит, если в выражении будет найден любой символ, отсутствующий в списке знаков. Восклицательный знак, использованный вне квадратных скобок, соответствует самому себе.

Знак (-) для установления соответствия с самим собой можно использовать в начале (после восклицательного знака, если он есть) или в конце списка\_зваков.

Рассмотрим простой пример использования оператора Like. Предположим, что в таблице Клиенты требуется найти запись о представителе фирмы, название которой начинается на Ric. Однако вы не уверены в правильности написания фирмы. Попытаемся найти требуемую запись. Для задания условия выберите поле с названием и в строке Условие отбора поля напечатайте Like 'Ric\*' . Рассмотрим более сложный пример использования оператора Like. Допустим, вам надо выбрать всех клиентов, наименования которых начинаются на А или С. Для этого в строке Условие отбора поля Название введите следующее условие: Like ' [АС] \* '.

***Выбор записей по диапазону значений***

Часто необходимо выбрать из таблицы записи , данные которых попадают в диапазон значений. При этом границы диапазона заданы точно.

Для задания диапазона значений в окне конструктора запросов используются операторы > (больше), >= (не менее), < (меньше), <= (не более) и Between, которые вы можете использовать с текстовыми и цифровыми полями, а также полями дат. Для задания диапазона значений этого выполните следующие действия:

1) Откройте на экране окно конструктора запросов для нужной таблицы. 2) Выделите поля, которые хотите отобразить в запросе и перенесите их в бланк запроса. 3) Для задания условия отбора выберите поле и в строке Условие отбора поля напечатайте условие(например, >100000). 4) Нажмите кнопку Запуск, и на экране появится результирующая таблица, содержащая записи о значениях полей свыше 100 000.

В рассмотренном примере мы задавали только одну нижнюю границу диапазона. Очевидно, что можно использовать более сложные конструкции операторов

В заключение рассмотрим пример задания диапазона строковых данных. В этом случае кроме операторов сравнения вы можете использовать и оператор Like. Например, для получения списка клиентов, наименования которых начинаются с С по G, в строку Условие отбора поля Название введите условие отбора : Like '[C-G]\* '

Объединение критериев нескольких полей. В предыдущих примерах мы вводили условие запроса только для одного из полей таблицы. Однако довольно часто возникают ситуации, когда вам необходимо использовать более сложный критерий выборки, в котором задаются условия для нескольких полей таблицы или же несколько условий для одного поля. Если запись выбирается только в случае выполнения всех условий, то условие такого выбора называется логическим И, а запрос - И- запросом. Если же запись выбирается при выполнении хотя бы одного из всех условий, то условие такого поиска называется логическим ИЛИ, а запрос - ИЛИ- запросом.

Для задания И- выражения вы должны просто задать условие в строке Условие отбора для каждого из полей, образующих критерий. В качестве примера предположим, что из всех записей о заказах за определенный интервал времени требуется выбрать те, код доставки которых равен 1. Для решения этой задачи откройте запрос, в котором выбирались заказы в диапазоне дат, и выполните следующие действия:

1) Добавьте в бланк запроса поле Доставка. 2) Перейдите на строку Условие отбора поля и напечатайте 1 . 3) Выполните запрос, и вы увидите записи, содержащие сведения о заказах за указанный период дат, в которых использовалась доставка с кодом 1.

При задании ИЛИ- выражения каждое из условий выбора, образующих критерий должно располагаться на отдельной строке бланка запроса. Например, для выбора списка клиентов из США и Швеции нужно просто расположить первое условие в строке Условие отбора, а второе - в строке Или .

При формировании ИЛИ- выражения вы можете расположить условия выбора для различных полей в разных строках бланка запроса.

При вводе условия вы можете использовать операторы Ог и And, которые позволяют вам формировать в одной строке сложное условие выборки. Например, при поиске записей о клиентах из США и Швеции, вы можете поместить ИЛИ- выражение в одной строке. Результирующая таблица будет содержать те же записи, что и запрос. При вводе условия вы можете формировать любое допустимое в MS Access логическое условие, которое может содержать функции, операторы сравнения, Or, And, Not и скобки для изменения порядка выполнения выражения.

***Многотабличные запросы***

При выборе данных из таблиц наиболее часто используются многотабличные запросы, поскольку информация в реляционных базах данных содержится не в одной отдельной таблице, а в совокупности связанных таблиц. Связь между таблицами осуществляется на основании совпадающих полей. Для формирования многотабличного запроса нужно добавить в окно конструктора запросов все таблицы, участвующие в выборке, и определить условия их объединения. Для добавления таблицы выберите команду Запрос | Добавить таблицу или нажмите кнопку Добавить таблицу на панели инструментов. В открывшемся окне диалога «Добавление таблицы» выберите нужную таблицу. Образ таблицы появится в схеме данных запроса. Если в базе данных установлены отношения между таблицами, участвующими в запросе, то эта связь будет отображаться в виде линии, соединяющей таблицы. В этом случае вам не придется устанавливать связь между таблицами в конструкторе запросов. Если же между таблицами не существует связи, то вы можете установить требуемую связь, используя механизм перенести -и- оставить. Для этого выберите поле в одной из таблиц, нажмите кнопку мыши и перенесите выбранное поле на связываемое поле в другой таблице.

В отличие от определения постоянных отношений между таблицами, при задании условия объединения таблиц вы можете использовать любые поля таблиц.

***Объединение двух таблиц***

Вначале рассмотрим простое объединение двух таблиц на следующем примере. Отпускаемый клиентам товар может доставляться разными средствами. Для определения типа доставки товаров необходимо использовать информацию из двух связанных таблиц Т1 и ТО. Пусть при создании базы данных между этими таблицами уже определены постоянные отношения. Для решения этой задачи выполните следующие действия:

1) Добавьте таблицу Т1 в новое окно конструктора запросов. 2) Нажмите кнопку Добавить таблицу на панели инструментов и добавьте в запрос еще одну таблицу — ТО. Поскольку между этими таблицами установлено постоянное отношение, то после добавления таблицы в окне конструктора запросов автоматически отобразится связь между ними. 3) Перенесите в бланк запроса из таблицы ТО код товара, а из таблицы Т1 — код заказа и тип доставки . 4) Выполните запрос, и на экране появится результирующая таблица, содержащая требуемую информацию .

***Условие отбора в многотабличных запросах***

Теперь обратимся к более сложному запросу, в котором из всех записей в таблицах необходимо выбрать только те записи, которые отвечают определенному условию отбора.

