

Лабораторные работы по курсу «Проектирование и дизайн информационных систем»

Лабораторная работа № 1. Построение диаграммы декомпозиции первого уровня в нотации IDEF0

Цель работы:

- кратко описать выбранную предметную область (чем занимается предприятие, какие основные процессы в нем происходят)
- определить контекст моделирования
- построить контекстную диаграмму в нотации IDEF0

Задание

1. Согласно варианту, создайте контекстную диаграмму. Определите цель, точку зрения модели. Опишите свойства в соответствующих закладках диалога Model Properties.
2. Задайте входы, выходы, механизмы и управление.
3. Создайте декомпозицию контекстной диаграммы, состоящую из 2-3 блоков. Задайте автоматическую нумерацию блоков и ICOM-кодов.
4. Установите связи между блоками. Задайте имена дуг.
5. Сохраните проект в отдельный файл.

Разработать модель предметной области

«Проектирование ИС»

AllFusion Process Modeler (далее BPwin) — CASE-средство для моделирования бизнес-процессов, позволяющая создавать диаграммы в нотации IDEF0, IDEF3, DFD. В процессе моделирования BPwin позволяет переключаться с нотации IDEF0 на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. BPwin поддерживает функционально-стоимостной анализ (ABC).

Работа с программой начинается с создания новой модели, для которой нужно указать имя и тип (рис.1).

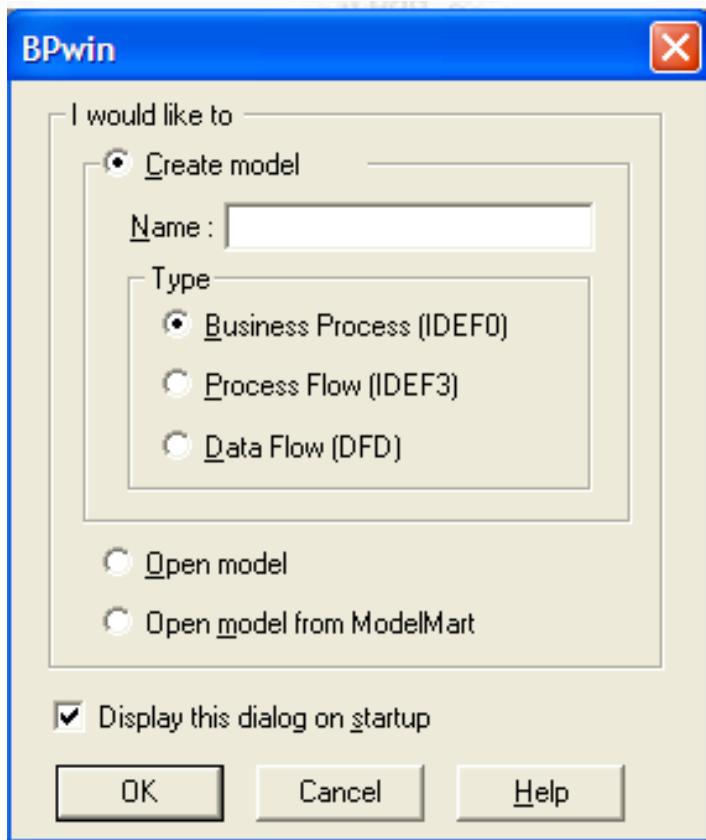


Рисунок 1 Окно создания новой модели

От выбора типа модели зависит в каких нотациях можно производить декомпозицию работ. Так, если выбрать тип *Business Process (IDEF0)*, то в созданной модели можно производить декомпозицию работ в нотациях IDEF0, IDEF3 и DFD; если выбран тип *Data Flow (DFD)* — в нотациях DFD и IDEF3; если же выбран тип *Process Flow (IDEF3)* — то только в нотации IDEF3.

