МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Колледж экономики, управления и права

Методические указания

по организации

практических занятий и самостоятельной работы студентов

по МДК.02.01

Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Часть 1

Специальности

09.02.05 Прикладная информатика (по отраслям)

Ростов-на-Дону
2018

Методические указания по МДК.02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.05 Прикладная информатика (по отраслям), предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

Составитель (автор): Д.А. Морозюк преподаватель колледжа ЭУП

Рассмотрены на заседании предметной (цикловой) комиссии специальностей 09.02.04 Информационные системы (по отраслям) и 09.02.05 Прикладная информатика (по отраслям)

Протокол № 1 от «31» августа 2018 г.

Председатель П(Ц)К специальности _____ С.В. Шинакова
личная подпись

и одобрены решением учебно-методического совета колледжа.

Протокол № 1 от «31» 08 2018 г

Председатель учебно-методического совета колледжа
_____ С.В. Шинакова
личная подпись

Рекомендованы к практическому применению в образовательном процессе.

Рецензенты:

_____ _____ _____
место работы должность ФИО

_____ _____ _____
место работы должность ФИО

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Колледж экономики, управления и права

Методические указания

по организации

практических занятий и самостоятельной работы студентов

по МДК.02.01

Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Часть 2

Специальности

09.02.05 Прикладная информатика (по отраслям)

Ростов-на-Дону
2018

Методические указания по МДК.02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности разработаны с учетом ФГОС среднего профессионального образования специальности 09.02.05 Прикладная информатика (по отраслям), предназначены для студентов и преподавателей колледжа.

Методические указания определяют этапы выполнения работы на практическом занятии, содержат рекомендации по выполнению индивидуальных заданий и образцы решения задач, а также список рекомендуемой литературы.

Составитель (автор): Д.А. Морозюк преподаватель колледжа ЭУП

Рассмотрены на заседании предметной (цикловой) комиссии специальностей 09.02.04 Информационные системы (по отраслям) и 09.02.05 Прикладная информатика (по отраслям)

Протокол № 1 от «31» августа 2018 г

Председатель П(Ц)К специальности _____ С.В. Шинакова
личная подпись

и одобрены решением учебно-методического совета колледжа.

Протокол № 1 от «31» 08 2018 г

Председатель учебно-методического совета колледжа
_____ С.В. Шинакова
личная подпись

Рекомендованы к практическому применению в образовательном процессе.

Рецензенты:

_____ _____ _____
место работы должность ФИО

_____ _____ _____
место работы должность ФИО

СОДЕРЖАНИЕ

Практическая работа №1

Составление технического задания

1. **Цель работы:** знакомство со стандартами, регламентирующими жизненный цикл разработки ИС. Приобретение практических навыков при разработке программного документа Техническое задание.

2. **Теоретические сведения**

Для обеспечения потребностей по разработке программного обеспечения (ПО) ИС необходимо взаимосвязанное совершенствование технических решений, технологий проектирования и программирования, инструментальных средств, а также обучения специалистов.

Стандарты для процессов, инструментальных средств и данных, которые отражают лучшую практику и защищают от неблагоприятных последствий, играют определенную роль в обеспечении указанных потребностей, в частности, поддерживая доверие потребителя к продуктам, услугам и технологиям разработки ПО.

Использование стандартов при разработке ПО ИС позволит обеспечить:

- повышение надежности;
- повышение эффективности применения и снижение затрат в сфере сопровождения программных средств (ПС);
- увеличение коэффициента повторного использования ПС общего назначения;
- повышение объективности оценок качества ПС;
- создание условий для конкуренции разработчиков на внутреннем рынке ПС;
- обеспечение конкурентоспособности отечественных ПС на мировом рынке.

Стандарты комплекса ГОСТ 34 на создание и развитие автоматизированных систем (АС) – обобщенные, но воспринимаемые как весьма жесткие по структуре жизненного цикла (ЖЦ) и проектной документации. Наиболее популярными можно считать ГОСТ 34.601-90 **(**Автоматизированные системы. Стадии создания) и ГОСТ 34.602-89 (Техническое задание на создание АС). Введение единой, достаточно качественно определенной терминологии, наличие достаточно разумной классификации работ, документов, видов обеспечения и др. способствует более полной и качественной стыковке разных систем, что особенно важно в условиях, когда разрабатывается все больше сложных комплексных АС.

В последние годы в стране идет интенсивная работа по гармонизации государственных стандартов Российской Федерации с международными стандартами ISO в области создания нормативной базы управления жизненным циклом программных средств и информационных систем. В основе стандартизации - ГОСТ Р ИСО/МЭК 12207-99 "Информационная технология. Процессы жизненного цикла программных средств", который является аутентичным переводом международного стандарта ISO/IEC 12207: 1995-08-01.

Техническое задание (ТЗ) на АС - утвержденный в установленном порядке документ, определяющий цели, требования и основные исходные данные необходимые для разработки АС и содержащий предварительную оценку экономической эффективности.

ТЗ содержит основные технические требования, предъявляемые к изделию и исходные данные для разработки; в ТЗ указываются назначение объекта, область его применения, стадии разработки документации, её состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов, полученных в ходе предпроектного обследования.

Как инструмент коммуникации в связке общения заказчик-исполнитель, техническое задание позволяет:

А) Обеим сторонам:

- представить готовый продукт;
- выполнить проверку готового продукта по пунктам (приёмочное тестирование – проведение испытаний);
- уменьшить число ошибок, связанных с изменением требований в результате их неполноты или ошибочности (на всех стадиях и этапах создания, за исключением испытаний).

Б) Заказчику:

- осознать, что именно ему нужно, опираясь на существующие на данный момент технические возможности и свои ресурсы;
- требовать от исполнителя соответствия продукта всем условиям, оговорённым в ТЗ.

В) Исполнителю:

– правильно понять суть задачи, показать заказчику «технический облик» будущего программного изделия или АС;
– спланировать выполнение проекта и работать по намеченному плану;
– отказаться от выполнения работ, не указанных в ТЗ.

3. Задание к выполнению

В соответствии с вариантом задания, определяющим предметную область на основании ГОСТ 34.602-89, разработать документ Техническое задание на создание АИС.

4. Варианты

1. ИС «Отдел кадров»;
2. ИС «Агентство аренды»;
3. ИС «Аптека»;
4. ИС «Ателье»;
5. ИС «Аэропорт»;
6. ИС «Библиотека»;
7. ИС «Кинотеатр»;
8. ИС «Поликлиника»;
9. ИС «Автосалон»;
10. ИС «Таксопарк».

5. Контрольные вопросы

1. Этапы развития проекта ИС?
2. Понятие модели жизненного цикла ПО ИС?
3. Обзор моделей жизненного цикла, их недостатки и преимущества?
4. Обзор и особенности методологии *RAD*?
5. Обзор стандартов, регламентирующих ЖЦ ПО ИС?
6. Назначение, содержание и степень адаптивности стандарта ГОСТ 34.601-90?
7. Стадии и этапы создания АС в соответствии с ГОСТ 34.601-90?
8. Назначение, содержание стандарта ГОСТ 34.602-89?
9. Назначение, содержание и степень адаптивности стандарта ГОСТ Р ИСО/МЭК 12207?
10. Содержание одного из основных процессов ЖЦ ПО ИС по ГОСТ Р ИСО/МЭК 12207 – Разработка?
11. Содержание стандарта ISO/IEC 15288?
12. Назначение программного документа Техническое задание на создание АС. Порядок разработки, согласования и утверждения документа?
13. Состав и содержание документа ТЗ?

Практическая работа №2

Проектирование информационной модели предметной области в нотации IDEF1X с помощью MS Visio

1. Цель занятия

Целью занятия является освоение технологии построения информационной модели логического и физического уровней в нотации IDEF1X с использованием пакета Microsoft Visio.

2. Задачи

Основными задачами занятия являются:

- приобретение студентами навыков построения информационной модели логического уровня;
- нормализации полученной модели;
- построения информационной модели физического уровня.

3. Краткие теоретические сведения

3.1. Понятие информационной модели. Уровни информационной модели

Методология IDEF1X – язык для семантического моделирования данных, основанный на концепции «сущность-связь».

Различают два уровня информационной модели: **логический** и **физический**.

Логическая модель позволяет понять суть проектируемой системы, отражая логические взаимосвязи между сущностями.

Различают 3 подуровня логического уровня модели данных, отличающиеся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity-Relationship Diagram (ERD);
- модель данных, основанная на ключах (Key Based Model (KB);
- полная атрибутивная модель (Fully Attributed Model (FA).

Физическая модель отражает физические свойства проектируемой базы данных (типы данных, размер полей, индексы). Параметры физической информационной модели зависят от выбранной системы управления базами данных (СУБД).

3.2. Основные элементы информационной модели логического уровня

3.2.1. Сущности и атрибуты

Сущность – это множество **реальных** или **абстрактных объектов** (людей, предметов, документов и т.п.), **обладающих общими атрибутами или характеристиками**. Любой объект системы может быть представлен только одной сущностью, которая должна быть уникально идентифицирована. *Именование сущности* осуществляется с помощью *существительного в единственном числе*. При этом имя сущности должно отражать **тип** или **класс** объекта, а не его **конкретный экземпляр** (например, **Студент**, а не **Петров**) (рис. 1).

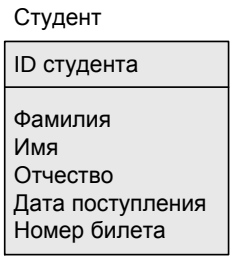


Рисунок 1 – Графическое представление сущности «Студент» в MS Visio

Любая сущность характеризуется набором атрибутов (**свойств**).

Атрибут сущности – характеристика сущности, то есть свойство реального объекта. Например, на рис. 1 атрибутами сущности **«Студент»** являются **«ID студента», «Фамилия», «Имя», «Отчество», «Дата поступления»** и **«Номер билета».**

В свою очередь, *атрибуты сущности* делятся на 2 вида: *собственные* и *наследуемые. Собственные* атрибуты являются уникальными в рамках модели. *Наследуемые* атрибуты передаются от сущности-родителя при установке связи с другими сущностями.

Первичный ключ (Primary Key, PK). Каждая сущность должна обладать *атрибутом* или *комбинацией атрибутов*, чьи значения *однозначно определяют* каждый *экземпляр сущности*. Эти атрибуты образуют *первичный ключ* сущности.

Внешний ключ (Foreign Key, FK). Если между двумя сущностями *имеется специфическое отношение* связи или *категоризации*, то *атрибуты*, входящие в *первичный ключ родительской* или *общей сущности*, *наследуются* в качестве *атрибутов сущностью-потомком* или *категориальной сущностью* соответственно. Эти атрибуты и называются внешними ключами. Наследуемый атрибут может использоваться в сущности в качестве части или целого первичного ключа, альтернативного ключа или не ключевого атрибута.

3.2.2. Отношения в IDEF1X-модели

При построении информационной модели различают следующие типы отношений между сущностями: *идентифицирующее, не идентифицирующее, не специфическое (многие-ко-многим) и отношения категоризации*.

Мощность отношения служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

3.3. Нормализация данных

Нормализация – это процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Процесс нормализации сводится к последовательному приведению структур данных к нормальным формам – формализованным требованиям к организации данных.

Первая нормальная форма (1НФ). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т.е. несколько значений для каждого экземпляра.

Вторая нормальная форма (2НФ). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый не ключевой атрибут полностью зависит от первичного ключа (не может быть зависимости от части ключа).

Третья нормальная форма (3 НФ). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой не ключевой атрибут не зависит от другого не ключевого атрибута (не должно быть зависимости между не ключевыми атрибутами).

4. Рекомендации по выполнению практического занятия

Практическое занятие выполняется группой студентов (2-3 человека) в пакете Microsoft Visio.

Данное занятие может выполняться на основе результатов функционального моделирования предметной области.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

5. Методика выполнения практического занятия

Упражнение 1. Построение логической информационной модели уровня «сущность-связь»

5.1. Составление пула – списка потенциальных сущностей

Информационная модель может быть построена на основе функциональной модели или без нее. Использование функциональной модели в качестве основы для информационного моделирования позволяет создать структуру базы данных, полностью соответствующей функциям предприятия. Названия всех интерфейсных дуг функциональной модели (выполненной в нотации IDEF0) заносятся в *пул* – список потенциальных сущностей. Только в данном случае информационная модель будет адекватна выполняемым функциям. Функциональная модель для рассматриваемого примера представлена в *приложении А*.

Список потенциальных сущностей (при использовании программного продукта MS Office Visio для функционального моделирования) должен быть составлен вручную. В случае использования CASE-средства *AllFusion Process Modeler* отчет по интерфейсным дугам генерируется автоматически. Список потенциальных сущностей для рассматриваемого примера будет представлен таблицей вида (рис. 2).

Arrow Name
Варианты заданий
График
Графическая часть
Задание
Замечания, дополнения
Курсовая работа
Литература
Методические указания
Оценка за курсовую работу
Положение о курсовом проектировании
Пояснительная записка
Преподаватель
Расчеты
Список литературы
Студент

Рисунок 2 – Пул – список потенциальных сущностей

Теперь из этого списка необходимо выделить сущности, остальные интерфейсные дуги будут преобразованы в атрибуты сущностей.
В качестве сущностей выделим следующие:
1) задание;
2) пояснительная записка;
3) курсовая работа;
4) положение о курсовом проектировании;
5) студент;
6) преподаватель;
7) график;
8) методические указания.

5.2. Создание логической модели «сущность-связь»

1. Запустите MS Visio.
2. На закладке выбора шаблона выберите категорию *Программное обеспечение и базы данных* и в ней элемент *Схема модели базы данных*. Нажмите кнопку *Создать* в правой части экрана.
3. Установите необходимые параметры страницы (масштаб, ориентация страницы).
4. MS Visio поддерживает различные нотации моделей баз данных. Для того чтобы задать нотацию IDEF1X, необходимо выбрать пункты меню База данных → Параметры → Документ. В открывшемся окне на вкладке Общие установить переключатель в меню Набор символов на IDEF1X. Меню Имена, видимые на схеме, позволяет указать, какие имена атрибутов сущности будут отображены на диаграмме (концептуальные, физические или оба варианта одновременно). В данном случае для логического представления информационной модели необходимо выбрать пункт *Концептуальные имена* (рис. 3).

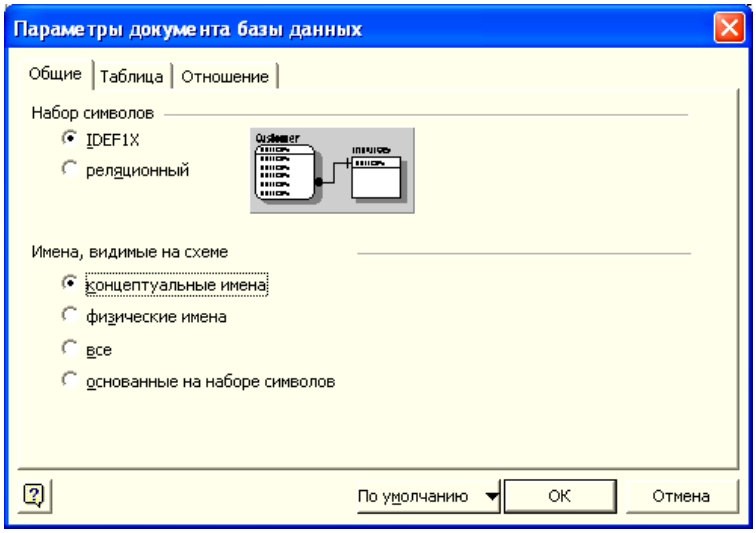


Рисунок 3 – Настройка параметров модели

В закладке *Отношение* окна *Параметры документа базы данных* в меню *Показывать* отметить галочкой пункт *Мощность*, в меню *Отображение вида* выбрать пункт *Показывать вербальную фразу*, снять галочку в пункте *Обратный текст* (рис. 4). Данные настройки позволят отобразить имя и мощность связи в модели.

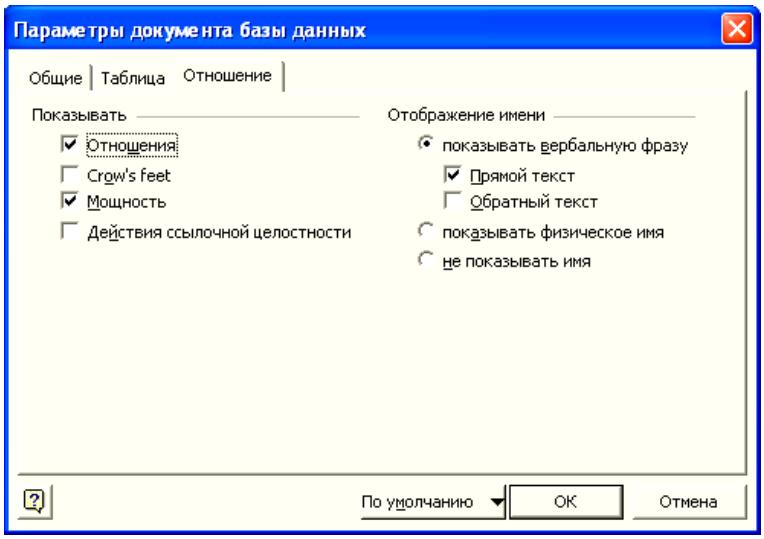


Рисунок 4– Настройка вида отношений информационной модели

5. Для того чтобы создать сущность, необходимо перетащить элемент  на рабочее поле. Переход в режим редактирования сущности осуществляется двойным щелчком по сущности или по нажатию правой кнопки мыши и выбора пункта меню *Свойства базы данных*.

Чтобы задать имя сущности, в окне *Свойства базы данных* нужно выбрать категорию *Определение*, снять галочку в пункте *Синхронизация имен при вводе* (в противном случае, физическое и логическое имя сущности будут совпадать, что по практическим соображениям не всегда удобно) и задать концептуальное имя сущности. Руководствуясь данным алгоритмом, создадим 8 сущностей, определенных в пункте 5.1 (см. рис. 5).

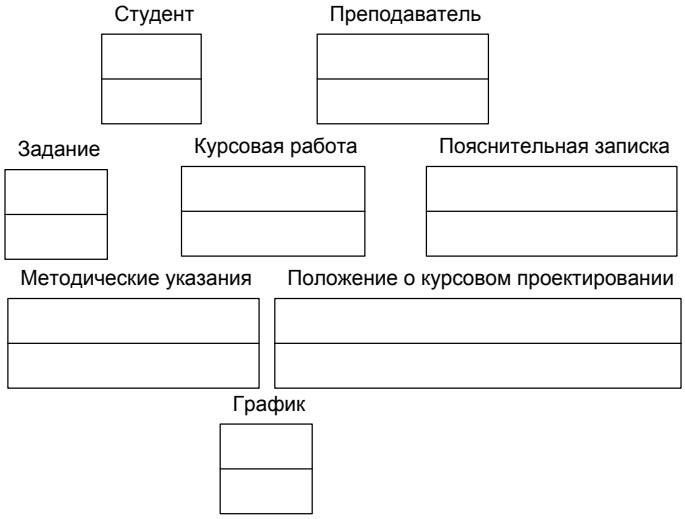


Рисунок 5– Сущности информационной модели логического уровня

6. Далее необходимо установить связи между сущностями.

Сначала составим описание предметной области на естественном языке.
Любой студент должен выполнить одну или несколько курсовых работ.
Каждая курсовая работа должна выполняться одним студентом (в идеале).
Каждая курсовая работа выполняется в соответствии с методическими указаниями и положением о курсовом проектировании.
Курсовая работа сдается по графику.
Курсовая работа оформляется в виде пояснительной записки.
Преподаватель проводит консультации, проверяет и ставит оценку за курсовую работу.
Таким образом, сформулируем имена связей:
СТУДЕНТ выполняет ***КУРСОВУЮ РАБОТУ***.
ПРЕПОДАВАТЕЛЬ проверяет ***КУРСОВУЮ РАБОТУ***.
КУРСОВАЯ РАБОТА выполняется в соответствии с ***ЗАДАНИЕМ***.
КУРСОВАЯ РАБОТА оформляется в виде ***ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ***.
МЕТОДИЧЕСКИЕ УКАЗАНИЯ определяют требования к ***КУРСОВОЙ РАБОТЕ***.
КУРСОВАЯ РАБОТА организуется согласно ***ПОЛОЖЕНИЮ ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ***.
КУРСОВАЯ РАБОТА сдается по ***ГРАФИКУ***.
Во всех случаях сущность *Курсовая работа* является дочерней, за исключением связи с сущностью *Пояснительная записка*. Определим типы связей и построим модель (см. рис. 6). В дальнейшем можно будет подкорректировать связи между сущностями.

Чтобы установить **связи** между сущностями, необходимо перетащить на рабочую область элемент , поднести один конец стрелки к родительской сущности, другой – к дочерней.
Примечание. При правильном связывании каждая сущность будет подсвечена красным цветом.
В MS Office Visio по умолчанию используется *не идентифицирующее* отношение. Чтобы изменить **тип связи**, необходимо двойным щелчком по связи открыть окно *Свойства базы данных* и в категории в категории *Прочее* указать тип отношения (идентифицирующее, не идентифицирующее). В этой же категории указывается мощность связи (см. рис. 6).

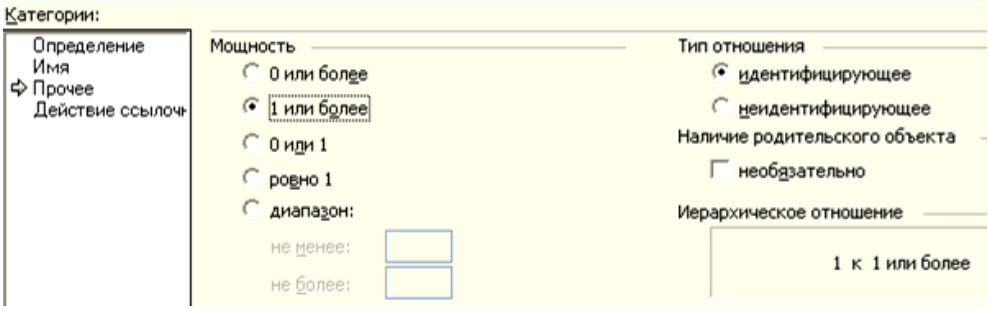


Рисунок 6 – Определение типа связи и мощности

Примечание. Кроме того, при не идентифицирующем отношении нужно указать, является ли наличие родительской сущности обязательным (т.е. может ли существовать экземпляр дочерней сущности, если не существует экземпляра родительской). Если наличие родительского объекта является необязательным, графически это отобразится в виде не закрашенного ромба со стороны родительской сущности.
Следующий шаг – в категории *Имя* в поле *Вербальная фраза* нужно указать имя отношения (рис. 7). Также можно указать имя связи в поле *Обратная фраза* для спецификации отношения потомок-родитель (в нашем случае обратная фраза отображаться не будет).
Примечание. Все изменения при закрытии окна свойств сохраняются автоматически.

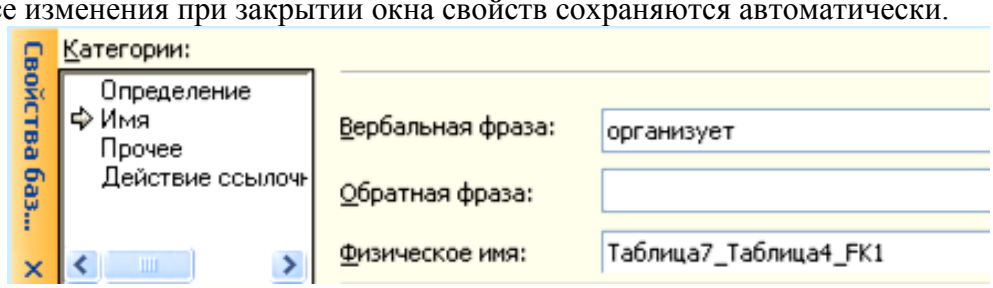


Рисунок 7 – Определение имени отношения

После определения имен, типов связей и задания мощностей получим информационную модель, представленную на рис. 8.

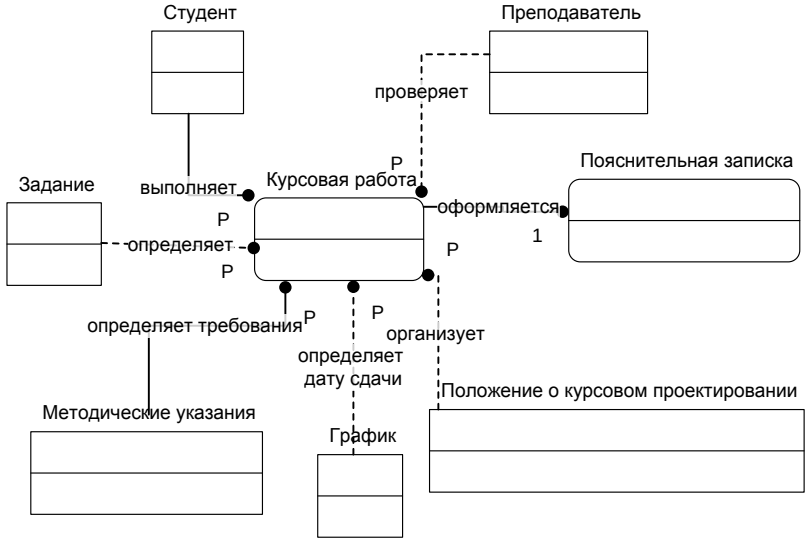


Рисунок 8 – Информационная модель уровня «сущность-связь»

Упражнение 2. Разработка логической модели данных, основанной на ключах

1. Необходимо определить ключевые атрибуты для каждой сущности, обращая внимание на то, что дочерние сущности наследуют ключевые атрибуты от родительских (см. рис. 10).

Для этого двойным щелчком мыши по сущности откроем окно редактирования ее свойств, перейдем в категорию *Столбцы*, по нажатию кнопки *Добавить* введем имя поля (например, для сущности *Задание* ключевым атрибутом будет являться *Вариант задания*). Чтобы сделать атрибут ключевым, необходимо отметить галочкой пункт *РК* (рис. 9). Данное поле становится обязательным автоматически.

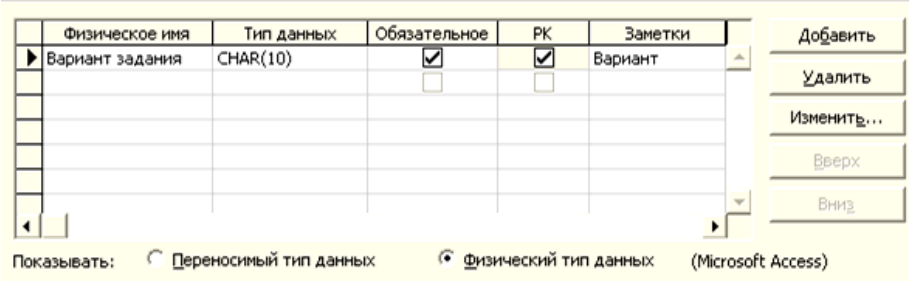


Рисунок 9 – Определение ключевого атрибута

Аналогичным образом зададим ключевые атрибуты для всех сущностей информационной модели. Результат представлен на рис. 10.

Как видно из рисунка 10 по сравнению с информационной моделью уровня «сущность-связь», был изменен тип связи между сущностями *Методические указания* и *Курсовая работа*, поскольку ключевые атрибуты сущности *Методические указания* для сущности *Курсовая работа* будут являться избыточными (зная номер зачетной книжки, можно узнать специальность и курс, на котором учится студент).

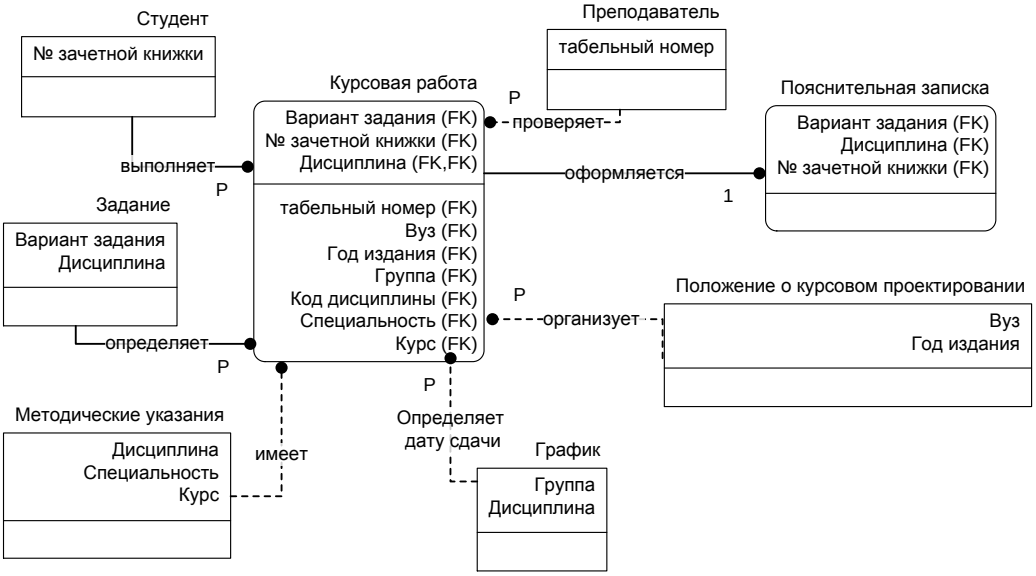


Рисунок 10 – Информационная модель с ключевыми атрибутами

2. Кроме того, отметим, что три сущности (Задание, График, Методические указания) содержат одинаковые атрибуты Дисциплина. Это является некорректным. Чтобы устранить данную ошибку, выделим одноименную сущность и свяжем ее идентифицирующими связями с вышеуказанными сущностями (рис. 11).

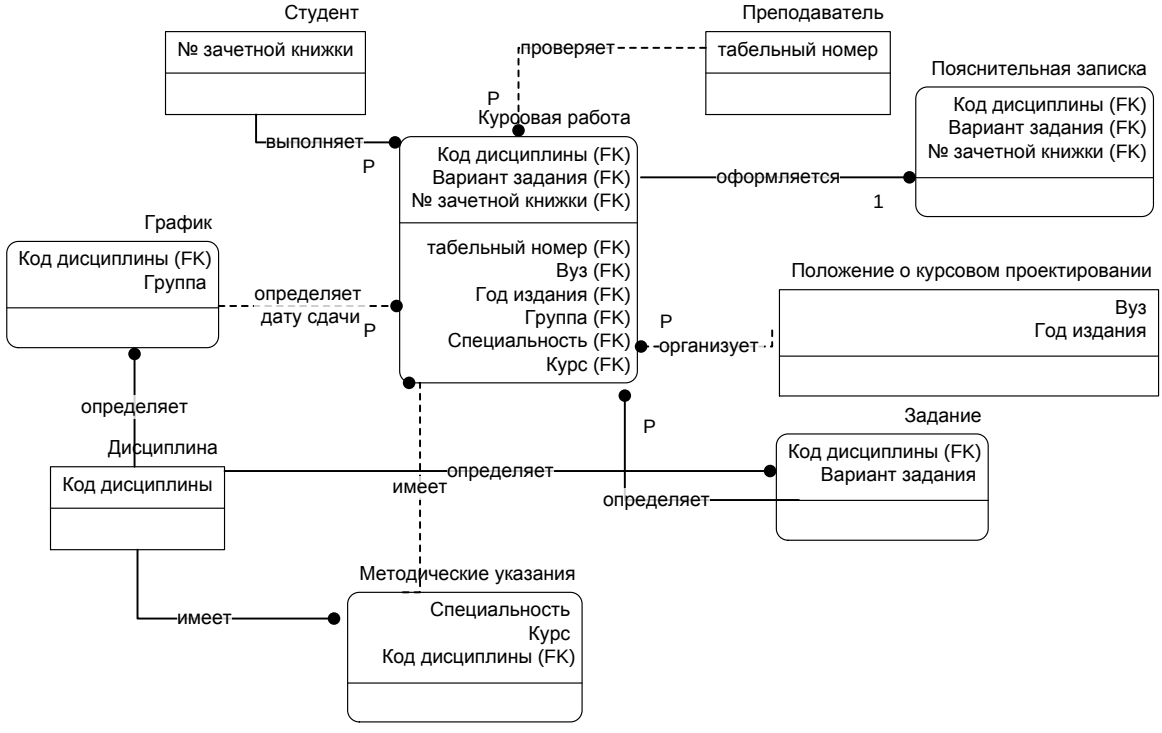


Рисунок 11 – Скорректированная информационная модель, основанная на ключах

Упражнение 3. Создание полной атрибутивной модели

Для того чтобы получить полную атрибутивную модель, необходимо дополнить сущности не ключевыми атрибутами. Дополненная модель представлена на рисунке 11.

Примечание. Если атрибут не является обязательным, нужно убедиться, что в окне *Свойства базы данных* в категории *Столбцы* в пункте *Обязательное* не стоит галочка. Не обязательные к заполнению атрибуты справа от имени имеют пометку *(О)*.

Упражнение 4. Нормализация полной атрибутивной модели

1. Проверим, все ли атрибуты имеют атомарные значения, т.е. среди атрибутов не должно встречаться повторяющихся групп, нескольких значений для каждого экземпляра (например, номер телефона_1, номер телефона_2). Видим, что атрибут *Авторы* в сущности *Методические указания* не удовлетворяет требованиям 1 НФ (у методических указаний может быть несколько авторов). Необходимо выделить сущность, которая будет содержать сведения об авторах методических указаний. Поскольку авторами всегда являются преподаватели вузов, новую сущность выделять не имеет смысла, свяжем сущности *Методические указания* и *Преподаватель*, предварительно удалив атрибут *Авторы*. Остальные атрибуты соответствуют 1 НФ. Атрибутивная модель, приведенная к 1 НФ, представлена на рис. 12.

Рисунок 12 – Полная атрибутивная модель

2. Приведем модель ко 2 НФ. Проверим, все ли атрибуты зависят от составного ключа, а не от его части. Проверка показала, что все не ключевые атрибуты сущностей полностью зависят от составного ключа. Значит, модель удовлетворяет требованиям 2 НФ.

3. Проверим, есть ли транзитивная зависимость между не ключевыми атрибутами. Проверка показала, что взаимозависимости между не ключевыми атрибутами нет. Таким образом, модель, представленная на рисунке 13, приведена к 3 НФ.

Примечание. К нормализации относились также действия, выполненные в п. 2 упражнения 2.

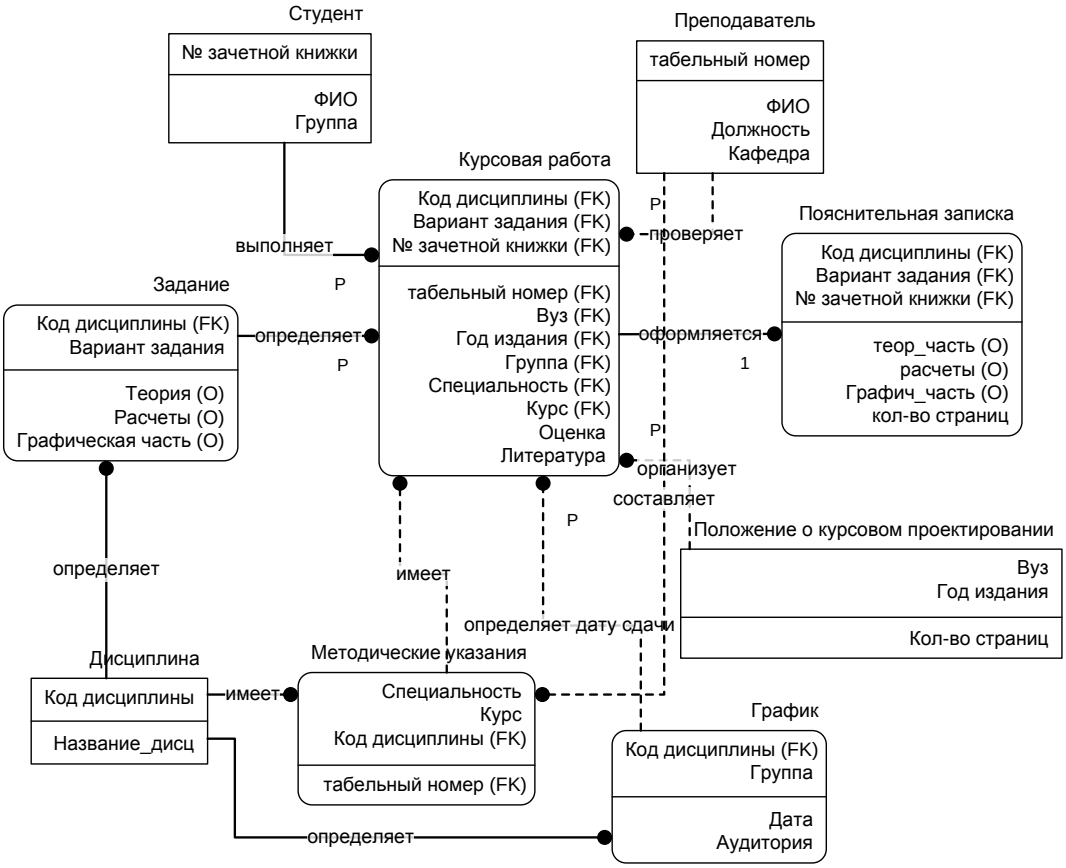


Рисунок 13 – Информационная модель, приведенная к 1 НФ

<u>Упражнение 5</u>. Создание физической модели

1. Необходимо переключиться на физический уровень представления информационной модели. Для этого нужно выбрать пункты меню *База данных* → *Параметры* → *Документ*. В открывшемся окне на вкладке *Общие* установить переключатель в меню *Имена, видимые на схеме*. В данном случае для физического представления информационной модели необходимо выбрать пункт *Физические имена* (рис. 14).

2. В закладке *Таблица* окна *Параметры документа базы данных* в меню *Отображать* выбрать пункт *Вертикальные линии,* в меню *Типы данных – Показывать физические* и в меню *Порядок – Физический порядок* (рис. 15).

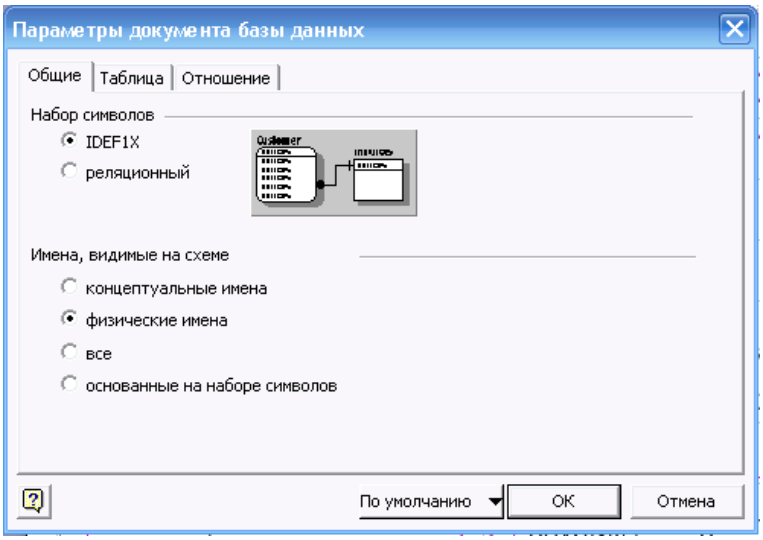


Рисунок 14 – Настройка параметров модели

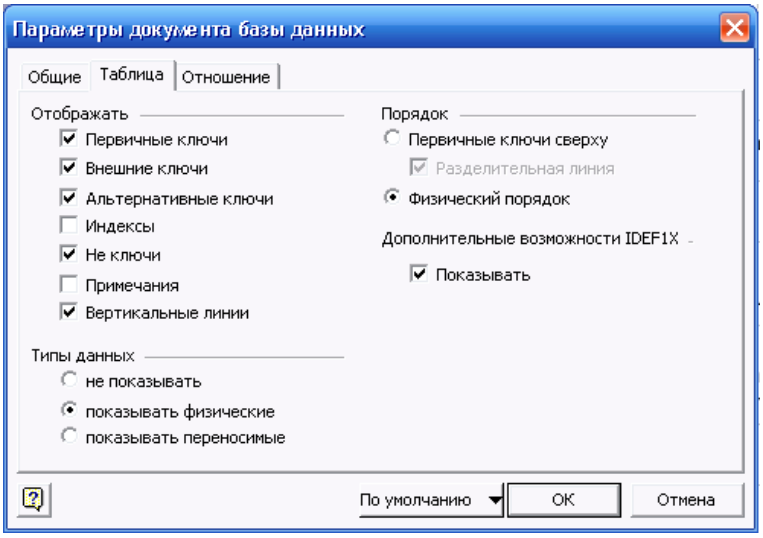


Рисунок 15 – Настройка параметров отображения сущности

3. В закладке Отношение окна Параметры документа базы данных в меню Отображение вида выбрать пункт Показывать физическое имя (рис. 16).

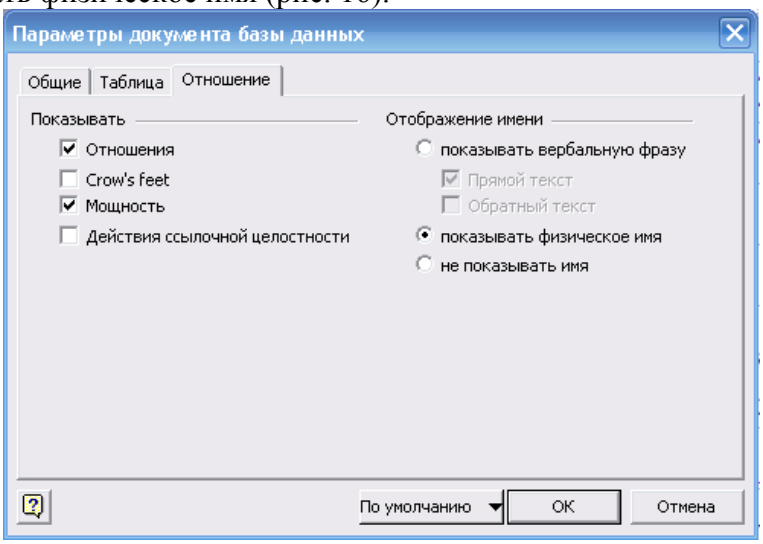


Рисунок 16 – Настройка вида отношений информационной модели

По окончании настройки документа информационная модель будет выглядеть, как представленная на рис. 17.

4. Для каждого атрибута (поля) необходимо определить тип данных.

Типы данных можно представить в виде правил, ограничивающих вид сведений, которые могут быть введены в каждый столбец таблицы базы данных. Например, чтобы в поле, которое предназначено только для дат, нельзя было ввести имя, этому полю назначается тип данных «Дата».

Примечание *(Выбор между переносимыми и физическими типами данных).*

Переносимые типы данных — это обобщенные типы данных, соответствующие в разных системах баз данных простым, совместимым между собой физическим типам.

Физические типы данных — это типы данных, поддерживаемые целевой базой данных.

Щелкните сущность, содержащую атрибуты, для которых требуется установить типы данных.

В окне *Свойства базы данных* в списке *Категории* выберите вариант *Столбцы*.

Под списком столбцов установите переключатель в положение *Физический тип данных*.

В группе *Тип данных* для каждого атрибута выберите необходимый вариант из множества альтернатив (рис. 18). Описание типов данных приведено в *Приложении Б*.

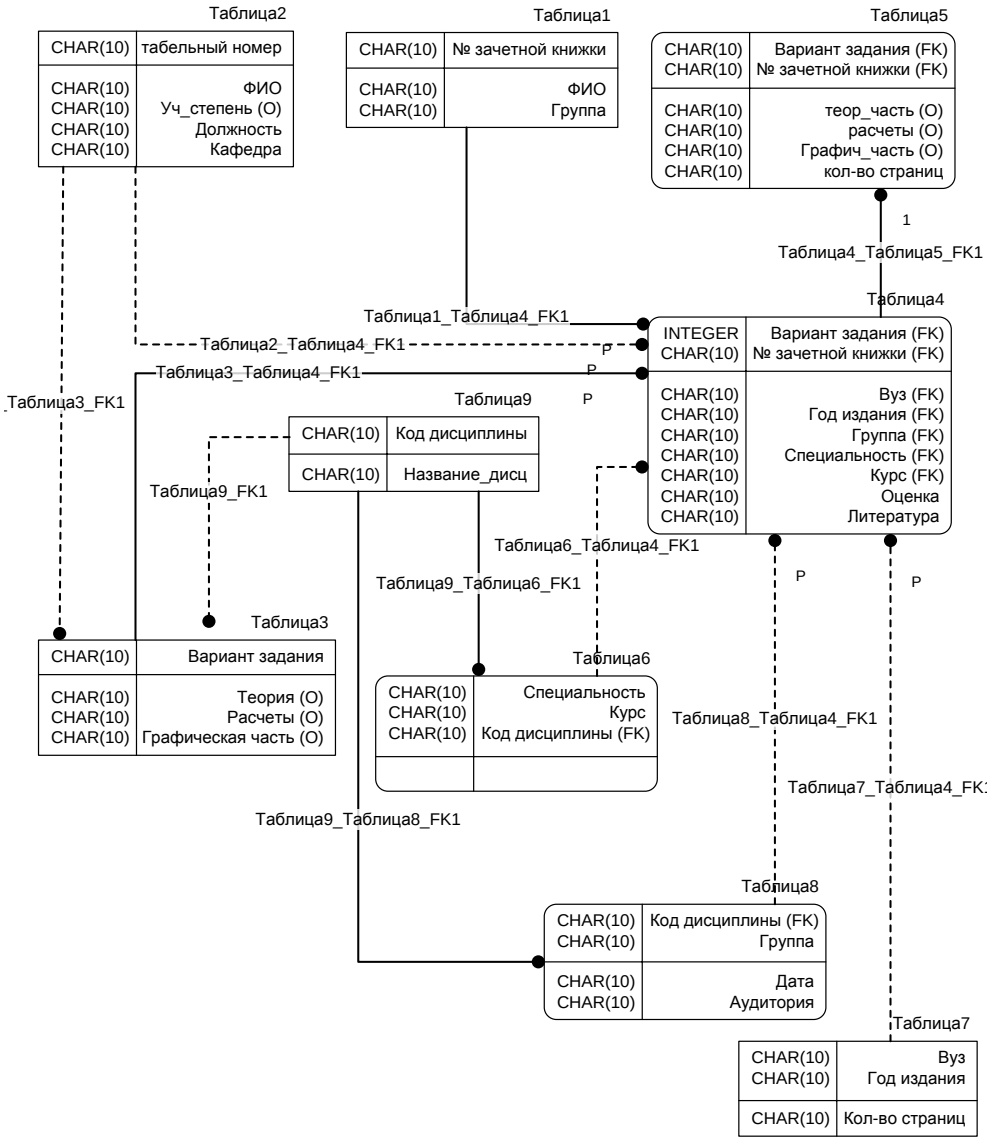


Рисунок 17 – Вид физической модели

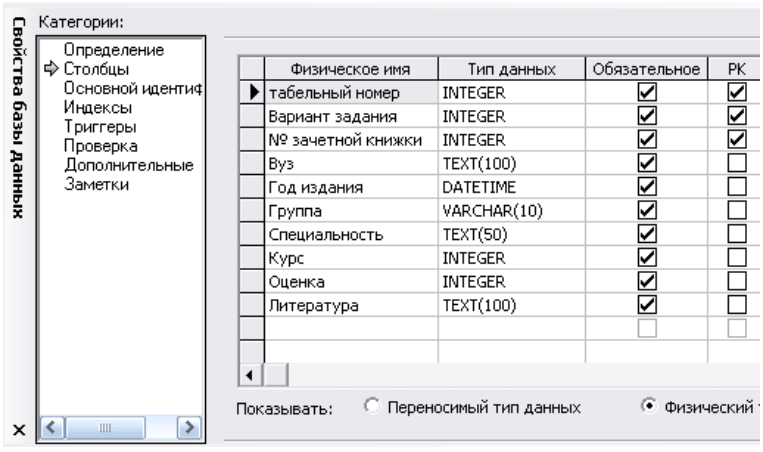


Рисунок 18– Определение типа данных атрибутов сущности

После того, как будут выполнены все действия, физическая модель будет выглядеть, как показано на рис. 19.

Таким образом, проделав все вышеперечисленные действия, получим информационную модель физического уровня, на основе которой может быть сгенерирована схема БД (в нашем случае в MS Office Access).

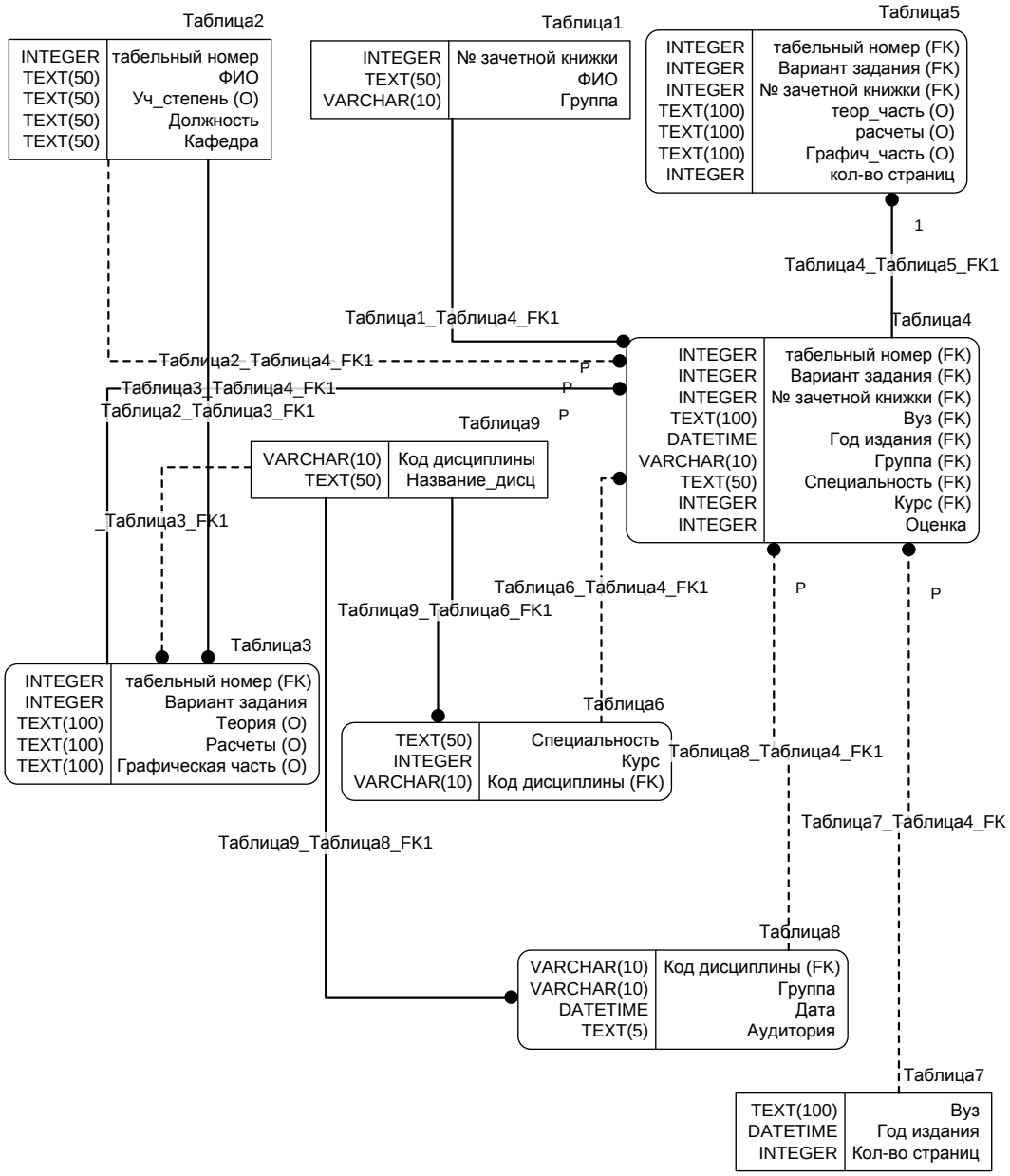


Рисунок 19 – Физическая модель базы данных

6. Задание

В соответствии с вариантом задания, определенным преподавателем, последовательно выполнить следующие действия:

1) создать информационную модель логического уровня (выполнить упражнения 1-3). Минимальное количество сущностей – 4;

2) провести нормализацию полученной модели (упражнение 4);

3) на основе нормализованной логической модели построить информационную модель физического уровня (упражнение 5).

7. Варианты

1. Проектирование ИС «Отдел кадров»;
2. Проектирование ИС «Агентство аренды»;
3. Проектирование ИС «Аптека»;
4. Проектирование ИС «Ателье»;
5. Проектирование ИС «Аэропорт»;
6. Проектирование ИС «Библиотека»;
7. Проектирование ИС «Кинотеатр»;
8. Проектирование ИС «Поликлиника»;
9. Проектирование ИС «Автосалон»;
10. Проектирование ИС «Таксопарк».

8. Порядок выполнения работы

Для выполнения работы необходимо:

а) повторить правила техники безопасности при работе с вычислительной техникой;

б) изучить соответствующий раздел лекционного курса, а также теоретическую часть настоящего методического указания;

в) выполнить лабораторную работу согласно описанной в пункте 5 методике в соответствии с вариантом задания;

г) в соответствии с требованиями, приведенными в разделе 8 практикума, оформить отчет по лабораторной работе;

е) защитить лабораторную работу в соответствии с требованиями преподавателя.

9. Требования к содержанию и оформлению отчета

Отчет должен содержать:

1) титульный лист;
2) название практического занятия, цель;
3) пул – список потенциальных сущностей;
4) нормализованную информационную модель логического уровня;
5) информационную модель физического уровня;
6) выводы по проделанной работе.

Отчет должен быть представлен в бумажном виде.

10. Контрольные вопросы

1. Какие диаграммы позволяет строить нотация IDEF1X?
2. Для чего предназначена диаграмма «сущность-связь»?
3. Чем отличается полная атрибутивная модель от диаграммы «сущность-связь»?
4. Какие виды отношений существуют и чем они отличаются?
5. Что представляет собой нормализация?
6. В чем разница между логическим уровнем модели данных и физическим?

11. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вендров А. М. Практикум по проектированию программного обеспечения экономических информационных систем. : - М.: Финансы и статистика, 2004.-190 с.

2. Вендров А. М. Проектирование программного обеспечения экономических информационных систем -М.: Финансы и статистика, 2006.-543 с.

3. Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite. – М.: ДИАЛОГ-МИФИ, 2005 – 432с.

ПРИЛОЖЕНИЕ А

Функциональная модель процесса «Выполнить курсовую работу»

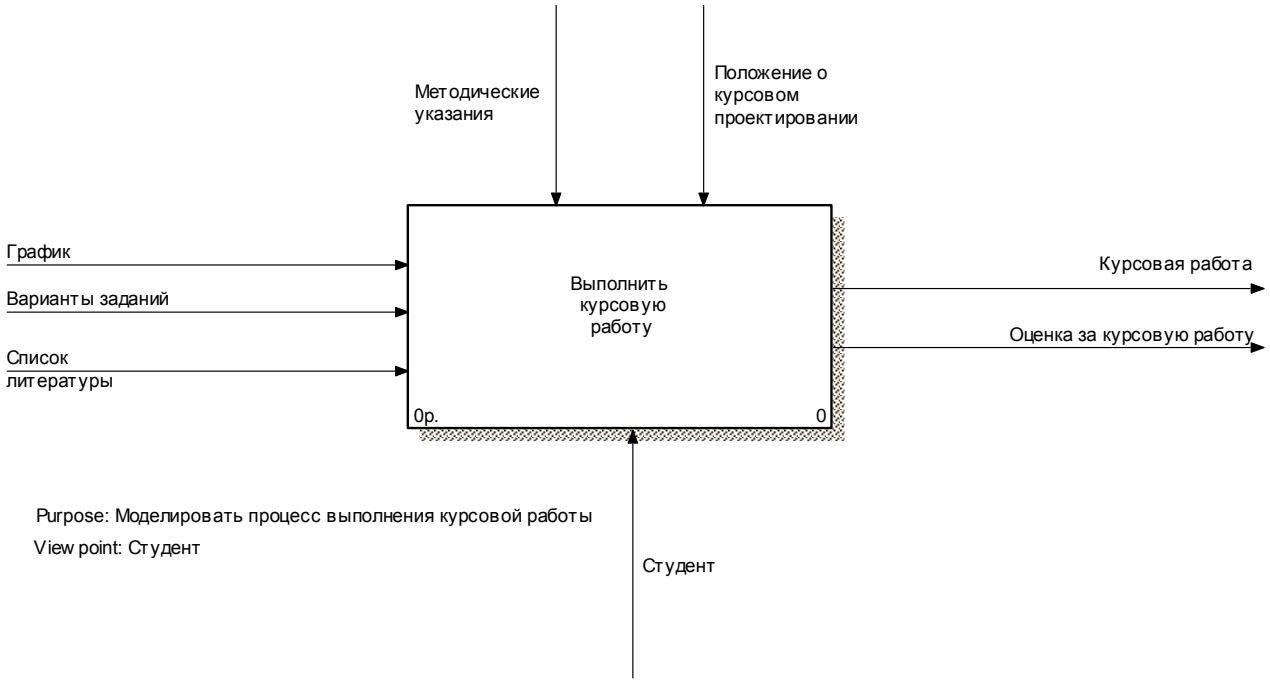


Рисунок А.1 – Контекстная диаграмма процесса выполнения курсовой работы

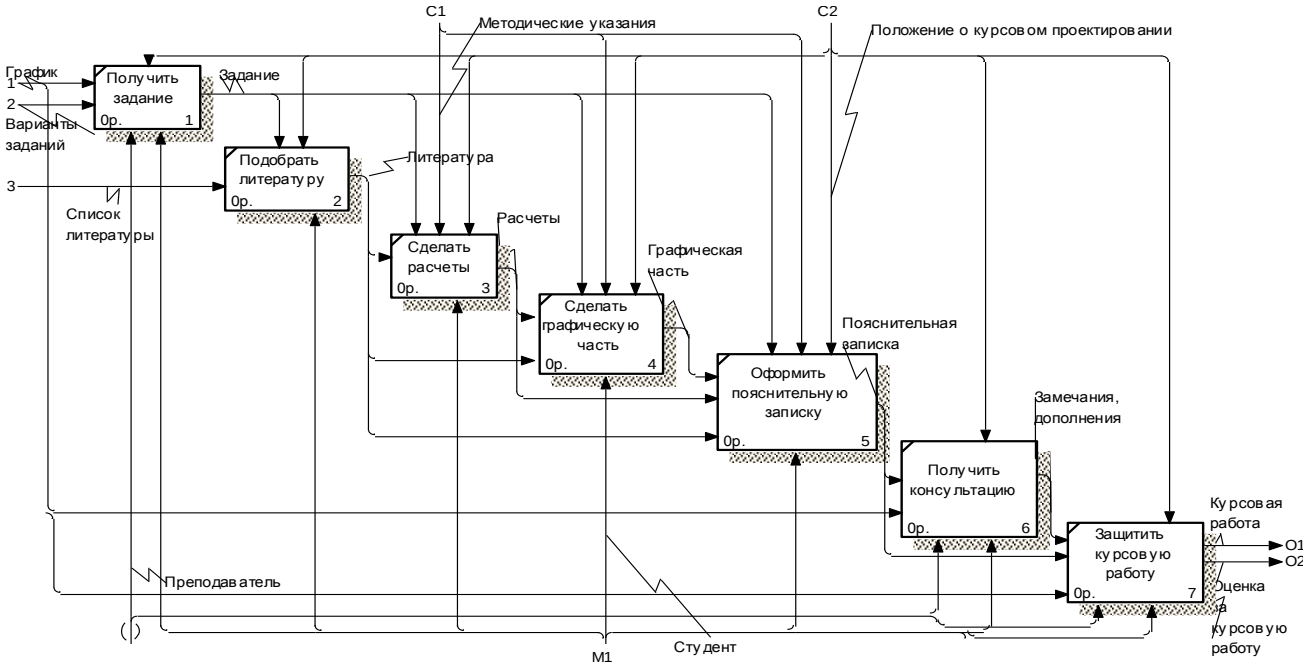


Рисунок А.2 – Диаграмма декомпозиция блока А0

ПРИЛОЖЕНИЕ Б

Типы данных Microsoft Office Access

1. Символьные типы

Символьные типы используются для представления как строк символов, так и отдельных символов.

Таблица Б.1 – Перечень символьных типов

Тип данных	Назначение	Размер
CHAR	Строковый тип	до 32767 байт. по умолчанию 1 байт
VARCHAR	Тоже, что и CHAR	
LONG VARCHAR	Символьный тип произвольной длины. Аналог MEMO-полям в dBase, FoxPro, Access	Длина произвольная. Ограничена максимальным размером файлов базы данных (2 гигобайта)
TEXT	Тоже, что и LONG VARCHAR	

2. Числовые типы

Числовые типы предназначены для обозначения целых, вещественных и денежных типов.

Таблица Б.2 – Перечень числовых типов

Тип данных	Диапазон значений	Точность - число знаков после запятой	Размер
INTEGER	от -2 147 483 648 до +2 147 483 647	0	4 байта
SMALLINT	от -32 768 до +32 767	0	2 байта
REAL	от -3.4 e-38 до 3.4 e+38	до 6	4 байта
DOUBLE	от -1.797 e-308 до +1.797 e+308	до 15	8 байт
DECIMAL	числа, состоящие из N цифр с M цифрами в дробной части. По умолчанию N=30, M=6	M	сколько требуется
NUMERIC	Тоже, что и DECIMAL		

3. Типы дата/время

Типы дата/время предназначены для хранения времени, дат и дат совместно с временем.

Таблица Б.3 – Форматы представления данных типа дата/время, определяемые по умолчанию

Тип данных	Формат, используемый по умолчанию
DATETIME	'YYYY-MM-DD HH:NN:ss.SSS'

- **YYYY** – четыре цифры, обозначающие год:
- **MM** – две цифры, обозначающие месяц:
- **DD** – две цифры, обозначающие день:
- **HH** – две цифры, обозначающие часы:
- **NN** – две цифры, обозначающие минуты:
- **ss** – две цифры, обозначающие секунды:
- **SSS** – три цифры, обозначающие доли секунд.

По умолчанию составляющие времени HH, NN, ss, SSS принимаются равными нулю, а DD - единице.

4. Двоичные типы

Двоичные типы предназначены для представления двоичных данных, включая изображения и другую информацию, не обрабатываемую собственными средствами СУБД.

Таблица Б4 -

Таблица Б.4 – Двоичные типы

Тип данных	**Назначение**	**Размер**
BIT	Тип для представления значений 0 и 1. Аналог полей типа Logical в dBase, FoxPro	1 байт
BINARY	Тоже, что и CHAR, за исключением операций сравнения. В отличии от CHAR, данные этогопо умолчанию 1 байт типа сравниваются на полное совпадение двоичных кодов байтов	до 32767 байт
LONG	Тип для представления двоичных данных произвольной длины	Длина произвольная. Ограничена максимальным размером файлов базы данных (2 гиго- байта)

Практическая работа №3

Функциональное моделирование предметной области в нотации IDEF0 с помощью MS Visio

1. Цель занятия

Целью занятия является получение навыков создания и редактирования функциональных моделей в нотации IDEF0 с помощью MS Visio.

2. Задачи

Основными задачами занятия являются:

– приобретение студентами навыков построения функциональной модели;
– приобретение студентами навыков редактирования функциональной модели.

3. Краткие теоретические сведения

3.1. Основные сведения по методологии IDEF0

Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Цель моделирования. Модель не может быть построена без четко сформулированной цели. Пример цели: «Описать функциональность предприятия с целью написания спецификаций ИС».

Точка зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Цель и точка зрения документируются.

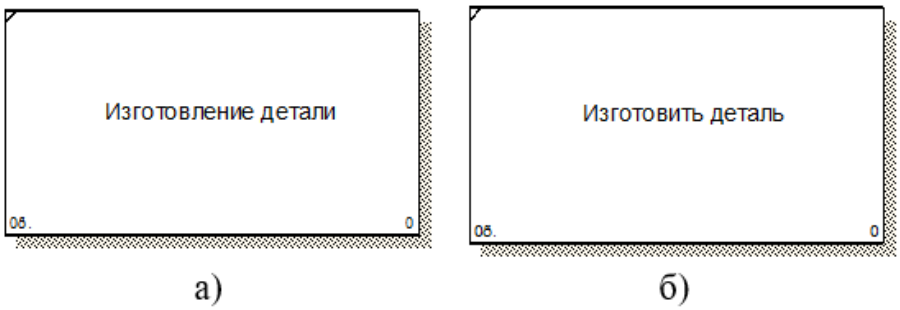
Основные элементы IDEF0-модели

В основе методологии IDEF0 лежат 4 основных понятия:

– функциональный блок;
– интерфейсная дуга (стрелка);
– декомпозиция;
– глоссарий.

1. Функциональный блок

Функциональные блоки обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Графически функциональные блоки изображаются в виде прямоугольников. Все блоки должны быть названы и определены. Имя функционального блока должно быть выражено сочетанием отглагольного существительного, обозначающего процесс, или глаголом (рис. 1):



а) б)

Рисунок 20 – Примеры работ

Определение функционального блока заносится в глоссарий или словарь работ (Activity Dictionary).

Все функциональные блоки модели нумеруются. Номер состоит из префикса и числа. Может использоваться префикс любой длины, но обычно используется префикс А. Контекстная (корневая) работа (функциональный блок) имеет номер А0.

2. Интерфейсная дуга (стрелка – Arrow)

Взаимодействие функциональных блоков с внешним миром и между собой описывается в виде интерфейсных дуг (стрелок). Стрелки представляют собой некую информацию и обозначаются существительными (например, «Заготовка», «Изделие») или именуемыми сочетаниями (например, «Готовое изделие»). Все стрелки должны быть определены. Определения заносятся в словарь стрелок – глоссарий (Arrow Dictionary).

В IDEF0 различают 4 типа стрелок (рис.2).

Каждая стрелка имеет свое расположение относительно функционального блока.

Рисунок 21 – Типы стрелок

Вход (Input) – материал или информация, которые используются или преобразуются работой для получения результата (выхода). Стрелка Input рисуется входящей в левую грань работы.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Рисуется как входящая в верхнюю грань работы.

Выход (Output) – материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Изображается исходящей из правой грани работы.

Механизм (Mechanism) – ресурсы, которые выполняют работу, например, персонал предприятия, станки, устройства и т.д. Рисуется как входящая в нижнюю грань работы.

3. Глоссарий

Набор определений, ключевых слов и т.д., которые характеризуют каждый объект модели.

4. Декомпозиция

Разбиение системы на крупные фрагменты – функции, функции – на подфункции и т.д. до конкретных процедур.

Модель может содержать 4 типа диаграмм:

- контекстную (в каждой модели может быть только 1 контекстная диаграмма);
- декомпозиции;
- дерева узлов;
- только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой общее описание системы и ее взаимодействия с внешней средой.

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов – диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и т.д., до достижения нужного уровня подробности описания.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами.

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения либо для специальных целей.

Все диаграммы имеют нумерацию. Контекстная диаграмма имеет номер A-0, декомпозиция контекстной диаграммы – номер A), остальные диаграммы-декомпозиции – номера по соответствующему узлу (например, A1, A2, A21 и т.д.).

3.2. Особенности MS VISIO

3.2.1 Для построения функциональной модели бизнес-процесса, используя MS Visio, необходимо запустить программу.

В открывшейся программе выбрать: Файл – Фигуры – Блок-схема – Фигуры схемы IDEF0.

3.2.2 Используемые блоки для построения функциональной модели:

Блок заголовка – рамка, которую необходимо установить на весь лист и оформить в соответствии с правилами оформления диаграмм в нотации IDEF0

Блок текста необходим для описания точки зрения и цели на контекстной диаграмме.

Блок действия – для описания работ, рассматриваемых в процессе.

Одностороннее соединение – элемент изображения интерфейсных дуг, таких как вход/выход, механизм/управление.

Соединительная линия IDEF0 – объект для изображения интерфейсных дуг между работами в модели.

4. Методика выполнения

В качестве примера рассматривается процесс выполнения студентом курсовой работы (курсового проекта).

4.1. Создание контекстной диаграммы

1. Запустите MS Visio.
2. На закладке выбора шаблона выберите категорию *Блок-схема* и в ней элемент *Схема IDEF0*. Нажмите кнопку *Создать* в правой части экрана.
3. Окно программы примет вид, подобный рис. 3.

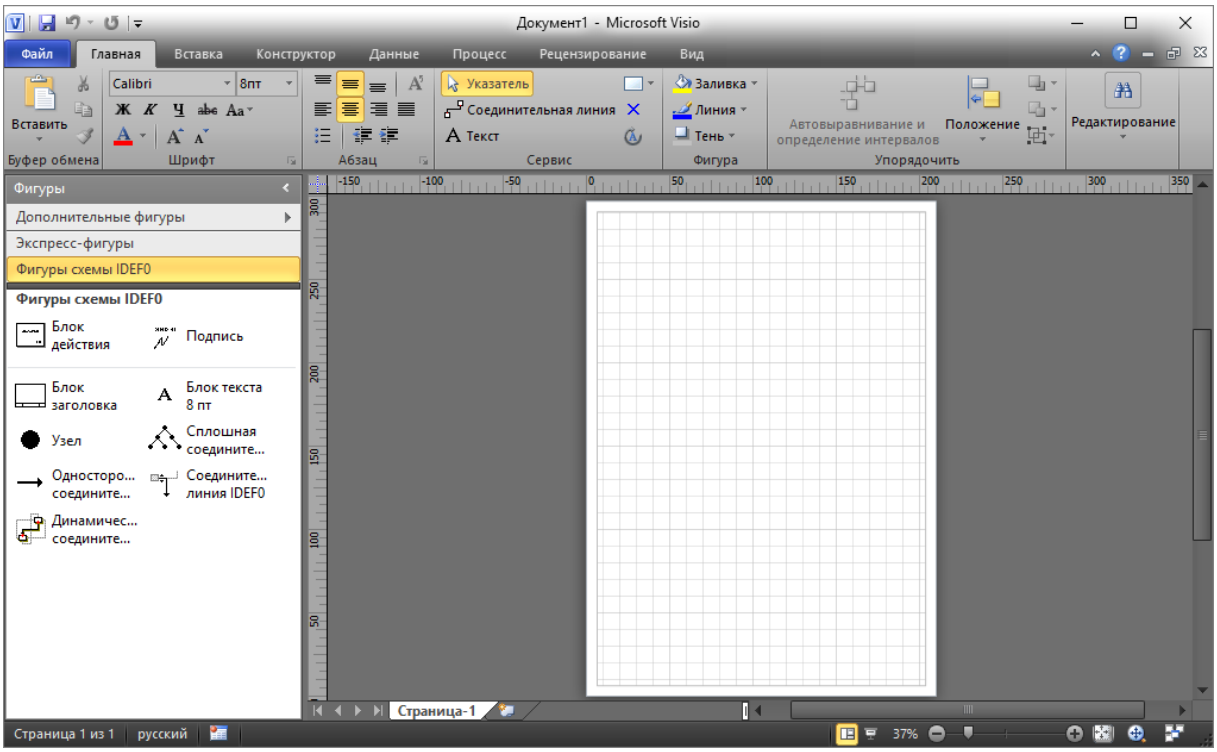


Рисунок 22 – Окно программы с выбранным шаблоном

4. Создание мастерской страницы.

4.1. Для удобства переведите страницу в альбомный вид: Конструктор – Параметры страницы – Ориентация – Альбомная;

4.2. Перетащите Блок заголовка на пустую страницу, удерживая нажатой правую кнопку мыши;

4.3. Заполнить поле «Заголовок» (рисунок 4), предложенное в открывшемся окне: внести номер контекстной диаграммы и имя рассматриваемого процесса, в данном случае: А-0 Выполнить курсовую работу;

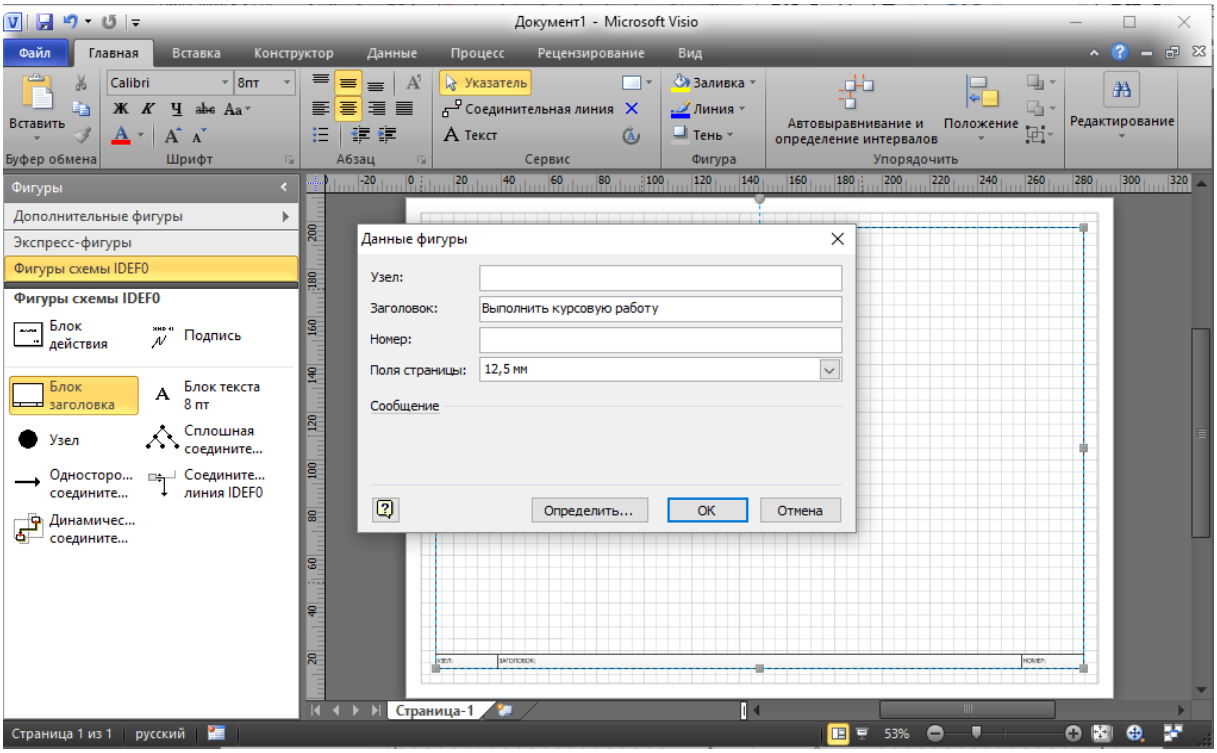


Рисунок 23 – Мастерская страница

4.4. Далее, имя заголовка фигуры «*Блок заголовка*» должно соответствовать номеру и названию задачи, декомпозиция которой будет изображена в данной области. Например: А1 Получить задание.

5. Определение цели и точки зрения.

5.1. С помощью кнопки *Блока текста* внесите текст в поле диаграммы – точку зрения и цель (рис. 5).

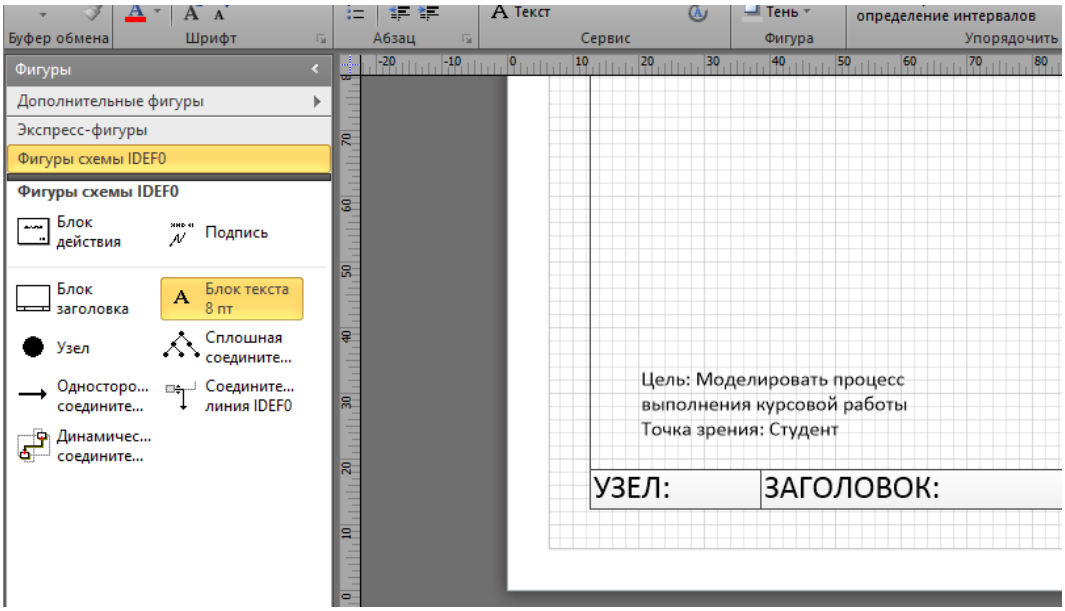


Рисунок 24 – Цель и точка зрения

6. В область диаграммы (поле *Блока заголовка*) внесите *Блок действия*. В открывшемся окне «Данные фигуры» внесите имя процесса и идентификатор процесса (рис. 6).

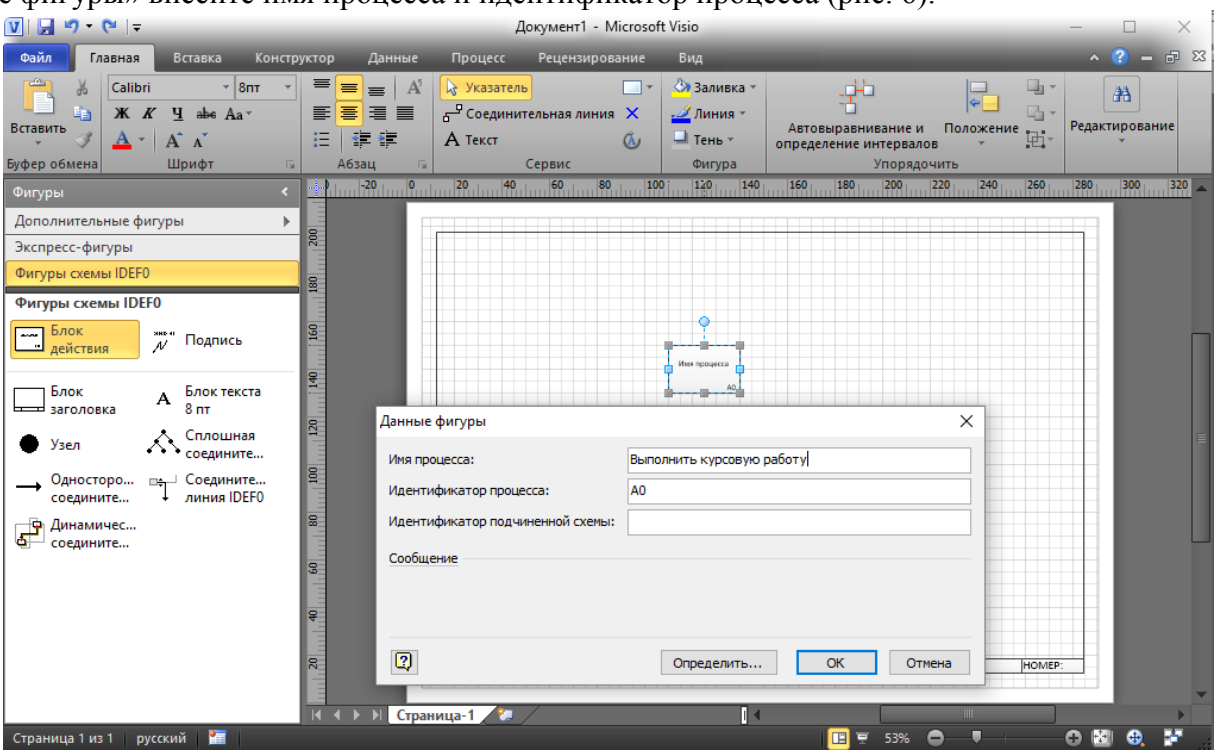


Рисунок 25 – Блок действия

7. С использованием блока *Односторонняя соединительная линия* создайте стрелки на контекстной диаграмме (табл. 1). Чтобы добавить текст необходимо дважды щелкнуть по стрелке.

Таблица 5 – Стрелки контекстной диаграммы

Имя стрелки (Arrow Name)	**Определение стрелки (Arrow Definition)**	**Тип стрелки (Arrow Type)**
График	График консультаций и сроки сдачи	Input
Список литературы	Источники информации для выполнения курсовой работы	Input
Варианты заданий	Список заданий на курсовую работу, подлежащий распределению между студентами	Input
Методические указания	Документ, содержащий указания по выполнению курсовой работы, описывающий содержание ее частей и основные требования	Control
Положение о курсовом проектировании	Документ, отражающий организационные требования по выполнению и сдаче курсовой работы	Control
Курсовая работа	Документ, являющийся основанием для получения оценки	Output
Оценка за курсовую работу	Результат выполнения курсовой работы	Output
Студент	Тот, кто выполняет курсовую работу	Mechanism

8. Результат выполнения предыдущих пунктов представлен на рис. 7.

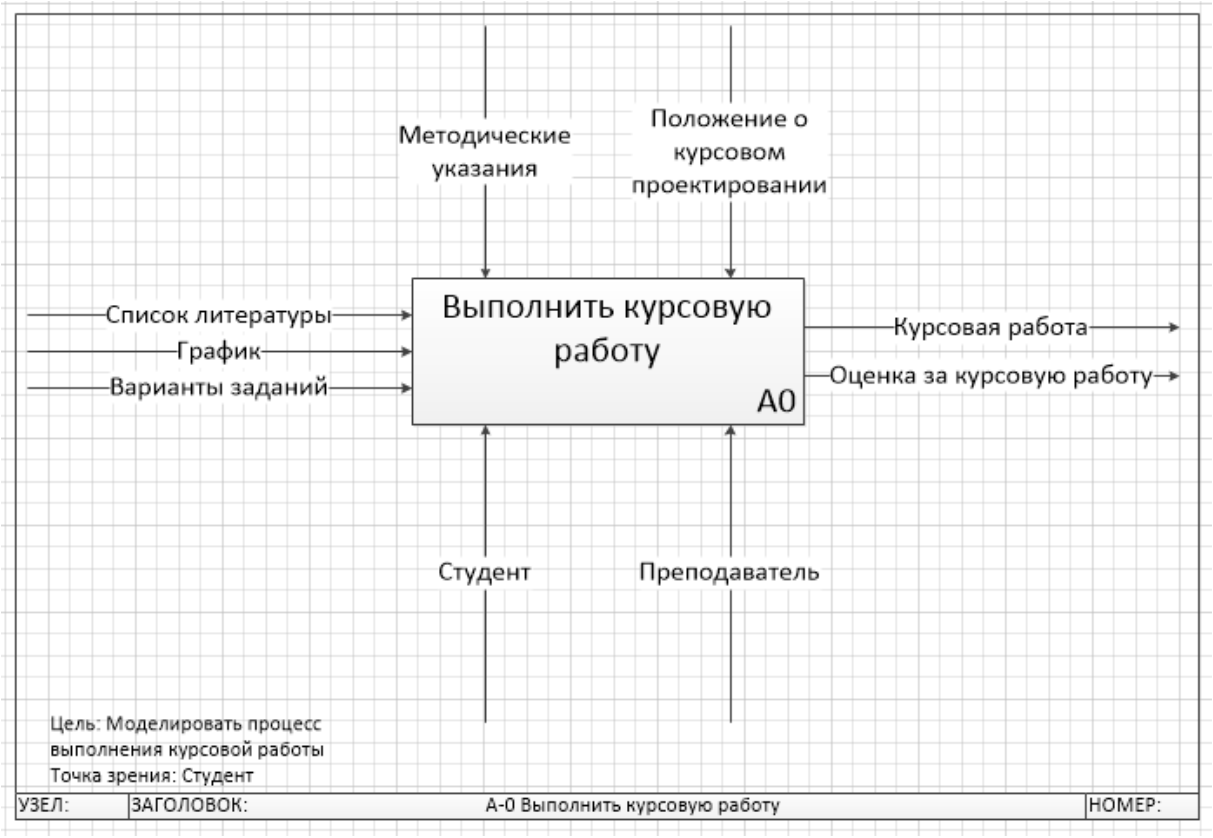


Рисунок 26 – Контекстная диаграмма

4.2. Создание диаграммы декомпозиции

1. Для построения декомпозиции диаграммы создайте новую страницу путем нажатия правой кнопкой мыши в нижнем левом углу окна на ярлык Страница 1. Выбрать пункт Добавить страницу (рис. 8)

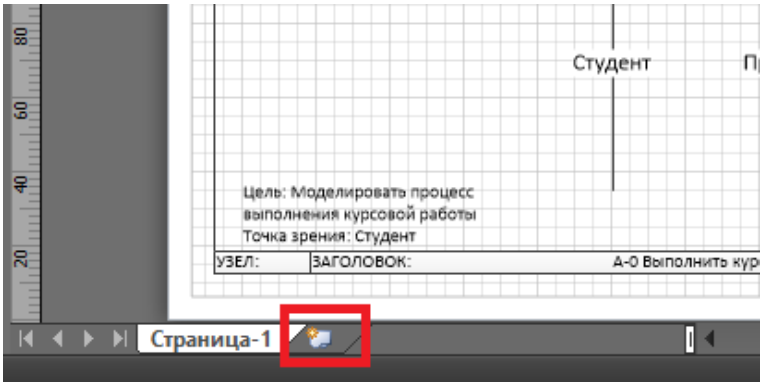


Рисунок 27 – Добавление страницы

2. Переименуйте страницы в соответствии с уровнем декомпозиции, например: A-0, A1 и т.д.
3. Распределите работы диаграммы декомпозиции в области Блока заголовка в соответствии с табл. 2.

Таблица 6 – Работы диаграммы декомпозиции A0

Имя работы (Activity Name)	**Определение (Definition)**
Получить задание	Выбрать задание из списка, согласовать его с преподавателем
Подобрать литературу	Выбрать из списка литературы подходящие источники
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой работы согласно заданию
Сделать графическую часть	При необходимости сделать графики и чертежи

Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Получить консультацию	Получить консультацию у преподавателя перед защитой, выявить неточности и недостатки
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя

4. Распределите стрелки для диаграммы декомпозиции в соответствии с контекстной диаграммой. Для этого «перенесите» входные и выходные стрелки, связанные с декомпозируемой работой, в поле декомпозиции.

Итог выполнения вышеописанных шагов представлен на рис. 9.

Разветвление стрелок. График (расписание) необходимо для того, чтобы прийти на консультацию и на защиту, т.е. необходимо подвести одноименную стрелку к 2 работам. Для разветвления стрелки необходимо от фрагмента стрелки до сегмента работы провести стрелку, состоящую из нескольких блоков Однонаправленное соединение.

Слияние стрелок. Для слияния двух стрелок выхода необходимо провести работы аналогичные разветвлению.

ICOM-метки. Используя блок текста, расставьте ICOM метки.

Результат выполнения предыдущих пунктов представлен на рисунке (рис. 9).

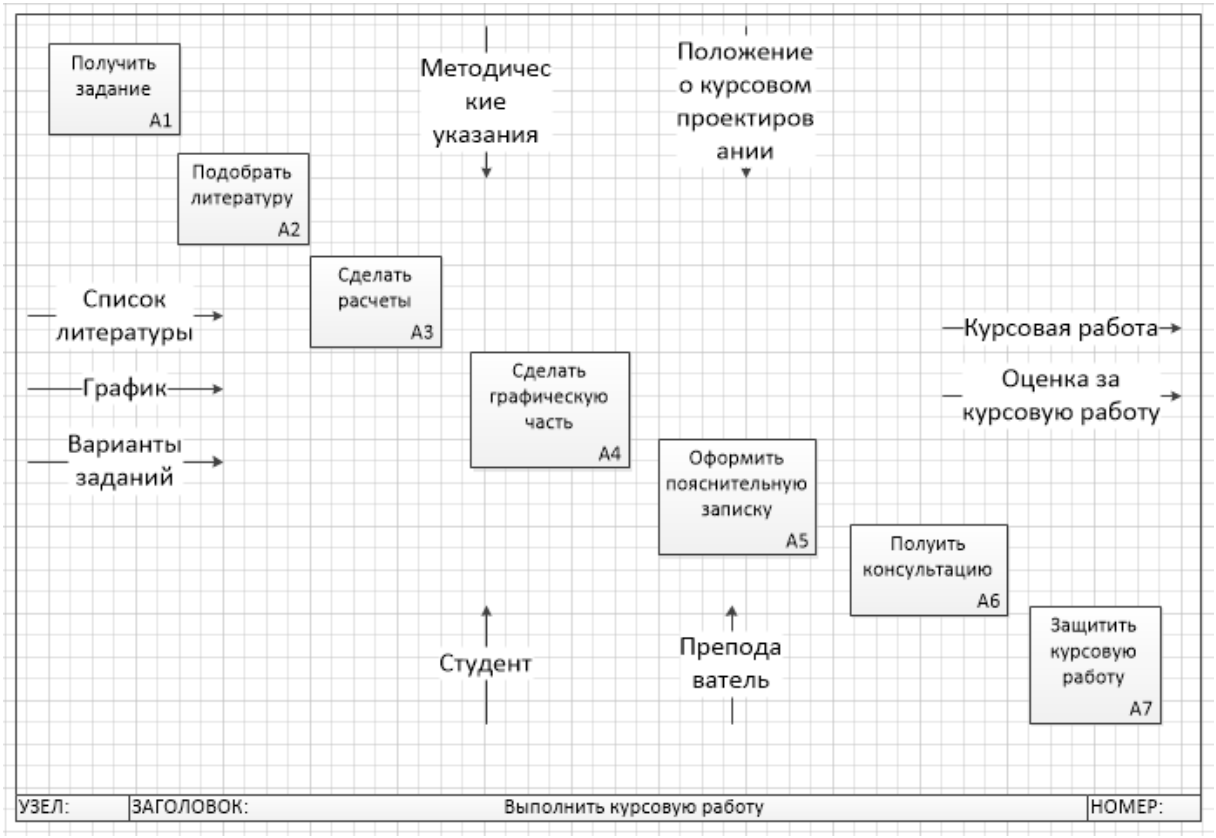


Рисунок 28 – Диаграмма декомпозиции

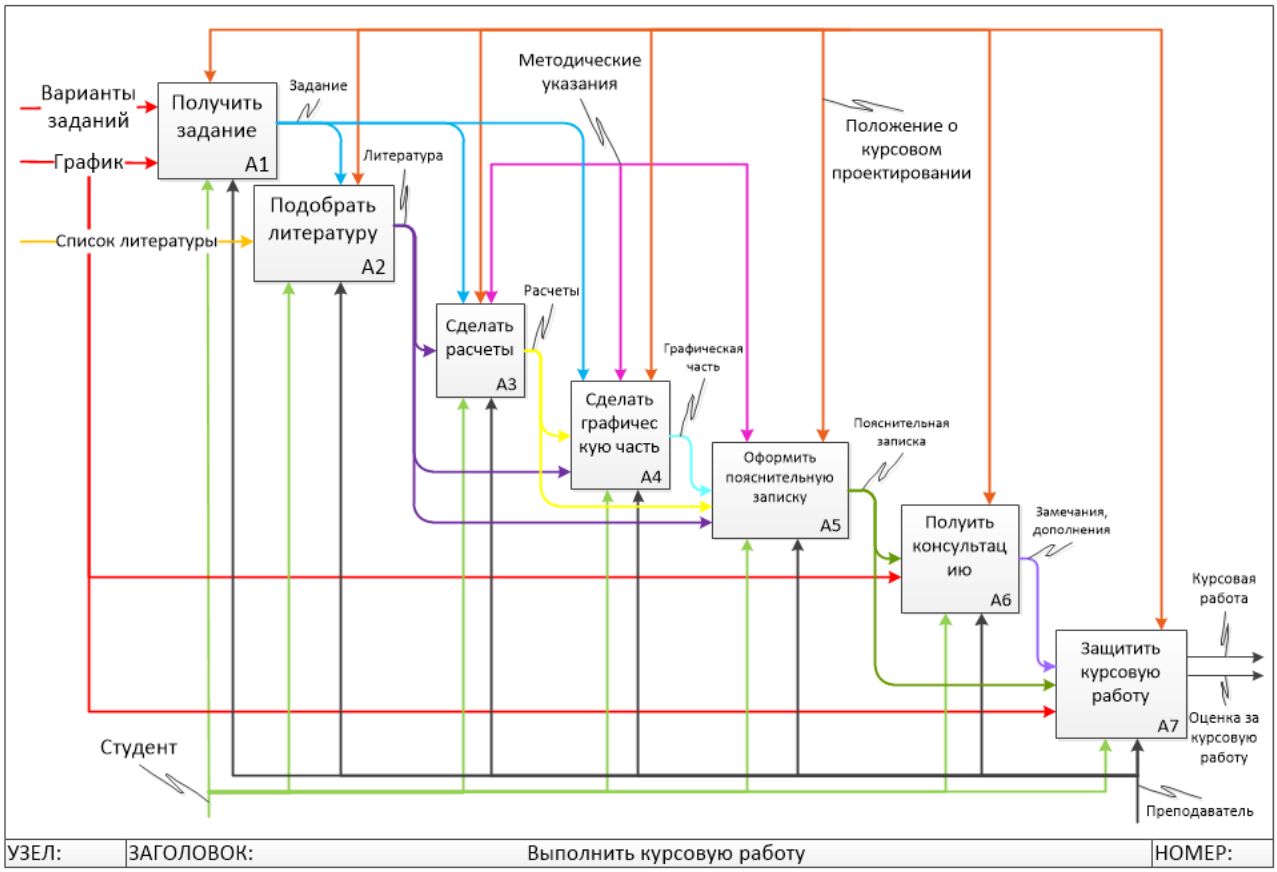


Рисунок 29 – Диаграмма декомпозиции блока A0

4.3. Создание дерева узлов

Дерево узлов – это диаграмма, отображающая иерархию работ процесса (рис. 11).