Для задания условия отбора перейдите на строку Условие отбора поля, по которому производится выборка, снимите флажок Вывод на экран для этого поля, так как не имеет смысла выводить столбец, каждая строка которого будет содержать одно и то же значение. Перейдите на Условие отбора поля другой записи, а затем снимите для нее флажок Вывод на экран. Далее добавьте в бланк запроса дополнительные необходимые поля и выполните запрос. Вы увидите на экране результирующую таблицу, содержащую информацию обо всех интересующих вас значениях. Итоговые запросы

При необходимости можно объединить в запросе любое количество требуемых таблиц. Однако довольно часто имеющихся в таблицах данных оказывается недостаточно. Например, вам нужно найти сумму, максимальную величину в поле или количество записей, содержащих определенную величину. В этом случае вам необходимо выполнить итоговые вычисления. Для этих целей используются математические возможности MS Access, который отлично приспособлен для выполнения таких вычислений.

Для определения суммы значений полей или нахождения среднего следует создать итоговый запрос. Для его создания, как и для обычного запроса, откройте новое окно конструктора запросов. Далее выберите используемые в запросе таблицы, а затем перенесите в бланк запроса нужные поля.

Запросы, выполняющие вычисления в группах записей, называются итоговыми запросами. В итоговом запросе выполняется не только Суммирование, но и другие виды вычислений. Например, вы можете найти среднее, минимальное и максимальное значения поля.

Для создания итогового запроса выберите Вид| Групповые операции или нажмите кнопку Групповые операции на панели инструментов. В бланке запроса появится новая строка с наименованием Групповая операция. В этой строке вы должны указать тип выполняемого вычисления (например, суммирование – Sum).

Таблица 3 содержит перечень всех допустимых видов итоговых операций, которые можно выбрать из раскрывающегося списка в строке Групповая операция. Две из них — StDev и Var -являются известными операциями статистки. Операция Count требует некоторых дополнительных разъяснений. Она действует следующим образом: просматриваются все записи для указанного поля, и вычисляется количество записей, которые содержат в этом поле какое-либо значение. Другими словами, данная операция пропускает пустые значения.

Таблица 2 -Типы операций, доступные в строке Групповая операция

бланка запроса



Для удаления строки Групповая операция достаточно нажать еще раз на кнопку Групповая операция.

***Группировка полей запроса***

Группировка позволяет получить вычисляемую информацию о подгруппах таблицы. Например, сгруппировав данные в таблице Заказано по полю КодТовара, можно получить сведения об итоговом количестве заказов каждого вида товара.

Как обычно, создание запроса начните с добавления таблицы в окно конструктора запросов. Далее выполните следующие действия:

1) После размещения полей в бланке запроса выберите команду Вид | Групповые операции для добавления в бланк запроса строки Групповая операция. 2) В строке Групповая операция группируемого столбца установите значение Группировка, а в другом столбце можно подсчитать количество записей, установив значение Count.

При формировании групповых итоговых запросов довольно часто данные упорядочиваются по итоговым полям, Например, в запросе для поля, в котором определяется количество записей, установлен признак сортировки по убыванию. Это означает, что в результирующей таблице первыми будут расположены наиболее часто встречающиеся записи.

Поле, используемое для группировки, не обязательно должно находиться в гой же таблице, что и итоговое поле. Оно может находиться в другой таблице.

***Группировка по нескольким полям***

Вы можете группировать данные не только по одному полю, но и по нескольким полям одновременно, а также создавать группы внутри групп. Для этого вы можете сгруппировать записи сначала по одному полю, затем — по другому. Порядок полей группировки, размещенных в бланке запроса, определяет порядок группировки выбираемых данных.

***Включение в запрос выражений***

MS Access предоставляет вам возможность выполнять итоговые операции над вычисляемыми полями выборки. Например, таблица Заказы содержит данные о клиентах, а таблица Заказано — о количестве товаров, проданных в каждой партии и цене товара. На основании этой информации вы можете вычислить итоговую стоимость заказа каждым покупателем по каждому товару.

Чтобы включить в итоговый запрос выражение, добавьте в бланк запроса вычисляемое поле и укажите тип итоговых имений, осуществляемых над этим полем (т.е. выражение для вычисления).

Выражения для вычисления могут содержать формулы, связанные арифметическими операторами. В качестве элементов формулы могут использоваться поля, константы и функции. Для изменения порядка вычислений в выражениях используются круглые скобки.

***Изменение наименований итоговых полей***

По умолчанию поля результирующей таблицы имеют те же наименования, что и поля исходной таблицы, а итоговым полям присваиваются наименования в соответствии с принятым в MS Access соглашением. Начальная часть имени обычно содержит имя итоговой операции, за которым следует имя поля, над которым выполняется итоговая операция. Во многих случаях предпочтительнее давать таким полям более осмысленные наименования. MS Access позволяет вам по своему усмотрению изменить наименования полей результирующей таблицы.

Для изменения наименования поля в строке Поле перед именем поля или выражения напечатайте новое имя поля и отделите его двоеточием.

Изменение наименований полей в бланке запроса действует только на поля в результирующей таблице. Имена полей в исходной таблице остаются без изменения.

***Использование более сложных условий отбора записей***

***в итоговых запросах***

В итоговых запросах существует два типа критериев отбора записей. Первый тип аналогичен критерию, используемому в обычных запросах. Он исключает записи, не удовлетворяющие заданному критерию, перед выполнением итоговых вычислений. Второй тип является специфичным для итоговых запросов. Данный критерий применяется к результату итоговых вычислений. Критерии вводятся в строке Условие отбора.

ADO.NET не накладывает ограничений на язык или синтаксис, но наиболее распространенным форматом языка запросов к источнику данных является одна из разновидностей языка структурированных запросов SQL. В Microsoft SQL Server поддерживается язык T-SQL, а в Oracle – язык PL/SQL, но оба этих языка являются текстовыми, а их синтаксис и назначение очень похожи, хотя и не совпадают.

SQL позволяет выполнить запросы к базе данных, указав критерии отбора с помощью команды SELECT, либо обрабатывать данные командами INSERT, UPDATE, DELETE.

Команда, посылаемая базе данных через ADO.NET, может иметь одну из следующих форм:

Простая команда выборки:

*SELECT имяполя1, имяполя2,…, имяполя N FROM имя\_таблицы WHERE имяполя = ‘ значение’;*

Параметризованная команда для гибкого многократного использования

*SELECT имяполя1, имяполя2,…, имяполя N FROM имя\_таблицы WHERE имяполя = @Параметр;*

Выполнение хранимой процедуры

EXEC имя\_процедуры (@Параметр);

В ADO.Net поддерживаются приведенные выше команды, которые можно выполнить с помощью класса команды.