После ввода имени модели и выбора ее типа BPWin сразу предложит задать параметры модели (рис. 2):

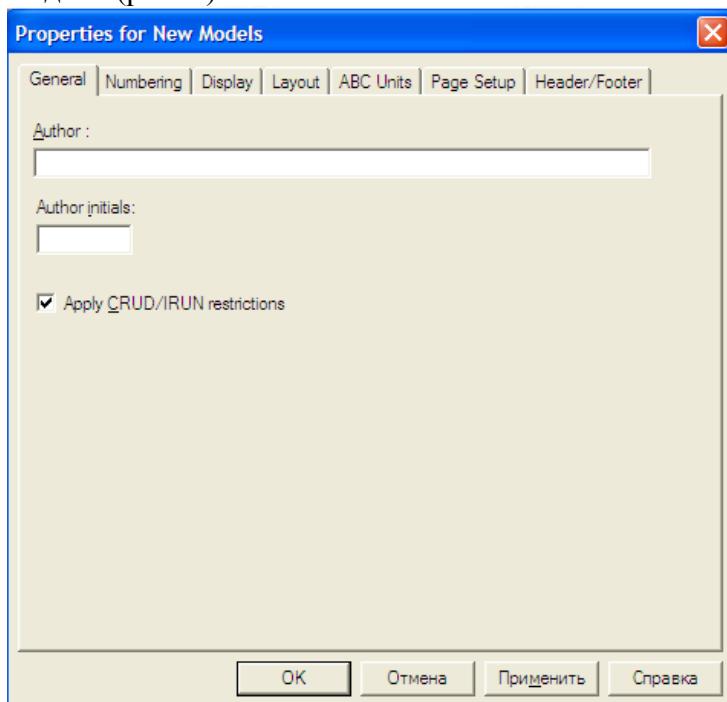


Рисунок 2 Задание параметров модели

- *General* — автор модели и его инициалы;
- *Numbering* — формат нумерации работ и диаграмм и порядок ее отображения на диаграммах;
- *Display* — список элементов отображения на диаграммах;
- *Layout* — параметры расположения;
- *ABC Units* — единицы функционально-стоимостного анализа;
- *Page Setup* — параметры страницы;
- *Header/Footer* — параметры верхнего и нижнего колонтитула.

После задания свойств модели появляется главное окно программы (рис. 3), состоящее из трех основных частей:

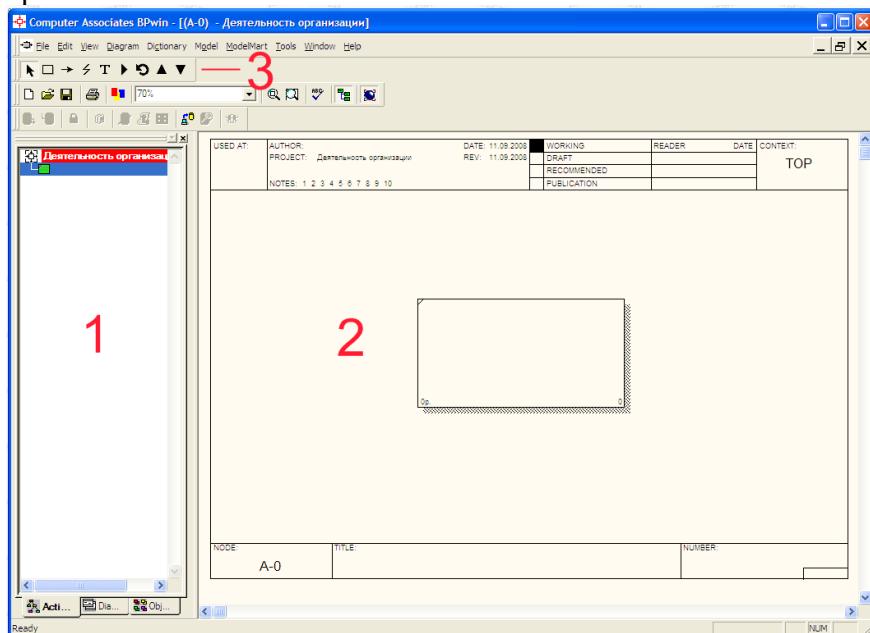


Рисунок 3 Окно работы программы

- 1 - обозреватель модели (Model Explorer) — отображает структуру модели (имеющиеся диаграммы и их иерархию);
- 2 - основная часть — в ней отображаются диаграммы, с которыми ведется работа;
- 3 - панели инструментов, из которых наибольший интерес представляет панель инструментов *Model Toolbox*.

Примечание. В созданной модели с настройками по умолчанию некорректно отображаются русские символы. Чтобы устранить этот недостаток, необходимо подкорректировать используемые в модели шрифты. Для этого в меню *Model -> Default Fonts* необходимо последовательно пройтись по всем пунктам (рис. 4), выбрать в выпадающем списке *Script* значение *кириллический* и поставить галочку *Change all occurrences* (рис. 5).