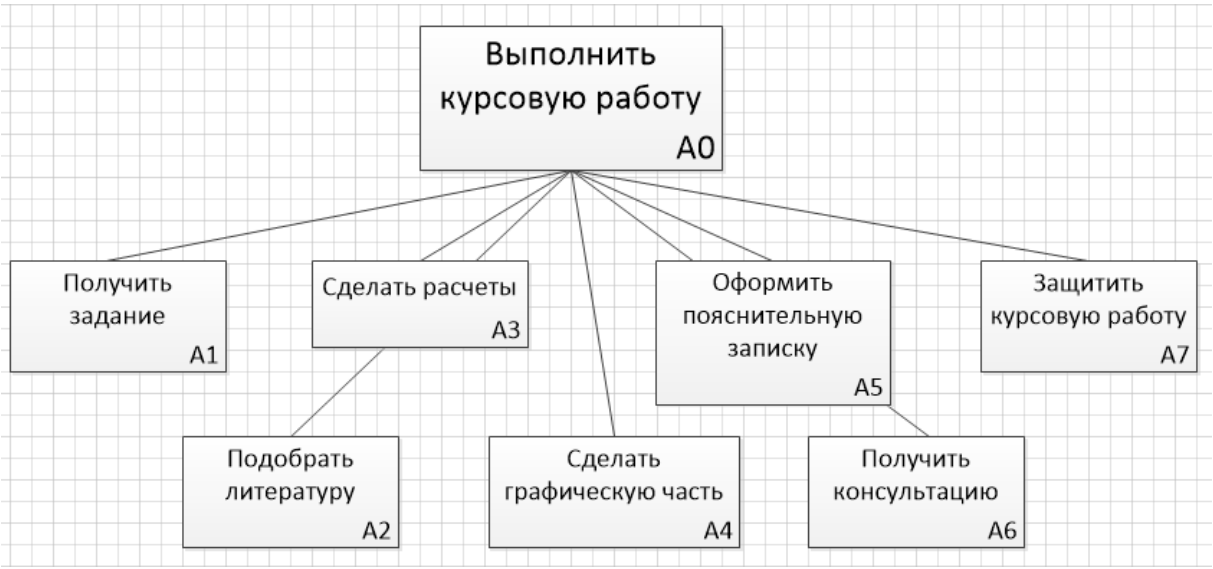


Рисунок 30 – Диаграмма узлов

Для построения диаграммы:

– создайте новую страницу;
– присвойте имя странице: дерево узлов;
– постройте дерево узлов, используя фигуры схемы IDEF0.

4.4. Создание глоссария

Глоссарий – это словарь ключевых слов, повествований, изложений, используемых при описании процесса (рис. 12, 13).

Для построения глоссария:

– создайте документ Microsoft Office Word;

– создайте 2 таблицы: описание работ процесса, описание интерфейсных дуг процесса;
– наименование столбцов таблиц: имя (работы/дуги, описание);
– заполните таблицы в соответствии с ранее разработанной моделью процесса.

Name	Definition
Выполнить курсовую работу	Текущие процессы выполнения курсовой работы
Защитить курсовую работу	Сдать готовую курсовую работу и ответить на вопросы преподавателя
Оформить пояснительную записку	Оформить текстовую часть и объединить все сделанные части в единое целое
Подобрать литературу	Выбрать из списка литературы подходящие
Получить задание	Выбрать задание из списка, согласовать его с
Получить консультацию	Получить консультацию у преподавателя перед
Сделать графическую часть	При необходимости сделать графики и чертежи
Сделать расчеты	Выполнить (если необходимо) расчетную часть курсовой

Рисунок 31 – Словарь работ

Name	Definition
Варианты заданий	Список заданий на курсовую работу, подлежа
График	График консультаций и сроки сдачи
Графическая часть	Выполненная графическая часть курсовой раб
Задание	Выдается на консультации преподавателем, чт
Замечания, дополнения	Замечания преподавателя, полученные на кон
Курсовая работа	Документ, являющийся основанием для получ
Литература	Выбранные источники, необходимые для выпо
Методические указания	Документ, содержащий указания по выполнен
Оценка за курсовую раб	Результат выполнения курсовой работы
Положение о курсовом п	Документ, отражающий организационные треб
Пояснительная записка	Теоретическая часть + расчеты + графическая
Преподаватель	Тот, кто оценивает курсовую работу
Расчеты	Выполненная расчетная часть курсовой работ
Список литературы	Источники информации для выполнения курсо
Студент	Тот, кто выполняет курсовую работу

Рисунок 32 – Словарь стрелок

5. Задание

На основе информационной модели предметной области, разработанной в практическом занятии №1, составить функциональную модель в нотации IDEF0.

Отчет по практическому занятию выполняется в формате MS Word, который содержит экранные формы моделей согласно заданию.

6. Варианты

11. Проектирование ИС «Отдел кадров»;
12. Проектирование ИС «Агентство аренды»;
13. Проектирование ИС «Аптека»;
14. Проектирование ИС «Ателье»;
15. Проектирование ИС «Аэропорт»;
16. Проектирование ИС «Библиотека»;
17. Проектирование ИС «Кинотеатр»;
18. Проектирование ИС «Поликлиника»;
19. Проектирование ИС «Автосалон»;
20. Проектирование ИС «Таксопарк».

7. Требования к построению модели

1. На контекстной диаграмме необходимо указать точку зрения и цель моделирования.
2. Количество блоков любой декомпозиции не менее 3-х и не более 9.
3. Количество декомпозиций – 3 уровня декомпозиции.

8. Контрольные вопросы

1. Каковы цели функционального моделирования?
2. Назовите основные компоненты функциональной модели.
3. Какие виды интерфейсных дуг различают в IDEF0?
4. Для чего нужна цель и точка зрения?
5. Что такое функциональный блок?
6. Какие виды диаграмм может содержать функциональная модель?

Практическая работа №4

Метод функционального моделирования IDEF0 с помощью Ramus Educational

1. Цель занятия

Целью занятия является изучение основ структурного подхода к проектированию ИС. Освоение принципов построения IDEF0-диаграммы классов в программной среде Ramus Educational.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами структурного подхода к проектированию ИС;

– изучить диаграмму IDEF0 (Integration Definition for Function Modeling) для предметной области «Выполнение курсовой работы»;

– построить с помощью программного средства Ramus Educational диаграмму IDEF0 согласно индивидуальному заданию (вариант получить у преподавателя).

3. Краткие теоретические сведения

3.1. Общие положения структурного метода

Сущность структурного подхода к разработке ИС заключается в декомпозиции (разбиении) системы на автоматизируемые функции, которые в свою очередь делятся на подфункции, на задачи и так далее. Процесс декомпозиции продолжается вплоть до определения конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны.

В основе структурного метода лежит несколько общих принципов:

– разбиение системы на множество независимых задач, доступных для понимания и решения;

– иерархическое упорядочивание, т.е. организация составных частей проблемы в древовидные структуры с добавлением новых деталей на каждом уровне.

К основным принципам относятся:

– абстрагирование, т.е. выделение существенных аспектов системы и отвлечение от несущественных;

– формализация, т.е. общее методологическое решение проблемы;

– непротиворечивость, состоящая в обосновании и согласовании элементов системы;

– иерархическая структуризация данных.

3.2. Метод функционального моделирования SADT

На основе метода SADT, предложенного Д. Россом, разработана методология IDEF0 (Icam DEFinition), которая является основной частью программы ICAM (Интеграция компьютерных и промышленных технологий), проводимой по инициативе ВВС США. Методология IDEF0 является наиболее признанным эффективным средством анализа, конструирования и отображения бизнес-процессов, применяемым также и широко за пределами США.

Метод SADT применяется при моделировании широкого круга систем, для которых определяются требования и функции, после чего проводится их реализация.

Методология SADT представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели предметной области, которая отображает функциональную структуру, производимые функции и действия, а также связи между ними.

Результат применения метода SADT – модель, которая состоит из диаграмм, фрагментов текстов и глоссария со ссылками друг на друга. Все функции и интерфейсы представляются диаграммами в виде, соответственно, блоков и дуг. Место соединения дуги с блоком определяет тип интерфейса. Управляющая информация входит в блок сверху, в то время как информация, которая подвергается обработке (исходные данные), указывается с левой стороны блока, а результаты работы функции (выход, результат) – с правой стороны. Механизм, осуществляющий операцию (человек или автоматизированная система), задается дугой, входящей в блок снизу (рис. 1).

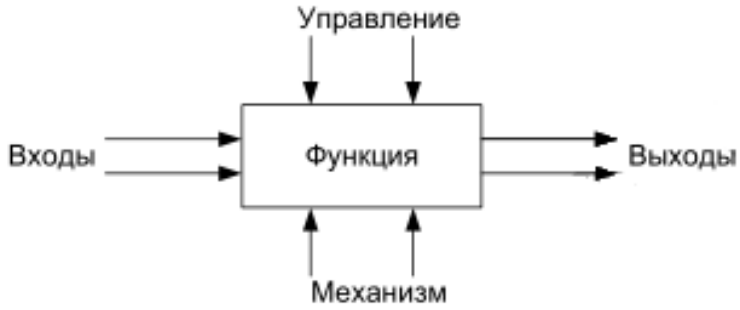


Рисунок 33 – Структура модели

Описание системы с помощью SADT называется **моделью**. **Субъектом** моделирования служит сама система. Однако моделируемая система никогда не существует изолированно: она всегда связана с окружающей средой. По этой причине в методологии SADT подчеркивается необходимость точного определения границ системы, т.е. модель устанавливает точно, что является и что не является субъектом моделирования, описывая то, что входит в систему, и, подразумевая то, что лежит за ее пределами. SADT-модель должна иметь единственный субъект.

С определением модели тесно связана позиция (называемая **точкой зрения**), с которой наблюдается система и создается ее модель. «Точку зрения» лучше всего представлять себе как место (роль, должность) человека или объекта в рассматриваемой системе, на которое надо «встать», чтобы увидеть систему в действии и необходимой полноте. У конкретной модели может быть только одна точка зрения.

Обычно вопросы для SADT-модели формулируются на самом раннем этапе проектирования, при этом основная суть этих вопросов должна быть выражена в одной-двух фразах, которые становятся целью модели.

После того как определены субъект, цель и точка зрения модели, начинается первая интеграция процесса моделирования по методологии SADT. Субъект определяет, что включить в модель, а что исключить из нее. Точка зрения диктует автору модели выбор нужной информации о субъекте и форму ее представления. Цель становится критерием окончания моделирования. Конечным результатом этого процесса является набор тщательно взаимоувязанных описаний, начиная с описания самого верхнего уровня системы и заканчивая подробным описанием ее деталей или отдельных операций.

Каждое из таких тщательно взаимосогласованных описаний называется диаграммой и имеет определенный уровень детализации. SADT-модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели менее детализированы, чем диаграммы нижних уровней. Другими словами, модель SADT можно представить в виде древовидной структуры диаграмм, где верхняя диаграмма является наиболее общей, а самые нижние – максимально детализированы (рис. 2).

Каждый блок на диаграмме имеет свой номер. Блок любой диаграммы может быть детализирован диаграммой нижнего уровня, которая, в свою очередь, также может детализироваться с помощью необходимого числа диаграмм. Таким образом, формируется иерархия диаграмм. Для того чтобы указать положение любой диаграммы или блока в иерархии, им присваивают уникальные обозначения. Например, **A41** (A сокр. от Activity) является диаграммой, которая детализирует блок 1 на диаграмме A4. Аналогично, A4 детализирует блок 4 на диаграмме A0, которая является самой верхней (родительской) диаграммой модели.

Некоторые дуги имеют начало в одном из блоков диаграммы и завершение в другом, у других же начало может исходить от границ диаграммы – дуги управления, механизма, дуги входа и выхода, перенесенные с родительской (верхнего уровня) диаграммы. Таким образом, источник или получатель этих пограничных дуг может быть обнаружен только на родительской диаграмме.

Также следует сказать о так называемых «туннельных дугах». Туннельные дуги означают, что данные, выраженные этими дугами не рассматриваются на следующем уровне детализации (как бы проходят «насквозь»). Если «туннель» расположен в месте соединения дуги с блоком « », то данные этой дуги не обязательны на следующем уровне детализации. Если же «туннель» находится на противоположном конце дуги « » – это значит, что данные дуги не описываются на родительской диаграмме. Граничные дуги должны продолжаться (дублироваться) на родительской диаграмме, делая ее полной и непротиворечивой (рис. 3).

Для упрощения понимания приведенных диаграмм, следует расшифровать применяемую в IDEF систему обозначений, позволяющую аналитику точно идентифицировать и проверять по дугам связи между диаграммами. Эта схема кодирования дуг – «**ICOM**» – получила название по первым буквам английских эквивалентов слов вход (**Input**), управление (**Control**), выход (**Output**), механизм (**Mechanism**).

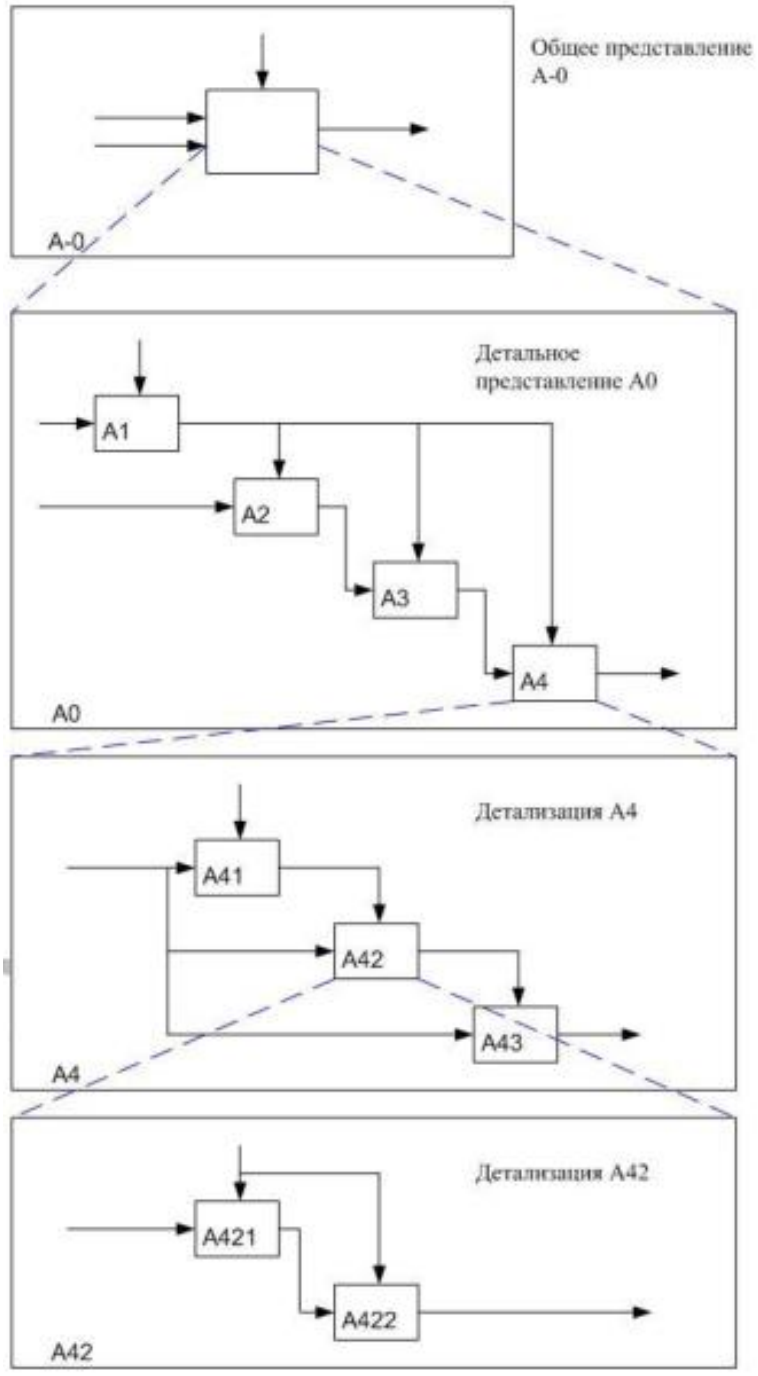


Рисунок 34 – Структура SADT-модели. Иерархия и декомпозиция диаграмм

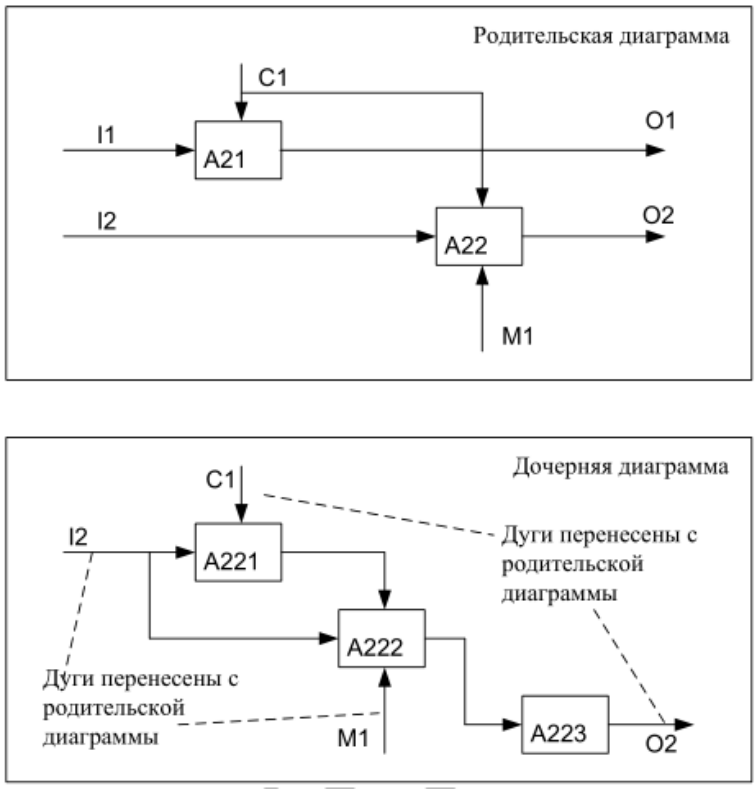


Рисунок 35 – Соответствие дуг родительской и дочерних диаграмм

4. Методика выполнения лабораторной работы

В качестве примера рассматривается процесс выполнения студентом курсовой работы (курсового проекта).

Программное обеспечение «Ramus» предназначено для использования в проектах, в которых необходимо описание бизнес-процессов предприятия. «Ramus» поддерживает методологии моделирования бизнес-процессов IDEF0 и DFD, а также имеет ряд дополнительных возможностей, призванных удовлетворить потребности команд разработчиков систем управления предприятиями.

«Ramus» обладает гибкими возможностями построения отчетности по графическим моделям, позволяющие создавать отчеты в форме документов, регламентирующих деятельность предприятия.

Ramus Educational имеет достаточно интуитивный интерфейс пользователя, позволяющий быстро и просто создавать сложные модели.

4.1 Начало работы

1. Запустите программу Ramus Educational. В появившемся окне (рис. 4) предлагается создать новый проект или открыть уже существующий.

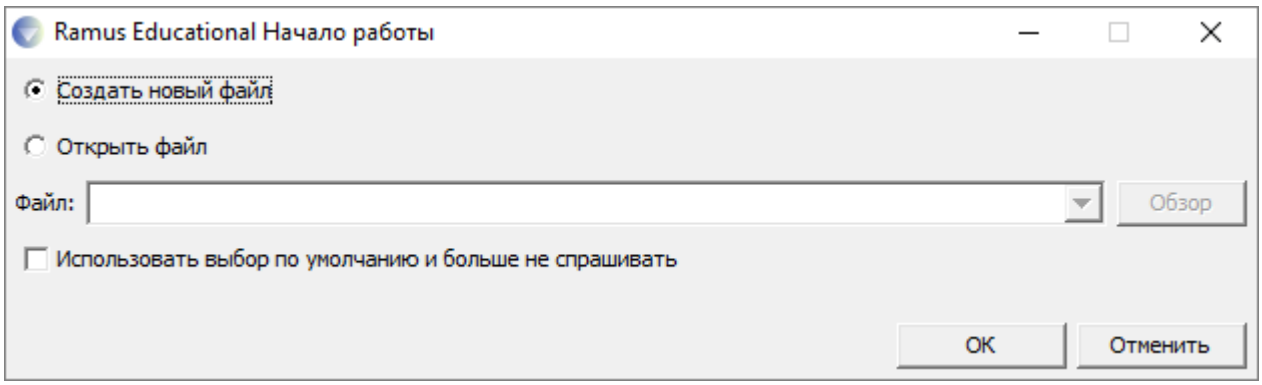


Рисунок 36 – Окно запуска Ramus Educational

2. После нажатия на кнопку «ОК» осуществляется запуск мастера проекта.

– На первом шаге (рис. 5) в соответствующие поля необходимо внести сведения об авторе, названии проекта и модели, а также выбрать тип нотации модели (IDEF0 или DFD).

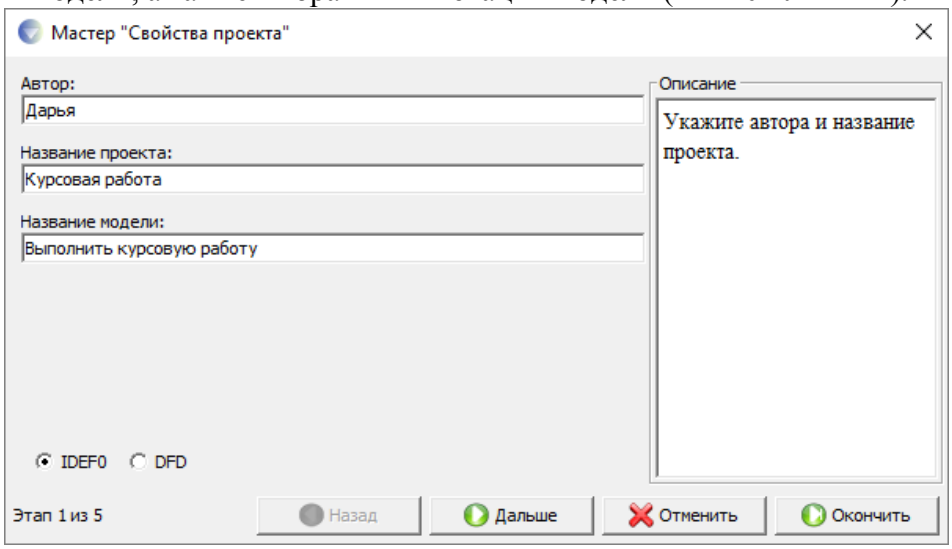


Рисунок 37 – Окно мастера создания проекта. Шаг 1

– На втором шаге вводится название организации, использующей данный проект.

– На третьем – дается краткое описание будущего проекта.

– Четвертый шаг позволяет создать несколько основных классификаторов (в данном случае можно пропустить этот шаг). Так как модели процессов реальных предприятий могут содержать значительное количество объектов (документы, персонал, функции и т.д.), то в Ramus предусмотрена возможность упорядочено хранить информацию об этих объектах в виде системы классификаторов. Классификация объектов упрощает поиск и обработку информации об объектах модели, а также и об объектах непосредственно не представленных на диаграммах процессов, но относящихся к процессам предприятия.

– На пятом, заключительном, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (пропустить данный шаг).

При необходимости можно завершить работу мастера, нажав кнопку «Окончить».

После завершения работы мастера, откроется рабочее пространство «Диаграммы», в котором можно приступить к рисованию графической модели (рис. 6). В верхней части приводятся сведения о проекте, введенные пользователем посредством мастера диаграмм.

Программа Ramus Educational обладает гибким графическим интерфейсом, который можно настроить под нужды и предпочтения конкретного пользователя: ненужные окна можно закрыть/свернуть; можно менять их размеры и месторасположение; также можно группировать два и более окон в одном, при этом содержимое вложенных окон будет размещено на вкладках общего окна (данный функционал возможен не для всех комбинаций окон).

3. Сохраните созданную модель, выбрав опцию меню «Файл» – «Сохранить как».

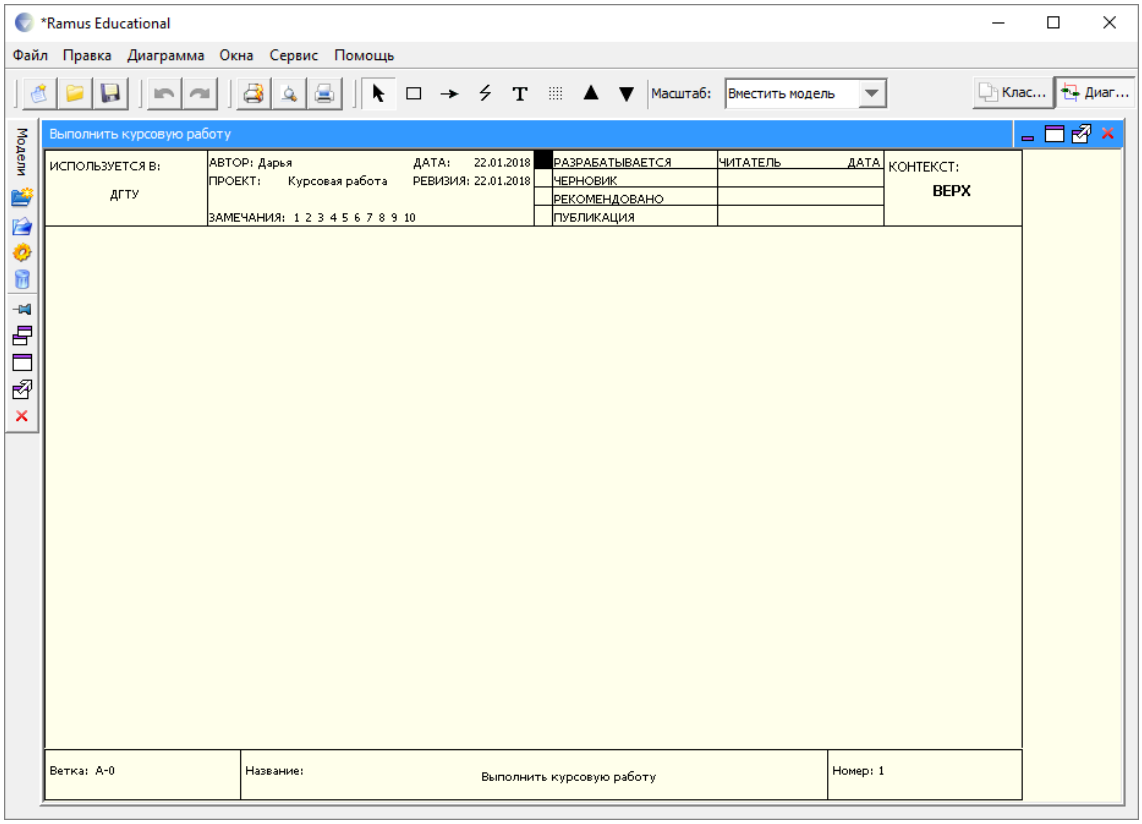


Рисунок 38 – Рабочее пространство

4.2 Создание контекстной диаграммы

1. На панели инструментов выберите пиктограмму функции ( ) и мышью укажите месторасположение на рабочем пространстве.

1. Дайте данному функциональному блоку имя «Выполнить курсовую работу». Для этого дважды щелкните внутри блока.

2. Используя пиктограмму панели инструментов , создайте стрелки на контекстной диаграмме согласно Таблица 1.

Таблица 7 – Контекстная диаграмма

Имя стрелки (Arrow Name)	**Определение стрелки (Arrow Definition)**	**Тип стрелки (Arrow Type)**
График	График консультаций и сроки сдачи	Input
Список литературы	Источники информации для выполнения курсовой работы	Input
Варианты заданий	Список заданий на курсовую работу, подлежащий распределению между студентами	Input
Методические указания	Документ, содержащий указания по выполнению курсовой работы, описывающий содержание ее частей и основные требования	Control
Положение о курсовом проектировании	Документ, отражающий организационные требования по выполнению и сдаче курсовой работы	Control
Курсовая работа	Документ, являющийся основанием для получения оценки	Output
Оценка за курсовую работу	Результат выполнения курсовой работы	Output
Студент	Тот, кто выполняет курсовую работу	Mechanism

3. В результате должна получиться контекстная диаграмма, показанная на рис. 7.

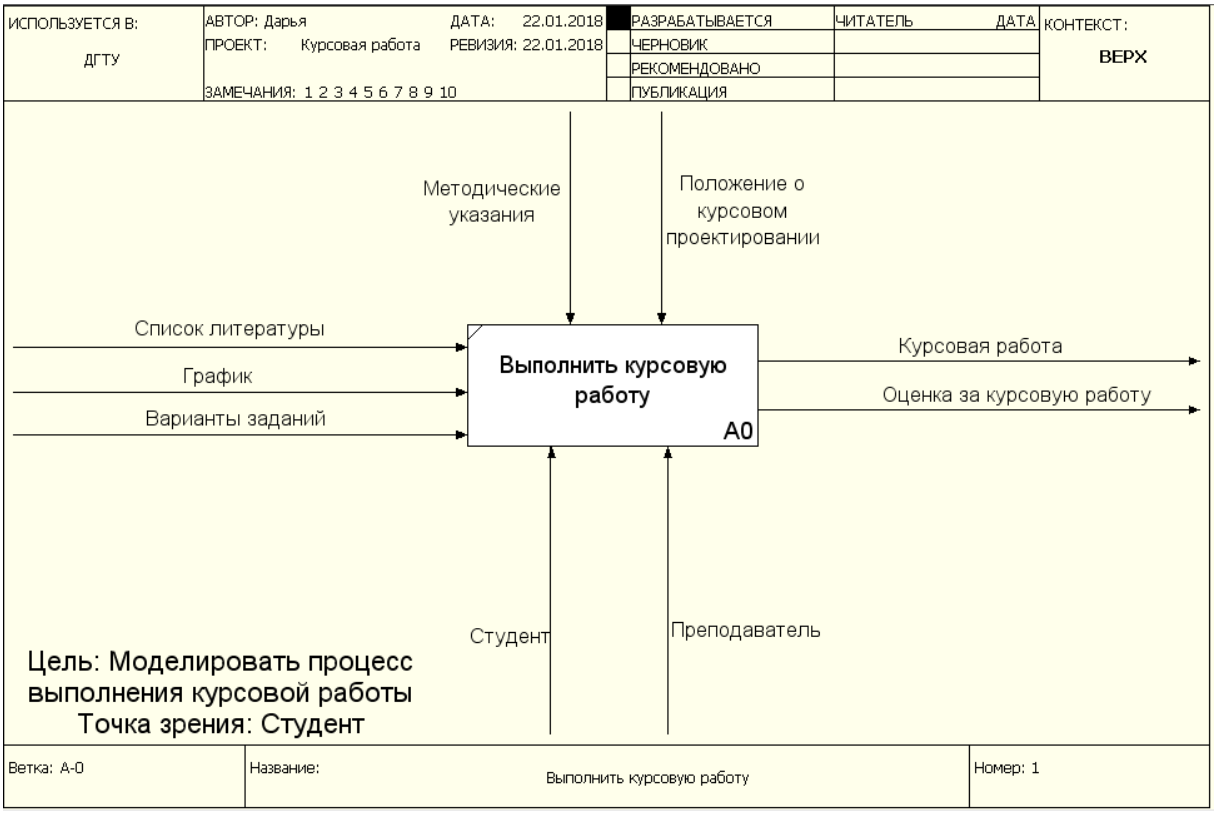


Рисунок 39 – Контекстная диаграмма предметной области «Выполнение курсовой работы»

4.3 Создание диаграммы декомпозиции

1. Выберите в палитре инструментов кнопку перехода на нижний уровень  , в диалоговом окне «Создание новой диаграммы» (рис. 8) установите количество функциональных блоков 7, укажите тип диаграммы (IDEF0) и нажмите кнопку ОК.

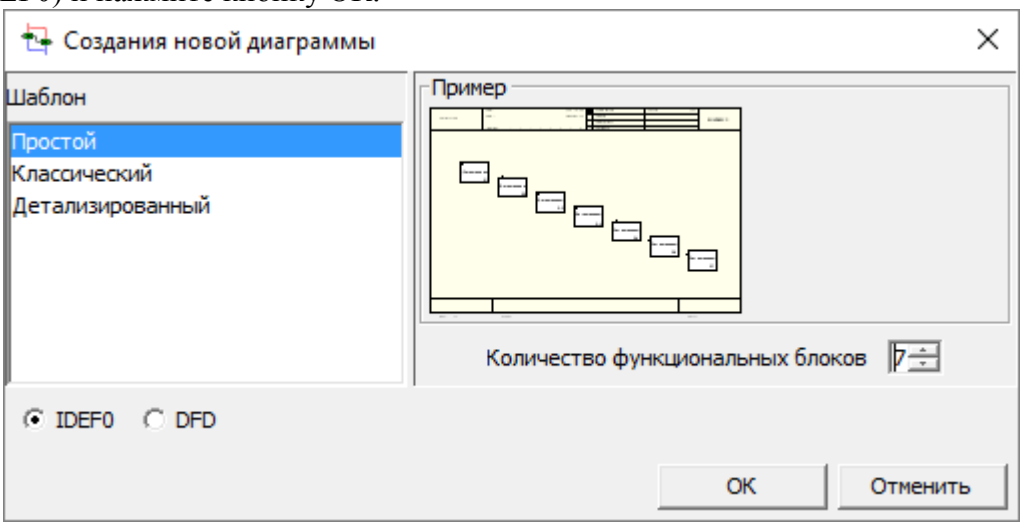


Рисунок 40 – Диалоговое окно создания детализирующей диаграммы

2. Автоматически будет создана диаграмма первого уровня декомпозиции (рис. 9) с перенесенными в нее потоками родительской диаграммы.

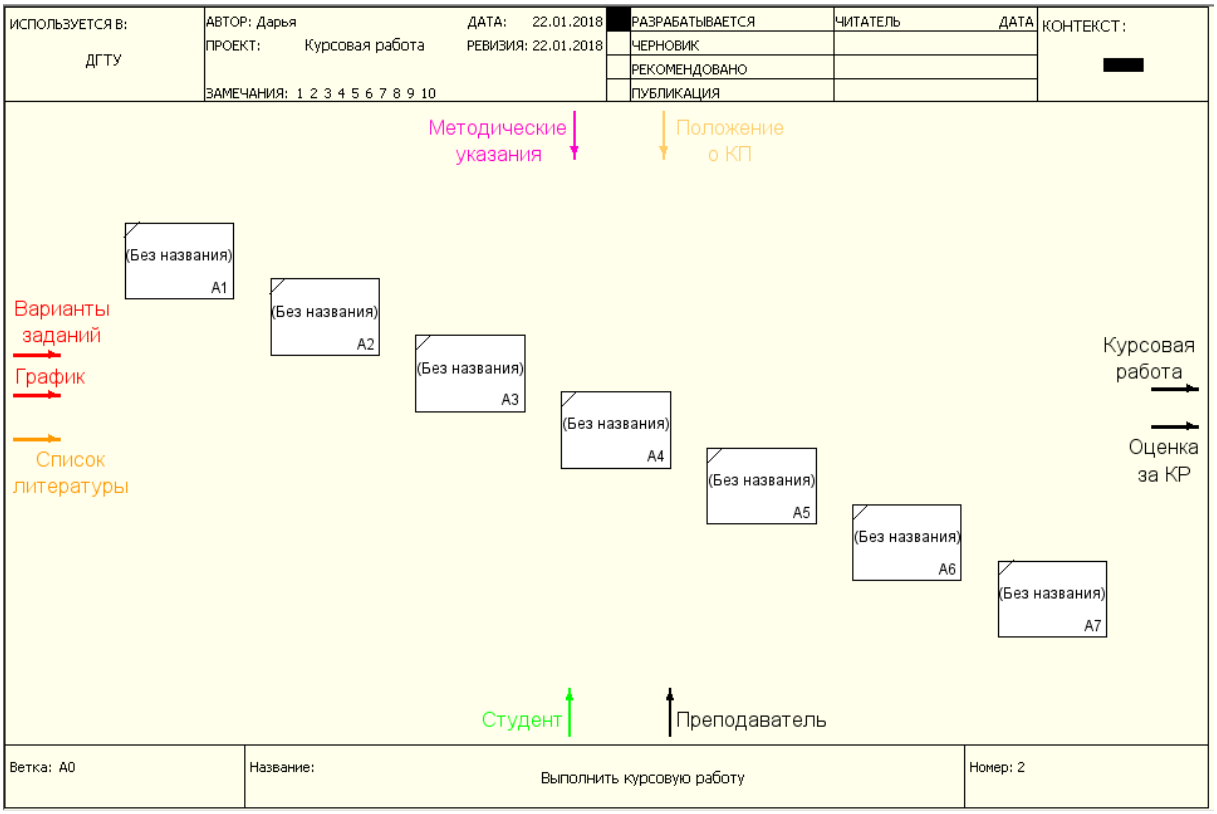


Рисунок 41 – Рабочее пространство детализирующей диаграммы

3. Выделите первую работу (функциональный блок), затем двойным щелчком мыши или, выбрав в контекстном меню пункт «Редактировать активный элемент», откройте окно свойств и внесите имя работы. Повторите операцию для оставшихся работ.

4. Выделив необходимый поток (стрелку) и, удерживая левую клавишу мыши, соедините его требуемым образом (через вход, управление, механизм или выход) с соответствующим функциональным блоком. В результате должна получиться детализирующая диаграмма, представленная на рис. 10.

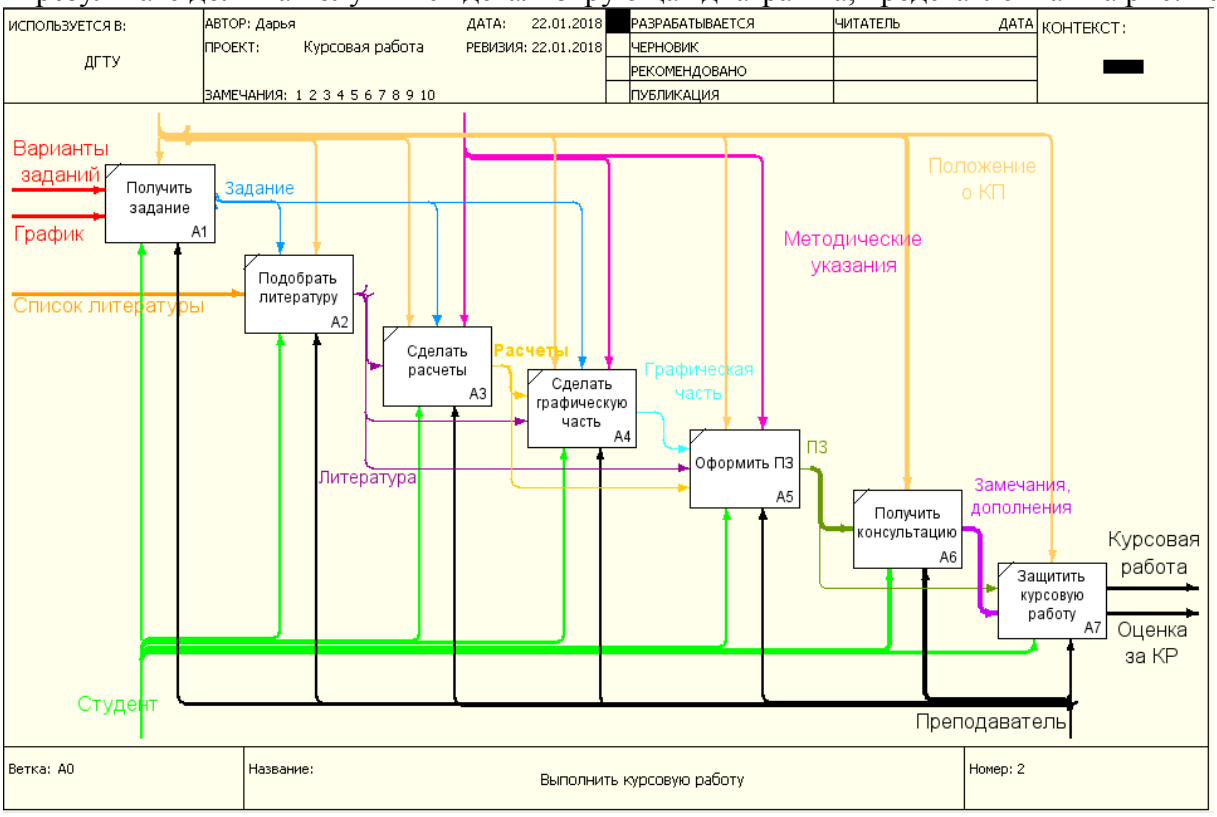


Рисунок 42 – Детализированная диаграмма первого уровня

5. Задание

На основе информационной модели предметной области, разработанной в практическом занятии №1, составить функциональную модель в нотации IDEF0.

6. Требования к построению модели

4. На контекстной диаграмме необходимо указать точку зрения и цель моделирования.
5. Количество блоков любой декомпозиции не менее 3-х и не более 9.
6. Количество декомпозиций – 3 уровня декомпозиции.

7. Контрольные вопросы

7. Каковы цели функционального моделирования?
8. Назовите основные компоненты функциональной модели.
9. Какие виды интерфейсных дуг различают в IDEF0?
10. Для чего нужна цель и точка зрения?
11. Что такое функциональный блок?
12. Какие виды диаграмм может содержать функциональная модель?

Практическая работа №5

Построение организационных диаграмм с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания организационных диаграмм в ручном и автоматическом режимах.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения организационных диаграмм с помощью MS Visio;

– ;

– .

3. Краткие теоретические сведения

3.1. Общие сведения об организационных диаграммах

Организационная диаграмма – это схема иерархии отчетности, которая используется для отображения отношений между сотрудниками, должностями и группами.

Организационные диаграммы могут быть как простыми схемами, так и большими и сложными на основе сведений из внешнего источника данных. Фигуры организационной диаграммы могут отображать основные сведения, например, имя и должность, или подробные сведения, например, отдел и учетный отдел. К фигурам организационной диаграммы можно даже добавлять рисунки.

Организационная диаграмма – это схематическое представление об иерархии внутри компании, а также распределении полномочий и ответственности между сотрудниками и отделами.

Как показывает опыт, применение руководителями, менеджерами данных схем значительно повышает эффективность управления предприятием.

Использование организационных диаграмм позволяет решить сразу несколько задач:

1. проанализировать состояние дел в компании на текущий момент,
2. вовремя обнаружить какую-то проблему в организации и системе управления,
3. избежать появления новых проблем и ошибок.

В зависимости от поставленных целей организационные диаграммы могут быть простыми или развернутыми.

Простые диаграммы содержат краткую информацию о каждом из сотрудников (имя, должность и место в системе организационной иерархии), а также раскрывают численный состав персонала.

Развернутые диаграммы несут дополнительные сведения о функциях и объектах управления сотрудников компании.

Организационные диаграммы позволяют визуально представить характер взаимоотношений внутри компании, благодаря чему удается своевременно обнаружить и разрешить возникающие в организации проблемы.

Основные **преимущества организационных диаграмм:**

1. Они позволяют выявить главных игроков в организации и проанализировать их отношения с остальным персоналом.
2. Сотрудники компании получают ясное представление о том, кто возьмет на себя ответственность при разрешении тех или иных организационных проблем в соответствии с корпоративными стандартами.
3. Повышают осведомленность сотрудников о функциях и профессиональном статусе каждого субъекта, работающего в компании, тем самым способствуя совершенствованию процесса коммуникации внутри фирмы.

Следует отметить, что организационные диаграммы имеют ряд ограничений. Например, отражая лишь структуру и формальный характер взаимодействия персонала, они не затрагивают личные взаимоотношения людей в коллективе. Кроме того, с помощью этих схем нельзя определить стиль менеджмента, выбранный руководством предприятия.

Организационные диаграммы описывают связи при помощи **трех типов графических элементов:**

1. ***Линия:*** указывает на прямые отношения между руководителями и подчиненными. Для описания взаимосвязей между различными иерархическими уровнями организации линии рисуют слева или справа от диаграммы.

2. ***Ступенька:*** служит для иллюстрации отношений между помощниками менеджера, а также связей между сферами деятельности, в которых помощники могут давать советы руководителю, но при этом их мнение не является авторитетным.

3. ***Функционал:*** показывает связи между должностями специалистов и сферами деятельности, в которых мнение специалистов является авторитетным для руководителя.

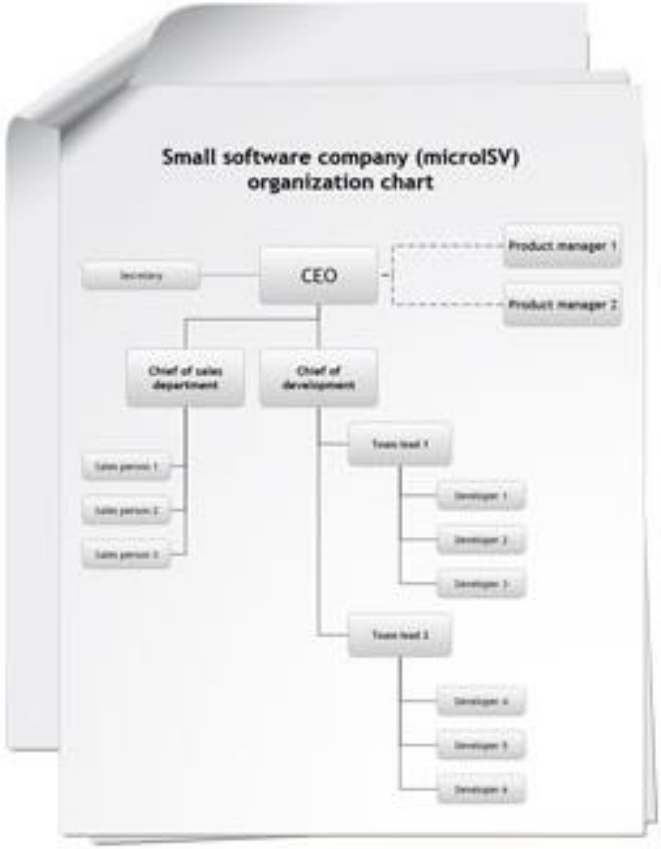


Рисунок 43 – Пример организационной диаграммы

Типы организационных структур

1. ***линейная модель:*** каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности;
2. ***функциональная модель:*** «одно подразделение = одна функция»;
3. ***линейно-функциональная модель:*** ступенчатая иерархическая;
4. ***процессная модель:*** «одно подразделение = один процесс»;
5. ***матричная модель:*** «один процесс или один проект = группа сотрудников из разных функциональных подразделений»;
6. ***множественная (смешанная).***

В ***линейной структуре*** управления каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности. *Достоинство* – простота, экономичность, предельное единоначалие. Основной *недостаток* – высокие требования к квалификации руководителей. Сейчас практически не используется.

Функциональная организационная структура – связь административного управления с осуществлением функционального управления.

Линейно-функциональная структура – линейные руководители являются единоначальниками, а им оказывают помощь функциональные органы. Линейные руководители низших ступеней административно не подчинены функциональным руководителям высших ступеней управления. Она применялась наиболее широко.

Преимущества:

– четкая система взаимных связей внутри функций и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью функций, составляющих деятельность;

– ясно выраженная ответственность;

– быстрая реакция исполнительных функциональных подразделений на прямые указания вышестоящих.

Недостатки:

– в работе руководителей практически всех уровней оперативные проблемы («текучка») доминируют над стратегическими;

– слабые горизонтальные связи между функциональными подразделениями порождают волокиту и перекладывание ответственности при решении проблем, требующих участия нескольких подразделений;

– малая гибкость и приспособляемость к изменению ситуации;

– критерии эффективности и качества работы подразделений и организации в целом разные и часто взаимоисключающие;

– большое число «этажей» или уровней управления между работниками, выпускающими продукцию, и лицом, принимающим решение;

– перегрузка управленцев верхнего уровня;

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств высших управленцев.

Процессная модель. Истоки концепции управления процессами ведут к теориям управления, разработанным еще в XIX веке. В 80-х годах XIX-го века Фредерик Тейлор предложил менеджерам использовать методы процессного управления для наилучшей организации деятельности. В начале 1900-х годов Анри Файоль разработал концепцию реинжиниринга – осуществление деятельности в соответствии с поставленными задачами путем получения оптимального преимущества из всех доступных ресурсов.

Преимущества процессных структур:

– четкая система взаимных связей внутри процессов и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью операций и действий, направленных на достижение поставленной цели и получение заданного результата;

– наделение сотрудников большими полномочиями и увеличение роли каждого из них в работе компании приводит к значительному повышению их отдачи;

– быстрая реакция исполнительных процессных подразделений на изменение внешних условий;

– в работе руководителей стратегические проблемы доминируют над оперативными;

– критерии эффективности и качества работы подразделений и организации в целом согласованы и сонаправлены.

Недостатки процессной структуры:

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств рядовых работников и исполнителей.

– управление смешанными в функциональном смысле рабочими командами – более сложная задача, нежели управление функциональными подразделениями;

– наличие в команде нескольких человек различной функциональной квалификации неизбежно приводит к некоторым задержкам и ошибкам, возникающим при передаче работы между членами команды. Однако потери здесь значительно меньше, чем при традиционной организации работ, когда исполнители подчиняются различным подразделениям компании.

Матричная модель. Матричные структуры совмещают принципы построения функциональных и процессных систем. В этих структурах существуют жестко регламентированные процессы, находящиеся под управлением менеджера процесса. При этом деятельность осуществляется работниками, находящимися в оперативном подчинении менеджера процесса и в административном подчинении руководителя в функциональном «колодце».

Преимущества

– Комплексный подход к реализации проекта, решению проблемы;

– Концентрация усилий на решении одной задачи, на выполнении одного конкретного проекта;

– Большая гибкость структуры;

– Активизация деятельности руководителей проектов и исполнителей в результате формирования проектных групп;

– Усиление личной ответственности конкретного руководителя как за проект в целом, так и за его элементы.

Недостатки

– При наличии нескольких организационных проектов или программ проектные структуры приводят к дроблению ресурсов и заметно усложняют поддержание и развитие производственного и научно-технического потенциала компании как единого целого;

– От руководителя проекта требуется не только управление всеми стадиями жизненного цикла проекта, но и учет места проекта в сети проектов данной компании;

– Формирование проектных групп, не являющихся устойчивыми образованиями, лишает работников осознания своего места в компании;

– При использовании проектной структуры возникают трудности с перспективным использованием специалистов в данной компании;

– Наблюдается частичное дублирование функций.

Смешанные структуры. Если применять различные модели организации деятельности в пределах отдельных бизнес-процессов, то можно использовать преимущества той или иной организационной модели. При этом для организации в целом будет применяться процессная организация основных структурных блоков, а в рамках отдельных блоков могут применяться различные модели.

4. Методика выполнения лабораторной работы

В качестве примера рассматривается процесс создания организационной диаграммы ВУЗа двумя способами:

1. вручную;
2. с помощью мастера.

4.1 Создание организационной диаграммы вручную

Для построения организационной диаграммы вручную необходимо проделать следующие действия:

1. Запустить *MS Visio*.
2. В разделе *Выберите шаблон* выбрать категорию *Бизнес*, а затем *Организационная диаграмма* и нажать кнопку *Создать*.

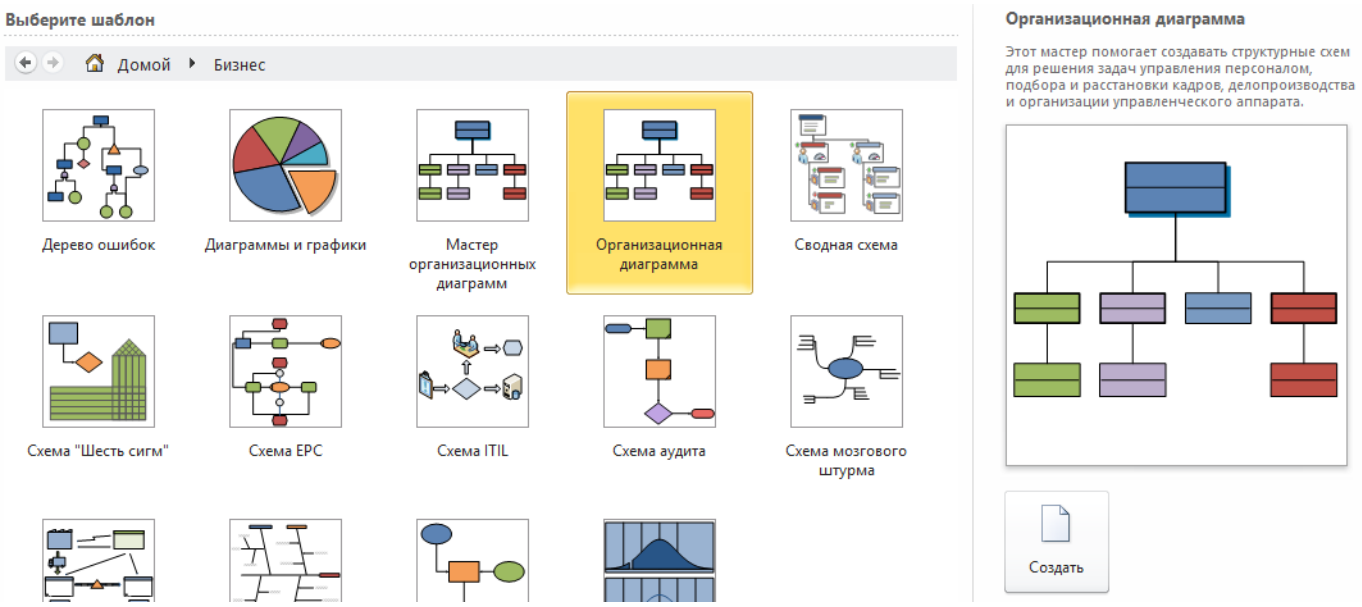


Рисунок 44 – Выбор шаблона

В разделе фигур можно увидеть основные элементы, необходимые для построения организационной диаграммы (рис. 3).

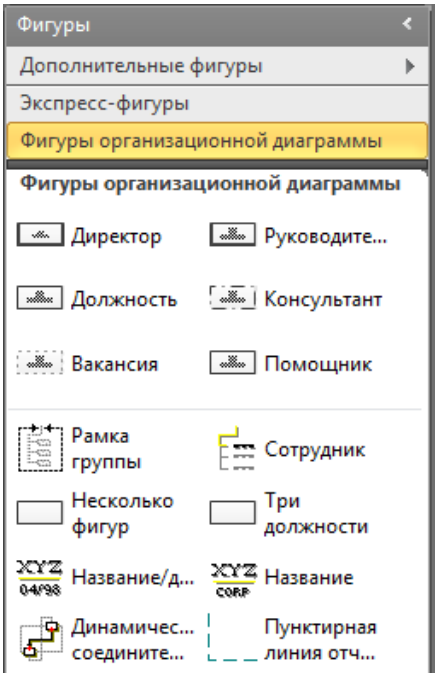


Рисунок 45 – Элементы организационной диаграммы в MS Visio

3. Перетащите фигуру *Директор* из набора фигур *Фигуры организационной диаграммы* в центр вверху страницы документа.

4. Надстройка организационных диаграмм отображает анимированное диалоговое окно (рис. 4), показывающее, как нужно добавлять в схему дополнительные фигуры.

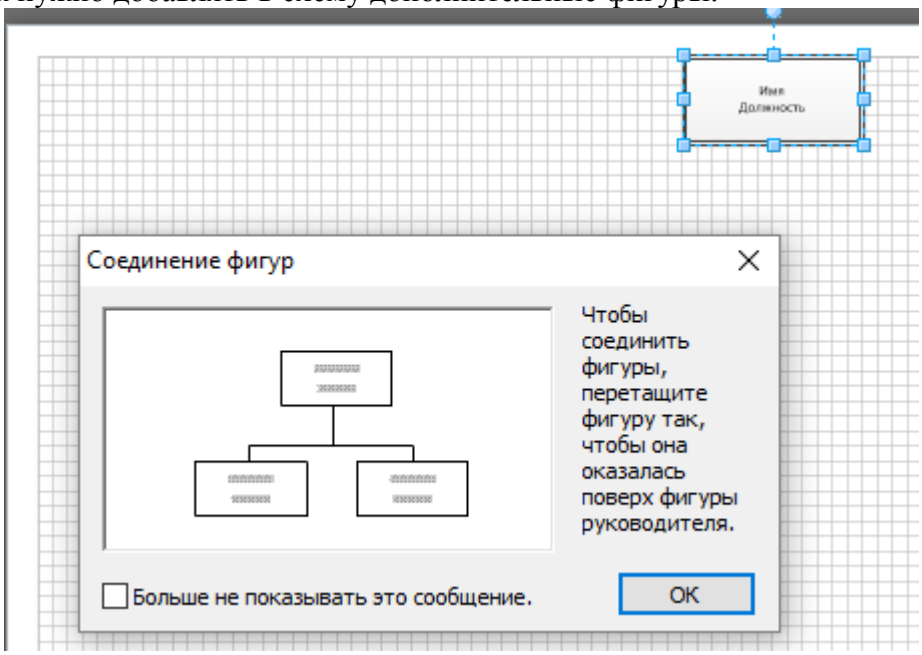


Рисунок 46 – Надстройка организационных диаграмм

5. Не снимая выделения с фигуры, при необходимости введите *ФИО*, затем нажмите клавишу ENTER и во второй строке введите *Должность*. (рис. 5).

При создании организационной диаграммы ВУЗа можно ограничиться только указанием должностей.

Рисунок 47 – Фигура Директор

6. Добавьте еще две фигуры *Директор* и расположите по обе стороны от фигуры *Ректор*. Соедините их используя *Соединительную линию.* В результате должна получиться схема, показанная на рисунке 6.

Рисунок 48 – Схема, полученная при выполнении шага 6

7. Перетащите фигуру *Руководитель* на фигуру *Директор*. Затем, при необходимости, введите ФИО, нажмите клавишу ENTER и введите *Должность*.

Надстройка автоматически размещает новую фигуру под фигурой *Директор*.

8. Повторите предыдущий шаг и обратите внимание, что надстройка разместила вторую фигуру руководителя сбоку от первой. Разместите все необходимые фигуры *Руководитель* в соответствии с рисунком 7.

Рисунок 49 – Фигуры Руководитель

9. Используя фигуры *Руководитель* и *Несколько фигур* добавьте необходимые *Должности,* относящиеся к руководителю **Первый проректор** (рис. 8)**.**

При использовании элемента *Несколько фигур* необходимо выполнить следующие действия:

– перетащить элемент Несколько фигур на необходимую фигуру;
– затем, в появившемся окне задать необходимые параметры, например, как на рисунке 9;
– нажать ОК.

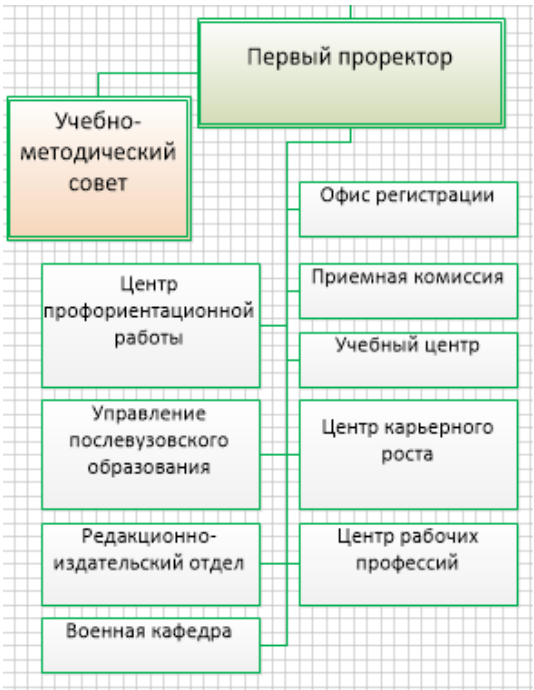


Рисунок 50 – Часть диаграммы «Первый проректор»

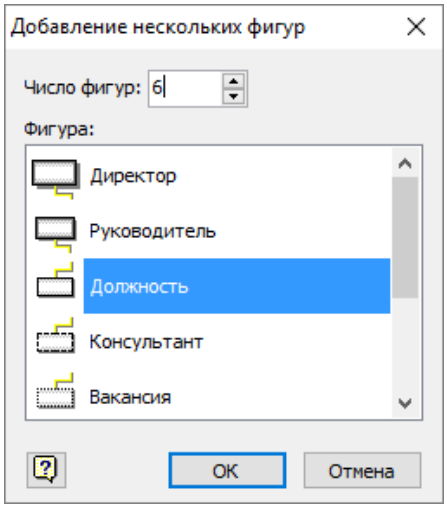


Рисунок 51 – Добавление нескольких фигур

10. Повторите предыдущий шаг и создайте полную организационную диаграмму (рис. 10-13).

Рисунок 52 – Часть диаграммы «Проректор по стратегическому развитию»

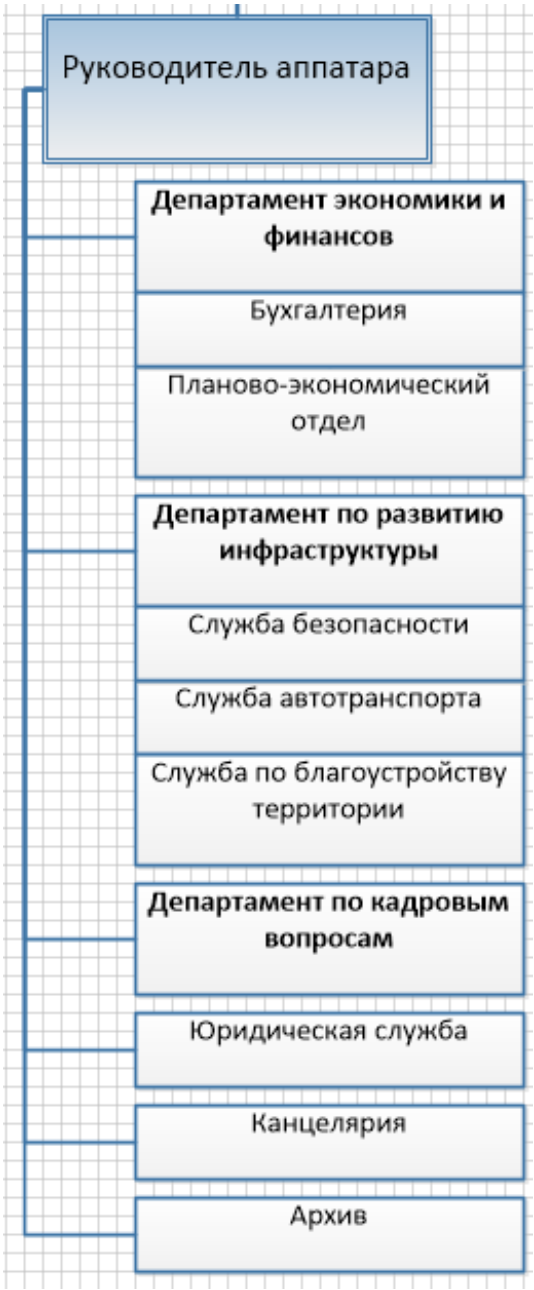


Рисунок 53 – Часть диаграммы «Руководитель аппарата»

Рисунок 54 – Часть диаграммы «Проректор по научной работе»

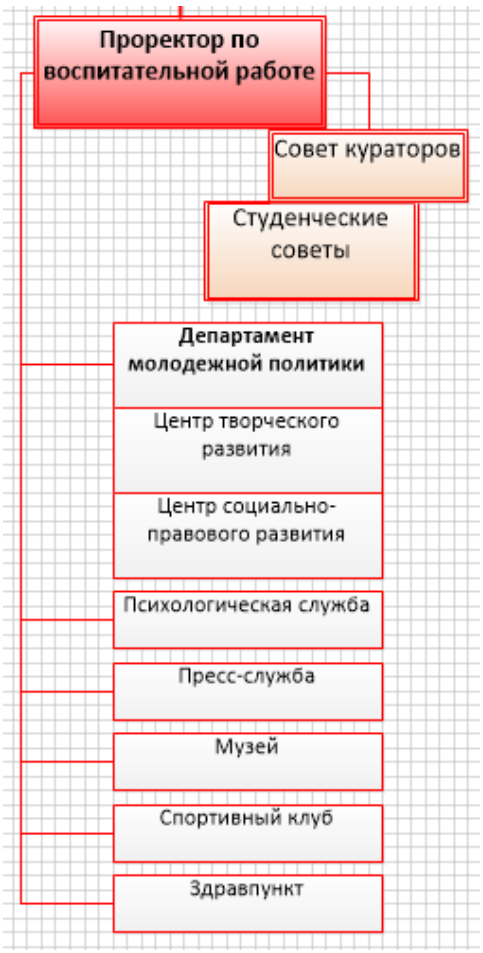


Рисунок 55 – Часть диаграммы «Проректор по ВР»

11. В результате выполнения всех описанных шагов организационная диаграмма ВУЗа должна принять вид, представленный на рисунке 14

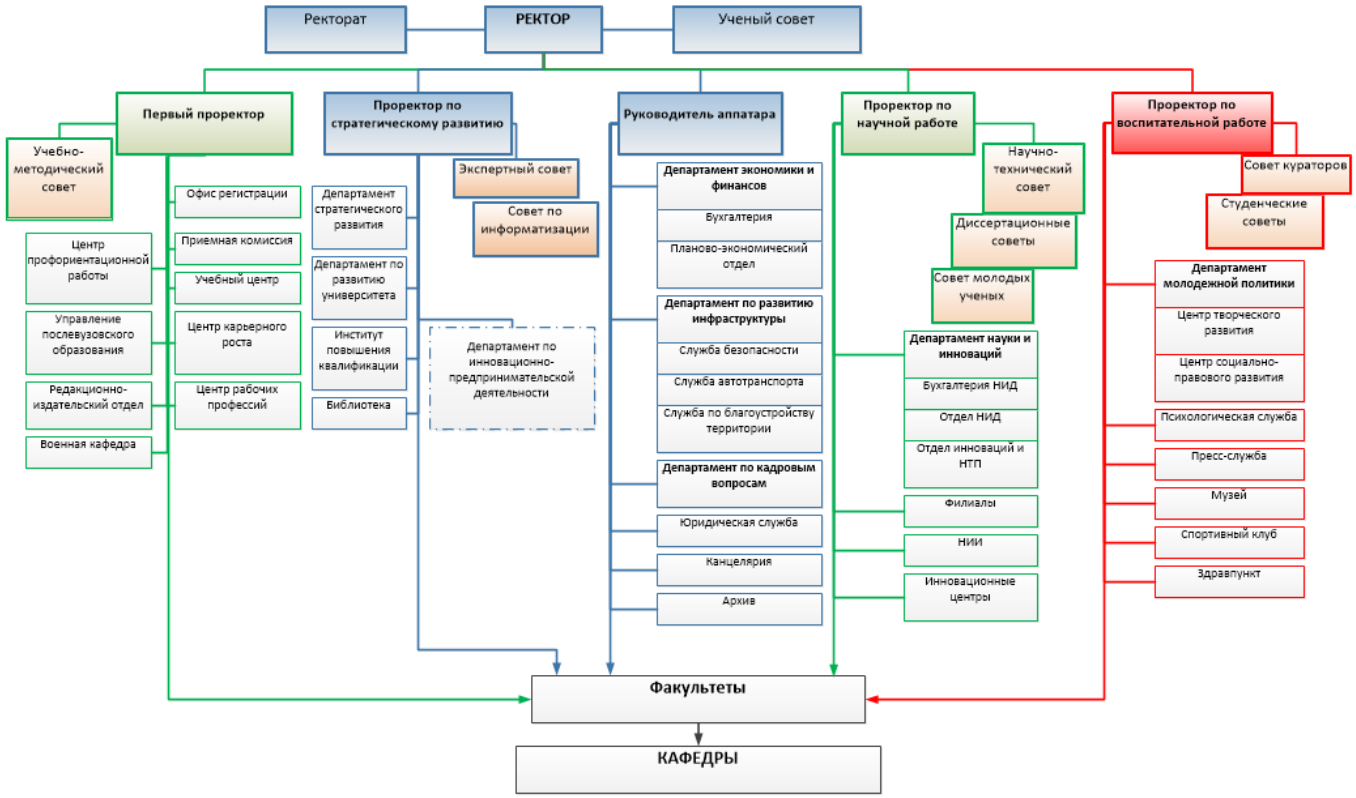


Рисунок 56 – Организационная диаграмма ВУЗа

4.2 Создание организационной диаграммы с помощью мастера диаграмм на основе данных

Построение организационной диаграммы с помощью мастера рассмотрим также на примере ВУЗа, однако немного упростим структуру.

Для построения организационной диаграммы с помощью мастера диаграмм необходимо проделать следующие действия:

1. Запустить *MS Visio*.
2. В разделе *Выберите шаблон* выбрать категорию *Бизнес*, а затем *Мастер организационных диаграмм* (рис. 15) и нажать кнопку *Создать*.

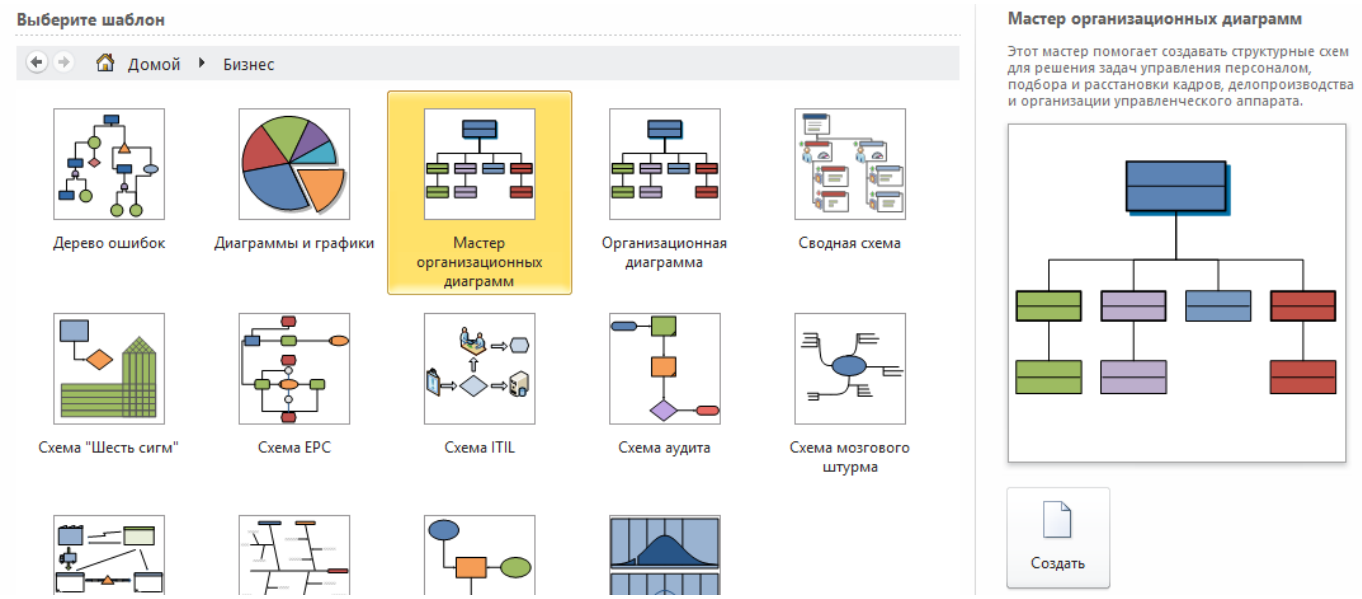


Рисунок 57 – Выбор шаблона «Мастер организационных диаграмм»

3. На первой странице мастера организационных диаграмм выберем переключатель *По данным, введенным с помощью мастера* (рис. 16). Обратите внимание – в описании этого переключателя подтверждается, что будет создан новый источник данных.

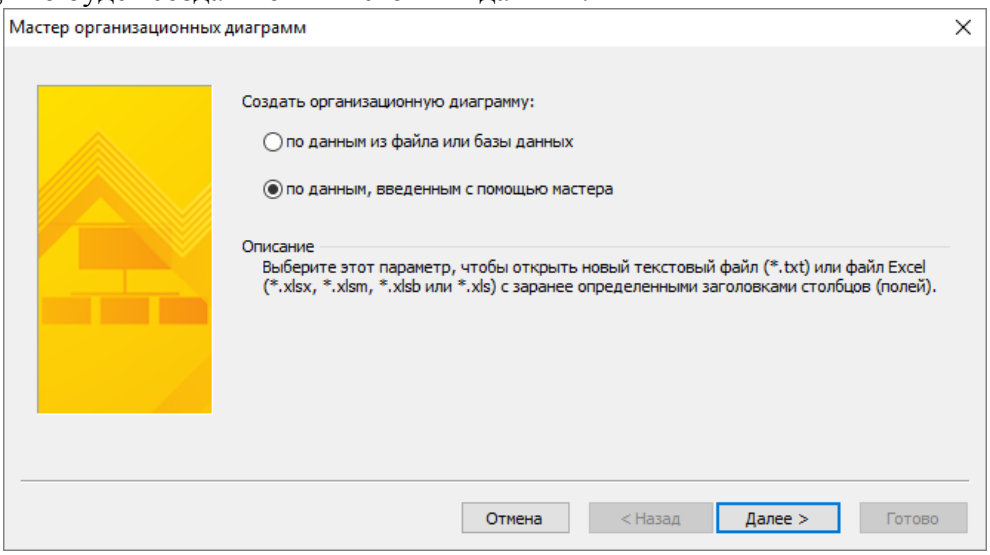


Рисунок 58 – Мастер организационных диаграмм. Шаг 1

4. Далее необходимо нажать кнопку *Далее* и выбрать тип файла.

На странице выбора типа файла выберем переключатель *Excel* и щелкнем на кнопке *Обзор*. В открывшемся диалоговом окне выберем папку, в которую нужно сохранить файл, в поле *Имя файла* введем *Данные оргдиаграммы* и щелкнем на кнопке Сохранить. Имя файла отображается в поле *Имя нового файла*. (рис. 17).

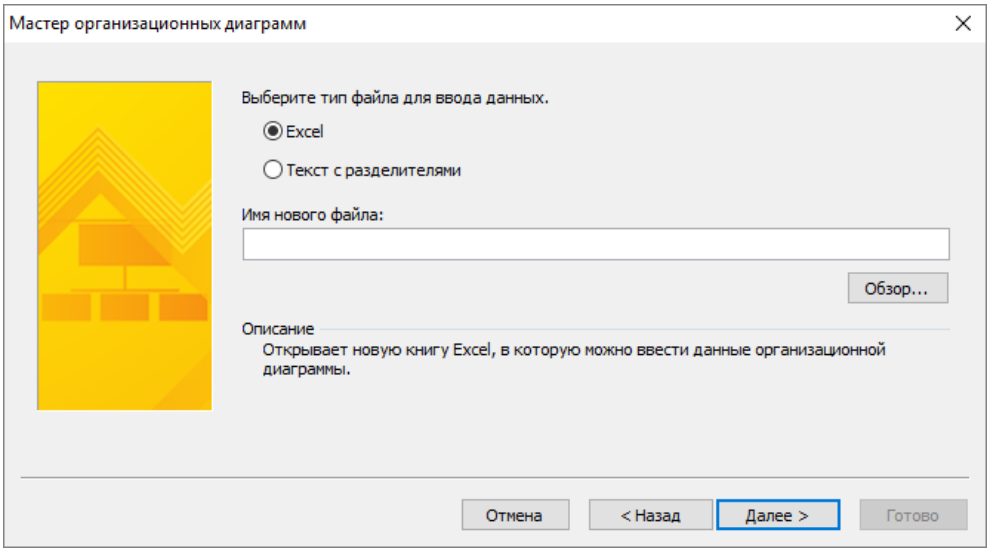


Рисунок 59 – Мастер организационных диаграмм. Шаг 2

5. Щелкнем на кнопке *Далее*. MS Visio показывает на необходимость ввести свои данные поверх ранее введенных (рис. 18).

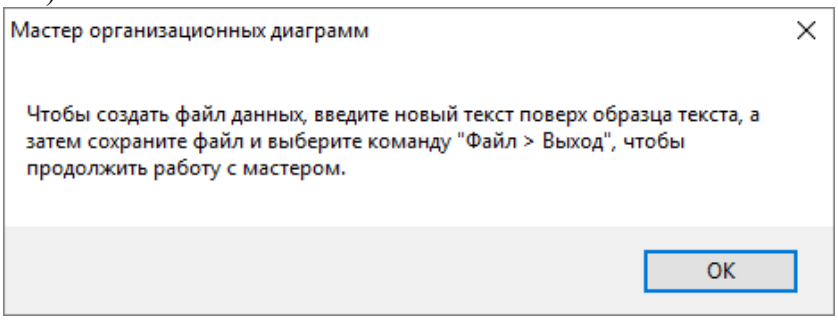


Рисунок 60 – Мастер организационных диаграмм. Шаг 3

6. Щелкнем на кнопке *Ок*. В Excel отображается форматированная книга. ***На рисунке 19 показана книга Excel для ввода данных диаграммы, при этом заголовок каждого столбца включает примечание с инструкциями по вводу данных в этом столбце.***

	A	B	C	D	E
1	Имя	Руководитель	Введите фамилию лица, которому подчиняется данный сотрудник, как оно показано в поле СТЬ	ОТДЕЛ	ТЕЛЕФОН
2					
3	Сергей Белов		Ге	Директор	x5555
4	Ольга Петрова	Сергей Белов	Руководитель разработки	Разработка продукта	x6666
5	Игорь Сергеев	Ольга Петрова	Разработчик программного обеспечения	Разработка продукта	x6667
6					

Рисунок 61 – Форматированная книга Excel по умолчанию

7. Далее вносим свои данные в соответствии с рисунком 20.

	A	B	C	D	E
1	**Имя**	**Руководитель**	**ДОЛЖНОСТЬ**	**ОТДЕЛ**	**ТЕЛЕФОН**
2					
3	Иванов Иван		Ректор	Ректорат	
4	Петров Петр	Иванов Иван	Ректорат	Ректорат	
5	Смирнов Андрей	Иванов Иван	Ученый совет	Ректорат	
6	Сидоров Михаил	Иванов Иван	Первый проректор	Ректорат	
7	Кожемяко Галина	Иванов Иван	Проректор по стратегическому развитию	Ректорат	
8	Куликова Наталья	Иванов Иван	Руководитель аппарата	Ректорат	
9	Лымарь Александра	Иванов Иван	Проректор по научной работе	Ректорат	
10	Голубенко Артем	Иванов Иван	Проректор по воспитательной работе	Ректорат	
11	Евидченко Денис	Сидоров Михаил	Начальник приемной комиссии	Приемная комиссия	
12	Бершатская Светлана	Сидоров Михаил	Начальник военной кафедры	Военная кафедра	
13	Маньшина Кристина	Кожемяко Галина	Начальник департамента развития	Департамент стратегического развития	
14	Шмелева Оксана	Кожемяко Галина	Библиотекарь	Библиотека	
15	Игнатенко Денис	Кожемяко Галина	Начальник департамента экономики	Департамент экономики и финансов	
16	Ненашев Виктор	Куликова Наталья	Начальник службы безопасности	Департамент по развитию инфраструктуры	
17	Ткачев Анатолий	Куликова Наталья	Начальник отдела кадров	Департамент по кадровым вопросам	
18	Сидоренко Александр	Куликова Наталья	Архивариус	Архив	
19	Акинин Павел	Лымарь Александра	Начальник департамента науки	Департамент науки и инноваций	
20	Смирнова Оксана	Голубенко Артем	Начальник департамента молодежной политики	Департамент молодежной политики	
21	Ермаков Дмитрий	Голубенко Артем	Начальник психологической службы	Психологическая служба	
22	Павлова Мария	Голубенко Артем	Врач	Здравпункт	

Рисунок 62 – Данный организационной диаграммы

8. Закрываем Excel. После закрытия Excel происходит возврат в программу MS Visio, где открыта страница импорта фотографий мастера (рис. 21).

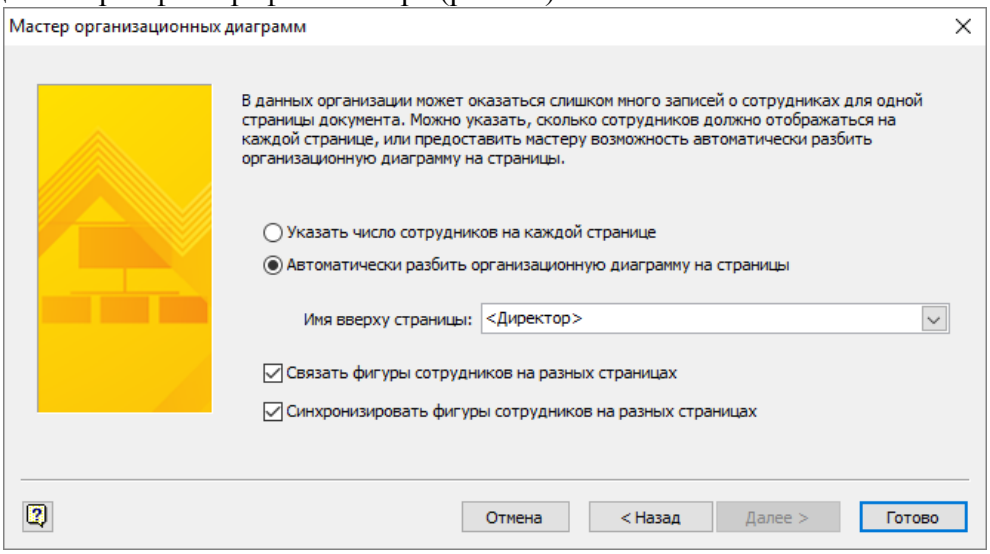


Рисунок 63 – Мастер организационных диаграмм. Шаг 4

9. Щелкнем на кнопке *Готово*, чтобы отобразить организационную диаграмму (рисунок 22).

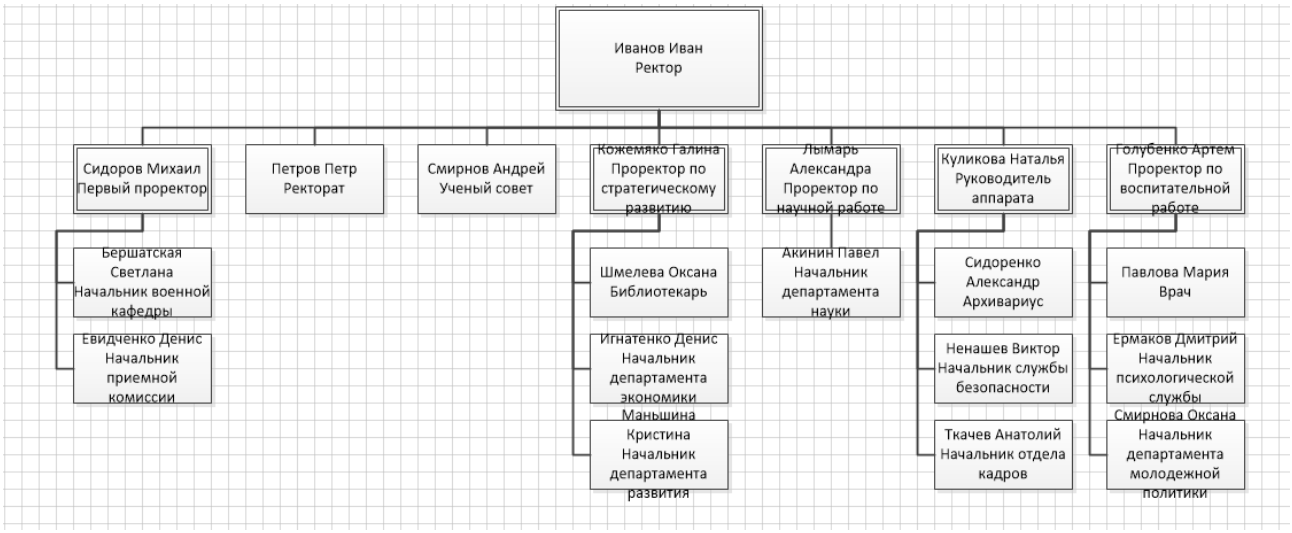


Рисунок 64 – Сформированная оргдиаграмма

10. На последнем шаге необходимо произвести форматирование диаграммы. В результате организационная диаграмма должна принять вид, показанный на рисунке 23.

Рисунок 65 – Итоговый вид диаграммы

5. Задание

Построить организационную диаграмму в соответствии в вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения организационной диаграммы, а также скриншоты результатов согласно заданию.

6. Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. «Таксопарк».

7. Контрольные вопросы

1. Что такое организационная диаграмма?
2. Способы построения оргдиаграмм в MS Visio?
3. Каковы принципы создания организационных диаграмм в MS Visio?
4. Какие существуют типы организационных структур? Перечислите их преимущества и недостатки.

Практическая работа №6

Построение диаграмм прецедентов на языке UML с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания диаграмм прецедентов (вариантов использования) на языке UML.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения диаграмм прецедентов с помощью MS Visio;

– получить навыки создания диаграмм прецедентов.

3. Краткие теоретические сведения

3.1. Общие сведения о языке UML

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

В языке UML используется ***четыре основных вида графических конструкций***:

– **Значки или пиктограммы**. Значок представляет собой графическую фигуру фиксированного размера и формы. Примерами значков могут служить окончания связей элементов диаграмм или некоторые другие дополнительные обозначения.

– **Графические символы на плоскости**. Такие двумерные символы изображаются с помощью некоторых геометрических фигур и могут иметь различную высоту и ширину с целью размещения внутри этих фигур других конструкций языка UML.

Наиболее часто внутри таких символов помещаются строки текста, которые уточняют семантику или фиксируют отдельные свойства соответствующих элементов языка UML. Информация, содержащаяся внутри фигур, имеет важное значение для конкретной модели проектируемой системы, поскольку регламентирует реализацию соответствующих элементов в программном коде.

– **Пути,** которые представляют собой последовательности из отрезков линий, соединяющих отдельные графические символы. При этом концевые точки отрезков линий должны обязательно соприкасаться с геометрическими фигурами, служащими для обозначения вершин диаграмм, как принято в теории графов. С концептуальной точки зрения путям в языке UML придается особое значение, поскольку они являются простыми топологическими сущностями.

– **Строки текста.** Служат для представления различных видов информации в некоторой грамматической форме. Предполагается, что каждое использование строки текста должно соответствовать синтаксису в нотации языка UML, посредством которого может быть реализован грамматический разбор этой строки.

При графическом изображении диаграмм следует придерживаться следующих ***основных рекомендаций***:

1. Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области.
2. Все сущности на диаграмме модели должны быть одного концептуального уровня. Здесь имеется в виду согласованность не только имен одинаковых элементов, но и возможность вложения отдельных диаграмм друг в друга для достижения полноты представлений.
3. Вся информация о сущностях должна быть явно представлена на диаграммах. Речь идет о том, что, хотя в языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию (например, в случае неявного указания видимости атрибутов и операций классов), необходимо стремиться к явному указанию свойств всех элементов диаграмм.
4. Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем использовании на практике. Например, наличие замкнутых путей при изображении отношений агрегирования или

композиции приводит к ошибкам в программном коде, который будет реализовывать соответствующие классы. Наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен также приводит к неоднозначной интерпретации и может служить источником проблем.

5. Диаграммы не следует перегружать текстовой информацией. Принято считать, что визуализация модели является наиболее эффективной, если она содержит минимум пояснительного текста.

6. Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов.

7. Количество типов диаграмм для конкретной модели приложения не является строго фиксированным.

3.2. Диаграмма вариантов использования (usecase diagram)

Разработка диаграммы вариантов использования преследует цели:

1. Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
2. Сформулировать общие требования к функциональному поведению проектируемой системы.
3. Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
4. Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

При этом **актером (actor)** или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

В свою очередь, **вариант использования (usecase)** служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

В самом общем случае, диаграмма вариантов использования представляет собой граф специального вида, который является графической нотацией для представления конкретных вариантов использования, актеров, возможно некоторых интерфейсов, и отношений между этими элементами.

Вариант использования (use case) **–** конструкция или стандартный элемент языка UML, который применяется для спецификации общих особенностей поведения системы или любой другой сущности предметной области без рассмотрения внутренней структуры этой сущности. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером.

Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Такой пояснительный текст получил название ***примечания или сценария*****.**

Отдельный вариант использования обозначается на диаграмме эллипсом (рис. 1), внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами.

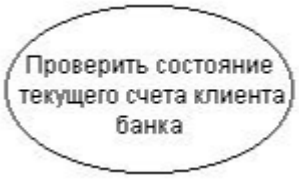


Рисунок 66 – Графическое обозначение варианта использования

Актер (actor) представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка «человечка» (рис. 2), под которой записывается конкретное имя актера.

Рисунок 67 – Графическое обозначение актера

Интерфейс (interface) служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры. В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов или функциональности для актеров. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации.
На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя (рис. 3).