Создать объект команды можно одним из трех перегруженных вариантов конструктора:

1 способ:

SqlCommand testCommand =new SqlCommand ( );

testCommand. Connection = con;

testCommand. CommandText = “ текст команды”;

2 способ:

SqlCommand testCommand =new SqlCommand (“ <<текст команды>>”, con);

3 способ:

SqlCommand testCommand = con. CreateCommand( );

testCommand. CommandText = “ текст команды”;

*Если используется хранимая процедура, то в свойстве CommandType объекта команды нужно указать CommandType. StoredProcedure.*

Для выполнения команды запроса необходимо использовать следующие методы:

|  |  |
| --- | --- |
| ExecuteReader () | Возвращает объект DataReader для доступа к данным в режиме однонаправленного чтения. |
| ExecuteNonQuery() | Направляет текст команды в хранилище данных. |
| ExecuteScalar () | Предназначена для запросов, возвращающих единственное значение. |

Пример1:

con=new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\Institut.mdb");

con.Open();

// Испольуем метод CreateCommand для создания команды

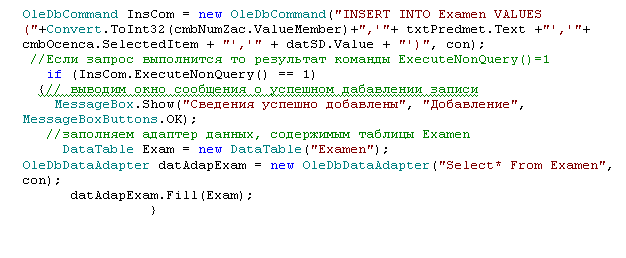
cmd = con.CreateCommand();

cmd.CommandText = "Select NumZac,Familia,Name,Gruppa From Students";

//выполняем команду SQL

rdr = cmd.ExecuteReader();

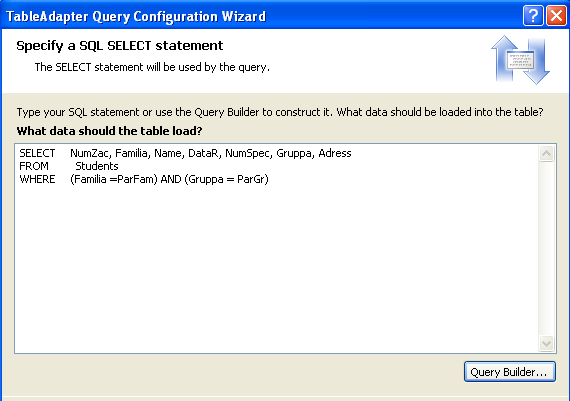
Пример 2 Реализация запроса для добавления данных в таблицу Экзамены и обновление содержимого адаптера данных:



Другой способ создания запросов в Visul Studio - использовать построитель запросов QueryBilder.

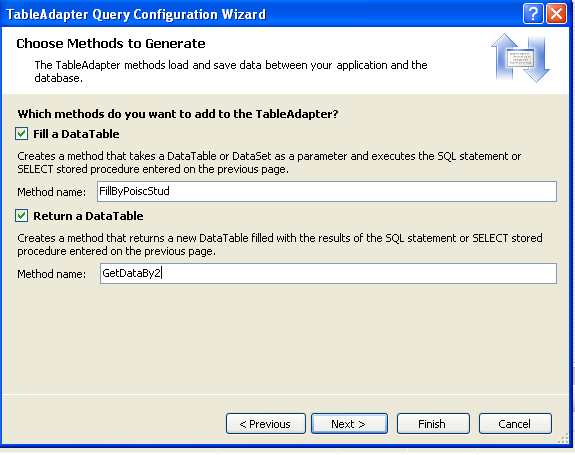
Например, создадим запрос о выборе данных о студенте по вводимой фамилии и группе. Для этого:

* Перейти на схему – файл **\*.xsd** на панели ***Solution Explorer***.
* Щелкните правой клавишей мыши, выберите ***View Designer***
* В окне на таблице Students щелкните правой клавишей и выберите ***Add->Query***.
* Тип запроса SELECT, для изменения текста запроса, предложенного по умолчанию, нажмите на кнопку ***QueryBilder***.
* Выберите Add Table из списка таблиц выберите Students.
* Отметьте галочками поля для вывода в запрос. В колонке Filter в строке Familia введите = ParFam.
* В колонке Filter в строке Gruppa введите = ParGr

**

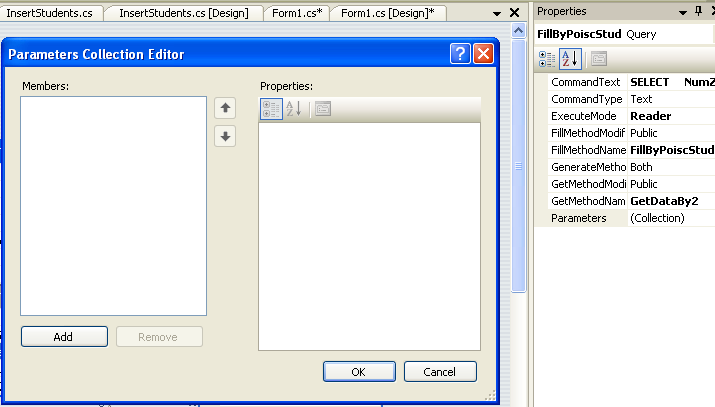
*Рисунок 12 – Окно мастера для создания запросов*

Нажмите ОК .



*Рисунок 13 – Окно ввода названия метода запроса.*

В окне введите название метода FillByPoiscStud. Затем Next.



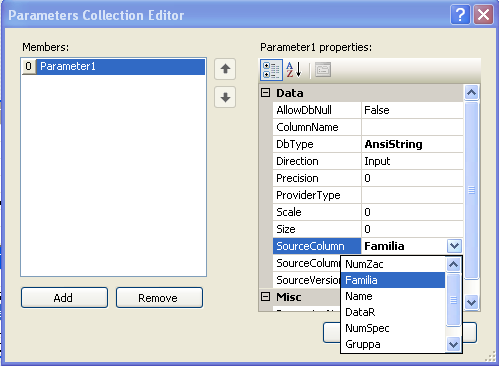
*Рисунок 14 – Диалоговое окно для определения параметров, используемых в запросе.*

Закройте мастер запросов.