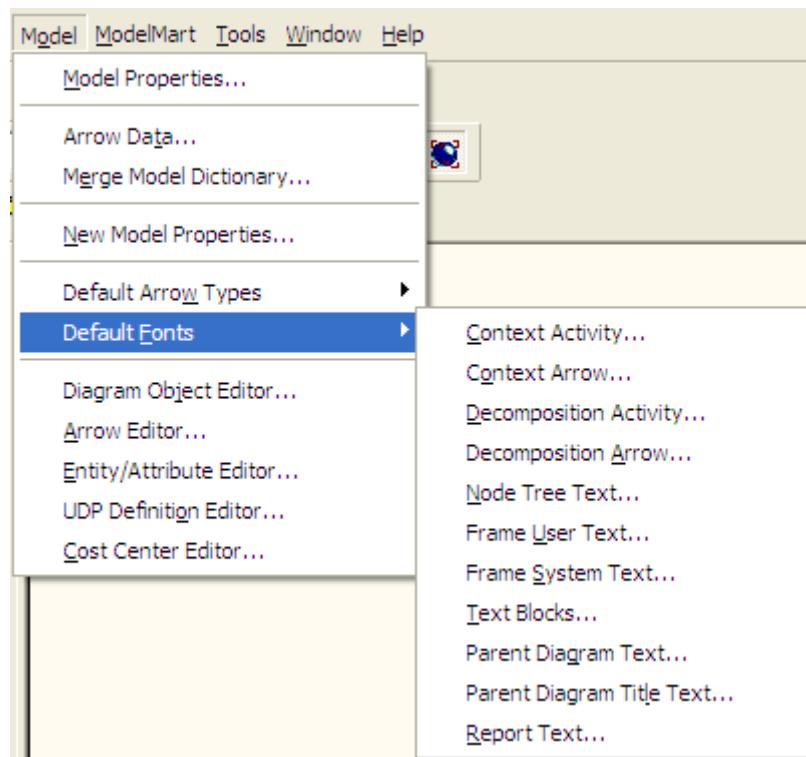


Рисунок 4. Пункты меню, отвечающие за настройки шрифта

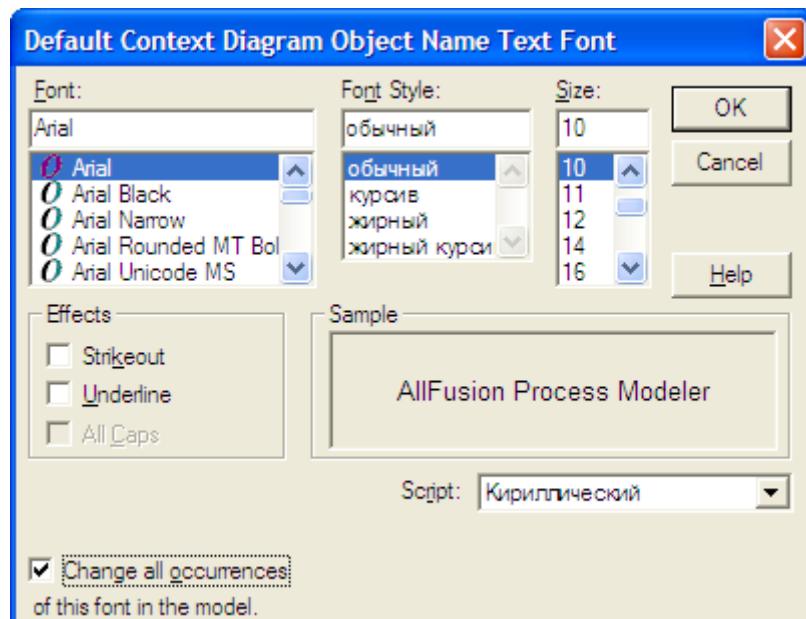


Рисунок 5. Параметры шрифта

Панель инструментов *Model Toolbox*.

Данная панель инструментов отвечает за создание разнообразных графических элементов модели. В зависимости от типа текущей диаграммы набор кнопок на ней меняется.