Рисунок 68 – Графическое обозначение интерфейса

В качестве имени может быть существительное, которое характеризует соответствующую информацию или сервис (например, «датчик», «сирена», «видеокамера»), но чаще строка текста (например, «запрос к базе данных», «форма ввода», «устройство подачи звукового сигнала»).
Если имя записывается на английском, то оно должно начинаться с заглавной буквы I, например, ISecurelnformation, ISensor.
Графический символ отдельного интерфейса может соединяться на диаграмме сплошной линией с тем вариантом использования, который его поддерживает. Сплошная линия в этом случае указывает на тот факт, что связанный с интерфейсом вариант использования должен реализовывать все операции, необходимые для данного интерфейса, а возможно и больше (рис. 4а).
Кроме этого, интерфейсы могут соединяться с вариантами использования пунктирной линией со стрелкой (рис. 4б), означающей, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Рисунок 69 – Графическое обозначение взаимосвязей интерфейсов и вариантов использования

Примечания (notes) в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Применительно к диаграммам вариантов использования примечание может носить самую общую информацию, относящуюся к общему контексту системы.
Графически примечания обозначаются прямоугольником с «загнутым» верхним правым уголком (рис. 5). Внутри прямоугольника содержится текст примечания. Примечание может относиться к любому элементу диаграммы, в этом случае их соединяет пунктирная линия. Если примечание относится к нескольким элементам, то от него проводятся, соответственно, несколько линий. Разумеется, примечания могут присутствовать не только на диаграмме вариантов использования, но и на других канонических диаграммах.

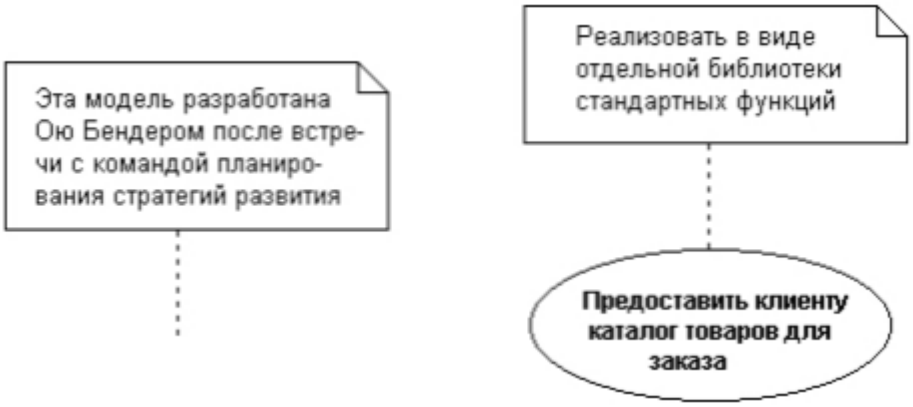


Рисунок 70 – Пример обозначения примечания

3.3. Отношения на диаграмме вариантов использования

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов.
Один актер может взаимодействовать с несколькими вариантами использования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис. Следует заметить, что два варианта использования, определенные для одной и той же сущности, не могут взаимодействовать друг с другом, поскольку каждый из них самостоятельно описывает законченный вариант использования этой сущности. Более того, варианты использования всегда предусматривают некоторые сигналы или сообщения, когда взаимодействуют с актерами за пределами системы. В то же время могут быть определены другие способы для взаимодействия с элементами внутри системы.
В языке UML имеется несколько стандартных ***видов отношений между актерами и вариантами использования:***

1. Отношение ассоциации (association relationship)

Отношение ассоциации является одним из фундаментальных понятий в языке UML и в той или иной степени используется при построении всех графических моделей систем в форме канонических диаграмм.
Применительно к диаграммам вариантов использования оно служит для обозначения специфической роли актера в отдельном варианте использования. Другими словами, ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы. Таким образом, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования, так же, как и на других диаграммах, отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь дополнительные условные обозначения, такие, например, как имя и кратность (рис. 6).

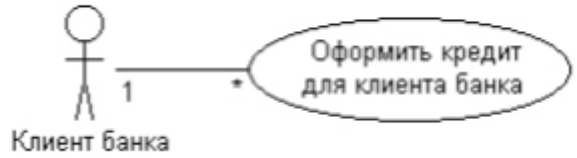


Рисунок 71 – Графическое обозначение отношения ассоциации

2. Отношение расширения (extend relationship)

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров.

Так, если имеет место отношение расширения от варианта использования А к варианту использования В, то это означает, что свойства экземпляра варианта использования В могут быть дополнены благодаря наличию свойств у расширенного варианта использования А.

Отношение расширения между вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом «extend» («расширяет»), как показано на рис. 7

Рисунок 72 – Графическое обозначение отношения расширения

Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

Один из вариантов использования может быть расширением для нескольких базовых вариантов, а также иметь в качестве собственных расширений несколько других вариантов. Базовый вариант использования может дополнительно никак не зависеть от своих расширений.

3. Отношение обобщения (generalization relationship)

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В.

В этом случае вариант А будет являться специализацией варианта В. При этом В называется предком или родителем по отношению А, а вариант А – потомком по отношению к варианту использования В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения.

Графически данное отношение обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования (рис. 8). Эта линия со стрелкой имеет специальное название – стрелка «обобщение».

Рисунок 73 – Графическое обозначение отношения обобщения

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами

поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.
Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Например, отношение обобщения от актера А к актеру В отмечает тот факт, что каждый экземпляр актера А является одновременно экземпляром актера В и обладает всеми его свойствами. В этом случае актер В является родителем по отношению к актеру А, а актер А, соответственно, потомком актера В. При этом актер А обладает способностью играть такое же множество ролей, что и актер В.
Графически данное отношение также обозначается стрелкой обобщения, т. е. сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительского актера (рис. 9).

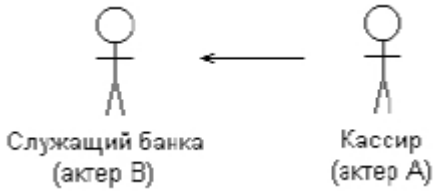


Рисунок 74 – Графическое обозначение отношения обобщения между актерами

4. Отношение включения (include relationship)

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в отношении включения.
Отношение включения, направленное от варианта использования А к варианту использования В, указывает, что каждый экземпляр варианта А включает в себя функциональные свойства, заданные для варианта В. Эти свойства специализируют поведение соответствующего варианта А на данной диаграмме. Графически данное отношение обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом «include» («включает»), как показано на рис. 10

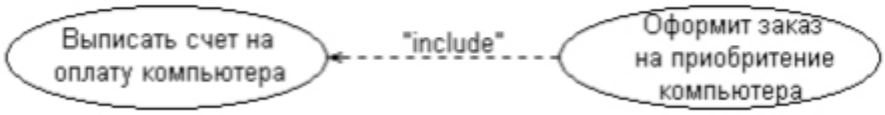


Рисунок 75 – Графическое обозначение отношения включения

5. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

9. Запустите MS Visio.
10. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
11. Окно программы примет вид, подобный рис. 11.

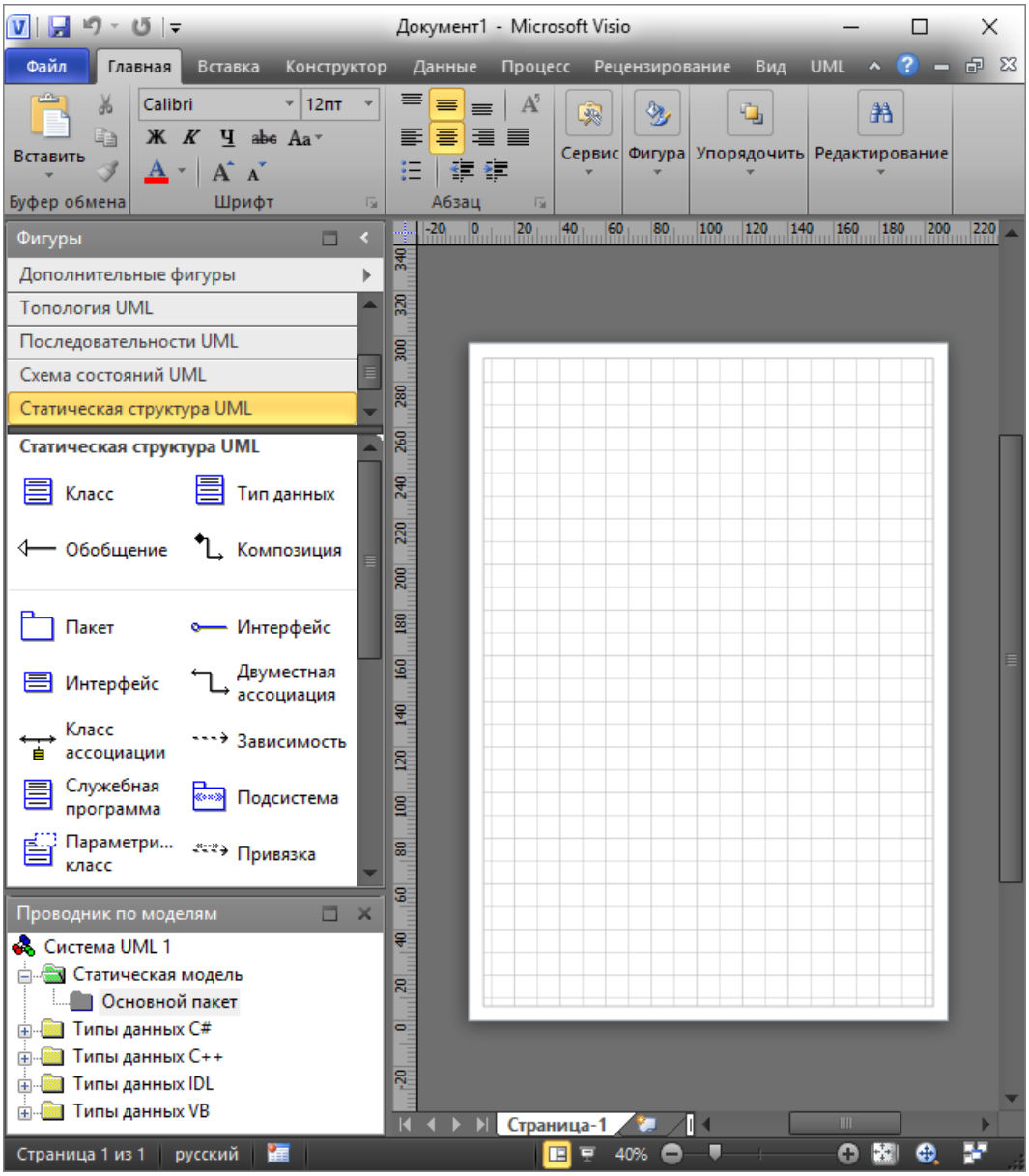


Рисунок 76 – Схема модели UML в MS Visio

12. Далее необходимо открыть все фигуры, необходимые для построения UML-диаграмм. Для этого в левой части экрана необходимо нажать кнопку *Дополнительные фигуры*. В открывшемся вспомогательном меню выбрать *Программы и БД -> Программное обеспечение* и выбрать все доступные фигуры для построения UML (рис. 12).

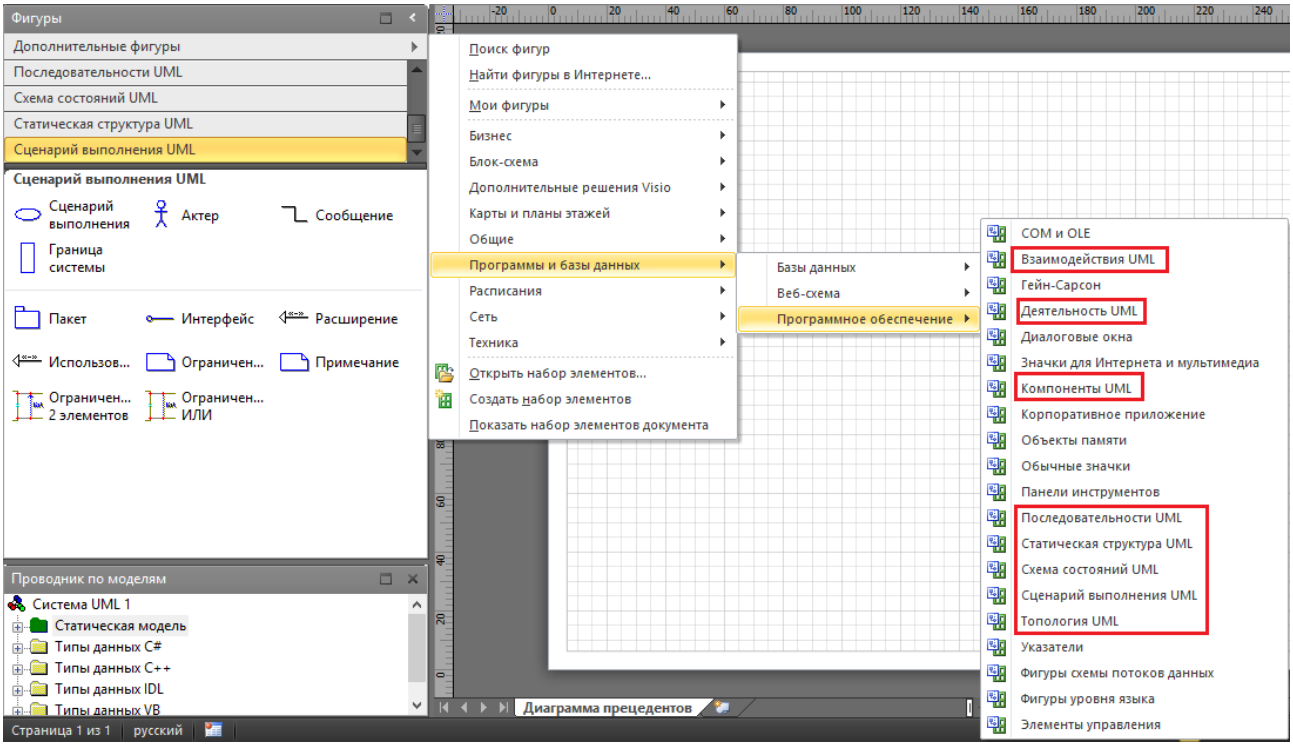


Рисунок 77 – Добавление фигур UML

13. После этого необходимо провести следующие этапы моделирования.

13.1. Выбор актеров.

В качестве актеров данной системы могут выступать два субъекта, один из которых является продавцом, а другой – покупателем. Каждый из этих актеров взаимодействует с рассматриваемой системой продажи товаров по каталогу и является ее пользователем, т. е. они оба обращаются к соответствующему сервису «Оформить заказ на покупку товара». Как следует из существа выдвигаемых к системе требований, этот сервис выступает в качестве варианта использования разрабатываемой диаграммы, первоначальная структура которой может включать в себя только двух указанных актеров и единственный вариант использования (рис. 14).

– В группе фигур *Сценарий выполнения UML* выбрать блок *Граница системы* и добавить его на лист.

– Внутрь границы системы добавить блок *Сценарий выполнения* и добавить к нему название, дважды щелкнув внутри блока.

– Добавить два блока *Актер* – покупатель и продавец.

– С помощью блока *Сообщение* установите связь актеров и варианта использования. Двойным щелчком правой кнопки мыши по блоку Сообщение откройте окно *Свойств ассоциации UML*, проведите настройки в соответствии с рисунком 13.

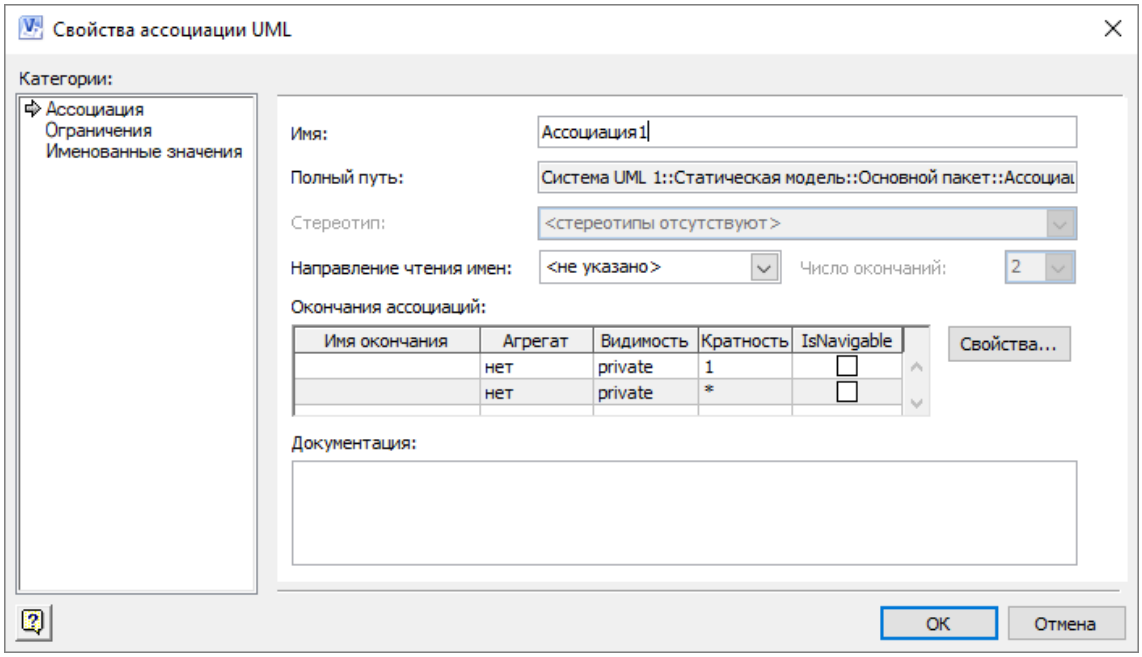


Рисунок 78 – Свойства ассоциации UML

Рисунок 79 – Исходная диаграмма вариантов использования системы по продаже товаров

13.2. Выделение дополнительных вариантов использования.

Детализировать вариант использования «Оформить заказ на продажу товара» можно выделив следующие дополнительные варианты использования:

- обеспечить покупателя информацией – является отношением включения;
- согласовать условия оплаты – является отношением включения;
- заказать товар со склада – является отношением включения;
- запросить каталог товаров – является отношением расширения.

Так как в MS Visio отсутствует отношение включения, его необходимо добавить самостоятельно. Для этого перейти на вкладку *UML* -> в группе *Модель* выбрать пункт *Стереотипы*. В открывшемся окне нажать кнопку *Создать* и настроить стереотип в соответствии с рисунком 15.

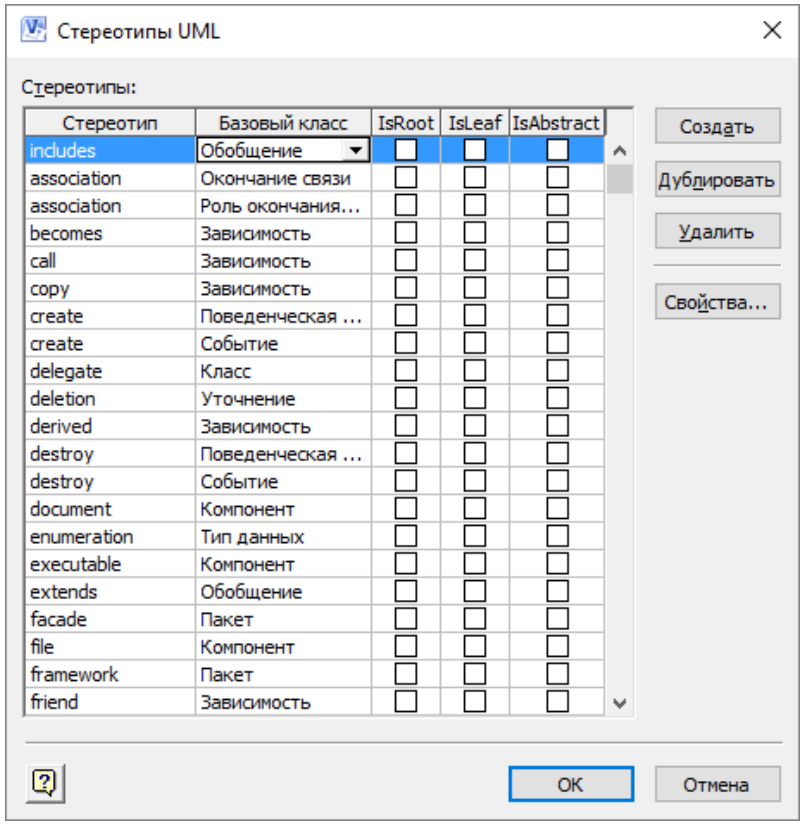


Рисунок 80 – Создание стереотипа

Далее на новом листе необходимо добавить границу системы и все варианты использования. После чего соединить варианты использования с помощью блока *Расширение.*

Для того, чтобы изменить тип отношения дважды щелкните по стрелке и окне свойств задайте необходимые параметры.

Дополненная диаграмма вариантов использования примет вид, показанный на рисунке 16.

Рисунок 81 – Дополненная диаграмма вариантов использования

13.3. Написание описательной спецификации для каждого варианта использования.

Спецификация для варианта использования «Оформить заказ на покупку компьютера» приведена в таблице 1.

Таблица 8 – Спецификация варианта использования

Раздел	**Описание**
Краткое описание	Покупатель желает оформить заказ на покупку компьютера, который он выбрал в каталоге товаров. При условии, что клиент зарегистрирован и выбранный компьютер есть в наличии оформляется заказ. Если клиент не зарегистрирован, то предлагается ему пройти регистрацию, и после этого заказать выбранный компьютер. Если компьютера нет в наличии, то предлагается заказать товар со склада в течении заданного срока поставки.
Субъекты	Продавец, Покупатель
Предусловия	В каталоге товаров имеются компьютеры, которые можно заказать. У покупателей есть доступ к системе для регистрации. Продавцы умеют пользоваться рассматриваемой системой продажи. У покупателя есть бонусы.
Основной поток	Зарегистрированный покупатель имеет возможность заказать любой компьютер из каталога товаров. В случае наличия выбранного компьютера оформляется заказ с присвоением ему уникального номера. После этого покупателю предлагается выбрать способ оплаты и способ получения компьютера. В случае отсутствия компьютера в наличии предлагается оформить заказ со склада и ожидания его поставки в рамках указанного срока или выбрать другой компьютер.
Альтернативный поток	Покупатель не зарегистрирован. В этом случае, прежде чем оформить заказ на компьютер, ему предлагается пройти регистрацию. Попытка заказать товар, который отсутствует на складе Начисление бонусов
Постусловия	Заказ оформлен и определен срок поставки компьютера и место его получения

На рисунках 17-19 приведены примеры диаграмм вариантов использования для различных систем.

Рисунок 82 – Пример 1

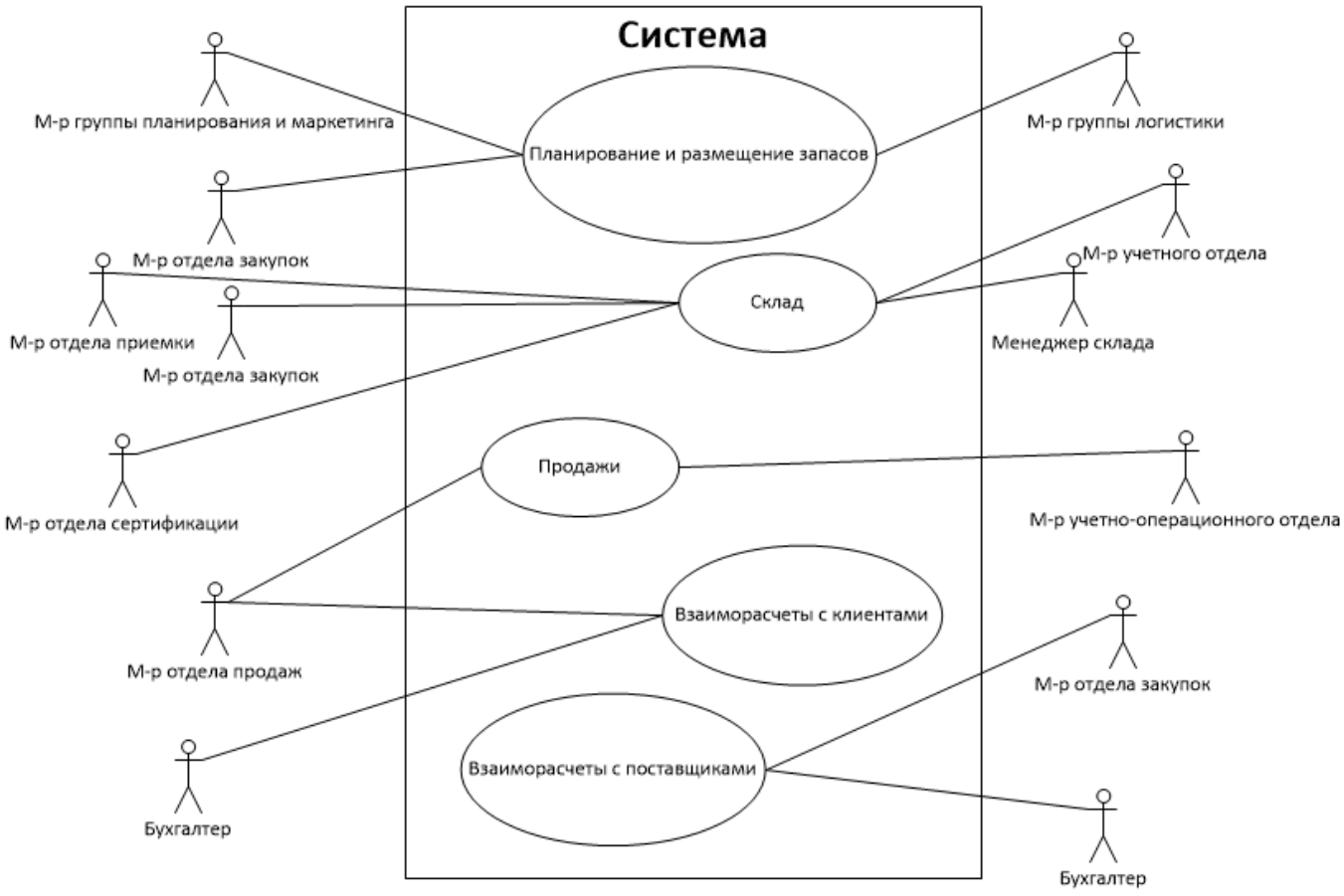


Рисунок 83 – Пример 2

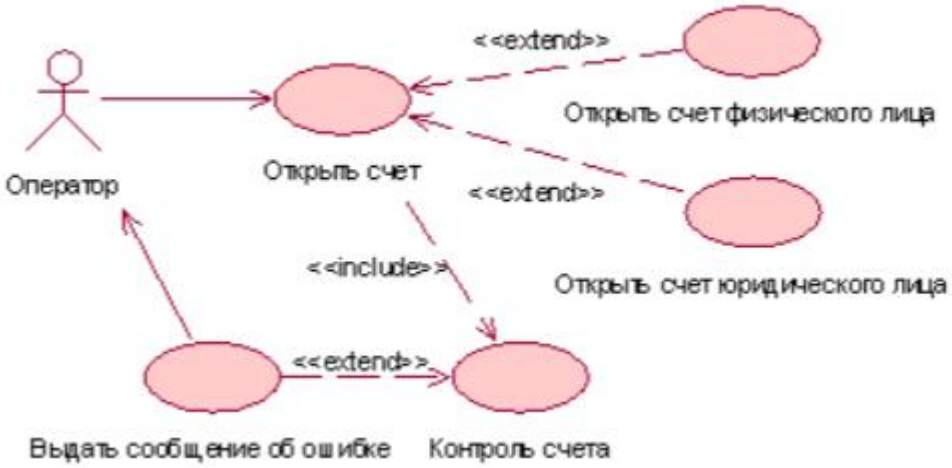


Рисунок 84 – Пример 3

6. Задание

Построить диаграмму прецедентов (вариантов использования) в соответствии с вариантом. Составить спецификцию.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения организационной диаграммы, а также скриншоты результатов согласно заданию.

7. Варианты

11. «Отдел кадров»;
12. «Агентство аренды»;
13. «Аптека»;
14. «Ателье»;
15. «Аэропорт»;
16. «Библиотека»;
17. «Кинотеатр»;
18. «Поликлиника»;
19. «Автосалон»;
20. «Таксопарк».

8. Контрольные вопросы

1. Для чего используется язык UML?
2. Назначение диаграммы вариантов использования?
3. Что такое «актер»?
4. Что такое «вариант использования»?
5. Что такое «интерфейс»?
6. Что такое «примечание»?
7. Перечислить виды отношений между актерами и вариантами использования, охарактеризовать каждое из них?

Практическая работа №7

Построение диаграмм классов на языке UML с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания диаграмм классов на языке UML, получение навыков построения диаграмм классов, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения диаграмм классов;

– ознакомиться с теоретическими вопросами построения диаграмм классов с помощью MS Visio;

– получить навыки создания диаграмм классов.

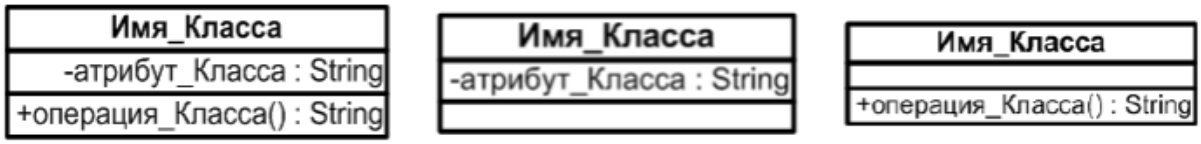
3. Краткие теоретические сведения

Диаграмма классов является одной из канонических диаграмм UML, создаваемой для визуализации структурированной статической модели предметной области. Этот вид диаграмм представляет собой графическое изображение объектов – классов с присущими им атрибутами, операциями и различных отношений между классами.

3.1. Классы

Класс (class) служит для обозначения множества объектов, обладающих функциональным набором одинаково описывающих параметров (атрибутов), реализуемых операций и однотипными отношениями с объектами других классов.

На диаграмме класс изображается габаритной прямоугольной рамкой, которая дополнительно может быть разделена горизонтальными линиями на секции, каждая из которых предназначена для указания имени, атрибутов (свойств) и реализуемых операций объектов данного класса (рис. 1 а, б, в).



а) б) в)

Рисунок 85 – а) Обозначение класса, б) Обозначение атрибутов, в) Обозначение операций

Имя класса является обязательным элементом в его обозначении и должно быть уникальным (хотя бы в пределах пакета), имеющее непосредственное отношение к контексту моделируемой предметной области.

В соответствии с принятым в языке UML общим соглашением в качестве имени класса используются существительные и прилагательные, каждое из которых начинается с заглавной буквы, записанные без пробелов. Например, в качестве имен классов могут быть использованы профессиональные термины: «Сотрудник», «Компания», «Руководитель», «Клиент», «Продавец», «Менеджер», «Офис», «Покупатель», «Датчик_Температуры» и др. Такое имена классов являются простыми.

Иногда возникает необходимость в явном указании пакета, к которому относится данный класс. С этой целью в условном обозначении перед именем класса указывается имя пакета и специальный символ разделитель – двойное двоеточие "::". Такое имя класса является квалифицированным. Текстовая строка имени класса в этом случае записывается в формате <Имя_пакета>::<Имя_класса>. Например, если определен пакет с именем «Банк», то имя класса «Счет» может быть записано так, как показано на рис. 2.

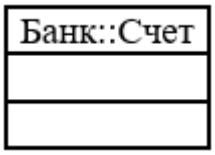


Рисунок 86 – Обозначение квалифицированного имени класса

3.2. Атрибуты классов

Содержательной характеристикой класса является атрибут, содержащий множество значений, которые могут принимать отдельные объекты этого класса. При этом, класс может иметь любое число атрибутов или не иметь ни одного. Так, например, атрибутами класса «Усилитель» являются частотный диапазон, выходная мощность, коэффициент нелинейных искажений, уровень шума и т. д.

Запись атрибута также представляет собой отдельную строку текста, содержащую обязательное имя, в котором обычно каждое слово пишется с заглавной буквы, за исключением первого, например, name (имя) или birth_Date (дата_Рождения).

Например, на рис. 3 указаны атрибуты класса Контейнер, в качестве которых выступают атрибут тип_Контейнера, атрибут регистрационный_Номер_Контейнера, регистрационный_Номер_После_Перерегистрации, рабочее_Состояние.

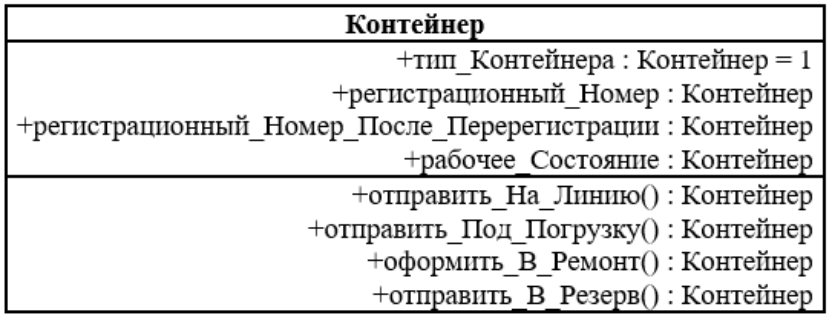
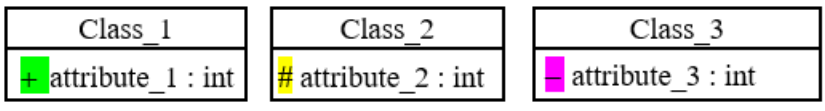


Рисунок 87 – Указание атрибутов класса Контейнер

Качественной характеристикой описания элементов класса является квантор видимости атрибута – потенциальная возможность других объектов модели оказывать влияние на отдельные аспекты поведения данного класса.

Эта характеристика может принимать одно из трех возможных значений и, соответственно, отображается при помощи специальных символов: символ "+" (public) обозначает атрибут с областью видимости типа общедоступный; атрибут с этой областью видимости доступен или виден из любого другого класса пакета, в котором определена диаграмма; например, для класса Class_1 указан атрибут общедоступного типа (рис. 4а); символ "#" (protected) обозначает атрибут с областью видимости типа защищенный; атрибут с этой областью видимости недоступен или невиден для всех классов, за исключением подклассов данного класса; например, для класса Class_2 указан атрибут защищенного типа (рис. 4б); символ "-" (private) обозначает атрибут с областью видимости типа закрытый; атрибут с этой областью видимости недоступен или невиден для всех классов без исключения; например, для класса Class_3 указан атрибут защищенного типа (рис. 4в);

Квантор видимости при описании атрибутов может быть опущен, что будет означать тот факт, что видимость атрибута не указывается.



а) б) в)

Рисунок 88 – Обозначение видимости атрибутов класса

3.3. Операции классов

Операция (operation) класса – это реализация услуги, которая может быть запрошена у любого объекта данного класса, чтобы вызвать определенное его поведение. Класс может иметь любое число операций либо не иметь ни одной. Так автомобиль может перемещаться по грунту, корабль – перемещаться по воде, компьютер – производить вычисления.

Представление полного синтаксиса записи операций класса также подчиняется определенным синтаксическим правилам: каждой операции класса соответствует отдельная строка, которая состоит из квантора видимости операции, обязательного имени операции, выражения типа возвращаемого операцией значения и, возможно, строки-свойства данной операции:

< квантор видимости >< имя операции > (список параметров) : < выражение типа возвращаемого значения >{строка-свойство}

Квантор видимости, как и в случае атрибутов класса, может принимать одно из трех возможных значений и, соответственно, также отображается при помощи специального символа.

Для именования операции используются короткие глагольные конструкции, описывающие некоторое поведение класса, которому принадлежит операция. Обычно каждое слово в имени операции пишется с заглавной буквы, за исключением первого.

Например, запись +создать() – может обозначать абстрактную операцию по созданию отдельного объекта класса, которая является общедоступной и не содержит формальных параметров, запись +нарисовать(форма: Многоугольник = прямоугольник, цвет_заливки: Color = (O, O, 255)) – может обозначать операцию по изображению на экране монитора прямоугольной области синего цвета, если не указываются другие значения в качестве аргументов данной операции.

(список–параметров) содержит необязательные аргументы, синтаксис которых совпадает с синтаксисом атрибутов;

< выражение типа возвращаемого значения > является необязательной спецификацией и зависит от конкретного языка программирования;

{строка-свойство} показывает значения свойств, которые применяются к данной операции.

Например, запись запросить_Счет_Клиента(номер_счета:Integer) – обозначает операцию по установлению наличия средств на текущем счете клиента банка. При этом аргументом данной операции является номер счета клиента, который записывается в виде целого числа (например, «123456»).

Рисунок 89 – Обозначение операции «запросить_Счет_Клиента»

Квантор видимости для операции может быть опущен. В этом случае его отсутствие означает, что видимость операции не указывается.

3.4. Отношения между классами

Классы на диаграмме связываются различными типами отношений. При этом совокупность типов таких отношений фиксирована в языке UML и предопределена их семантикой.

3.4.1. Отношение зависимости

Отношением зависимости (dependency relationship) называют связь по использованию, когда изменение в спецификации одного класса может повлиять на поведение другого. Отношение зависимости используется в такой ситуации, когда некоторое изменение одного элемента модели может потребовать изменения другого зависимого от него элемента модели. Отношение зависимости графически изображается пунктирной линией между соответствующими элементами со стрелкой на одном из ее концов, направленной к той сущности, от которой зависит данная сущность. Например, некая сущность Класс_2 использует другую сущность Класс_1 (рис. 6).

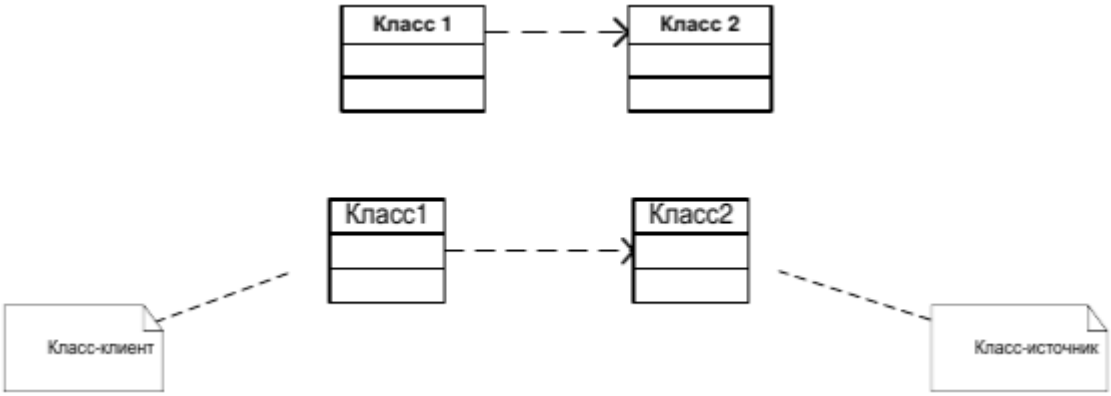


Рисунок 90 – Обозначение отношения зависимости

В качестве класса-клиента и класса-источника зависимости может выступать множество элементов модели. В этом случае одна линия со стрелкой, выходящая от источника зависимости, расщепляется в некоторой точке на несколько отдельных линий, каждая из которых имеет отдельную стрелку для класса-клиента.

Например, если функционирование сущности Класс_С зависит от особенностей реализации сущностей Класс_А и Класс_Б, то данная зависимость может быть изображена следующим образом (рис. 7).

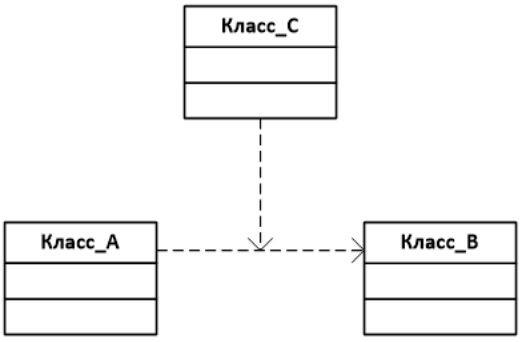


Рисунок 91 – Обозначение зависимости между классом-клиентом (Класс_С) и классами-источниками (Класс_А и Класс_В)

3.4.2. Отношение ассоциации

Ассоциацией (association relationship) называется структурная связь, показывающая, что объекты одного класса некоторым образом связаны с объектами другого или того же самого класса.

Ассоциация может отображаться графически линией со стрелкой (маркером в виде треугольника), показывающей направление следования классов и кратность – количество объектов, связанных отношением. Отсутствие стрелки рядом с именем ассоциации означает, что порядок следования классов в рассматриваемом отношении не определен (рис. 8).

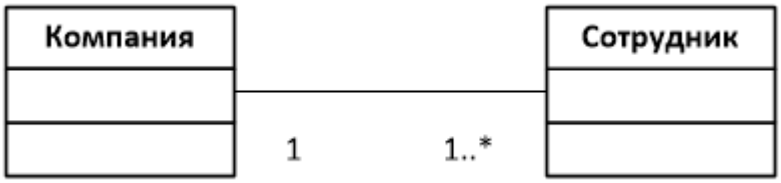


Рисунок 92 – Обозначение отношения ассоциации

Так, в примере на рис. 8 кратность «1» для класса «Компания» означает, что каждый сотрудник может работать только в одной компании. Кратность «1..*» для класса «Сотрудник» означает, что в каждой компании могут работать несколько сотрудников, общее число которых заранее неизвестно и ничем не ограничено.

Специальной формой или частным случаем отношения ассоциации является отношение агрегации, которое, в свою очередь, тоже имеет специальную форму – отношение композиции (см. пункт 3.4.4).

3.4.3. Отношение агрегации

Отношение агрегации (generalization relationship) имеет место между несколькими классами в том случае, если один из классов представляет собой некоторую сущность, включающую в себя в качестве составных частей другие сущности.

Данное отношение имеет фундаментальное значение для описания структуры сложных систем, поскольку применяется для представления системных взаимосвязей типа «часть – целое».

Это отношение по своей сути описывает декомпозицию или разбиение сложной системы на более простые составные части, которые также могут быть подвергнуты декомпозиции, если в этом возникнет необходимость в последующем.

Так, автомобиль состоит из кузова, двигателя, трансмиссии и т.п., а в состав приемопередающего устройства входят передатчик, приемник и антенно-фидерное устройство.

Графически отношение агрегации изображается сплошной линией, один из концов которой представляет собой геометрическую фигуру – ромб. Этот ромб указывает на тот из классов, который представляет собой «целое» (рис. 9).

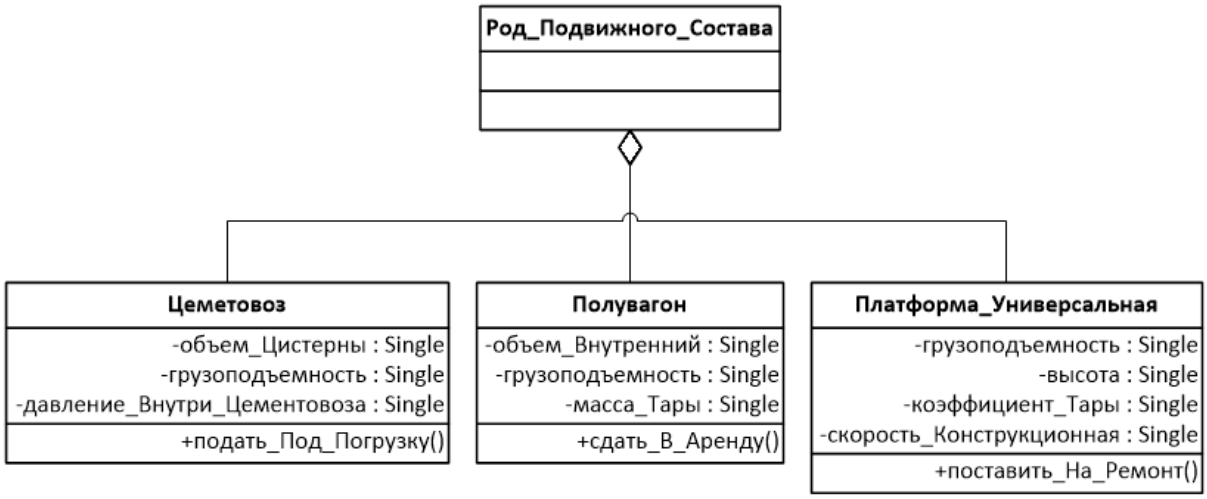


Рисунок 93 – Обозначение отношения агрегации

Примером отношения агрегации может служить деление класса Аналитическая_информация на составные части: Отчет_по_грузу, Отчет_по_контейнерам, Отчет_по_тарифам (рис. 10).

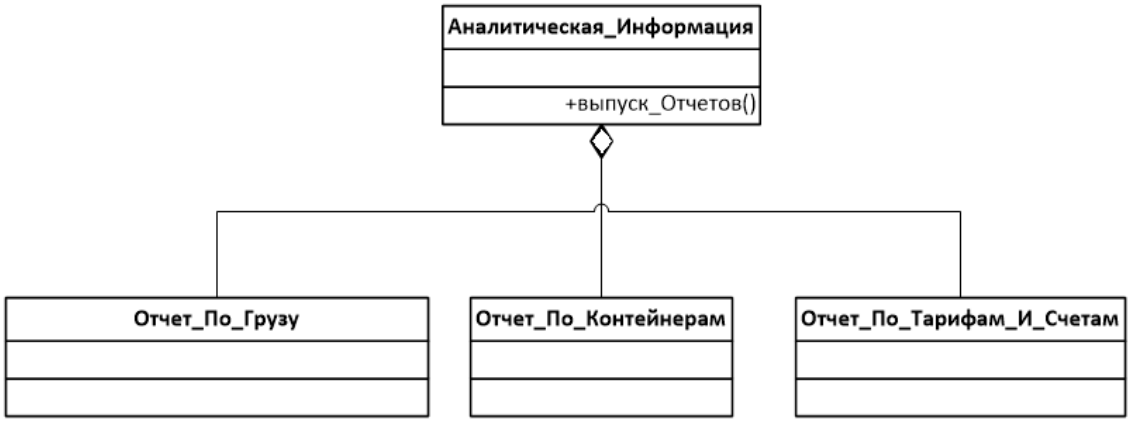


Рисунок 94 – Пример отношения агрегации

Отношение агрегации обладает кратностью. Так, класс Система_обеспечения_безопасности_объектов может содержать содержит одну подсистему Сопровождение_грузов, которая в свою очередь может содержать, например, четыре класса Охрана_вооруженная, каждый из которых может принадлежать лишь одному классу Сопровождение_грузов (рис. 11).

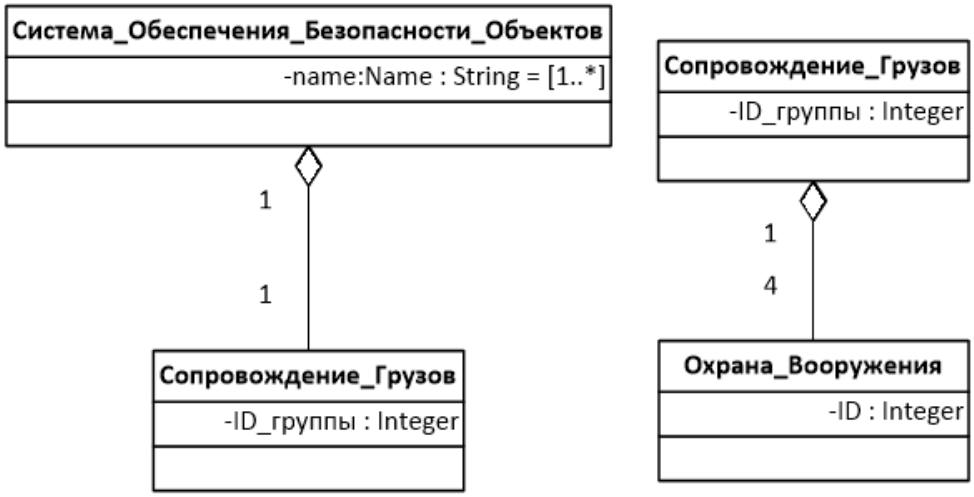


Рисунок 95 – Пример обозначения кратности отношения агрегации

3.4.4. Отношение композиции

Отношение композиции (realization relationship) служит для выделения специальной формы отношения «часть-целое», при которой составляющие части в некотором смысле находятся внутри целого.

Специфика взаимосвязи между ними заключается в том, что части не могут выступать в отрыве от целого, т. е. с уничтожением целого уничтожаются и все его составные части.

Графически отношение композиции изображается сплошной линией, один из концов которой представляет собой закрашенный внутри ромб. Этот ромб указывает на тот из классов, который представляет собой класс-композицию или «целое» (рис. 12).

Рисунок 96 – Обозначение отношения композиции

Применительно к классу Заказ_на_перевозку_грузов отношение композиции может иметь следующий вид (рис. 13).

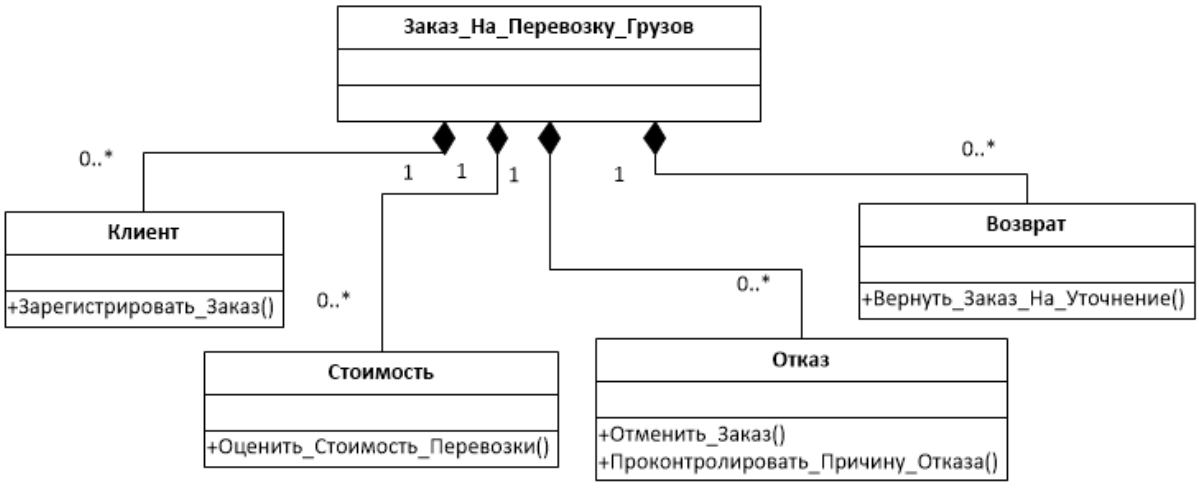


Рисунок 97 – Обозначение на диаграмме отношения композиции

3.4.4. Отношение обобщения

Отношение обобщения (генерализация) является обычным таксономическим отношением между более общим элементом (класс-предок) и более частным или специальным элементом (класс-потомок).

Применительно к диаграмме классов данное отношение описывает иерархическое строение классов и наследование их свойств и поведения. При этом предполагается, что класс-потомок обладает всеми свойствами и поведением класса-предка, а также имеет свои собственные свойства и поведение, которые отсутствуют у класса-предка.

На диаграммах отношение обобщения обозначается сплошной линией с треугольной стрелкой на одном из концов, направленной на более общий класс (класс-предок или суперкласс) от более специального класса (класса-потомка или подкласса) (рис. 14).

Как правило, на диаграмме может указываться несколько линий для одного отношения обобщения, что отражает его таксономический характер. В этом случае более общий класс разбивается на подклассы одним отношением обобщения, например, так, как показано на рис. 14.

В этом случае данные отдельные линии изображаются сходящимися к единственной стрелке, имеющей с ними общую точку пересечения. Родительский Класс Отчет_По_Заказам_На_Перевозку имеет три потомка Отчет_По_Количеству_Заказов, Отчет_По_Клиентам, Отчет_За_Период, которые наследуют структуру и поведение родительского класса.

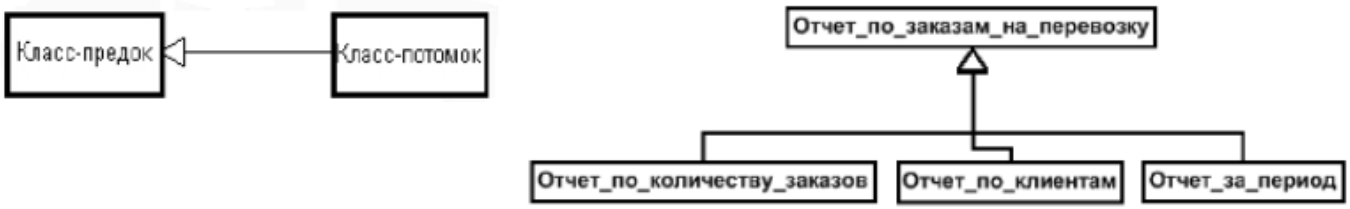


Рисунок 98 – Обозначение на диаграмме отношения обобщения

Для связей обобщения язык UML содержит ограничения. В большинстве случаев ограничение размещается рядом с элементом и заключается в фигурные скобки, например {complete}.

В качестве ограничений могут быть использованы следующие ключевые слова языка UML:

1. {complete} означает, что в данном отношении обобщения специфицированы все классы-потомки, и других классов-потомков у данного класса-предка быть не может.

Например, класс Клиент_банка является предком для двух классов: Физическое_лицо и Компания, и других классов-потомков он не имеет.

На соответствующей диаграмме классов это можно указать явно, записав рядом с линией обобщения данную строку-ограничение (рис. 15).

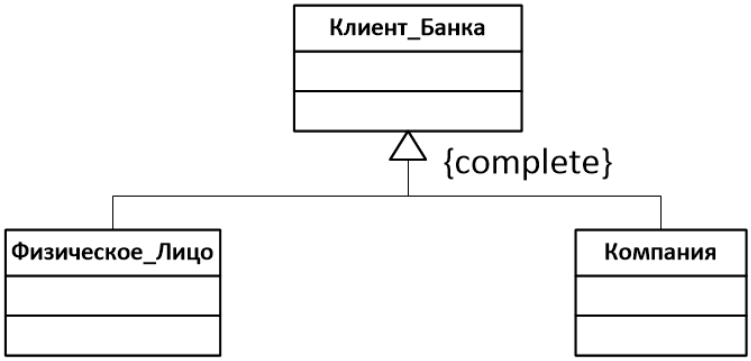


Рисунок 99 – Обозначение ограничения {complete} отношения обобщения

2. {incomplete} означает тот факт, что на диаграмме указаны в обобщении не все классы-потомки. В последующем, возможно, восполнить их перечень, не изменяя уже построенную диаграмму.

3. {disjoint} означает тот факт, что классы-потомки не могут содержать объектов, одновременно являющихся экземплярами двух или более классов.

В приведенном выше примере это условие также выполняется, поскольку предполагается, что никакое конкретное физическое лицо не может являться одновременно и конкретной компанией. В этом случае рядом с линией обобщения можно записать данную строку-ограничение.

4. {overlapping} означает, что отдельные экземпляры классов-потомков могут принадлежать одновременно нескольким классам.

Например, класс Транспорт может быть специализирован путем создания подклассов Наземный_Транспорт и Водный_Транспорт, автомобиль – амфибия относится к обоим классам.

4. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

14. Запустите MS Visio.
15. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
16. Окно программы примет вид, подобный рис. 16.

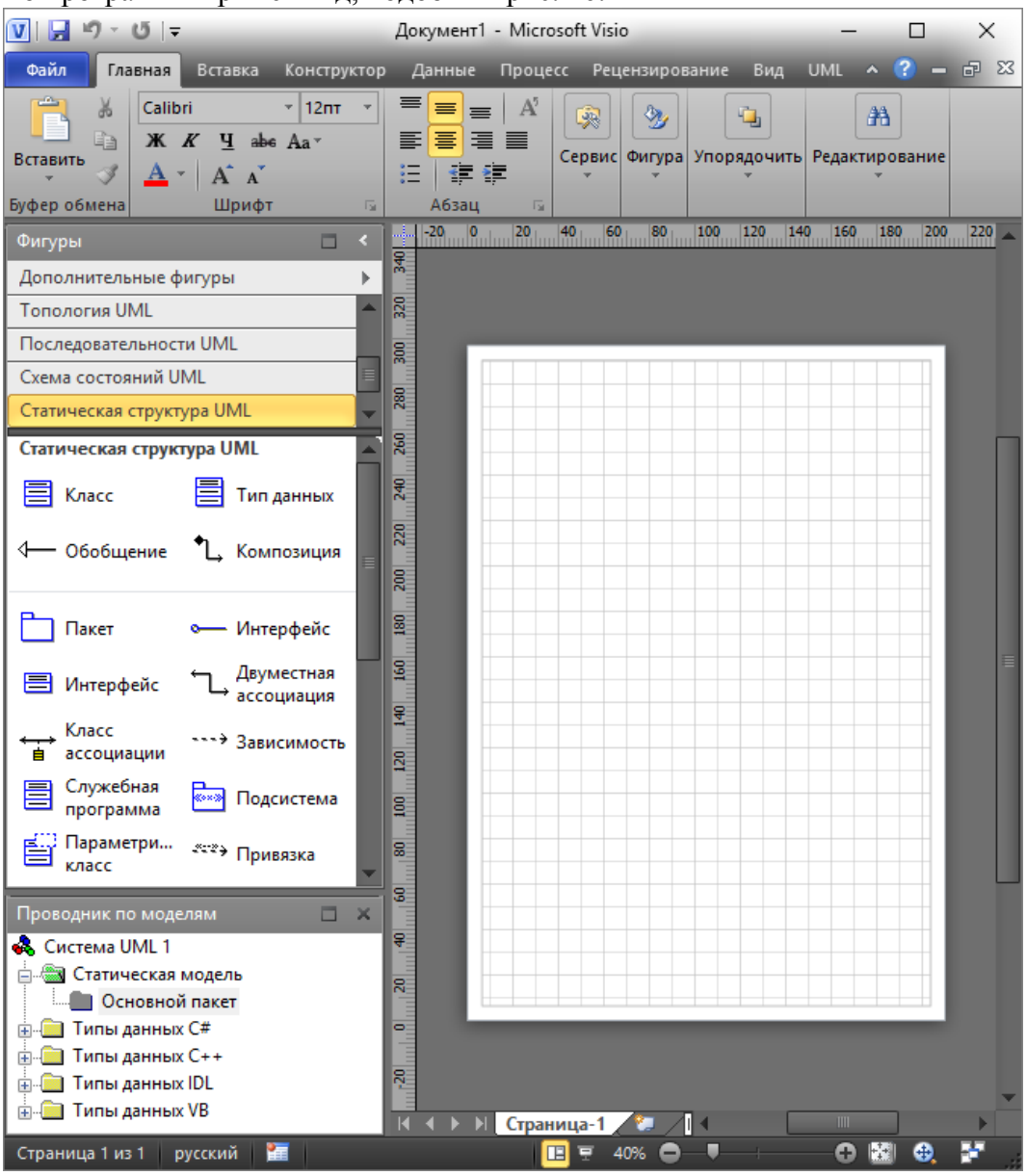


Рисунок 100 – Схема модели UML в MS Visio

17. Ознакомьтесь с элементами графического интерфейса и найдите обязательные панели инструментов **Фигуры**, содержащие категории **Деятельность UML, Взаимодействия UML, Компоненты UML, Топология UML, Последовательности UML, Схема состояний UML, Статическая структура UML, Сценарий выполнения UML, Проводник по моделям**, содержащий иерархическую структуру объектов Системы UML1, Рабочую область, ярлык Страница_1, горизонтальную и вертикальную линейки. (рис. 17).

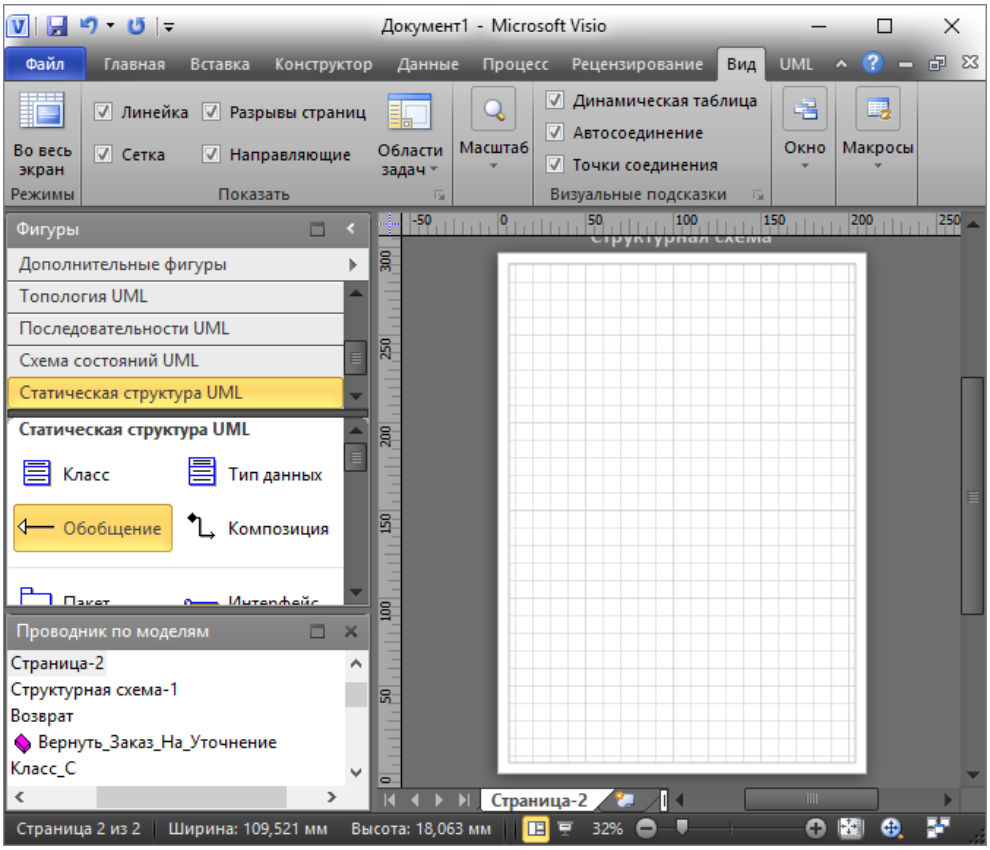


Рисунок 101 – Добавление фигур UML

18. Установите следующие параметры страницы: **Ориентация** – Альбомная, **Автоподбор размера** – выключен, **Имя страницы** – *Диаграмма классов для системы продажи товаров по каталогу.*

19. Перейдите в категорию **Статическая структура UML**, ознакомьтесь с содержимым этой категории и найдите элементы: Класс, Пакет, Подсистема, Интерфейс, Метакласс, Двусторонняя ассоциация, Обобщение, Композиция, Примечание, Ограничение и др.

20. Создайте поэтапно статическую структуру классов UML, с помощью которой может быть сформирована некоторая функциональная часть системы, например, *Система продажи товаров по каталогу*. Для чего:

– Выберите структурные элементы (идентифицируйте классы), участвующие в организации продаж, например, *Продавец, Товар, Заказ, Заказ_Оплата, Клиент, Корпоративный_Клиент, Частный_Клиент* и создайте предварительный вариант совокупности классов с указанием имен (один из возможных вариантов представлен на рис. 18).

– Установите для каждого класса атрибуты в соответствии с перечнем и содержательным описанием бизнес-процессов:

например, для класса *Продавец* в качестве атрибутов могут выступать данные: *фамилия, имя, отчество, телефон*. В данном случае все атрибуты видимы, принадлежат основному пакету *Продавец* (рис. 19).

для класса *Товар* в качестве атрибутов могут выступать данные: тип, марка, артикул (рис. 20).

Рисунок 102 – Формирование статической структуры объектов диаграммы

Рисунок 103 – Описание атрибутов класса Продавец

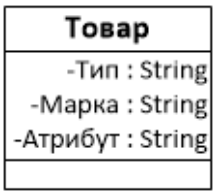


Рисунок 104 – Описание атрибутов класса Товар

для класса *Заказ* в качестве атрибутов могут выступать данные: количество, цена, статус, а в качестве операций – сформировать заказ.

Рисунок 105 – Описание атрибутов и операций класса Заказ

для класса *Заказ_Оплата* в качестве атрибутов могут выступать данные: дата получения, проплачен, номер, цена, а в качестве операций – отправить, закрыть.

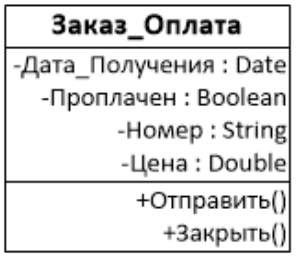


Рисунок 106 – Описание атрибутов и операций класса Заказ_Оплата

для класса *Клиент* в качестве атрибутов могут выступать данные: имя, адрес, а в качестве операций – кредитный рейтинг.

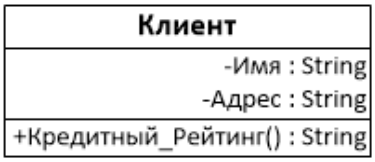


Рисунок 107 – Описание атрибутов и операций класса Клиент

для класса *Корпоративный_Клиент* в качестве атрибутов могут выступать данные: контактное имя, кредитный рейтинг, кредитный лимит, а в качестве операций – сделать, напоминание, счет за месяц.

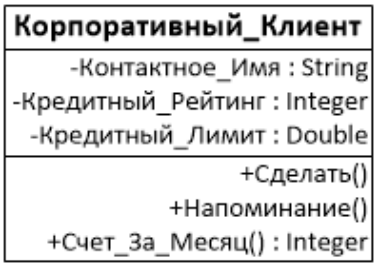


Рисунок 108 – Описание атрибутов и операций класса Корпоративный_Клиент

для класса *Частный_Клиент* в качестве атрибутов могут выступать данные: номер кредитной карты.

Рисунок 109 – Описание атрибутов класса Частный_Клиент

для класса *Вариант_Оплаты* в качестве атрибутов могут выступать данные: тип оплаты, а в качестве операций – выбор варианта оплаты.

Рисунок 110 – Описание атрибутов и операций класса Вариант_Оплаты

для класса *Каталог_Товаров* в качестве атрибутов могут выступать данные: тип, марка, артикул, а в качестве операций – проверить наличие.

Рисунок 111 – Описание атрибутов и операций класса Каталог_товаров

для класса *Склад* в качестве атрибутов могут выступать данные: товар, наличие, количество, а в качестве операций – Проверить наличие.

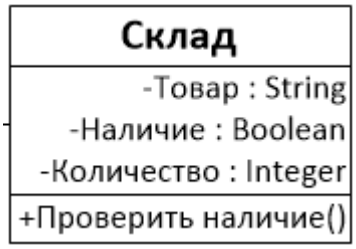


Рисунок 112 – Описание атрибутов и операций класса Склад

– Убедитесь, что все элементы наполнены адекватным содержанием и расположите все структурные элементы диаграммы наиболее оптимально на странице для установления отношений между ними.

В качестве примера на рис. 26 показан набор классов, описывающих реализацию системы продаж товаров по каталогу. Акцент сделан на классе *Клиент*, с которым связан класс *Заказ_Оплата* посредством двусторонней ассоциации «один-ко-многим», *Вариант_Оплаты* – двусторонней ассоциацией «один-к-одному» и классы *Корпоративный_Клиент* и *Частный_Клиент* посредством отношения обобщения. Классы *Заказ_Оплата* и *Товар* связаны с классом *Заказ* посредством двусторонней ассоциации «один-ко-многим». Класс *Товар* связан с классом *Продавец* двусторонней ассоциацией «многие-ко-многим» и классом *Каталог_Товаров* двусторонней ассоциацией «один-ко-многим». Класс *Каталог_Товаров* связан посредством двусторонней ассоциации «многие-ко-многим» с классом *Склад.*

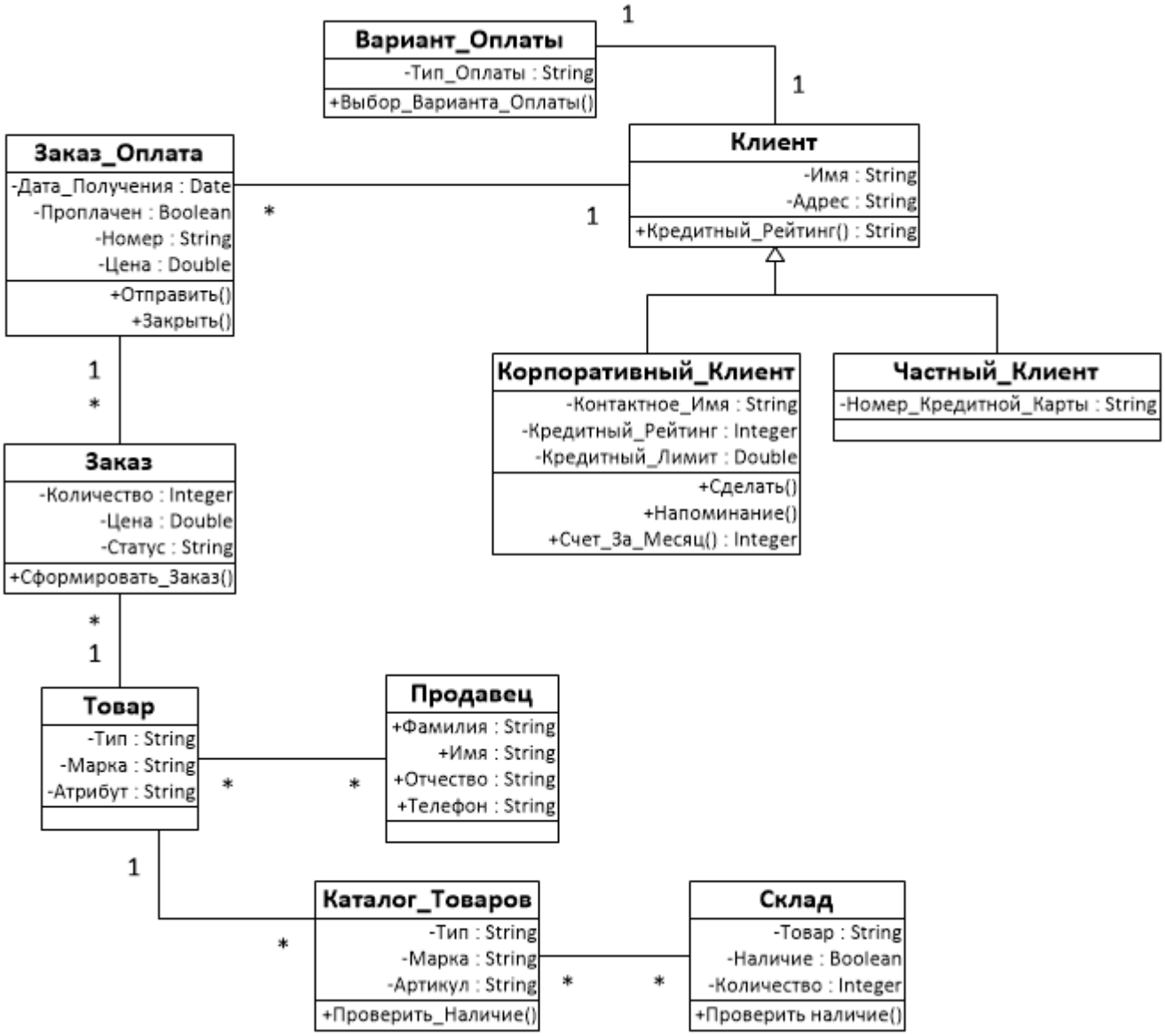


Рисунок 113 – Фрагмент диаграммы классов, описывающей реализацию систем продаж товаров по каталогу

21. Создайте новую страницу с именем *Диаграмма классов учета клиентов*, и установите следующие опции: **Ориентация** – Альбомная, **Автоподбор размера** – выключен.

22. Идентифицируйте классы учета клиентов, осуществляющих заказы и создайте диаграмму классов с указанием их имен, атрибутов, операций, например, так как показано на рис. 30.

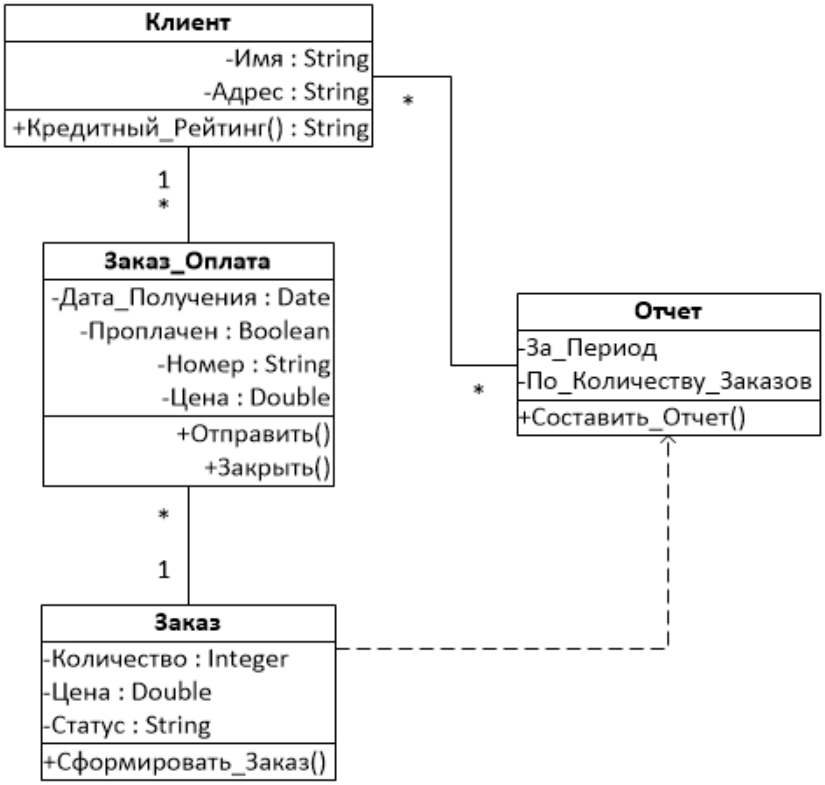


Рисунок 114 – Фрагмент диаграммы классов, описывающей процесс формирования отчетности

5. Задание

Построить диаграмму классов в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

6. Варианты

21. «Отдел кадров»;
22. «Агентство аренды»;
23. «Аптека»;
24. «Ателье»;
25. «Аэропорт»;
26. «Библиотека»;
27. «Кинотеатр»;
28. «Поликлиника»;
29. «Автосалон»;
30. «Таксопарк».

7. Контрольные вопросы

8. Каково назначение диаграммы классов?
9. Назовите основные элементы диаграммы классов.
10. Какие виды связей доступны в диаграмме классов?
11. Для чего используется каждый вид связи?
12. Как создать диаграмму классов в VISIO?

Практическая работа №8

Построение диаграмм состояний на языке UML с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания диаграмм состояний на языке UML, получение навыков построения диаграмм состояний, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения диаграмм состояний на языке UML;

– ознакомиться с теоретическими вопросами построения диаграмм состояний с помощью MS Visio.

3. Краткие теоретические сведения

Диаграмма состояний описывает процесс изменения состояний только одного класса, а точнее – одного экземпляра определенного класса, т. е. моделирует все возможные изменения в состоянии конкретного объекта. При этом изменение состояния объекта может быть вызвано внешними воздействиями со стороны других объектов или извне. Именно для описания реакции объекта на подобные внешние воздействия и используются диаграммы состояний.

Диаграмма состояний описывает возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение элемента модели в течение его жизненного цикла. Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий.

Хотя диаграммы состояний чаще всего используются для описания поведения отдельных экземпляров классов (объектов), но они также могут быть применены для спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, операции и методы.

Диаграмма состояний по существу является графом специального вида, который представляет некоторый автомат. Понятие автомата в контексте UML обладает довольно специфической семантикой, основанной на теории автоматов. Вершинами этого графа являются состояния и некоторые другие типы элементов автомата (псевдосостояния), которые изображаются соответствующими графическими символами. Дуги графа служат для обозначения переходов из состояния в состояние. Диаграммы состояний могут быть вложены друг в друга, образуя вложенные диаграммы более детального представления отдельных элементов модели.

Рисунок 115 – Пример диаграммы состояний для технического устройства типа «Компьютер»

Состояние

Под состоянием понимается абстрактный метакласс, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений атрибутов класса или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Рисунок 116 – Графическое обозначение состояния

Секция «Список внутренних действий» содержит перечень внутренних действий или деятельностей, которые выполняются в процессе нахождения моделируемого элемента в данном состоянии. Каждое из действий записывается в виде отдельной строки и имеет следующий формат:

<метка-действия '/' выражение-действия>

Метка действия указывает на обстоятельства или условия, при которых будет выполняться деятельность, определенная выражением действия:

– ***entry*** – эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент входа в данное состояние (входное действие);

– ***exit*** – эта метка указывает на действие, специфицированное следующим за ней выражением действия, которое выполняется в момент выхода из данного состояния (выходное действие);

– ***do*** – эта метка специфицирует выполняющуюся деятельность («doactivity»), которая выполняется в течение всего времени, пока объект находится в данном состоянии, или до тех пор, пока не закончится вычисление, специфицированное следующим за ней выражением действия. В последнем случае при завершении события генерируется соответствующий результат;

– ***include*** – эта метка используется для обращения к подавтомату, при этом следующее за ней выражение действия содержит имя этого подавтомата.

Пример: Аутентификация входа

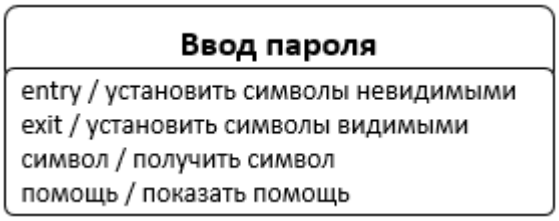


Рисунок 117 – Состояние для ввода пароля

Начальное и конечное состояния

Начальное состояние представляет собой частный случай состояния, которое не содержит никаких внутренних действий (псевдосостояния). В этом состоянии находится объект по умолчанию в начальный момент времени. Оно служит для указания на диаграмме состояний графической области, от которой начинается процесс изменения состояний.

Конечное (финальное) состояние представляет собой частный случай состояния, которое также не содержит никаких внутренних действий (псевдосостояния). В этом состоянии будет находиться объект по умолчанию после завершения работы автомата в конечный момент времени.

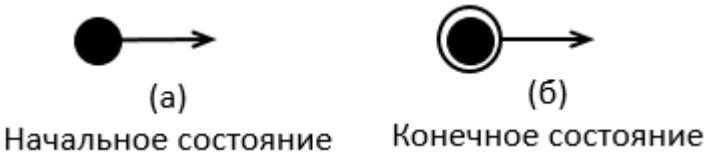


Рисунок 118 – Графическое изображение начального и конечного состояний

Переход

Простой переход (simpletransition) представляет собой отношение между двумя последовательными состояниями, которое указывает на факт смены одного состояния другим. Пребывание моделируемого объекта в первом состоянии может сопровождаться выполнением некоторых действий, а переход во второе состояние будет возможен после завершения этих действий, а также после удовлетворения некоторых дополнительных условий. В этом случае говорят, что переход срабатывает, Или происходит срабатывание перехода. До срабатывания перехода объект находится в предыдущем от него состоянии, называемым исходным состоянием, или в источнике (не путать с начальным состоянием – это разные понятия), а после его срабатывания объект находится в последующем от него состоянии (целевом состоянии).

Переход осуществляется при наступлении некоторого события: окончания выполнения деятельности (doactivity), получении объектом сообщения или приемом сигнала. На переходе указывается имя события. Кроме того, на переходе могут указываться действия, производимые объектом в ответ на внешние события при переходе из одного состояния в другое. Срабатывание перехода может

зависеть не только от наступления некоторого события, но и от выполнения определенного условия, называемого сторожевым условием. Объект перейдет из одного состояния в другое в том случае, если произошло указанное событие и сторожевое условие приняло значение «истина».

На диаграмме состояний ***переход изображается*** сплошной линией со стрелкой, которая направлена в целевое состояние. Каждый переход может помечен строкой текста, которая имеет следующий общий формат:

<сигнатура события>'['<сторожевое условие>']' <выражение действия>

Событие

Событие представляет собой спецификацию некоторого факта, имеющего место в пространстве и во времени. Про события говорят, что они «происходят», при этом отдельные события должны быть упорядочены во времени. После наступления некоторого события нельзя уже вернуться к предыдущим событиям, если такая возможность не предусмотрена явно в модели.

События играют роль стимулов, которые инициируют переходы из одних состояний в другие. В качестве событий можно рассматривать сигналы, вызовы, окончание фиксированных промежутков времени или моменты окончания выполнения определенных действий. Имя события идентифицирует каждый отдельный переход на диаграмме состояний и может содержать строку текста, начинающуюся со строчной буквы.

Сторожевое условие

Сторожевое условие (guardcondition), если оно есть, всегда записывается в прямых скобках после события и представляет собой некоторое булевское выражение.

Если сторожевое условие принимает значение «истина», то соответствующий переход может сработать, в результате чего объект перейдет в целевое состояние. Если же сторожевое условие принимает значение «ложь», то переход не может сработать, и при отсутствии других переходов объект не может перейти в целевое состояние по этому переходу. Однако вычисление истинности сторожевого условия происходит только после возникновения ассоциированного с ним события, инициирующего соответствующий переход.

В общем случае из одного состояния может быть несколько переходов с одним и тем же событием-триггером. При этом никакие два сторожевых условия не должны одновременно принимать значение «истина». Каждое из сторожевых условий необходимо вычислять всякий раз при наступлении соответствующего события.

Пример

Рисунок 119 – Диаграмма состояний для моделирования почтовой программы клиента

Составное состояние и подсостояние

Составное состояние (compositestate) – такое сложное состояние, которое состоит из других вложенных в него состояний. Последние будут выступать по отношению к первому как подсостояния (substate).

Рисунок 120 – Графическое изображение составного состояния

Последовательные подсостояния (sequential substates) используются для моделирования такого поведения объекта, во время которого в каждый момент времени объект может находиться в одном и только одном подсостояний. Поведение объекта в этом случае представляет собой последовательную смену подсостояний, начиная от начального и заканчивая конечным подсостояниями. Хотя объект продолжает находиться в составном состоянии, введение в рассмотрение последовательных подсостояний позволяет учесть более тонкие логические аспекты его внутреннего поведения.

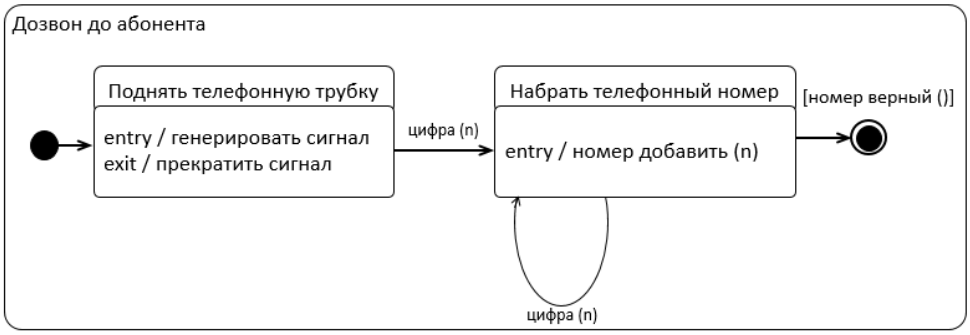


Рисунок 121 – Пример составного состояния с двумя вложенными последовательными подсостояниями

Параллельные подсостояния (concurrentsubstates) позволяют специфицировать два и более подавтомата, которые могут выполняться параллельно внутри составного события. Каждый из подавтоматов занимает некоторую область (регион) внутри составного состояния, которая отделяется от остальных горизонтальной пунктирной линией. Если на диаграмме состояний имеется составное состояние с вложенными параллельными подсостояниями, то объект может одновременно находиться в каждом из этих подсостояний.

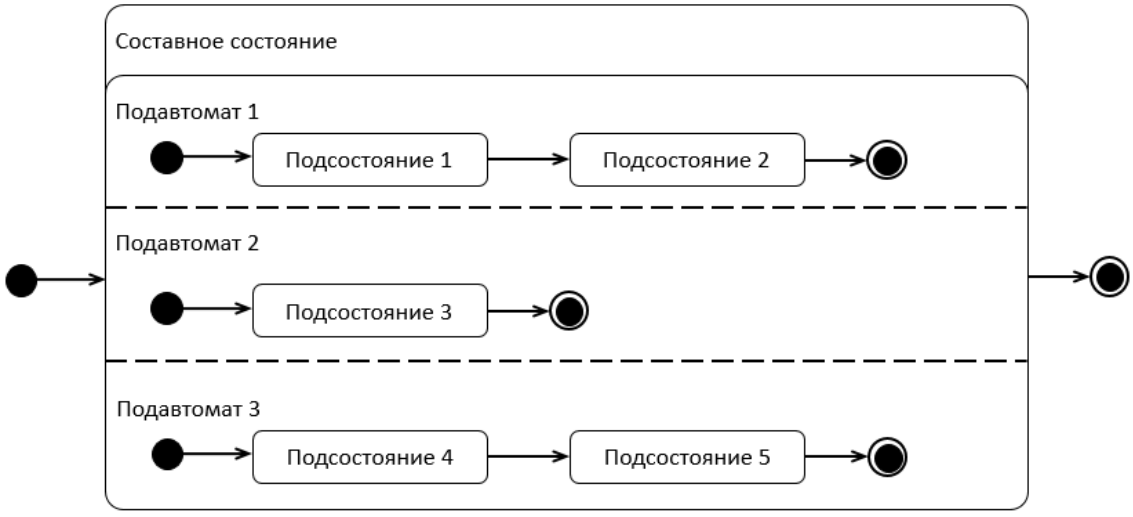


Рисунок 122 – Графическое изображение составного состояния с вложенными параллельными подсостояниями

Синхронизирующие состояния

Поведение параллельных подавтоматов независимо друг от друга, что позволяет реализовать многозадачность в программных системах. Однако в отдельных случаях может возникнуть необходимость учесть в модели синхронизацию наступления отдельных событий. Для этой цели в языке UML имеется специальное псевдосостояние, которое называется синхронизирующим состоянием.

Синхронизирующее состояние (synch state) обозначается небольшой окружностью, внутри которой помещен символ звездочки "*". Оно используется совместно с переходом-соединением или переходом-ветвлением для того, чтобы явно указать события в других подавтоматах, оказывающие непосредственное влияние на поведение данного подавтомата.

Для иллюстрации использования синхронизирующих состояний рассмотрим упрощенную ситуацию с моделированием процесса постройки дома. Предположим, что постройка дома включает в себя строительные работы (возведение фундамента и стен, возведение крыши и отделочные работы) и работы по электрификации дома (подведение электрической линии, прокладка скрытой электропроводки и установка осветительных ламп). Очевидно, два этих комплекса работ могут выполняться параллельно, однако между ними есть некоторая взаимосвязь.

В частности, прокладка скрытой электропроводки может начаться лишь после того, как будет завершено возведение фундамента и стен. А отделочные работы следует начать лишь после того, как будет закончена прокладка скрытой электропроводки. В противном случае отделочные работы придется проводить повторно. Рассмотренные особенности синхронизации этих параллельных процессов учтены на соответствующей диаграмме состояний с помощью двух синхронизирующих состояний.

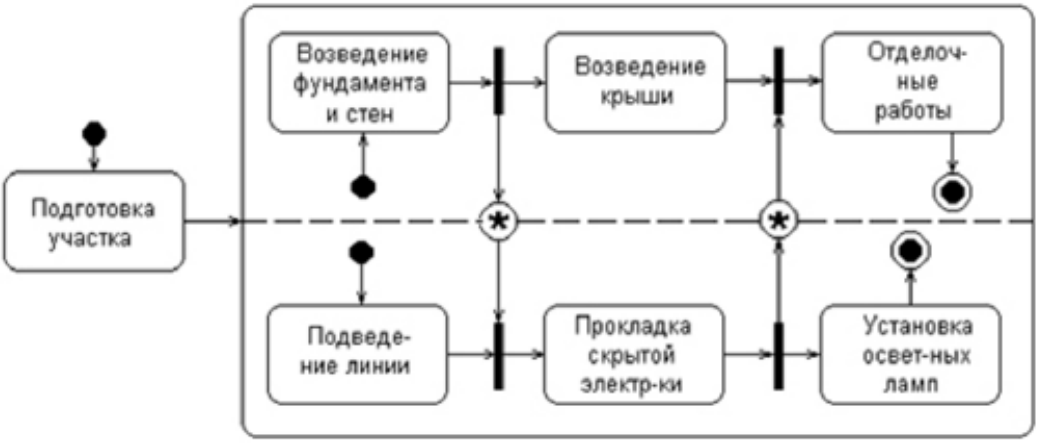


Рисунок 123 – Диаграмма состояний для примера со строительством дома

Примеры диаграмм состояний изображены на рисунках 10-12.

Рисунок 124 – Диаграмма состояний для моделирования запроса данных из БД

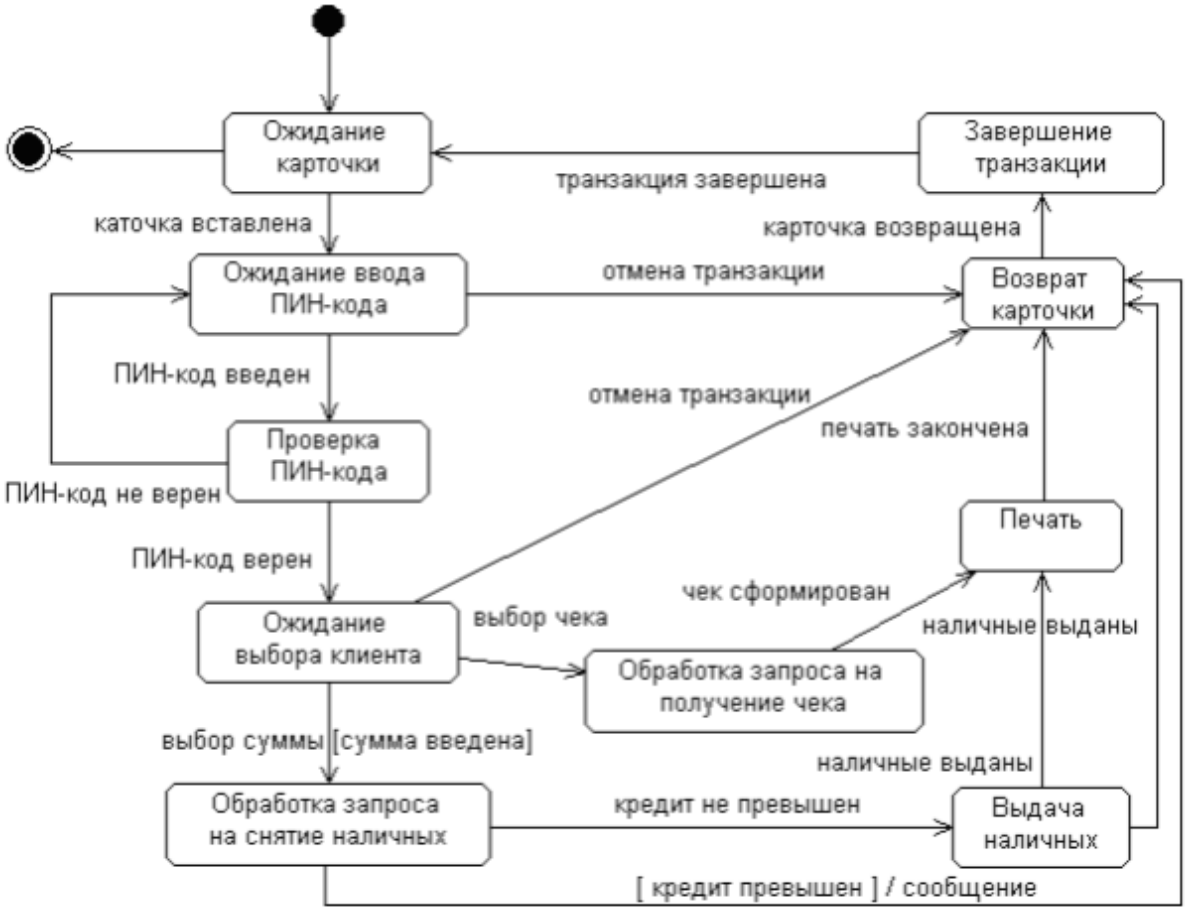


Рисунок 125 – Диаграмма состояний для моделирования поведения банкомата

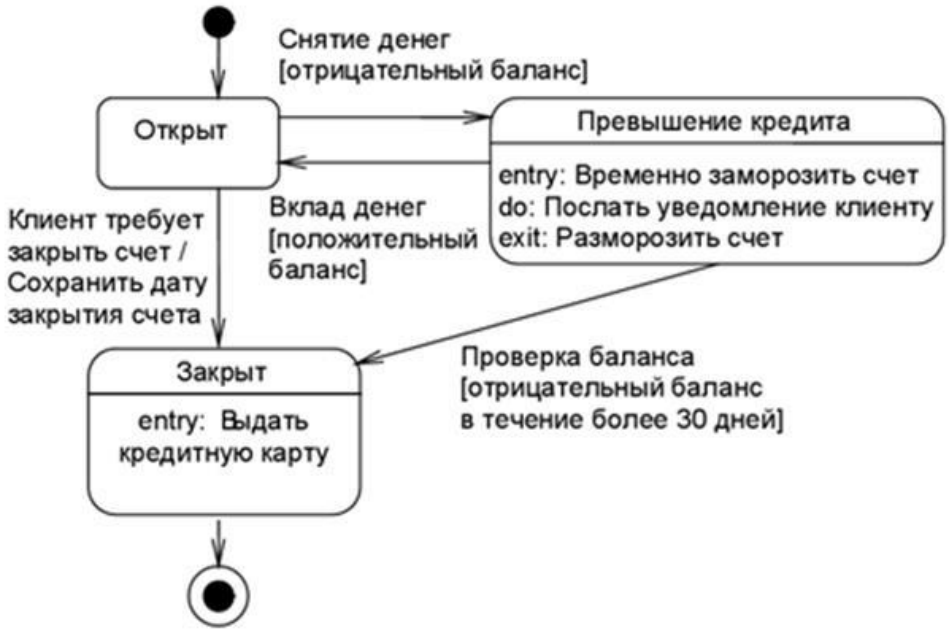


Рисунок 126 – Диаграмма состояний моделирования деятельности сотрудника банка

4. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

23. Запустите MS Visio.
24. На экране выбора шаблона выберите категорию *Программы и БД* и в ней элемент *Схема модели UML*. Нажмите кнопку *Создать* в правой части экрана.
25. Окно программы примет вид, подобный рис. 13.