Затем вновь щелкните правой клавишей по вновь созданному запросу в окне свойств выберите строку **Parameters**, щелкните по **…**

В появившемся окне нажмите **Add**.

В окне свойств выберите Source Column из списка выберите поле **Familia**.



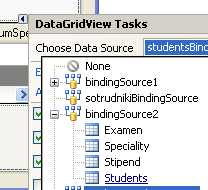
*Рисунок 15 –Диалоговое окно связывания поля с параметром запроса.*

Затем в строке Parameter Name введите **ParFam**

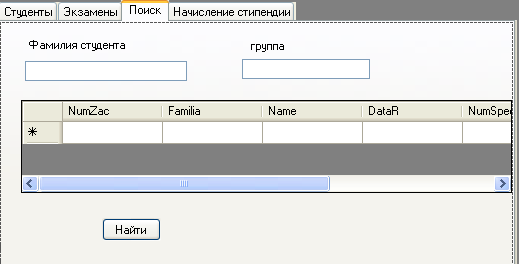
Аналогично добавьте еще один параметр в строке Source Column выберите Gruppa, в строке Parameter name введите **ParGr**.

Добавьте вкладку на форму с названием Поиск с именем **tabPoisc**. На ней расположите метку с названием **Фамилия студента** и метку **группа**, а также два объекта ***TextBox***  со свойством Name = **txtPoiscSt** и Name = **txtPoiscGr**

На вкладке расположите кнопку с названием **Найти** и для вывода данных перетащите объект ***DataGridView***. Укажите в качестве источника данных bindingSource 2 -> Students.

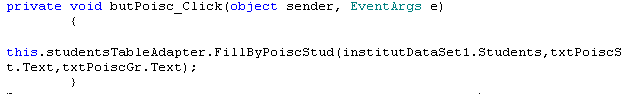


*Рисунок 16 – Контекстное меню добавления источника данных для объекта DataGridView.*

**

*Рисунок 17- Пример содержимого на вкладке Поиск формы.*

Дважды щелкните по кнопке для перехода в программный код и введите следующий текст



Запустите запрос на выполнение введите в текстовое поле значение фамилии и во второе поле группу. Нажмите на кнопку Найти.

**2.8 Реализация отчетов**

Под отчетом понимается форматированное представление данных, выводимое на экран, принтер или в файл.

При создании отчета в MS Access можно воспользоваться средствами автоотчета, ускоряющими процесс создания отчета, или разработать для отчета специальный формат с помощью конструктора отчетов. Конструктор отчетов позволяет создавать отчеты, как в табличном виде, так и в свободной форме.

Табличный отчет представляет собой напечатанную таблицу, в которой данные упорядочены по столбцам и строкам. Каждый из столбцов отчета содержит поле исходной таблицы или вычисляемое поле, а строка представляет собой запись.

Табличный отчет позволяет напечатать данные из таблиц в наиболее простом и естественном виде. Однако табулированное представление данных в отчете имеет свои недостатки. На практике в ряде случаев (почтовые этикетки, чеки, письма и т.д.) поля исходной таблицы должны располагаться в специально отведенных для них местах отчета. Очевидно, что табличный отчет не пригоден для этих целей.

Отчеты в свободной форме позволяют устранить ограничения, свойственные табличным отчетам. С помощью конструктора отчетов можно разработать специальный формат отчета, где поля исходной таблицы будут расположены в требуемых местах отчета.

В данной работе мы освоим основные средства, используемые при создании табличных отчетов и отчетов в свободной форме, а также, как напечатать отчет в соответствии с разработанным форматом и как улучшить формат отчета, не выводя его на печать.

Средства MS Access по разработке отчетов предназначены для создания макета отчета, по которому может быть осуществлен вывод данных из таблиц в виде выходного печатного документа. Эти средства позволяют конструировать отчет сложной структуры, обеспечивающий вывод взаимосвязанных данных из многих таблиц. При этом могут быть выполнены самые высокие требования к оформлению документа.

В процессе конструирования формируется состав и содержание разделов отчета, а также размещение в нем значений, выводимых из полей таблиц базы данных. Оформляются заголовки, подписи реквизитов отчета, размещаются вычислимые реквизиты. Средства конструирования отчета позволяют группировать данные по нескольким уровням. Для каждого уровня могут производиться вычисления итогов, определяться заголовки и примечания. При формировании отчета могут производиться разнообразные вычисления.

При необходимости вывода в отчет данных из многих таблиц можно использовать многотабличный запрос.

***Рекомендации, предшествующие созданию отчета***

Прежде чем приступить к созданию отчета, вы должны дать ответ на следующие вопросы:

- С какой целью создается настоящий отчет и чем он будет вам полезен?

- Какая информация и из каких таблиц должна быть представлена в отчете?

- Отчет какого вида вы предполагаете создать (табличный, в свободной форме или почтовые наклейки) ?

- Предполагается ли группировка данных ? Четкие ответы на поставленные вопросы облегчат работу при создании отчета.

Рекомендуется, по крайней мере, при создании первых отчетов, прежде чем приступить к работе с конструктором отчетов, прорисовать формат отчета на бумаге. Это позволит избежать ошибок и сократить время на разработку отчета. По мере приобретения навыков работы с конструктором станет возможным обходиться без эскиза формата на бумаге, улучшая отчет прямо на экране.

***Окно конструктора отчетов***

Основными объектами отчета являются пояснительный текст и поля отчета. Для улучшения восприятия информации в отчет могут добавляться OLE объекты, рисунки, линии и прямоугольники. Поля отчета могут непосредственно соответствовать полям исходной таблицы или являться результатом вычислений над ними.

При создании и модификации отчетов конструктор отчетов позволяет удалять, добавлять, перемещать области вместе с расположенными в них объектами. Можно установить цвет и управлять параметрами отображения любых элементов и областей отчета.

Для открытия окна конструктора отчетов для модификации созданного ранее отчета выполните следующие действия: 1. В окне базы данных перейдите на вкладку «Отчет». 2. Установите указатель мыши на модифицируемый отчет. 3. Нажмите кнопку Конструктор,

Для открытия окна конструктора отчетов при создании нового отчета необходимо выполнить следующее: 1. В окне базы данных перейдите на вкладку «Отчет». 2. Нажмите кнопку Создать. 3. В открывшемся окне диалога «Новый отчет» из списка способов создания отчетов выберите Конструктор и нажмите кнопку ОК.