Таблица 1 - Вид и назначение кнопок *Model Toolbox*

Вид кнопки	Название кнопки	Назначение кнопки
	Pointer Tool	Превращает курсор в стрелку указателя для того, чтобы можно было выделять объекты

- IDEF0		
- DFD	Activity Box Tool	Добавление на диаграмму новой работы
- IDEF3		
	Precedence Arrow Tool	Добавление на диаграмму новой стрелки
	Squiggle Tool	Связывание названия стрелки с самой стрелкой
	Text Tool	Добавление на диаграмму текста
	Diagram Dictionary Editor	Вызов окна менеджера диаграмм для просмотра имеющихся диаграмм по типам и переход к выбранной
	Go to Sibling Diagram	Переход между стандартной диаграммой, деревом узлов и FEO диаграммой
	Go to Parent Diagram	Переход к родительской диаграмме
	Go to Child Diagram	Переход к дочерней диаграмме
- DFD	External Reference Tool	Добавление на диаграмму внешней сущности
- DFD	Data store Tool	Добавление на диаграмму хранилища данных
- IDEF3	Junction Tool	Добавление на диаграмму перекрестка
- IDEF3	Referent Tool	Добавление на диаграмму объекта ссылки

Созданная модель уже содержит контекстную диаграмму с единственной работой ("черный ящик") в той нотации, которая была выбрана на этапе создания модели. Теперь необходимо дать этой работе название и при необходимости задать ее свойства. Для этого нужно вызвать окно свойств работы, дважды щелкнуть по ней мышью (рис. 6).