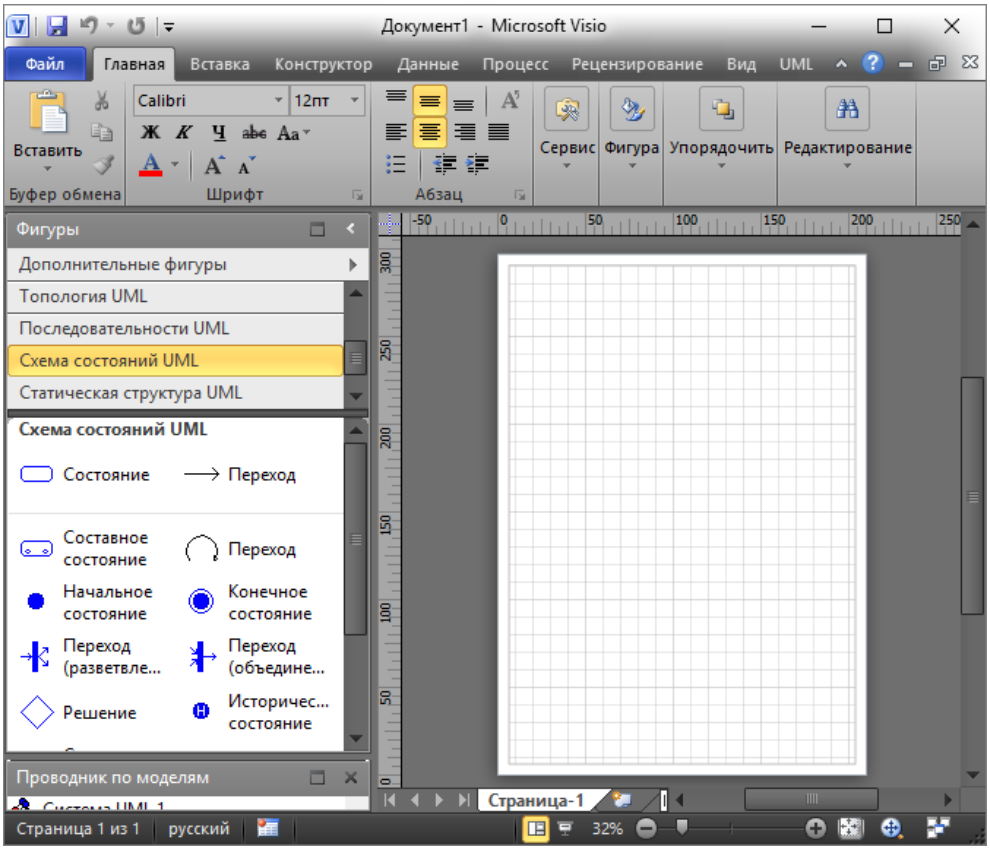


Рисунок 127 – Окно MS Visio для работы с диаграммами состояний

Клиент оформляет заказ. Класс Заказ имеет атрибут Статус. Проследим динамику движения заказов в системе с помощью ***диаграммы состояний, составленной для класса Заказ.***

Данные о сформированном заказе поступают продавцу, который проверяет наличие товаров из заказа, проверяет оплату заказа, комплектует его и делает отметку о готовности. После оплаты заказа он выдается клиенту. Продавец делает отметку о том, что заказ выдан.

Если после проверки кредитного рейтинга клиента, он окажется отрицательным, то заказ будет отклонен.

Построим диаграмму состояний для класса Заказ. Для этого, в файле с диаграммой классов, созданной в практическом занятии 8, необходимо проделать следующие действия:

1. Щелкнуть правой кнопкой мыши по *классу Заказ*.
2. В контекстном меню выбрать пункт *Схемы*.
3. Т.к. в настоящее время уже созданных схем нет, нажать кнопку *Создать* и выбрать *Схема состояний*.
4. Переименовать созданный лист в *Схема состояний-Заказ*.
5. Построить диаграмму состояний для класса Заказ в соответствии с рисунком 14.

После формирования заказа он должен быть оплачен. Обработка заказа подразумевает проверку наличия товара и проверку оплаты. Переход в одно из состояний На комплектации, Укомплектован, Выдан означает смену Статуса заказа.

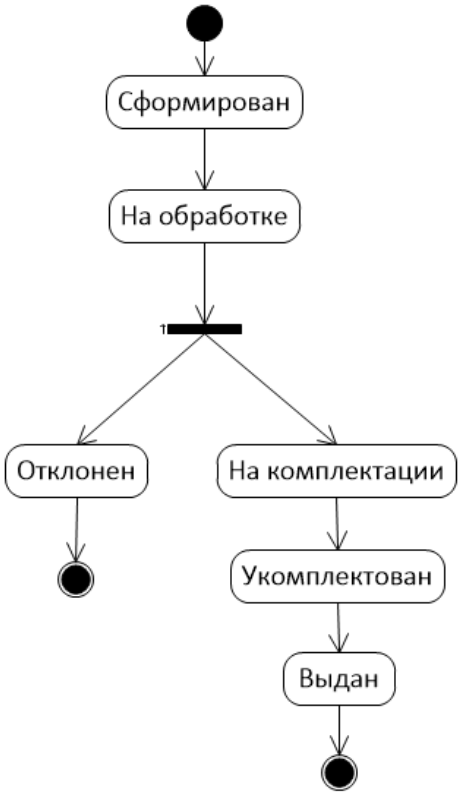


Рисунок 128 – Диаграмма состояний для класса Заказ

Далее опишем с помощью ***диаграммы состояний процесс оплаты заказа клиентом***, которому соответствует класс ЗаказОплата.

Построим диаграмму состояний для проверки оплаты заказа.

Чтобы проверить оплату заказа, необходимо определить, существует ли сам заказ. Результатом проверки оплаты заказа является вывод либо сообщения о произведенной оплате с параметрами (дата оплаты), либо сообщения об ожидании оплаты.

Событием, предшествующим проверке оплаты заказа, является занесение информации о заказе в базу данных заказов.

Чтобы построить диаграмму состояний для класса ЗаказОплаты, необходимо проделать действия, описанные в пунктах 1-4 построения диаграммы состояний для класса Заказ. Полученная диаграмма должна иметь вид, изображенный на рис. 15.

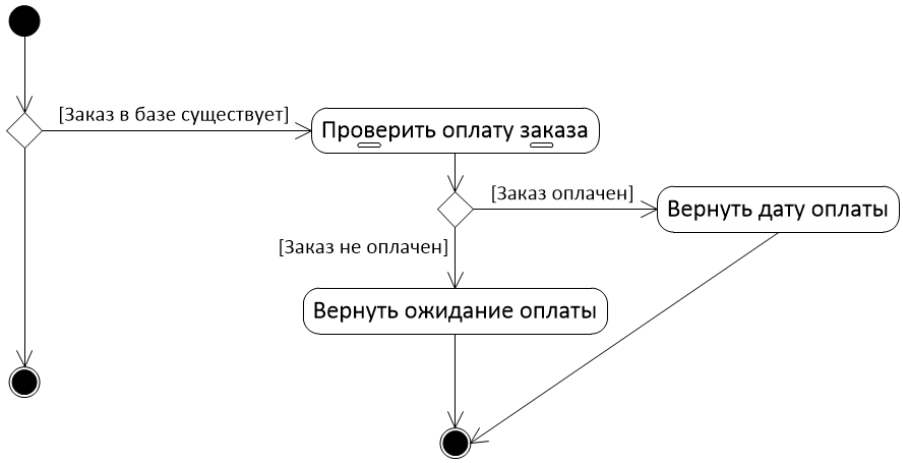


Рисунок 129 – Диаграмма состояний проверки платы заказа

На этой диаграмме есть составное состояние ***«Проверить оплату заказа»***, т.к. оно включает в себя *проверку кредитного рейтинга клиента* и *проверку выбора варианта оплаты клиентом*.

Оплату заказа может произвести только клиент с положительным кредитным рейтингом, поэтому необходимым условием проверки оплаты заказа является проверка кредитоспособности клиента. Если клиент имеет отрицательный кредитный рейтинг, то заказ отклоняется, и на этом дальнейшие события не имеют смысла.

Если кредитный рейтинг клиента положительный, то необходимо проверить, выбрал ли клиент вариант оплаты. Событие, которое переводит систему в состояние ожидания выбора варианта оплаты клиентом, является получение сообщения о кредитоспособности клиента.

Оплата может быть произведена наличными средствами в магазине или с помощью безналичного расчета. В первом случае необходимо договориться с клиентом о дате и времени его прибытия в магазин. Во втором случае необходимо сообщить клиенту о наличии/поступлении товара. Событие, которое переводит систему в состояние ожидания оплаты, является выбор клиентом варианта оплаты.

Соответствующие диаграммы состояний имеют вид (рис. 16 и рис. 17).

Для создания диаграмм состояний, которые входят в состав составного состояния, нужно:

1. Щелкнуть правой кнопкой мыши по Составному состоянию и выбрать пункт Схема.
2. Либо, в Проводнике по моделям выделить название составного состояния и создать новую страницу с помощью кнопки  .

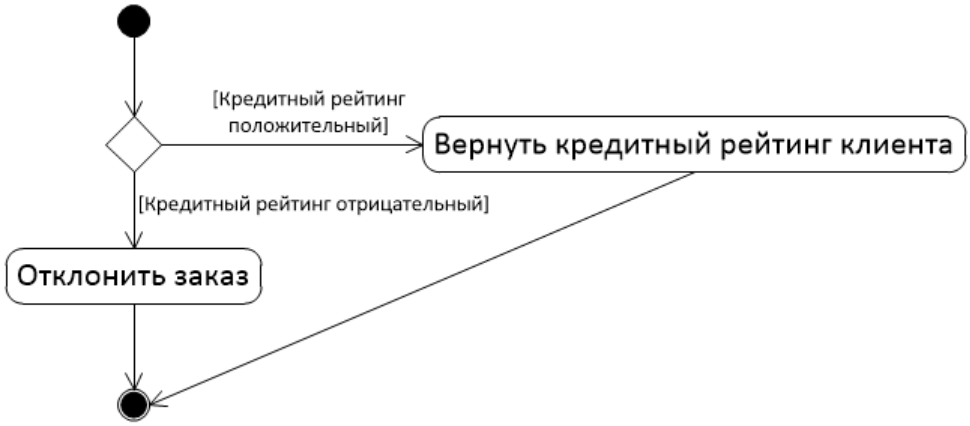


Рисунок 130 – Диаграмма состояний для проверки кредитного рейтинга клиента

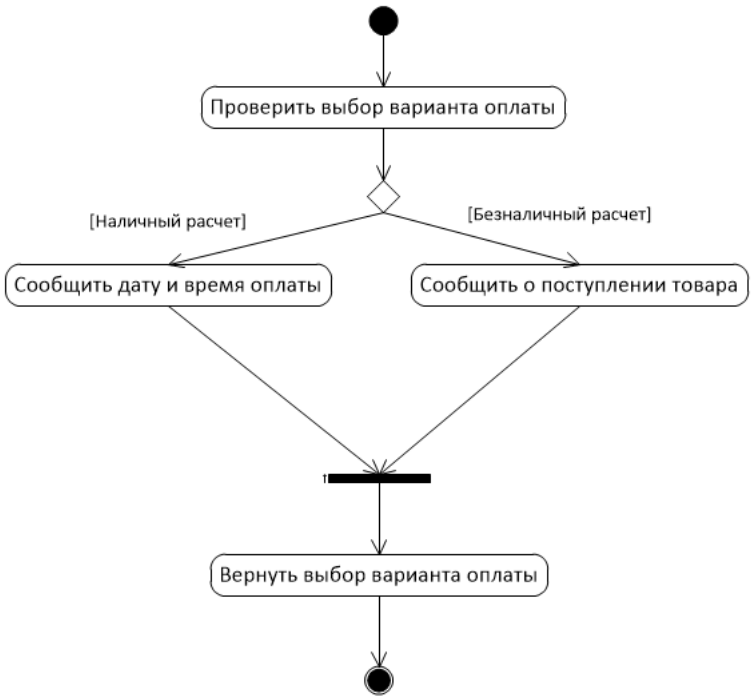


Рисунок 131 – Диаграмма состояний для проверки варианта оплаты

5. Задание

Построить диаграмму состояний в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Создать диаграммы состояния не менее чем для трех классов, описанных в практическом занятии №8.

6. Варианты

31. «Отдел кадров»;

32. «Агентство аренды»;
33. «Аптека»;
34. «Ателье»;
35. «Аэропорт»;
36. «Библиотека»;
37. «Кинотеатр»;
38. «Поликлиника»;
39. «Автосалон»;
40. «Таксопарк».

7. Контрольные вопросы

13. Каково назначение диаграммы состояний?
14. Назовите основные элементы диаграммы состояний.
15. Как создать диаграмму состояний в VISIO?
16. В чем отличие диаграммы классов и состояний?

Практическая работа №9

Построение диаграмм деятельности на языке UML с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания диаграмм деятельности на языке UML, получение навыков построения диаграмм деятельности, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения диаграмм деятельности на языке UML;

– ознакомиться с теоретическими вопросами построения диаграмм деятельности с помощью MS Visio.

3. Краткие теоретические сведения

При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Традиционно для этой цели использовались блок-схемы или структурные схемы алгоритмов. Каждая такая схема акцентирует внимание на последовательности выполнения определенных действий или элементарных операций, которые в совокупности приводят к получению желаемого результата.

Алгоритмические и логические операции, требующие выполнения в определенной последовательности, окружают нас постоянно. Например, чтобы позвонить по телефону, нам предварительно нужно снять трубку или включить его. Для приготовления кофе или заваривания чая необходимо вначале вскипятить воду. Чтобы выполнить ремонт двигателя автомобиля, требуется осуществить целый ряд нетривиальных операций, таких как разборка силового агрегата, снятие генератора и некоторых других.

С увеличением сложности системы строгое соблюдение последовательности выполняемых операций приобретает все более важное значение. Если попытаться заварить кофе холодной водой, то мы можем только испортить одну порцию напитка. Нарушение последовательности операций при ремонте двигателя может привести к его поломке или выходу из строя. Еще более катастрофические последствия могут произойти в случае отклонения от установленной последовательности действий при взлете или посадке авиалайнера, запуске ракеты, регламентных работах на АЭС.

Для моделирования процесса выполнения операций в языке UML используются так называемые **диаграммы деятельности**. Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов. Отличие заключается в семантике состояний, которые используются для представления не деятельностей, а действий, и в отсутствии на переходах сигнатуры событий. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии. Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Таким образом, диаграммы деятельности можно считать частным случаем диаграмм состояний. Основным направлением использования диаграмм деятельности является визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. При этом каждое состояние может являться выполнением операции некоторого класса либо ее части, позволяя использовать диаграммы деятельности для описания реакций на внутренние события системы.

В контексте языка UML **деятельность (activity)** представляет собой некоторую совокупность отдельных вычислений, выполняемых автоматом. При этом отдельные элементарные вычисления могут приводить к некоторому результату или действию (action). На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Сам же результат может привести к изменению состояния системы или возвращению некоторого значения.

Состояние действия и деятельности

Состояние деятельности (activity state) – состояние в графе деятельности, которое служит для представления процедурной последовательности действий, требующих определенного времени. Переход из состояния деятельности происходит после выполнения специфицированной в нем ду-деятельности, при этом ключевое слово do в имени деятельности не указывается. Состояние деятельности не может иметь внутренних переходов, поскольку оно является элементарным.

Состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время.

Состояние деятельности можно представлять себе, как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

Состояние действия (action state) является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения алгоритма (процедуры) или потока управления.

Графически состояние действия изображается фигурой, напоминающей прямоугольник, боковые стороны которого заменены выпуклыми дугами (рис. 1). Внутри этой фигуры записывается выражение действия (action-expression), которое должно быть уникальным в пределах одной диаграммы деятельности.

Рисунок 132 – Графическое обозначение состояния действия

Действие может быть записано на естественном языке, некотором псевдокоде или языке программирования. Никаких дополнительных или неявных ограничений при записи действий не накладывается. Рекомендуется в качестве имени простого действия использовать глагол с пояснительными словами (рис. 1а). Если же действие может быть представлено в некотором формальном виде, то целесообразно записать его на том языке программирования, на котором предполагается реализовывать конкретный проект (рис. 1б).

Иногда возникает необходимость представить на диаграмме деятельности некоторое сложное действие, которое, в свою очередь, состоит из нескольких более простых действий. В этом случае можно использовать специальное обозначение так называемого **состояния поддеятельности (subactivity state).** Такое состояние является графом деятельности и обозначается специальной пиктограммой в правом нижнем углу символа состояния действия (рис. 2). Эта конструкция может применяться к любому элементу языка UML, который поддерживает «вложенность» своей структуры. При этом пиктограмма может быть дополнительно помечена типом вложенной структуры.

Рисунок 133 – Графическое обозначение состояния поддеятельности

Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния. Они имеют такие же обозначения, как и на диаграмме состояний (см. практическое занятия №9). При этом каждая деятельность начинается в начальном состоянии и заканчивается в конечном состоянии. Саму диаграмму деятельности принято располагать таким образом, чтобы действия следовали сверху вниз. В этом случае начальное состояние будет изображаться в верхней части диаграммы, а конечное – в ее нижней части.

Переходы

При построении диаграммы деятельности используются только такие переходы, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия. Этот переход переводит деятельность в последующее состояние сразу, как только закончится действие в предыдущем состоянии. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит единственный переход, то он может быть никак не помечен. Если же таких переходов несколько, то сработать может только один из них. Именно в этом случае для каждого из таких переходов должно быть явно записано сторожевое условие в прямых скобках. При этом для всех выходящих из некоторого состояния переходов должно выполняться требование истинности только одного из них. Подобный случай встречается тогда, когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви в зависимости от значения некоторого промежуточного результата. Такая ситуация получила название ветвления, а для ее обозначения применяется специальный символ.

Графически ветвление на диаграмме деятельности обозначается небольшим ромбом, внутри которого нет никакого текста (рис. 3). В этот ромб может входить только одна стрелка от того состояния действия, после выполнения которого поток управления должен быть продолжен по одной из взаимно исключающих ветвей. Принято входящую стрелку присоединять к верхней или левой вершине символа ветвления. Выходящих стрелок может быть две или более, но для каждой из них явно указывается соответствующее сторожевое условие в форме булевского выражения.

В качестве **примера** рассмотрим фрагмент известного алгоритма нахождения корней квадратного уравнения. В общем случае после приведения уравнения второй степени к каноническому виду: $a * x * x + b * x + c = 0$ необходимо вычислить его дискриминант. Причем, в случае отрицательного дискриминанта уравнение не имеет решения на множестве действительных чисел, и дальнейшие вычисления должны быть прекращены. При неотрицательном дискриминанте уравнение имеет решение, корни которого могут быть получены на основе конкретной расчетной формулы.

Графически фрагмент процедуры вычисления корней квадратного уравнения может быть представлен в виде диаграммы деятельности с тремя состояниями действия и ветвлением (рис. 3). Каждый из переходов, выходящих из состояния «Вычислить дискриминант», имеет сторожевое условие, определяющее единственную ветвь, по которой может быть продолжен процесс вычисления корней в зависимости от знака дискриминанта. Очевидно, что в случае его отрицательности, мы сразу попадаем в конечное состояние, тем самым завершая выполнение алгоритма в целом.

Рисунок 134 – Фрагмент диаграммы деятельности для алгоритма нахождения корней квадратного уравнения

В рассмотренном примере, как видно из рис. 3, выполняемые действия соединяются в конечном состоянии. Однако это вовсе не является обязательным. Можно изобразить еще один символ ветвления, который будет иметь несколько входящих переходов и один выходящий.

Один из наиболее значимых недостатков обычных блок-схем или структурных схем алгоритмов связан с проблемой изображения параллельных ветвей отдельных вычислений. Поскольку распараллеливание вычислений существенно повышает общее быстродействие программных систем, необходимы графические примитивы для представления параллельных процессов. В языке UML для этой цели используется специальный символ для разделения и слияния параллельных вычислений или потоков управления. Таким символом является **прямая черточка.**

Такая черточка изображается отрезком горизонтальной линии, толщина которой несколько шире основных сплошных линий диаграммы деятельности. При этом разделение (concurrent fork) имеет один входящий переход и несколько выходящих (рис. 4а). Слияние (concurrent join), наоборот, имеет несколько входящих переходов и один выходящий (рис. 4б).

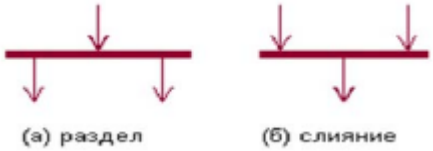


Рисунок 135 – Графическое обозначение разделения и слияния параллельных потоков управления

Для иллюстрации особенностей параллельных процессов выполнения действий рассмотрим **пример** с приготовлением напитка. Достоинство этого примера состоит в том, что он практически не требует никаких дополнительных пояснений в силу своей очевидности (рис. 5).

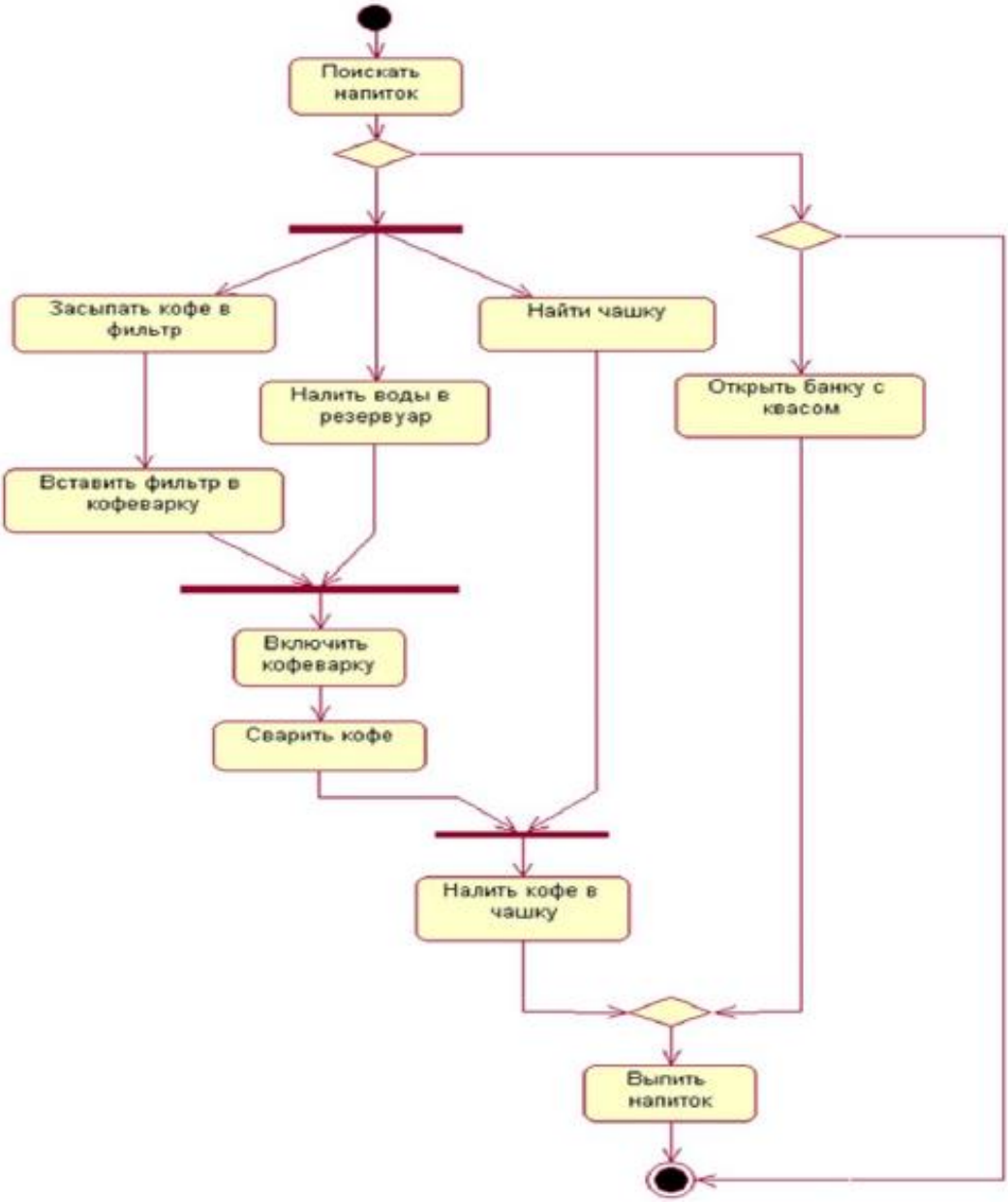


Рисунок 136 – Диаграмма деятельности для примера с приготовлением напитка

Хотя диаграмма деятельности предназначена для моделирования поведения систем, время в явном виде отсутствует на этой диаграмме. Ситуация здесь во многом аналогична диаграмме состояний.

Таким образом, диаграмма деятельности есть не что иное, как специальный случай диаграммы состояний, в которой все или большинство состояний являются действиями или состояниями поддеятельности. А все или большинство переходов являются переходами, которые срабатывают по завершении действий или под-деятельностей в состояниях источниках.

Дорожки

Диаграммы деятельности могут быть использованы не только для спецификации алгоритмов вычислений или потоков управления в программных системах. Не менее важная область их применения связана с моделированием бизнес-процессов. Действительно, деятельность любой компании (фирмы) также представляет собой не что иное, как совокупность отдельных действий, направленных на достижение требуемого результата. Однако применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несет ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому.

Для моделирования этих особенностей в языке UML используется специальная конструкция, получившее название ***дорожки (swimlanes)***. Имеется в виду визуальная аналогия с плавательными дорожками в бассейне, если смотреть на соответствующую диаграмму. При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением (отделом, группой, отделением, филиалом) компании (рис. 6).

Рисунок 137 – Вариант диаграммы деятельности с дорожками

Названия подразделений явно указываются в верхней части дорожки. Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение компании. Порядок следования дорожек не несет какой-либо семантической информации и определяется соображениями удобства.

В качестве примера рассмотрим фрагмент диаграммы деятельности торговой компании, обслуживающей клиентов по телефону. Подразделениями компании являются отдел приема и оформления заказов, отдел продаж и склад.

Этим подразделениям будут соответствовать три дорожки на диаграмме деятельности, каждая из которых специфицирует зону ответственности подразделения. В данном случае диаграмма деятельности заключает в себе не только информацию о последовательности выполнения рабочих действий, но и о том, какое из подразделений торговой компании должно выполнять то или иное действие (рис. 7).

Из указанной диаграммы деятельности сразу видно, что после принятия заказа от клиента отделом приема и оформления заказов осуществляется распараллеливание деятельности на два потока (переход-разделение). Первый из них остается в этом же отделе и связан с получением оплаты от клиента за заказанный товар. Второй инициирует выполнение действия по подбору товара в отделе продаж (модель товара, размеры, цвет, год выпуска и пр.). По окончании этой работы инициируется действие по отпуску товара со склада. Однако подготовка товара к отправке в торговом отделе начинается только после того, как будет получена оплата за товар от клиента и товар будет отпущен со склада (переход-соединение). Только после этого товар отправляется клиенту, переходя в его собственность.

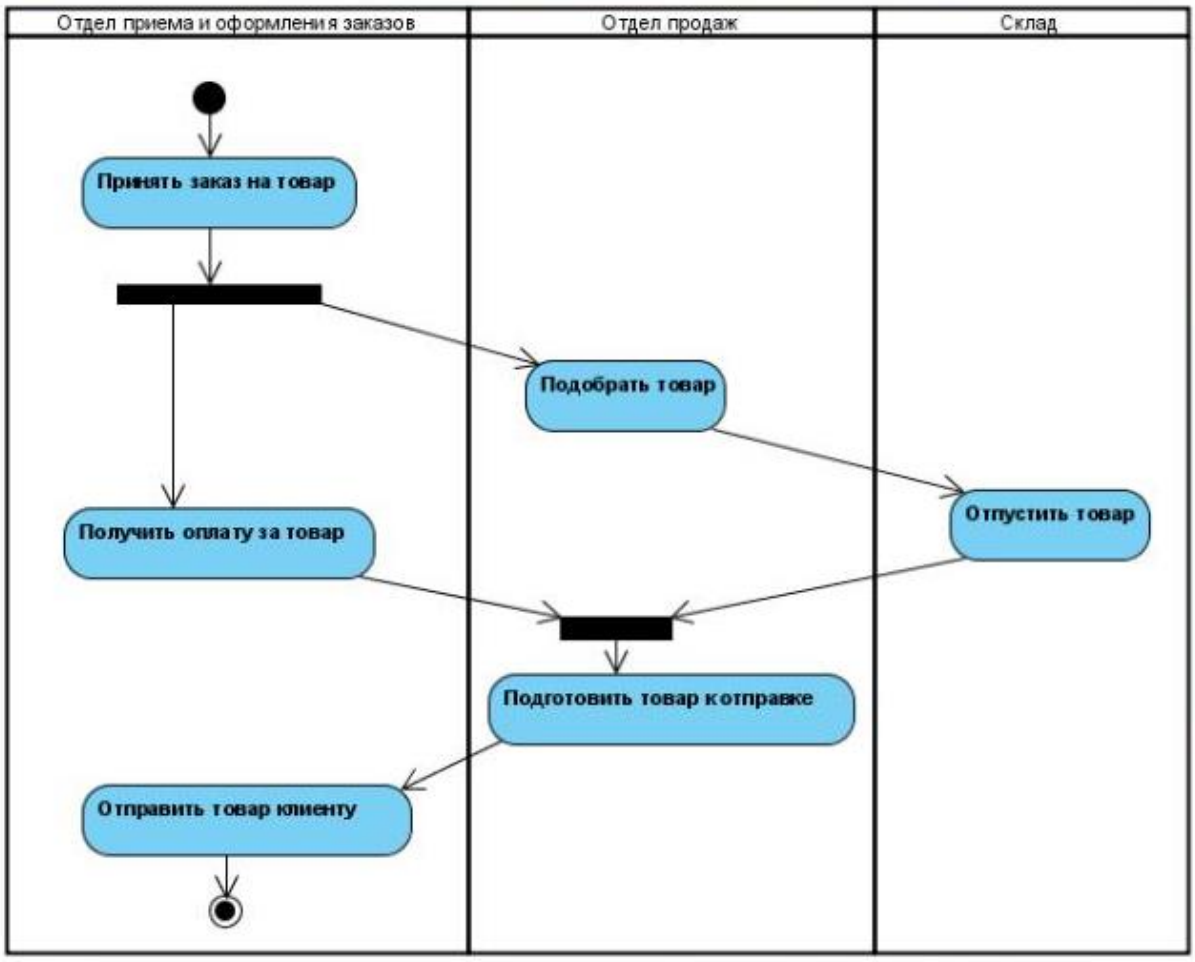


Рисунок 138 – Фрагмент диаграммы деятельности для торговой компании

Объекты

В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий. При этом действия специфицируют вызовы, которые передаются от одного объекта графа деятельности к другому. Поскольку в таком ракурсе объекты играют определенную роль в понимании процесса деятельности, иногда возникает необходимость явно указать их на диаграмме деятельности.

Для **графического представления объектов** используются прямоугольник класса, с тем отличием, что имя объекта подчеркивается. Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой. Соответствующая зависимость определяет состояние конкретного объекта после выполнения предшествующего действия.

На диаграмме деятельности с дорожками расположение объекта может иметь некоторый дополнительный смысл. А именно, если объект расположен на границе двух дорожек, то это может означать, что переход к следующему состоянию действия в соседней дорожке ассоциирован с готовностью некоторого документа (объект в некотором состоянии). Если же объект целиком расположен внутри дорожки, то и состояние этого объекта целиком определяется действиями данной дорожки.

Возвращаясь к предыдущему примеру с торговой компанией, можно заметить, что центральным объектом процесса продажи является заказ или вернее состояние его выполнения. Вначале до звонка от клиента заказ как объект отсутствует и возникает лишь после такого звонка. Однако этот заказ еще не заполнен до конца, поскольку требуется еще подобрать конкретный товар в отделе продаж. После его подготовки он передается на склад, где вместе с отпуском товара заказ окончательно дооформляется. Наконец, после получения подтверждения об оплате товара эта информация заносится в заказ, и он считается выполненным и закрытым. Данная информация может быть представлена графически в виде модифицированного варианта диаграммы деятельности этой же торговой компании (рис. 8).

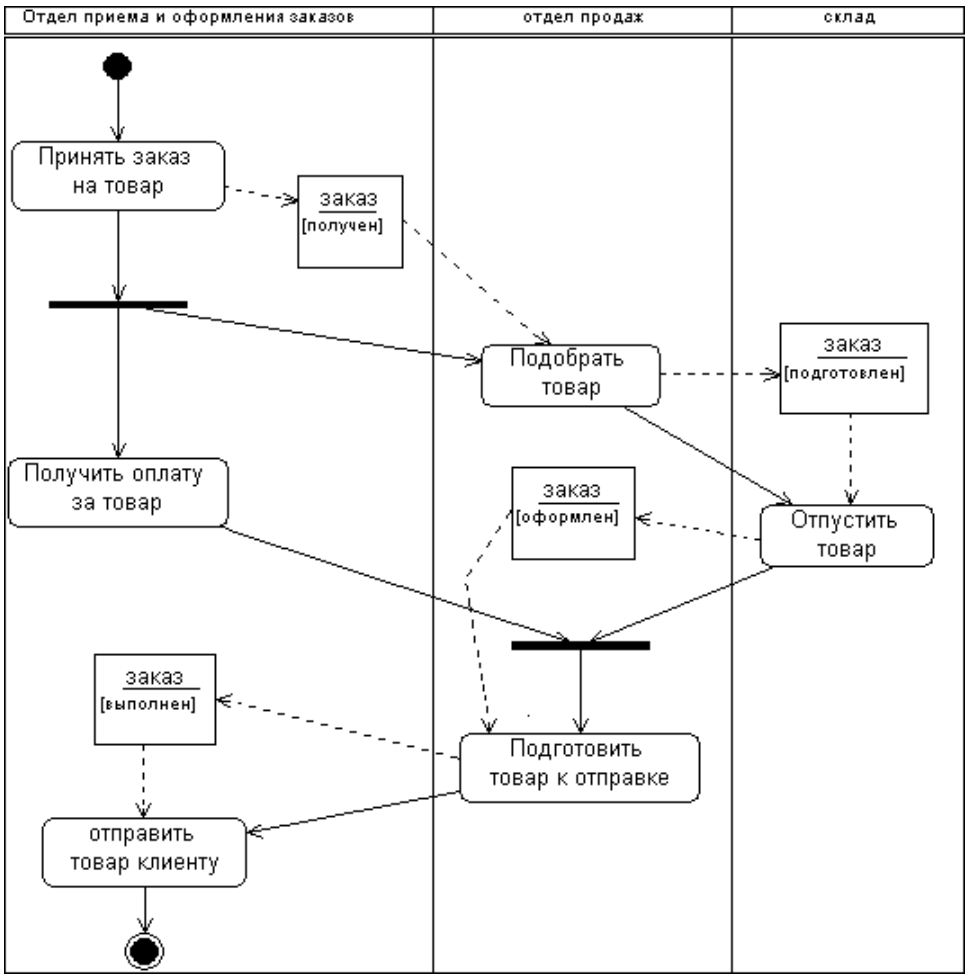


Рисунок 139 – Фрагмент диаграммы деятельности торговой компании с объектом-заказом

4. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

26. Запустите MS Visio.
27. Откройте файл, созданный в практических занятиях №8-9 и содержащий диаграмму классов.
28. В проводнике по моделям должны отображаться все классы и диаграммы, созданные ранее (рис. 9).

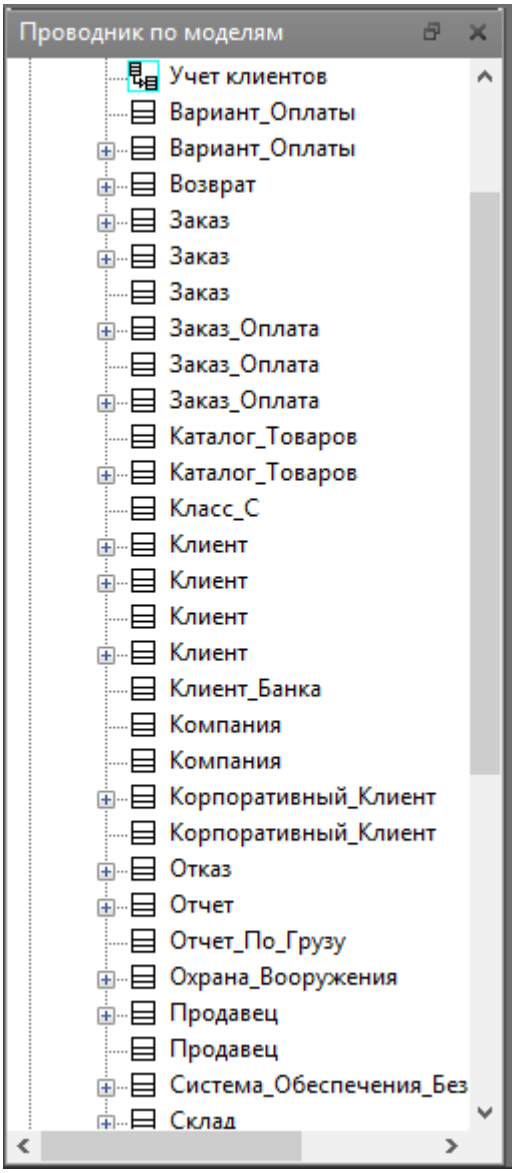


Рисунок 140 – Проводник по моделям

Опишем с помощью диаграммы деятельности процесс формирования заказа и выдачу товара. В бизнес-процессе участвуют 3 действующих лица: клиент, продавец и система оплаты. Следовательно, необходимо добавить 3 дорожки для распределения ответственности между этими лицами.

Для этого, в файле с диаграммой классов, созданной в практическом занятии 8, необходимо проделать следующие действия:

6. Щелкнуть правой кнопкой мыши по *классу Заказ*.
7. В контекстном меню выбрать пункт *Схемы*.
8. Нажать кнопку *Создать* и выбрать *Деятельность*.
9. Переименовать созданный лист в *Деятельность-Заказ*.
10. Построить диаграмму деятельности для класса Заказ. Для это выполните действия, описанные ниже.

a. Добавить 3 элемента *Дорожка* и изменить их названия на *Клиент*, *Продавец* и *Система оплаты* соответственно.

b. Добавить элементы *Начальное состояние, Конечное состояние, Состояние действия, Решение, Переход (объединение),* изменить их названия и задать расположение в соответствии с рисунком 10.

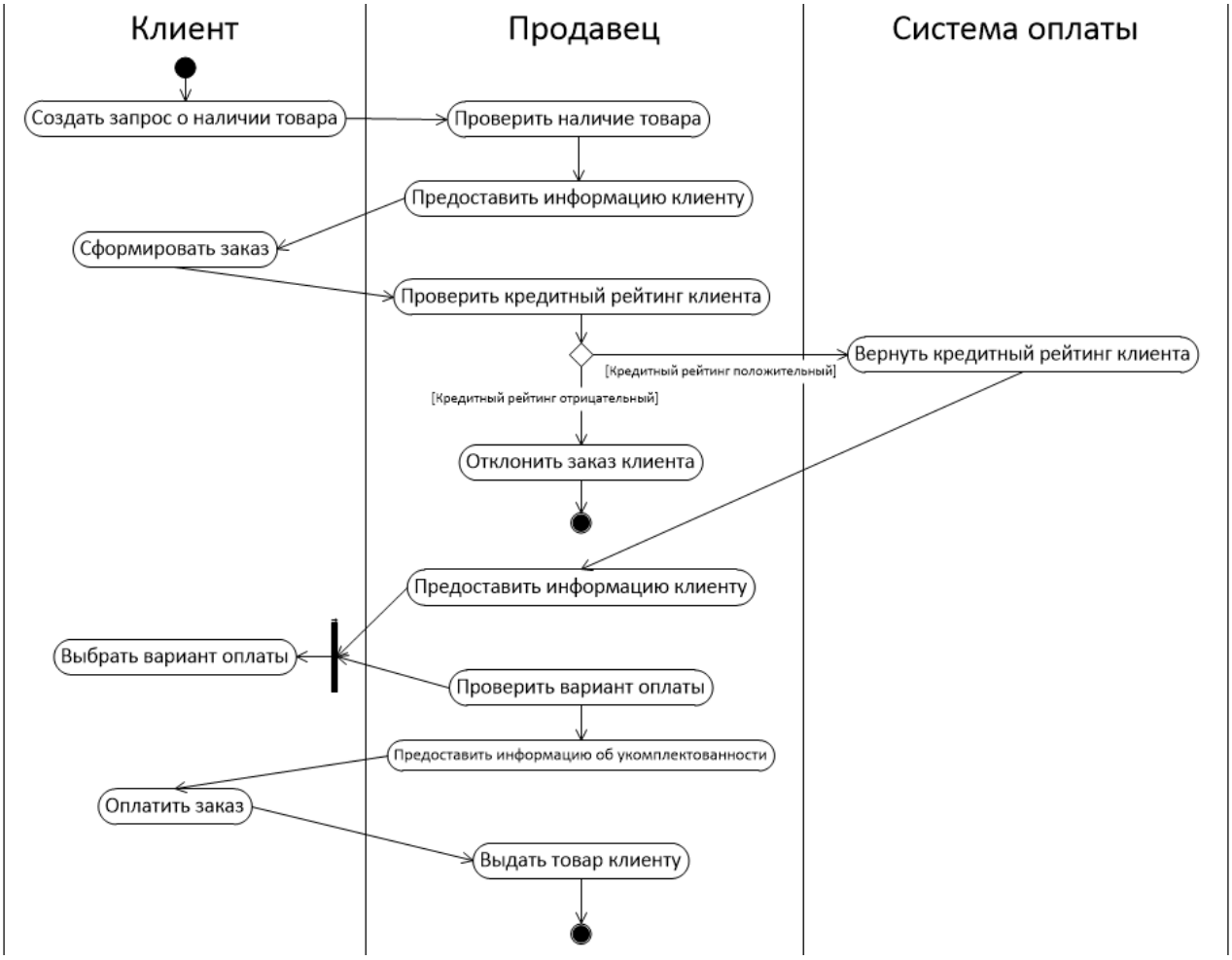


Рисунок 141 – Диаграмма деятельности

5. Задание

Построить диаграмму деятельности в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Создать диаграммы деятельности не менее чем для трех классов, описанных в практическом занятии №8.

6. Варианты

41. «Отдел кадров»;
42. «Агентство аренды»;
43. «Аптека»;
44. «Ателье»;
45. «Аэропорт»;
46. «Библиотека»;
47. «Кинотеатр»;
48. «Поликлиника»;
49. «Автосалон»;
50. «Таксопарк».

7. Контрольные вопросы

17. Дайте определение понятию «диаграмма деятельности».
18. Опишите назначение диаграммы деятельности.
19. Дайте определение понятиям «состояние деятельности» и «состояние действия». Графическое изображение состояния.
20. Приведите пример ветвления и параллельных потоков управления процессами на диаграмме деятельности.

21. Какие переходы используются на диаграмме деятельности?
22. Что представляет собой дорожка на диаграмме деятельности?
23. Как графически изображаются объекты на диаграмме деятельности?

Практическая работа №10

Построение диаграмм последовательностей на языке UML с помощью MS Visio

1. Цель занятия

Целью занятия является изучение основ создания диаграмм последовательностей на языке UML, получение навыков построения диаграмм последовательностей, применение приобретенных навыков для построения объектно-ориентированных моделей определенной предметной области.

2. Задачи

Основными задачами занятия являются:

– ознакомиться с теоретическими вопросами построения диаграмм последовательностей на языке UML;

– ознакомиться с теоретическими вопросами построения диаграмм последовательностей с помощью MS Visio.

3. Краткие теоретические сведения

Диаграммы последовательностей описывают взаимодействия множества объектов, включая сообщения, которыми они обмениваются.

В отличие от диаграммы классов, на которой изображаются абстрактные элементы в виде классов, на диаграмме последовательностей используются конкретные экземпляры классов – ***объекты***. Объекты отображаются прямоугольником без полей. Для того чтобы подчеркнуть, что это экземпляр абстрактной сущности, название объекта подчеркивается. При необходимости через двоеточие после названия можно указать сущность (класс) экземпляром которой является этот объект. Отметим, что объект может быть экземпляром не только класса, но и других абстракций, например, актера. Обратите внимание, что при указании в качестве классификатора актера изменится графическое обозначение объекта (рис. 1).

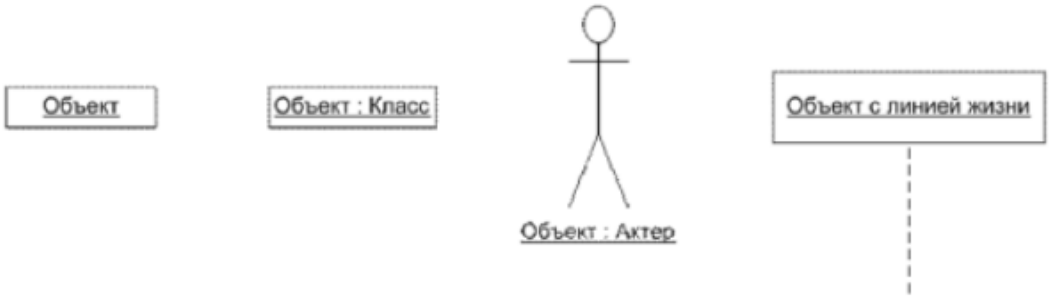


Рисунок 142 – Графическое обозначение объекта UML

На диаграмме последовательностей у объекта может присутствовать ***линия жизни***, на которой отмечаются происходящие с объектом события. Линия жизни отображается пунктирной линией (рис. 2).

Между собой объекты могут быть связаны связями. ***Связь*** – это экземпляр отношения ассоциация, и имеет такое же графическое обозначение, что и ассоциация.

На диаграмме последовательностей объекты обмениваются сообщениями. ***Сообщение*** – это спецификация передачи данных от одного объекта другому, который предполагает какое-то ответное действие. Графически сообщение обозначается сплошной линией со стрелкой.

Часто операция вызывает какую-либо операцию в объекте. Очевидно, что класс, экземпляром которого является объект, должен иметь такую операцию. Привязка сообщения к операции класса объекта выполняется в свойствах сообщения (рис. 2).

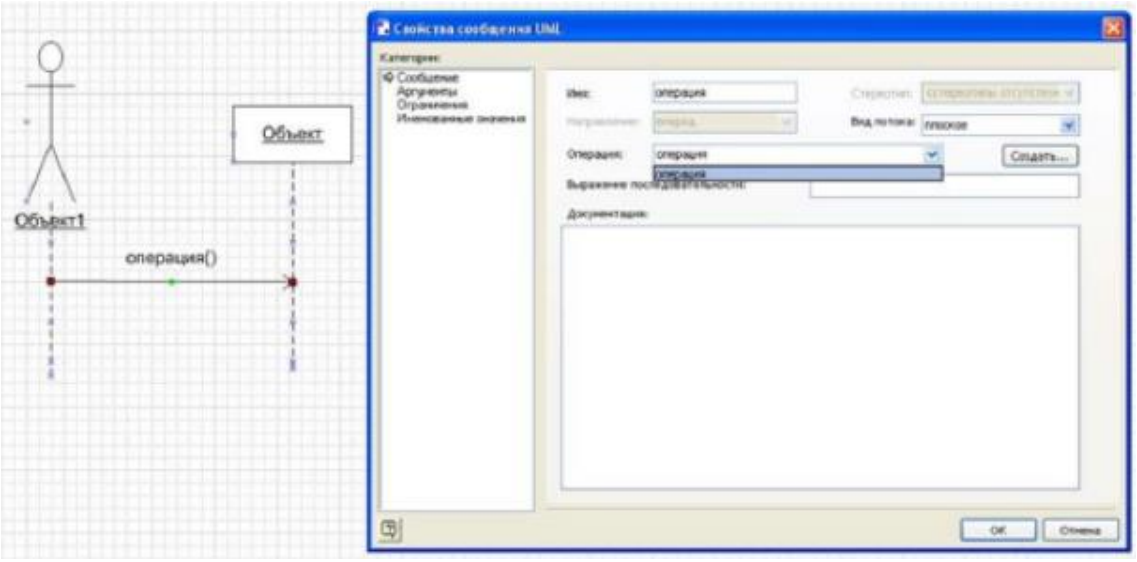


Рисунок 143 – Сообщение UML и его свойства

При построении диаграммы классов обычно определяются только основные свойства сущностей, а такие детали, как операции, удобно создавать при построении диаграммы последовательности, для чего в свойствах сообщения UML есть кнопка создания операции.

Диаграммы последовательностей, как и другие диаграммы для отображения динамических свойств системы, могут быть выполнены в контексте многих сущностей UML. Они могут описывать поведение системы в целом, подсистемы, класса или операции класса и др. К сожалению, Visio недостаточно гибка в плане поддержки раскрытия содержания отдельных элементов с помощью других диаграмм. Например, кликнув правой кнопкой мыши по классу можно обнаружить, что для его описания можно создать лишь диаграммы классов, состояний и деятельности. Поэтому возможность привязать диаграмму последовательностей к элементу, который она реализует, средствами Visio невозможно, эту связь нужно подразумевать.

Диаграммы последовательностей будем делать в контексте прецедентов с диаграммы прецедентов, реализуя те функции, которые должна выполнять наша система.

При построении динамических диаграмм используется уже разработанная структура информационной системы. Для диаграммы последовательностей не нужно придумывать объекты, а достаточно определить, экземпляры каких классов участвуют в этом действии.

Определив необходимые объекты (как экземпляры классов, так и экземпляры актеров), вторым этапом построения диаграмм последовательностей определяются сообщения, пересылаемые между актерами. Фактически определяется последовательность шагов, для выполнения нужного действия.

4. Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

1. Запустите MS Visio.
2. Откройте файл, созданный в практических занятиях №8-10 и содержащий диаграмму классов.
3. В проводнике по моделям должны отображаться все классы и диаграммы, созданные ранее.

Построим диаграмму последовательности для варианта использования «Обеспечить покупателя информацией» (рис. 4). Для этого добавим на диаграмму последовательности линии жизни и соотнесем объекты с актерами, инициирующими вариант использования «Обеспечить покупателя информацией», и с необходимыми классами.

Для **добавления диаграммы последовательности в проект MS Visio** выполните следующие действия:

1. В проводнике по моделям найдите ветку «Основной пакет».
2. Нажмите по ней правой кнопкой мыши > Создать …
3. В контекстном меню выберите пункт «Схема последовательностей».

Добавим сообщения, которыми обмениваются объекты для исполнения варианта использования.

Если объект имеет операцию (посмотреть в практическом занятии №8 «Диаграммы классов» наличие операции у класса, которому принадлежит объект).

1. Из группы фигур «Последовательности UML» добавить три фигуры типа «Линия жизни объекта». Для изменения названия необходимо дважды щелкнуть левой кнопкой мыши по фигуре. Откроется окно свойств объекта и, если в данном файле нет ранее созданных классов, окно создания нового класса. В данном примере необходимо создать три класса «Товар», «Каталог товаров» и «Заказ» и соответственно три объекта с такими же названиями.
2. С помощью поиска фигур найти фигуру «Актер» и добавить ее в рабочую область. Двойным щелчком левой кнопки мыши задать имя «Клиент».
3. Добавить фигуру «Линия жизни» и соедините ее начало с фигурой «Клиент».
4. Протянуть все линии жизни вниз листа.
5. Добавить фигуры «Сообщение» и соединить, руководствуясь следующими принципами:

5.1. Соединить фигурой «Сообщение» линию жизни клиента с линией жизни объекта товар. Двойным щелчком по сообщению открыть окно свойств и выбрать операцию запросить товар.

5.2. Соединить фигурой «Сообщение» линию жизни клиента с линией жизни объекта заказ. Двойным щелчком по сообщению открыть окно свойств и выбрать операцию сформировать заказ.

5.3. Соединить фигурой «Сообщение» линию жизни объекта товар с линией жизни объекта каталог товаров. Двойным щелчком по сообщению открыть окно свойств и выбрать операцию проверить наличие.

5.4. Соединить фигурой «Сообщение (возврат)» линию жизни объекта товар и линию жизни клиента. Двойным щелчком по сообщению открыть окно свойств и задать текст сообщения «Предоставить информацию».

6. Добавить фигуры «Активация» и расположить их на диаграмме в соответствии с рисунком 4.

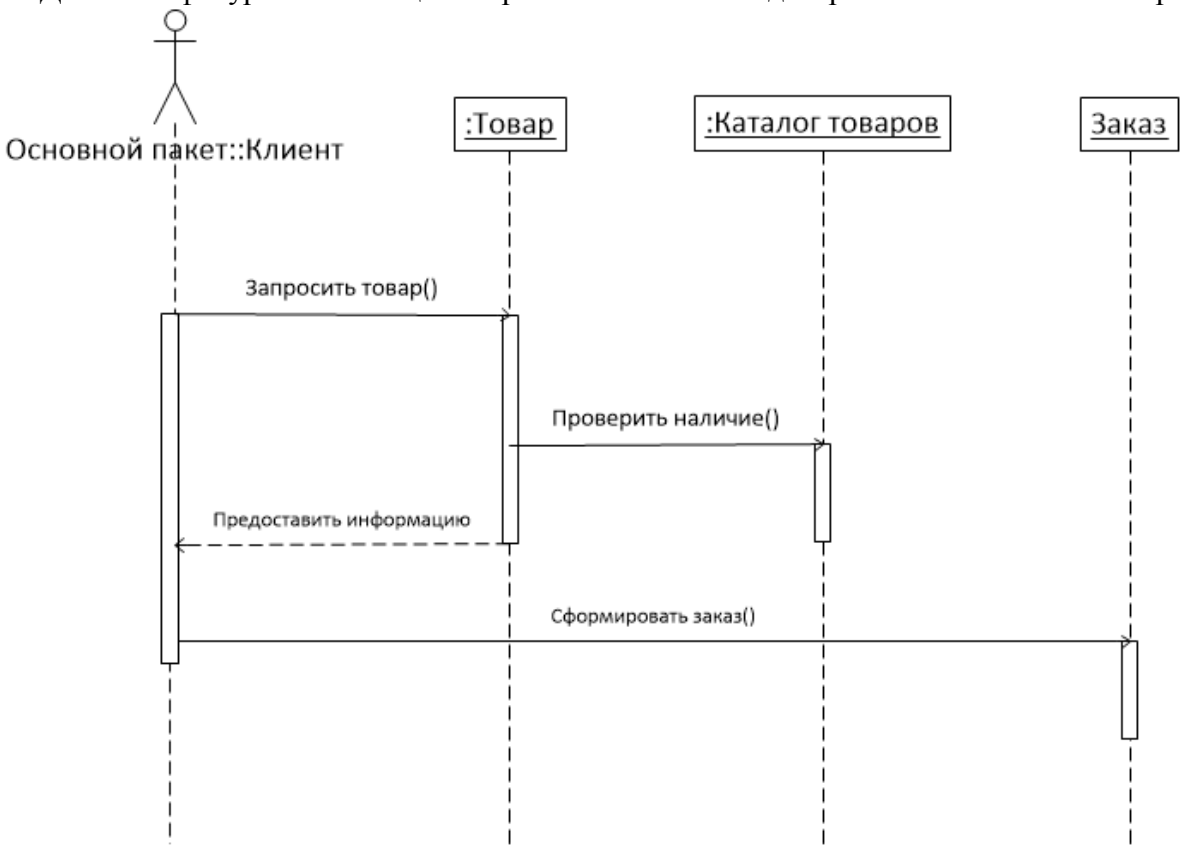


Рисунок 144 – Диаграмма последовательности для варианта использования «Обеспечить покупателя информацией»

При построении диаграмм последовательностей можно вносить коррективы в диаграмму классов. Если объект класса получает новую операцию, то она добавляется в соответствующий класс на диаграмме классов как метод.

Построим диаграмму последовательности для варианта использования «Согласовать условия оплаты» (рис. 5). Действия по построению диаграммы аналогичны построению диаграммы последовательности для варианта использования «Обеспечить покупателя информацией».

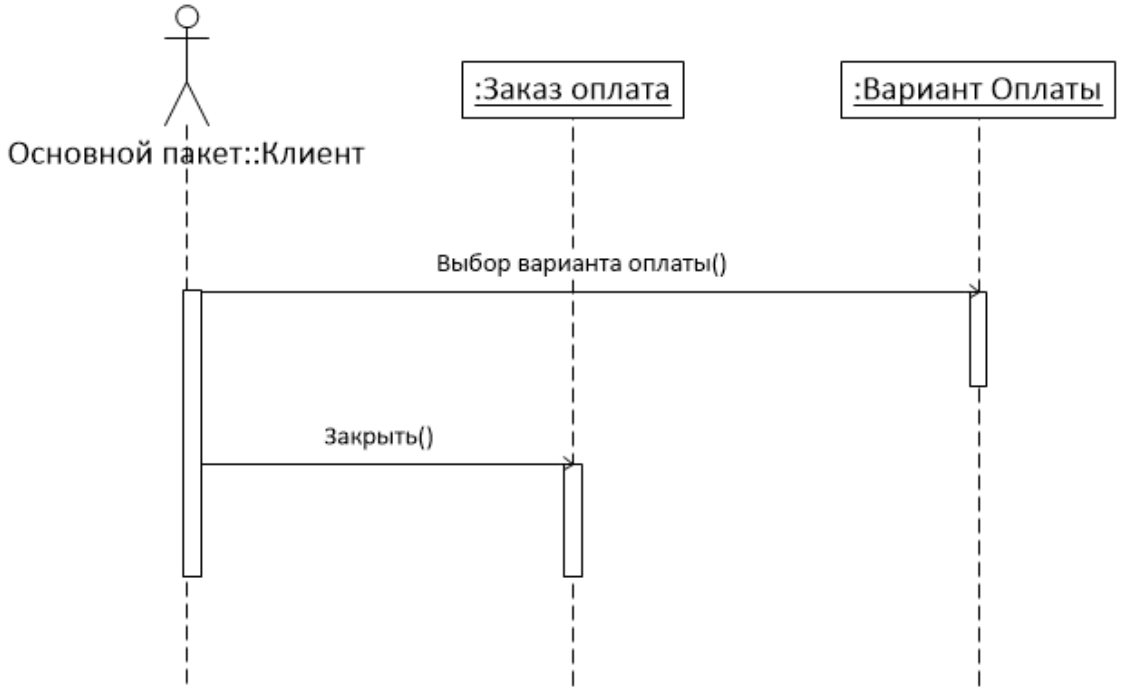


Рисунок 145 – Диаграмма последовательности для варианта использования «Согласовать условия оплаты»

Построим диаграмму последовательности для варианта использования «Заказать товар со склада» (рис. 6). Действия по построению диаграммы аналогичны построению предыдущих диаграмм последовательностей.

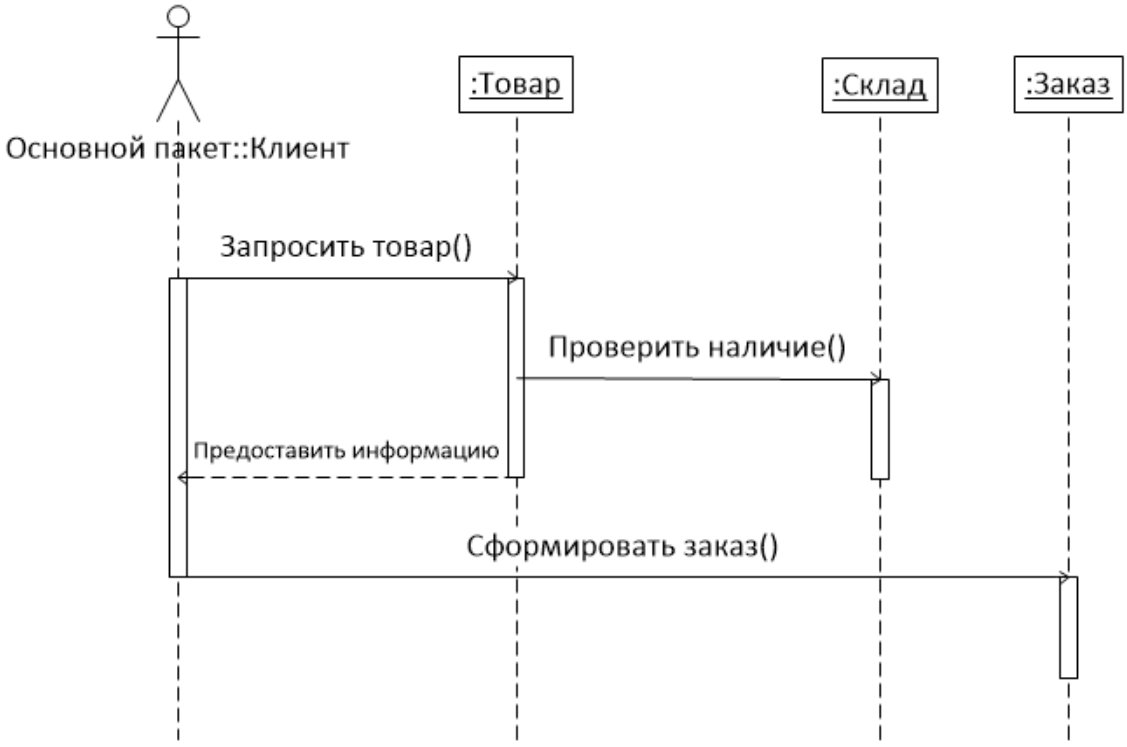


Рисунок 146 – Диаграмма последовательности для варианта использования «Заказать товар со склада»

Построим диаграмму последовательности для системы продажи товаров по каталогу (рис. 7).

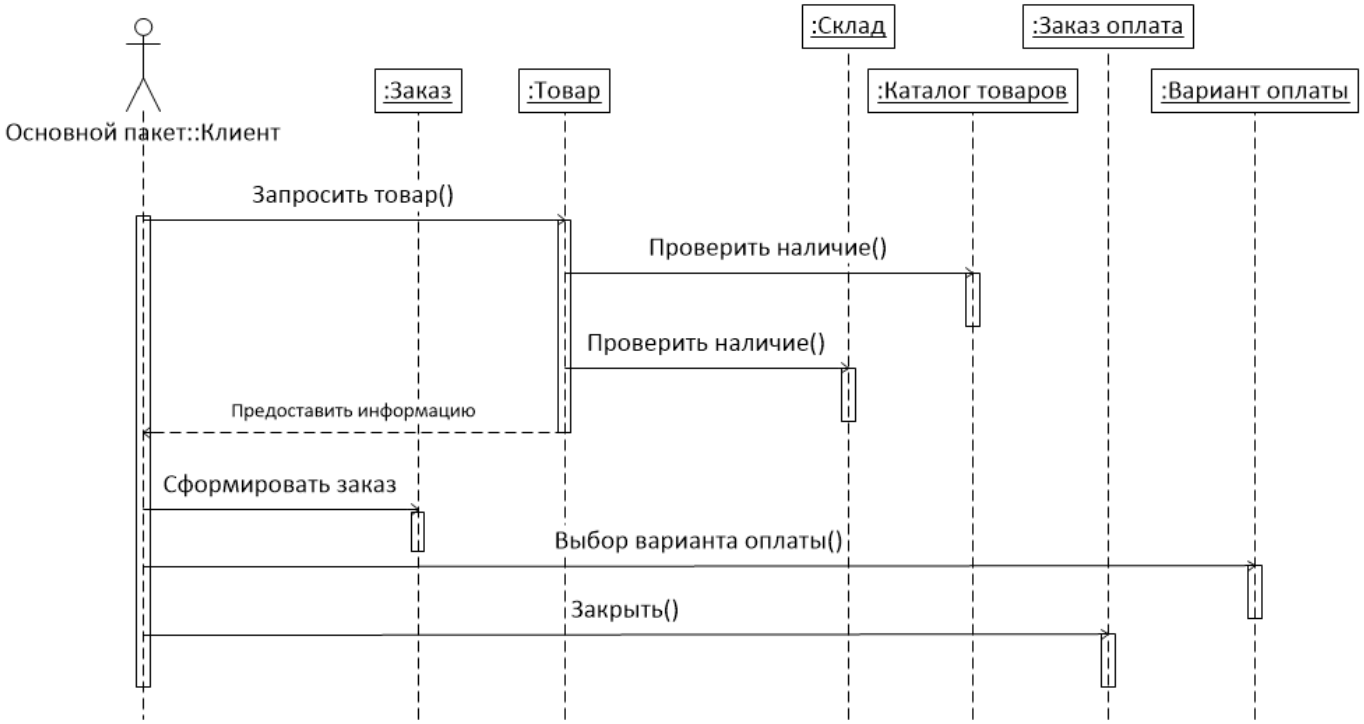


Рисунок 147 – Диаграмма последовательности для системы продажи товаров по каталогу

5. Задание

Построить диаграмму последовательности для каждого варианта использования, определенных в практическом занятии №7 и для всей системы в целом в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

6. Варианты

51. «Отдел кадров»;
52. «Агентство аренды»;
53. «Аптека»;
54. «Ателье»;
55. «Аэропорт»;
56. «Библиотека»;
57. «Кинотеатр»;
58. «Поликлиника»;
59. «Автосалон»;
60. «Таксопарк».

7. Контрольные вопросы

24. Для чего предназначена диаграмма последовательности?
25. Назовите и охарактеризуйте элементы диаграммы последовательности.
26. Что такое сообщение?
27. Что такое линия жизни?
28. Назовите виды сообщений.

Практическая работа №11
Знакомство с программой Cisco Packet Tracer

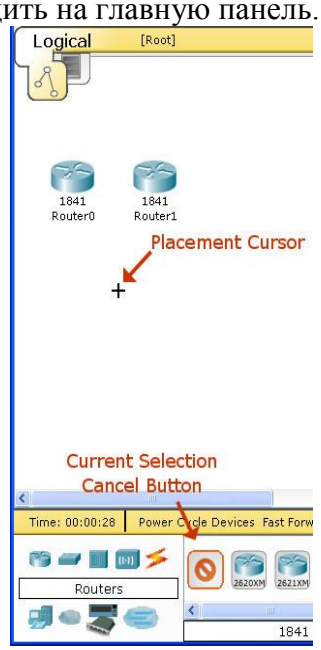
Цель работы: Изучение интерфейсной части программы симуляции сетей Cisco Packet Tracer.

1 ЗНАТЬ

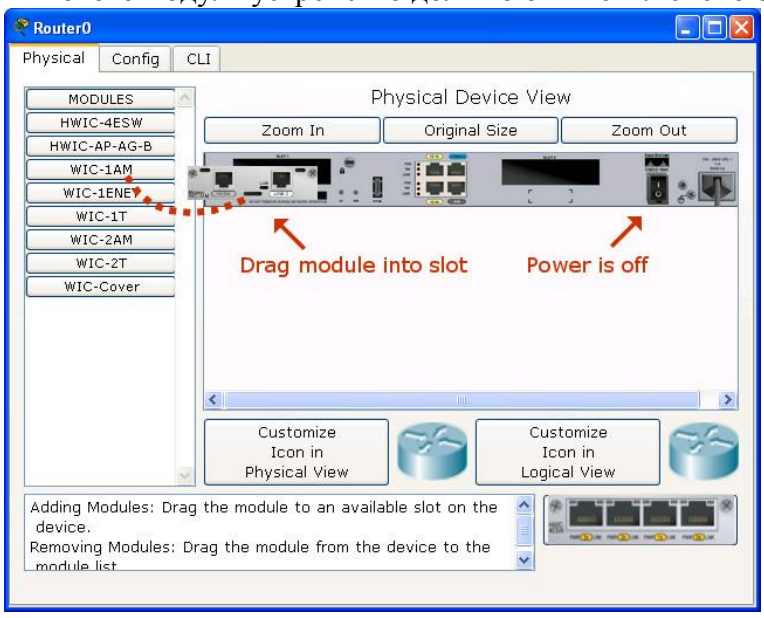
1. Что такое КС
2. Классификацию КС
3. Виды топологий КС

Cisco Packet Tracer – программа для моделирования сетей любой сложности. Позволяет использовать различные устройства: концентраторы (hub), коммутаторы (switch), маршрутизаторы (router), локальные станции, беспроводные устройства.

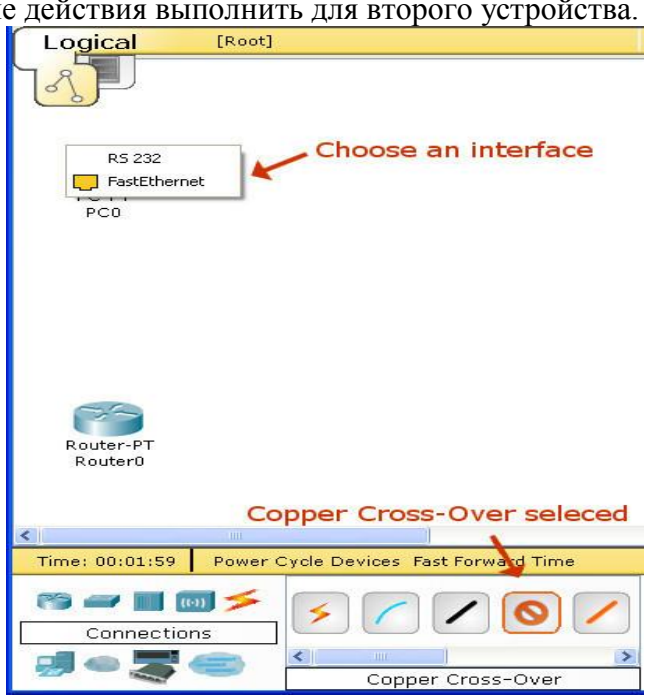
Логическое пространство Cisco Packet Tracer. Для того, чтобы расположить устройство, необходимо выбрать его из меню и перетащить на главную панель.



Большинство из устройств в Packet Tracer имеют модули расширения, необходимые для подключения дополнительных портов. Добавление модулей осуществляется в панели настройки устройства. При подключении нового модуля устройство должно быть отключено от электросети.

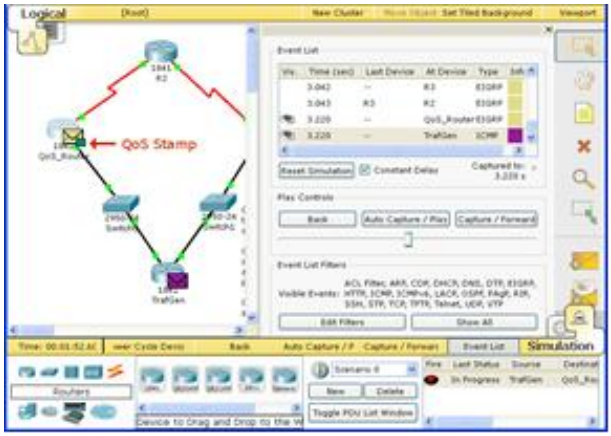


Для соединения устройств между собой необходимо выбрать подходящие кабели, расположенных на панели **Connections**. Затем нужно щелкнуть правой кнопкой мыши по одному из устройств и выбрать порт подключения. Аналогичные действия выполнить для второго устройства.



Режим реального времени. В режиме реального времени, сеть всегда работает независимо от действий пользователя. Конфигурирование сети осуществляется в реальном времени. Для настройки и диагностики сети можно использовать Simple PDU и Complex PDU для наглядной отправки пакета.

Режим симуляции. В режиме симуляции можно просмотреть работу сети в более медленном темпе, исследуя пути, по которым пересылаются пакеты. При переключении в режим симулирования, появляется специальная панель.



Во время моделирования можно кликнуть на пересылаемом пакете и получить о нем подробную информацию.

2 ЗАДАНИЕ

1. Изучите интерфейс программы Packet Tracer
2. Изучите доступные для построения сети оборудование и кабели
3. Добавьте несколько устройств в рабочую область
4. Попробуйте соединить, добавленные ранее устройства

3 ПРИМЕР ВЫПОЛНЕНИЯ

1. Запустите Packet Tracer

Основное окно программы показано на рисунке 1.

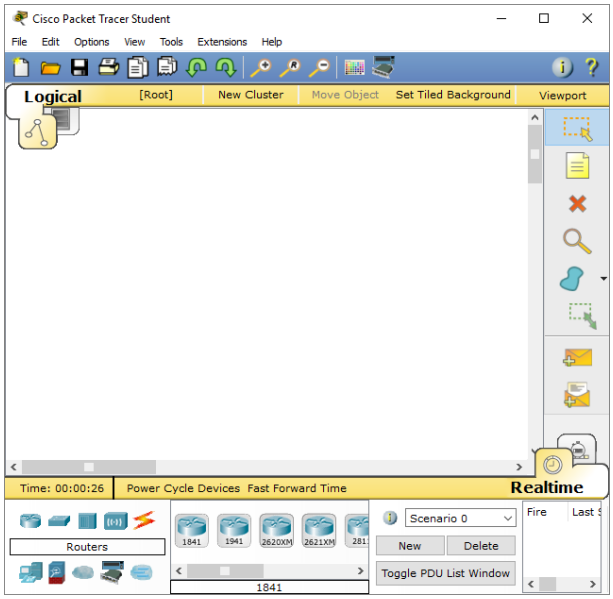


Рисунок 148 – Главное окно программы Cisco Packet Tracer

Панель инструментов. Панель инструментов, показанная на рисунке 2, с помощью пиктограмм дублирует основные пункты главного *меню* программы.

Рисунок 149 – Панель инструментов

Оборудование. Основная часть окна занята областью, в которой показана моделируемая сеть. Выбор типа объекта осуществляется в нижней области основного окна, показанной на рисунке 3. Справа от данной области расположены сами объекты сети.

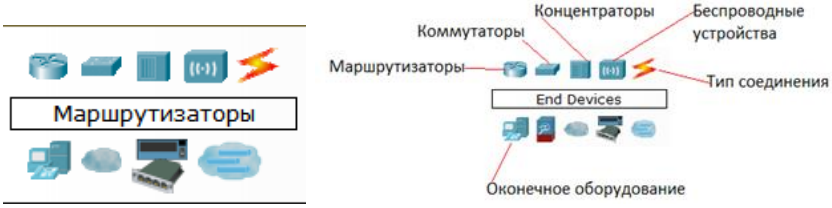


Рисунок 150 – Выбор типа элемента

2. Изучите доступные типы устройств.

Маршрутизаторы (Routers) – это сетевое устройство, пересылающее пакеты данных между различными сегментами сети. Маршрутизаторы работают на сетевом (третьем) уровне сетевой модели OSI.

Доступны маршрутизаторы различных серий, а также обобщенный (Generic) маршрутизатор, выполняющий общие для всех маршрутизаторов функции.

Коммутаторы (Switches) – сетевое устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного или нескольких сегментов сети. Коммутатор работает на канальном (втором) уровне модели OSI. Коммутатор передаёт данные только получателю.

Доступны управляемые коммутаторы разных серий.

Концентраторы (Hubs) – сетевое устройство для объединения компьютеров в сеть Ethernet с применением кабельной инфраструктуры типа витая пара. Концентратор работает на первом (физическом) уровне сетевой модели OSI, ретранслируя входящий сигнал с одного из портов в сигнал на все остальные (подключённые) порты, реализуя, таким образом, свойственную Ethernet топологию общая шина, с разделением пропускной способности сети между всеми устройствами.

Беспроводные устройства. Generic – обобщенная точка доступа, Linksys – беспроводной маршрутизатор с интегрированными службами Linksys.

Линии связи – различные виды соединений между устройствами, показанные на рисунке 4.

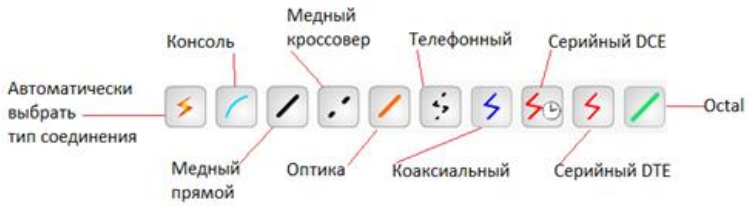


Рисунок 151 – Типы линий связи

Соединения:

1. Авто (auto) – автоматическое определение типа соединения (автоматически определяется наилучший способ соединения устройств),
2. Консоль (console) – соединение при помощи консольного кабеля (COM порт на ПК и вход Console на устройствах Cisco),
3. Медный прямой (copper straight-through)– соединение при помощи кабеля типа витая пара прямое,
4. Медный кроссовер (copper cross-over)– соединение при помощи кабеля типа витая пара перекрестное,
5. Оптика (fiber) – соединение при помощи волоконно-оптической линии связи (ВОЛС),
6. Телефонный (phone) – соединение при помощи телефонной линии,
7. Коаксиальный (coaxial) – соединение при помощи коаксиального кабеля,
8. Серийный DCE и DTE (serial DCE and DTE) – последовательные каналы связи.

Графическое меню

Рисунок 152 – Графическое меню (повернуто)

На этом рисунке слева направо:

Инструмент **Select** (Выбрать) можно активировать клавишей Esc. Он используется для выделения одного или более объектов для дальнейшего их перемещения, копирования или удаления.

Инструмент **Move Layout** (Переместить слой, горячая клавиша M) используется для прокрутки больших проектов сетей.

Инструмент **Place Note** (Сделать пометку, клавиша N) добавляет текст в рабочей области проекта.

Инструмент **Delete** (Удалить, клавиша Del) удаляет выделенный объект или группу объектов.

Инструмент **Inspect** (Проверка, клавиша I) позволяет, в зависимости от типа устройства, просматривать содержимое таблиц (ARP, NAT, таблицы маршрутизациии др.).

Инструмент **Drawapolygon** (Нарисовать многоугольник) позволяет рисовать прямоугольники, эллипсы, линии и закрашивать их цветом.

Инструмент **Resize Shape** (Изменить размер формы, комбинация клавиш Alt+R) предназначен для изменения размеров рисованных объектов (четырехугольников и окружностей).

3. Выберите раздел «коммутаторы» – устройство 2950T-24 (24 портовый Fast Ethernet коммутатор) и добавьте его на рабочее пространство, щёлкнув в нужном месте изменившим свой вид курсором. Теперь добавьте другие виды устройств на рабочее пространство.

4. Выберите на панели инструментов кнопку «создать пометку» поместите над установленным коммутатором его краткое описание (например, 24 портовый Fast Ethernet коммутатор).

5. Выберите инструмент «выбрать», выделите все созданное ранее и нажмите инструмент «Удалить», затем подтвердите это действие.

6. Установите курсор на любом устройстве. Программа высветит текущее состояние портов на предмет их активности, вкратце их текущее состояние и физическое размещение в физической модели.

3 ВОПРОСЫ ДЛЯ ЗАЩИТЫ РАБОТЫ

1. Что такое компьютерная сеть?
2. Какие топологии вы знаете?
3. Охарактеризуйте топологию «звезда» и «шина»

4. Что такое концентратор?
5. Что такое коммутатор?
6. Что такое роутер?
7. Для соединения каких устройств используется прямой кабель?
8. Для соединения каких устройств используется кроссовый кабель?

Практическая работа №12

Настройка сети с двумя маршрутизаторами

Цель работы: Провести настройку сетевого коммутатора с использованием Cisco Packet Tracer.

Изучить требования к адресному пространству подсети; сконфигурировать интерфейсы Serial и Fast Ethernet; протестировать и проверить конфигурацию.

Часть 1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Коммутирующие концентраторы (Switched Hubs) или, как их еще называют, коммутаторы (Switches), переключатели и свичи, могут рассматриваться, как простейший и очень быстрый мост. Они позволяют разделить единую сеть на несколько сегментов для увеличения допустимого размера сети или с целью снижения нагрузки (трафика) в отдельных частях сети.

Как уже отмечалось, в отличие от мостов, коммутирующие концентраторы не принимают приходящие пакеты, а только переправляют из одной части сети в другую те пакеты, которые в этом нуждаются. Они в реальном темпе поступления битов пакета распознают адрес приемника пакета и принимают решение о том, надо ли этот пакет переправлять, и, если надо, то кому. Никакой обработки пакетов не производится, хотя и контролируется их заголовок. Коммутаторы практически не замедляют обмена по сети. Но они не могут преобразовывать формат пакетов и протоколов обмена по сети. Поскольку коммутаторы работают с информацией, находящейся внутри кадра, часто говорят, что они ретранслируют кадры, а не пакеты, как репитерные концентраторы.

Коллизии коммутатором не ретранслируются, что выгодно отличает его от более простого репитерного концентратора. Можно сказать, что коммутаторы производят более глубокое разделение сети, чем концентраторы. Они разделяют на части зону коллизий (Collision Domain) сети, то есть область сети, на которую распространяются коллизии.

Логическая структура коммутатора довольно проста (рис. 1).

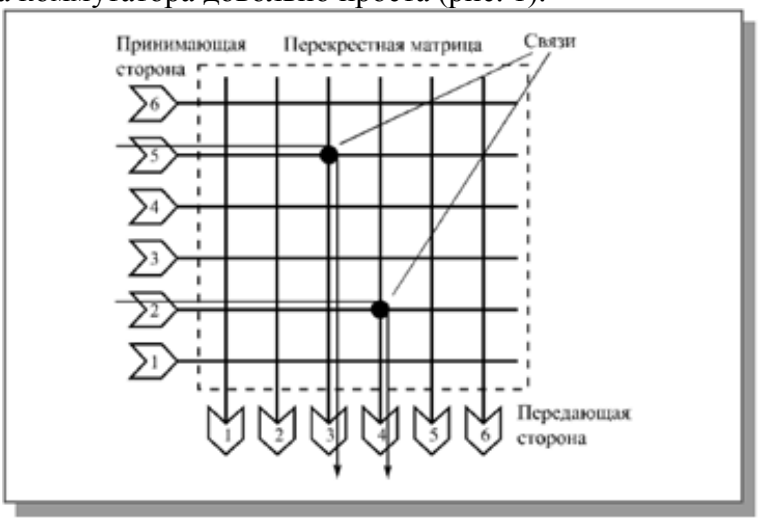


Рис. 1. Логическая схема коммутатора

Она включает в себя так называемую перекрестную (коммутационную) матрицу (Crossbar Matrix), во всех точках пересечения которой могут устанавливаться связи на время передачи пакета. В результате пакет, поступающий из любого сегмента, может быть передан в любой другой сегмент. В случае широковещательного пакета, адресованного всем абонентам, он передается во все сегменты одновременно, кроме того сегмента, по которому он пришел (рис. 2).

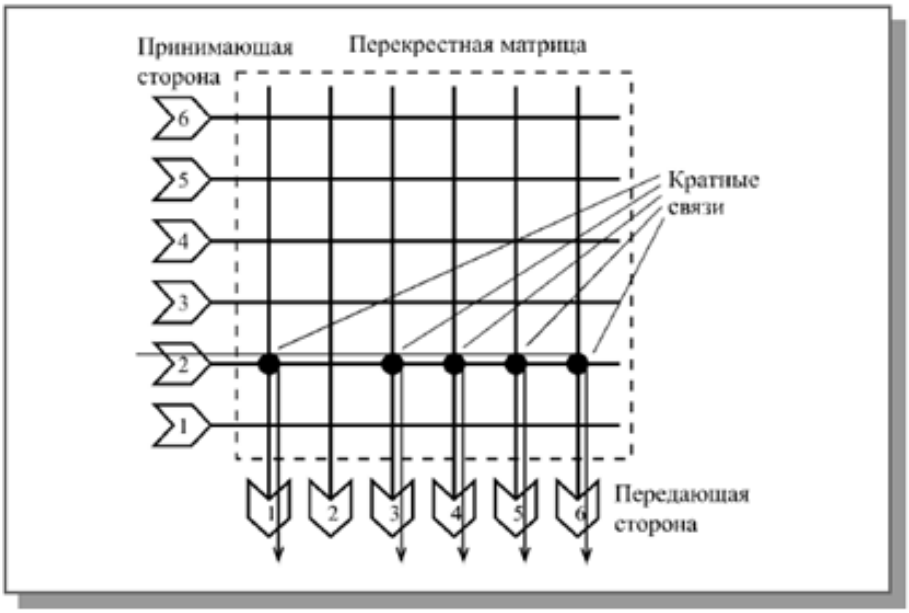


Рис. 2. Ретрансляция широковещательного пакета

Помимо перекрестной матрицы коммутатор включает в себя память, в которой он формирует таблицу MAC-адресов всех компьютеров, подключенных к каждому из его портов. Эта таблица создается на этапе инициализации сети и затем периодически обновляется для учета изменений конфигурации сети. Именно на основании анализа этой таблицы делается вывод о том, какие связи надо замыкать, куда отправлять пришедший пакет. Коммутатор читает MAC-адреса отправителя и получателя в пришедшем пакете и передает пакет в тот сегмент, в который он адресован. Если пакет адресован абоненту из того же сегмента, к которому принадлежит отправитель, то он не ретранслируется вообще. Широковещательный пакет не передается в тот сегмент, к которому присоединен абонент отправитель пакета. Адрес отправителя пакета заносится в таблицу адресов (если его там еще нет).

Коммутаторы выпускаются на различное число портов. Чаще всего встречаются коммутаторы с 6, 8, 12, 16 и 24 портами. Следует отметить, что мосты, как правило, редко поддерживают более 4 портов. Различаются коммутаторы с допустимым количеством адресов на один порт. Этот показатель определяет предельную сложность подключаемых к порту сегментов (количество компьютеров в каждом сегменте). Некоторые коммутаторы позволяют разбивать порты на группы, работающие независимо друг от друга, то есть один коммутатор может работать как два или три.

Так же, как и концентраторы, коммутаторы выпускаются трех видов в зависимости от сложности, возможности наращивания количества портов и стоимости:

- коммутаторы с фиксированным числом портов (обычно до 30);
- модульные коммутаторы (с числом портов до 100);
- стековые коммутаторы.

Коммутаторы характеризуются двумя показателями производительности:

Максимальная скорость ретрансляции пакетов измеряется при передаче пакетов из одного порта в другой, когда все остальные порты отключены.

Совокупная скорость ретрансляции пакетов измеряется при активной работе всех имеющихся портов. Совокупная скорость больше максимальной, но максимальная скорость, как правило, не может быть обеспечена на всех портах одновременно, хотя коммутаторы и способны одновременно обрабатывать несколько пакетов (в отличие от моста).

Главное правило, которого надо придерживаться при разбиении сети на части (сегменты) с помощью коммутатора, называется "правило 80/20". Только при его выполнении коммутатор работает эффективно. Согласно этому правилу, необходимо, чтобы не менее 80 процентов всех передач происходило в пределах одной части (одного сегмента) сети. И только 20 процентов всех передач должно происходить между разными частями (сегментами) сети, проходить через коммутатор. На практике это обычно сводится к тому, чтобы сервер и активно работающие с ним рабочие станции (клиенты) располагались на одном сегменте. Это же правило 80/20 применимо и к мостам.

Существует два класса коммутаторов, отличающихся уровнем интеллекта и способами коммутации:

- коммутаторы со сквозным вырезанием (Cut-Through);

- коммутаторы с накоплением и ретрансляцией (Store-and-Forward, SAF).

ВЫПОЛЕНИЕ

Построенная сетевая топология (Рис.3) содержит следующее сетевое оборудование:

- Маршрутизатор;
- Коммутатор;
- ПК;
- Прямой кабель;
- Перекрестный кабель.