На экране откроется окно конструктора отчетов. Создание отчета аналогично созданию формы. Имея опыт создания форм, вы уже понимаете, что создание отчета в конструкторе отчетов является достаточно трудоемкой работой.

***Области отчетов***

Окно конструктора отчетов аналогично окну конструктора форм и также содержит объекты (надписи, поля ввода) и сетку. Также как и в формах, в отчетах существуют области - заголовок отчета, верхний и нижний колонтитулы, область данных и область итогов:

- Заголовок отчета размещается в начале отчета

- Верхний колонтитул отображается в начале каждой страницы и под заголовком отчета

- Область заголовка группы отображается перед первой записью каждой группы

- Область данных отображается для каждой записи, входящей в отчет.

- Область примечания группы отображается после области данных последней записи каждой группы - Нижний колонтитул отображается в нижней части каждой страницы

- Область примечаний отображается в конце отчета

При формировании отчета для печати или предварительного просмотра MS Access начинает печать с заголовка отчета и верхнего колонтитула. Затем до конца страницы печатается область данных, после которой MS Access помещает нижний колонтитул, разделитель страниц и верхний колонтитул следующей страницы. Каждая страница содержит верхний колонтитул, столько областей данных, сколько помещается на странице, и нижний колонтитул На последнюю страницу отчета, перед нижним колонтитулом, помещается область примечаний отчета

Если в отчете имеются области заголовка и примечания группы, порядок печати практически не меняется Разница заключается в том, что перед первой записью каждой группы помещается область заголовка группы и после последней записи каждой группы помещается область примечания группы.

Объекты, расположенные в верхнем колонтитуле, предназначены для печати информации или рисунков в верхней части каждой страницы отчета, а объекты расположенные в нижнем колонтитуле — в нижней части. И соответственно, объекты заголовка отчета печатаются на первой странице отчета, а объекты области итогов — на последней. Если в отчете есть группы, то для каждой группы можно создать области верхнего и нижнего колонтитула.

***Добавление заголовка и итогов***

Как правило, все отчеты содержат заголовок отчета, а в большинстве случаев и итоговую часть отчета. Для их формирования добавьте в отчет области «Заголовок отчета» и «Примечание отчета», используя команду Вид | Заголовок отчета/Примечание отчета. В отчете появятся указанные вами области.

Размещение даты печати отчета Обычно в заголовке отчета или нижнем колонтитуле размещают дату печати отчета. Для создания данного элемента в форме необходимо выполнить следуют действия:

1.Выбрать инструмент Поле на панели элементов.

2. Установить указатель мыши на место в области верхнего колонтитула, в котором вы предполагаете разместить поле даты. В отчете появится связанный объект, состоящий из поля ввода и надписи к нему.

3. Выделите надпись связанного поля и удалите ее нажав клавишу Delete.

4. Выделите поле ввода и откройте для него окно свойств. 5. Перейдите на вкладку «Данные» и нажмите кнопку вызова построителя свойства Данные. На экране откроется окно диалога «Построитель выражения».

6. Откройте раздел «Встроенные функции» папки Функции и, используя функцию Now из списка функций, задайте выражение для значения поля = Now ( ).

7. Нажмите кнопку раскрытая списка свойства Формат вкладки «Макет» и выберите устраивающий вас формат отображения даты в отчете.

Теперь при печати отчета вы будете знать дату печати отчета.

***Размещение номера страницы отчета***

Для размещения номера страницы отчета, как и даты, существует специальная команда.

1. Находясь в окне конструктора отчетов, выполните команду Вставка | Номер страницы.

2. В открывшемся окне диалога «Номера страниц» задайте формат и расположение создаваемого объекта, Список Выравнивание определяет расположение объекта на странице.

3. Установив все нужные значения, нажмите кнопку ОК.

***Группировка данных***

Создание индексов таблицы или сортировка позволяют распечатывать записи в требуемом порядке. Однако этих средств далеко не всегда оказывается достаточно. На практике часто возникает необходимость объединять записи в группы. Для этой цели используется команда Сортировка и группировка меню Вид или контекстного меню, которые позволяют создавать до 10 уровней вложенности групп и выполнять над ними следующие операции: - Напечатать текст, идентифицирующий конкретные группы.

Напечатать каждую группу с новой страницы.

Для добавления группы в отчет необходимо в окне конструктора отчета выполнить следующие действия:

1. Выполните команду Вид | Сортировка и группировка. На экране откроется окно диалога «Сортировка и группировка», которое содержит весь список ранее созданных групп. Можно редактировать или удалять имеющиеся группы, а также добавлять новые.

2. Для ввода поля или выражения группировки установите курсор на первую свободную строку после введенных значений и, используя кнопку раскрытия списка, открывающую список полей таблицы, задайте выражение сортировки.

3. Перейдите в нижнюю часть окна диалога «Сортировка и группировка». В рамке «Свойства группы» находятся свойства, относящиеся к создаваемой группе:

* Заголовок группы. Помещает в отчете в области заголовка группы заголовок при группировке по выбранному полю или выражению.
* Примечание группы. Помещает в отчете в области примечания группы примечание при группировке по выбранному полю или выражению. Группировка. Определяет способ группировки данных на конкретном уровне группировки по полю или выражению в зависимости от типа данных. При группировке по выражению доступны все значения для всех типов полей. По умолчанию, для всех типов данных задается группировка По полному значению Интервал. Указывает величину интервалов, на которые должны быть разбиты значения в группируемом поле. Допустимые интервалы группировки определяются типом данных и значением свойства Группировка поля или выражения, по которому проводится группировка. Например, значение свойства Интервал может определять группировку записей по первому символу в названии товара.
* Не разрывать. Указывает, следует ли печатать разные элементы данной группы, в том числе заголовок, область данных и область примечаний, на одной странице отчета. Например, можно указать, что заголовок группы всегда должен печататься на одной странице с первой областью данных. Свойство Не разрывать может принимать следующие значения:

- Нет. (Значение по умолчанию. Группа печатается без обязательного размещения разных областей на одной странице).

- Полную группу (Вся группа печатается на одной странице).

- Первую область ( Заголовок группы печатается на текущей странице только в данных том случае, если вместе с ним помещается первая запись область данных).

4. Скорректируйте свойство Заголовок группы. Для этого нажмите на кнопку раскрытия списка и выберите Да.

5. Закройте окно диалога «Сортировка и группировка».

6. В конструкторе отчетов перенесите поле и надпись к нему в созданную область «Заголовок группы». Теперь при просмотре отчета вы увидите в начале каждой группы наименования объектов, по которым вы осуществили группировку.