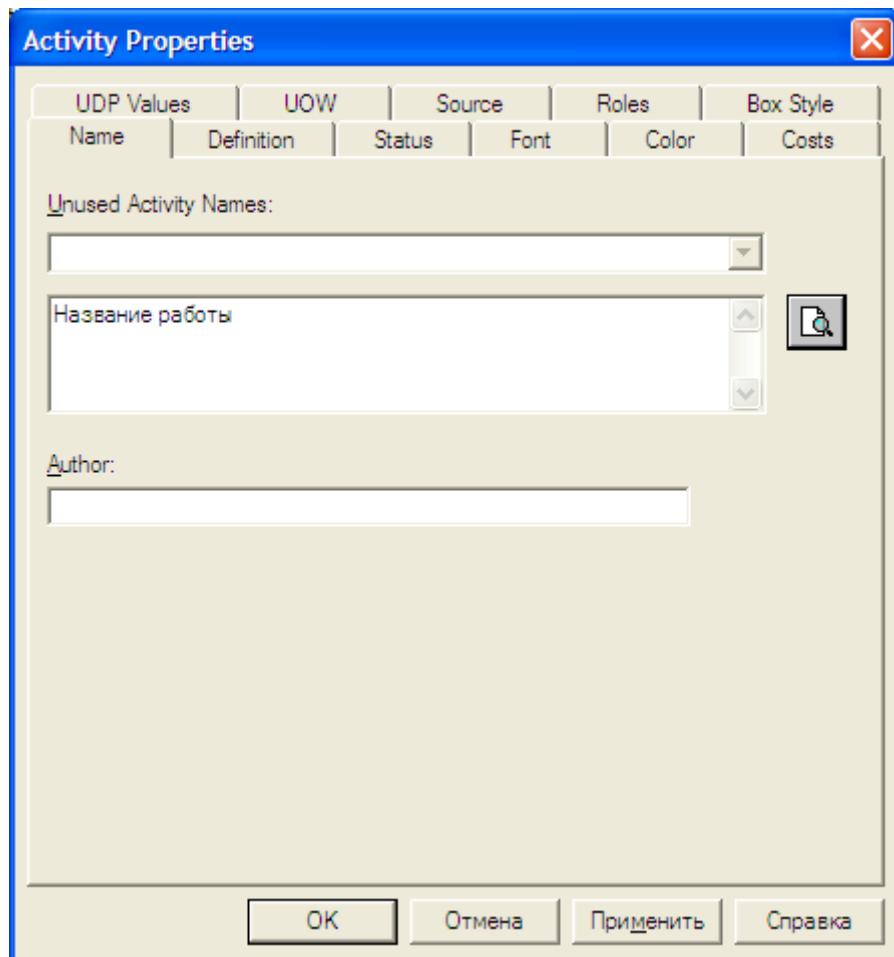


Рисунок 6. Окно свойств работы

Далее необходимо разместить на диаграмме стрелки. Для этого следует нажать на *Model Toolbox* кнопку *Precedence Arrow Tool* (курсор примет форму крестика со стрелкой), щелкнуть по тому месту, откуда стрелка должна выходить и затем щелкнуть по тому месту, куда стрелка должна заходить (BPwin подсветит эти места при наведении на них курсора). Для задания названия стрелки нужно нажать на *Model Toolbox* кнопку *Pointer Tool* и затем дважды щелкнуть по стрелке. В появившемся окне *Arrow Properties* название работы вводится в поле *Arrow Name* или выбирается из списка имеющихся названий стрелок.

После размещения стрелок на диаграмме можно проводить декомпозицию ее работ. Для этого следует нажать на *Model Toolbox* кнопку *Go to Child Diagram* и затем щелкнуть по работе, которую нужно декомпозировать. Появится окно, в котором необходимо выбрать в какой нотации проводить декомпозицию и количество дочерних работ (рис. 7).

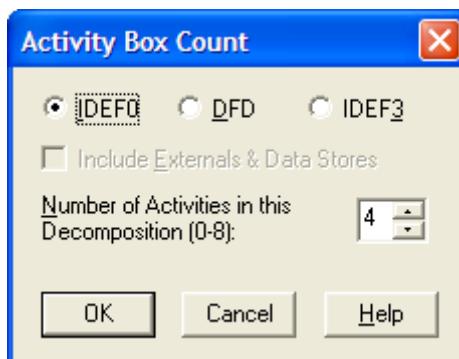


Рисунок 7. Создание дочерней диаграммы

После создания дочерней диаграммы BPwin автоматически создаст указанное число работ и разместит граничные стрелки по краям диаграммы. Далее следует связать граничные стрелки со входами работ (при необходимости можно добавить новые граничные стрелки) и связать работы между собой. Дальнейшая декомпозиция работ проводится аналогичным образом.

Целью данной и большинства последующих работ является моделирование деятельности выбранного предприятия. Для этого будут применяться методологии:

- IDEF0 - методология функционального моделирования
- IDEF3 - методология описания процессов
- DFD - методология моделирования потоков данных
- IDEF1X - методология моделирования данных

Диаграммы в первых трех методологиях будут создаваться с помощью CASE-средства AllFusion Process Modeler, IDEF1X - с помощью AllFusion ERwin Data Modeler.

Каждая диаграмма в нотациях IDEF0, IDEF3, DFD предназначена для описания одного или нескольких бизнес-процессов. **Бизнес-процесс** - это устойчивая, целенаправленная совокупность взаимосвязанных видов деятельности (последовательность работ), которая по определенной технологии преобразует входы в выходы, представляющие ценность для потребителя.

Результатом моделирования бизнес-процессов является модель бизнес-процессов, которая относится к одному из трех типов:

- модель AS-IS (как есть) - модель текущей организации бизнес-процессов предприятия
- модель TO-BE (как будет) - модель идеальной организации бизнес-процессов
- модель SHOULD-BE(как должно бы быть) - идеализированная модель, не отражающая реальную организацию бизнес-процессов предприятия

В лабораторных работах будет создаваться модель AS-IS.

Перед началом построения диаграмм необходимо изучить выбранную предметную область. В этой и последующих работах в качестве предметной области будет выступать вымышленное предприятие по сборке и продаже настольных компьютеров и ноутбуков. Компания не производит комплектующие самостоятельно, а только собирает и тестирует компьютеры. Основные процедуры в компании:

- продавцы принимают заказы клиентов;
- сотрудники группируют заказы по типам компьютеров;
- сотрудники собирают и тестируют компьютеры;
- сотрудники упаковывают компьютеры согласно заказам;
- кладовщик отгружает клиентам заказы
- снабженцы заказывают и доставляют комплектующие, необходимые для сборки.

Компания использует купленную бухгалтерскую информационную систему, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Построение модели какой-либо системы в методологии IDEF0 начинается с определения **контекста моделирования**, который включает в себя субъекта моделирования, цель моделирования и точку зрения на модель.

Под **субъектом** понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, необходимо определить, что в дальнейшем будет рассматривать как компоненты системы, а что как внешнее воздействие.