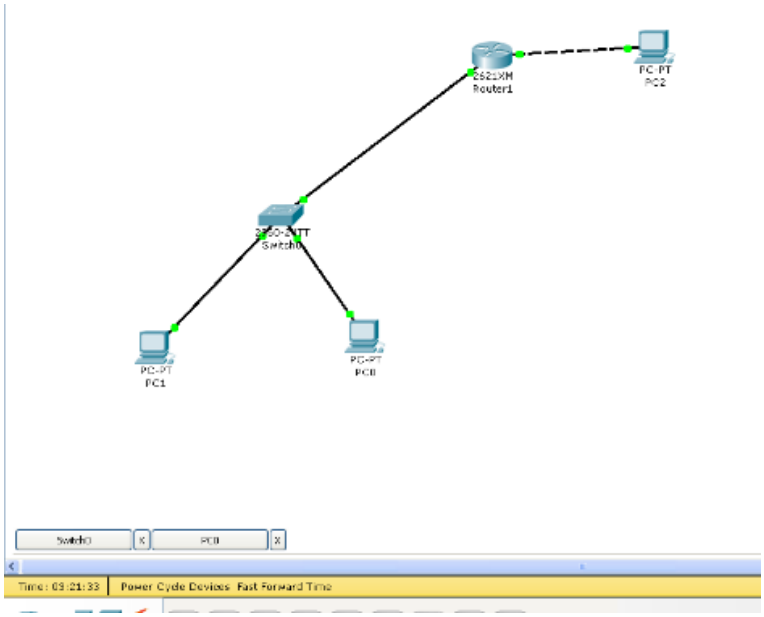
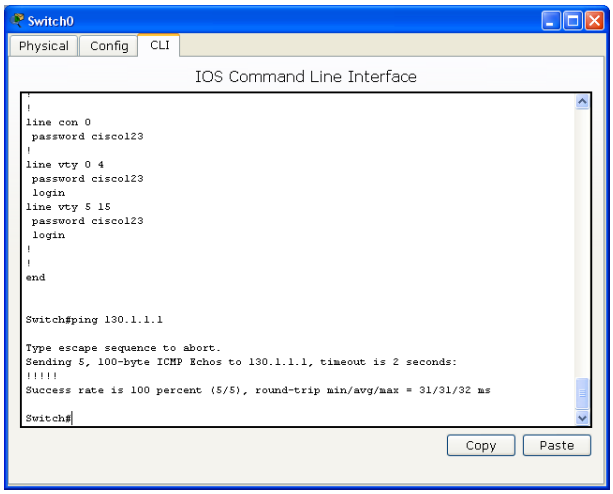


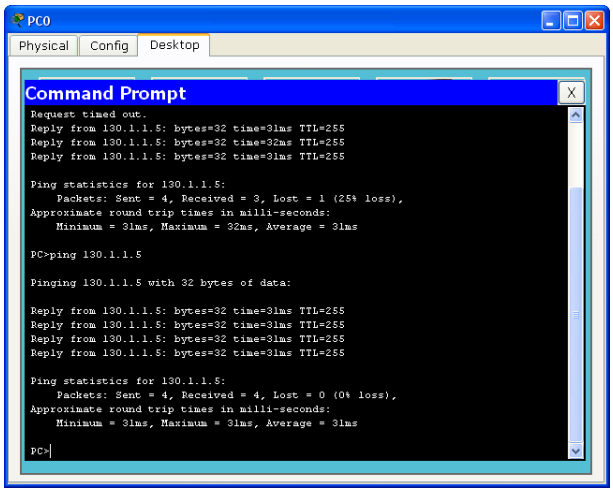
Рис.3. Топология сети

Табл.1. Информация о настройках оборудования

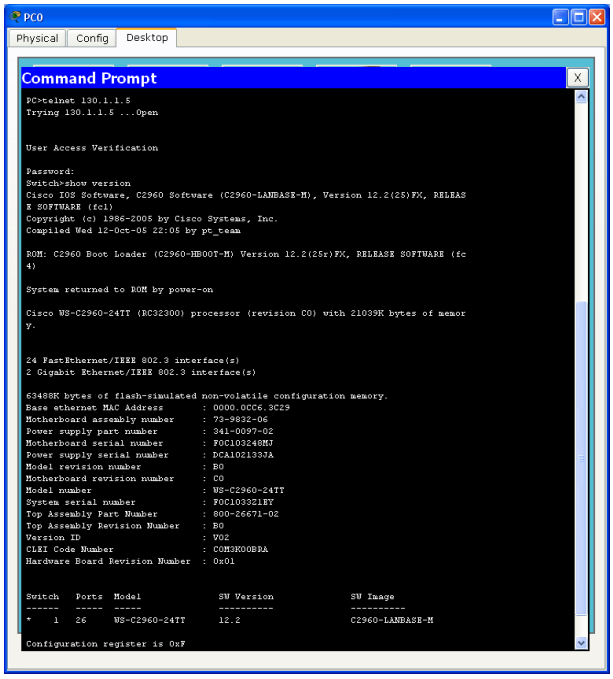
Device	IP address	Mask	Gateway
PC0	192.168.1.1	255.255.255.0	192.168.1.10
PC1	192.168.1.2	255.255.255.0	192.168.1.10
PC2	192.168.2.1	255.255.255.0	192.168.2.10
Router1 Fa 0/0	192.168.1.10	255.255.255.0	-
Router1 Fa 0/1	192.168.2.10	255.255.255.0	-



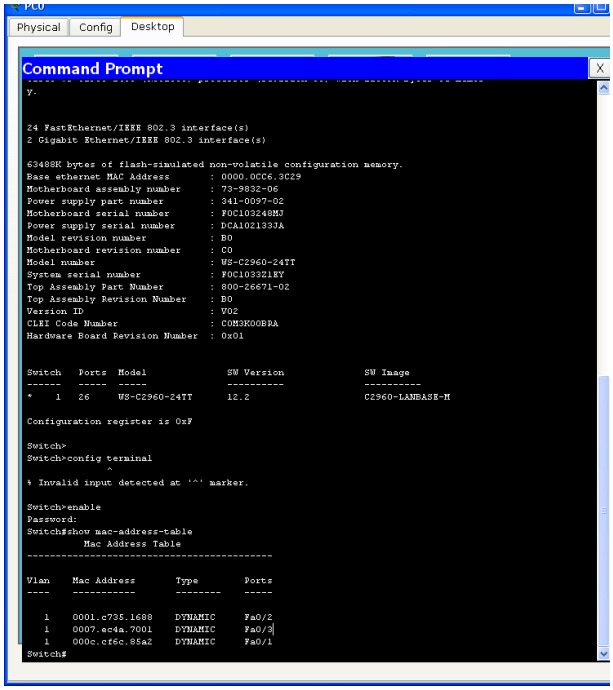
Выполнение эхо-запроса с коммутатора на маршрутизатор



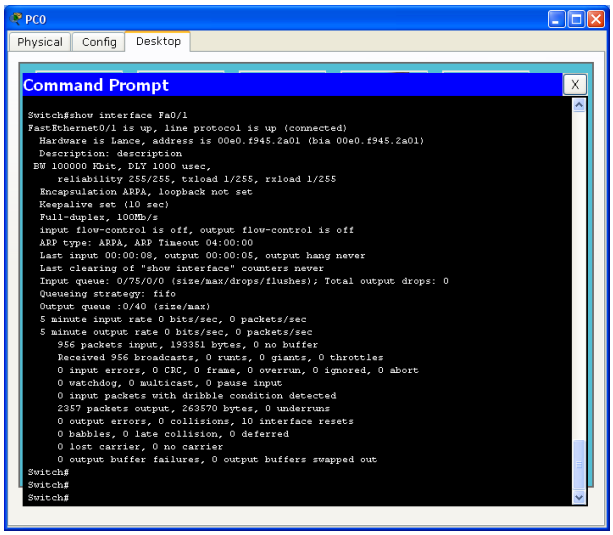
Выполнение эхо-запроса с хоста 130.1.1.2 до коммутатора



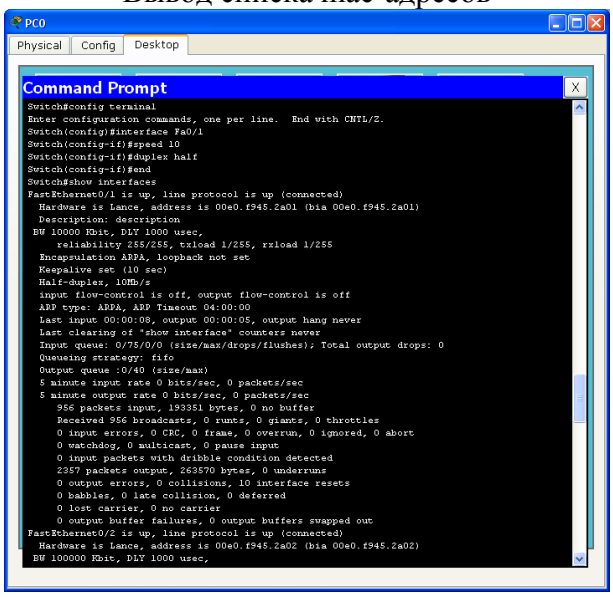
Выполнение telnet соединения



Вывод списка mac-адресов



Просмотр информации об интерфейсе Fa0/1



Настройка полудуплексного режима(10 Mb/s)

Часть 2. Конфигурация подсети и маршрутизатора

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Адреса и маски

Существует два варианта протокола IP – IPv4 и IPv6, отличающихся разрядностью сетевых адресов и возможностью назначения гибкого приоритета пакета. В одной и той же сети оба протокола могут сосуществовать, но пока таких сетей, где поддерживался бы протокол IPv6, достаточно мало. Более того, сети, с поддержкой и администрированием которых нам придется столкнуться, скорее всего, созданы не сегодня и, следовательно, построены на основе "старого" варианта IP, IPv4. В существующей литературе, если надо подчеркнуть, что речь идет об IPv6, указывают, что это именно этот вариант IP, а если пишут об IPv4, употребляют термин "IP".

Здесь и далее в этой книге речь пойдет об IPv4, но многие из изложенных принципов будут справедливы и тогда, когда все перейдут на IPv6.

Мы будем называть сетевым интерфейсом физическое или виртуальное (т.е. подразумеваемое или программно эмулируемое) устройство, которое способно выполнять функции приема пакетов данных от других подобных устройств и передачи им пакетов данных. Характерным примером сетевого интерфейса являются сетевые адаптеры (сетевые карты) и модемы.

Каждый сетевой интерфейс, осуществляющий прием и передачу пакетов по протоколу IP, должен иметь уникальный сетевой адрес. Под уникальным здесь понимается такой адрес, который в пределах данной IP-сети не принадлежит ни одному другому сетевому интерфейсу. Один интерфейс может иметь несколько IP-адресов, но один и тот же IP-адрес не может принадлежать разным сетевым интерфейсам.

В протоколе IP определено, что IP-адрес состоит из четырех байт и записывается в виде четырех десятичных чисел, отделенных друг от друга точками. Каждое число соответствует значению одного из этих байтов. IP-адреса объединены в блоки, которые называются сетями. В этом значении слово "сеть" употребляется реже, чем в более привычном значении "совокупность компьютеров и других сетевых устройств". Блоки адресов (сети) классифицированы по классам сетей, которые отличаются друг от друга особенностями маршрутизации.

Разбиение всего множества адресов на блоки последовательных адресов (например, 212.133.5.0-212.133.5.255) обусловлено тем, что в каждой точке сети должно быть известно, в каком направлении следует отправить пакет, адресованный сетевому интерфейсу с неким адресом. Промежуточные маршрутизаторы, которые объединяют большие сети и являются своеобразными "вокзалами" для пакетов, следующих из одного региона планеты в другой, не должны хранить записи о местоположении каждого сетевого интерфейса и о маршруте следования к нему. Им достаточно знать, предположим, что пакет, следующий по адресу 212.133.5.13, следует отправлять туда же, куда и прочие пакеты из вышеуказанного диапазона адресов. Это же относится и к маршрутизаторам, объединяющим сеть среднего офиса с двумя-тремя диапазонами адресов, но всю тяжесть нагрузки легче представить на примере более загруженных систем.

Мы можем уподобить пакет транзитной посылке, которая следует из одного города в другой на разных поездах с несколькими "пересадками". На каждом из промежуточных вокзалов работники почты знают, в какую сторону следует отправить посылку, когда она проходит мимо них, но их совершенно не заботит, куда ее отправят на следующем вокзале: там об этом позаботятся сотрудники следующего почтового отделения.

Чтобы облегчить маршрутизаторам работу по запоминанию диапазонов адресов, были придуманы маски сетей, определяющие, какую часть IP-адреса занимает номер сети, а какую – номер компьютера1) в этой сети. Фактически, номер компьютера нужен только тому маршрутизатору, к которому непосредственно подключена локальная сеть, где находится компьютер – адресат пакета. В отличие от номера компьютера, номер сети используется всеми промежуточными маршрутизаторами, которые передают пакет друг другу от места отправки до места назначения.

Рассмотрим, как используется маска сети, на примере. Представим себе сеть, состоящую из трех сегментов. В каждом из них – по 40 компьютеров. Мы уже знаем, что для того, чтобы снабдить уникальными адресами каждый из них, нам понадобится 120 адресов, т.е. меньше, чем доступно в сети класса C. Стало быть, мы можем назначить для каждой подсети свой диапазон адресов, при этом достаточно использовать адреса только из одной сети класса C. Выберем три диапазона так, чтобы в каждом из них было не менее 40 доступных адресов: 192.168.0.0-192.168.0.63, 192.168.0.64-192.168.0.127 и 192.168.0.128-192.168.0.191. Адреса сетей в целом и широковещательные адреса включены в диапазоны.

Компьютерам, которые находятся в одном сегменте, мы присваиваем адреса только из одного диапазона. Теперь наша задача – объяснить маршрутизатору, через какие сетевые интерфейсы следует передавать пакеты в каждый из сегментов, и именно здесь нам поможет маска сети.

У нас есть совершенно неизменная часть адресов в нашей сети – 192.168.0. Эти три байта адреса одинаковы для всех компьютеров нашей сети. Видно, что адреса в разных диапазонах отличаются значением последнего байта. Отметим, что адреса первого диапазона в двух старших битах этого байта имеют нули (действительно, двоичное представление чисел до 63 включительно дает значения от 00000000 до 00111111, старшие два бита выделены жирным шрифтом). Во втором диапазоне в упомянутых битах содержится 01: значения от 64 до 127 представляются в двоичном виде числами от 01000000 до 01111111. Аналогично, третий диапазон дает нам двоичные числа от 10000000 до 10111111. Выделенные биты разнятся между диапазонами, но одинаковы в пределах диапазона. Значение этих двух битов (их может быть больше, это зависит от числа диапазонов – подсетей, на которые разбита сеть) принято называть номером подсети; первый диапазон называют нулевой подсетью, второй – первой подсетью и т.д., смотря по значению этих битов.

Чтобы сообщить маршрутизатору, что к его первому сетевому адаптеру присоединена нулевая подсеть, ко второму – первая и к третьему – вторая, мы должны всего лишь указать маску сети на каждом из его сетевых интерфейсов. Так как мы уже договорились считать номером сети в IP-адресе значения тех битов, которые в маске имеют двоичное значение "1", то маска сети, разделенной на четыре подсети, будет иметь значение 255.255.255.192.

Почему значение последнего байта маски – 192? Потому, что 11000000 двоичное дает именно десятичное 192. Почему мы делим сеть на четыре подсети, хотя речь шла о трех диапазонах? Дело в том, что сеть можно разделить только на такое количество подсетей, которое кратно степени двойки. Поэтому вместо трех подсетей приходится брать ближайшее большее их количество.

Теперь остается назначить каждому интерфейсу маршрутизатора адреса из диапазона той подсети, которая присоединена к этому интерфейсу, и насладиться его четкой работой – по пересылке пакетов между интерфейсами.

Классы сетей

Самой крупной IP-сетью в мире является глобальная сеть Internet. Ее адресное пространство разделено на диапазоны адресов, которые называются "сетями". Сети разделены на классы. Адреса сетевых интерфейсов компьютеров в сети, как правило, относятся к классам A, B или C.

Разделение сетей на классы определяется в RFC 950 и ряде других RFC2). Кроме сетей A, B, C есть и другие сети, представляющие собой меньшие по размеру блоки сетей, которые используются для различных служебных надобностей.

Сети классов от A до C отличаются значением первого байта IP-адреса. Значение первого байта сети класса A находится в диапазоне от 1 до 126, класса B – от 128 до 191, класса C – от 192 до 223. Классы сетей, в адресах которых первый байт имеет значение от 224 до 254, именуются от D до F. Они имеют служебное назначение, и их адреса не используются обычными сетевыми интерфейсами.

Предполагается, что в сетях класса A номер сети занимает первый байт, а остальные три – это номер компьютера (точнее, номер сетевого интерфейса). Таким образом, в сети класса A может быть до 16777214 интерфейсов. В сети класса B номер сети занимает два байта адреса, максимальное число интерфейсов в такой сети – 65534. В сети класса C номер сети занимает три байта, номер компьютера – один, максимальное число интерфейсов в такой сети – 254.

Зарезервированные сетевые адреса

Надо иметь в виду, что один компьютер может иметь несколько сетевых интерфейсов, а каждый интерфейс может иметь несколько адресов.

Некоторые адреса в каждой сети являются зарезервированными и не могут использоваться для адресации какого-либо интерфейса.

Так, IP-адрес, в котором поле номера компьютера заполнено двоичными нулями, используется в качестве номера данной сети в целом. Например, адрес 131.45.0.0 обозначает целую сеть класса B. IP-адрес, в котором поле номера компьютера заполнено двоичными единицами, является широковещательным адресом сети (broadcast address) и применяется для одновременной рассылки пакета всем компьютерам данной сети. Получив пакет с адресом получателя 131.45.255.255, каждый компьютер сети 131.45.0.0 (класс B) воспримет этот пакет как предназначенный ему. Эти зарезервированные адреса используются, например, в целях управления маршрутизацией.

Существуют и другие зарезервированные адреса. Адрес 127.0.0.1 всегда указывает на локальный внутренний интерфейс системы. "127.0.0.1" обозначает для системы то же самое, что для человека – слово

"я". Этот локальный внутрений интерфейс требуется для того, чтобы одна программа (клиент) могла обратиться к другой программе (серверу), работающей на том же компьютере, стандартным образом. Например, можно обратиться из броузера на вашем компьютере к веб-серверу на вашем же компьютере. Локальный интерфейс обычно называется lo или lo0 (от слова loopback – "петля").

Адрес 0.0.0.0 используется для обозначения маршрута по умолчанию (основного шлюза), так как этот адрес означает "все сети".

Основной шлюз

Он зарезервирован для указания "всех остальных адресатов". Если пакет не удается соотнести с конкретной строкой таблицы маршрутизации, он отправляется в шлюз согласно указывающей на него строке таблицы маршрутизации. Например, таблица маршрутизации нашего хоста выглядит так:

```
# netstat -rn
Routing Table: IPv4
Destination Gateway Flags Ref Use Interface
192.168.5.0 192.168.5.33 U 1 2 elxl0
224.0.0.0 192.168.5.33 U 1 0 elxl0
default 192.168.5.1 UG 1 0
127.0.0.1 127.0.0.1 UH 61 1013 lo0
```

Адрес нашего компьютера в этой сети – 192.168.5.33. Предположим, нам надо отправить пакет по адресу 192.168.5.30. Для этого наша система посмотрит в таблицу маршрутизации и обнаружит там маршрут 192.168.5.0, для которого указан шлюз (gateway) 192.168.5.33 – наш собственный интерфейс. Стало быть, компьютер 192.168.5.30 находится в непосредственно присоединенной к нам сети и ему надо послать пакет напрямую. Ситуация изменится, если адресатом будет, скажем, компьютер 192.168.10.1. Тогда в таблице маршрутизации вначале не найдется подходящего маршрута – ведь там нет отдельной строки для сети 192.168.10.1, верно? Тогда пакет будет отправлен в основной шлюз, "шлюз по умолчанию", тот, что в выводе netstat обозначен словом default. Адрес назначения пакетов "во все остальные сети", тех самых, которые отправляются в шлюз, в таблице маршрутизации в ядре обозначается как 0.0.0.0.

Пакеты, предназначенные для отправки "всем остальным", направляются в основной шлюз. Основной шлюз (default gateway) – это такое место, куда любой компьютер сети отправляет пакет, если не знает, в какую сторону его лучше отправить. Действие такого шлюза подобно действию почтальона. Если Вы хотите сделать сюрприз девушке, которая живет с вами в одном подъезде, вы можете положить нежное письмо ей прямо в почтовый ящик. Если же адресат живет в другом городе, вы положите конверт в другой почтовый ящик, тот, из которого почтальон вынимает почту для отправки. Основной шлюз имеет сетевой интерфейс, работающий таким "почтовым ящиком" для пакетов, которые адресованы из локальной сети вовне.

ВЫПОЛНЕНИЕ

На рис. 4 приведена сетевая топология, построенная с использованием следующего сетевого оборудования:

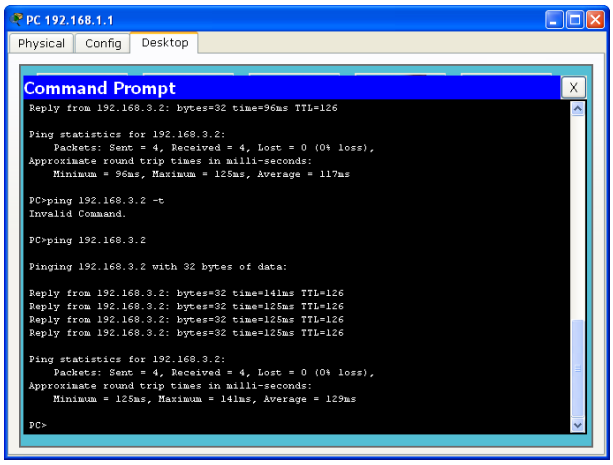
- персональных компьютеров;
- коммутатора;
- маршрутизатора.

Для данной подсети потребовалось создание 3 подсетей, а именно:

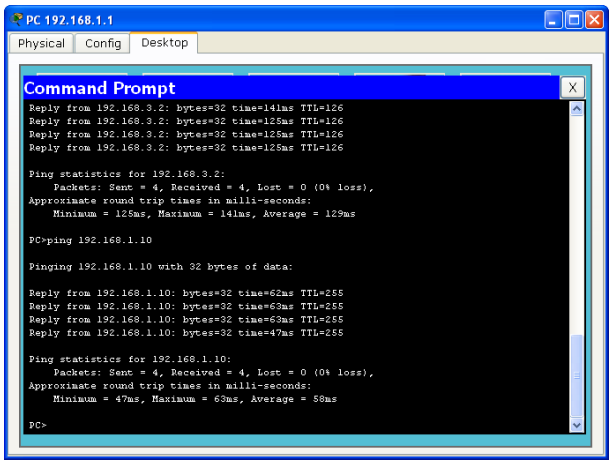
- 192.168.1.0/24;
- 192.168.2.0/24;
- 192.168.3.0/24.

Чтобы подсоединить к маршрутизатору кабель DTE, необходимо выкл router и вставить в него модуль NM-4A/S. Вкл router и подключить кабель.

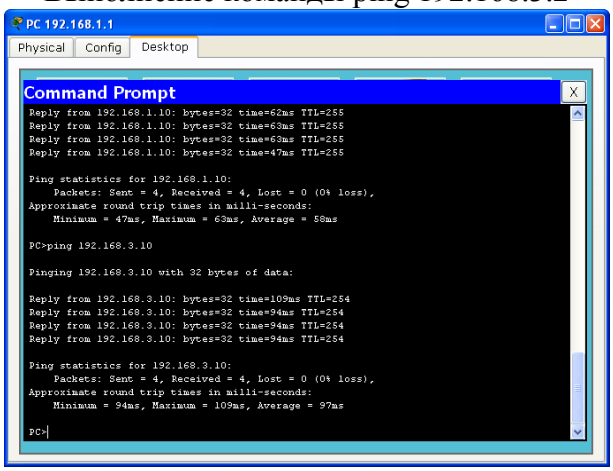
Рис. 4 Топология сети



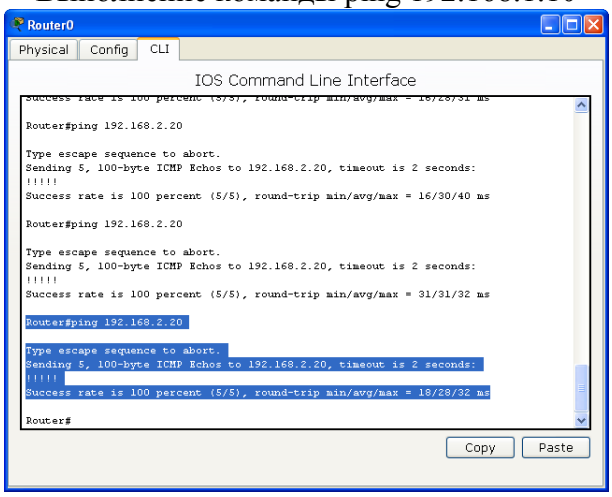
Выполнение команды ping 192.168.3.2



Выполнение команды ping 192.168.1.10



Выполнение команды ping 192.168.3.10



Выполнение команды «ping 192.168.2.10»(Router0) с Router1

Практическая работа №13
Настройка сервера DHCP и DNS

Цель занятия: Научиться подключать и настраивать оконечные и сетевые устройства для их взаимодействия в клиент-серверной модели сети.

Научиться осуществлять настройку маршрутизатора для обеспечения передачи данных из одной сети в другую.

КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Сервер (от англ. server, обслуживающий). В зависимости от предназначения существует несколько определений понятия сервер.

1. Сервер (сеть) – логический или физический узел сети, обслуживающий запросы к одному адресу и/или доменному имени (смежным доменным именам), состоящий из одного или системы аппаратных серверов, на котором выполняются один или система серверных программ

2. Сервер (программное обеспечение) – программное обеспечение, принимающее запросы от клиентов (в архитектуре клиент-сервер).

3. Сервер (аппаратное обеспечение) – компьютер (или специальное компьютерное оборудование) выделенный и/или специализированный для выполнения определенных сервисных функций.

3. Сервер в информационных технологиях – программный компонент вычислительной системы, выполняющий сервисные функции по запросу клиента, предоставляя ему доступ к определённым ресурсам.

Взаимосвязь понятий. Серверное приложение (сервер) запускается на компьютере, так же называемом "сервер", при этом при рассмотрении топологии сети, такой узел называют "сервером". В общем случае может быть так, что серверное приложение запущено на обычной рабочей станции, или серверное приложение, запущенное на серверном компьютере в рамках рассматриваемой топологии, выступает в роли клиента (т.е. не является сервером с точки зрения сетевой топологии).

Клиент-серверная система характеризуется наличием двух взаимодействующих самостоятельных процессов – клиента и сервера, которые, в общем случае, могут выполняться на разных компьютерах, обмениваясь данными по сети.

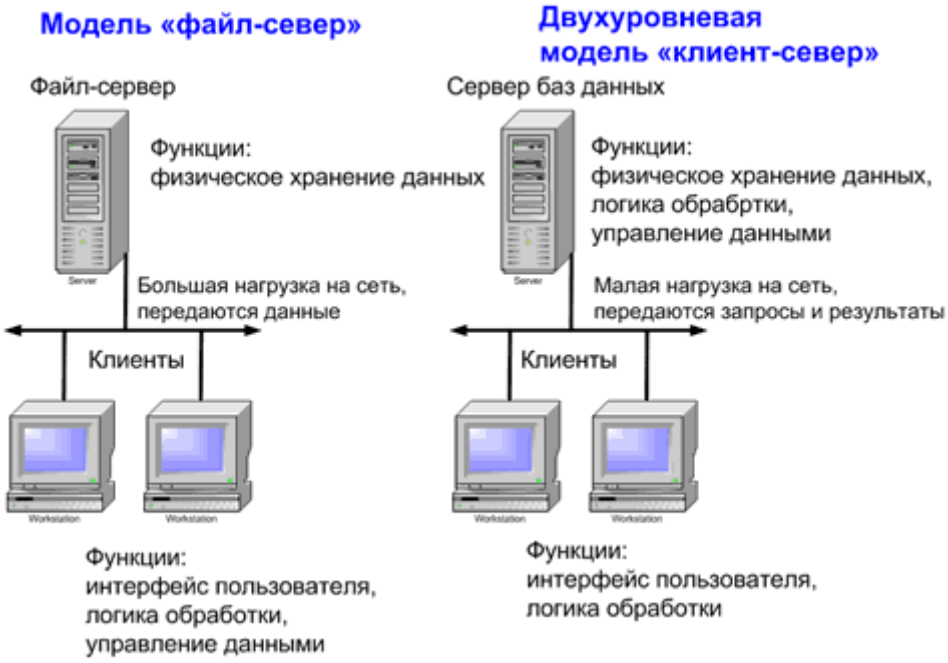
Процессы, реализующие некоторую службу, например службу файловой системы или базы данных, называются серверами (servers). Процессы, запрашивающие службы у серверов путем посылки запроса и последующего ожидания ответа от сервера, называются клиентами (clients) .

По такой схеме могут быть построены системы обработки данных на основе СУБД, почтовые и другие системы. Мы будем говорить о базах данных и системах на их основе. И здесь удобнее будет не просто рассматривать клиент-серверную архитектуру, а сравнить ее с другой – файл-серверной.

В файл-серверной системе данные хранятся на файловом сервере (например, Novell NetWare или Windows NT Server), а их обработка осуществляется на рабочих станциях, на которых, как правило, функционирует одна из, так называемых, "настольных СУБД" - Access, FoxPro, Paradox и т.п..

Приложение на рабочей станции "отвечает за все" – за формирование пользовательского интерфейса, логическую обработку данных и за непосредственное манипулирование данными. Файловый сервер предоставляет услуги только самого низкого уровня - открытие, закрытие и модификацию файлов. Обратите внимание – файлов, а не базы данных. Система управления базами данных расположена на рабочей станции.

Таким образом, непосредственным манипулированием данными занимается несколько независимых и несогласованных между собой процессов. Кроме того, для осуществления любой обработки (поиск, модификация, суммирование и т.п.) все данные необходимо передать по сети с сервера на рабочую станцию (см. рис. Сравнение файл-серверной и клиент-серверной моделей).



В клиент-серверной системе функционируют (как минимум) два приложения – клиент и сервер, делящие между собой те функции, которые в файл-серверной архитектуре целиком выполняет приложение на рабочей станции. Хранением и непосредственным манипулированием данными занимается сервер баз данных, в качестве которого может выступать Microsoft SQL Server, Oracle, Sybase и т.п..

Формированием пользовательского интерфейса занимается клиент, для построения которого можно использовать целый ряд специальных инструментов, а также большинство настольных СУБД. Логика обработки данных может выполняться как на клиенте, так и на сервере. Клиент посылает на сервер запросы, сформулированные, как правило, на языке SQL. Сервер обрабатывает эти запросы и передает клиенту результат (разумеется, клиентов может быть много).

Таким образом, непосредственным манипулированием данными занимается один процесс. При этом, обработка данных происходит там же, где данные хранятся - на сервере, что исключает необходимость передачи больших объемов данных по сети.

Что дает архитектура клиент-сервер?

Посмотрим на данную архитектуру с точки зрения потребностей бизнеса. Какие же качества привносит клиент-сервер в информационную систему?

Надежность

Сервер баз данных осуществляет модификацию данных на основе механизма транзакций, который придает любой совокупности операций, объявленных как транзакция, следующие свойства:

- **Атомарность** – при любых обстоятельствах будут либо выполнены все операции транзакции, либо не выполнена ни одна; целостность данных при завершении транзакции;
- **Независимость** – транзакции, инициированные разными пользователями, не вмешиваются в дела друг друга;
- **устойчивость к сбоям** – после завершения транзакции, ее результаты уже не пропадут.

Механизм транзакций, поддерживаемый сервером баз данных, намного более эффективен, чем аналогичный механизм в настольных СУБД, т.к. сервер централизованно контролирует работу транзакций. Кроме того, в файл-серверной системе сбой на любой из рабочих станций может привести к потере данных и их недоступности для других рабочих станций, в то время, как в клиент-серверной системе сбой на клиенте, практически, никогда не сказывается на целостности данных и их доступности для других клиентов.

Масштабируемость

Масштабируемость – способность системы адаптироваться к росту количества пользователей и объема базы данных при адекватном повышении производительности аппаратной платформы, без замены программного обеспечения.

Общеизвестно, что возможности настольных СУБД серьезно ограничены - это пять-семь пользователей и 30-50 Мб, соответственно. Цифры, разумеется, представляют собой некие средние

значения, в конкретных случаях они могут отклоняться как в ту, так и в другую сторону. Что наиболее существенно, эти барьеры нельзя преодолеть за счет наращивания возможностей аппаратуры.

Системы же на основе серверов баз данных могут поддерживать тысячи пользователей и сотни ГБ информации – дайте им только соответствующую аппаратную платформу.

Безопасность

Сервер баз данных предоставляет мощные средства защиты данных от несанкционированного доступа, невозможные в настольных СУБД. При этом, права доступа администрируются очень гибко – до уровня полей таблиц. Кроме того, можно вообще запретить прямое обращение к таблицам, осуществляя взаимодействие пользователя с данными через промежуточные объекты – представления и хранимые процедуры. Так что администратор может быть уверен – никакой слишком умный пользователь не прочитает то, что ему читать не положено.

Гибкость

В приложении, работающем с данными, можно выделить три логических слоя:

- **пользовательского интерфейса**
- **правил логической обработки** (бизнес-правил);
- **управления данными** (не следует только путать логические слои с физическими уровнями, о которых речь пойдет ниже).

Как уже говорилось, в файл-серверной архитектуре все три слоя реализуются в одном монолитном приложении, функционирующем на рабочей станции. Поэтому изменения в любом из слоев приводят однозначно к модификации приложения и последующему обновлению его версий на рабочих станциях.

В двухуровневом клиент-серверном приложении, показанном на рисунке выше, как правило, все функции по формированию пользовательского интерфейса реализуются на клиенте, все функции по управлению данными - на сервере, а вот бизнес-правила можно реализовать как на сервере используя механизмы программирования сервера (хранимые процедуры, триггеры, представления и т.п.), так и на клиенте.

В трехуровневом приложении появляется третий, промежуточный уровень, реализующий бизнес-правила, которые являются наиболее часто изменяемыми компонентами приложения.

DHCP (Dynamic Host Configuration Protocol – протокол динамической настройки узла) – сетевой протокол, позволяющий компьютерам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP. Данный протокол работает по модели «клиент-сервер». Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей TCP/IP.

DNS (Domain Name System – система доменных имён) – компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене (SRV-запись).

Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.

Основой DNS является представление об иерархической структуре доменного имени и зонах. Каждый сервер, отвечающий за имя, может делегировать ответственность за дальнейшую часть домена другому серверу (с административной точки зрения — другой организации или человеку), что позволяет возложить ответственность за актуальность информации на серверы различных организаций (людей), отвечающих только за «свою» часть доменного имени.

DNS-сервер – приложение, предназначенное для ответов на DNS-запросы по соответствующему протоколу. Также DNS-сервером могут называть хост, на котором запущено приложение.

Ход выполнения:

1. Запустите программный пакет Packet Tracer с помощью исполняемого файла Cisco Packet Tracer.exe. Откроется основное окно программы.

2. Добавьте в рабочую зону следующие устройства: 2 сервера (Generic Server-PT); коммутатор (2950-24); компьютеры (Generic PC-PC) в количестве 2 шт; маршрутизатор (1841).

3. Подключите устройства как показано на рисунке 1. Учтите, что пунктирная линия – это кроссоверный кабель, а прямая – прямой медный.

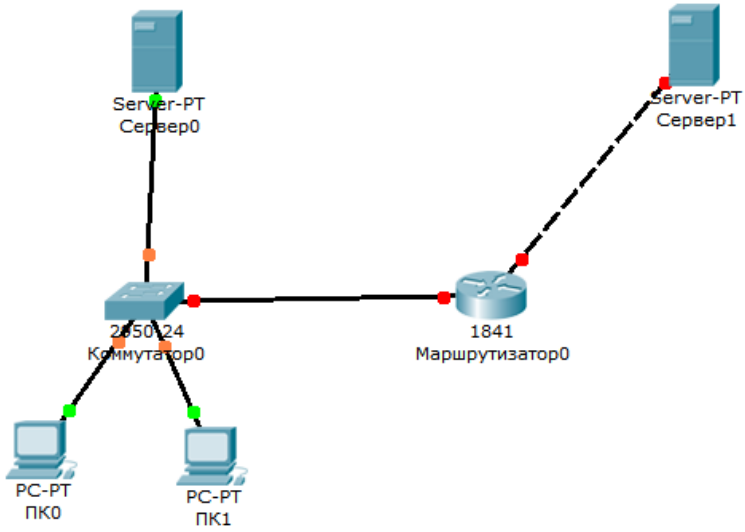


Рисунок 1. Соединение элементов сети

4. Осуществите настройку Сервер0. Для этого в настройках конфигураци изменить следующие параметры:

- Глобальные настройки: Шлюз - статический – 192.168.1.10
- Интерфейс: Fast Ethernet – ip адрес 192.168.1.1
- Службы: HTTP – выключено; DHCP – включено (основной шлюз – 192.168.1.10; DNS – сервер 192.168.1.1; начальный ip адрес – 192.168.1.2; макс.кол-во пользователей - 256; и нажать кнопку «сохранить»); служба DNS – включено (добавить 1 запись имя INTERNET , адрес 192.168.2.1 и нажать кнопку «добавить»)

5. Осуществите настройку рабочих станций (Рабочий стол – настройка IP – установить в позицию DHCP – убедиться в том, что адреса выданы (рисунок 2))

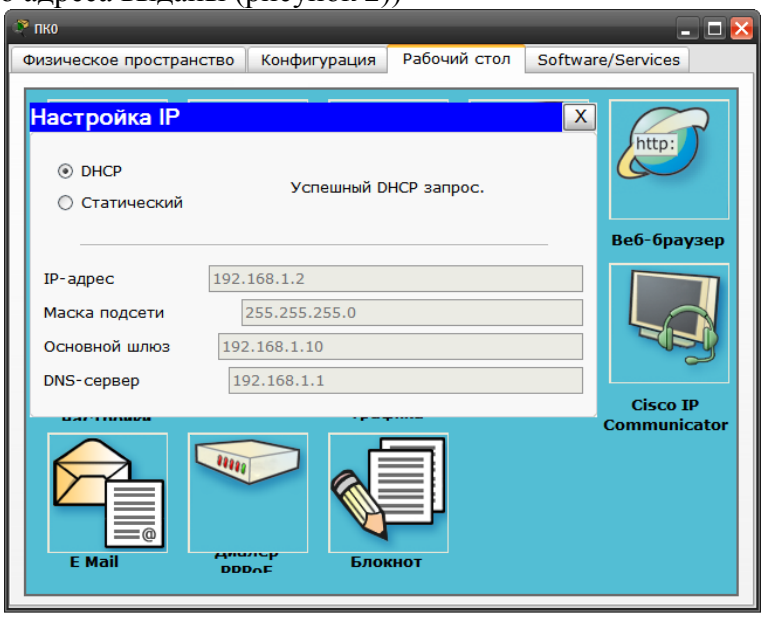


Рисунок 2 – успешное выполнение DHCP запроса

6. Настройте Сервер1 следующим образом:

- Глобальные настройки: Шлюз - статический – 192.168.2.10
- Интерфейс: Fast Ethernet – ip адрес 192.168.2.1;
- Службы: HTTP – включено; DHCP – выключено; DNS – выключено
- в службе HTTP изменить html-страницу: заменить строчку **<hr>Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.**

на **<hr>Hello World!! This is internet server**

Теперь рассмотрим интерфейс настройки маршрутизатора.

Окно настройки как и окно настройки конечных устройств состоит из основных частей:

Физическое пространство, Конфигурация и в добавок CLI – command line interface (командная строка) Настройка через Конфигурацию так же может быть осуществлена через CLI, окно настройки приведено на рисунке 3.

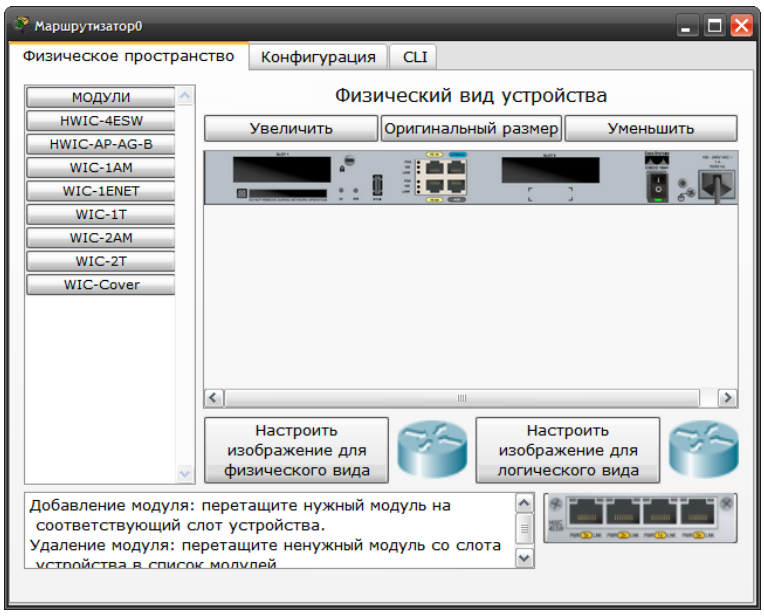


Рисунок 3 – настройка маршрутизатора.

Основные настройки, необходимые для работоспособности нашей сети можно осуществить из вкладки КОНФИГУРАЦИЯ.

7. Осуществить настройку подключенного маршрутизатора по следующим параметрам:

- интерфейс Fast Ethernet 0/0 – ip адрес – 192.168.1.10, маска подсети 255.255.255.0; состояние порта – «вкл»
- интерфейс Fast Ethernet 0/1 – ip адрес – 192.168.2.10, маска подсети 255.255.255.0; состояние порта – «вкл»

8. Теперь ожидаем некоторое время (обычно около 1 минуты), для того что бы прогрузился порт коммутатора.

9. Убедившись в том, что все индикаторы состояния сети горят зелёным, проверьте работоспособность соединения с помощью на любой рабочей станции с помощью утилиты ping (рабочий стол – командная строка – ping 192.168.2.1)

10. Теперь проверим работоспособность служб DNS сервера0, и HTTP сервера1. Зайдите на любой персональный компьютер – рабочий стол – откройте веб браузер. В поисковой строке введите INTERNET и нажмите кнопку подтверждения. В окне браузера должна отобразиться изменённая нами html – страничка., хранящаяся на сервере (рисунок 4)

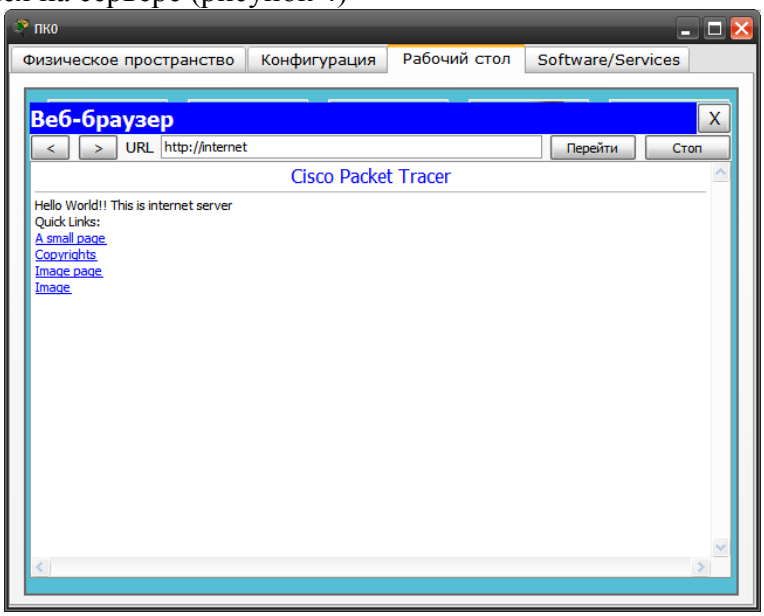


Рисунок 4. Проверка работоспособности http протокола

11. Сохраните созданную модель сети.

Практическая работа №14
Настройка электронной почты

Цель работы: Провести анализ работы протоколов уровня приложений и транспорта с использованием программного сетевого эмулятора Cisco Packet Tracer.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Модель TCP/IP. Семейство протоколов TCP/IP основано на четырехуровневой эталонной модели. Все протоколы, входящие в семейство протоколов TCP/IP, расположены на трех верхних уровнях этой модели.

Каждый уровень модели TCP/IP соответствует одному или нескольким уровням семиуровневой эталонной модели OSI (Open Systems Interconnection – взаимодействие открытых систем), предложенной ISO – международной организацией по стандартам (International Standards Organization). Типы служб и протоколов, используемых на каждом уровне модели TCP/IP, более подробно описаны в следующей таблице.

Уровень	**Описание**	**Протоколы**
Приложение	Определяет прикладные протоколы TCP/IP и интерфейс программ со службами транспортного уровня, необходимый для использования сети.	HTTP, Telnet, FTP, TFTP, SNMP, DNS, SMTP, X Windows, другие прикладные протоколы
Транспортный	Обеспечивает управление сеансами связи между компьютерами. Определяет уровень служб и состояние подключения, используемые при транспортировке данных.	TCP, UDP
Интернет	Упаковывает данные в IP-датаграммы, содержащие информацию об адресах источника и приемника, которая используется для перенаправления датаграмм от узла к узлу и по сетям. Выполняет маршрутизацию IP-датаграмм.	IP, ICMP, ARP, RARP
Сетевого интерфейса	Определяет средства и принципы физической передачи данных по сети, включая преобразование битов данных в электрические или другие сигналы аппаратными устройствами, непосредственно подключенными к среде передачи, такой как коаксиальный кабель, оптоволокно или витая пара.	Ethernet, Token Ring, FDDI, X.25, Frame Relay, RS-232, v.35

ПРОТОКОЛЫ УРОВНЯ ПРИЛОЖЕНИЙ.

SMTP (Simple Mail Transfer Protocol) – простой почтовый протокол. Он поддерживает передачу почтовых электронных сообщений по сети Интернет. Протокол называется простым, потому что обеспечивает передачу информации пользователям, готовым к немедленной доставке. Передача осуществляется в режиме 7-битовых слов. Он требует наличия программ перехода от принятого в большинстве программ формата с 8-разрядными словами к формату с 7-разрядными словами. Система поддерживает:

- посылку одиночных сообщений одному или более получателям;
- посылку сообщений, включающих в себя текст, голосовые сообщения, видео или графические материалы.

Протокол передачи файлов (FTP – File Transfer Protocol) используется для передачи файлов от одного компьютера к другому. Обеспечивает просмотр каталогов удаленного компьютера, копирование, удаление и пересылку файлов. FTP отличается от других протоколов тем, что устанавливает два соединения между хостами. Одно используется для передачи информации, а другое – для управления передачей.

DNS (Domain Name System) – служба доменных имен. Она осуществляет присвоение уникальных имен всем пользователям и узлам сети Интернет и устанавливает логическую связь с их сетевыми адресами. Доменное имя представляется иерархической структурой, имеющей несколько уровней. Типовые имена доменов верхнего уровня закреплены следующим образом:

.com – коммерческие организации;

.gov – правительственные учреждения;

.org – некоммерческие организации;

.net – центры поддержки сети;

.int – международные организации;

.mil – военные структуры.

SNMP (Simple Network Management Protocol) – простой протокол управления сетью. Он обеспечивает набор фундаментальных действий по наблюдению и обслуживанию Интернета.

Протокол разработан так, чтобы он мог контролировать устройства, созданные различными изготовителями и установленные на различных физических сетях. Другими словами, SNMP освобождает задачи управления от учета физических характеристик управляемых устройств и от основной технологии организации сети.

Сетевая файловая система (NFS – Network File System). Это один из многих протоколов (например, на рисунке показан еще один протокол RPC –Remote Procedure Call – вызов удаленной процедуры), который позволяет использование файлов, содержащих процедуры управления и периферии в другом компьютере.

Тривиальный (простейший) протокол передачи файлов TFTP (Trivial File Transfer Protocol). Используется в простых случаях при начальной загрузке рабочих станций или загрузке маршрутизаторов, не имеющих внешней памяти.

Протокол передачи гипертекста (HTTP – Hyper Text Transfer Protocol) – транспортный протокол, который применяется в Интернете при обмене документами, представленными на языке описания гипертекстовых документов.

Язык разметки гипертекста (HTML – Hyper Text Markup Language). Является одним из главных языков, используемых в сети WWW.

ПРОТОКОЛЫ УРОВНЯ ТРАНСПОРТА

Протокол управления передачей TCP (Transmission Control Protocol) является обязательным стандартом TCP/IP, который описан в документе RFC 793 «Transmission Control Protocol (TCP)» и предоставляет надежную службу доставки пакетов, ориентированную на установление соединения. Протокол TCP:

- гарантирует доставку IP-датаграмм;
- выполняет разбиение на сегменты и сборку больших блоков данных, отправляемых программами;
- обеспечивает доставку сегментов данных в нужном порядке;
- выполняет проверку целостности переданных данных с помощью контрольной суммы;
- посылает положительные подтверждения, если данные получены успешно. Используя избирательные подтверждения, можно также посылать отрицательные подтверждения для данных, которые не были получены;
- предлагает предпочтительный транспорт для программ, которым требуется надежная передача данных с установлением сеанса связи, например, для БД «клиент-сервер» и программ электронной почты.

КАК РАБОТАЕТ TCP. TCP основан на связи «точка-точка» между двумя узлами сети. TCP получает данные от программ и обрабатывает их как поток байтов. Байты группируются в сегменты, которым TCP присваивает последовательные номера, необходимые для правильной сборки сегментов на узле-приемнике.

Чтобы два узла TCP могли обмениваться данными, им нужно сначала установить сеанс связи друг с другом. Сеанс TCP инициализируется с помощью процесса, называемого трехэтапным установлением связи. В этом процессе синхронизируются номера последовательности и передается управляющая информация, необходимая для установления виртуального соединения между узлами.

По завершении процесса трехэтапного установления связи начинается пересылка и подтверждение пакетов в последовательном порядке между этими узлами. Аналогичный процесс используется TCP перед прекращением соединения для того, чтобы убедиться, что оба узла закончили передачу и прием данных.

Протокол UDP. Протокол датаграмм пользователя UDP (User Datagram Protocol) является стандартом TCP/IP, описанным в документе RFC 768 «User Datagram Protocol (UDP)». UDP используется некоторыми программами вместо TCP для быстрой, простой, но ненадежной передачи данных между узлами TCP/IP.

UDP обеспечивает службу датаграмм, не ориентированную на установление соединения, что означает, что UDP не гарантирует ни доставку, ни правильность порядка доставки датаграмм. Узел-источник, которому требуется надежная связь, должен использовать либо протокол TCP, либо программу, которая сама обеспечивает подтверждения и следит за правильностью порядка датаграмм.

UDP	TCP
Служба, не ориентированная на установление соединения; сеанс связи между узлами не устанавливается.	Служба, ориентированная на установление соединения; между узлами устанавливается сеанс связи.
UDP не гарантирует и не подтверждает доставку данных, а также не гарантирует порядок их доставки.	TCP гарантирует доставку при помощи подтверждений и контроля порядка принимаемых данных.
Программы, использующие UDP, ответственны за обеспечение надежности передачи данных.	Программам, использующим TCP, гарантируется надежность передачи данных.
UDP – быстрый протокол с небольшими накладными расходами, поддерживающий связь «точка-точка» и «точка-многие точки».	TCP медленнее, требует больших накладных расходов и поддерживает только связь «точка-точка».

ВЫПОЛНЕНИЕ

На рис.1 приведена спроектированная сеть, которая включает в себя следующее оборудование:

- Маршрутизаторы;
- ПК;
- Сервер.

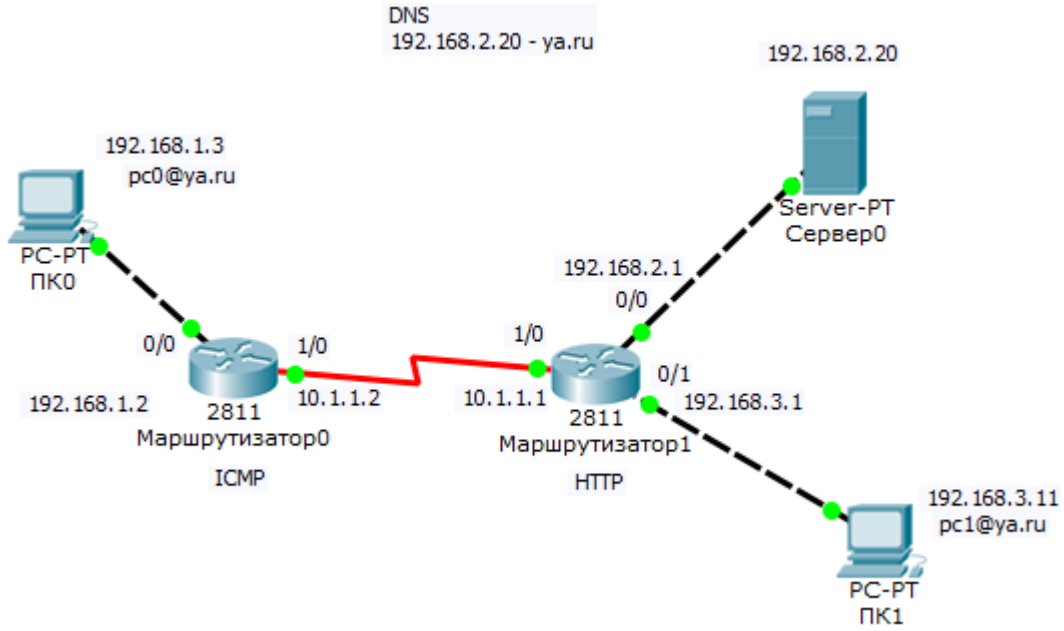


Рис. 1. Результат построения сети

1. Настроить email и DNS на сервере в соответствии с рисунками 2 и 3

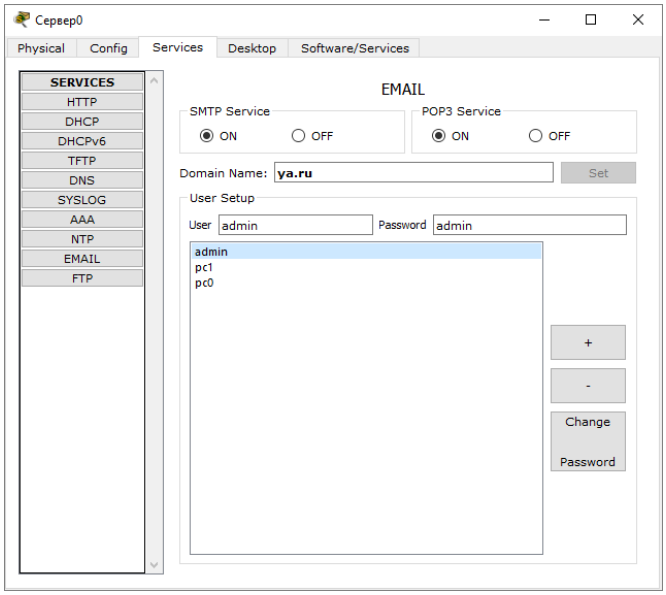


Рис.2. Настройка e-mail

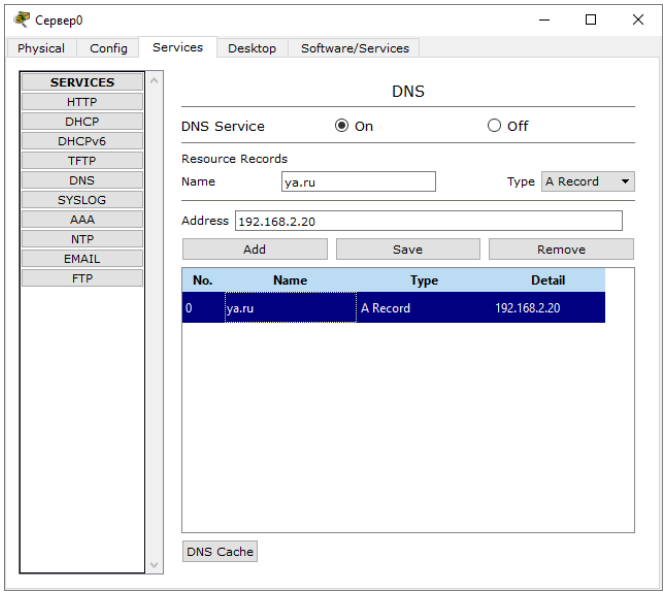


Рис.3. Настройка DNS-сервиса

2. Настроить электронную почту на ПК0 в соответствии с рисунком 4. Провести аналогичные настройки на ПК1

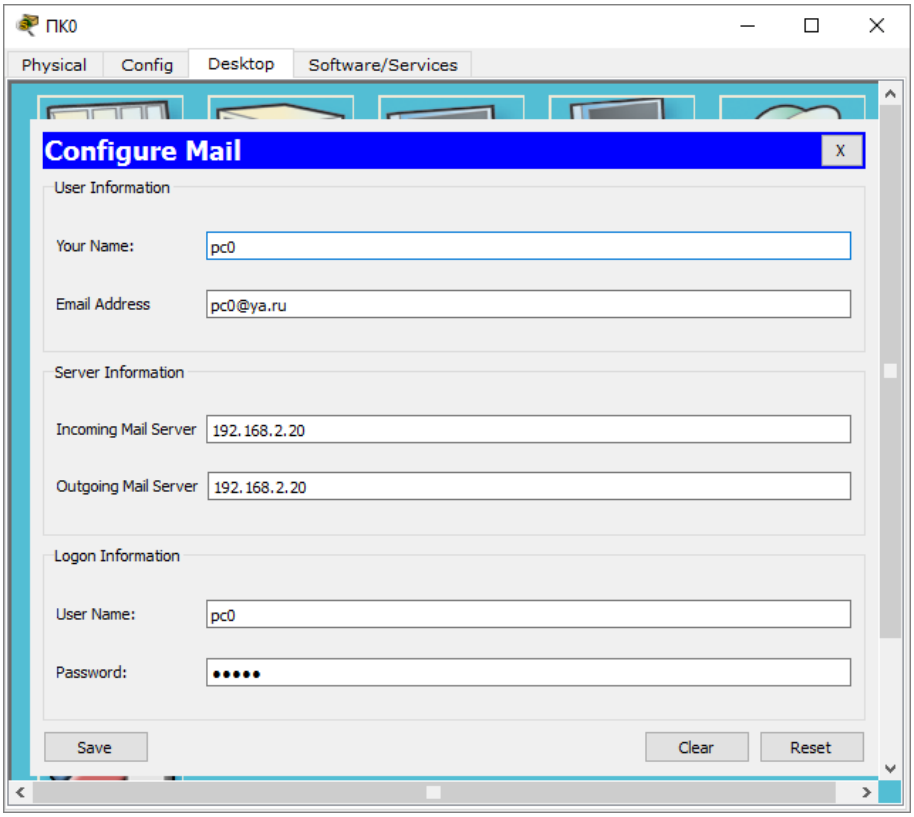


Рис.4. Настройка электронной почты на ПК0

3. Отправить email сообщение с ПК0 на ПК1. Проверить получение на ПК1

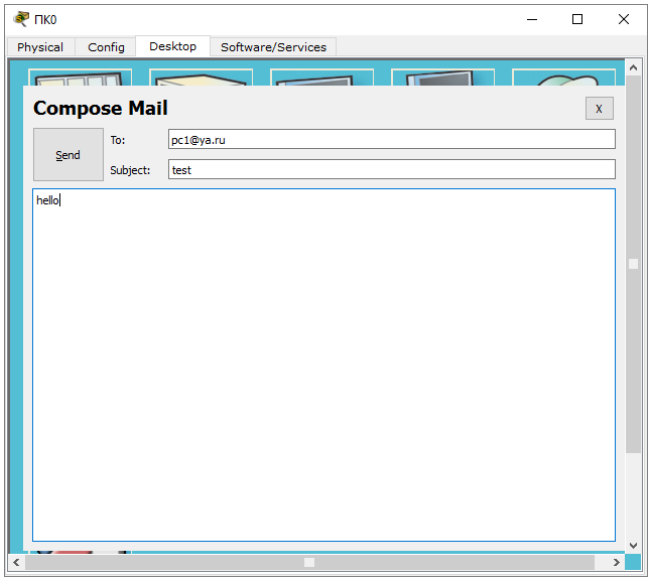


Рис.5. Передача сообщения

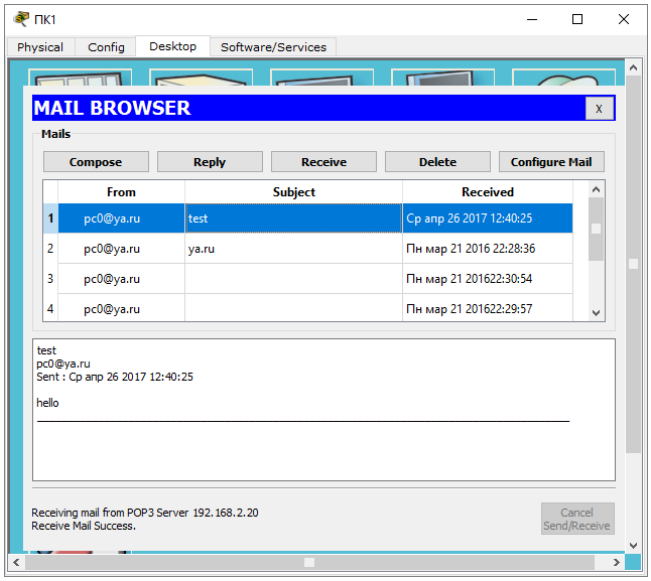


Рис.6. Получение сообщения

Практическая работа №15
Проектирование и разработка БД в СУБД MS Access

Цель работы:

1. Научиться создавать таблицы баз данных
2. Освоить технологию заполнения базы данных.
3. Научиться использовать индексирование для ускорения поиска в БД.
4. Научиться работать с формами баз данных;
5. Освоить все способы создания форм.
6. Научиться формировать простые запросы на выборку баз данных;
7. Научиться разрабатывать запросы с параметрами;
8. Освоить технологию создания различных запросов.
9. Освоить технологию создания отчетов с группированием данных

1 Основные понятия и термины

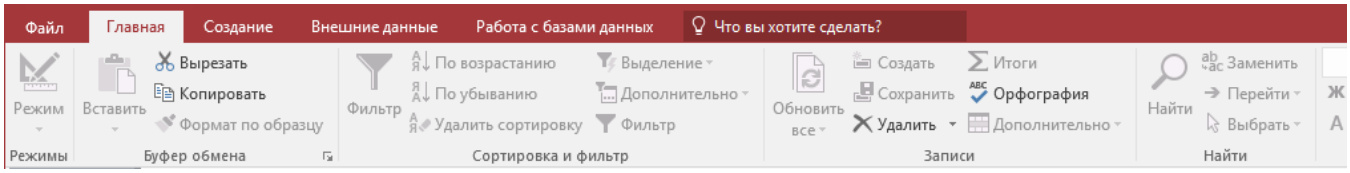
Под базой данных обычно понимают некоторое хранилище информации, включающее в себя множество однотипных элементов с различными свойствами, которые сгруппированы по определенным признакам.

Чтобы приступить к созданию новой базы данных «с нуля» – щелкните мышкой по значку **Новая база данных.** Выбрать одну из недавно открывавшихся баз данных можно по правому краю окна, в области **Открыть последнюю базу данных.** Наконец, чтобы создать новую базу данных на основе шаблона – щелкните мышкой по требуемому шаблону, а затем в правой части окна нажмите появившуюся кнопку **Создать.** Предварительно можно задать имя файла для создаваемой базы данных.

2 Устройство окна Microsoft Office Access

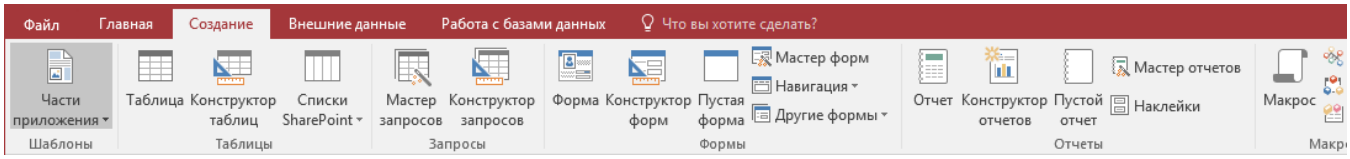
Рассмотрим ленту инструментов и составляющих ее вкладок:

– **Главная** – данная вкладка доступна по умолчанию и содержит команды, позволяющие выбрать режим представления базы данных (режим таблицы или конструктора), вырезать/вставить/скопировать данные с одного места на другое, задать шрифтовой оформление, произвести некоторые основные операции с записями в базе данных, а также фильтрацию и сортировку данных.



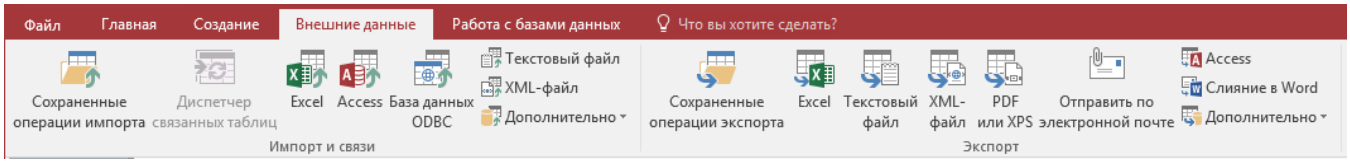
Вкладка Главная

– **Создание** – на этой вкладке размещены команды создания всевозможных элементов/объектов базы данных – таблиц, форм, отчетов и т. п.



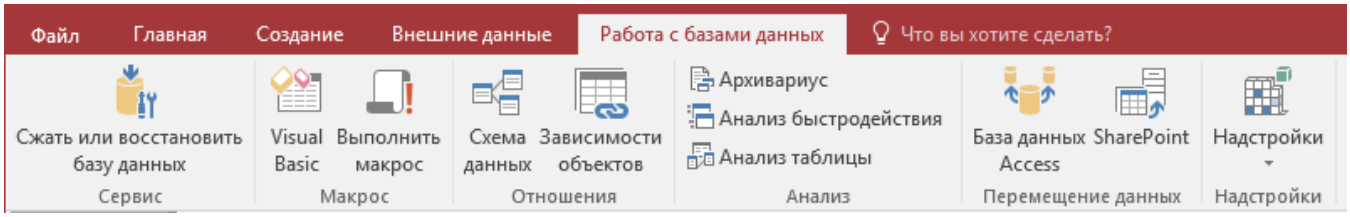
Вкладка Создание

– **Внешние данные** – команды данной вкладки призваны обеспечить преобразование данных из базы данных в, например, таблицы Excel, и наоборот – импорта данных из источников различного происхождения.



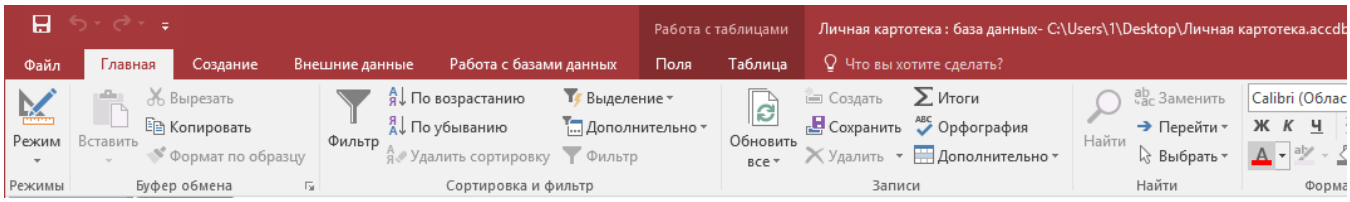
Вкладка Внешние данные

– **Работа с базами данных** – здесь вы найдете команды производства различного рода общих работ с объектами базы данных, такие как команды отображения схемы данных, показа зависимостей между объектами, анализа данных и т. п.



Вкладка Работа с базами данных

– Может еще присутствовать вкладка **Режим таблицы**, появляющаяся при создании таблицы базы данных. Появление на ленте инструментов других всевозможных контекстных вкладок, в зависимости от того, с каким объектом базы данных вы в данный момент работаете.



Вкладка Режим таблицы

3 Создание новой базы данных

Построение любой базы данных Access предусматривает ряд действий, которые должны выполняться в определенной последовательности. Типичный перечень действий пользователя при создании новой базы данных должен включать в себя следующие операции:

1. Проектирование структуры базы данных. Например, если пользователь создает новую базу данных для личного использования, то ему необходимо:

– определить цель создания базы данных (глобальную задачу, решаемую с помощью этой базы данных);

– выбрать набор объектов, информация о которых будет храниться в базе данных;

– составить набор значимых атрибутов каждого хранимого в базе данных объекта;

– составить перечень функций, которые должна выполнять база данных, а также обдумать форму выдачи результатов их выполнения.

2. Создание новой пустой базы данных.
3. Создание таблиц в новой базе данных. Перечень таблиц соответствует набору объектов, информация о которых должна храниться в базе данных. Перечень полей каждой таблицы соответствует набору атрибутов объекта и / или связям этого объекта с другими объектами.
4. Настройка свойств каждой таблицы (установка свойств полей, создание ключей и индексов) и установка связей между таблицами.
5. Заполнение всех созданных таблиц требуемыми данными.
6. Создание дополнительных объектов базы данных (например, форм, запросов или отчетов), которые будут использоваться для реализации требуемых функций базы данных.

Создание таблиц

В программе Access предусмотрены три способа создания таблиц:

- создание новой таблицы путем ввода данных;
- создание новой таблицы с использованием шаблона;
- создание новой таблицы с помощью Конструктора таблиц.

Для создания новой таблицы можно воспользоваться любым из перечисленных способов, в зависимости от того, какой из них в данный момент удобнее пользователю.

Создание таблиц путем ввода данных обычно используют в случае, когда структура таблицы очень проста (например, это таблица-справочник) или пользователь затрудняется сразу определить свойства всех полей новой таблицы. Тогда при создании новой таблицы путем ввода данных программа Access самостоятельно «догадывается» о том, как установить свойства полей. При необходимости позже можно переключить таблицу в режим Конструктора, чтобы посмотреть заданные по умолчанию свойства полей и изменить нежелательные настройки.

Использование встроенных шаблонов для создания новых таблиц оправдано в том случае, если пользователю нужно создать какую-то типовую таблицу, для которой в Access уже разработан профессиональный шаблон.

Режим Конструктора применяется пользователями Access намного чаще двух предыдущих режимов, потому что он предоставляет наибольшие возможности для создания новых таблиц. С помощью Конструктора можно получить новую таблицу любой сложности, обладающую нужными пользователю свойствами. Теперь рассмотрим все эти способы на примере создания таблиц учебной базы данных.

4 Пример построения базы данных в MS Access

С помощью первого метода мы создадим таблицу-справочник **Группы контактов,** в которой будет храниться информация о группах людей, занесенных в вашу личную картотеку.

Таблица **Группы контактов** будет иметь следующую структуру:

- поле **Код группы** – содержит уникальный код группы контактов;
- поле **Наименование группы** – содержит наименование группы контактов;
- поле **Описание** – содержит необязательное описание группы контактов.

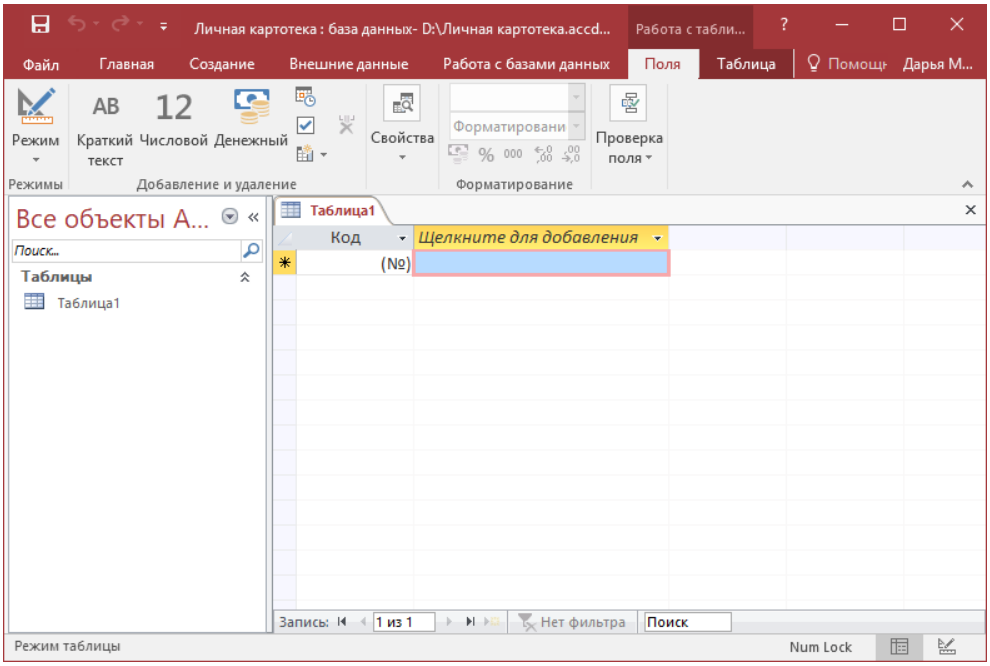
Примерные данные для заполнения этой таблицы приведены ниже (табл. 1).

Таблица 1. Данные для таблицы Группы контактов

Код группы	Наименование группы	Описание
1	Коллеги	Мои коллеги по работе
2	Друзья	Мои близкие друзья
3	Знакомые	Мои знакомы
4	Родственники	Мои родственники

Для того, чтобы создать таблицу **Группы контактов** с помощью ввода данных в пустую табличную форму, выполните следующие действия:

1. Сначала создайте новую базу данных с названием **Личная картотека.** Для этого запустите Access, в разделе **Новая база данных** нажмите кнопку **Новая база** данных, щелкните поле **Имя файла** и наберите на клавиатуре название учебной базы – **Личная картотека**. Затем нажмите кнопку **Создать.** Программа Access создаст учебную базу данных и загрузит рабочую среду, настроенную на создание новой таблицы.



Форма ввода данных в новую таблицу

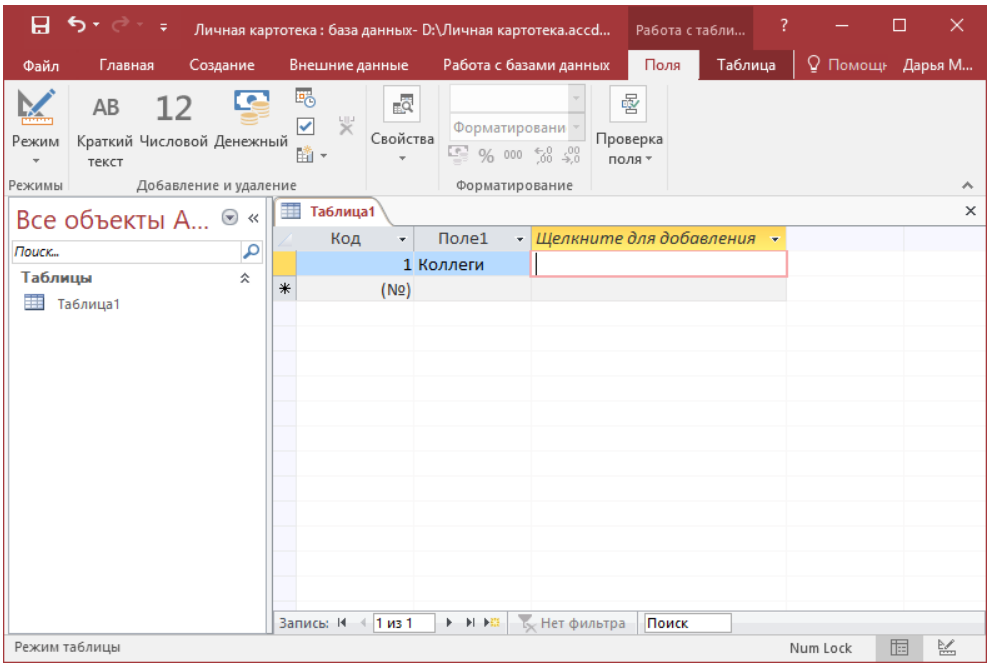
2. Щелкните мышью ячейку под словом **Добавить поле** и введите название первой группы контактов – **Коллеги** (табл. 1.).

3. Нажмите клавишу **Enter,** Microsoft Access автоматически заполнит код группы, присвоит новому полю с наименованием группы имя **Поле1,** добавит в форму еще одну строку и отобразит макет для создания нового поля таблицы.

4. Щелкните мышью свободную ячейку под названием первой группы контактов и введите название второй группы контактов (см.табл. 1). Окончив ввод названия, нажмите клавишу **Enter.**

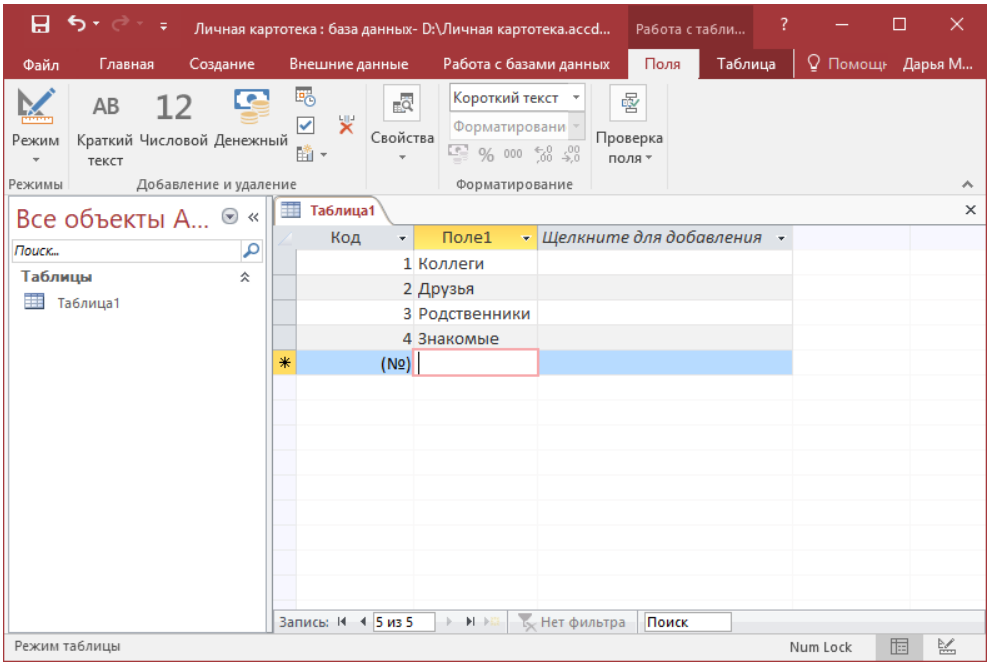
5. Точно также введите все остальные названия групп контактов, указанные в табл. 1. В результате будут заполнены два первых поля таблицы.

6. Щелкните мышью ячейку под словом **Добавить поле** и введите описание первой группы контактов – **Мои коллеги по работе** (табл. 1).



Ввод назначения категории

7. Нажмите клавишу **Enter** Microsoft Access присвоит имя **Поле2** новому полю с описанием группы и отобразит макет для создания еще одного нового поля таблицы. Но в таблице уже есть все необходимые поля, поэтому новое поле создавать не надо.

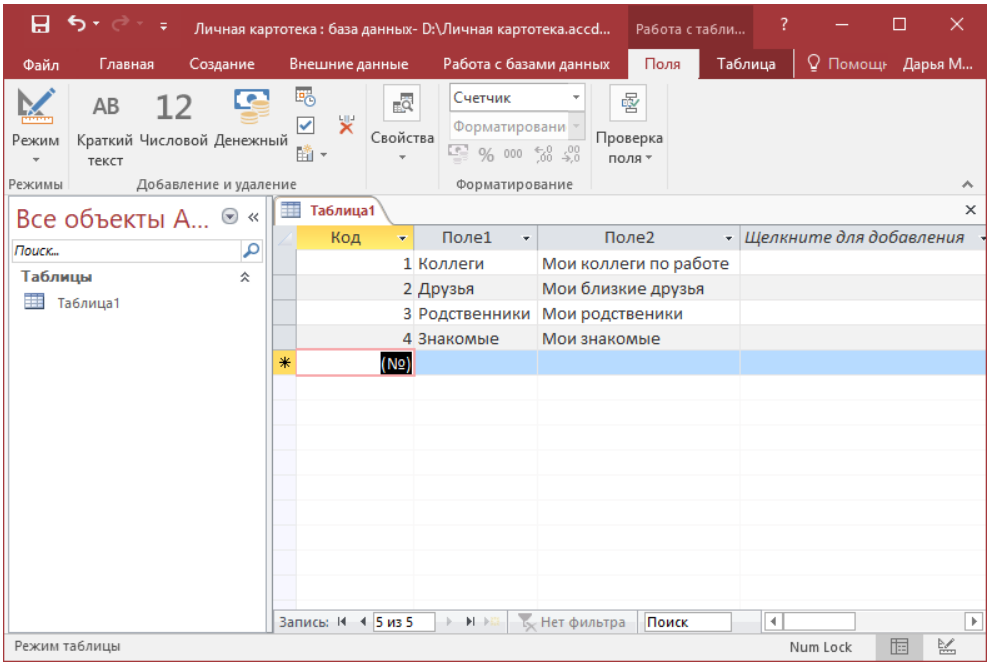


Создание третьего поля

8. Щелкните мышью свободную ячейку под описанием первой группы контактов и введите описание второй группы контактов (табл. 1). Окончив ввод описания, нажмите клавишу **Enter.**

9. Точно так же введите все остальные описания групп контактов (табл. 1). В результате будут заполнены все поля новой таблицы.

10. Щелкните правой кнопкой мыши на заголовке первого столбца таблицы и выберите пункт **Переименовать столбец** в появившемся на экране контекстном меню.



Заполненная таблица

11. Когда стандартное название столбца подсветится, нужно ввести название первого поля таблицы – **Код группы**.

12. Нажмите клавишу **Enter.**

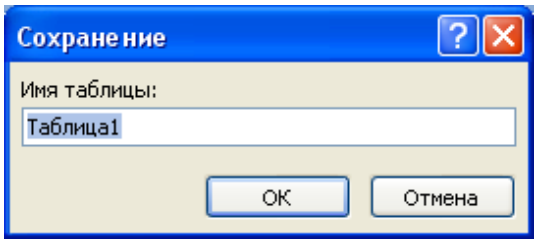
13. Аналогично меняются названия двух других столбцов новой таблицы (табл. 1).

	Код	Наименование группы	Описание	Щелкни
	1	Коллеги	Мои коллеги по работе	
	2	Друзья	Мои близкие друзья	
	3	Родственники	Мои родственники	
	4	Знакомые	Мои знакомые	
*	(№)			

Изменение названий полей таблицы

Теперь изменим установленное по умолчанию имя новой таблицы **(Таблица 1).** Сначала закроем окно таблицы, щелкнув мышью по кнопке **Закрыть** в заголовке окна таблицы. На экране появится окно диалога, предлагающее сохранить сделанные изменения.

15. Нажмите кнопку **Да**, чтобы сохранить все изменения и закрыть окно диалога.
16. Access отобразит диалоговое окно ввода имени для сохранения новой таблицы.



Запрос имени новой таблицы

В поле **Имя таблицы** наберите название новой таблицы – **Группы контактов**, а затем нажмите кнопку **ОК**, чтобы сохранить новую таблицу под указанным именем и закрыть окно диалога. После этого окно таблицы закроется, а в области переходов появится новое название таблицы.

4.1 Использование конструктора таблиц

В случае использования режима **Конструктора** новые таблицы создаются путем задания имен полей, их типов и свойств. Именно таким способом мы создадим вторую таблицу учебной базы данных **Мои контакты.**

В таблице **Мои контакты** будут храниться максимально подробные сведения о тех людях, с которыми вы общаетесь. Эта таблица будет иметь следующую структуру:

- поле **Код контакта** – содержит уникальный код человека, сведения о котором занесены в картотеку;
- поле **Фамилия** – содержит фамилию человека, занесенного в картотеку;
- поле **Имя** – содержит имя человека, занесенного в картотеку;
- поле **Отчество** – содержит отчество человека, занесенного в картотеку;
- поле **Дата рождения** – содержит дату рождения человека, занесенного в картотеку;
- поле **Код группы** – определяет, к какой группе ваших контактов относится этот человек. В этом поле будет храниться код соответствующей ему группы контактов из таблицы **Группы контактов;**
- поле **Телефон** – служит для хранения номера мобильного телефона человека, занесенного в картотеку;
- поле **Электронная почта** – служит для хранения адреса электронной почты человека, занесенного в картотеку;
- поле **Домашний адрес** – служит для хранения домашнего адреса человека, занесенного в картотеку;

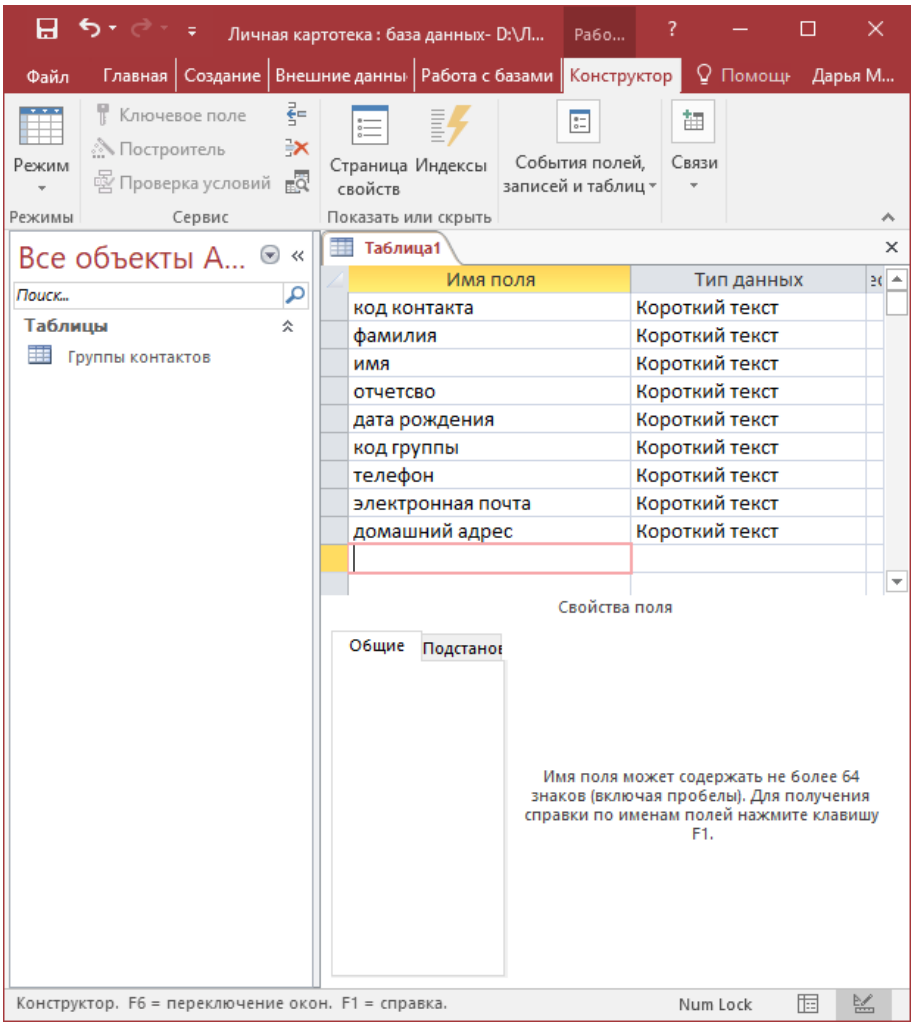
Для того, чтобы создать таблицу **Мои контакты** с помощью Конструктора таблиц, выполним следующие действия:

1. Откройте учебную базу данных **Личная картотека** в Microsoft Access
2. Щелкните заголовок вкладки **Создание** на ленте команд.
3. Щелкните кнопку **Конструктор таблиц,** чтобы запустить создание новой таблицы в режиме **Конструктора**. На экране появится пустая форма **Конструктора таблиц.**
4. В столбце **Имя поля** щелкните верхнюю ячейку и введите название первого поля таблицы – **Код контакта**.

5. Щелкните мышью в пустой ячейке под названием первого поля новой таблицы. В столбце **Тип данных** для первого поля по умолчанию будет установлен тип **Текстовый.**

6. Аналогично введите остальные названия столбцов.

7. Теперь изменим предложенные по умолчанию свойства столбцов новой таблицы. В строке **Код контакта** щелкните слово **Текстовый.** В результате в конце ячейки появится кнопка со стрелкой.

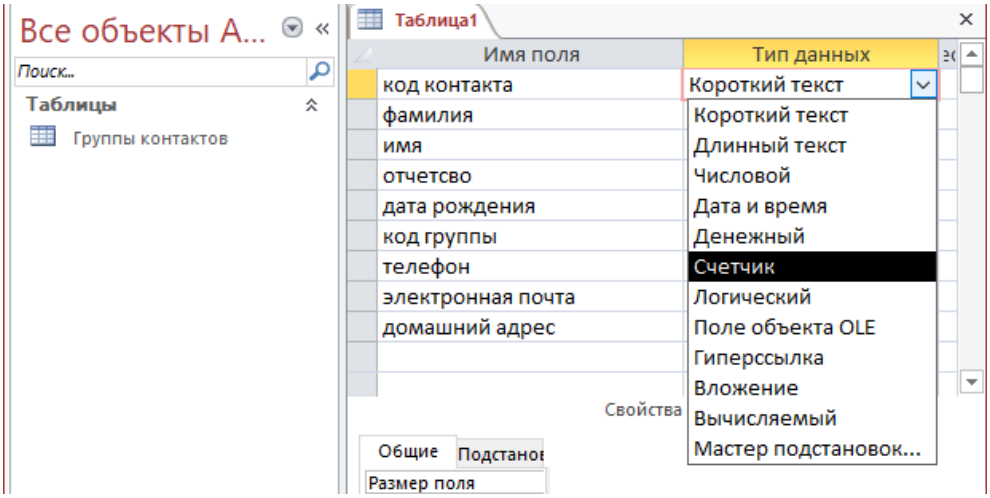


Ввод имен полей таблицы

8. Щелкните кнопку со стрелкой. На экране раскроется список выбора типа данных для поля **Код контакта**.

9. Щелкните вариант **Счетчик.**

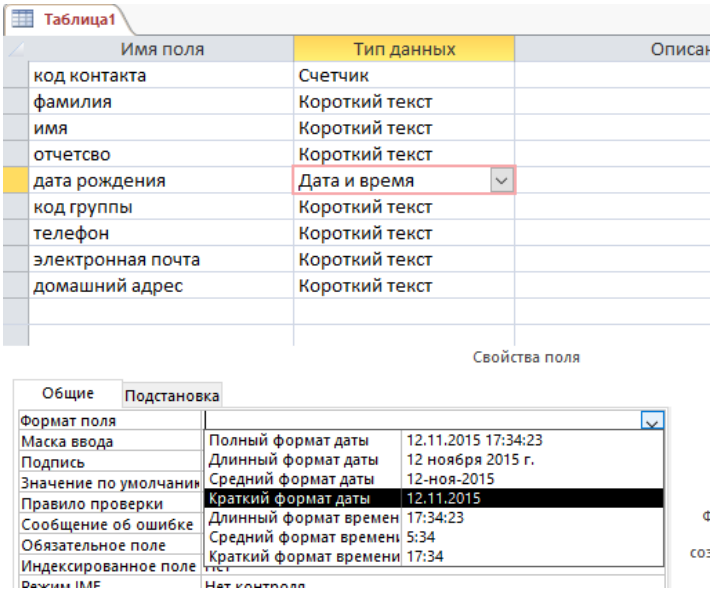
10. Для полей **Фамилия, Имя и Отчество** подходит установленный по умолчанию вариант **Текстовый,** нужно только изменить максимально возможную длину строки для этих полей. По умолчанию устанавливается предел длины в 255 символов. Для поля **Фамилия** будет достаточно 50 символов, а для полей **Имя** и **Отчество** хватит и по 20. Для коррекции длины поля щелкните его название в **Конструкторе**, тогда под списком полей отобразятся свойства выбранного поля. На вкладке **Общие** введите новое значение максимальной длины в поле **Размер поля.** Для поля **Фамилия** введите значение 50 и нажмите клавишу **Enter**. Точно так же установите длину полей **Имя** и **Отчество,** равную 20.



Изменение и выбор типа данных

11. Щелкните поле **Дата рождения,** раскройте меню выбора типа поля и щелкните вариант **Дата/время.** На вкладке *Общие* щелкните в поле **Формат поля.** Когда в конце поля появится кнопка со стрелкой, щелкните ее и выберите опцию **Краткий формат даты** в раскрывшемся меню. В этом случае дата рождения будет отображаться в привычном кратком формате.

12. Щелкните поле **Код группы** и выберите для него тип **Числовой.** С помощью этого поля вы свяжите таблицу **Мои контакты с** таблицей **Группы контактов.**



Выбор формата поля

13. Для поля **Телефон и Электронная почта** оставьте тип **Текстовый**, но измените максимальную длину. У каждого из этих четырех полей сначала щелкните название поля, затем перейдите на вкладку **Общие** и введите в поле **Размер поля** значение 20.