7. Для расположения надписей, относящихся к объекту группировки, при печати на одной странице, откройте окно диалога «Сортировка и группировка» и для свойства Не разрывать данной группы установите значение Полную группу.

***Удаление сортировки или группы***

Для удаления сортировки или группы, находясь в окне конструктора отчета, выполните команду Вид| Сортировка и группировка. В окне диалога «Сортировка и группировка» выделите сгруппированные или отсортированные поля нажатием на область выбора поля. Затем просто наймите клавишу Delete. Выберите Да для подтверждения вашего выбора.

***Отображение промежуточной суммы в конце каждой***

***страницы отчета***

МS Ассеss не позволяет использовать в нижнем колонтитуле функцию Sum. Рассмотрим отчет, в нижнем колонтитуле которого требуется разместить сумму, на которую, например, выполнены заказы. Для этого выполните следующие действия:

1. В любой другой области создайте элемент управления, подсчитывающий значение этой суммы,

2. Установите для его свойства Вывод на экран значение Нет. Кроме этого, в зависимости от вашего желания, выберите для его свойства Сумма с накоплением значение Для группы или Для всего.

3. Создайте в нижнем колонтитуле не связанное поле. В свойство Данные этого поля введите имя элемента управления, который подсчитывает сумму и содержит выражение = [Сумма]. Он будет отображать промежуточную сумму, вычисляемую элементом управления Сумма.

***Нумерация записей в области данных отчета***

На практике может возникнуть необходимость пронумеровать записи в группе или по всему отчету. Для того чтобы сделать это, выполните следующее:

1) Добавьте в область данных отчета несвязанное поле.

2) Откройте окно свойств этого элемента управления.

3) В поле ввода свойства Данные введите = l.

4) Для свойства Сума с накоплением установите значение Для всего или Для группы в зависимости от того, что вы собираетесь нумеровать. Расположение информации в области данных в несколько колонок

Для экономии бумаги отчет с данными, расположенными в столбец, можно преобразовать в отчет с заголовком и с областью данных, выводимой в два столбца. Для создания подобного отчета необходимо использовать опции макета страницы:

1) Создайте отчет, у которого информация располагается в один столбец.

2) Для расположения данных в два столбца выполните команду Файл | Макет страницы. В открывшемся окне диалога «Макет страницы» нажмите ярлык «Макет».

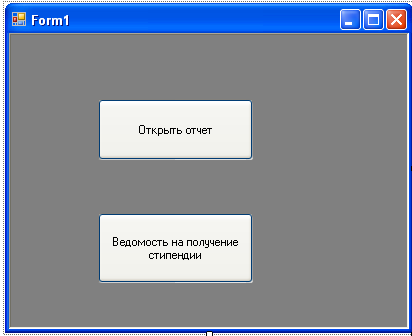
3) В поле ввода Число элементов по горизонтали укажите количество столбцов, равное 2.

4) В рамке Размер элементов уменьшите значение параметра Ширина.

5) Выберите альбомную или книжную ориентацию элементов, установив соответствующую опцию.

6) Нажмите ОК.

Приведем пример формы, на которой будут расположены кнопки для открытия отчета:

**

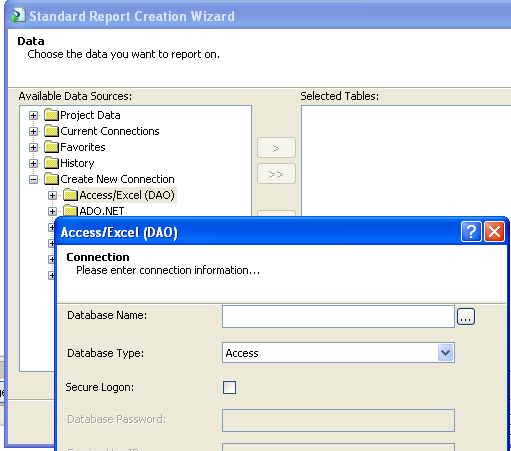
*Рисунок 18- Пример формы с кнопками для открытия отчетов.*

Для этого перетащите элемент **Button** с панели **Toolbox**. Щелкните правой кнопкой по данному объекту, и измените свойство **Text**, введите выводимый на кнопке текст *«Открыть отчет»*, аналогично перетащите вторую кнопку и в свойстве Text – введите *«Ведомость на получение стипендии»*

Для того, чтобы создать отчет средствами Visual Studio необходимо:

1. Выбрать *Project -> Add Windows Form* из списка файлов выбрать *CristalReport* .
2. В строке Name задать имя отчета, например,**Otchet1.rpt**
3. Нажать кнопку **Add**.
4. Выбрать **As a Blank Report** и нажать ОК.
5. В правом появившемся окне **Field Explorer** щелкнуть правой клавишей по пункту **Database Fields** выбрать Database Expert.

Выбрать пункт **Access/Exel (DAO**). Затем в строке Database Name, нажать на кнопку и выбрать базу данных, которая будет являться источником данных.

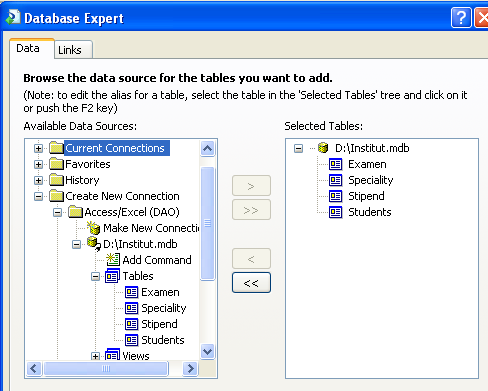


*Рисунок 19 –Диалоговое окно подключения источника данных к отчету.*

Нажмите на кнопку Готово.

В следующем окне раскройте содержимое пункта Tables.

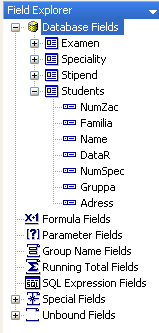
Выберите все таблицы из базы данных, нажимая на >.

**

*Рисунок 20 –Окно выбора полей таблиц, которые будет содержать отчет.*

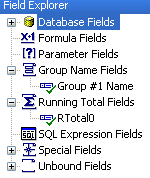
Нажмите ОК.

Затем в окне **Field Explorer** раскрыть список **Database Fields**

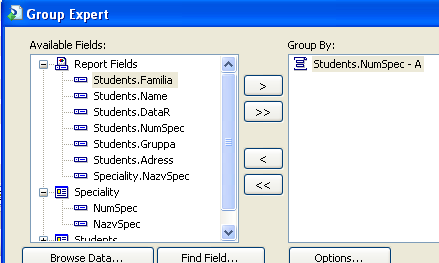
**

*Рисунок 21- Окно Field Explorer.*

Перетаскивая поля из данного окна расположите, чтобы получилась структура следующего отчета (см. рис.24).

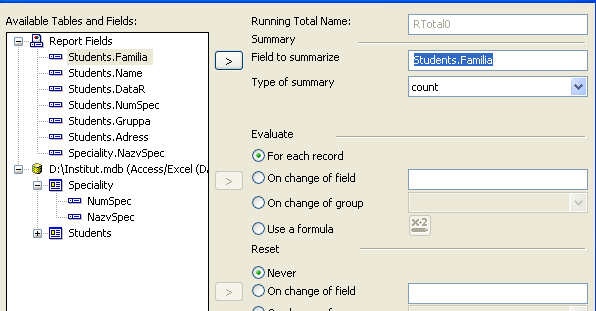
Для включения в отчет группировки. Выбрать:**Group Name Fields,** щелкнуть правой клавишей выбрать - **Insert Group.**

В поле из списка выбрать поле NumSpec из таблицы Students

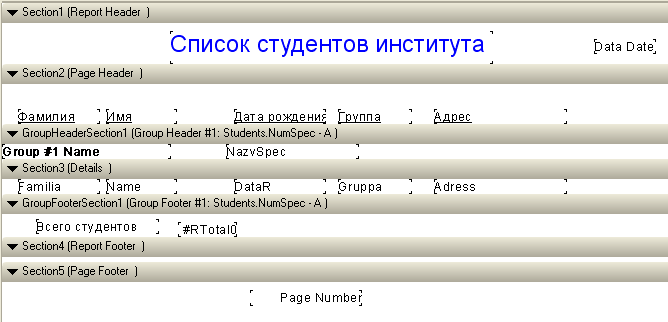
**

*Рисунок 22 –Диалоговое окно добавления группировки в отчет.*

Для подсчета итогового значения количества, добавить текстовое поле, в котором ввести «**Всего студентов**», затем щелкнуть правой клавишей по строке **Running Total Fields**, выбрать – **New**. Выбрать поле **Familia**, из списка выбрать функцию **Count.**

**

*Рисунок 23 –Диалоговое окно добавления агрегатных функций.*

**

*Рисунок 24 – Окно отчета в режиме дизайнера.*

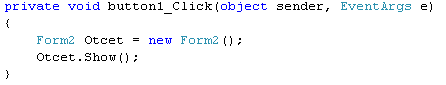
Добавим в проект вторую форму для отображения содержимого отчета в приложении.

* Выбрать **Project -> Add Windows Form -> Windows Form -> Add**.
* Расположите на форме элемент для просмотра отчетов **CrystalReportViewer.** Для этого перетащите с панели Toolbox элемент **CrystalReportViewer.**
* В появившемся окне нажмите на маленький треугольник в правом верхнем углу, в раскрывшемся меню выберите **Choose a Crystal Report** .

В появившемся окне из поля со списком выберите имя созданного отчета.

Перейдите на первую форму.

* Измените свойства первой формы **IsMDIContainer** на значение **true**.
* Щелкните дважды по кнопке с надписью Открыть отчет и введите следующий программный код:



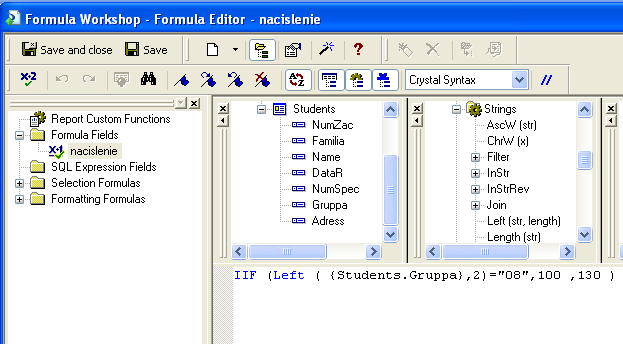
Пример создания отчета – ведомости на получение стипендии. Если студент первого курса – сумма стипендии 100 руб, остальным 130 руб.

* Добавить в проект файл Crystal Reports.
* В окне выбрать **As a Blank Report**.
* В конструкторе отчета перейти к **DataBase Field - > Database Expert**
* Добавить таблицы **Students, Stipend**
* Добавьте группировку в отчет по полю Gruppa

Group Name Fields -> Insert Group -> из поля со списком выбрать Gruppa -> OK.

* В заголовок отчета Report Header, добавьте объект Text Object. Введите «Ведомость на получение стипендии».
* В заголовок группы добавьте текст «Группа».
* В область Details перетащите поля Фамилия и Имя.
* Создать формулу. Formula Fields -> New -> Введите имя nacislenie ->Use Editor

Введите следующую формулу:

**

*Рисунок 25 –Диалоговое окно создателя формул.*

Функция Left( ) вырезает слева из указанного строкового источника, указанное во втором параметре количество символов, в нашем случае источник – значение поля Gruppa, количество символов – 2.

Закрыть окно построителя формул и сохранить изменения.

* Перетащить формулу nacislenie в область отчета Details.
* В Group Footer в текстовом объекте введите «Всего выдача на группу в сумме»
* Создать агрегатную функцию для подсчета суммы выдачи для каждой группы.

Runnind Total Fields -> New

**

*Рисунок 26 –Диалоговое окно добавления группировки и агрегатных функций в отчет.*

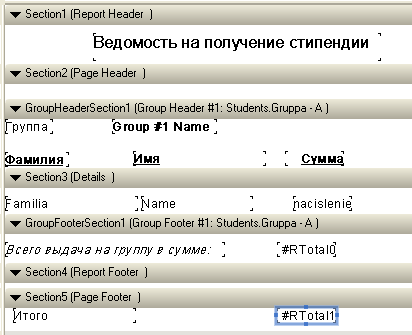
Перетащить RTotal0 в Group Footer.

Создайте еще один объект Running Total для суммирования по всем группам

**Примечание: Выберите среди пунктов Reset пункт Never.**

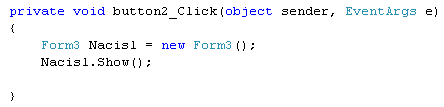
* Расположите его в область отчета Page Footer, добавив текстовый объект с надписью «Итого».