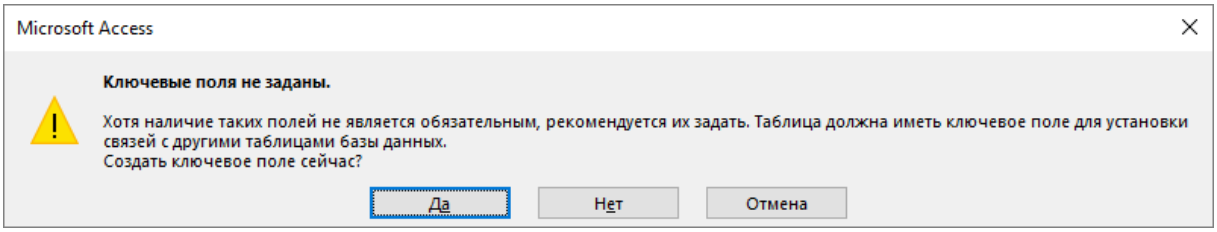
14. Для поля **Домашний адрес** оставьте тип **Текстовый** с длиной 255 символов (полные почтовые адреса могут быть очень длинными).

15. При необходимости введите описание каждого поля таблицы в столбце **Описание.** Описания полей нужны для того, чтобы пользователь не запутался в назначении полей таблицы, особенно если таблица очень большая и в качестве имен полей используются англоязычные сокращения. Поскольку при создании таблицы мы использовали понятные названия полей, вводить их описания необязательно.

16. Для сохранения таблицы нажмите кнопку **Сохранить** на панели быстрого доступа.

17. Access отобразит диалоговое окно ввода имени для сохранения новой таблицы. В поле **Имя таблицы** введите название для новой таблицы – **Мои контакты**. После этого нажмите кнопку **ОК,** чтобы сохранить новую таблицу в учебной базе данных и закрыть окно диалога.

18. При сохранении таблицы Microsoft Access выдаст предупреждение о том, что в таблице не заданы ключевые поля, и предложит создать их автоматически.



Окно предупреждения Access

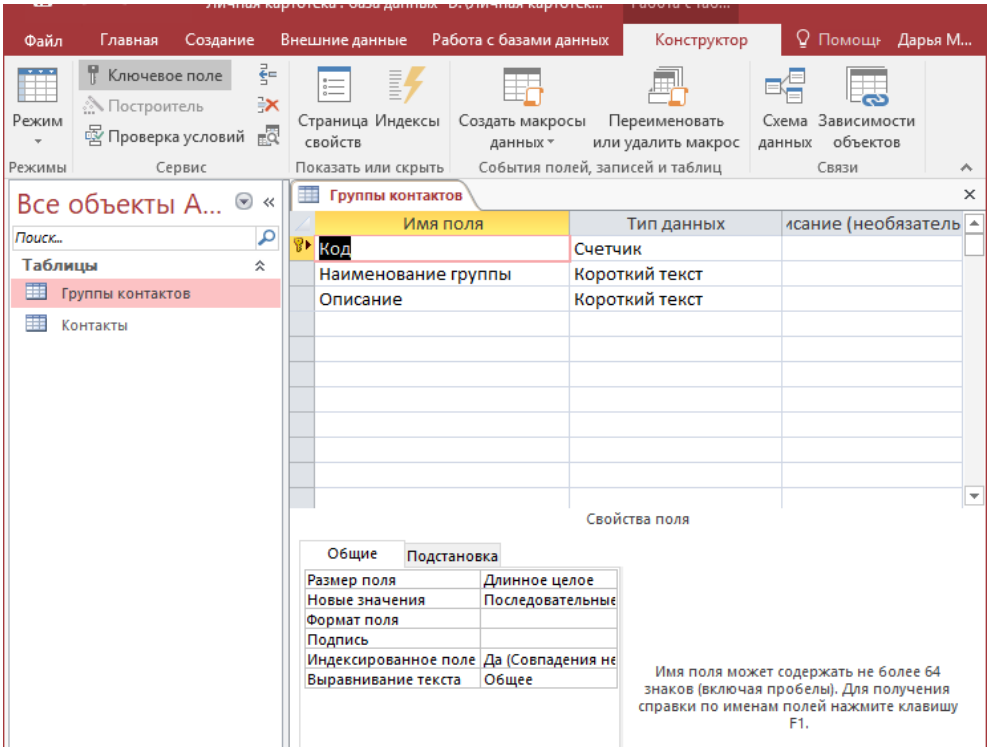
Ключевые поля таблицы вы создадите позже вручную, поэтому нажмите кнопку **Нет** для сохранения таблицы без ключевых полей.

20. Щелкните кнопку **Закрыть** «*Мои контакты*», чтобы закрыть сохраненную таблицу и выйти из режима **Конструктора**.

4.2 Изменение таблицы. Свойства полей таблицы

Для того, чтобы просмотреть или изменить свойства полей таблицы базы данных, выполните следующие действия:

1. Откройте базу данных Access (например, учебную базу данных **Личная картотека**).
2. В области переходов дважды щелкните название таблицы, свойства полей которой вы хотите просмотреть (например, для просмотра таблицы **Группы контактов** используйте ссылку **Группы контактов:** *таблица*). На экране появится окно выбранной таблицы.
3. Перейдите в режим **Конструктора**. Для этого откройте вкладку **Главная** на ленте инструментов и щелкните по стрелке под кнопкой **Режим.** На экране раскроется меню выбора режима работы с таблицей.
4. Щелкните команду **Конструктор.** Состав полей открытой вами таблицы отобразится в окне Конструктора таблиц.
5. В столбце **Имя поля** щелкните по названию любого поля таблицы (например, щелкните **Наименование группы).** Тип данных выбранного поля указан напротив названия поля, а дополнительные свойства поля отобразятся в области **Свойства поля**.



Свойства поля таблицы

6. Чтобы изменить какое-либо свойство поля, щелкните ячейку напротив названия свойства и укажите новое значение. Например, чтобы установить максимальную длину наименования группы контактов в 20 символов, на вкладке **Общие** щелкните ячейку напротив свойства **Размер поля**, введите в нее значение 20 и нажмите клавишу **Enter**.

4.3 Формат отображения данных, задание маски ввода

Формат отображения данных в каком-либо поле таблицы определяет, как будет выглядеть введенное в это поле значение при просмотре таблицы.

Формат отображения данных в поле таблицы указывается с помощью свойства **Формат поля**, которое доступно при просмотре таблицы в режиме **Конструктора**. Свойство **Формат поля** влияет на отображение данных в формах и отчетах.

Для каждого типа данных поля (например, чисел, текста, даты/времени) в Access предусмотрен свой набор встроенных стандартных форматов.

Свойство **Маска ввода** позволяет проконтролировать ввод данных в таблицу и максимально упростить процесс ввода для пользователя. **Маску ввода** используют в том случае, когда данные таблицы должны содержать определенные символы в некоторых позициях строки, вводимой пользователем с клавиатуры. Самым распространенным примером таких данных являются номера телефонов. **Маска ввода** должна обеспечить возможность вводить только цифры номера, а остальные символы (скобки вокруг кода города, дефис между цифрами номера) будут добавляться автоматически.

Маска ввода состоит из четырех частей, которые разделяются точкой с запятой. Первая часть является обязательной, а две остальные – необязательными. Например, **Маска ввода** для телефонных номеров может выглядеть следующим образом: (999) 00-00-00; 0.

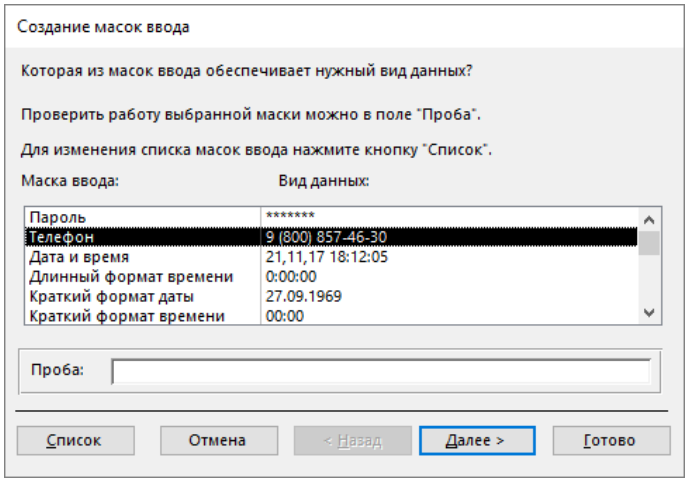
- **Первая часть** – определяет строку маски и состоит из местозаполнителей и текстовых констант.
- **Вторая часть** – указывает, следует ли сохранять знаки маски вместе с данными в базе данных. Нужно использовать «0», если в базе данных необходимо сохранить маску и данные. Если необходимо сохранить только данные, то следует поставить «1».
- **Третья часть** – определяет местозаполнитель, который служит для обозначения позиции данных. По умолчанию в **Масках ввода** Access используется знак подчеркивания – «_». Если вам нужно использовать другой знак, введите нужный символ в третью часть маски.

При задании первой части **Маски ввода** необходимо использовать специальные символы, приведенные ниже:

- **0** – в данную позицию должна быть введена цифра, знак плюс (+) и минус (-) не допускаются;
- **9** – в данную позицию должна быть введена цифра или пробел, знаки плюс (+) и минус (-) не допускаются;
- **#** – в данную позицию должна быть введена цифра, пробел, знаки плюс (+) или минус (-);
- **L** – в данную позицию должна быть введена буква;
- **?** – в данную позицию может быть введена буква или пробел;
- **A** – в данную позицию должна быть введена буква или цифра;
- **a** – в данную позицию должна быть введена буква, цифра или пробел;
- **&** – в данную позицию должен быть введен произвольный символ или пробел;
- **C** – в данную позицию может быть введен произвольный символ или пробел;
- **.** – десятичный разделитель (зависит от региональных установок Windows);
- **,** – разделитель групп разрядов (зависит от региональных установок Windows);
- **/– :** – разделитель в значениях даты и времени;
- **<** – преобразует все символы справа к нижнему регистру;
- **>** – преобразует все символы справа к верхнему регистру;
- **|** – указывает, что маску нужно заполнять справа налево. Этот символ следует использовать в случае, когда символы в левой части маски являются необязательными;
- **** – указывает, что следующий символ необходимо рассматривать в качестве постоянного символа, даже если он является специальным символом маски. Например, \ A будет выводить в маске букву A;
- **«символы»** – вместо того, чтобы многократно использовать обратный слеш (\) для нескольких стоящих подряд постоянных символов, можно просто заключить эту последовательность символов в двойные кавычки.

Для того, чтобы установить **Маску ввода** для поля таблицы, нужно выполнить следующие действия:

1. Перейти в режим **Конструктора**.
2. В столбце **Имя поля** щелкните по названию нужного поля таблицы (например, Телефон).
3. На вкладке **Общие** просмотрите свойства этого поля таблицы и найдите свойство **Маска ввода**.
4. Щелкните поле ввода, расположенное напротив надписи **Маска ввода**.
5. Введите требуемую **Маску ввода** для этого поля. Например, для ввода пяти или шестизначного номера телефона с необязательным кодом города и символом-заполнителем в виде звездочки наберите маску ввода (999) -90-00-00;0;*.



6. Нажмите кнопку *Сохранить* на панели быстрого доступа, чтобы сохранить установленную для поля **Маску ввода**.
7. Переключитесь в режим таблицы, чтобы проверить работу новой маски ввода. Когда таблица отобразится на экране, попробуйте ввести какое-либо значение в то поле, для которого вы задали маску.

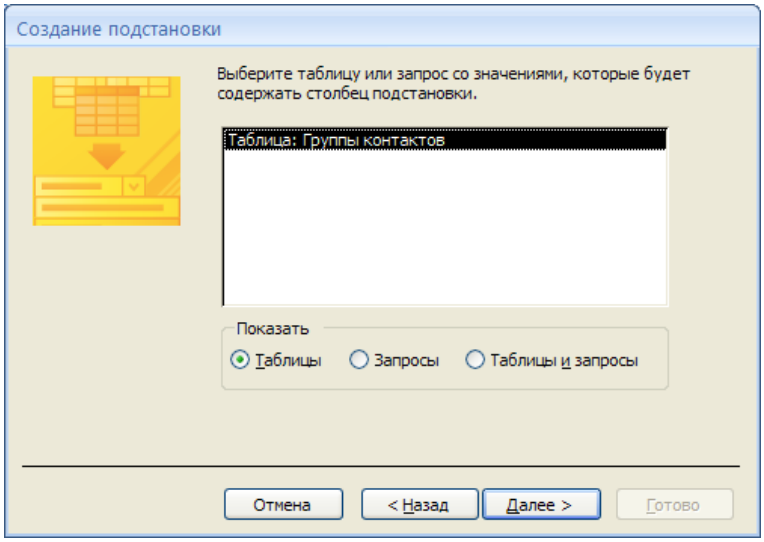
4.4 Использование Мастера подстановок

Используя операцию подстановки, можно просто выбирать значения поля из списка вместо того, чтобы вводить их с клавиатуры. Список значений может быть либо фиксированным, либо содержаться в таблице или запросе.

Чтобы сформировать столбец подстановок для поля таблицы, можно использовать **Мастер подстановок**. В качестве примера создадим столбец подстановок для поля **Код группы** в таблице **Мои контакты**.

Для того, чтобы создать столбец подстановки для поля с помощью Мастера подстановок, выполните следующие действия:

1. Откройте базу данных **Личная картотека**.
2. В области переходов выберите таблицу **Мои контакты**.
3. Перейдите в режим **Конструктора**.
4. В столбце **Тип данных** щелкните по ячейке напротив названия **Код группы**.
5. Щелкните строку **Мастер подстановок** в раскрывшемся меню. На экране появится окно **Мастер подстановок**.



Окно Мастер подстановок

6. По умолчанию в окне **Мастера** отмечен нужный нам вариант создания столбца подстановки (из таблицы или запроса), поэтому просто нажимаем кнопку **Далее**. На экране появится следующее окно **Мастера**

7. В этом окне требуется указать источник значений столбца подстановки. По умолчанию выбрана правильная таблица-источник **Группы контактов**, поэтому нажимаем **Далее.**

8. В следующем окне **Мастера** нужно указать поле таблицы-источника, которое содержит данные для столбца подстановки. В списке **Допустимые поля** щелкните на поле **Наименование группы** и нажмите кнопку  , чтобы перенести его в список **Выбранные поля.** Нажмите кнопку **Далее.**

9. В следующем окне предлагается выбрать режим сортировки значений в столбце подстановки. **Раскройте список 1** и щелкните в нем строку **Наименование группы**. Нажмите кнопку **Далее.**

10. В следующем окне предлагается посмотреть, как будет выглядеть столбец подстановок. По умолчанию столбец с кодами элементов подстановки не отображается. Чтобы вывести коды элементов столбца подстановки, снимите отметку флажка **Скрыть ключевой столбец.**

11. Нажмите кнопку **Далее**. На экране появится последнее окно **Мастера,** в котором предлагается выбрать подпись для столбца подстановки. По умолчанию предложено имя **Код группы.**

12. Нажмите кнопку **Готово**, чтобы завершить работу **Мастера подстановок.**

13. Переключитесь в режим таблицы, чтобы проверить работу столбца подстановки. Когда таблица отобразится на экране, щелкните в любой ячейке столбца **Код группы**. В ячейке появится кнопка раскрытия списка, в котором будут перечислены наименования групп контактов.

4.5 Определение ключевых полей

Ключевое поле – это одно или несколько полей, комбинация значений которых однозначно определяет каждую запись в таблице. Если для таблицы определены ключевые поля, то Microsoft Access предотвращает дублирование или ввод пустых значений в ключевое поле. Ключевые поля используются для быстрого поиска и связи данных из разных таблиц при помощи запросов, форм и отчетов.

В Microsoft Access можно выделить три типа ключевых полей:

- счетчик;
- простой ключ;
- составной ключ.

Поле-счетчик характеризуется тем, что Access автоматически заполняет его последовательными или случайными числами без какого-либо участия пользователя. Чтобы создать *ключевое поле-счетчик,* нужно определить поле с типом данных **Счетчик** и выбрать его в качестве ключа.

Для создания *простого ключа* достаточно иметь поле, которое содержит уникальные значения (например, номера или коды). Если выбранное поле содержит повторяющиеся или пустые значения, его нельзя определить как ключевое. Для определения записей, содержащих повторяющиеся данные, можно выполнить запрос на поиск повторяющихся записей. Если устранить повторы путем изменения значений

невозможно, следует либо добавить в таблицу поле счетчика и сделать его ключевым, либо определить составной ключ.

Составной ключ необходим в случае, если невозможно гарантировать уникальность записи с помощью одного поля. Он представляет собой комбинацию нескольких полей. Для определения составного ключа необходимо сделать ключевыми те поля таблицы, которые должны входить в состав ключа.

В качестве примера определим ключевые поля в таблицах учебной базы данных. В таблице **Группы контактов** ключевое поле **Код группы** было создано автоматически. А сейчас мы определим ключевое поле для таблицы **Мои контакты.** Им будет поле **Код контакта,** однозначно идентифицирующее человека, внесенного в вашу личную картотеку.

Для того, чтобы определить ключевое поле в таблице базы данных, выполните следующие действия:

1. Откройте в Access учебную базу данных **Личная картотека.**
2. В области переходов дважды щелкните таблицу **Мои контакты.**
3. Перейдите в режим Конструктора.
4. В столбце **Имя поля** щелкните название поля **Код контакта.**

Щелкните контекстную **вкладку Работа с таблицами** на ленте команд, чтобы отобразить команды работы с таблицей в режиме Конструктора.

5. В группе **Сервис** нажмите кнопку **Ключевое поле.** После этого выбранное поле таблицы станет ключевым и около него появится значок в виде желтого ключа.

Создание ключевого поля

6. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить сделанные изменения, закройте таблицу.

4.6 Создание индексов

С целью ускорения поиска и сортировки данных в любой СУБД используются индексы. *Индекс* – это специальное средство, обеспечивающее быстрый доступ к данным в таблице на основе значений одного или нескольких столбцов.

По существу, индекс представляет собой упорядоченный список значений и ссылок на те записи, в которых хранятся эти значения. Чтобы найти нужные записи, СУБД сначала ищет требуемое значение в индексе, а затем по ссылкам быстро отбирает соответствующие записи.

Индексы бывают двух типов:

– **простые индексы** индексы, созданные по одному столбцу таблицы (например, по полю **Мобильный телефон** в таблице Мои контакты);

– **составные** индексы, построенные по нескольким столбцам таблицы (например, по полям **Фамилия** и **Имя** в таблице **Мои контакты).**

Индексировать можно любые поля таблицы, кроме имеющих тип данных **Поле MEMO, Гиперссылка и Поле объекта OLE.** Каждое ключевое поле таблицы индексируется автоматически.

В качестве примера создадим в таблице **Мои контакты** учебной базы данных простой индекс по полю **Мобильный телефон** и составной по полям **Фамилия и Имя.**

Для того, чтобы создать индекс в таблице базы данных, выполните следующие действия:

1. Откройте в Access учебную базу данных **Личная картотека.**
2. В области переходов дважды щелкните таблицу **Мои контакты.**

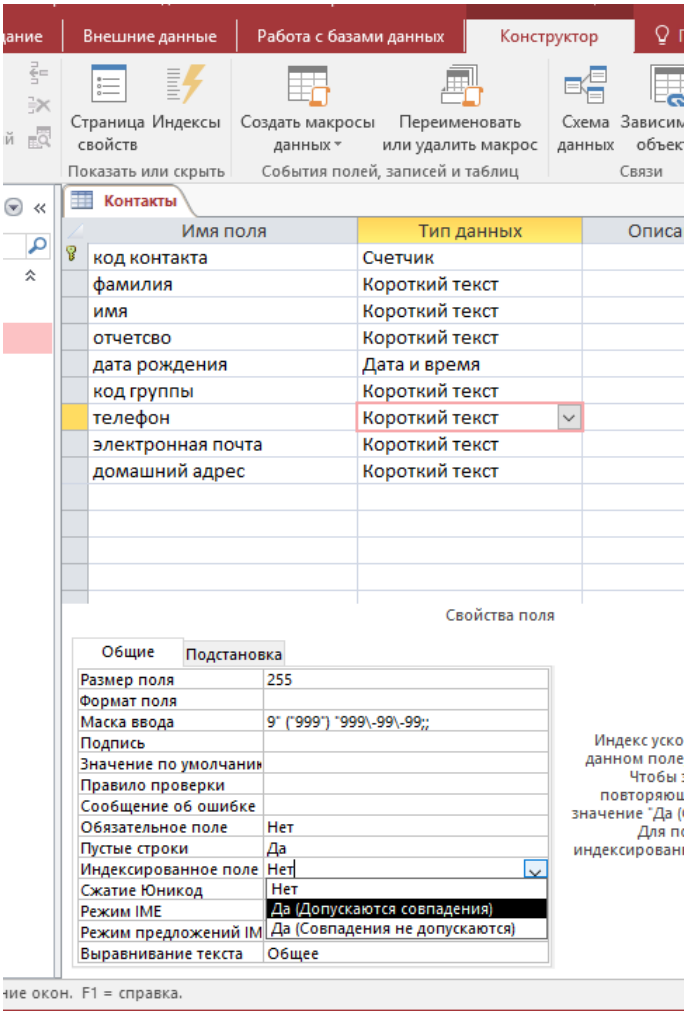
3. Перейдите в режим **Конструктора**. Состав полей таблицы **Мои контакты** отобразится в окне **Конструктора таблиц**.

4. Для установки индекса по полю **Телефон** в столбце **Имя поля** щелкните название поля **Телефон.**

5. На вкладке **Общие** щелкните поле ввода после свойства **Индексированное поле.**

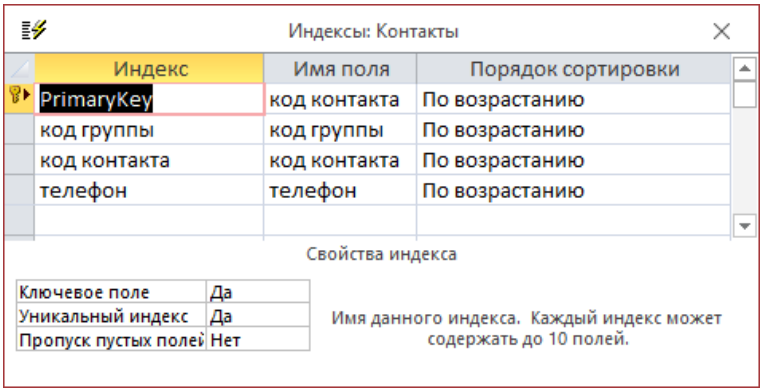
6. Щелкните кнопку со стрелкой, появившуюся в конце поля.

7. Выберите вариант **Да (Совпадения не допускаются)** в раскрывшемся списке. Этот вариант был выбран потому, что по логике вещей совпадения номеров мобильных телефонов у разных людей невозможны. Если некоторые ваши знакомые совместно используют один номер мобильного телефона, выберите вариант **Да (Допускаются совпадения).**



Создание простого индекса

8. Для создания индекса по полям **Фамилия** и **Имя** щелкните контекстную вкладку **Работа с таблицами** на ленте инструментов и в группе **Показать или скрыть** нажмите кнопку **Индексы**. На экране появится диалоговое окно **Индексы: Мои контакты**.

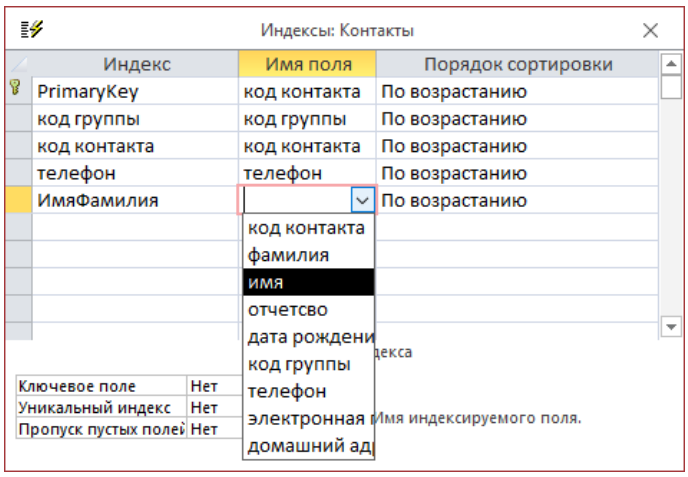


Создание составного индекса

9. Щелкните первую пустую ячейку в столбце **Индекс** и введите название нового индекса (например, **Имя Фамилия**).

10. В той же строке щелкните ячейку столбца **Имя поля.**

11. Щелкните кнопку со стрелкой, появившуюся в конце ячейки. На экране раскроется список имен полей таблицы **Мои контакты**.



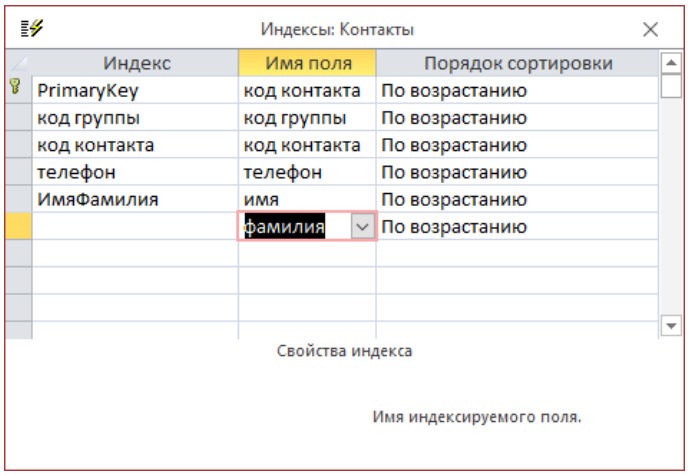
Выбор поля составного индекса

12. Щелкните строку **Имя**, чтобы выбрать первое поле для создания составного индекса.

13. Щелкните следующую свободную ячейку столбца **Имя поля.**

14. Щелкните кнопку со стрелкой, появившуюся в конце ячейки. На экране раскроется список имен полей таблицы **Мои контакты**.

15. Щелкните строку **Фамилия**, чтобы выбрать второе поле для создания составного индекса. Соответствующая ячейка в столбце **Индекс** должна остаться пустой.



Готовый составной список

16. Закройте окно диалога **Индексы: Мои контакты,** щелкнув по кнопке **Закрыть** в заголовке окна.

17. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить сделанные изменения.

18. Щелкните кнопку **Закрыть** в заголовке окна таблицы, чтобы закрыть таблицу и выйти из режима **Конструктора.**

4.7 Связывание таблиц на схеме данных

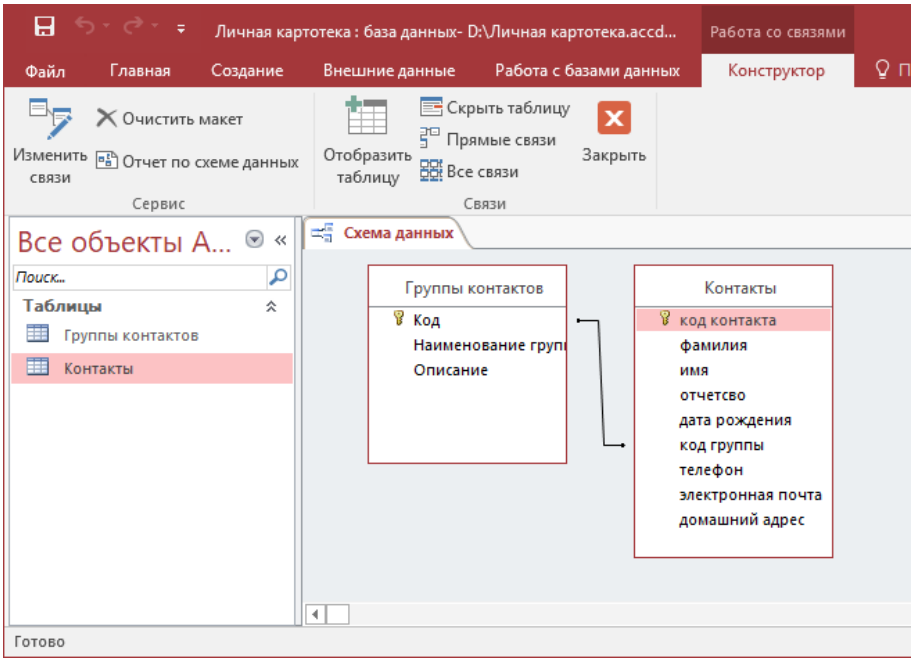
При работе со сложной базой данных пользователю часто требуется просматривать, создавать, удалять и модифицировать связи между таблицами базы данных. Все эти действия удобно выполнять с помощью схемы данных.

В качестве примера работы со схемой данных рассмотрим установку связи между таблицами

учебной базы данных **Личная картотека.**

Для того, чтобы связать таблицы базы данных на схеме данных, выполните следующие действия:

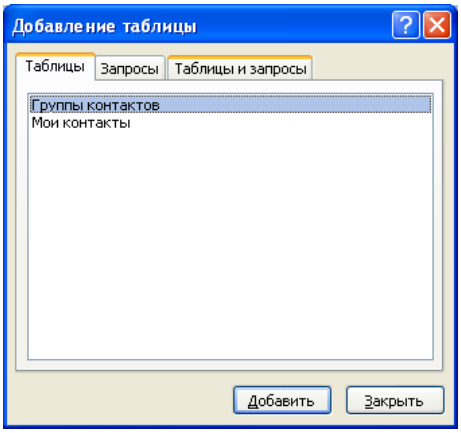
1. Откройте в Access учебную базу данных **Личная картотека.**
2. Щелкните вкладку **Работа с базами данных** на ленте инструментов.
3. В группе **Показать или скрыть** щелкните кнопку **Схема данных.**
4. На экране откроется окно схемы данных, а на ленте инструментов появится контекстная вкладка **Работа со связями**.



Режим схемы данных

Для учебной базы данных схема данных была сгенерирована автоматически. В окне схемы данных уже находятся обе таблицы базы данных и между ними установлена связь по полю **Код группы.**

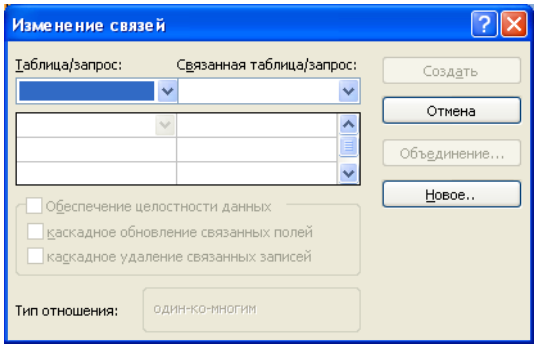
Если вам нужно добавить в схему данных какой-либо отсутствующий в ней объект базы данных (например; таблицу или запрос), щелкните контекстную вкладку **Работа со связями** и в группе **Связи** нажмите кнопку **Отобразить таблицу.** На экране появится диалоговое окно **Добавление таблицы.**



Диалоговое окно Добавление таблицы

Щелкните вкладку, соответствующую типу добавляемого объекта, а затем выберите нужный объект базы данных. После этого нажмите кнопку **Добавить. Окно** диалога закроется, а новый объект появится на схеме данных.

6. Чтобы изменить связи между объектами на схеме данных, щелкните контекстную вкладку **Работа со связями** и в группе **Сервис** нажмите кнопку **Изменить связи.** На экране появится диалоговое окно **Изменение связей.**



Диалоговое окно «Изменение связей»

Выберите связываемые объекты и поля, которыми они связаны в базе данных. После этого нажмите кнопку **ОК** для установки связи между этими объектами по указанным полям.

7. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить сделанные изменения.

8. Для выхода из режима работы со схемой данных щелкните кнопку **Закрыть** на контекстной вкладке **Работа со связями**.

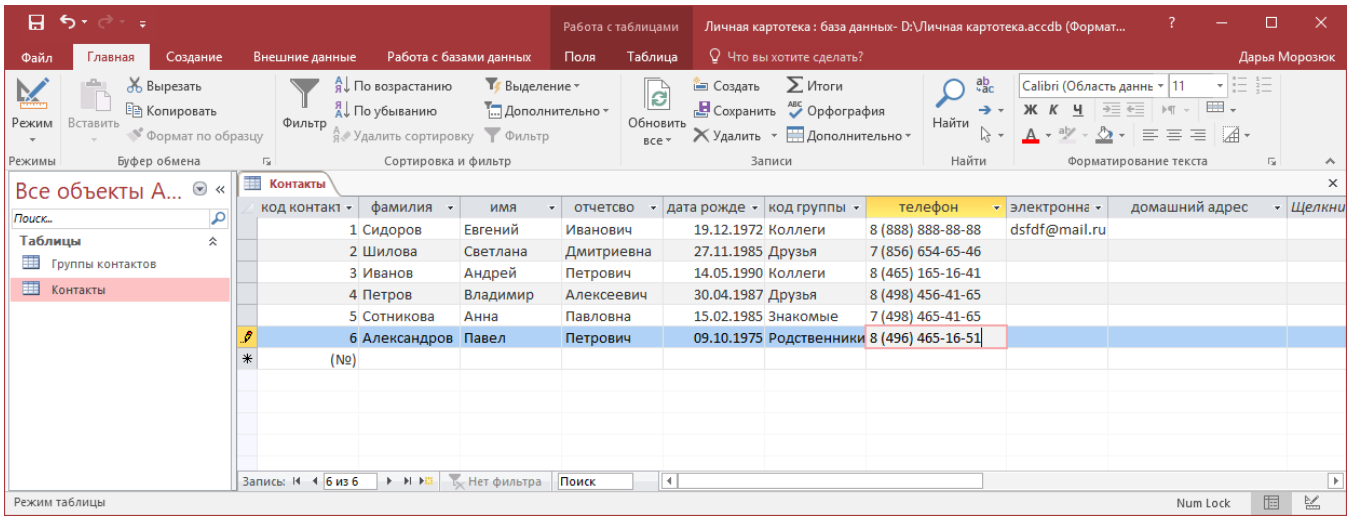
4.8 Ввод данных

Для того, чтобы ввести данные в таблицу базы данных, выполните следующие действия:

1. Откройте в Access учебную базу данных **Личная картотека.**
2. В области переходов дважды щелкните таблицу **Мои контакты.**

Выбранная таблица будет открыта в режиме таблицы.

3. Щелкните мышью ту ячейку, в которую вы хотите ввести данные. При вводе значений в последнюю строку таблицы новая строка будет добавлена автоматически.
4. Наберите на клавиатуре содержимое ячейки таблицы. При этом вы можете использовать стандартные операции копирования и вставки.
5. Для примера в таблицу Мои контакты были внесены семь записей



Данные таблицы «Мои контакты»

6. Чтобы изменить устаревшие или ошибочные данные в какой-либо ячейке таблицы, щелкните эту ячейку и введите новое значение.

4.9 Навигация по таблице

Для перемещения по записям таблицы используется панель навигации, расположенная на нижней границе окна таблицы.

Панель навигации

Для навигации по таблице базы данных выполните следующие действия:

1. Откройте в Access вашу базу данных (например, учебную базу данных **Личная картотека).**
2. В области переходов дважды щелкните нужную таблицу (например, таблицу **Мои контакты).** Выбранная таблица будет открыта в режиме таблицы.
3. Для перемещения по записям таблицы выполните нужные вам действия из предложенных ниже вариантов:

– чтобы перейти к первой записи таблицы, щелкните – **Первая запись** на панели навигации;

– чтобы перейти к предыдущей записи таблицы, щелкните кнопку – **Предыдущая запись** на панели навигации;

– чтобы перейти к следующей записи таблицы, щелкните кнопку – **Следующая запись** на панели навигации;

– чтобы перейти к последней записи таблицы, щелкните кнопку – **Последняя запись** на панели навигации.

4. Чтобы добавить в таблицу новую строку, щелкните кнопку – **Новая (пустая) запись** на панели навигации.

5 Формы баз данных

5.1 Работа с формами

Формы представляют собой настраиваемые диалоговые окна, которые являются специализированными объектами базы данных (как таблицы или запросы) и сохраняются в файле базы данных.

Форма – это объект базы данных, который можно использовать для ввода, изменения или отображения данных из таблицы или запроса. Формы можно рассматривать как окна, через которые пользователи могут просматривать и изменять базу данных. Рационально построенная форма ускоряет работу с базой данных, поскольку пользователям не требуется вручную искать то, что им нужно.

Далее мы подробно рассмотрим все перечисленные выше способы создания новых форм на примере создания форм для таблиц учебной базы данных **Личная картотека.** Начнем со способов автоматического создания формы.

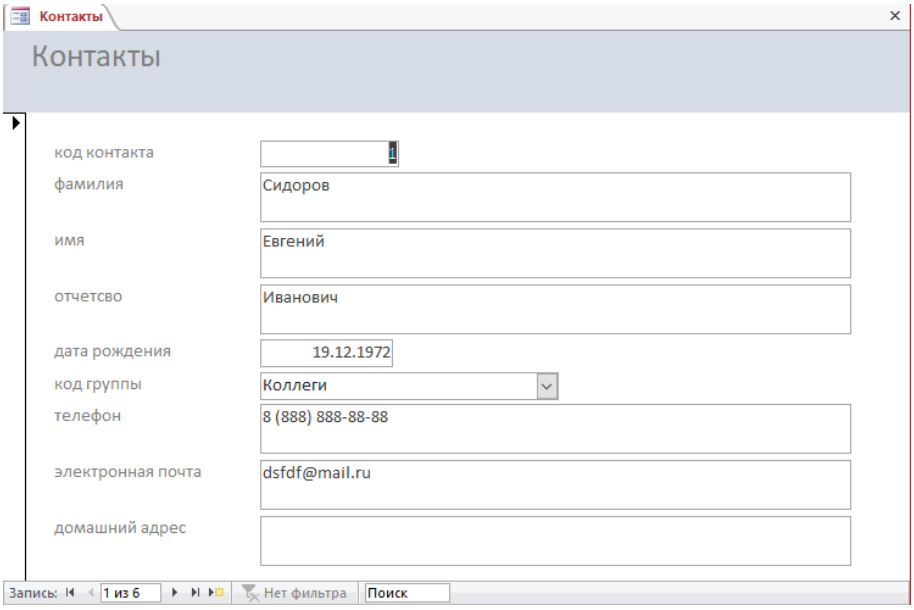
Самыми простыми являются первые два способа создания форм – автоматическое создание простой и разделенной формы. С помощью любого из этих способов можно создать форму всего одним щелчком мыши.

Чтобы автоматически создать простую форму для существующей таблицы базы данных, нужно воспользоваться инструментом **Форма.** При использовании этого средства в новой форме размещаются все поля выбранной таблицы базы данных. Сразу после создания новой формы можно начать ее использование либо при необходимости изменить ее в режиме **Макета** или **Конструктора**.

Для того, чтобы автоматически создать простую форму, выполните следующие действия:

1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. В области переходов щелкните имя таблицы, для которой вы хотите создать форму (например, **Группы контактов).**
3. Щелкните вкладку **Создание**.
4. В группе **Формы** щелкните кнопку **Форма.** Приложение Access создаст форму для выбранной таблицы и отобразит ее в режиме макета.

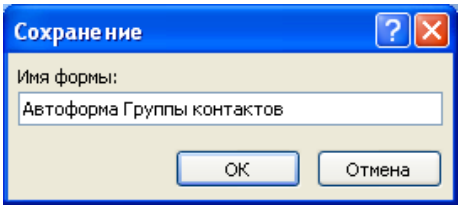
В режиме макета можно внести изменения в структуру формы при одновременном отображении данных, содержащихся в таблице. Например, при необходимости можно настроить размер полей формы в соответствии с вводимыми в таблицу данными.



Простая форма

5. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новую форму в базе данных. На экране появится окно запроса имени новой формы.

6. Введите желаемое название формы в поле **Имя формы** (например, *Автоформа Группы контактов*) и нажмите кнопку **ОК**. Название новой формы появится в области переходов.



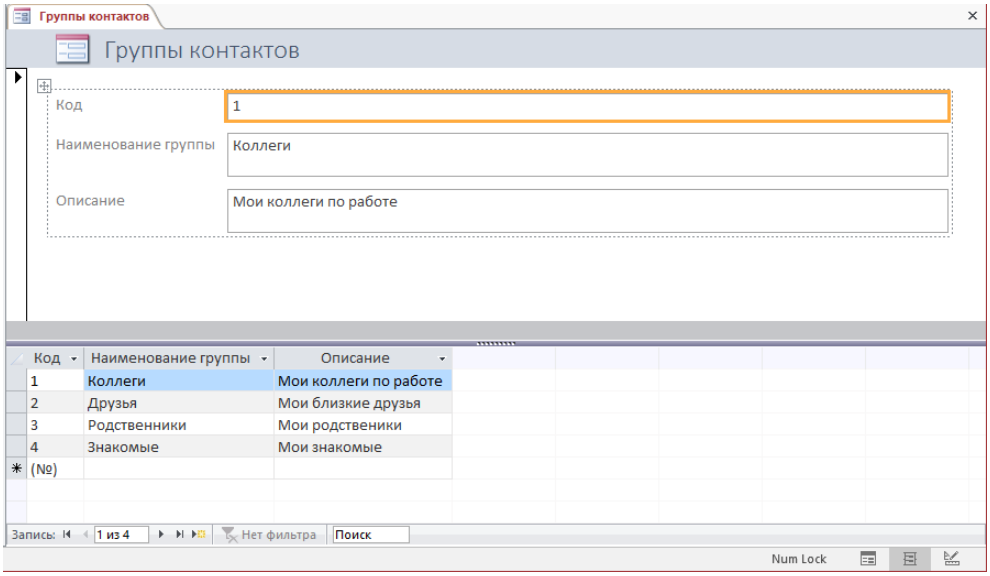
Ввод имени формы

5.2 Создание разделенной формы

Отличие разделенной формы от простой состоит в том, что она отображает данные, содержащиеся в таблице, одновременно в двух представлениях – в режиме формы и в режиме таблицы.

Эти два представления связаны с одним и тем же источником данных (таблицей базы данных) и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в каждой части формы (при условии, что источник записей допускает обновление, а параметры формы не запрещают такие действия).

Преимущества использования разделенной формы обусловлены тем, что в такой форме пользователь может попеременно работать с двумя разными типами представления данных. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы.



Разделенная форма

Для того, чтобы автоматически создать разделенную форму, выполните следующие действия:

1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. В области переходов щелкните имя таблицы, для которой вы хотите создать форму (например, **Группы контактов).**
3. Щелкните вкладку **Создание**.
4. В группе **Формы** щелкните кнопку **Разделенная форма.** Приложение Access создаст форму для выбранной таблицы и отобразит ее в режиме макета.

В режиме макета можно внести изменения в структуру формы при одновременном отображении данных, содержащихся в таблице. Например, при необходимости можно настроить размер полей формы в соответствии с вводимыми в таблицу данными.

5. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новую форму в базе данных. На экране появится окно запроса имени новой формы.
6. Введите желаемое название формы в поле **Имя формы** (например, *Разделенная форма Группы контактов*) и нажмите кнопку **ОК.** Название новой формы появится в области переходов.

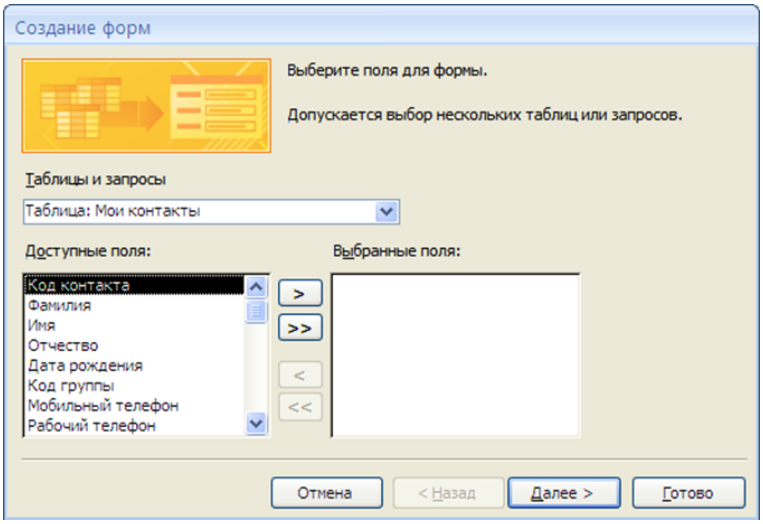
5.3 Создание формы с помощью мастера

Для получения большей свободы выбора отображаемых на форме полей вместо рассмотренных выше инструментов автоматического создания форм можно воспользоваться **Мастером форм**.

В случае использования **Мастера форм** при создании новой формы можно указать способ группировки и сортировки данных, а также включить в форму поля из нескольких таблиц или запросов при условии, что заранее заданы отношения между этими таблицами и запросами.

Для того, чтобы создать новую форму с помощью **Мастера**, выполните следующие действия:

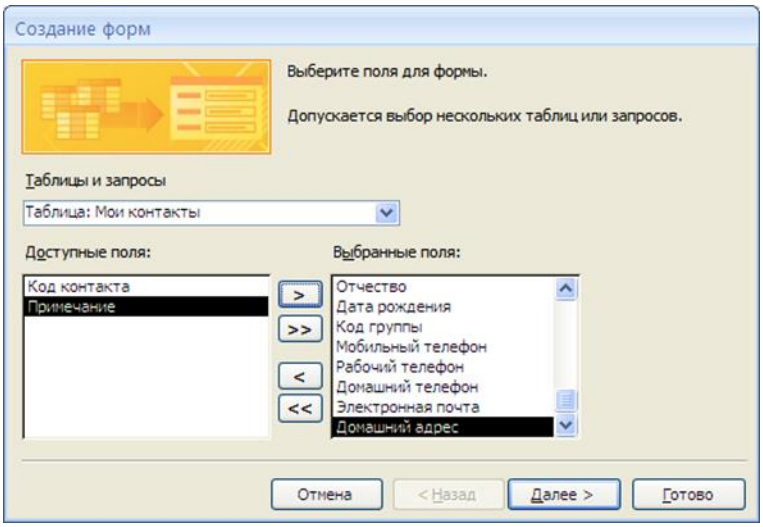
1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. В области переходов щелкните имя таблицы, для которой вы хотите создать форму (например, **Группы контактов).**
3. Щелкните вкладку **Создание**.
4. В группе **Формы** щелкните кнопку  – **Другие формы.**
5. Выберите команду **Мастер форм** в раскрывшемся меню.
6. На экране появится первое окно **Мастера форм**, в котором предлагается выбрать таблицу и определить набор полей для новой формы.
7. Определим таблицу, для которой будет создана новая форма. Мы создаем форму для таблицы **Мои контакты,** поэтому в поле **Таблицы и запросы** оставим значение по умолчанию. Если вам нужно создать форму для другой таблицы, раскройте список **Таблицы и запросы** и выберите название этой таблицы.



Первое окно «Мастера»

8. В списке **Доступные поля** отметьте поля таблицы, которые должны быть на новой форме. Если вы хотите использовать весь набор полей исходной таблицы, щелкните кнопку >>. Чтобы выбрать только часть полей таблицы, щелкните каждое требуемое поле и нажмите кнопку > для его переноса в список **Выбранные поля.**

Для ввода данных в таблицу **Мои контакты** нам понадобятся все поля этой таблицы, за исключением первого и последнего. Поле **Код контакта** заполняется автоматически, а поле **Примечание** используется очень редко и при добавлении новой записи обычно остается незаполненным. Для рения выбора полей сначала щелкните кнопку >>, чтобы перенести в список **Выбранные поля** все поля таблицы **Мои контакты**, после этого щелкните в списке **Выбранные поля** поле **Код контакта** и нажмите кнопку <, чтобы возвратить его в список **Доступные поля**. Точно также удалите из списка **Выбранные поля** поле **Примечание**. В результате в списке **Выбранные поля** должны остаться все поля таблицы **Мои контакты**, за исключением первого и последнего поля.



Выбор полей формы

9. Нажмите кнопку **Далее.** На экране появится второе окно **Мастера.**
10. Выберите внешний вид новой формы, щелкнув один из предложенных вариантов:

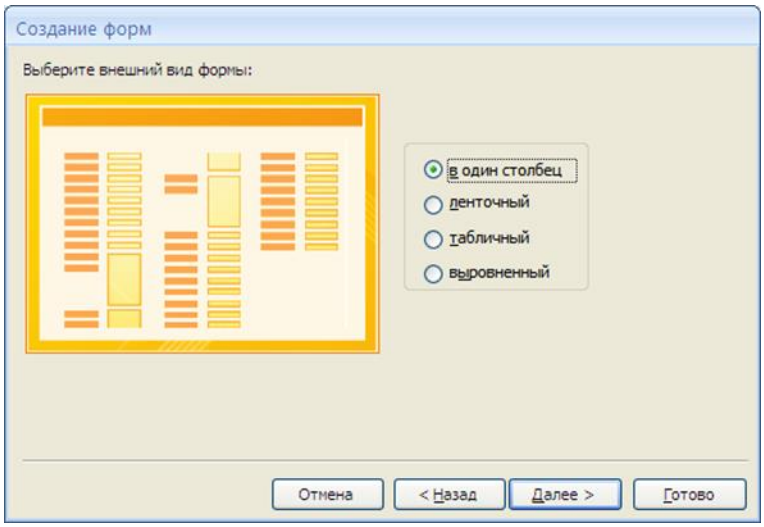
– **В один столбец** – при выборе этого варианта все поля новой формы будут расположены в один столбец;

– **Ленточный** – при выборе этого варианта будет создана ленточная форма;

– **Табличный** – при выборе этого варианта новая форма будут выглядеть как обычная таблица;

– **Выровненный** – при выборе этого варианта будет создана ленточная форма с полями, выровненными по ширине страницы.

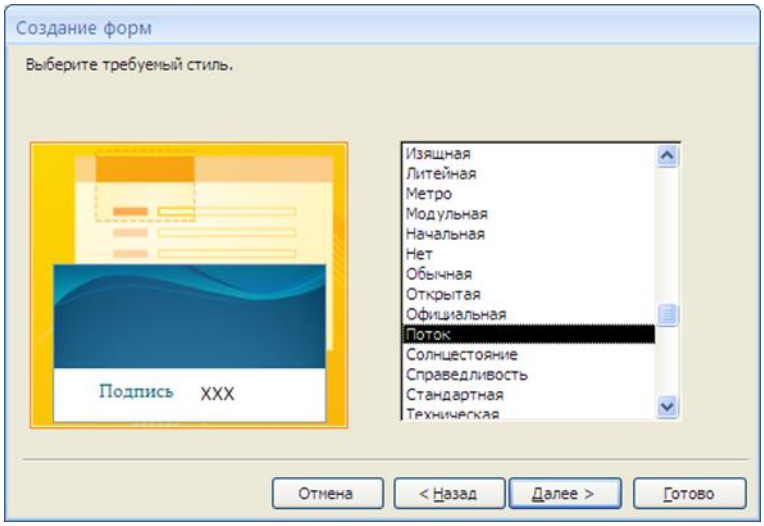
Например, для формы по таблице **Мои контакты** отметьте вариант **Ленточный.**



Второе окно «Мастера»

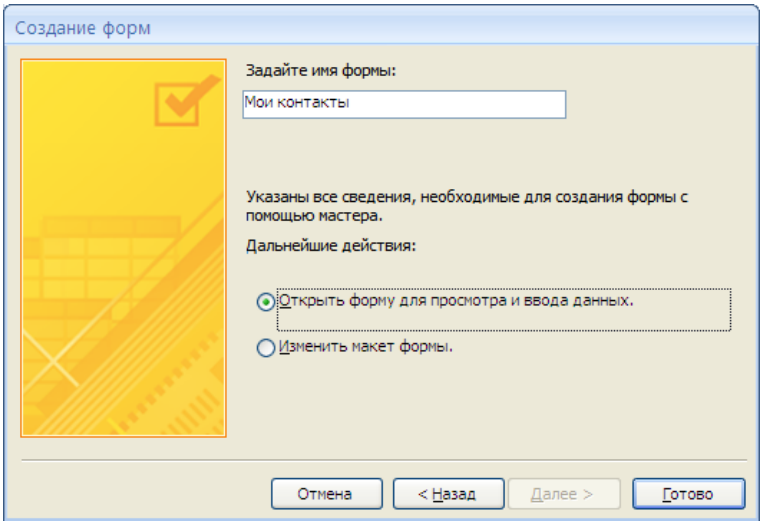
11. Нажмите кнопку **Далее**. На экране появится третье окно Мастера.
12. В предложенном списке выберите желаемый стиль оформления новой формы.

Например, для оформления формы по таблице **Мои контакты** щелкните вариант **Поток** вместо установленного по умолчанию варианта **Стандартная.** Для просмотра образца стиля щелкните название стиля в списке.

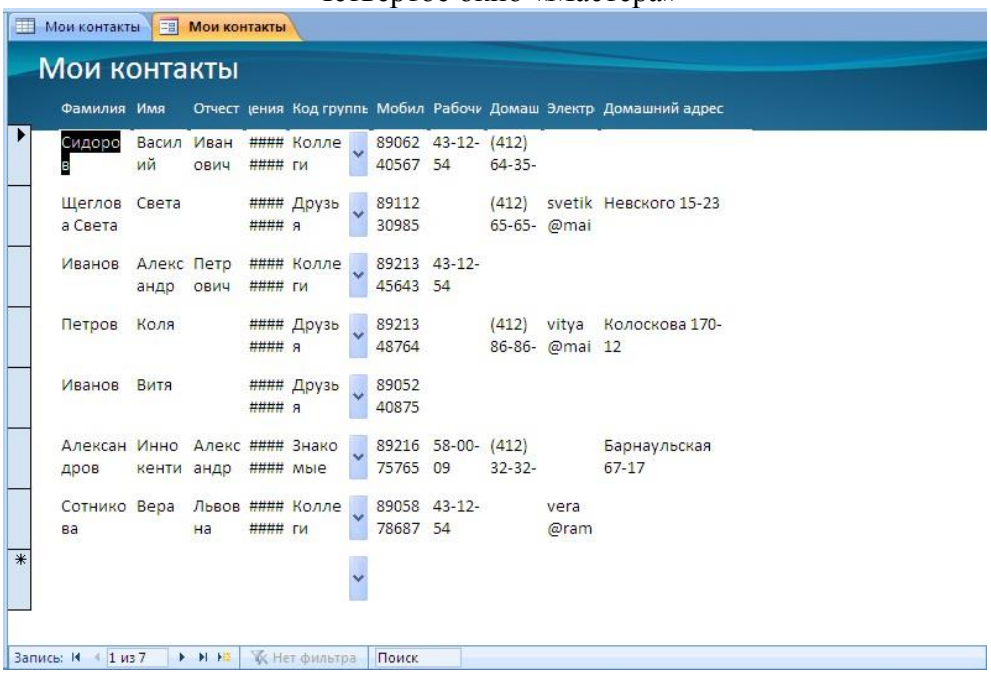


Третье окно «Мастера»

13. Нажмите кнопку Далее. На экране появится четвертое окно мастера
14. Введите название новой формы в верхнее поле ввода. Например, для формы по таблице **Мои контакты** наберите название *Мастер-форма Мои контакты*.
15. Нажмите кнопку **Готово**. В области переходов появится название новой формы, а в рабочей области откроется окно этой формы.



Четвертое окно «Мастера»



Форма созданная «Мастером»

5.4 Создание формы с дополнительными элементами

В форме, созданной с помощью средства **Форма,** одновременно отображается только одна запись таблицы. Если необходимо создать форму, в которой отображается сразу несколько записей, и одновременно с этим требуются более широкие возможности настройки, чем у таблицы, можно воспользоваться инструментом **Несколько элементов.**

Форма, созданная с помощью инструмента **Несколько элементов,** внешне напоминает таблицу. Данные в такой форме располагаются в строках и столбцах, при этом одновременно отображается несколько записей таблицы-источника. Однако форма с несколькими элементами предоставляет гораздо больше возможностей настройки, чем таблица. Например, в такую форму можно добавлять требуемые графические элементы, кнопки и другие элементы управления.

Для того, чтобы создать форму с дополнительными элементами, выполните следующие действия:

1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. В области переходов щелкните имя таблицы, для которой вы хотите создать форму (например, **Группы контактов).**
3. Щелкните вкладку **Создание**.
4. В группе **Формы** щелкните кнопку **Несколько элементов.** Приложение Access создаст форму для выбранной таблицы и отобразит ее в режиме макета.

В режиме макета можно внести изменения в структуру формы при одновременном отображении данных, содержащихся в таблице. Например, можно настроить размер полей формы в соответствии с вводимыми в таблицу данными либо добавить в эту форму дополнительные элементы управления.

Форма с несколькими элементами

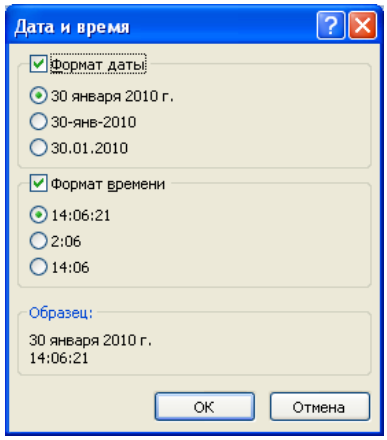
5. При необходимости добавьте на форму дополнительные элементы, воспользовавшись командами вкладки **Формат,** предназначенной для работы с макетами форм.

Для вставки элемента щелкните одну или несколько кнопок группы

Элементы управления:

- **Эмблема** – служит для выбора изображения, которое будет использоваться в качестве эмблемы новой формы;
- **Заголовок** – позволяет изменить заголовок новой формы;
- **Номер страницы** – служит для вставки номеров страниц;
- **Дата и время** – добавляет на форму элемент, отображающий текущее значение даты и / или времени;
- **Добавить поле** – используется для вставки в макет формы нового поля.

Например, для отображения текущих даты и времени в области заголовка формы щелкните кнопку **Дата и время, в** появившемся окне диалога установите желаемый формат отображения даты и времени и нажмите кнопку **ОК.**



Вставка даты и времени

В правой части области заголовка формы появятся текущие значения даты и времени.

6. Нажмите кнопку **Сохранить** на панели быстрого доступа.

5.5 Создание пустой формы

Для того, чтобы создать пустую форму для таблицы базы данных, выполните следующие действия:

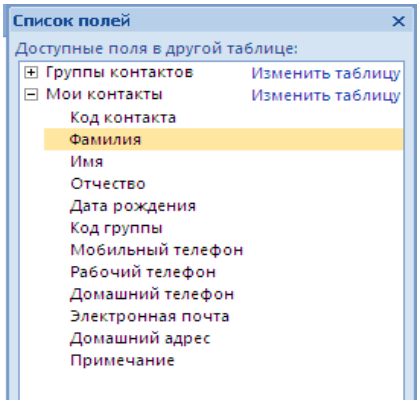
1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. Щелкните вкладку **Создание**.

3. В группе **Формы** щелкните кнопку  – **Пустая форма.** Приложение Access создаст пустую форму и отобразит ее в режиме макета.

4. В области **Список полей** щелкните знак плюс (+) рядом с таблицей, содержащей поля, которые нужно включить в форму. Если в форме должны использоваться поля из нескольких таблиц, то нужно раскрыть списки полей всех нужных вам таблиц. В примере мы создадим форму по таблице **Мои контакты,** поэтому щелкните знак плюс перед ее названием. В результате на экране появится перечень полей таблицы **Мои контакты**.

5. Добавьте в пустой макет новой формы все нужные вам поля таблицы. Для вставки поля щелкните название этого поля в области **Список полей** и перетащите его мышью на форму.

В примере на форму были добавлены **Имя, Фамилия, Мобильный телефон и Код контакта**.

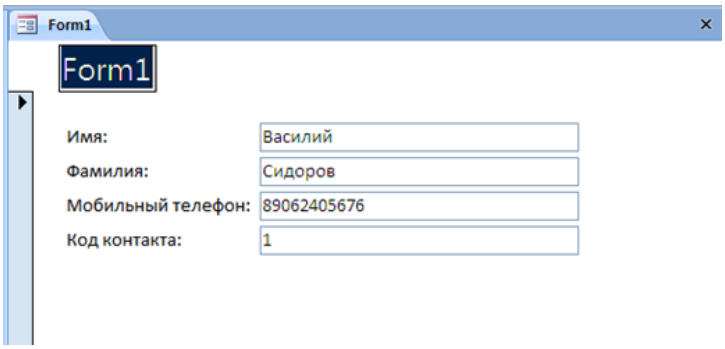


Выбор полей для формы

6. При необходимости добавьте на форму дополнительные элементы, воспользовавшись командами контекстной вкладки **Формат**. Для этого щелкните одну или несколько кнопок группы **Элементы управления:**

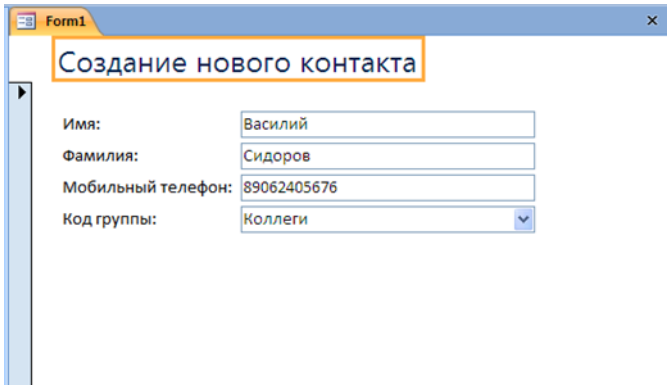
- **Эмблема** – служит для выбора изображения, которое будет использоваться в качестве эмблемы новой формы;
- **Заголовок** – позволяет изменить заголовок новой формы;
- **Номер страницы** – служит для вставки номеров страниц;
- **Дата и время** – добавляет на форму элемент, отображающий текущее значение даты и / или времени;
- **Добавить поле** – используется для вставки в макет формы нового поля.

Мы добавим в новую форму строку заголовка. Для создания заголовка щелкните кнопку **Заголовок** и введите желаемый заголовок формы (например, *Создание нового контакта*) в поле, появившееся в верхней части макета формы.



Добавление заголовка формы

Окончив ввод заголовка, нажмите клавишу Enter. В верхней части формы появится новый заголовок.



Готовая форма

7. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новую форму в базе данных. На экране появится окно запроса имени новой формы.

5.6 Создание формы в конструкторе

Режим Конструктора предоставляет максимальные возможности для создания любых нестандартных форм, требующихся разработчику базы данных. При проектировании формы в Конструкторе можно использовать расширенный набор элементов управления, который недоступен в обычном режиме редактирования макета формы.

Разработчик также имеет возможность настраивать внешний вид формы и расположенных на ней элементов управления в соответствии со своими требованиями и предпочтениями. В Конструкторе можно поменять цвет, стиль оформления, положение и реакцию на действия пользователя у любого элемента управления, расположенного на форме, а также настроить любые свойства самой формы.

5.7 Основные элементы управления

Каждая форма Access обязательно содержит несколько элементов управления, с помощью которых осуществляется доступ к данным, содержащимся в таблицах базы данных. В терминологии Microsoft Access *элементами управления* называются улучшающие интерфейс пользователя объекты, которые используются для отображения данных или выполнения других действий и позволяют просматривать данные и работать с ними.

Наиболее широко используемый элемент управления — поле (текстовое или числовое), которое служит для отображения, ввода и корректировки данных, хранящихся в ячейке таблицы базы данных. К распространенным элементам управления форм Access также относятся кнопки, флажки, переключатели, списки, надписи, а также рамки объектов для отображения графики и объектов OLE.

Создание форм, содержащих необходимые элементы управления, существенно упрощает процесс ввода данных в таблицу и позволяет предотвратить многие ошибки. Программное управление формами и размещенными на них элементами управления осуществляется с помощью процедур, написанных на Visual Basic – встроенном языке программирования Access.

По функциональному признаку любой элемент управления можно отнести к одной из трех следующих групп:

– **Присоединенные элементы управления** – элементы управления, источником данных которых служит поле таблицы или запроса. Каждый присоединенный элемент управления служит для отображения значений соответствующего ему поля базы данных. Значения могут быть текстовыми, числовыми, логическими, датами, рисунками или диаграммами. Например, для текстового поля формы, в котором отображается фамилия служащего, могут использоваться данные поля **Фамилия** в таблице **Служащие.**

– **Свободные элементы управления** – элементы управления, не имеющие источника данных (например, поля или выражения). Свободные элементы управления используются для вывода на экран дополнительных сведений, пояснений, линий, прямоугольников и рисунков. Примером свободного элемента является надпись, которая отображает заголовок формы.

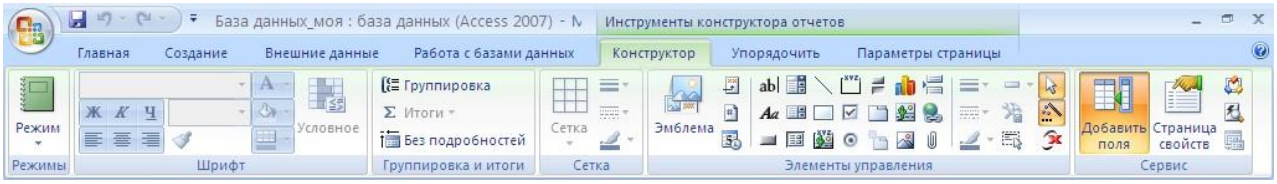
– **Вычисляемые элементы управления** элементы управления, источником данных которых является результат вычисления заданного пользователем выражения, а не поле какой-либо таблицы базы

данных. Для указания значения, которое должно содержаться в вычисляемом элементе управления, необходимо задать выражение, служащее источником данных элемента. *Выражение* – это сочетание математических операторов (+,–,*,/, =), имен других элементов управления, имен полей, функций, возвращающих единственное значение, и констант.

Например, в следующем выражении рассчитывается цена изделия с 25 % скидкой путем умножения значения поля **Цена за единицу** на константу (0,75): [Цена за единицу] □ 0,75

В выражении могут использоваться данные поля в базовой таблице или запросе формы или данные из другого элемента управления формы.

Базовые элементы управления, использующиеся при создании форм Access в режиме **Конструктора**, расположены в группе **Элементы управления** контекстной вкладки **Конструктор.**



Контекстная вкладка «Конструктор»

5.8 Конструирование формы

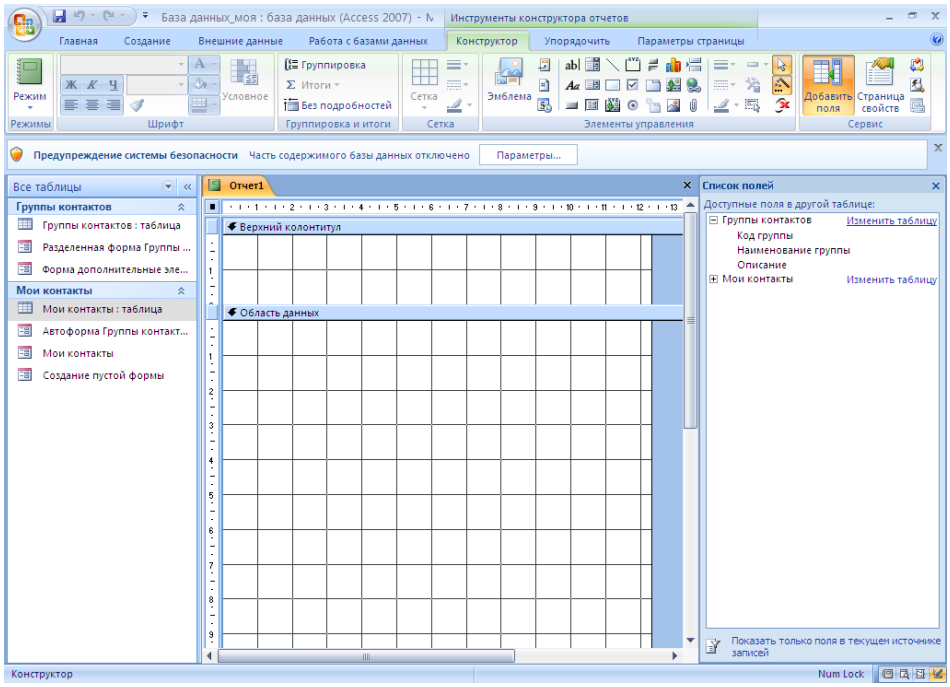
Перед тем, как приступить к созданию новой формы в **Конструкторе**, рассмотрим типовую структуру формы, использующуюся в Microsoft Access. Макет любой формы Access состоит из нескольких разделов, при этом каждый раздел характеризуется особым расположением на макете формы и допустимым набором элементов управления. Таким образом, каждая форма базы данных Access может включать следующие разделы:

- раздел **Заголовок формы** – определяет верхнюю часть формы. Этот раздел добавляется в форму вместе с разделом примечания формы. В область заголовка формы можно поместить текст, графику и другие элементы управления. При печати многостраничной формы раздел заголовка отображается только на первой странице;
- раздел **Верхний колонтитул** – определяет верхний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим нижний колонтитул страницы, и отображается только тогда, когда форма открыта в режиме предварительного просмотра. При печати многостраничной формы верхний колонтитул отображается вверху каждой страницы;
- раздел **Область данных** – определяет основную часть формы, содержащую данные, полученные из источника. Данный раздел может содержать элементы управления, отображающие данные из таблиц и запросов, а также неизменяемые данные (например, пояснительные надписи). При печати многостраничной формы этот раздел отображается на каждой странице;
- раздел **Нижний колонтитул** – определяет нижний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим верхний колонтитул страницы. Он отображается только тогда, когда форма открыта в режиме предварительного просмотра. При печати многостраничной формы нижний колонтитул отображается внизу каждой страницы;
- раздел **Примечание формы** – определяет нижнюю часть формы. Этот раздел добавляется в форму вместе с разделом заголовка формы. При печати многостраничной формы примечание формы будет отображено только внизу последней страницы.

В качестве примера рассмотрим конструирование формы для ввода базовых сведений о новом контакте в таблицу **Мои контакты** учебной базы данных **Личная картотека.**

Для того, чтобы создать новую форму с помощью **Конструктора**, выполните следующие действия:

1. Откройте требуемую базу данных (например, учебную базу данных **Личная картотека**).
2. Щелкните вкладку **Создание**.
3. В группе **Формы** щелкните кнопку **Конструктор форм.** Приложение Access создаст пустую форму и отобразит ее в режиме Конструктора.



Новая форма в Конструкторе

4. Щелкните контекстную вкладку **Конструктор**.

5. Добавьте в форму строку заголовка. Для этого нажмите кнопку  **(Заголовок)** в группе **Элементы управления,** щелкните мышью в верхней части формы и введите текст заголовка (например, *Добавление нового контакта*). Окончив ввод, нажмите клавишу **Enter.**

6. При необходимости переместите заголовок формы, если считаете, что он расположен неудачно. Для этого подведите указатель мыши к рамке заголовка и перетащите его в желаемое место раздела **Заголовок формы** (например, расположите его около левого края формы).

7. В области **Список полей** щелкните знак плюс (+) рядом с таблицей, содержащей поля, которые нужно включить в форму. Если в форме должны использоваться поля из нескольких таблиц, то нужно раскрыть списки полей всех нужных вам таблиц. В примере мы создадим форму по таблице **Мои контакты,** поэтому щелкните знак плюс перед ее названием. В результате на экране появится перечень полей таблицы **Мои контакты.**

8. Добавьте в раздел **Область данных** макета новой формы все нужные вам поля таблицы. Для вставки поля сначала щелкните название этого поля в области **Список полей,** а затем щелкните мышью в том месте раздела **Область данных** формы, где вы хотите поместить это поле. На форме появится группа из двух связанных элементов собственно поля ввода и подписи к нему (подпись совпадает с именем этого поля в исходной таблице).

Описанным выше способом добавьте на форму поля **Фамилия, Имя, Отчество, Код группы, Мобильный телефон и Электронная почта.**

Постарайтесь разместить их около нижней границы раздела **Область** данных, чтобы сверху осталось место для дополнительных элементов формы.

9. Добавьте на форму первую пояснительную надпись. Для этого перейдите на контекстную вкладку **Конструктор** и в группе **Элементы управления** щелкните кнопку **Надпись.** После этого щелкните мышью в верхней части раздела **Область данных** и введите текст надписи (*Паспортные данные*:). Окончив ввод, нажмите клавишу **Enter.**

10. Перетащите созданную надпись к верхнему левому углу раздела **Область данных**.

11. Щелкните на форме поле с подписью **Фамилия** и перетащите его под пояснительную надпись **Паспортные данные.** Например, расположите его немного ниже надписи с небольшим отступом по горизонтали.

12. Щелкните на форме поле с подписью **Отчество,** перетащите под поле **Фамилия** и расположите снизу от него на таком же уровне по горизонтали.

13. Щелкните на форме поле с подписью **Имя,** перетащите к полю **Фамилия** и расположите справа от него на таком же уровне по вертикали (поля **Фамилия** и **Имя** должны располагаться в одну строку, но на некотором расстоянии друг от друга).

14. Добавьте на форму вторую пояснительную надпись. Для этого перейдите на контекстную

вкладку **Конструктор** и в группе **Элементы управления** щелкните кнопку **Надпись.** После этого щелкните мышью на форме под полем **Отчество** и введите текст надписи (*Категория контактов:*). Окончив ввод, нажмите клавишу **Enter.**

15. Расположите вторую пояснительную надпись на уровне первой, немного отступив вниз от поля **Отчество.**

16. Щелкните на форме поле с подписью **Код контакта,** перетащите к надписи **Категория контактов** и расположите справа от нее на уровне поля **Имя.**

17. Добавьте на форму третью пояснительную надпись. Для этого перейдите на контекстную вкладку **Конструктор** и в группе **Элементы управления** щелкните кнопку **Надпись.** После этого щелкните мышью на форме под полем **Отчество** и введите текст надписи (*Средства связи:*). Окончив ввод, нажмите клавишу **Enter.**

18. Расположите третью пояснительную надпись на уровне второй, немного отступив от нее по вертикали.

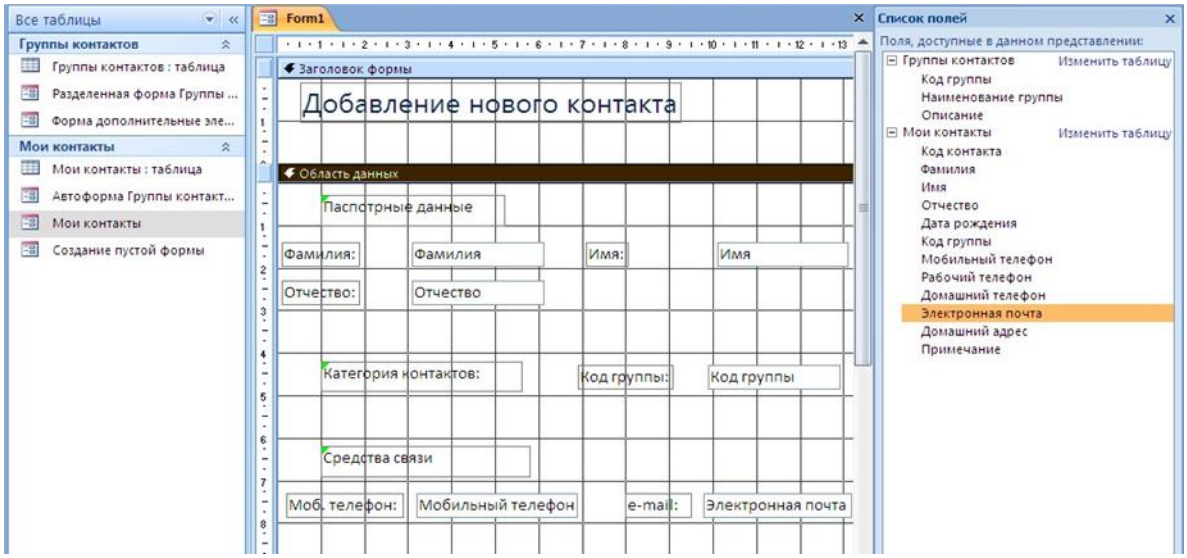
19. Щелкните на форме поле с подписью **Мобильный телефон,** перетащите под надпись **Средства связи** и расположите снизу от нее на уровне полей **Фамилия и Отчество.**

20. Щелкните на форме поле с подписью **Электронная почта,** перетащите к полю **Мобильный телефон** и расположите справа от него на уровне полей **Имя** и **Код группы.**

21. Измените подпись у поля **Мобильный телефон** (она слишком длинная и поэтому не видна полностью). Для корректировки щелкните внутри подписи **Мобильный телефон** два раза *с* небольшим интервалом между щелчками. В подписи должен появиться текстовый курсор. Наберите на клавиатуре новый текст подписи (*Моб. телефон*) и нажмите клавишу **Enter**.

22. Аналогично измените подпись поля **Электронная почта** (вместо текста *Электронная почта* наберите слово *e-mail*).

23. Теперь мы расположили на новой форме все нужные элементы управления. Результирующий вид макета новой формы после размещения всех элементов управления.



Форма с элементами

24. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новую форму в базе данных. На экране появится окно запроса имени новой формы.

25. Введите желаемое название формы в поле **Имя формы** (например, *Конструктор-форма Мои контакты*) и нажмите кнопку **ОК.** Название новой формы появится в области переходов.

26. Теперь немного «раскрасим» форму, чтобы она выглядела более привлекательно. Сначала настроим фон формы.

27. Щелкните правой кнопкой мыши в любом свободном месте раздела **Заголовок формы.** На экране появится контекстное меню.

28. Щелкните пункт **Цвет заливки / фона.** На экране раскроется подменю выбора цвета заливки, в котором представлены образцы цвета заливки.

29. Щелкните понравившийся вам цвет (например, *светло-голубой*). Раздел **Заголовок формы** будет закрашен выбранным цветом.

30. Щелкните правой кнопкой мыши в любом свободном месте раздела **Область** данных,

выберите в контекстном меню пункт **Цвет заливки/фона** и щелкните понравившийся образец цвета (например, выберите цвет на тон темнее светло-голубого). Раздел **Область данных** будет закрашен выбранным вами цветом.

31. Щелкните правой кнопкой мыши в любом свободном месте раздела Примечание формы, выберите в контекстном меню пункт **Цвет заливки/фона** и щелкните понравившийся образец цвета (например, выберите цвет на два тона темнее *светло-голубого*). Раздел **Примечание** формы будет закрашен выбранным вами цветом.

32. Измените цвет фона и текста заголовка формы (*Добавление нового контакта*). Для установки цвета текста щелкните заголовок правой кнопкой мыши, выберите в контекстном меню пункт **Цвет текста** и щелкните понравившийся образец цвета (например, выберите *ярко-желтый цвет*). Фон заголовка формы будет закрашен выбранным вами цветом.

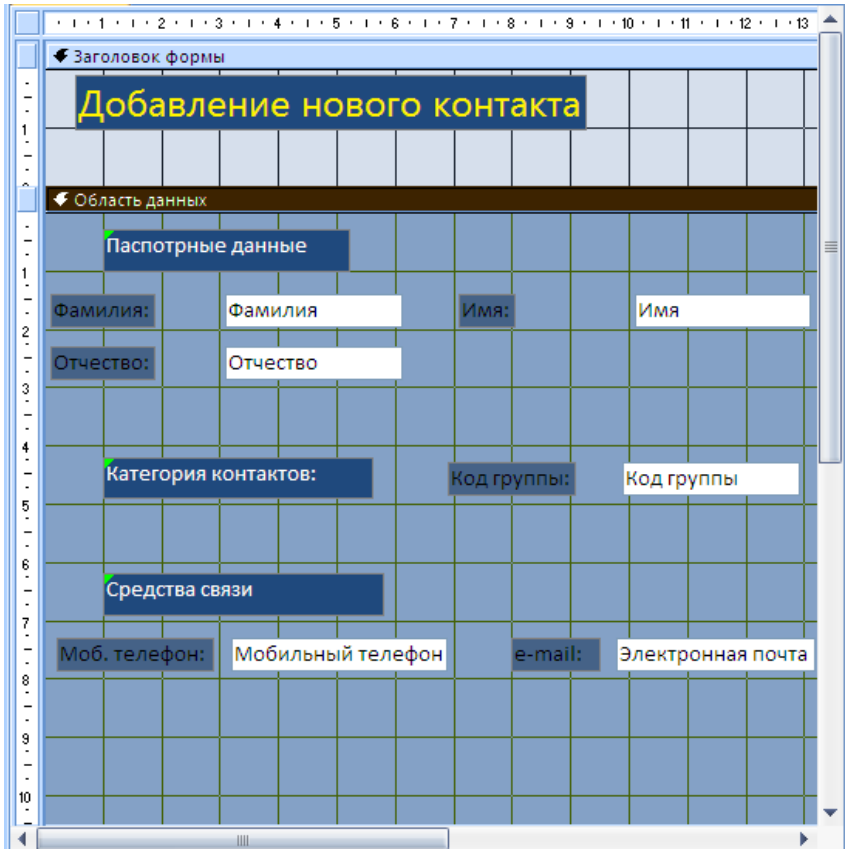
33. Для установки цвета фона заголовка формы щелкните заголовок правой кнопкой мыши, выберите в контекстном меню пункт **Цвет заливки/фона** и щелкните понравившийся образец цвета (например, выберите *темно-синий* цвет). Текст заголовка формы будет закрашен выбранным вами цветом.

34. Аналогично установке цвета текста и фона заголовка формы измените цвета трех пояснительных надписей (*Паспортные данные, Категория контактов* и *Средства связи*). Выберите для фона надписей *темно-синий* цвет, а для текста *белый*.

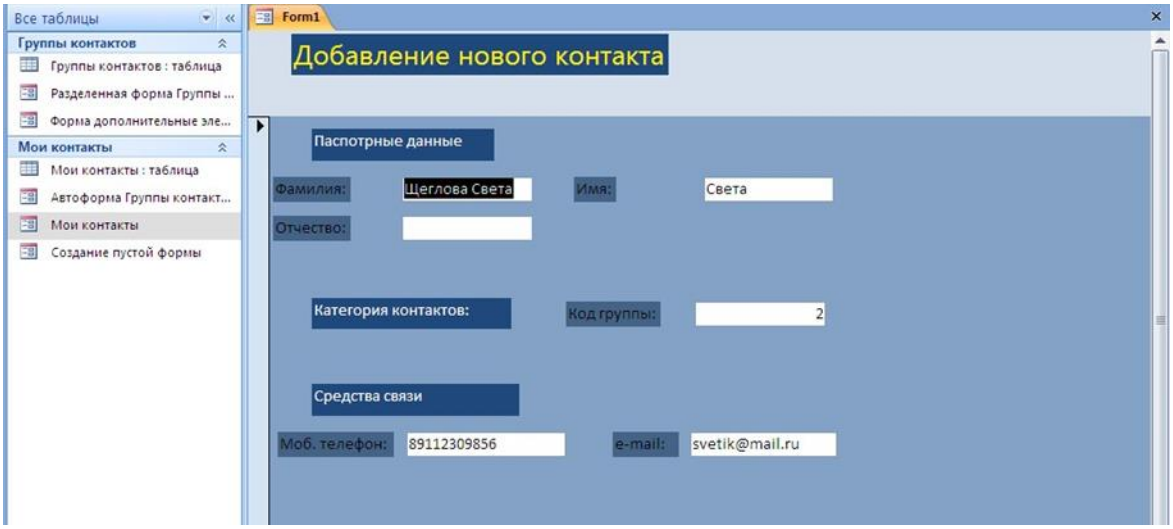
35. Точно так же установите цвета подписей полей (*Фамилия, Имя, Отчество, Код группы, Моб. телефон и e-mail*). Выберите для фона подписей цвет на два тона темнее *светло-голубого*, а цвет текста не изменяйте.

36. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новые цветовые настройки формы в базе данных.

37. Переключитесь из режима **Конструктора** в режим формы, чтобы посмотреть внешний вид готовой формы.



Готовая форма в Конструкторе

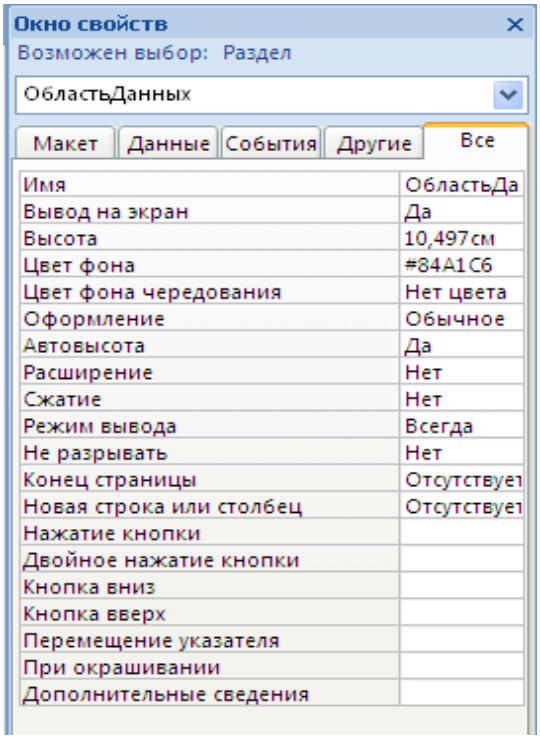


Окно готовой формы

5.9 Изменение свойств формы

Для того, чтобы изменить свойства формы в **Конструкторе**, выполните следующие действия:

1. Откройте вашу базу данных (например, учебную базу данных **Личная картотека).**
2. В области переходов дважды щелкните название нужной формы (например, выберите **Конструктор-форма Мои контакты).** Указанная вами форма откроется в используемом по умолчанию режиме формы.
3. Переключитесь в режим **Конструктора** форм.
4. Для настройки свойств раздела формы щелкните правой кнопкой мыши в любом свободном месте этого раздела формы и выберите команду **Свойства** из контекстного меню.



Свойства раздела формы

5. В правой части окна **Конструктора** форм появится область **Окно свойств,** в которой будут отображены все доступные свойства выбранного раздела формы.
6. Чтобы изменить какое-либо свойство раздела, щелкните поле справа от названия этого свойства и установите новое значение свойства (выберите требуемый параметр из предложенного списка или введите с клавиатуры).
7. Для изменения общих свойств формы щелкните правой кнопкой мыши в любом свободном месте формы и выберите требуемые опции из контекстного меню:

− чтобы настроить последовательность переходов между полями формы, щелкните опцию **Переходы;**

− чтобы изменить процедуры обработки событий формы, щелкните опцию **Обработка событий.**

8. Нажмите кнопку **Сохранить на** панели быстрого доступа, чтобы сохранить сделанные изменения.

5.10 Добавление элемента управления

Для того, чтобы добавить элемент управления на существующую форму, выполните следующие действия:

1. Откройте вашу базу данных (например, учебную базу данных **Личная картотека).**
2. В области переходов дважды щелкните название нужной формы (например, выберите **Конструктор-форма Мои контакты).**
3. Переключитесь в режим **Конструктора** форм.
4. Чтобы добавить на форму новый элемент управления, щелкните контекстную вкладку **Конструктор,** перейдите в группу **Элементы управления** и щелкните кнопку, соответствующую нужному вам элементу управления.
5. Щелкните мышью в том месте формы, где вы хотите расположить новый элемент управления.
6. Чтобы сделать копию одного из существующих элементов формы, нажмите на этом элементе правую кнопку мыши и выберите пункт **Копировать** в контекстном меню. После этого щелкните правой кнопкой мыши в месте вставки копии выбранного элемента и выберите пункт **Вставить** в контекстном меню.
7. При необходимости измените расположение добавленного элемента управления, перетащив его мышью в желаемое место формы.
8. Настройте требуемые свойства добавленного элемента управления. Для этого нажмите на этом элементе правую кнопку мыши и выберите пункт **Свойства** в контекстном меню, а затем установите желаемые значения нужных вам свойств элемента в области **Окно свойств,** расположенной в правой части окна **Конструктора**.
9. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить все сделанные изменения.

5.11 Использование формы

Основной целью создания простых форм является добавление новых записей, удаление записей или редактирование данных в одной или более таблицах, являющихся источником данных для формы. Добавление, удаление и редактирование записей с помощью формы происходит при работе с ней в режиме формы.

Добавление записи. В формах, как и в таблицах, предусмотрена пустая запись, которая предназначена для добавления новых записей в таблицу. Аналогично пустой записи любой таблицы базы данных, пустая запись формы отображается после всех заполненных записей. К этой записи можно перейти, пролистав в форме все записи с помощью кнопок перехода по записям или с помощью специальной кнопки перехода на новую запись. После ввода данных в эту запись формы и ее сохранения данные автоматически попадают в таблицу, связанную с формой.

При добавлении новой записи в таблицу с помощью формы необходимо, переместившись на пустую запись, заполнить поля этой формы в соответствии с правилами, определенными разработчиком формы. Перемещение между полями формы, используемыми для ввода и редактирования данных, происходит аналогично тому, как это делается в таблицах, за исключением того, что клавиши перемещения курсора вверх и вниз перемещают курсор между полями, а не между записями.

Для того, чтобы добавить новую запись в таблицу с помощью формы, выполните следующие действия:

1. Откройте вашу базу данных (например, учебную базу данных **Личная** картотека).
2. В области переходов дважды щелкните название нужной формы (например, для работы с таблицей **Мои контакты** выберите **Конструктор-форма Мои контакты).** Выбранная форма откроется на первой записи таблицы в используемом по умолчанию режиме формы.

3. Щелкните кнопку  **Новая (пустая) запись,** расположенную на панели навигации в нижней части окна формы.

Форма перейдет в режим добавления новой записи. Все поля формы будут очищены, в верхнем поле появится текстовый курсор.

4. Заполните поля формы данными, соответствующими новой записи в таблице **Мои контакты.** Щелкните поле **Фамилия** и введите фамилию человека (например, *Сидоров*).

5. Щелкните поле **Имя** и введите имя человека (например, *Иннокений)*

6. Щелкните поле **Отчество** и введите отчество человека (например, *Петрович*).

7. Раскройте список **Код группы** и выберите категорию контактов, к которой относится новый человек (например, *Знакомые*).

8. Щелкните поле **Мобильный телефон** и введите номер мобильного телефона человека (например, *89216234876*).

9. Поле **e-mail** оставьте не заполненным.

10. Нажмите клавишу **Enter** в последнем поле формы (это поле **e-mail),** чтобы сохранить добавленную запись. Введенные данные новой записи будут сохранены в таблице **Мои контакты**, а форма опять перейдет в режим добавления новой записи.

11. При необходимости продолжайте добавление новых записей в таблицу с помощью формы.

12. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить все сделанные изменения.

6 Формирование запросов в базе данных

6.1 Запросы на выборку

Запросы используются для просмотра, анализа и изменения данных в одной или нескольких таблицах базы данных Access. Например, можно использовать запрос для отображения данных из одной или нескольких таблиц и отсортировать их в определенном порядке, выполнить вычисления над группой записей, осуществить выборку из таблицы по определенным условиям. Запросы также могут служить источником данных для форм и отчетов Microsoft Access.

Сам запрос не содержит данных, но позволяет выбирать данные из таблиц и выполнять над ними ряд операций. В Microsoft Access существует несколько видов запросов:

- **запросы к серверу** – эти запросы используются для выборки данных с сервера;
- **запросы на автоподстановку** – эти запросы автоматически заполняют поля для новой записи;
- **запросы на выборку** – эти запросы выполняют выборку данных из таблиц базы данных;
- **запросы на изменение** – эти запросы дают возможность модифицировать данные в таблицах (в том числе удалять, обновлять и добавлять записи);
- **запросы на создание таблицы** – эти запросы позволяют создать новую таблицу на основе данных одной или нескольких существующих таблиц.

6.2 Создание запросов

В терминологии Microsoft Access любой запрос представляет собой обращение к данным для получения информации и выполнения действий с данными. Запрос можно использовать для получения ответа на простой вопрос, выполнения расчетов, объединения данных из разных таблиц или даже добавления, изменения или удаления данных в таблице. Запросы, используемые для извлечения данных из таблицы или выполнения расчетов, называются *запросами на выборку.* Запросы, используемые для добавления, изменения или удаления данных, называются *запросами на изменение.*

С помощью запросов можно получить ответы даже на самые сложные вопросы о данных, содержащихся в таблицах базы данных. Запросы можно использовать для фильтрации данных, выполнения расчетов на основе данных и отображения сводных данных. Кроме того, запросы позволяют автоматизировать выполнение многих задач управления данными и просматривать изменения в данных перед их использованием.

Запросы можно также использовать для включения данных в создаваемую форму или отчет. В хорошо структурированной базе данных сведения, которые требуется представить с использованием формы или отчета, чаще всего хранятся в разных таблицах. С помощью запроса можно собрать необходимые данные перед проектированием формы или отчета.

В Microsoft Access используются четыре основных способа создания нового запроса:

- создание запроса с использованием **Мастера запросов**;
- создание запроса с помощью **Конструктора запросов**;
- создание запроса в режиме SQL-редактора;
- создание запроса на основе существующего фильтра.

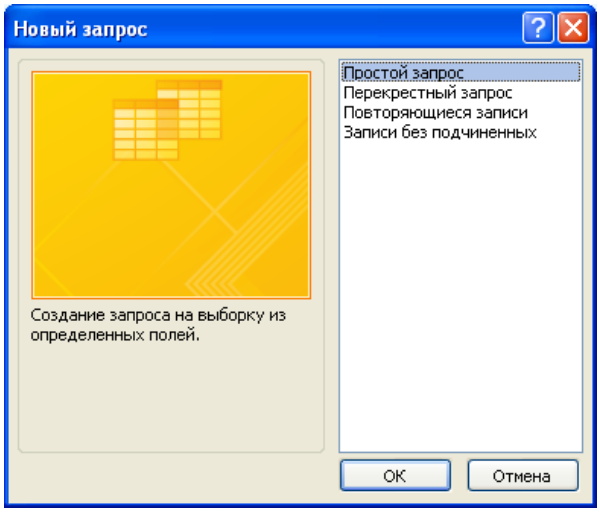
6.3 Создание с помощью мастера

Самый простой способ создания нового запроса в базе данных Access – это использование **Мастера запросов**.

Рассмотрим этот способ на примере создания запроса к таблице **Мои контакты** учебной базы данных **Личная картотека.**

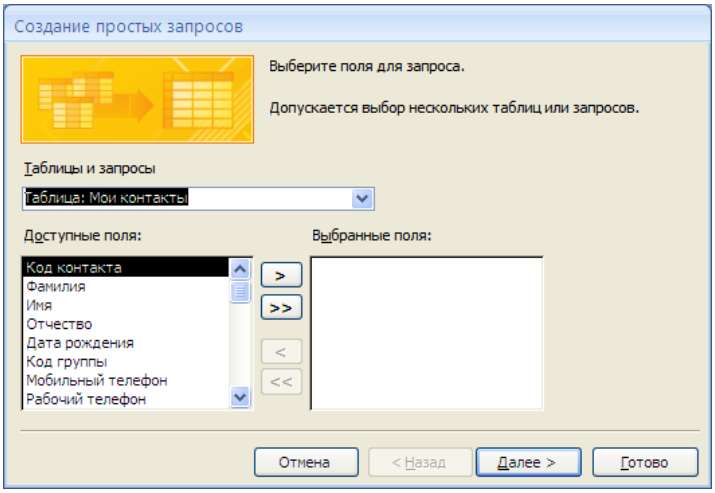
Чтобы создать простой запрос с помощью **Мастера запросов**, выполните следующие действия:

1. Откройте базу данных **Личная картотека.**
2. Щелкните вкладку **Создание**.
3. В группе **Другие** нажмите кнопку **Мастер запросов.** На экране появится первое диалоговое окно Мастера запросов.



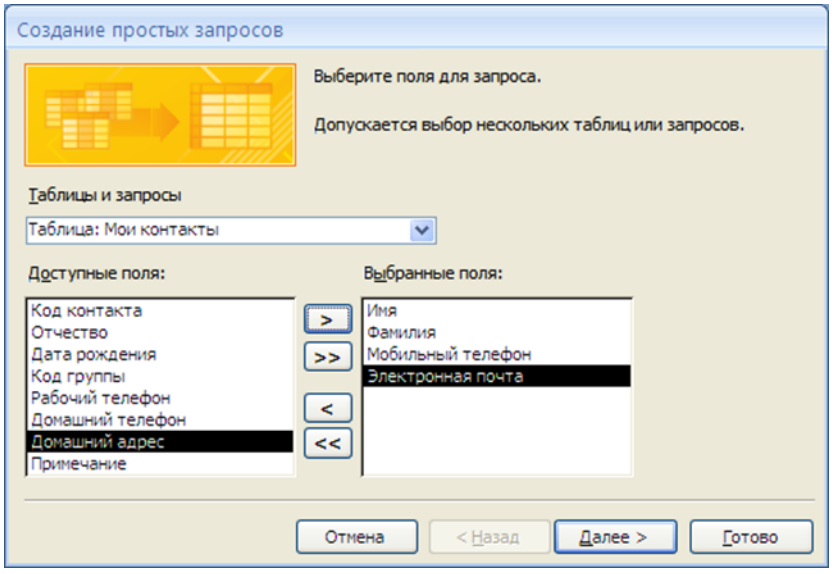
Первое окно мастера

4. Убедитесь, что в окне **Мастера** отмечен вариант **Простой запрос.**
5. Нажмите кнопку **ОК.** На экране появится следующее окно **Мастера**
6. Раскройте список **Таблицы и запросы** и выберите источник данных для создания нового запроса. Для примера нам понадобится таблица **Мои контакты.**



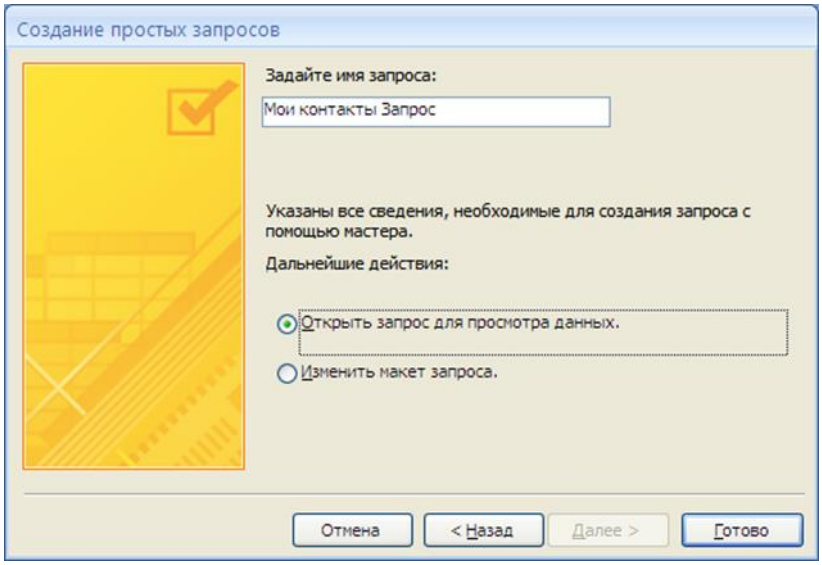
Второе окно Мастера

7. Выберите поля таблицы, которые будут использоваться в запросе. Для этого перенесите требуемые поля из списка **Доступные поля** в список **Выбранные поля.** Для учебного запроса нам понадобятся поля **Имя, Фамилия, Мобильный телефон и Электронная почта.** Последовательно щелкайте эти поля в списке **Доступные поля** и нажимайте кнопку > В результате список **Выбранные поля** будет содержать четыре наименования полей таблицы **Мои контакты**.



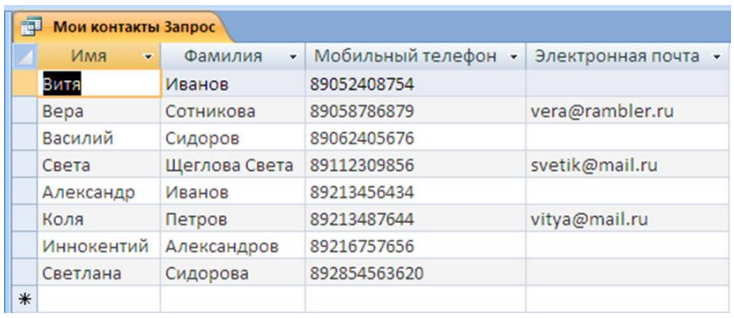
Выбор полей таблицы

8. Нажмите кнопку **Далее**. На экране появится последнее окно **Мастера запросов**.
9. В поле **Имя запроса** введите название нового запроса (например, *Мастер запрос Контакты*).



Последнее окно Мастера

10. Установите переключатель **Открыть запрос для просмотра данных,** чтобы посмотреть результат выполнения запроса.
11. Нажмите кнопку **Готово,** чтобы завершить работу **Мастера**. Access создаст новый запрос с указанным именем и сохранит его в текущей базе данных. В области переходов появится название только что созданного запроса, а в главном окне Access отобразится таблица, содержащая выбранные по этому запросу данные.



Имя	Фамилия	Мобильный телефон	Электронная почта
Витя	Иванов	89052408754	
Вера	Сотникова	89058786879	vera@rambler.ru
Василий	Сидоров	89062405676	
Света	Щеглова Света	89112309856	svetik@mail.ru
Александр	Иванов	89213456434	
Коля	Петров	89213487644	vitya@mail.ru
Иннокентий	Александров	89216757656	
Светлана	Сидорова	892854563620	

Результат запроса

6.4 Создание запроса в конструкторе

Второй способ создания нового запроса, более сложный и в то же время предоставляющий пользователю Access максимальные возможности, связан с использованием **Конструктора** запросов. **Конструктор** запросов используется не только для создания новых запросов, но и для изменения уже существующих в базе данных запросов.

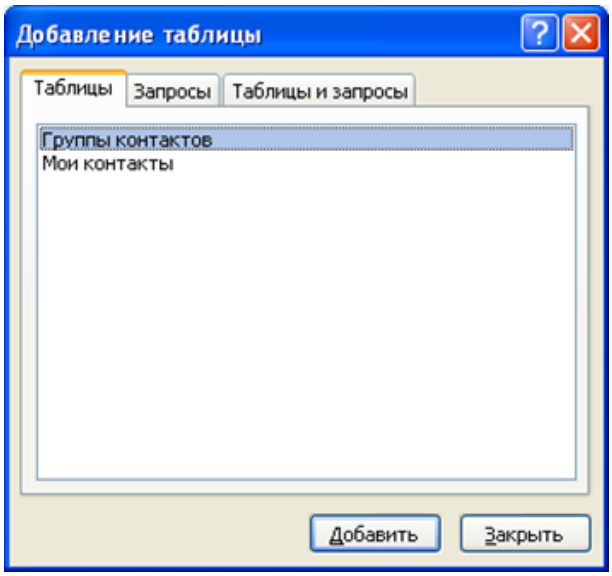
При активации режима **Конструктора** на ленте инструментов появляется контекстная вкладка **Конструктор,** содержащая базовые команды для работы с запросом. После запуска **Конструктора** на экране появляется его окно, в котором присутствуют все инструменты, необходимые для создания или изменения запроса.

В верхней части окна **Конструктора** запросов отображается базовая таблица запроса (или несколько таблиц, если запрос многотабличный) в том же виде, в каком эти таблицы отображаются в окне **Схема** данных. В нижней части окна **Конструктора** находится бланк запроса – специальная таблица, ячейки которой используются для определения структуры и свойств запроса. В бланке отображаются все столбцы базовых таблиц, включенные в результирующее множество запроса. Чтобы полностью просматривать бланк сложного запроса и все исходные таблицы, используются горизонтальные и вертикальные линейки прокрутки.

В качестве практического примера работы в **Конструкторе** создадим запрос к таблице **Группы контактов** учебной базы данных **Личная картотека.** По этому запросу будет выводиться код, соответствующий указанному в условии названию группы контактов (*например, код группы «Друзья»*).

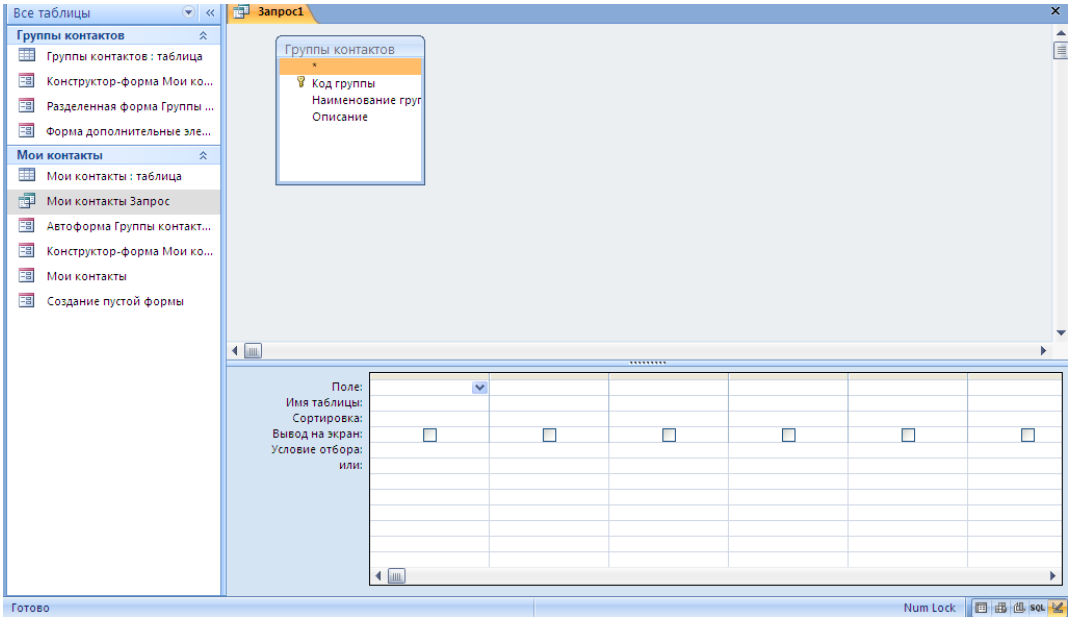
Чтобы создать простой запрос с помощью **Конструктора** запросов, выполните следующие действия:

1. Откройте базу данных **Личная картотека.**
2. Щелкните вкладку **Создание**.
3. В группе **Другие** нажмите кнопку **Конструктор запросов.** На экране появится рабочая среда **Конструктора запросов**, в которой откроется диалоговое окно **Добавление таблицы**.

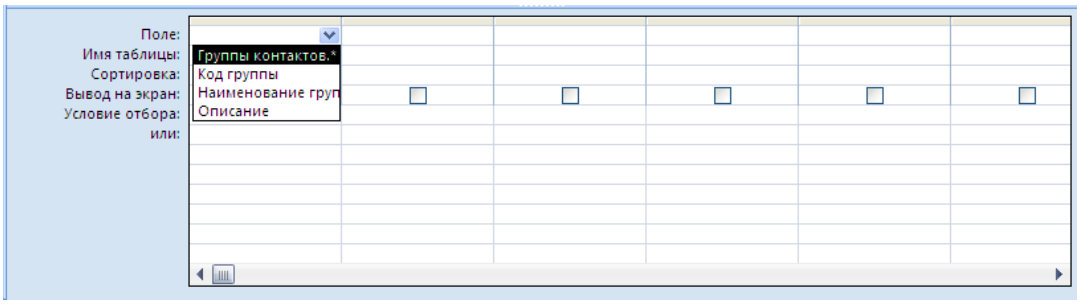


Диалоговое окно «Добавление таблицы»

4. На вкладке **Таблицы** щелкните имя таблицы **Группы контактов** и нажмите кнопку **Добавить.** После этого щелкните кнопку **Закрыть.** На экране появится пустой бланк нового запроса на основе таблицы **Группы контактов**
5. Раскройте первый список **Поле** и щелкните имя поля **Наименование группы**

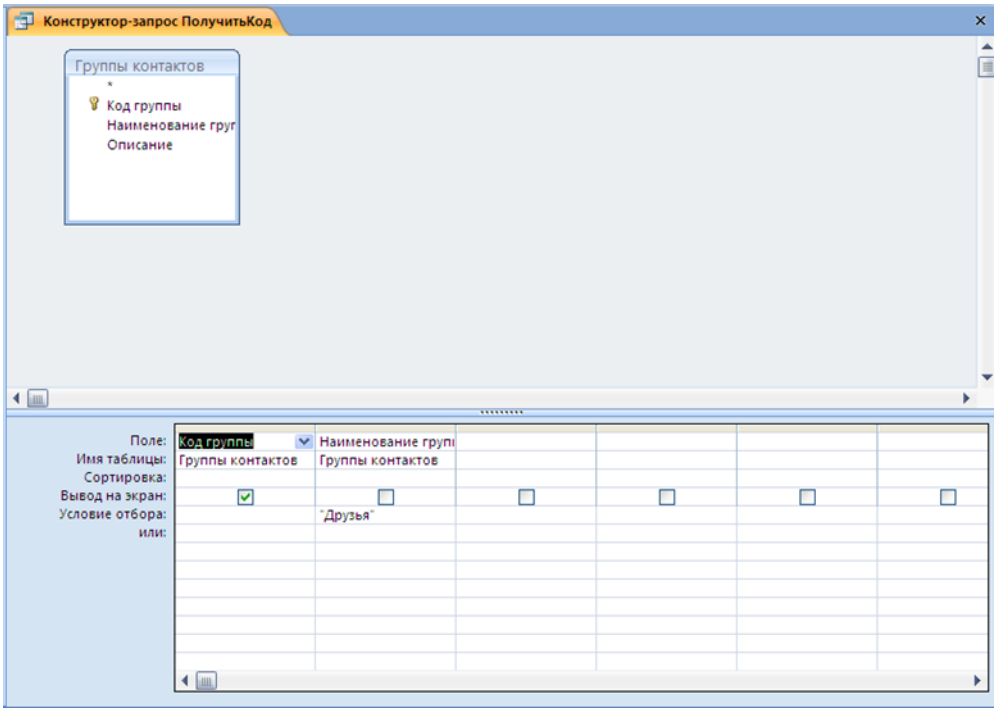


Бланк запроса в Конструкторе



Выбор поля таблицы

6. Отображать название группы контактов не нужно, поэтому снимите галочку в поле **Вывод на экран.**
7. Щелкните поле **Условие отбора** и наберите слово *Друзья*.
8. Раскройте второй список **Поле,** щелкните имя поля **Код группы.**
9. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить готовый запрос. На экране появится диалог ввода имени нового запроса.
10. В поле **Имя запроса** введите название нового запроса (например, *Конструктор-запрос ПолучитьКод*).
11. Нажмите кнопку **ОК.**
12. Если вы хотите посмотреть результат выполнения нового запроса, щелкните контекстную вкладку **Конструктор** и нажмите кнопку **Выполнить**. В результате выполнения запроса на экране отобразится код группы контактов «Друзья».



Готовый бланк запроса

7 Создание отчетов в базе данных

7.1 Создание отчетов

В отличие от других объектов базы данных, с помощью отчета невозможно изменять сведения, хранящиеся в базе данных. С помощью отчета можно только отображать данные, содержащиеся в таблицах или выбранные из базы данных по запросу. Зато именно отчет как нельзя лучше подходит для представления требуемых пользователю данных в виде полноценных документов, для которых можно выбрать требуемый стиль оформления, а затем напечатать на принтере или отправить по электронной почте.

С помощью средств Microsoft Access пользователь может создать множество различных отчетов любой степени сложности. К основным способам создания нового отчета в текущей базе данных Access можно отнести следующие варианты:

- автоматическое создание отчета с помощью средства **Отчет;**
- создание отчета с помощью **Мастера отчетов**;
- создание отчета с использованием средства **Пустой отчет**;
- создание отчета в **Конструкторе отчетов**;
- создание наклеек с помощью **Мастера наклеек.**

Каждый отчет Access содержит необходимые пользователю сведения, выбранные из таблиц и / или запросов базы данных. Наряду с данными, извлекаемыми из таблиц и запросов, в любом отчете обязательно содержится информация о макете отчета. В макете отчета хранятся заданные пользователем параметры страницы отчета, а также сведения о структуре и свойствах отчета и его отдельных элементов (подписей, заголовков, рисунков).

Таблицы и запросы, содержащие базовые данные для отчета, называются *источником записей* отчета. Если все поля, которые нужно включить в отчет, находятся в одной таблице, эта таблица и будет источником записей указанного отчета. Если требуемые поля находятся в нескольких таблицах, в качестве источника записей иногда приходится использовать один или несколько запросов. В некоторых случаях эти запросы уже существуют в базе данных, но гораздо чаще требуется формировать новые запросы специально для создания отчета.

7.2 Автоматическое создание отчета

Самый простой способ создания нового отчета заключается в использовании средства автоматического создания отчетов, входящего в состав Access. В этом случае новый отчет по выбранной таблице или запросу формируется сразу же, без указания какой-либо дополнительной информации. Таким образом, для автоматического создания отчета пользователю Access потребуется всего один щелчок мыши.

Автоматическое создание простого отчета на основе существующего запроса или таблицы базы данных выполняется с помощью инструмента **Отчет.** При использовании этого средства в новом отчете размещаются все поля выбранной таблицы базы данных или запроса. Сразу после создания нового отчета его можно просмотреть и при необходимости распечатать или отправить по электронной почте. Если пользователя не устраивает структура или форматирование созданного отчета, он может изменить требуемые параметры отчета в режиме **Макета** или в **Конструкторе**.

В качестве практического примера автоматически создадим простой отчет на основе таблицы **Группы контактов** учебной базы данных **Личная картотека.**

Чтобы автоматически создать простой отчет по таблице или запросу, выполните следующие действия:

1. Откройте требуемую базу данных Access (например, учебную базу данных **Личная картотека).**
2. В области переходов щелкните название того объекта базы данных (таблицы или запроса), который должен использоваться в качестве источника данных для нового отчета. Для приведенного примера щелкните имя таблицы **Группы контактов.**
3. На ленте инструментов перейдите на вкладку **Создание**.
4. В группе **Отчеты** щелкните кнопку **Отчет.** Приложение Access создаст новый отчет и отобразит его в режиме макета.

Простой отчет

В режиме макета можно внести изменения в структуру отчета при одновременном отображении данных, содержащихся в объекте-источнике. Например, при необходимости можно настроить размер полей, выбрать способ отображения данных или изменить форматирование отчета в соответствии с предпочтениями пользователя.

5. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить новый отчет в текущей базе данных.
6. Введите желаемое название отчета в поле **Имя** отчета (например, *Автоотчет Группы контактов*).
7. Нажмите кнопку **ОК.** Новый отчет будет сохранен в текущей базе данных, а его название появится в области переходов.

7.3 Создание отчета с помощью Мастера

Для получения большей свободы выбора полей, включаемых в создаваемый отчет, вместо рассмотренного выше инструмента автоматического формирования нового отчета можно

воспользоваться **Мастером отчетов**.

В случае использования **Мастера отчетов** при создании нового отчета можно указать способ группировки и сортировки данных, а также включить в отчет поля из нескольких таблиц или запросов (при условии, что в Access уже заданы отношения между этими таблицами и запросами).

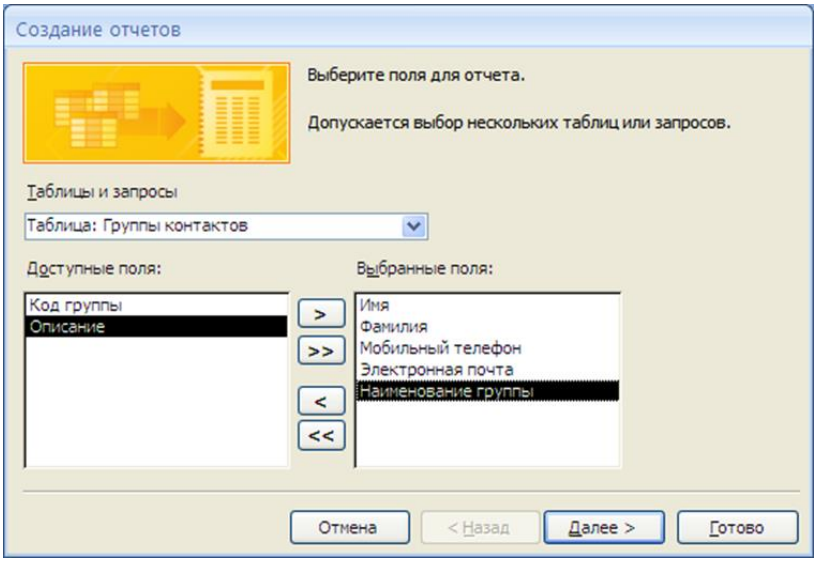
В качестве практического примера использования **Мастера отчетов** мы создадим отчет на основе данных таблиц **Мои контакты** и **Группы контактов** учебной базы данных **Личная картотека**.

Чтобы создать новый отчет по таблице или запросу с помощью **Мастера**, вы полните следующие действия:

1. Откройте требуемую базу данных Access (например, учебную базу данных **Личная картотека).**
2. В области переходов щелкните название того объекта базы данных (таблицы или запроса), который должен использоваться в качестве источника данных для нового отчета. Для приведенного примера щелкните имя таблицы **Мои контакты.**
3. Щелкните вкладку **Создание.**
4. В группе **Отчеты** щелкните кнопку **Мастер отчетов.** На экране появится первое диалоговое окно **Мастера отчетов**, в котором нужно выбрать исходную таблицу или запрос и определить набор полей для нового отчета.
5. Раскройте список **Таблицы и запросы** и выберите название объекта базы данных (таблицы или запроса), который должен использоваться в качестве источника данных для создания отчета. В примере мы создаем отчет на основе таблицы **Мои контакты,** поэтому в поле **Таблицы и запросы** оставьте значение по умолчанию.
6. В списке **Доступные поля** отметьте поля таблицы, которые должны быть включены в новый отчет. Если вы хотите использовать весь набор полей исходной таблицы, щелкните кнопку >>. Чтобы выбрать только часть полей таблицы, щелкните каждое требуемое поле и нажмите кнопку > для его переноса в список **Выбранные поля.**

Для отчета по таблице **Мои контакты** выберите поля **Имя, Фамилия, Мобильный телефон и Электронная почта.**

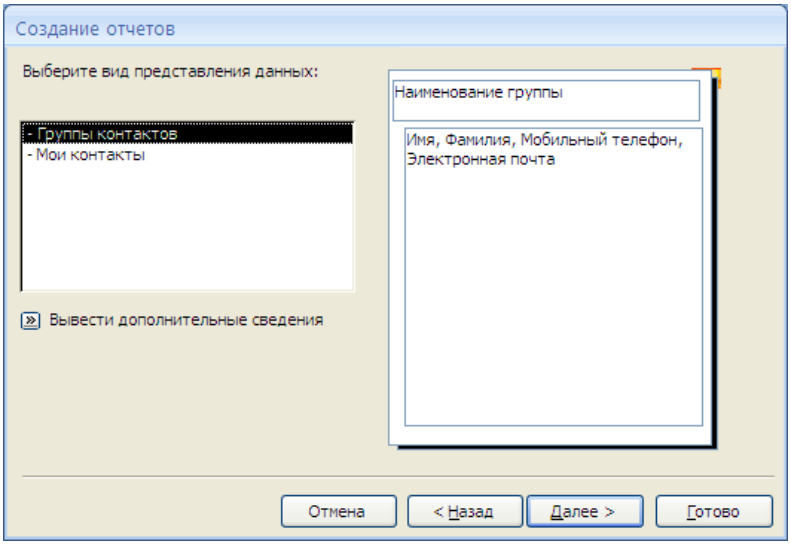
7. Для удобства использования отчета перечень контактов следует вывести по категориям (то есть добавить в отчет поле **Наименование группы** таблицы **Группы контактов** и установить группировку по этому полю). Раскройте список **Таблицы и запросы,** щелкните таблицу **Группы контактов** и добавьте поле **Наименование группы** в конец списка **Выбранные поля.**



Выбор полей отчета

8. Выберите один из видов представления данных, предложенных в окне **Мастера**. Для просмотра образца представления данных щелкните имя представления в списке. Выбранный образец будет показан в правой части окна **Мастера**.

Для приведенного примера щелкните представление **Группы контактов,** при котором контакты группируются по категориям.



Выбор представления данных

9. При необходимости выберите уровни группировки отчета:

– чтобы добавить уровень группировки по полю, щелкните название поля в списке доступных полей и нажмите кнопку [>];

– чтобы изменить порядок уровней группировки, используйте кнопку для [↑] перемещения уровня группировки на одну позицию вверх или кнопку [↓] для перемещения на одну позицию вниз;

– чтобы удалить уровень группировки по полю, щелкните название уровня на образце представления данных отчета и нажмите кнопку [<]

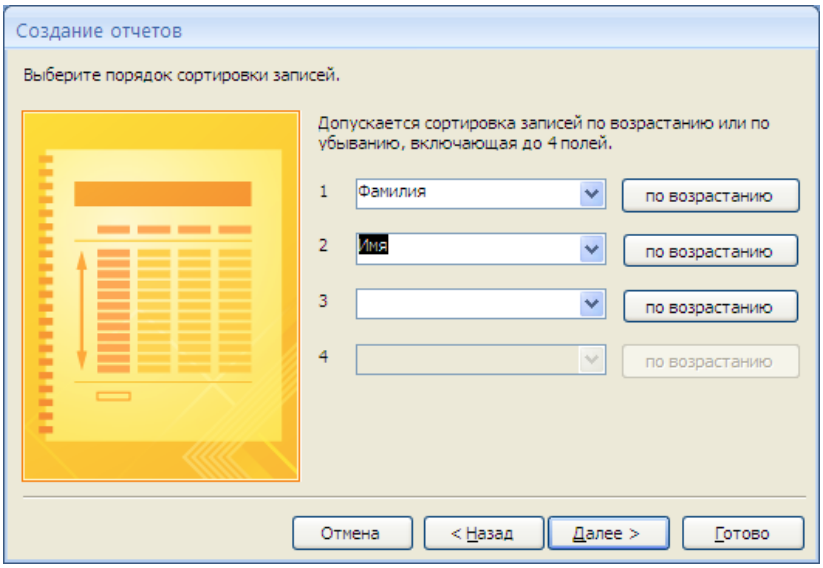
Для приведенного примера дополнительная группировка данных не требуется.

10. Нажмите кнопку Далее. На экране появится четвертое окно **Мастера.**

11. Установите желаемый порядок сортировки данных отчета. В окне **Мастера** вы можете выбрать до четырех полей сортировки, установив сортировку по возрастанию или по убыванию в каждом из отмеченных полей.

– Выберите поля для сортировки. Чтобы установить сортировку данных по какому-либо полю, раскройте список и щелкните название поля.

– Выберите способ сортировки. По умолчанию данные в щелкните кнопку **По возрастанию,** расположенную справа от списка (при этом ее название изменится на **По убыванию).**

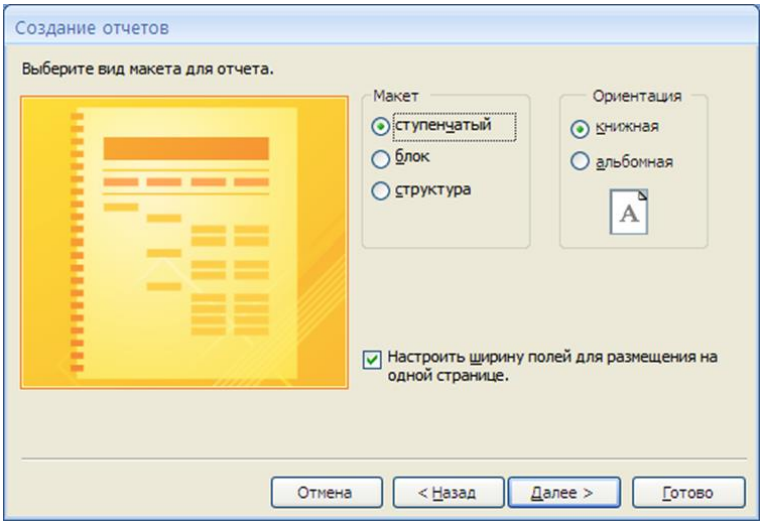


Выбор порядка сортировки

Для приведенного примера установите сортировку по полям **Фамилия** и **Имя.** Раскройте первый список и выберите поле **Фамилия,** а затем раскройте второй список и выберите поле **Имя**. Порядок сортировки по умолчанию изменять не нужно.

12. Нажмите кнопку **Далее.** На экране появится пятое окно **Мастера**.

13. В группе **Макет** выберите вид макета отчета, отметив один из предложенных вариантов:



Пятое окно Мастера

- **Ступенчатый** – при установке этого переключателя новый отчет будет оформлен по макету *«Ступенчатый»*;
- **Блок** – при установке этого переключателя новый отчет будет оформлен по макету *«Блок»*;
- **Структура** – при установке этого переключателя новый отчет будет оформлен по макету *«Структура»*.

Для приведенного примера оставьте макет *Ступенчатый*, предложенный по умолчанию.

14. В группе **Ориентация** выберите ориентацию страницы отчета, отметив один из предложенных вариантов:

- **Книжная** – при установке этого переключателя новый отчет будет иметь книжную ориентацию;
- **Альбомная** – при установке этого переключателя новый отчет будет иметь альбомную ориентацию.

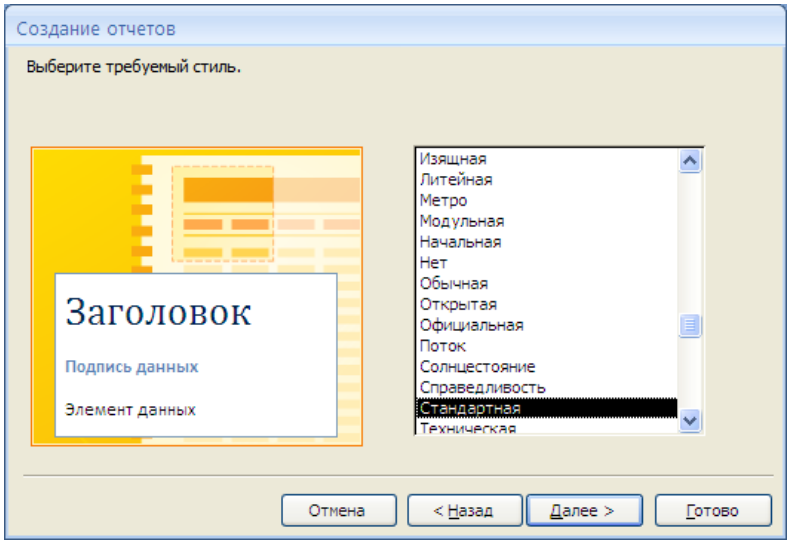
Для приведенного примера оставьте ориентацию **Книжная,** выбранную по умолчанию.

15. Если вы не хотите, чтобы ширина полей подстраивалась под ширину страницы отчета, снимите флажок **Настроить ширину полей для размещения на одной странице.** По умолчанию этот флажок установлен, и поля отчета имеют такую ширину, чтобы все они могли разместиться на одной странице.

16. Посмотрите образец оформления отчета с выбранными настройками в левой части окна **Мастера**.

17. Нажмите кнопку Далее. На экране появится шестое окно **Мастера**.

В предложенном списке выберите желаемый стиль оформления нового отчета. Для просмотра образца стиля щелкните название стиля в списке. Для приведенного примера выберите схему **Яркая** вместо схемы **Стандартная**, установленной по умолчанию.



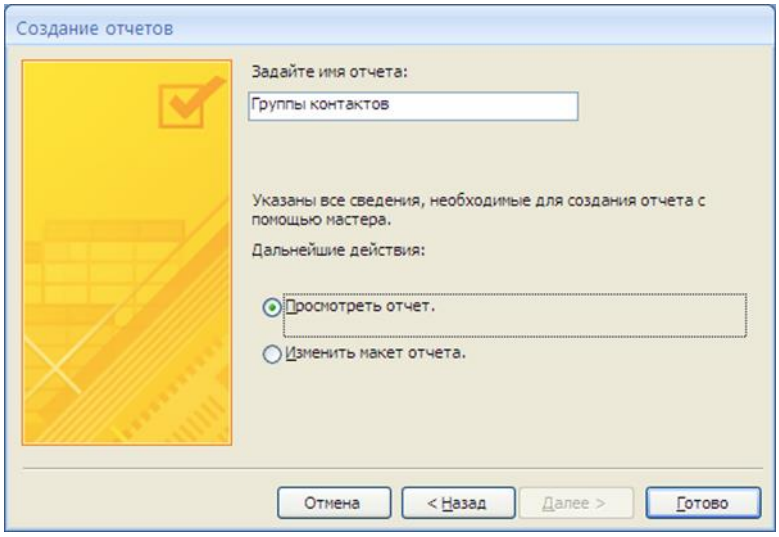
Шестое окно Мастера

18. Нажмите кнопку **Далее.** На экране появится последнее окно **Мастера.**

19. Введите название нового отчета в верхнее поле ввода. Для приведенного примера наберите название *Мастер-отчет Справочник контактов*.

20. В группе **Дальнейшие действия** оставьте вариант **Просмотреть отчет,** выбранный по умолчанию. В этом случае сразу после завершения работы **Мастера** можно будет просмотреть результирующий вид созданного отчета.

21. Нажмите кнопку **Готово.** Новый отчет будет сохранен в текущей базе данных. В рабочей области Access откроется окно предварительного просмотра созданного отчета.



Последнее окно Мастера

Мастер-отчет Справочник контактов

Наименование группы	Фамилия	Имя	Отчество	Мои контакты_К
Коллеги				
	Иванов	Александр	Петрович	Коллеги
	Сидоров	Василий	Иванович	Коллеги
	Сотникова	Вера	Львовна	Коллеги
Друзья				
	Иванов	Витя		Друзья
	Петров	Коля		Друзья
	Щеглова Света	Света		Друзья
Знакомые				
	Александров	Иннокентий	Александрович	Знакомые
	Сидорова	Светлана	Михайловна	Знакомые

21 февраля 2010 г. Стр. 1 из 1

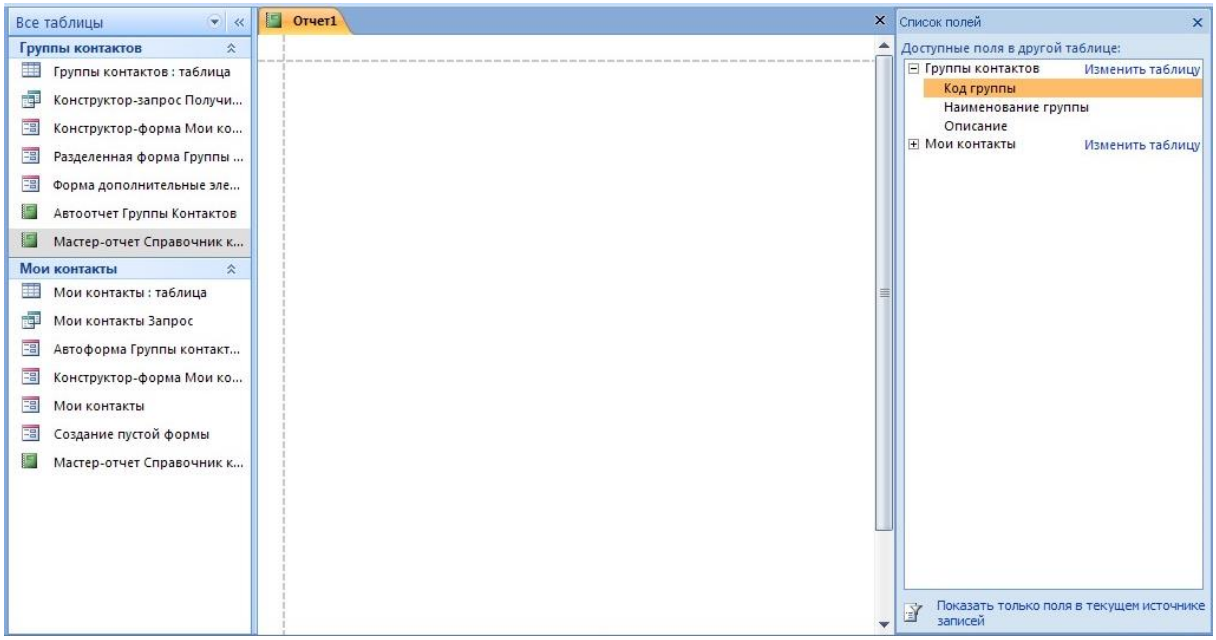
Отчет созданный «Мастером»

7.4 Создание пустого отчета

Пустой отчет — это очень быстрый способ создания отчета, особенно если в новом отчете должно содержаться небольшое количество полей.

Чтобы создать пустой отчет в базе данных Access, выполните следующие действия:

1. Откройте требуемую базу данных Access (например, учебную базу данных **Личная картотека).**
2. Щелкните вкладку **Создание.**
3. В группе **Отчеты** щелкните кнопку **Пустой отчет.** Приложение Access создаст пустой отчет и отобразит его в режиме макета.
4. В области **Список полей** щелкните знак плюс (+) рядом с таблицей, содержащей поля, которые нужно включить в отчет. Если в отчете должны выводиться данные из полей нескольких таблиц, раскройте списки полей всех нужных вам таблиц. В примере мы создадим отчет по таблице **Мои контакты,** поэтому щелкните знак плюс перед ее названием. В результате на экране появится перечень полей таблицы **Мои контакты**.

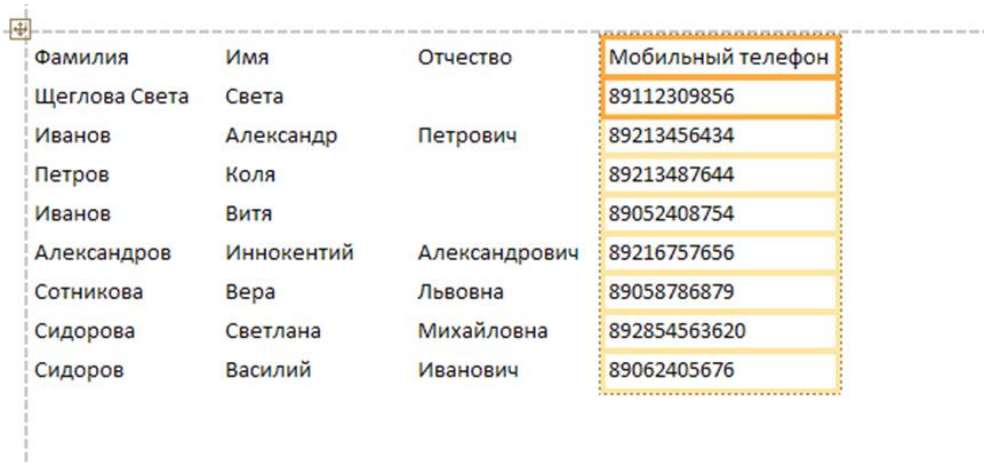


Пустой отчет

5. Добавьте в пустой макет нового отчета все нужные вам поля таблицы. Для вставки поля в

отчет дважды щелкните название этого поля в области **Список полей.** Значения выбранного поля появятся в отчете.

В примере в отчет были добавлены поля **Фамилия, Имя, Отчество и Мобильный телефон**.



Фамилия	Имя	Отчество	Мобильный телефон
Щеглова Света	Света		89112309856
Иванов	Александр	Петрович	89213456434
Петров	Коля		89213487644
Иванов	Витя		89052408754
Александров	Иннокентий	Александрович	89216757656
Сотникова	Вера	Львовна	89058786879
Сидорова	Светлана	Михайловна	892854563620
Сидоров	Василий	Иванович	89062405676

Отчет с добавленными полями

6. При необходимости установите сортировку данных в отчете. Например, фамилии людей следует выводить в алфавитном порядке, поэтому щелкните правой кнопкой мыши название поля **Фамилия** и выберите команду **Сортировка от А до Я в** появившемся контекстном меню.

7. Настройте параметры отображения данных в отчете, воспользовавшись командами контекстной вкладки **Формат**, расположенными в группе **Шрифт.**

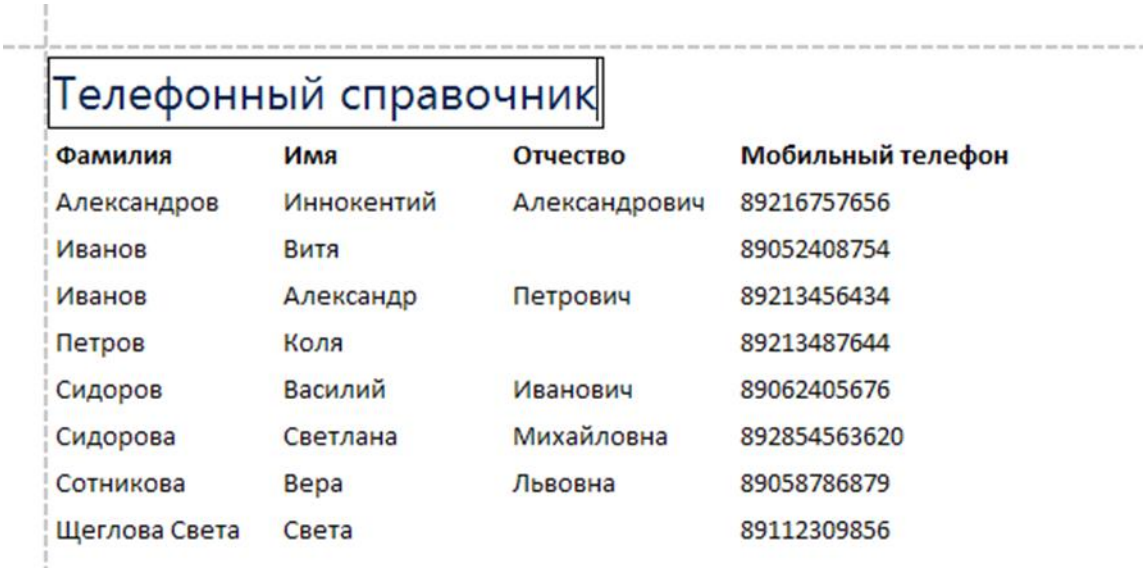
Для этого щелкните или выделите требуемый элемент данных отчета и измените его форматирование (оформление) аналогично тому, как это делается при оформлении текста действиям в редакторе Microsoft Word 2007. Инструменты все те же самые.

В качестве примера последовательно щелкните заголовки столбцов отчета **(Фамилия, Имя, Отчество и Мобильный телефон)** и установите для них полужирный шрифт.

8. При необходимости добавьте в отчет дополнительные элементы, воспользовавшись командами контекстной вкладки **Формат**. Для этого щелкните одну или несколько кнопок группы **Элементы управления:**

- **Эмблема** – служит для выбора изображения, которое будет использоваться в качестве эмблемы нового отчета;
- **Заголовок** – позволяет добавить или изменить заголовок нового отчета;
- **Номер страницы** – служит для вставки номеров страниц;
- **Дата и время** – добавляет в отчет элемент, отображающий текущее значение даты и / или времени.

В качестве примера мы добавим в отчет строку заголовка. Для создания заголовка щелкните кнопку **Заголовок** и введите желаемый заголовок отчета (например, Телефонный справочник) в поле, появившееся в верхней части макета отчета. Окончив ввод заголовка, нажмите клавишу Enter. В верхней части отчета появится новый заголовок.



Телефонный справочник

Фамилия	Имя	Отчество	Мобильный телефон
Александров	Иннокентий	Александрович	89216757656
Иванов	Витя		89052408754
Иванов	Александр	Петрович	89213456434
Петров	Коля		89213487644
Сидоров	Василий	Иванович	89062405676
Сидорова	Светлана	Михайловна	892854563620
Сотникова	Вера	Львовна	89058786879
Щеглова Света	Света		89112309856

Отчет с заголовком

9. При необходимости примените один из стандартных стилей оформления отчета. Для этого перейдите на контекстную вкладку **Формат**, в группе **Автоформат** и в раскрывшемся списке щелкните понравившийся образец схемы оформления отчета (например **Поток**). Отчет будет отформатирован в соответствии с выбранной схемой.

Телефонный справочник

Фамилия	Имя	Отчество	Мобильный телефон
Александров	Иннокентий	Александрович	89216757656
Иванов	Витя		89052408754
Иванов	Александр	Петрович	89213456434
Петров	Коля		89213487644
Сидоров	Василий	Иванович	89062405676
Сидорова	Светлана	Михайловна	892854563620
Сотникова	Вера	Львовна	89058786879
Щеглова Света	Света		89112309856

Готовый отчет

10. Нажмите кнопку **Сохранить** на панели быстрого доступа, чтобы сохранить готовый отчет в текущей базе данных.

11. Введите желаемое название отчета в поле **Имя отчета** (например, Пустой отчет Телефонный справочник) и нажмите кнопку **ОК.** От чет будет сохранен в базе данных, а его название появится в области переходов.

7.5 Создание отчета в Конструкторе

Подобно ранее изученным нами формам, любой отчет Access может состоять из нескольких разделов, каждый из которых имеет свое функциональное назначение. Чтобы создавать правильно работающие отчеты, пользователю необходимо хорошо понимать назначение каждого раздела отчета. В частности, от выбора раздела для размещения вычисляемого элемента управления напрямую зависит способ вычисления результата и, следовательно, итоговое значение этого элемента.

Просматривать и изменять структуру отчета удобнее всего в режиме Конструктора отчетов.

Сейчас мы познакомимся с перечнем возможных разделов отчета, которые можно увидеть в Конструкторе, и кратко рассмотрим назначение каждого из этих разделов.

- **Заголовок отчета** – печатается только один раз в самом начале отчета. В заголовок отчета включается информация, которая обычно помещается на обложке, например, эмблема компании, название отчета или дата. Если в заголовке отчета размещен вычисляемый элемент управления, использующий статистическую функцию (например, Бит), значение этой функции (в рассматриваемом случае – сумма) рассчитывается для всего отчета. Заголовок отчета печатается перед верхним колонтитулом.
- **Верхний колонтитул** – печатается вверху каждой страницы отчета. Например, верхний колонтитул можно использовать в тех случаях, когда нужно, чтобы название отчета повторялось на каждой странице.
- **Заголовок группы** – печатается перед каждой новой группой записей отчета и обычно используется для печати названия группы. Например, если отчет о контактах сгруппирован по категориям контактов, в заголовках групп можно указать названия категорий контактов. Если в заголовок группы поместить вычисляемый элемент управления, использующий статистическую функцию (например, Бит), значение этой функции (сумма) будет рассчитываться только для текущей группы.
- **Область данных** — этот раздел отчета печатается один раз для каждой строки данных из источника записей. В области данных размещаются элементы управления, составляющие основное содержание отчета.
- **Примечание группы** – печатается в конце каждой группы записей отчета. Примечание группы можно использовать для печати сводной информации по группе.
- **Нижний колонтитул** – печатается внизу каждой страницы отчета. Используется для нумерации страниц и для печати постраничной информации.
- **Примечание отчета** — печатается один раз в самом конце отчета. Примечание отчета можно использовать для печати итогов и другой сводной информации по всему отчету.

При создании отчетов в режиме **Конструктора** можно использовать тот же набор элементов управления, что и при создании форм. Базовые элементы управления, использующиеся при создании отчетов, расположены в группе **Элементы управления** вкладки **Конструктор** на ленте инструментов.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Как создать простую таблицу БД?
2. Как перейти в режим конструктора?
3. Что такое область навигации (переходов)?
4. Чем режим конструктора отличается от режима таблицы?
5. Для чего используются маска ввода?
6. Как установить маску ввода?
7. Что такое мастер подстановок?
8. Для чего используются индексы?
9. Как произвести индексирование?
10. Какие типы данных используются в MS Access 2013?
11. Как создать простую форму БД?
12. Для чего используются формы БД?
13. Назовите способы создания форм?
14. В чем особенность пустой формы?
15. Как создать форму с помощью мастера?
16. Какие свойства есть у форм?
17. Расскажите об элементах управления формой?
18. Как добавить на форму элементы управления?
19. Какие дополнительные элементы есть у формы?
20. Перечислите и расскажите о разделах формы.
21. Как создать простой запрос в БД?
22. Для чего используются запросы БД?
23. Перечислите способы создания запросов?
24. Чем создание запроса в режиме конструктора отличается от мастера запросов?
25. Для чего используются отчеты?

26. Перечислите способы создания отчетов?
27. Что такое ключевое поле?
28. Как создать связь между таблицами в MS Access?
29. Как задать перекрестный запрос?
30. Что такое выборка с группировкой?
31. Что такое активный запрос?

САМОСТОЯТЕЛЬНАЯ РАБОТА №1

Цель работы:

1. Изучение приемов задания структуры таблиц базы данных, заполнения их записями, установления связи между ними.

1 Создание таблиц в СУБД Access Связи между таблицами

Задание 1 – Создание таблиц

1. Создать папку *Работы по Access* в папке со своей *фамилией*. создать новую базу данных с именем **Центр-Сервис**. Сохранить ее в папке *Работы по Access*.
2. Создать таблицу **Услуги** с помощью режима **Таблица**.

Название поля	Тип данных	Примечания
Код услуги	Счетчик	Первичный ключ
Наименование услуги	Текстовый	
Стоимость	Числовой	
Количество дней	Числовой	

Далее в режиме **Конструктора** скорректировать таблицу.

3. Создать таблицу **Контакты** с помощью **Мастера таблиц**, просмотрев список предлагаемых таблиц для делового и личного применения. включить необходимые поля. Далее в режиме **Конструктора** скорректировать таблицу.

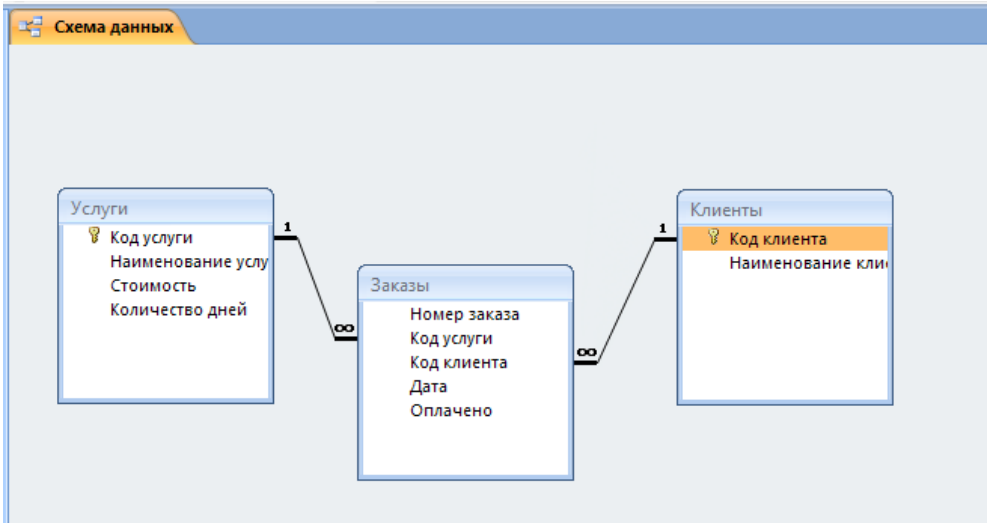
Название поля	Тип данных	Примечания
Код клиента	Счетчик	Первичный ключ
Наименование клиента	Текстовый	

4. Создать таблицу **Заказы** в режиме **Конструктора**.

Название поля	Тип данных	Примечания
Номер заказа	Числовой	
Код услуги	Числовой	
Код клиента	Числовой	
Дата	Дата/Время	Краткий формат даты
Оплачено	Логический	

Задание 2 – Построение связи между таблицами

1. Создать схему данных, вкладка **Работа с базами данных** команда **Схема данных**



Установите флажки *Обеспечение целостности данных* и *Каскадное обновление связанных полей*.

2. Заполните таблицы **Услуги** и **Клиенты** данными. Например:

Таблица **Услуги**			
Код услуги	**Наименование услуги**	**Стоимость**	**Количество дней**
1	Ремонт холодильника	3500	4
2	Установка спутникового телевидения	6000	2
3	Сборка компьютера	1500	1
4	Ремонт ресивера	2500	7
5	Подключение телефона	7500	2
6	Подключение компьютера	2000	2

Таблица **Услуги**			
Код услуги	**Наименование услуги**	**Стоимость**	**Количество дней**
7	Диагностика компьютера	500	5
8	Монтаж сети	8000	3
9	Подключение Интернет	1600	1

Таблица **Клиенты**	
Код клиента	**Наименование клиента**
1	ООО «Прометей»
2	ОАО «ЕлецГазСтрой»
3	ЗАО «Юбилей»
4	ООО «ЕлецУниСтрой»
5	ОАО «Газпром»
6	ТОО «Коспар»
7	ЧП Меркулов С.А.

3. Сохранить базу данных в своей папке.

САМОСТОЯТЕЛЬНАЯ РАБОТА №2

Цель работы:

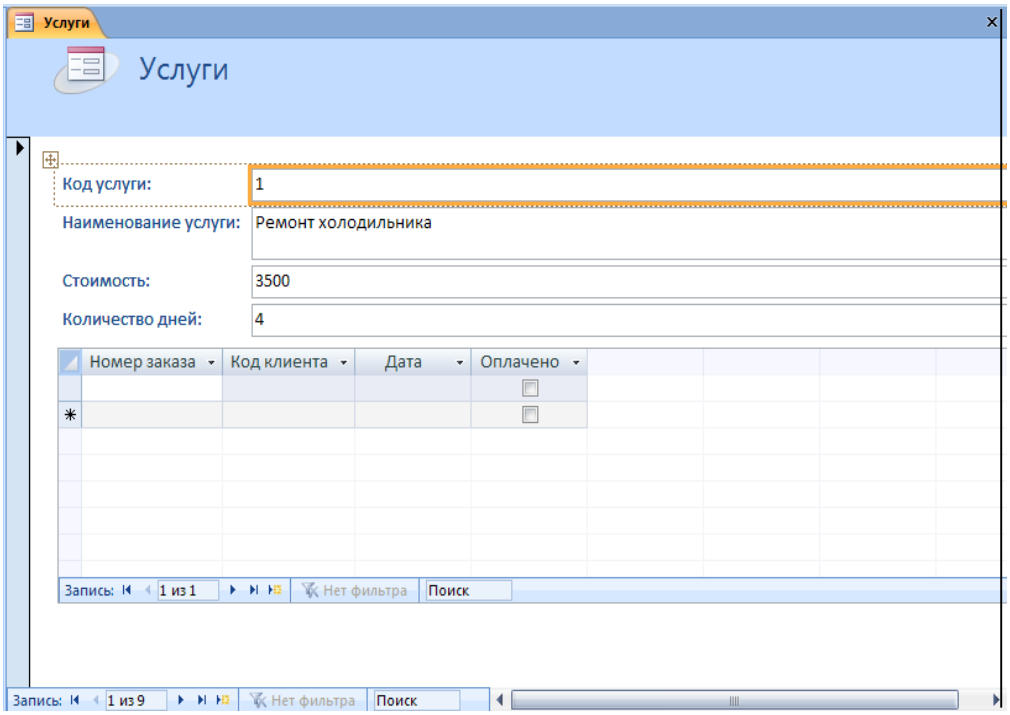
1. Изучение приемов создания простых форм, базирующихся на таблицах; проведение настройки формы с помощью Конструктора.

1 Создание форм в СУБД ACCESS

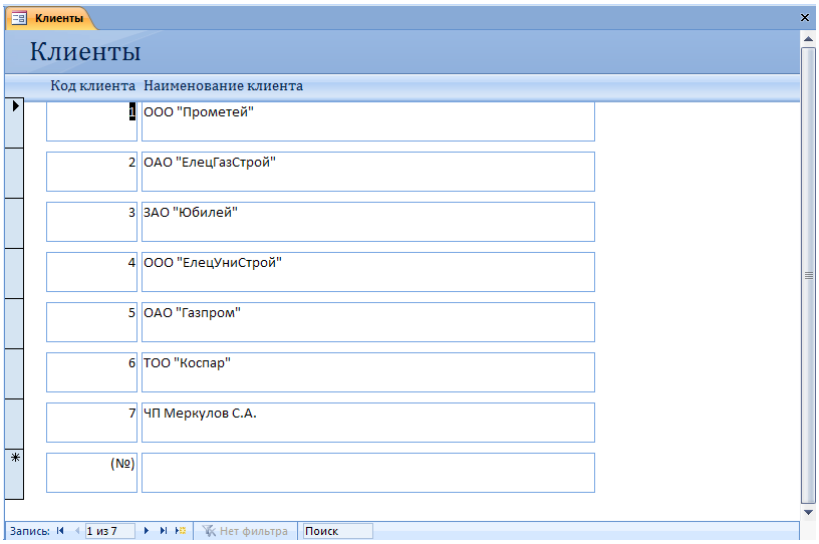
Задание 1 – Создание простых форм, базирующихся на таблицах.

1. Открыть из папки *Работы по Access, базу* данных **Центр-Сервис**. Перейдите на вкладку **Создание** команда **Форма**.

2. Создать форму для таблицы **Услуги**. Закрыть новую форму после ее демонстрации на экране, сохранив под именем ***Услуги***.



3. Создать с помощью **Мастера** форму для таблицы **Клиенты**, выбрав все поля таблицы, вид формы – *ленточный*, стиль *стандартный*. сохранить под именем ***Клиенты***.



4. Создать форму для таблицы **Заказы**, с помощью *Автоформы*. закрыть новую форму после ее демонстрации на экране, сохранив под именем **Заказы**.

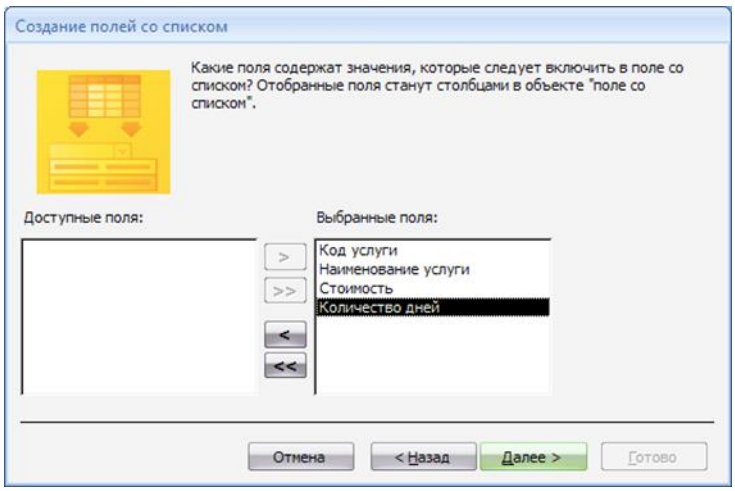
Задание 2 – Настройка формы с помощью Конструктора.

1. Произвести настройку формы **Заказы**. При автоматической генерации формы средствами Access, поля *Код услуги* и *Код клиента* представлены полями для ввода, что делает ненаглядным заполнение таблицы данными, т.к. необходимо помнить коды услуг и коды клиентов из соответствующих таблиц Клиенты и услуги. Оптимальным решением является выбор значения из справочника. Для этого служит элемент управления, называемый полем со списком.

Для настройки формы **Заказы** необходимо ее активизировать и войти в режим **Конструктора**. Далее найти и активизировать вместе с подписью поля Код услуги и Код клиента. Нажать клавишу **Delete**, удалив выбранные поля. Проверить правильность выполнения операции, переключившись в режим **Формы**.

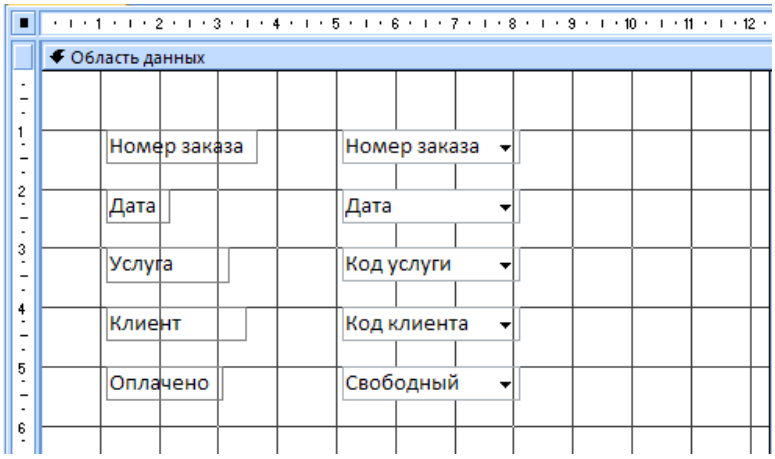
Для добавления элемента *Поле со списком* щелкнуть по соответствующему инструменту на

вкладке **Конструктор**, затем щелкнуть в разделе *Область данных* формы. Появится диалоговое окно *Создание полей со списком*, в котором необходимо выбрать способ, которым поле со списком получает свои значения. В данном случае оставить предлагаемый вариант – выбор из таблицы или запроса. Далее, во втором диалоговом окне выбрать конкретную таблицу **Услуги**. В третьем диалоговом окне выбрать отдельные реквизиты таблицы для списка с помощью кнопки >, либо все с помощью кнопки >>. В создаваемую форму включить все реквизиты.



В следующем диалоговом окне оставить все без изменений и перейти к окну, где нужно указать поле *Код услуги* как источник для значений. В последнем диалоговом окне ввести подпись к полю – **Услуга**. После всех действий нажать кнопку **Готово.**

Аналогично добавить поле со списком для выбора клиента по его наименованию из таблицы **Клиенты**. После выполнения всех действий в режиме Конструктора форма **Заказы** примет вид:



Сохранить форму и закрыть ее окно.

2. С помощью формы заполнить таблицу Заказы. Например:

Таблица Заказы					
Номер заказа	**Наименование клиента**	**Дата**	**Наименование услуги**	**Стоимость**	**Оплачено**
520	ОАО "ЕлецГазСтрой"	25.05.2006	Ремонт ресивера	2500	Да
521	ОАО "Газпром"	25.05.2006	Установка спутникового телевидения	6000	Да
522	ТОО "Коспар"	25.05.2006	Установка спутникового телевидения	6000	Да
523	ООО "ЕлецУниСтрой"	25.05.2006	Ремонт холодильника	3500	Нет
524	ЧП Меркулова С.А.	25.05.2006	Диагностика компьютера	500	Да

525	ЧП Меркулова С.А.	26.05.2006	Монтаж сети	8000	Да
526	ООО "Прометей"	26.05.2006	Ремонт ресивера	2500	Нет
527	ТОО "Коспар"	26.05.2006	Сборка компьютера	1500	Да
528	ЗАО "Юбилей"	27.05.2006	Подключение телефона	7500	Да
529	ЗАО "Юбилей"	27.05.2006	Установка спутникового телевидения	6000	Да
530	ОАО "Газпром"	27.05.2006	Подключение компьютера	2000	Да
531	ТОО "Коспар"	29.05.2006	Подключение Интернет	1600	Да
532	ОАО "ЕлецГазСтрой"	01.06.2006	Диагностика компьютера	500	Да
533	ООО "Прометей"	01.06.2006	Ремонт холодильника	3500	Да
534	ТОО "Коспар"	02.06.2006	Подключение компьютера	2000	Нет
535	ЧП Меркулова С.А.	03.06.2006	Монтаж сети	8000	Да
536	ООО "ЕлецУниСтрой"	03.06.2006	Подключение телефона	7500	Да
537	ЧП Меркулова С.А.	03.06.2006	Сборка компьютера	1500	Да
538	ЧП Меркулова С.А.	04.06.2006	Монтаж сети	8000	Да
539	ЗАО "Юбилей"	05.06.2006	Подключение Интернет	1600	Нет

3. Сохранить базу данных.

САМОСТОЯТЕЛЬНАЯ РАБОТА №3

Цель работы:

1. Изучение приемов создания запросов на выборку, перекрестных запросов, активных запросов.

1 Создание запросов в СУБД Access

Задание 1 – Создание запросов на выборку.

1. Открыть из папки *Работы по Access, базу* данных **Центр-Сервис**. Перейти на вкладку *Создание → Запросы*.

2. Создать запросы.

1) Название запроса: *Список всех клиентов* Источник данных: таблица *Клиенты* Результат выполнения запроса:

Клиенты Запрос	
Код клиента	**Наименование клиента**
1	ООО "Прометей"
2	ОАО "ЕлецГазСтрой"
3	ЗАО "Юбилей"
4	ООО "ЕлецГазСтрой"
5	ОАО "Газпром"
6	ТОО "Коспар"
7	ЧП Меркулов С.А.

2) Название запроса: *Список всех заказов, упорядоченных по клиентам, а затем по дате.* Источник данных: таблицы *Заказы, Услуги, Клиенты*. Результат выполнения запроса:

Таблица Заказы Запрос					
Номер заказа	**Наименование клиента**	**Дата**	**Наименование услуги**	**Стоимость**	**Оплачено**
521	ОАО "Газпром"	25.05.2006	Установка спутникового телевидения	6000	Да
520	ОАО "ЕлецГазСтрой"	25.05.2006	Ремонт ресивера	2500	Да
523	ООО "ЕлецУниСтрой"	25.05.2006	Ремонт холодильника	3500	Нет

522	ТОО "Коспар"	25.05.2006	Установка спутникового телевидения	6000	Да
524	ЧП Меркулова С.А.	25.05.2006	Диагностика компьютера	500	Да
526	ООО "Прометей"	26.05.2006	Ремонт ресивера	2500	Нет
527	ТОО "Коспар"	26.05.2006	Сборка компьютера	1500	Да
525	ЧП Меркулова С.А.	26.05.2006	Монтаж сети	8000	Да
529	ЗАО "Юбилей"	27.05.2006	Установка спутникового телевидения	6000	Да
528	ЗАО "Юбилей"	27.05.2006	Подключение телефона	7500	Да
530	ОАО "Газпром"	27.05.2006	Подключение компьютера	2000	Да
531	ТОО "Коспар"	29.05.2006	Подключение Интернет	1600	Да
532	ОАО "ЕлецГазСтрой"	01.06.2006	Диагностика компьютера	500	Да
533	ООО "Прометей"	01.06.2006	Ремонт холодильника	3500	Да
534	ТОО "Коспар"	02.06.2006	Подключение компьютера	2000	Нет
536	ООО "ЕлецУниСтрой"	03.06.2006	Подключение телефона	7500	Да
535	ЧП Меркулова С.А.	03.06.2006	Монтаж сети	8000	Да
537	ЧП Меркулова С.А.	03.06.2006	Сборка компьютера	1500	Да
538	ЧП Меркулова С.А.	04.06.2006	Монтаж сети	8000	Да
539	ЗАО "Юбилей"	05.06.2006	Подключение Интернет	1600	Нет

Примечание: т.к. в условии указан признак сортировки, то в отсутствующей строке выбрать порядок сортировки – *по возрастанию*.

3) Название запроса: *Список услуг стоимостью более 2000 р., упорядоченный по стоимости.*

Источник данных: таблица *Услуги*

Результат выполнения запроса:

Список услуг более 2000 р		
Наименование услуги	**Стоимость**	**Количество дней**
Ремонт ресивера	2500	7
Ремонт ресивера	2500	1
Ремонт холодильника	3500	4
Установка спутникового телевидения	6000	
Установка спутникового телевидения	6000	2
Подключение телефона	7500	2
Монтаж сети	8000	3

Примечание: Условие отбора записей вводится в одноименной строке в столбце, соответствующем условию задачи.

4) Название запроса: *Список услуг, которые выполняются не более чем за 2 дня или не менее чем за неделю.*

Источник данных: таблица *Услуги*. Результат выполнения запроса:

Список услуг выполняемых не менее чем за 2 дня		
Наименование услуги	**Стоимость**	**Количество дней**
Установка спутнткового телевидения	6000	2
Ремонт ресивера	2500	1
Ремонт ресивера	2500	7
Подключение телефона	7500	2
Подключение компьютера	2000	2
Подключение Интернет	1600	1

*Примечание: Если в запросе имеется два и более условия, связанные отношением или, то они должны отражаться на различных строках группы полей Условие отбора или на одной строке, но с использованием функций языка **SQL**, например, **<=2 Or >=7**.*

5) Название запроса: *Список услуг, в наименовании которых встречается последовательность символов «ключ»*

Источник данных: таблица *Услуги*

Результат выполнения запроса:

Запрос		
Наименование услуги	**Стоимость**	**Количество дней**
Подключение телефона	7500	2
Подключение компьютера	2000	2
Подключение Интернет	1600	1

Примечание: Для обозначения оставшейся части слова используют знак *******. После ввода условия *ключ*, СУБД автоматически изменит значение на выражение* ***Like "*ключ*".***

6) Название запроса: *список услуг стоимостью более 4000р., которые выполняются не более чем за 3 дня.*

Источник данных: таблица *Услуги*

Результат выполнения запроса:

Запрос		
Наименование услуги	**Стоимость**	**Количество дней**
Установка спутникового телевидения	6000	2
Подключение телефона	7500	2
Монтаж сети	8000	3

Примечание: Если в запросе имеются два или более условия, связанные отношением и, то они обязательно должны располагаться на одной строке поля Условие отбора.

7) Название запроса: *список неоплаченных заказов.* Источник данных: таблица *Услуги, Заказы, Клиенты.* Результат выполнения запроса:

Запрос Неоплаченные заказы					
Номер заказа	**Наименование клиента**	**Дата**	**Наименование услуги**	**Стоимость**	**Оплачено**
526	ООО "Прометей"	26.05.2006	Ремонт ресивера	2500	Нет
526	ООО "Прометей"	26.05.2006	Ремонт ресивера	2500	Нет
534	ТОО "Коспар"	02.06.2006	Подключение компьютера	2000	Нет
539	ЗАО "Юбилей"	05.06.2006	Подключение Интернет	1600	Нет

Примечание: В условиях отбора логического поля вводится текст, соответствующий запросу – ***Нет*** *или* ***No, Да*** *–* ***Yes.***

8) Название запроса: *Список заказов на определенный день, значение которого является параметром запроса.*

Источник данных: таблица *Заказы, Услуги, Клиенты.*

Результат выполнения запроса (например, для даты 01.06.2006):

Запрос по определенной дате				
Номер заказа	**Дата**	**Наименование клиента**	**Наименование услуги**	**Стоимость**
532	01.06.2006	ОАО "ЕлецГазСтрой"	Диагностика компьютера	500
533	01.06.2006	ООО "Прометей"	Ремонт холодильника	3500

Примечание: В запросе с параметрами в строку Условия отбора напротив имени поля, для которого задается условие, записывается не конкретное значение, а указывается условие его ввода. Оно

представляет собой произвольное предложение, которое заключается в квадратные скобки. При выполнении такого запроса сначала должно выводиться диалоговое окно с подсказкой, а затем, после ввода с клавиатуры необходимой информации и нажатии клавиши ***Enter****, выдается результат выполнения запроса.*

9) Название запроса: *список заказов за определенный период.*

Источник данных: таблица *Заказы, Услуги, Клиенты.*

Результат выполнения запроса (например, на период с 27.05.2006 до 01.06.2006):

Запрос за определенный период времени				
Номер заказа	**Дата**	**Наименование клиента**	**Наименование услуги**	**Количество дней**
528	27.05.2006	ЗАО "Юбилей"	Подключение телефона	2
529	27.05.2006	ЗАО "Юбилей"	Установка спутникового телевидения	2
530	27.05.2006	ОАО "Газпром"	Подключение компьютера	2
531	29.05.2006	ТОО "Коспар"	Подключение Интернет	1
532	01.06.2006	ОАО "ЕлецГазСтрой"	Диагностика компьютера	5
533	01.06.2006	ООО "Прометей"	Ремонт холодильника	4

Примечание: запрос аналогичен предыдущему с той разницей, что вместо ввода конкретный значений границ периода задаются два параметра запроса, соответствующие начальной и конечной дате. В данном случае условия, связанные отношением и, записываются в одной ячейке строки Условия отбора, при этом применяются операторы языка ***SQL >, <, =, <=, >=, And****.*

10) Название запроса: *список оплаченных клиентом заказов.*

Источник данных: таблица *Заказы, Услуги, Клиенты.*

Результат выполнения запроса (например, для клиента ЗАО «Юбилей»):

Запрос список оплаченных клиентом услуг					
Номер заказа	**Наименование клиента**	**Дата**	**Наименование услуги**	**Стоимость**	**Оплачено**
528	ЗАО "Юбилей"	27.05.2006	Подключение телефона	7500	Да
529	ЗАО "Юбилей"	27.05.2006	Установка спутникового телевидения	6000	Да

Примечание: в запросе условия связаны отношением и, причем одно из них задается в строке Условие отбора, а второе является параметром запроса.

Задание 2 – Создание запросов на выборку с вычисляемым полем.

1) Название запроса: *список невыполненных заказов.*

Источник данных: таблица *Заказы, Услуги, Клиенты.*

Результат выполнения запроса (например, для текущей даты 03.06.2006):

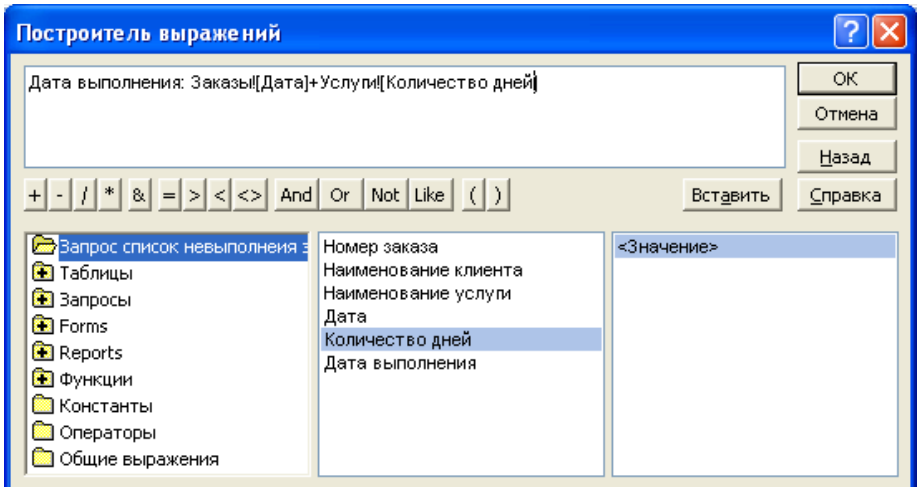
Запрос список невыполнения заказа					
Номер заказа	**Наименование клиента**	**Наименование услуги**	**Дата**	**Количество дней**	**Дата выполнения**
535	ЧП Меркулов С.А.	Монтаж сети	03.06.2006	3	06.06.2006
536	ООО "УлецУниСтрой"	Подключение телефона	03.06.2006	2	05.06.2006
537	ЧП Меркулов С.А.	Сборка компьютера	03.06.2006	1	04.06.2006

Примечание: Дата выполнения заказа складывается из даты поступления и количества дней, необходимых на ремонт. Т.е. мы должны добавить еще одно поле, которое образуется путем вычисления по формуле: [Дата]+[Количество дней]. Для создания вычисляемого поля можно ввести выражение в строку Поле в свободном столбце или воспользоваться ***Построителем выражений.***

Для обеспечения решения поставленной задачи, необходимо установить ограничение на созданное поле с помощью функции ***Now ()***, которая возвращает текущую дату. Функция записывается в строке *Условие отбора* непосредственно с клавиатуры или через **Построитель выражений**.

Для проверки правильности запроса необходимо изменить значение текущей даты. Для этого можно:

- изменить в настройках компьютера текущую дату, присвоив ей необходимое значение;



- ввести в базу данных несколько новых записей с датами, близкими к системным;
- строке *Условия отбора* столбца Дата выполнения вместо функции ***Now ()***, ввести значение конкретной даты, например *#03.06.2006#.*

2) Название запроса: *список заказов за текущий месяц.*

Источник данных: таблица *Заказы, Услуги, Клиенты.*

Результат выполнения запроса (например, для текущей даты 03.06.2006):

Запрос Список заказов за текущий месяц					
Номер заказа	**Наименование клиента**	**Наименование услуги**	**Дата**	**Месяц**	**Год**
532	ОАО "ЕлецГазСтрой"	Диагностика компьютера	01.06.2006	6	2006
533	ООО "Прометей"	Ремонт холодильника	01.06.2006	6	2006
534	ТОО "Коспар"	Подключение компьютера	02.06.2006	6	2006
535	ЧП Меркулов С.А.	Монтаж сети	03.06.2006	6	2006
536	ООО "УлецУниСтрой"	Подключение телефона	03.06.2006	6	2006
537	ЧП Меркулов С.А.	Сборка компьютера	03.06.2006	6	2006
539	ЗАО "Юбилей"	Подключение Интернет	05.06.2006	6	2006

Примечание: Номер месяца определяется с помощью функции Month, параметром которой является какая-либо дата. Т.е. в запросе необходимо создать еще одно поле Месяц, построив выражение Месяц: Month ([Дата]). При этом необходимо задать условие отбора Month(Now()) для текущей даты, или Month(#03.06.2006#) – для конкретно определенной даты. Т.к. в базе данных могут находиться записи прошлых лет, необходимо аналогичным образом организовать выбор по году.

3) Название запроса: *Список услуг со стоимостью, увеличенной на проценты (параметры запроса) относительно исходной.*

Источник данных: таблица *Услуги.*

Результат выполнения запроса (например, при значении параметра 30 %):

Запрос Стоимость увеличенная на проценты		
Наименование услуги	**Стоимость**	**Новая стоимость**
Ремонт холодильника	3500	4550
Установка спутникового телевидения	6000	7800
Сборка компьютера	1500	1950
Ремонт ресивера	2500	3250
Подключение телефона	7500	9750
Подключение компьютера	2000	2600
Диагностика компьютера	500	650
Монтаж сети	8000	10400
Подключение Интернет	1600	2080

Примечание: В запросе получается комбинация запроса с параметром и запроса с вычисляемым полем. Для решения задачи необходимо создать новое вычисляемое поле Новая стоимость, которое будет определяться выражением [Стоимость](100+[на сколько % увеличить])/100.*

Задание 3 – Создание запросов на выборку с группировкой.

1) Название запроса: *Стоимость услуг для клиентов.* Источник данных: таблицы Заказы, *Услуги, Клиенты.* Результат выполнения запроса:

Запрос Стоимость услуг для клиентов	
Наименование клиента	**Sum-Стоимость**
ООО "Прометей"	6000
ОАО "ЕлецГазСтрой"	3000
ЗАО "Юбилей"	15100
ООО "УлецУниСтрой"	11000
ОАО "Газпром"	8000

Запрос Стоимость услуг для клиентов	
Наименование клиента	**Sum-Стоимость**
ТОО "Коспар"	11100
ЧП Меркулов С.А.	26000

Примечание: Создание запроса с группировкой обеспечивается строкой Групповая операция в ***Конструкторе*** *запросов. Если строка отсутствует, то ее необходимо активировать с помощью*

одноименной кнопки на панели инструментов. Напротив поля, по которому происходит группировка, в строке Групповая операция выбирается из списка значение ***Группировка****. Для этого поля, в котором происходит вычисление, выбирается соответствующая операция (в данном запросе – операция* ***Sum****). Необходимо отметить, что, несмотря на отсутствие в выходном наборе полей из таблицы Заказы, в качестве источника данных указание данной таблицы обязательно, т.к. она обеспечивает связь между таблицами Клиенты и Услуги.*

2) Название запроса: *Стоимость оплаченных услуг по видам.*

Источник данных: таблицы Заказы, *Услуги.*

Результат выполнения запроса:

Запрос Стоимость оплаченных услуг по видам	
Наименование услуги	**Sum-Стоимость**
Ремонт холодильника	7000
Установка спутникового телевидения	18000
Сборка компьютера	3000
Ремонт ресивера	5000
Подключение телефона	15000
Подключение компьютера	4000
Диагностика компьютера	1000
Монтаж сети	24000
Подключение Интернет	3200

Примечание: В запросы с группировкой также можно включать дополнительные поля для отбора записей, при этом можно отключить вывод данного поля на экран.

Задание 4 – Создание перекрестных запросов.

1) Название запроса: *Общая стоимость заказов, выполненных предприятием для каждого клиента по каждой услуге.*

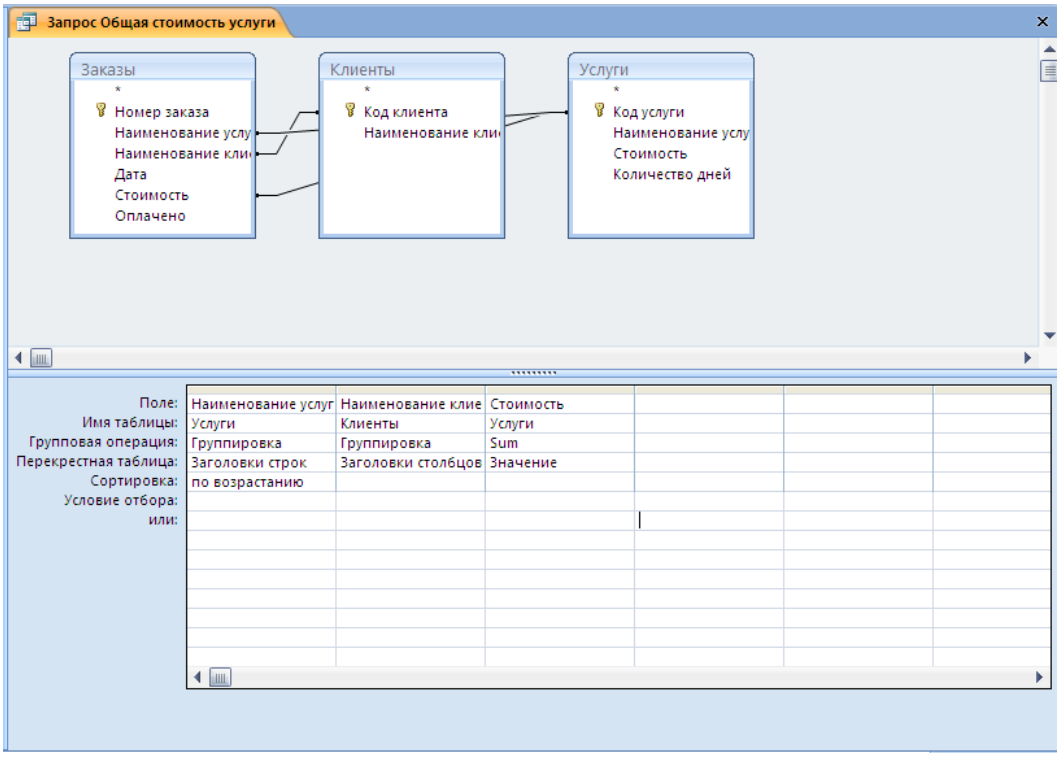
Источник данных: таблицы Заказы, *Услуги.*

Результат выполнения запроса:

Запрос Общая стоимость заказа							
Наименование услуги	**ЗАО Юбилей**	**ОАО Газпром**	**ОАО ЕлецГазСтрой**	**ООО Прометей**	**ООО УлецУниСтрой**	**ТОО Коспар**	**ЧП Меркулов С.А.**
Диагностика компьютера			500				500
Монтаж сети							24000
Подключение Интернет	1600					1600	
Подключение компьютера		2000				2000	
Подключение телефона	7500				7500		
Ремонт ресивера			2500	2500			
Ремонт холодильника				3500	3500		
Сборка компьютера						1500	1500
Установка спутникового телевидения	6000	6000				6000	

Примечание: В данном запросе необходимо сгруппировать данные по двум измерениям: строкам и столбцам. На пересечении каждого столбца и каждой строки рассчитывается итоговое значение, соответствующее заголовкам столбцов и строк. По умолчанию в режиме Конструктора проектируется запрос на выборку. Для перехода к перекрестному запросу необходимо выполнить команду Тип запроса: перекрестный, вкладки ***Конструктор*** *ленты инструментов, группа Тип запроса. В результате в бланк запроса будут добавлены две новые строки: Групповая операция и Перекрестная таблица и убрана строка Вывод на экран.*

В строку *Поле* необходимо поместить те поля, содержимое которых будет использоваться в качестве заголовков строк и заголовков столбцов перекрестной таблицы, а также расчетных значений.



Напротив каждого из таких полей в строке *Перекрестная таблица* из списка выбираются соответствующие значения: *Заголовки строк*, *Заголовки столбцов*, *Значения*. Для поля, по которому ведется расчет, в строке *Групповая операция* устанавливается значение, соответствующее условию запроса.

2) Название запроса: *Общая стоимость оплаченных заказов, выполненных предприятием по каждой услуге за каждый месяц.*

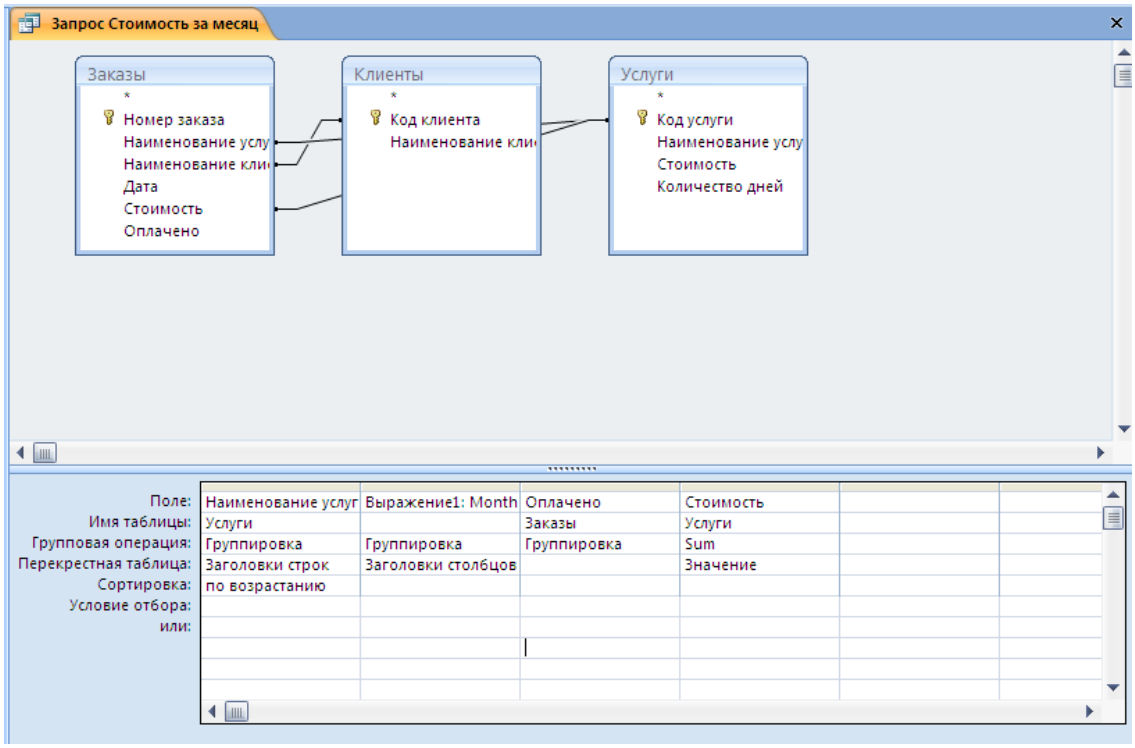
Источник данных: таблицы Заказы, *Услуги.*

Результат выполнения запроса:

Запрос Общая стоимость по каждой услуге за каждый месяц		
Наименование услуги	**5**	**6**
Диагностика компьютера	500	500
Монтаж сети	16000	8000
Подключение Интернет	1600	
Подключение Интернет		1600
Подключение компьютера	2000	
Подключение компьютера		2000
Подключение телефона	7500	7500
Ремонт ресивера	2500	
Ремонт ресивера	2500	
Ремонт холодильника		3500
Ремонт холодильника	3500	

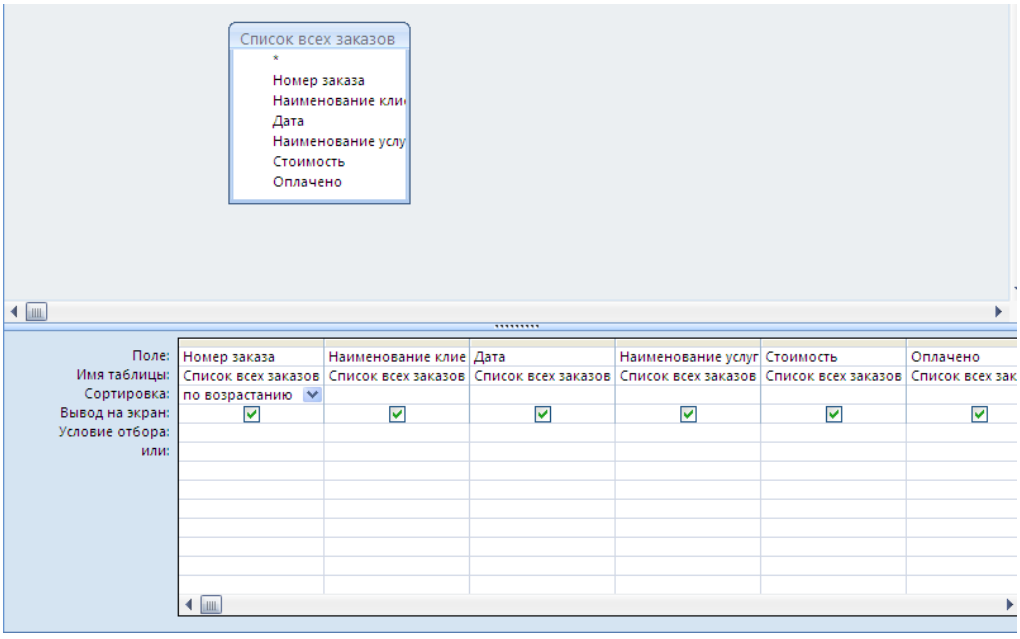
Запрос Общая стоимость по каждой услуге за каждый месяц		
Наименование услуги	**5**	**6**
Сборка компьютера	1500	1500
Установка спутникового телевидения	18000	

Примечание: Аналогично запросу с группировкой в перекрестном запросе можно использовать вычисляемые поля и условия на значение.