В результате у вас должна получиться следующая структура отчета:

**

*Рисунок 27 – Окно отчета в режиме дизайнера.*

* Создайте форму, на которой расположите объект **CrystalReportViewer - Choose a Crystal Report – выберите имя второго отчета.**
* Перейдите на первую форму и для второй кнопки с надписью «Ведомость на получение стипендии» введите следующий программный код:



**2.9\*. Соединение базы данных Access c Delphi**

Чтобы организовать работу программы с БД, потребуется следующие компоненты:

- **ADOConnection** – используется для подключения к БД (закладка палитры ADO, в некоторых версиях dbGO);

- **ADOTable** – связывается с конкретной таблицей БД (закладка ADO);

- **DataSource** – компонент, используется как связка данных из таблиц, с отображающими и управляющими компонентами Delphi (закладка Data Access);

- **DBGrid** – таблица, позволяющая вывести содержимое таблицы БД на пользовательскую форму (закладка Data Controls);

- **DBNavigator** – кнопочная панель, способная управлять данными в привязанной к ней таблице (закладка Data Controls).

После того, как все компоненты установлены на форму, можно приступить к их настройке. Для решения нашей задачи никакого программного кода не потребуется. Только настройка свойств компонентов:

## Свойство ConnectionString компонента ADOConnection

Свойство ConnectionString удобно настраивается в специальных окнах. Открываем редактор свойства ConnectionString в инспекторе объектов кнопкой «…». В появившемся окне оставляем все настройки как они были по умолчанию и нажимаем кнопку «Build…».

Получаем еще одно окно с несколькими закладками.

В первой закладке «Поставщик данных» выбираем в списке «Microsoft.Jet.OLEDB.4.0» и жмем «Далее >>».

Во второй закладке «Соединение» в поле «1. Выберете или введите имя базы данных:» жмем «…» и привычным окном выбираем нужный файл БД. Здесь стоит отметить, что при выборе адреса, будет указан полный путь, начиная от корневой директории. При таком указании директории, становится неудобно использование программы на другом компьютере, так как приходится создавать все директории, что и на компьютере, где была создана программа. Чтобы не сталкиваться с такой проблемой, удобней указать относительный путь. Если в этом поле указать только имя файла БД, то программа будет открывать ее из той же директории, где она находится, в какую бы директории ее не разместили. Естественно файл БД должен находиться в той же папке, куда вы сохраняете свой проект Delphi.

Нажимаем кнопку «ОК».

Снова видим предыдущее окно, но уже с заполненной строкой. Снова нажимаем «ОК». Все, настройка свойства ConnectionString завершена.

## Свойство LoginPrompt компонента ADOConnection

Это свойство логического типа определяет, будет ли при подключении к БД запрашиваться пароль. Пока никакие пароли не нужны, поэтому, чтобы оно не мешало при работе с программой ставим ему значение False.

## Свойство Connected компонента ADOConnection

Свойство определяет, выполнено ли подключение в данный момент. Чтобы не подключаться к БД программно после запуска приложения, указываем True. Если после присвоению свойству значения True, Delphi не вывел никаких сообщений об ошибках, значит, все выполнено правильно. В случае ошибок, стоит проверить, правильно ли указано имя файла БД или «Поставщик данных».

## Свойство Connection компонента ADOTable

Выбираем из списка созданный и настроенный компонент ADOConnection. Как альтернатива этому свойству у компонента ADOTable есть собственное свойство ConnectionString, которым точно так же можно настроить подключение прямо к БД. Но если планируется работать с несколькими таблицами, гораздо удобней настроить подключение один раз и далее для всех таблиц пользоваться им.

## Свойство Name компонента ADOTable

Выбираем из списка необходимую таблицу из БД.

## Свойство Active компонента ADOTable

Свойство, определяющее режим работы компонента. Указываем значение True. Если сообщений об ошибках нет, все нормально. Если есть, проверьте предыдущие два свойства.

## Свойство DataSet компонента DataSource

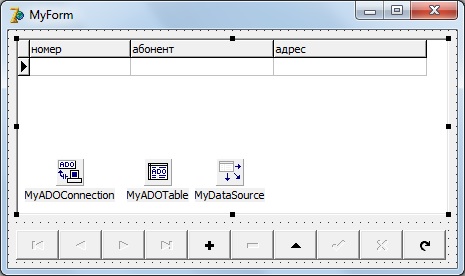
Из списка выбираем нужную таблицу, если их несколько. Обычно для удобства каждой таблице в БД соответствует своя пара ADOTable + DataSource.

На этом настройка невизуальных компонентов, использующихся для работы с данными БД, завешена. После этого нужно настроить пользовательскую форму для вывода информации и предоставления возможностей управления ей.

## Свойство DataSource компонентов DBGrid и DBNavigator

Свойство настраивается выбором из списка нужного компонента DataSource.

После всех этих операций получаем программу, которая будет подключаться к вашему файлу БД, выводить данные в простейшей табличной форме. Кнопочная панель позволит перемещаться по записям вперед, назад, перейти к последней или первой, добавлять новые записи, редактировать имеющиеся, а так же удалять не нужные записи.

  
Рисунок - Подключение Delphi с БД Access

# Список использованных источников

1. Хансен Г., Хансен Д. Базы данных и управление. - М.: Бином, 2014.
2. Дейт К. Введение в системы баз данных. - М.: "Вильямс", 2012.
3. Конноли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. - М.- С./П.- К., 2013.
4. Грофф Д.Р., Вайнберг П.Н. SQL: Полное руководство. - Киев: BMV, "Ирина", 2011.
5. Когаловский М.Р. Энциклопедия технологий баз данных. - М.: Финансы и статистика, 2013.
6. Григорьев Ю.А., Ревунков Г.И. Банки данных. - М.: МГТУ им. Баумана, 2012.
7. Уорсли Дж., Дрейк Дж. PostgreSQL. - Санкт-Петербург, Питер, - 2013.
8. Саймон А. Когаловский М.Р. Стратегические технологии баз данных: менеджмент на 2000 год. - М.: Бином, 2012.
9. Грабер М. Введение в SQL. - М.: ЛОРИ, 2013.
10. Барсегян А.А.,Куприянов М.С.,Степаненко В.В.,Холод И.И. Методы и модели анализа данных: OLAP и Data Mining. - Санкт-Петербург, БХВ-Петербург, 2014.
11. Хоменко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. - Санкт-Петербург, Корона, 2012.
12. Корнеев В.В., Гарев А. Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. - М. - Нолидж, 2013.