Задание 5 – Создание активных запросов.

1) Название запроса: *Создание новой таблицы, содержащей все заказы (таблица Все заказы).*
Источник данных: запрос Список всех заказов, упорядоченных по клиентам, а затем по дате.
Макет бланка запроса:



Примечание: В отличие от запросов на выборку, при создании активных запросов следует указать вид запроса (создание таблицы, добавление, удаление, обновление). После выполнения команды Создание таблицы, вкладка ***Конструктор*** *ленты инструментов группа* ***Тип запроса****, появляется диалоговое окно, в котором необходимо ввести имя создаваемой таблицы. Затем в строке Поле*

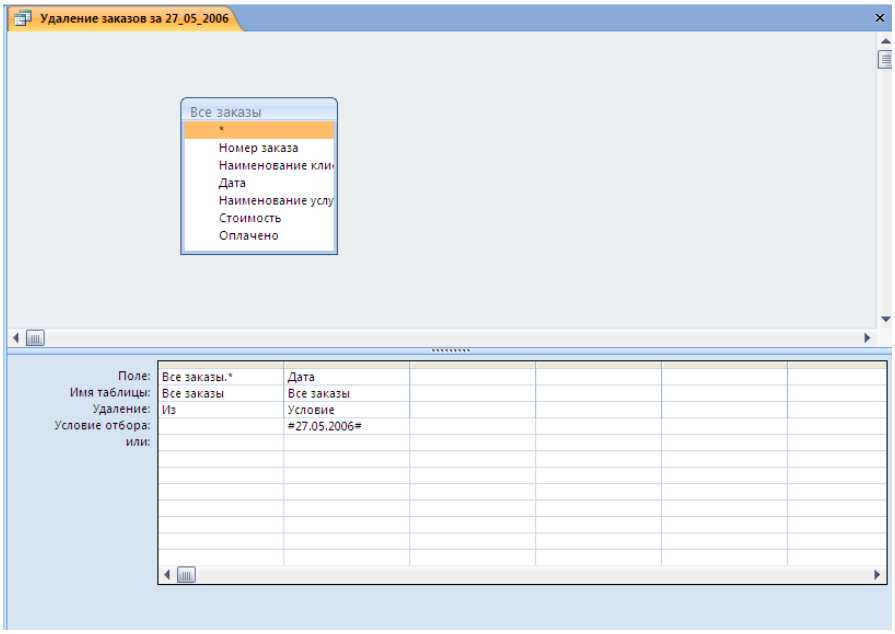
*помещаются поля запроса, значение которых будут использоваться в создаваемой таблице. Если потребуются все поля, то достаточно разместить в строке Поле символ *****, находящийся в списке полей запроса. Можно добавить поле для сортировки, но не отображать его.*

В результате выполнения запроса создается таблица *Все заказы*.

2) Название запроса: *Удаление заказов с датой оформления 27.05.2006.*

Источник данных: таблица *Все заказы.*

Макет бланка запроса:

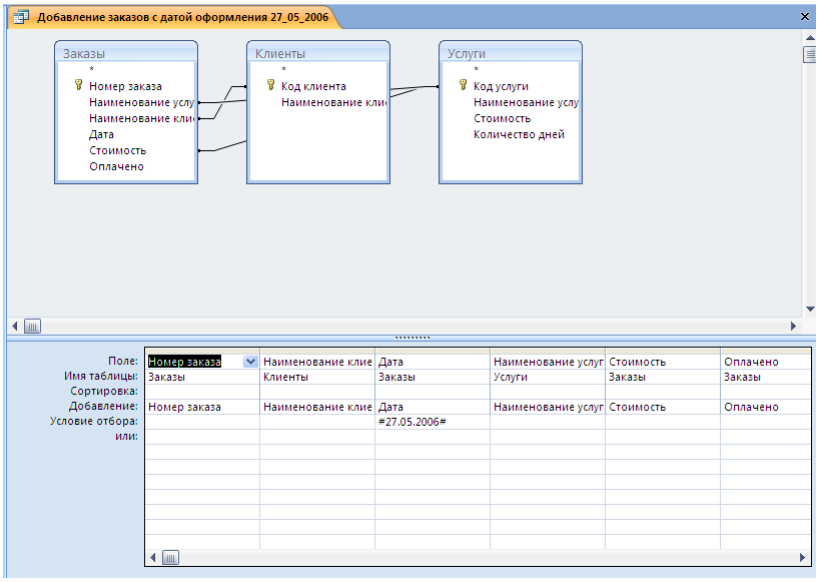


*Примечание: Для добавления в бланк строки Удаление необходимо выполнить команду **Тип запроса**: удаление. В бланк помещаются только те поля таблицы, по которым будут записываться условия на удаления записей. Также в строку Поле помещается имя таблицы, из которой это удаление будет происходить.*

3) Название запроса: *Добавление заказов с датой оформления 27.05.2006.*

Источник данных: таблицы *Заказы, Услуги, Клиенты.*

Макет бланка запроса:

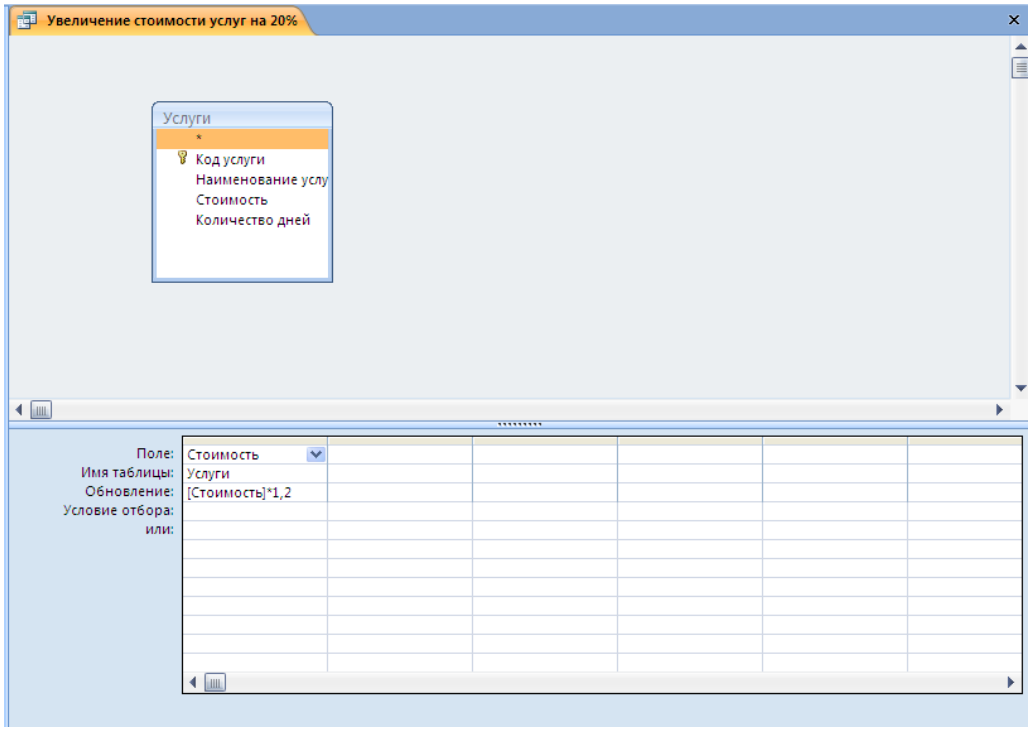


*Примечание. После выполнения команды **Тип команды**: добавление откроется диалоговое окно, в котором необходимо указать имя таблицы, в которую будут добавляться записи. В строку Поле помещаются имена тех полей, содержимое которых будет добавлено в таблицу Все заказы, соответствующие именам полей в строке Поле.*

4) Название запроса: *Увеличение стоимости услуг на 20%.*

Источник данных: таблица *Услуги.*

Макет бланка запроса:



Примечание В запросе на обновление записей добавляется новая строка Обновление, в которую записывается выражение, согласно которому будет производиться обновление. В строку Поле помещаются только те поля таблицы, обновление которых будет производиться.

Результат выполнения запроса:

Услуги			
Код услуги	**Наименование услуги**	**Стоимость**	**Количество дней**
1	Ремонт холодильника	4200	4
2	Установка спутникового телевидения	7200	2
3	Сборка компьютера	1800	1
4	Ремонт ресивера	3000	7
5	Подключение телефона	9000	2
6	Подключение компьютера	2400	2
7	Диагностика компьютера	600	5
8	Монтаж сети	9600	3
9	Подключение Интернет	1920	1

3. Сохранить базу данных.

САМОСТОЯТЕЛЬНАЯ РАБОТА №4

Цель работы:

1. Изучение приемов создания отчетов, разработка кнопочных форм.

1 Создание отчетов, кнопочных форм

Задание 1 – Создание отчетов.

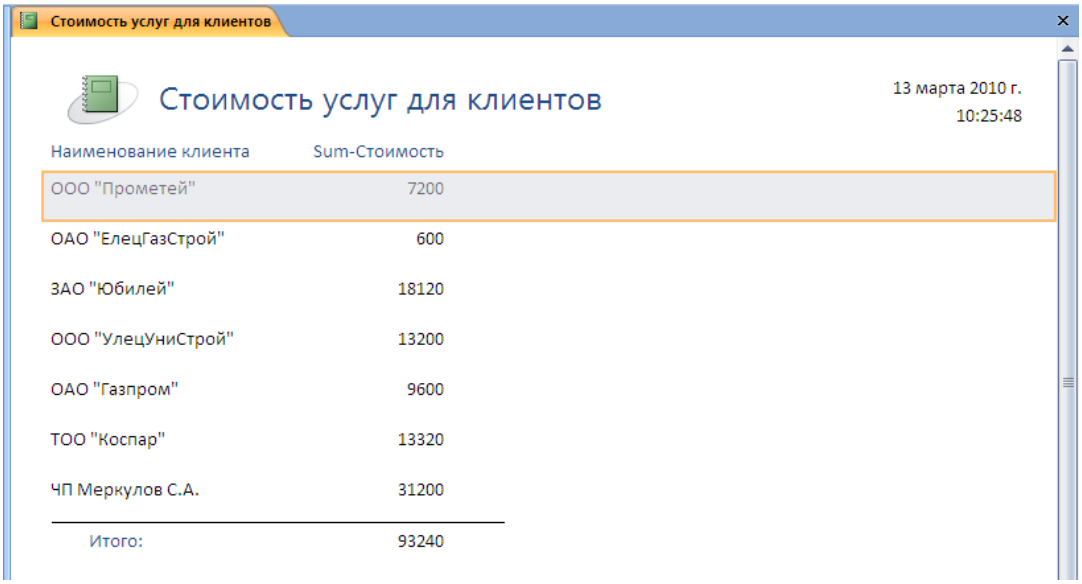
1. Открыть из папки **Работы по Access** базу данных Центр-Сервис. Перейдите на вкладку

Создание ленты инструментов Отчеты.

2. Создать отчет Стоимость услуг для клиентов.

Примечания. Выбрать в качестве источника реализованный запрос, на основе которого строится отчет. Использовать тип Автоотчет: ленточный

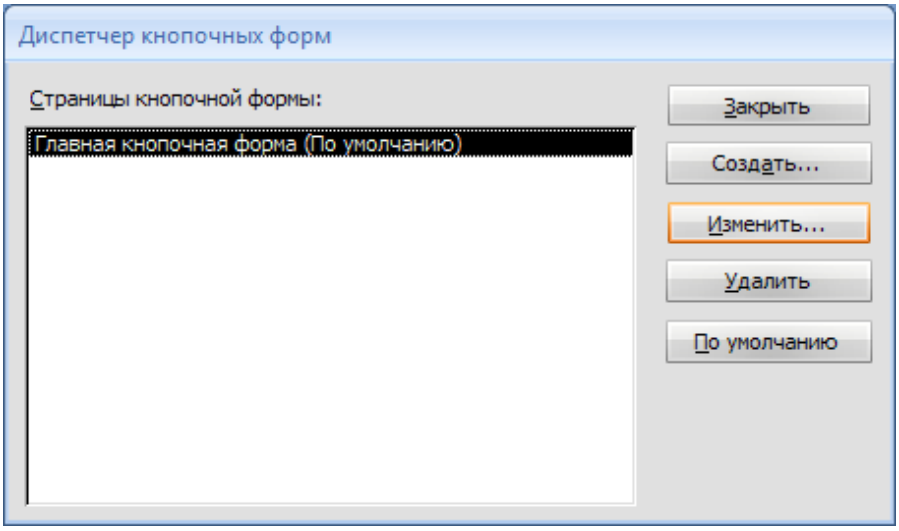
3. Дополнить отчет в режиме Конструктора итоговыми данными.



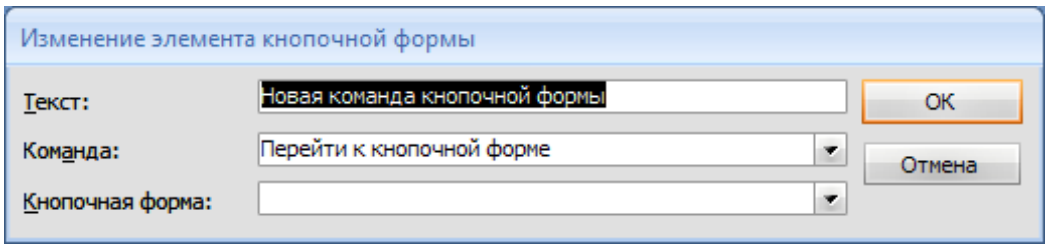
4. Создать отчеты: *Список всех заказов, Список всех клиентов, Список невыполненных заказов, Список неоплаченных заказов, Заказы на определенный день, Заказы по клиентам, Услуги*.

Задание 2 – Создание кнопочных форм

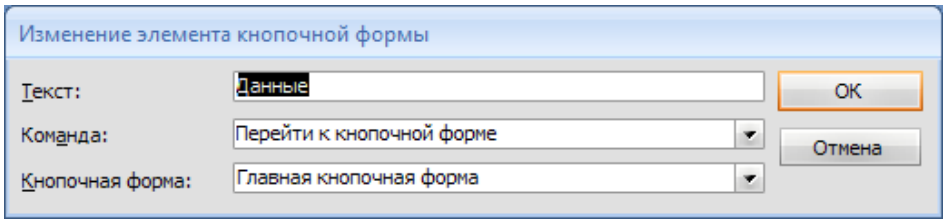
1. Создать кнопочную форму с помощью *Диспетчера кнопочных форм*.



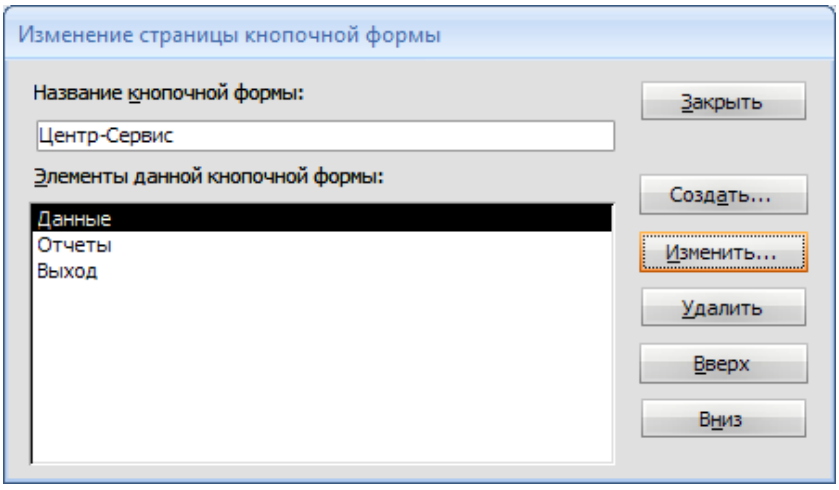
Примечания. Для создания кнопочной формы необходимо выполнить команду ***Диспетчер кнопочных форм*** *вкладки* ***Работа с базами данных*** *ленты инструментов. После открытия окна Диспетчера кнопочных форм следует щелкнуть по кнопке* ***Изменить****. В открывшемся диалоговом окне, после щелчка по кнопке* ***Создать****, появляется диалоговая страница Изменение элемента кнопочной формы.*



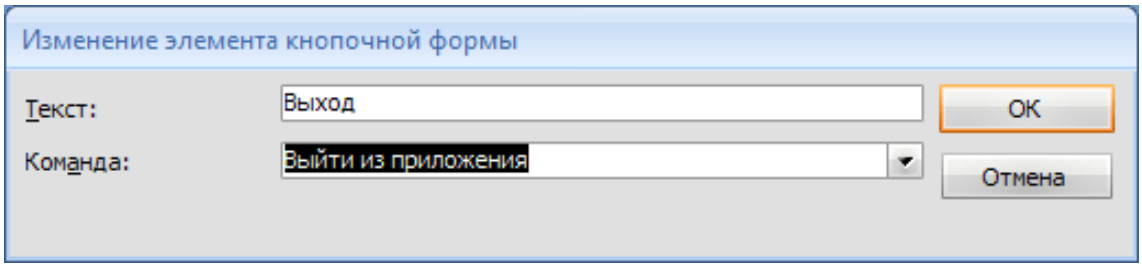
Изменить соответствующие поля на необходимые значения, причем, следует учитывать, что поле Команда и поле Кнопочная форма следует заполнять данными из списка.



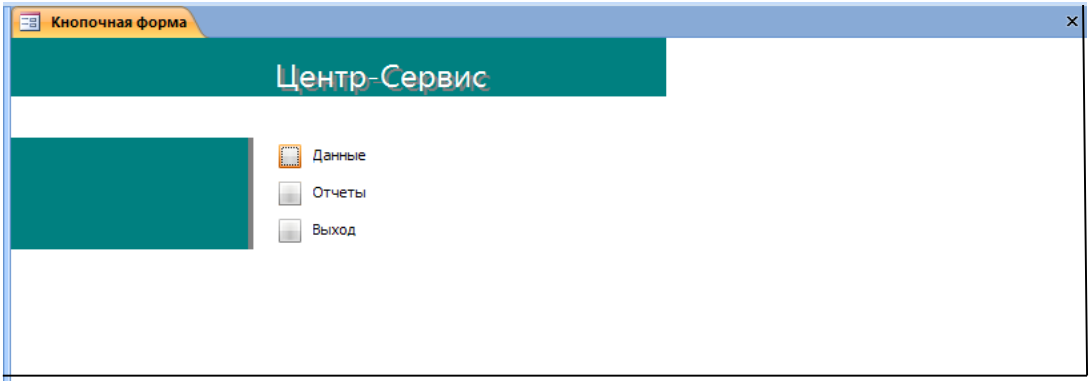
Щелкните кнопку ОК. Аналогично создайте еще элемент кнопочной формы (Отчеты).



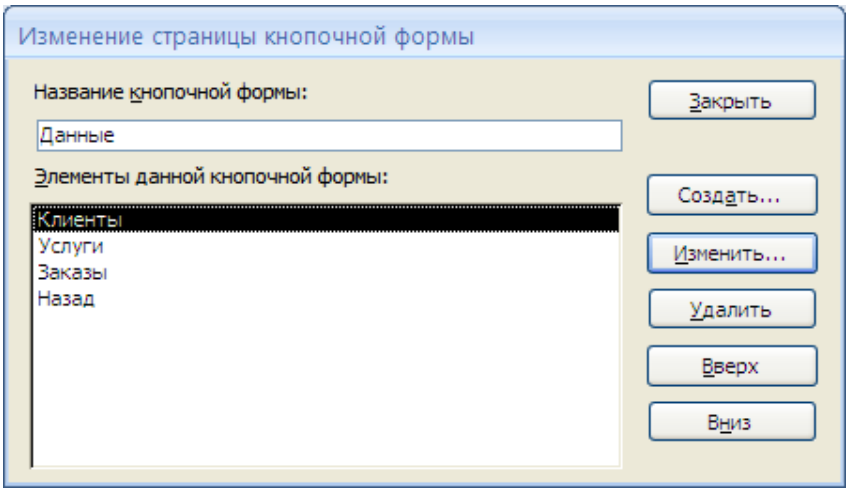
Добавить кнопку закрытия базы данных. Для этого необходимо в поле Команда выбрать Выход из приложения.

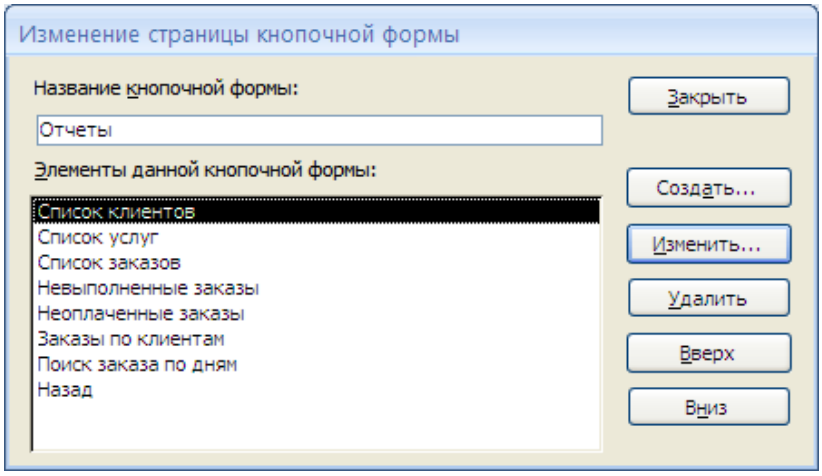


После внесения изменений должна получиться кнопочная форма:

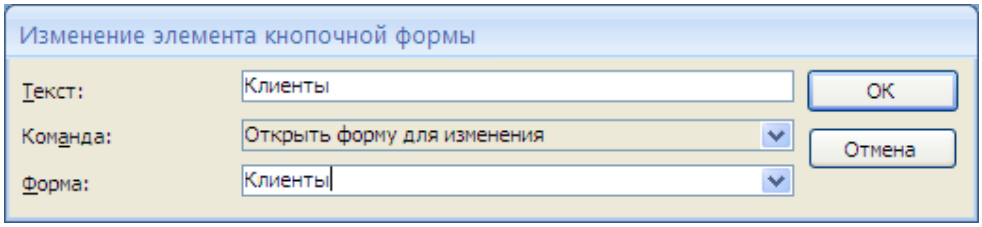


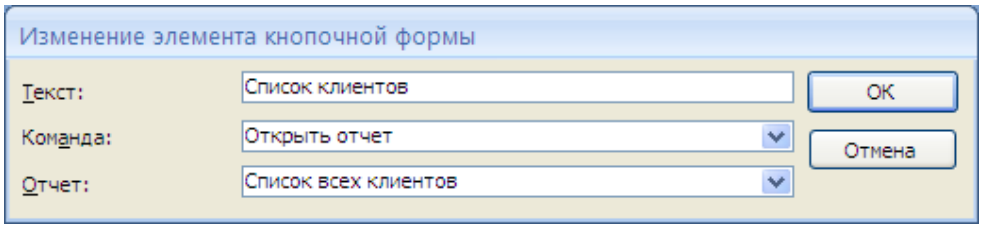
2. Создать подчиненные кнопочные формы.



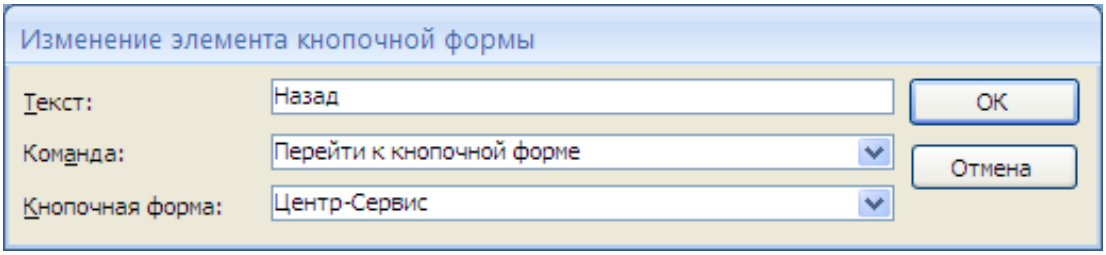


Примечание. Процедура создания подчиненных кнопочных форм аналогично созданию главной кнопочной формы. Отличие заключается в том, что в поле Команда выбирается значение Открыть форму для изменения или Открыть отчет, в зависимости от типа подчиненной кнопочной формы.





Для перехода на более высокий уровень, необходимо на подчиненной кнопочной форме создать кнопку *Назад*, используя команду *Перейти к кнопочной форме* и выбрав в поле *Кнопочная форма* главную кнопочную форму:



3. Закрыть *Диспетчер кнопочных форм*. Проверить правильность проектирования кнопочных форм.
4. Сохранить базу данных.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Для чего используются отчеты?
2. Перечислите способы создания отчетов?
3. Что такое кнопочное меню?
4. Как можно создать кнопочное меню?

САМОСТОЯТЕЛЬНАЯ РАБОТА №5
Основы проектирования структуры баз данных

Цель работы:

1. Овладеть основными теоретическими знаниями и практическими приемами проектирования схемы реляционных отношений базы данных.

1 Теоретический базис

1.1 Общие понятия

База данных – структурированная совокупность данных.

Система баз данных - это компьютеризированная система хранения баз данных, основная цель, которой содержать информацию и предоставлять её по требованию.

Основное назначение СУБД – обеспечить пользователя инструментом, позволяющим оперировать данными в терминах, не связанных с особенностями их хранения в ЭВМ.

Важнейшим понятием, используемым при проектировании любых информационных систем, является понятие «абстрагирование». *Абстрагирование* – это отбрасывание лишних элементов с выделением основных. Существует несколько уровней абстракции в структурных данных, а именно: функциональный, логический, физический.

С процессом проектирования структуры базы данных связывают следующие уровни абстракции:

- Внешняя модель или уровень представления (описание в терминах пользователей БД);
- Логический или концептуальный уровень (обобщенное описание предметной области, разрабатывается прикладными программистами);
- Внутренний или физический уровень (описание концептуальной модели на языке некоторой СУБД).

1.2 Основные понятия реляционной модели

В соответствии с реляционной моделью, БД является совокупностью отношений.

Отношение – это некоторое подмножество прямого произведения. В качестве альтернативного определения применимо: отношение – это множество кортежей. Размер кортежа называют арностью отношения.

На практике рассматривается только конечное множество кортежей. Естественно конечное множество кортежей записывается в виде таблиц, т.е. таблица – это отношение.

Столбцу таблицы соответствует некоторый атрибут (объекта). Значение этого атрибута выбирается из некоторого множества, которое называется *доменом*. Строку таблицы образует некоторое упорядоченное множество значений всех атрибутов таблицы, взятых из своих доменов.

Альтернативное название атрибутов – *поле.* На физическом уровне строка таблицы (некоторый кортеж) называется *хранимой записью*. Хранимая запись, состоит из хранимых полей (значение атрибутов, взятых из своих доменов). Совокупность хранимых записей (таблица) называется *хранимый файл* (хранимая БД).

Атрибуты или множество атрибутов значения, которых уникальным образом идентифицируют экземпляр объекта, называются *первичным ключом***,** т.к. все экземпляры объекта должны быть различны, то каждый объект должен иметь ключ.

Потенциальный ключ – это обобщение понятия первичного ключа. Потенциальные ключи также, как и первичный обладают свойством уникальной идентификации кортежа в отношении, но если первичный ключ в отношении должен быть выбран только один, то потенциальных ключей может быть несколько (первичный ключ выбирается из потенциальных).

Внешний ключ – это множество атрибутов объекта; каждому значению внешнего ключа соответствует значение потенциального ключа. Внешние ключи используются для связывания кортежей в реляционных базах данных.

1.3 Общий подход к ER-моделированию

Для того, чтобы представить, как устроена предметная область нужно задать множество объектов реального мира (главная проблема что считать объектом). Объект – семантическое понятие, которое может быть полезно при обсуждении устройств реального мира. Сущность реального мира – объекты – не обязательно материальны – важно понятие существенно и различимо для других.

Пример: объектами являются: студент, человек, преподаватель, аудитория. Различают *тип* объекта и *экземпляр* объекта

Пример: тип – аудитория; экземпляр – 358.

Объект обладает рядом свойств, которые иногда называют *атрибутами* объекта (набор атрибутов одноименных объектов в различных прикладных задачах может разниться).

Пример: человек обладает атрибутами фамилия, имя, отчество, дата рождения.

Для определения нового объекта иногда используют понятие *подтипа* (пилот – есть подтип сотрудник, обладает всеми свойствами сотрудника и, кроме того, свойствами: список разрешенных самолетов).

Между объектами могут возникать связи трех видов:

- один к одному 1:1 (пациент: место в палате);
- один к многим 1:n и многие к одному n:1;
- многие ко многим n:n (пациент : хирург).

При построении моделей используются следующие геометрические фигуры:

Объект -  Связь-  Атрибут - 

В настоящее время существует большое множество прикладных программ для создания графического представления структуры БД. При этом могут быть использованы как специализированные средства (например, Visio), так и средства построения графических образов (Umbrello, OO Draw).

2 Основы работы с MS Access

2.1 Создание новой БД в среде MS Access

MS Access является частью пакета MS Office и представляет собой СУБД. Запустите MS Access, используя команды меню «Пуск» -> «Программы» ->

«Microsoft Office» -> «Access». Перед вами откроется основное окно программы. Выполните пункты меню «Файл» - «Создать», после чего из вариантов выберите «Новая база данных» и сохраните вашу БД на диск.

Открывшееся окно представляет собой новую БД.

2.2 Объекты БД в среде MS Access

MS Access поддерживает следующие объекты:

- Таблица – отношение в реляционной терминологии;
- Запрос – сохраненный текст (модель) запроса на языке SQL к таблицам БД;
- Формы – конструирование интерфейса пользователя;
- Отчеты – конструирование выходной информации БД;
- Страницы – страницы доступа через web;
- Макросы – группы макрокоманд;
- Модули – программные модули на языке VBA, которые могут быть использованы в запросах, формах, отчетах, макросах.

2.3 Создание таблиц в среде MS Access

СУБД MS Access поддерживает различные варианты создания таблиц. Рассмотрим наиболее часто используемый вариант создания – в режиме конструктора. Для этого среди объектов БД выберите «Таблицы», после чего выполните пункт «Создание таблицы в режиме конструктора».

В появившемся окне предлагается ввести имена атрибутов таблицы с указанием их типов. Типы атрибутов выбираются из выпадающего списка вариантов, а конкретные характеристики указываются на панели внизу (например, для текстового формата можно указать максимальный размер).

Третий столбец на форме конструирования отношений – описание атрибутов. При создании сложных БД с большим числом отношений и атрибутов в них обязательно описывайте логику, которую вы закладываете в тот или иной атрибут.

После создания всех атрибутов не забудьте указать ключевое поле вашего отношения. Для этого выделите нужные поля и нажмите на иконку «ключевое поле» изображающую ключ на панели конструктора.

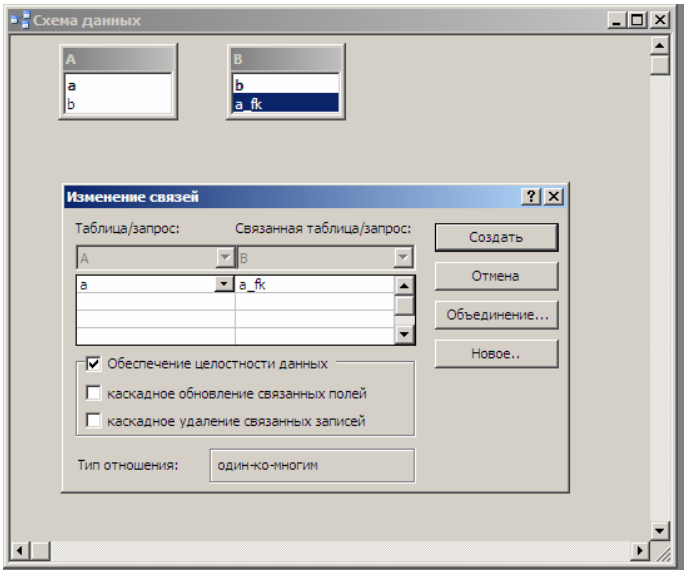
2.4 Создание схем данных в среде MS Access

СУБД MS Access реализует инструмент проектирования связей между таблицами (объектами) БД. Этот инструмент называется «Схема данных».

Для вызова этого инструмента можете воспользоваться соответствующей кнопкой на панели «База данных» (рис. 1.1)

Панель «База данных»

В рабочую область схемы можно добавлять и удалять таблицы, а также связи между ними. Для добавления связи между таблицами выберите необходимое поле (внешний ключ) и используя Drag'n'Drop протяните его до нужного поля связанной таблицы (потенциальный ключ). После чего, в появившемся окне свойств вновь созданной связи, можете выставить необходимые настройки (рис. 1.2).



Редактирование связей в MS Access

После окончания редактирования не забудьте сохранить схему.

3 Задание

1. Выбор предметной области, для которой будет проектироваться БД (например: склад, больница, аптека, аэропорт и т.д.) ;
2. Определение множества конечных пользователей, выбранной предметной области, которыми предполагается использование разрабатываемой БД;
3. Составление от лица выбранных пользователей (**в терминах предметной области!**) описания требований к разрабатываемой базе данных, в том числе включающих описания объектов предметной области и принципов их взаимодействия;
4. Выделение общих особенностей на базе описаний конкретных пользователей и отбрасывание лишних деталей (абстрагирование);
5. На базе абстрактного описания формирование логической схемы БД в следующих видах: модель «Сущность-Связь» (ER-модель) и инфологическая (или Даталогическая) модель данных;
6. Подготовка отчета с представлением результатов выполнения поставленных задач.

3.1 Темы баз данных

Выбор темы осуществляется в соответствии со списком группы.

1. Автосалон;
2. Видеопрокат;
3. Гостиница;
4. Кинотеатр;
5. Поликлиника;
6. Библиотека;
7. Фитнес-клуб;
8. Автошкола;
9. Химчистка;

10. Ресторан;
11. Аптека;
12. Турфирма;
13. Интернет-магазин;
14. Доставка пиццы;
15. Книжный магазин;
16. Компьютерные игры;
17. Автовокзал.

Комментарии:

Выполнить работу как показано в примере 1 и 2. При выборе предметной области рекомендуется выбирать предметную область, которая максимально знакома разработчику БД.

Результаты выполнения данной работы настоятельно рекомендуется использовать при выполнении следующих лабораторных работ, а также при реализации курсовой работы (проекта).

Таким образом, целесообразно выбрать предметную область один раз, чтобы планомерно в течение всего семестра использовать полученные на лабораторных работах знания и результаты в оформлении отчета по курсовой работе.

Проектирование инфологической диаграммы и ER-модели рекомендуется осуществлять с использованием специализированного программного инструментария (например, MS Visio).

Описание предметной области с точки зрения конечных пользователей системы не должно содержать специализированных терминов и жаргонизмов (кортежи, таблицы, сущности, функционал программы и т.п.), т.е. конечные пользователи формулируют для Вас, как для разработчика БД, требования, но исключительно в терминах своей предметной области.

На ER-диаграмме не забудьте указать не только связи между сущностями, но также их вид. Число сущностей может быть произвольным и зависит от уровня детализации выбранной предметной области, предъявляемых конечными пользователями требований, а также (безусловно) от ваших знаний в данной предметной области.

Физическая реализация разработанной схемы отношений не требуется.

3.2 Критерии оценивания (требования):

1. Оценивается только работа студента, оформленная в виде печатного отчета;
2. Работа может быть оценена *неудовлетворительно* (0 баллов) в случае, если студент не показывает знаний по сути сделанной работы, т.е. затрудняется ответить на вопросы, ответы на которые в явном или неявном виде присутствуют в его отчете;
3. Работа может быть оценена *удовлетворительно* (1-2 балла) в случае, если решены все поставленные перед студентом задачи, однако студент затрудняется ответить на дополнительные вопросы по материалам лекций, относящимся к выполняемой лабораторной работе, и/или имеются ошибки в составленной логической схеме и другие недостатки в оформлении работы;
4. Работа может быть оценена *хорошо* (3-4 балла) в случае, если решены все поставленные перед студентом задачи и студент правильно отвечает на дополнительные вопросы по материалам лекций, относящимся к выполняемой лабораторной работе, однако имеются ошибки в составленной логической схеме и/или другие недостатки в оформлении работы;
5. Работа может быть оценена *отлично* (5 баллов) в случае, если решены все поставленные перед студентом задачи, студент правильно отвечает на дополнительные вопросы по материалам лекций, относящимся к выполняемой лабораторной работе, и отсутствуют ошибки в составленной логической схеме.

ПРИМЕР 1

В качестве предметной области выполнения лабораторной работы выбирается – деканат. С точки зрения проектирования базы данных в данной предметной области имеются следующие конечные пользователи: декан факультета, заместители декана факультета, ведущий документовед. Опишем предметную область с точки зрения конечных пользователей.

Ведущий документовед:

В своей работе мне требуется постоянно иметь возможности просмотра и управления информацией о студенте как личной (фамилия, имя, отчество, паспортные данные, адреса, контакты и

т.д.), так и учебной (учебный статус: учится, представлен к отчислению, отчислен, в академическом отпуске, номер семестра обучения, год поступления, направление подготовки, текущая успеваемость и т.д.). Работа с контингентом студентов также предполагает наличие возможности объединения их в группы для добавления в приказы (на перевод, отчисление, стипендию и т.д.) с последующим выводом на печать. Важным аспектом работы документоведа является работа с текущей успеваемостью студентов – требуется обеспечить возможность внесения и модификации текущей успеваемости и информации о пропусках занятий, а также итоговой успеваемости студентов по различным изучаемым дисциплинам.

Заместитель декана:

В своей работе мне требуется оперативно получать контактную информацию о студентах (телефоны, адрес, icq, электронная почта), о учебных группах (составе, старостах и т.д.), о преподаваемых дисциплинах в каждой группе, а также о успеваемости отдельных студентов в случае поступления запроса направления на пересдачу. Требуется обеспечить возможность регистрации продления учебной сессии с регистрацией причины продления. Необходимо иметь возможность просмотра/поиска информации о преподавателях, закрепленных за различными группами, а также возможность регистрации информации о кураторах групп.

Важным аспектом является возможность регистрации различной информации о социальной и спортивной активности студента, его спортивных разрядах, участия в различных организациях, мероприятиях и т.д.

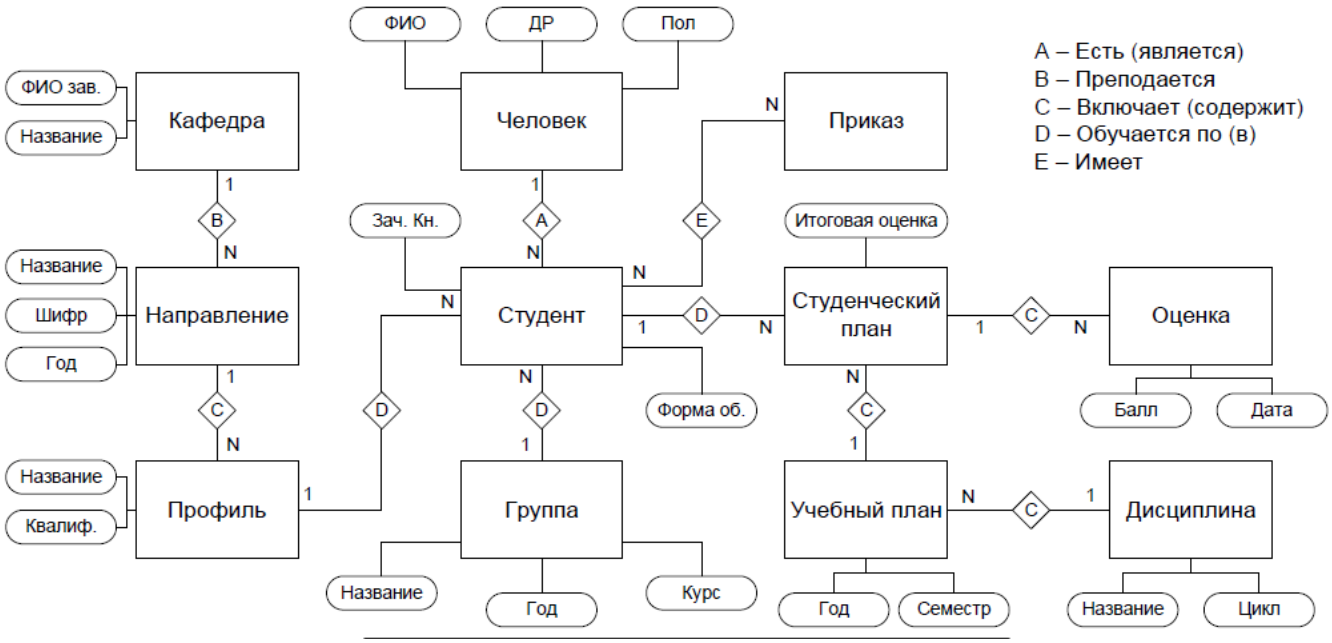
Декан:

Мне требуется доступ ко всем уже заявленным моими сотрудниками возможностями, а также важным аспектом является возможность мониторинга общих тенденций изменения ситуации по успеваемости. Требуется поддерживать возможность определения общего рейтинга студентов с учетом его учебных, спортивных и социальных достижений.

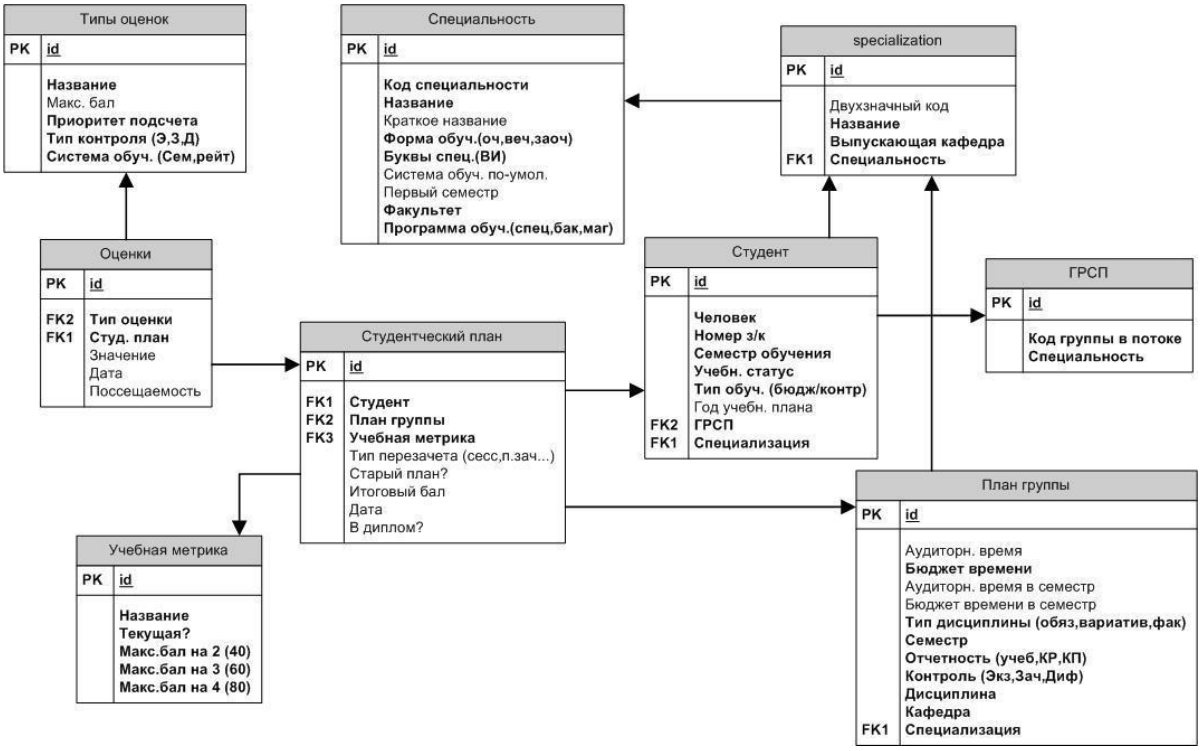
На базе представленных описаний можно выделить следующие базовые объекты предметной области: Студент, Человек, Преподаватель, Учебный план, Группа, Специализация (Профиль), Специальность (Направление подготовки), Учебный план группы, Студенческий план, Система оценивания, Оценка.

Ниже представлены примеры (фрагменты) ER–диаграммы и даталогической модели.

<u>ВНИМАНИЕ</u>! На ER-диаграмме представлена лишь некоторая абстракция конкретной предметной области с прикладной точки зрения, т.е. обусловленная решаемой задачей. Кроме того, в целях экономии места на конкретных примерах ниже представлены лишь некоторые атрибуты сущностей предметной области. В реальной лабораторной работе рекомендуется представить, как можно больше атрибутов характеризующие объекты предметной области.



ER-диаграмма



Даталогическая схема

ПРИМЕР 2

1. Выбор предметной области

В качестве предметной области выбраны «Школа».

2. Описание предметной области

База данных создаётся для информационного обслуживания администрации (директор, зам. Директора), преподавателей, родителей учащихся. БД должна содержать данные о классах, учебных предметах, учителях, учениках и успеваемости, а также должна предоставлять возможность получать разнообразные отчёты.

БД должна информировать пользователей:

1. Об ученическом составе классов;
2. О преподавательском составе школы;
3. О распределении учебной нагрузки преподавателей и классного руководства;
4. Об успеваемости учеников.

Определим границы информационной поддержки пользователей:

3. Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление в архив);
- обеспечение логической непротиворечивости БД;
- обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

4. Готовые запросы:

- получение списка учеников по классам;
- получение списка всех четвертных или годовых оценок;
- получение сведений об успеваемости учеников по конкретному предмету;
- получение сведений об отличниках;
- получение информации о преподавателях;
- получение списка преподавателей, ведущих определенный предмет;
- получение списка классных руководителей и др.

Должность	Требования к БД
Администрация	Работа со сведениями о классах и преподавателях: добавление, редактирование и удаление данных; просмотр успеваемости

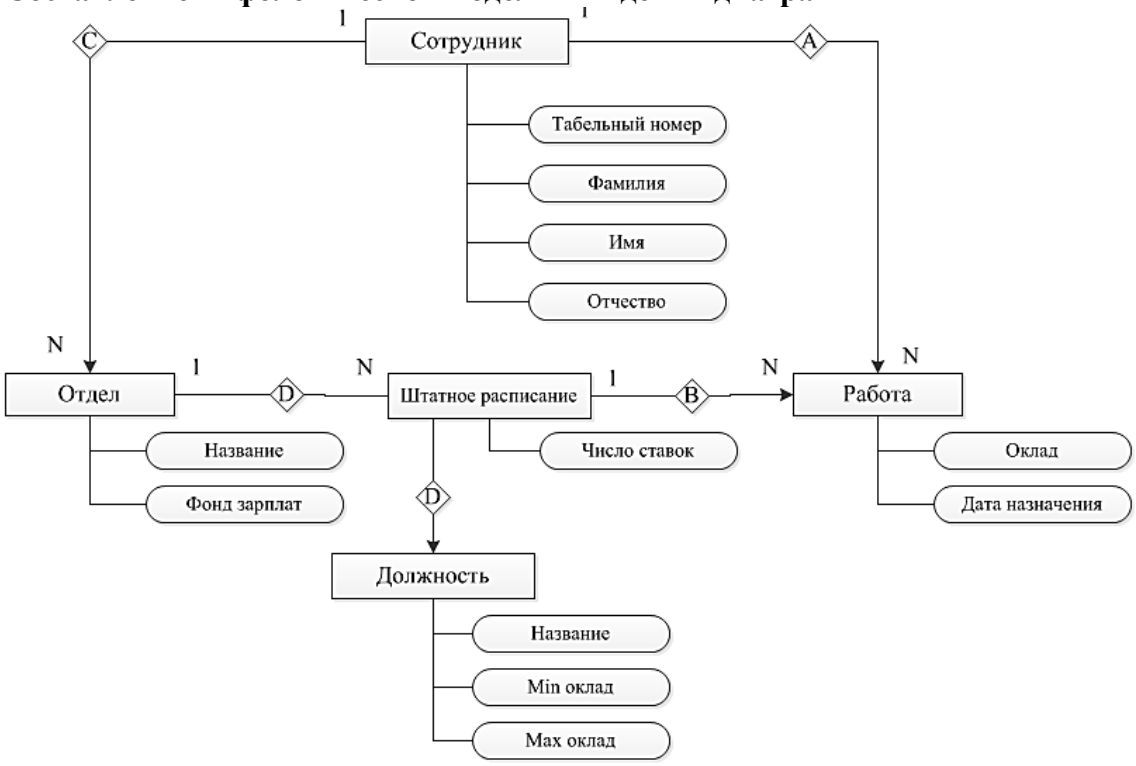
	учеников: четвертные и годовые оценки; просмотр портфолио учителей; просмотр нагрузки учителей школы
Преподаватели	Просмотр успеваемости учеников: четвертные и годовые оценки; просмотр портфолио учителей; редактирование успеваемости учеников по предмету; просмотр нагрузки учителей школы
Родители	Просмотр успеваемости своего ребенка: четвертные и годовые оценки; просмотр портфолио учителя

5. Выделение объектов и их атрибутов

Классы	**Учебные предметы**	**Учителя**
Номер	Название	Фамилия
Классный руководитель		Имя
Число учащихся		Отчество
		Дата рождения
		Адрес
		Вуз
		Год окончания вуза

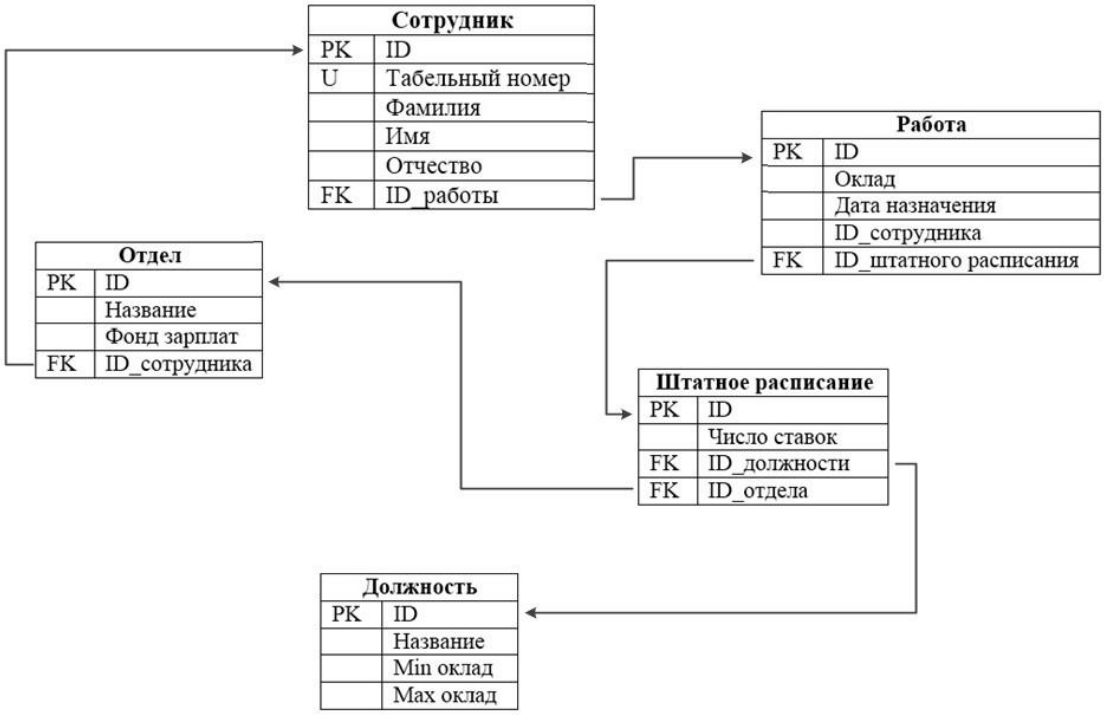
Ученики	**Успеваемость**
Класс	Предмет
Фамилия	Четверные оценки
Имя	Годовая оценка
Отчество	
Пол	
Дата рождения	

6. Составление инфологической модели в виде ER-диаграммы



A – Выполняет; B – Содержит; C – Принадлежит; D – Соответствует.

7. Составление даталогической модели



КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Что такое база данных? Что такое система баз данных?
2. Что такое система управления базами данных? Основное назначение?
3. Основные компоненты СУБД?
4. Что подразумевает понятие абстрагирование в СУБД?
5. Какие существуют уровни абстракции в структурных данных?
6. Опишите уровень представления
7. Опишите концептуальный уровень
8. Опишите физический уровень
9. Виды связей
10. Что такое отношение (таблица) в реляционной модели СУБД?
11. Что такое домен, атрибут (поле), картеж (хранимая запись) в реляционной модели СУБД?
12. Что такое первичный ключ?
13. Что такое потенциальный ключ?
14. Что такое внешний ключ?

САМОСТОЯТЕЛЬНАЯ РАБОТА №6
Изменение данных с использованием операторов языка SQL

Цель работы:

1. Изучить операторы языка SQL для изменения данных.

1 Теоретический базис

SQL — это язык программирования, предназначенный для работы с наборами фактов и отношениями между ними. В программах управления реляционными базами данных, таких как Access, язык SQL используется для работы с данными. Как и многие языки программирования, SQL является международным стандартом, признанным такими комитетами по стандартизации, как ISO и ANSI.

На языке SQL описываются наборы данных, помогающие получить ответы на вопросы. При использовании SQL необходимо применять правильный синтаксис. Синтаксис — это набор правил, позволяющих правильно сочетать элементы языка. Синтаксис SQL основан на синтаксисе английского языка и имеет много общих элементов с синтаксисом языка Visual Basic для приложений (VBA).

Запрос SQL – это запрос Access, который нельзя создать с помощью конструктора или мастера. Эти запросы могут быть созданы только с помощью языка SQL. К таким запросам относятся подчиненные запросы, запросы к серверу, запросы – объединения и управляющие запросы.

SQL состоит из инструкций, которые передаются программе, управляющей работой реляционной базы данных, предлагая ей выполнить определенные действия. Эти инструкции иногда называют предложениями.

Для создания запроса SQL надо выбрать в окне **База данных** вкладку **Запросы**, а затем выбрать «***Создание запроса в режиме конструктора***».

Окно **Добавление таблицы** закрыть. На панели инструментов конструктора запросов щелкнуть по кнопке **SQL** или выбрать команду **Запрос – Запрос SQL**. Далее выбирается вид запроса: объединение, к серверу, управление. На экран будет выведено окно для ввода инструкций языка SQL. После написания инструкций запрос выполняется (кнопка «Запуск») и сохраняется.

В запросе SQL можно осуществлять выборку данных из таблиц по различным критериям (например, по классу). Но запрос – выборку нетрудно создать и в режиме ***Конструктора запросов***, которым можно воспользоваться без знания инструкций SQL.

1.1 Создание таблицы: команда CREATE TABLE

1.2 Удаление таблиц

Уничтожить старую таблицу куда проще, чем создать новую. Введите простую команду:

Команда DROF TABLE **удаляет таблицу со всей данными**!

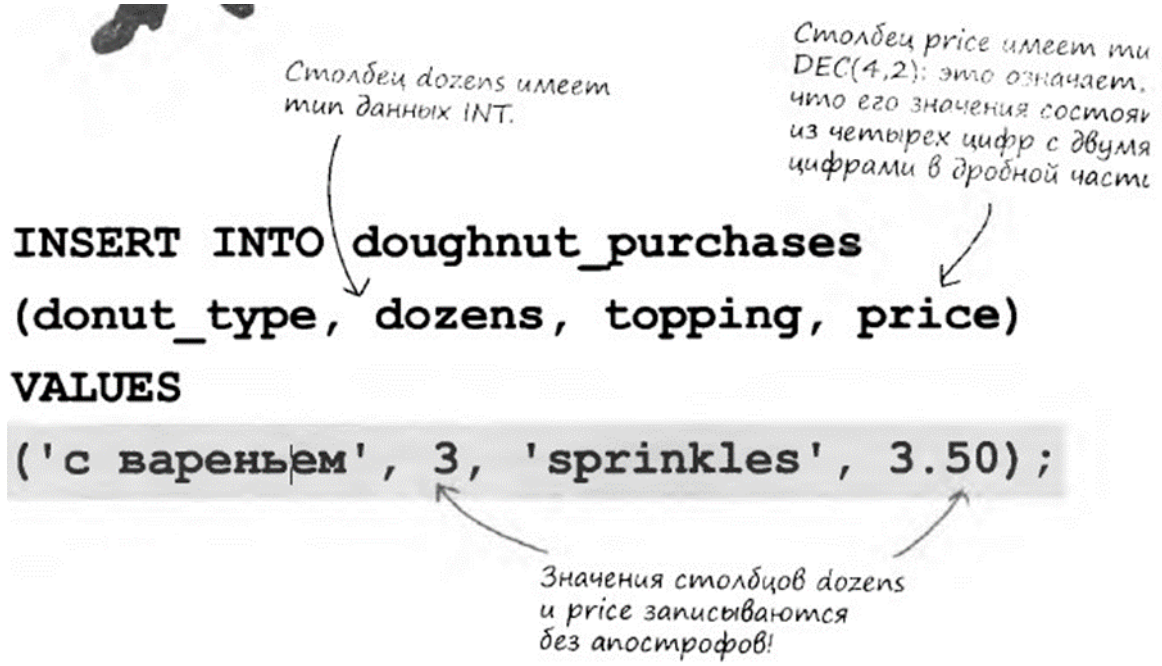
1.3 Заполнение таблицы

Для добавления данных в таблицу используется команда **INSERT.**

На приведенной ниже схеме показано, что делает каждая из частей команды. Значения во второй группе скобок должны следовать в том ж е порядке, что и имена столбцов. Ниже приведена не реальная команда, а «заготовка» — условный шаблон, демонстрирующий формат команды **INSERT.**

Создание команды INSERT

Порядок перечисления значений должен **ТОЧНО** совпадать с порядком перечисленных столбцов!



Ниже приведена команда **INSERT** для таблицы с данными о покупке пончиков. Обратите внимание: числовые значения столбцов **dozens** и **price** записываются без апострофов.

1.4 Правила первичных ключей

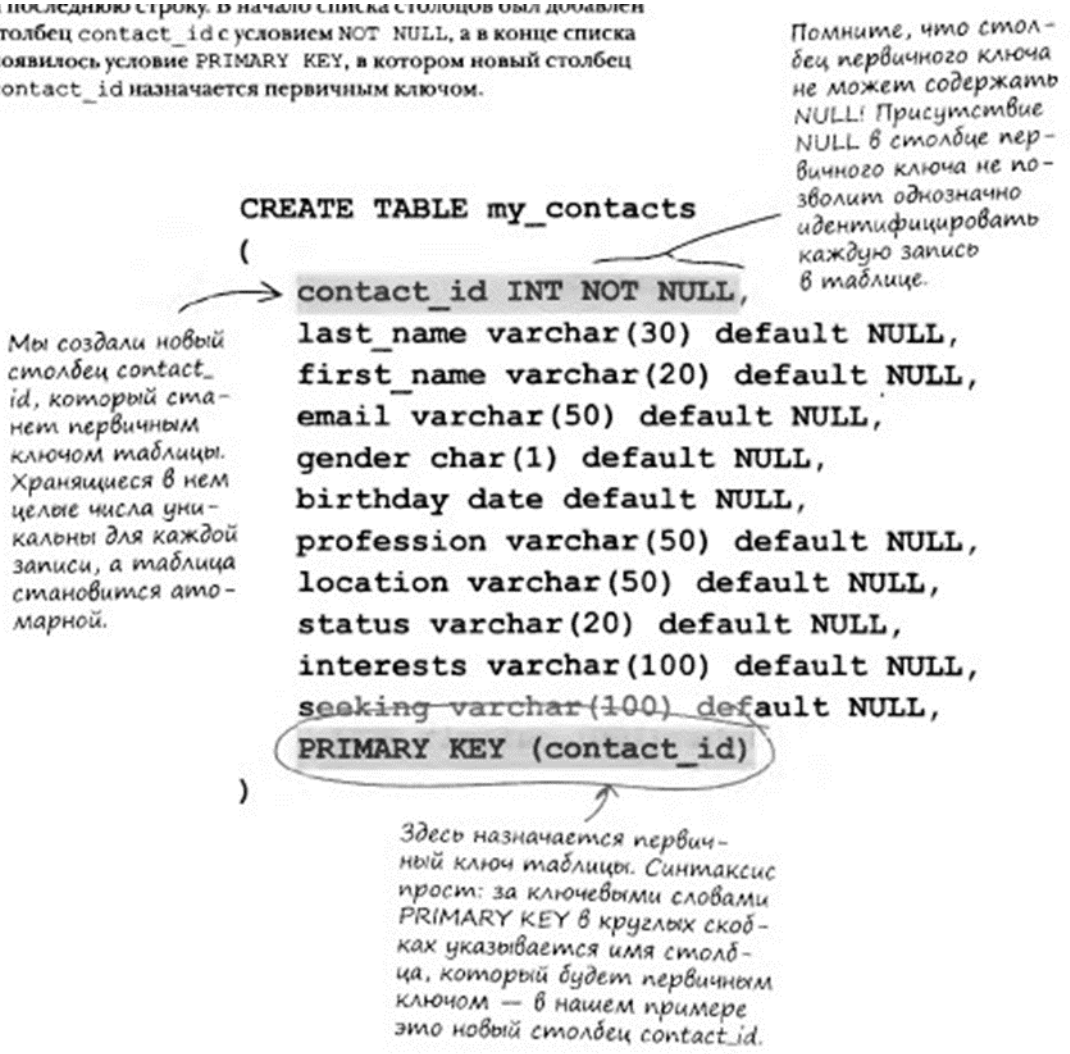
Столбец таблицы, который станет ее первичным ключом, назначается при создании таблицы.

Первичный ключ – столбец таблицы, имеющий. уникальное значение для каждой записи.

Команда **CREATE TABLE** с назначением первичного ключа.

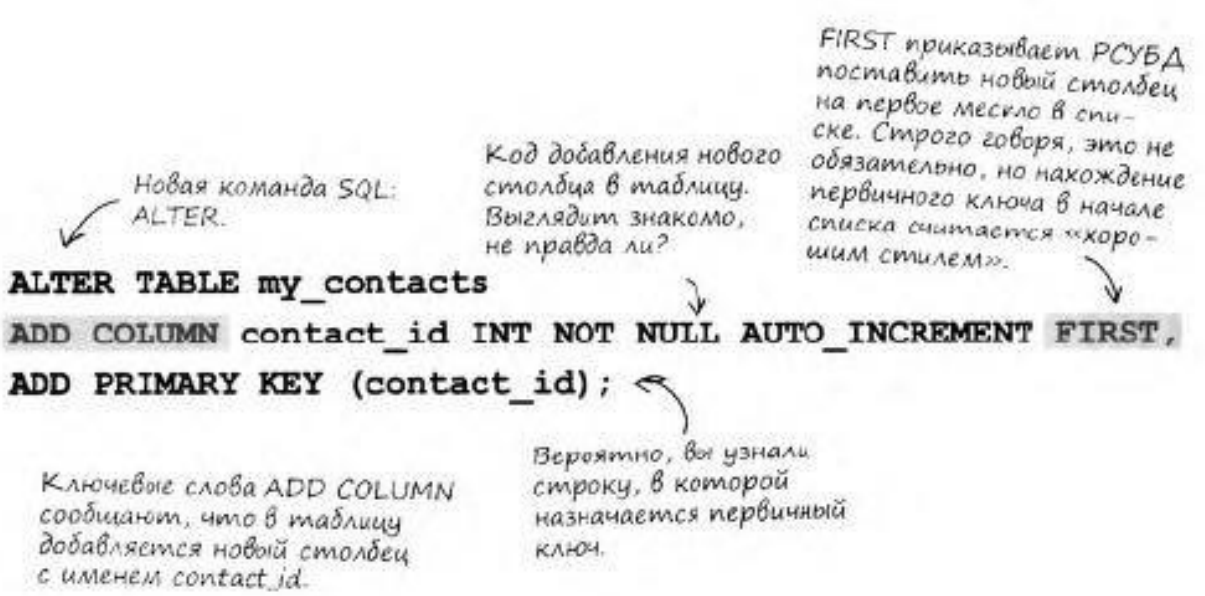
Перед вами код, полученный при выполнении команды **SHOW CREATE TABLE.** Мы удалили из него обратные апострофы и последнюю строку. В начало списка столбцов был добавлен столбец **contact_id** с условием **NOT NULL,** а в конце списка появилось условие **PRIMARY KEY,** в котором новый столбец **contact_id** назначается первичным ключом.

и последнюю строку. В начало списка столбцов был добавлен столбец `contact_id` с условием `NOT NULL`, а в конце списка появилось условие `PRIMARY KEY`, в котором новый столбец `contact_id` назначается первичным ключом.



Добавление первичного ключа в существующую таблицу

Перед вами код добавления первичного ключа AUTO INCREMENT в таблицу my_contacts.



Простейшая инструкция поиска информации имеет вид: SELECT <Список полей>
FROM <Имена таблиц>;

Select – ключевое слово, оно сообщает базе данных, что инструкция SQL является запросом – выборкой.

<Список полей> – перечисление полей через запятую, которые надо вывести в новую таблицу.

Символ * – на этом месте обозначает, что будут выбраны все поля указанных в предложении FROM таблиц.

From – всегда присутствует после слова Select и определяет, какие таблицы или запросы содержат поля, приведенные в инструкции Select.

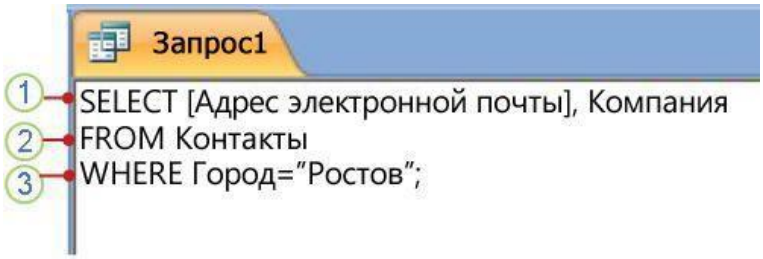
1.5 Инструкции SELECT

Инструкция SELECT служит для описания набора данных на языке SQL. Она содержит полное описание набора данных, которые необходимо получить из базы данных, включая следующее:

- таблицы, в которых содержатся данные;
- связи между данными из разных источников;
- поля или вычисления, на основе которых отбираются данные;
- условия отбора, которым должны соответствовать данные, включаемые в результат запроса;
- необходимость и способ сортировки.

Пример 1:

В приведенном ниже примере показано, как в Access может выглядеть инструкция SQL для простого запроса на выборку.



- Предложение SELECT
- Предложение FROM
- Предложение WHERE

Эту инструкцию SQL следует читать так: «Выбрать данные из полей "Адрес электронной почты“ и "Компания“ таблицы "Контакты“, а именно — те записи, в которых поле "Город“ имеет значение "Сиэтл“».

Разберем пример по предложениям, чтобы понять, как работает синтаксис SQL.

1.5.1 Предложение SELECT

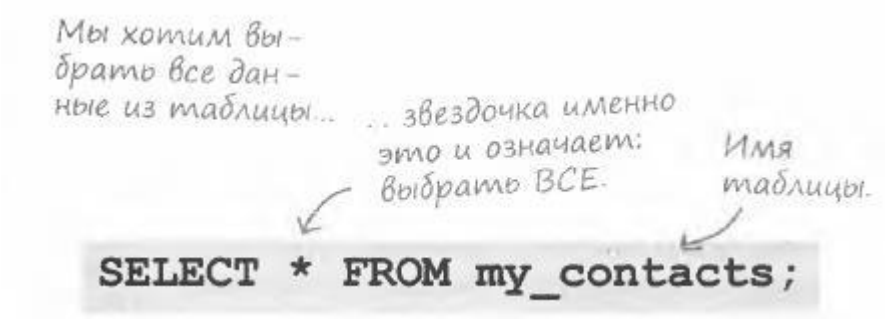
SELECT [E-mail Address], Company

Это предложение SELECT. Оно содержит оператор (SELECT), за которым следуют два идентификатора ([Адрес электронной почты] и Организация).

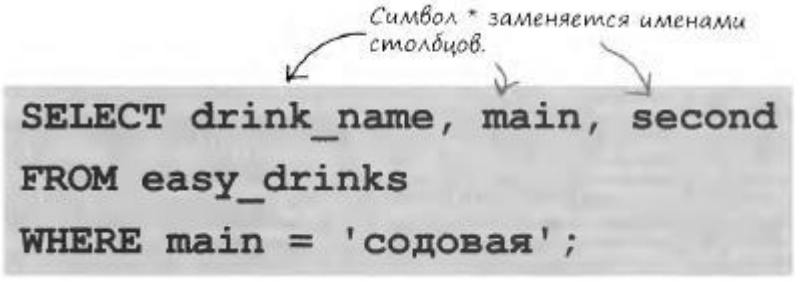
Если идентификатор содержит пробелы или специальные знаки (например, Адрес электронной почты), он должен быть заключен в прямоугольные скобки.

В предложении SELECT не нужно указывать таблицы, в которых содержатся поля, и нельзя задать условия отбора, которым должны соответствовать данные, включаемые в результаты.

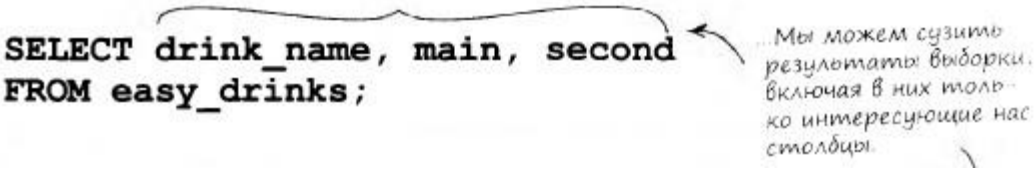
В инструкции SELECT предложение SELECT всегда стоит перед предложением FROM.



1.5.2 Выборка ограниченного набора столбцов



Указывая, какие столбцы должны возвращаться запросом, мы отбираем из полных результатов интересующую нас информацию. По аналогии с тем, как условие WHERE ограничивает количество возвращаемых записей, конструкция отбора столбцов ограничивает количество возвращаемых столбцов. По сути, вы поручаете работу по отбору информации SQL.



1.5.3 Предложение FROM

FROM Contacts

Это предложение FROM. Оно содержит оператор (FROM), за которым следует идентификатор (Контакты).

В предложении FROM не указываются поля для выборки.

1.5.4 Предложение WHERE

WHERE City = "Seattle"

Это предложение WHERE. Оно содержит оператор (WHERE), за которым следует выражение

(Город="Ростов").

ПРИМЕЧАНИЕ: в отличие от предложений SELECT и FROM предложение WHERE является необязательным элементом инструкции SELECT.

С помощью предложений SELECT, FROM и WHERE можно выполнять множество действий. Дополнительные сведения об использовании этих предложений см. в разделах, указанных в конце данной статьи.

1.6 Использование команды DELETE

Взгляните на построенную нами команду DELETE. Она работает именно так, как и следовало ожидать: все записи, соответствующие условию WHERE, удаляются из таблицы.

```
DELETE FROM clown_info
WHERE
activities = 'танцы';
```

1.6.1 Правила DELETE:

- Команда DELETE не позволяет удалить значение одного столбца или группы столбцов.
- Команда DELETE удаляет из таблицы одну или несколько записей (в зависимости от условия WHERE).
- Мы рассмотрели пример удаления одной записи из таблицы. Также возможно удаление сразу нескольких записей. Для этого критерий выбора удаляемых записей определяется при помощи условия WHERE. Синтаксис условия WHERE полностью совпадает с синтаксисом WHERE в команде SELECT (см. главу 2); в нем могут использоваться все конструкции из главы 2, в том числе LIKE, IN, BETWEEN и операторы сравнения.
- Будьте осторожны — следующая команда удаляет из таблицы все записи:

```
DELETE FROM your_table
```

ЗАДАНИЕ

1. Реализовать скрипт создания БД приведенных в примере ниже с использованием инструкций части DDL языка SQL (CREATE TABLE, CREATE INDEX и т.д.);
2. При создании таблиц не забывать указывать необходимые ограничения целостности и значения по умолчанию (DEFAULT, PRIMARY KEY, FOREGIN KEY и т.д.);
3. Заполнить таблицы созданной БД, выполняющие роль справочников, данными, с использованием инструкций части DML языка SQL (INSERT, UPDATE и т.д.);

Примеры создания и заполнения БД с помощью SQL:

1 Создание БД Аэропорт

CREATE TABLE Пассажир (
Паспортные_данные TEXT, Фамилия TEXT,
Имя TEXT,
Отчество TEXT,
№_билета INTEGER,
№_багажа INTEGER, Дата_рождения DATETIME,

CONSTRAINT Пассажир_pkey PRIMARY KEY (Паспортные_данные),
CONSTRAINT Пассажир_№_билета_fk FOREIGN KEY (№_билета) REFERENCESБилет(№_билета)
);
INSERT INTO Пассажир (Паспортные_данные, Фамилия, Имя, Отчество, №_билета, №_багажа, Дата_рождения)
VALUES ("6008153205", "Гребенников", "Владислав", "Викторович", 26254440, 0262, "16.06.1985");
ALTER TABLE Пассажир
ADD CONSTRAINT Пассажир_№_билета_fk FOREIGN KEY (№_билета) REFERENCESБилет(№_билета)
ALTER TABLE Пассажир
ADD CONSTRAINT Пассажир_№_багажа_fk FOREIGN KEY (№_багажа) REFERENCESБагаж(№_багажа)

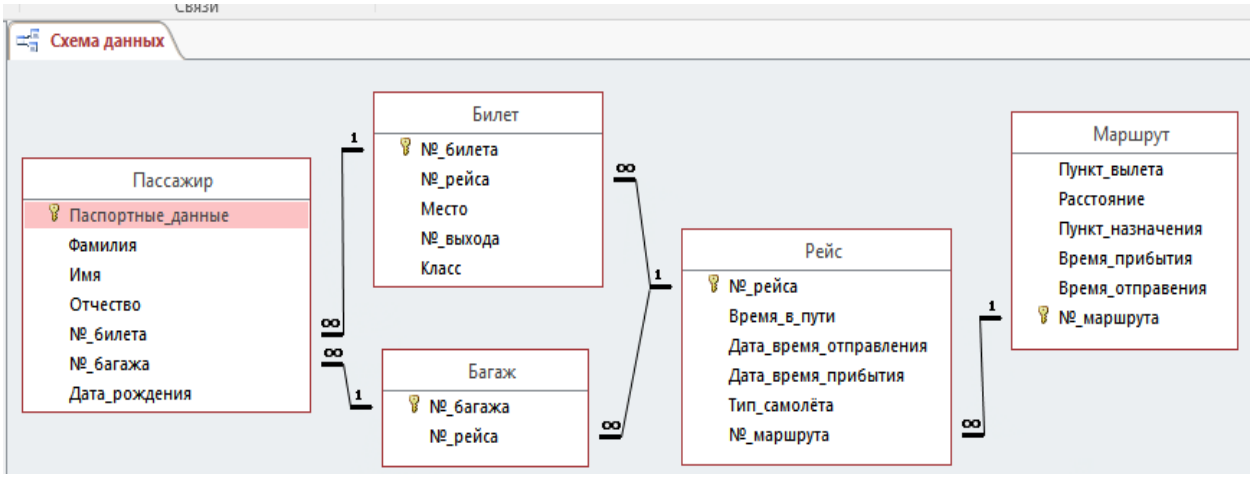
CREATE TABLE Багаж (
№_багажа TEXT,
№_рейса INTEGER,
CONSTRAINT Багаж_pkey PRIMARY KEY (№_багажа),
CONSTRAINT Багаж_№_рейса_fk FOREIGN KEY (№_рейса) REFERENCESРейс(№_рейса)
);
ALTER TABLE Багаж
ADD CONSTRAINT Багаж_№_рейса_fk FOREIGN KEY (№_рейса) REFERENCESРейс(№_рейса)
INSERT INTO Багаж(№_багажа, №_рейса) VALUES ('0262',453,);

CREATE TABLE Маршрут (
Пункт_вылета TEXT, Расстояние INTEGER, Пункт_назначения TEXT, Время_прибытия TIME, Время_отправения TIME, №_маршрута INTEGER,
CONSTRAINT Маршрут_pkey PRIMARY KEY (№_маршрута)
);
INSERT INTO Маршрут (Пункт_вылета, Расстояние, Пункт_назначения, Время_прибытия, Время_отправения, №_маршрута)
VALUES ('Ростов - на - Дону', 1300, 'Санкт - Питербург', '17:55', '15:00', '139452');

CREATE TABLE Рейс (
№_рейса INTEGER, Время_в_пути TIME,
Дата_время_отправления DATETIME, Дата_время_прибытия DATETIME, Тип_самолёта TEXT, №_маршрута INTEGER,
CONSTRAINT Рейс_pkey PRIMARY KEY (№_рейса),
CONSTRAINT Рейс_№_маршрута_fk FOREIGN KEY (№_маршрута) REFERENCES Маршрут(№_маршрута)
);
INSERT INTO Рейс (№_рейса, Время_в_пути, Дата_время_отправления, Дата_время_прибытия, Тип_самолёта, №_маршрута) VALUES (453, '2:55', '30.06.2014', '30.06.2014', 'A319', '139452');

CREATE TABLE Билет (
№_билета INTEGER,
№_рейса INTEGER, Место TEXT,
№_выхода INTEGER, Класс TEXT,
CONSTRAINT Билет_pkey PRIMARY KEY (№_билета),
CONSTRAINT Билет_№_рейса_fk FOREIGN KEY (№_рейса) REFERENCESРейс(№_рейса)
);
INSERT INTO Билет (№_билета, №_рейса, Место, №_выхода, Класс) VALUES (26254440, 453, '14B', 160, 'E');
ALTER TABLE Билет
ADD CONSTRAINT Билет_№_рейса_fk FOREIGN KEY (№_рейса) REFERENCESРейс(№_рейса)

Итоговая схема данных выглядит так:



2 Создание БД Школа

CREATE TABLE "Учителя"(
"код_учит" integer not null,
"фамилия" varchar(50) not null,
"имя" varchar(25) not null,
"отчество" varchar(50) not null,
"дата_рожд" date not null,
"ВУЗ" varchar(50) not null,
"год_ВУЗ" integer not null,
CONSTRAINT teachers_pkey PRIMARY KEY ("код_учит"));

CREATE TABLE "Классы"(
"класс" varchar(5) not null,
"число_уч" integer not null,
"клас_рук" integer not null,
CONSTRAINT class_pkey PRIMARY KEY ("класс"),
CONSTRAINT class_teachers_fkey FOREIGN KEY ("клас_рук") REFERENCES "Учителя" ("код_учит"));

CREATE TABLE "Ученики"(
"класс" varchar(5) not null,
"номер_уч" integer not null,
"фамилия" varchar(50) not null,
"имя" varchar(25) not null,
"отчество" varchar(50) not null,
"пол" char not null,
"адрес" varchar(50) not null,
CONSTRAINT stud_pkey PRIMARY KEY ("класс", "номер_уч"),
CONSTRAINT stud_class_fkey FOREIGN KEY ("класс") REFERENCES "Классы"("класс"));

CREATE TABLE "Нагрузка"(
"класс" varchar(5) not null,
"предмет" varchar(25) not null,
"код_учит" integer not null,
CONSTRAINT loading_pkey PRIMARY KEY ("класс", "предмет"),
CONSTRAINT loading_class_fkey FOREIGN KEY ("класс") REFERENCES "Классы"("класс"),
CONSTRAINT loading_teachers_fkey FOREIGN KEY ("код_учит") REFERENCES "Учителя"("код_учит"));

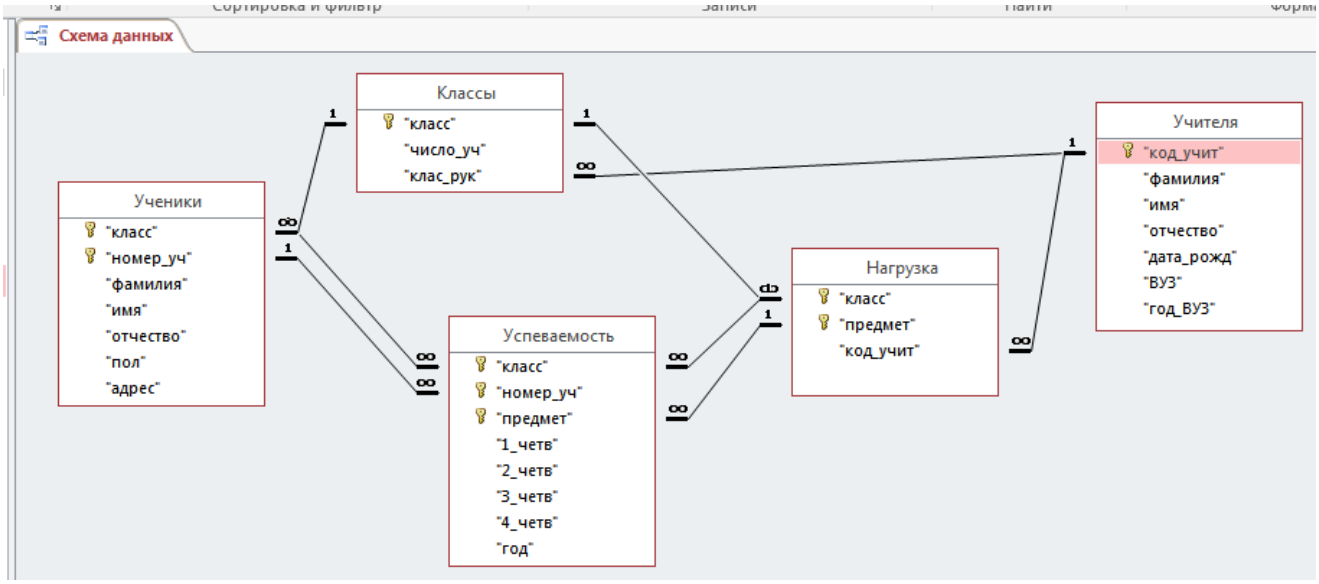
CREATE TABLE "Успеваемость"(

```
        "класс"         varchar(5)      not null,
        "номер_уч"      integer         not null,
        "предмет"       varchar(25)     not null,
        "1_четв"        integer         not null,
        "2_четв"        integer         not null,
        "3_четв"        integer         not null,
        "4_четв"        integer         not null,
        "год"           integer         not null,
        CONSTRAINT marks_pkey PRIMARY KEY ("класс", "номер_уч", "предмет"),
        CONSTRAINT marks_stud_fkey FOREIGN KEY ("класс", "номер_уч") REFERENCES
"Ученики"("класс", "номер_уч"),
        CONSTRAINT marks_loading_fkey FOREIGN KEY ("класс", "предмет") REFERENCES
"Нагрузка" ("класс", "предмет"));
        ------------------------------------------------------

INSERT INTO "Учителя"
        ("код_учит", "фамилия",   "имя",      "отчество",   "дата_рожд", "ВУЗ","год_ВУЗ")
values      (1,    'Аликина',   'Вера',       'Павловна',   '23.12.1963', 'ЮФУ',      1996),
            (2,    'Белых',     'Зинаида',    'Петровна',   '03.07.1960', 'РГУ',      1989),
            (3,    'Барсукова', 'Ирина',      'Ивановна',   '25.08.1972', 'ЮФУ',      1994),
            (4,    'Волегов',   'Михаил',     'Ильич',      '09.01.1970', 'РИНХ',     1993),
            (5,    'Диркс',     'Иван',       'Семенович',  '29.12.1965', 'РГУ',      1987),
            (6,    'Доброва',   'Галина',     'Сергеевна',  '12.11.1952', 'РГУ',      1976),
            (7,    'Жуковский', 'Дмитрий',    'Викторович', '24.10.1973', 'РИНХ',     1999);

INSERT INTO "Классы"
      ("класс",       "число_уч",   "клас_рук")
values ('7а',            34,           2),
       ('7б',            35,           5),
       ('8а',            31,           3),
       ('8б',            28,           1);

INSERT INTO "Ученики"
        ("класс",    "номер_уч",    "фамилия",  "имя", "отчество",    "пол",  "адрес")
  values ('7а',      1,             'Антипов',    'Петр', 'Иванович',   'м',    'Островского 3, кв. 12'),
         ('7а',      2,             'Березин',    'Игорь','Витпльевич','м',     'Пушкина 15, кв. 27'),
         ('7а',      3,             'Буракова',  'Инна','Владимировна', 'ж',    'Сормовская 1, кв. 3'),
         ('7б',      1,             'Асмолова',   'Вера', 'Павловна',   'ж',    'Кирова 5, кв. 87'),
         ('7б',      2,             'Вершинин',   'Олег', 'Николаевич', 'м',    'Пушкина 2, кв. 34'),
         ('7б',      3,          'Герасимова', 'Анна','Александровна',  'ж',    'Сормовская 4, кв. 21'),
         ('8а',      1,             'Антонов',    'Кирилл','Иванович', 'м',     'Садовая 5, кв. 56'),
         ('8а',      2,             'Веткина',    'Ирина','Андреевна', 'ж',     'Островского 3, кв. 41'),
         ('8а',      3,             'Вяткин',     'Иван', 'Павлович',   'м',    'Садовая 3, кв. 14'),
         ('8б',      1,          'Волегов','Кирилл','Дмитриевич',       'м',    'Садовая 3, кв. 4'),
         ('8б',      2,           'Гилев',  'Валерий','Петрович',       'м',    'Лебедева 43, кв. 4'),
         ('8б',      3,           'Ежова', 'Марина','Сергеевна',        'ж',    'Малькова 76, кв. 81');

INSERT INTO "Нагрузка"
        ("класс",       "предмет",      "код_учит")
values   ('7а',          'История',          1),
         ('7а',          'Литература',       3),
         ('7а',          'Математика',       5),
         ('7б',          'История',          1),
         ('7б',          'Литература',       3),
         ('7б',          'Математика',       6),
```

```
('8а',  'Информатика',  7),
('8а',  'История',  1),
('8а',  'Математика',  5),
('8б',  'Информатика',  7),
('8б',  'История',  4),
('8б',  'Математика',  5);
```

```
INSERT INTO "Успеваемость"
("класс",  "номер_уч",  "предмет",  "1_четв",  "2_четв",  "3_четв",  "4_четв",  "год")
values ('7а',  1,  'История',  4,  4,  4,  4),
('7а',  1,  'Литература',  3,  3,  3,  3,  3),
('7а',  1,  'Математика',  5,  5,  5,  5,  5),
('7а',  2,  'История',  5,  5,  4,  4,  4),
('7а',  2,  'Литература',  5,  4,  4,  4,  4),
('7а',  2,  'Математика',  4,  3,  4,  5,  4),
('7а',  3,  'История',  4,  4,  4,  4,  4),
('7а',  3,  'Литература',  4,  5,  5,  4,  5),
('7а',  3,  'Математика',  5,  4,  3,  3,  4),
('7б',  1,  'История',  3,  3,  3,  3,  3),
('7б',  1,  'Литература',  4,  4,  4,  4,  4),
('7б',  1,  'Математика',  3,  4,  3,  3,  3),
('7б',  2,  'История',  4,  4,  4,  4,  4),
('7б',  2,  'Литература',  4,  5,  5,  5,  5),
('7б',  2,  'Математика',  5,  5,  4,  5,  5),
('7б',  3,  'История',  3,  3,  3,  3,  3),
('7б',  3,  'Литература',  3,  3,  4,  3,  3),
('7б',  3,  'Математика',  4,  4,  3,  4,  4);
```

Итоговая схема данных выглядит так:



3 Реализация БД по индивидуальной теме

1. Реализовать скрипт создания БД в соответствии с вариантом с использованием инструкций части DDL языка SQL (CREATE TABLE, CREATE INDEX и т.д.);
2. При создании таблиц не забывать указывать необходимые ограничения целостности и значения по умолчанию (UNIQUE, DEFAULT, PRIMARY KEY, FOREGIN KEY и т.д.);
3. Заполнить таблицы созданной БД, выполняющие роль справочников, данными, с использованием инструкций части DML языка SQL (INSERT, UPDATE и т.д.);
4. Полученные инструкции (п.1-3) объединить в файл-скрипт и представить преподавателю для

защиты.

3.1 Темы баз данных

Выбор темы осуществляется в соответствии со списком группы.

1. Автосалон;
2. Видеопрокат;
3. Гостиница;
4. Кинотеатр;
5. Поликлиника;
6. Библиотека;
7. Фитнес-клуб;
8. Автошкола;
9. Химчистка;
10. Ресторан;
11. Аптека;
12. Турфирма;
13. Интернет-магазин;
14. Доставка пиццы;
15. Книжный магазин;
16. Компьютерные игры;
17. Автовокзал.

КОНТРОЛЬНЫЕ ВОПРОСЫ:

1. Как выполнить создание таблицы средствами языка SQL?
2. Как разделяются операторы SQL в случае нескольких операторов в запросе?
3. Каким образом выполнить простейшие операции вставки строк данных в таблицу средствами SQL?
4. Каким образом выполнить простейшие операции модификации строк таблицы средствами SQL?
5. Каким образом выполнить просмотр содержания таблицы?
6. Как заполнить таблицу средствами SQL?
7. Как выполнить создание таблицы средствами языка SQL?
8. Как разделяются операторы SQL в случае нескольких операторов в запросе?
9. Каким образом выполнить простейшие операции вставки строк данных в таблицу средствами SQL?
10. Каким образом выполнить простейшие операции модификации строк таблицы средствами SQL?
11. Каким образом выполнить просмотр содержания таблицы?
12. Как заполнить таблицу средствами SQL?