Цель моделирования. Модель не может быть построена без четко сформулированной цели. Цель должна отвечать на следующие вопросы:

- Почему этот процесс должен быть смоделирован?
- Что должна показывать модель?
- Что может получить читатель?

Точка зрения. Не смотря на то, что при построении модели учитываются мнения различных людей, модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. В течении моделирования важно оставаться на выбранной точке зрения.

В данной работе субъектом будет выступать само предприятие, а именно процессы, происходящие внутри него; цель моделирования - воспроизвести бизнес-процессы, происходящие на предприятии (модель AS-IS); точка зрения - с позиции директора как лица, знающего структуру предприятия в целом.

После определения контекста моделирования можно приступить к построению контекстной диаграммы (называемой еще "черным ящиком"). Данный тип диаграммы позволяет показать, что подается на вход работы и что является результатом работы, без детализации ее составляющих. Данная диаграмма содержит только одну работу, которая будет представлять всю деятельность предприятия в целом (рис.8).

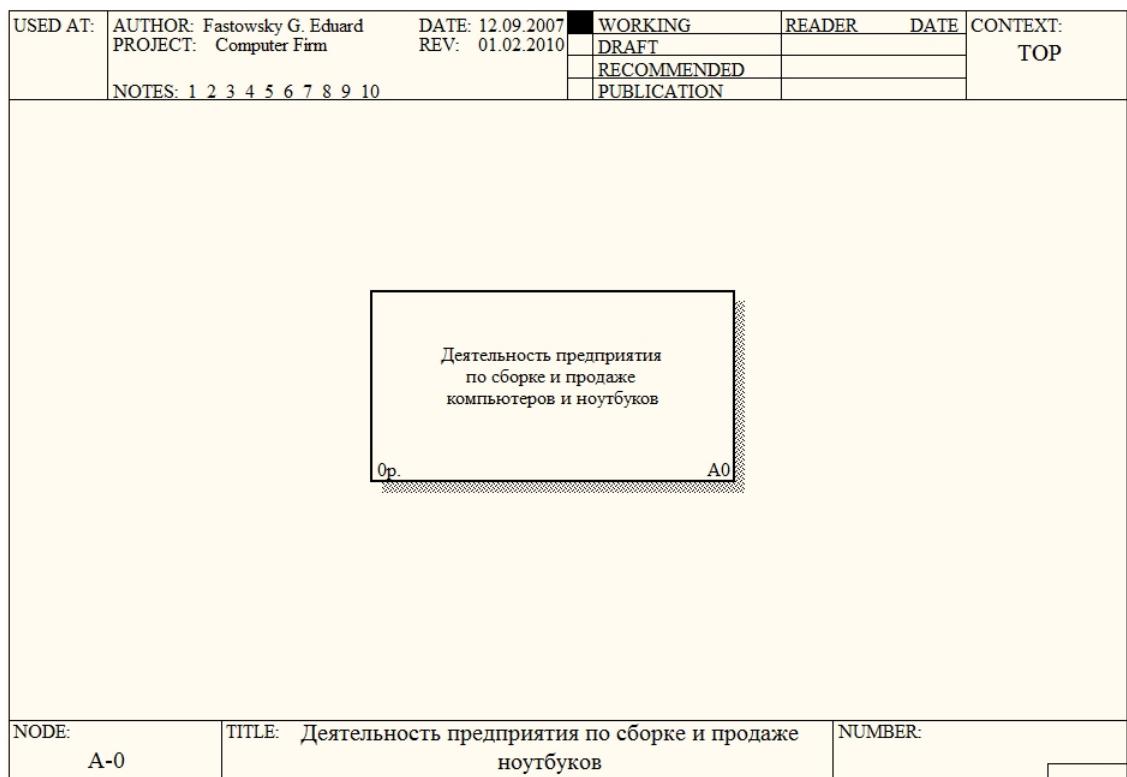


Рисунок 8. Контекстная диаграмма

Любая IDEF0 диаграмма состоит из прямоугольников, называемых работами (activity), и стрелок (arrow). Работа представляет собой некоторую конкретную функцию в рамках рассматриваемой системы. По требованиям стандарта название каждой работы должно

быть выражено отглагольным существительным (например, "Изготовление детали", "Оформление заказа" и т.д.). Каждая из четырех сторон прямоугольника имеет свое определенное значение (рис.9):

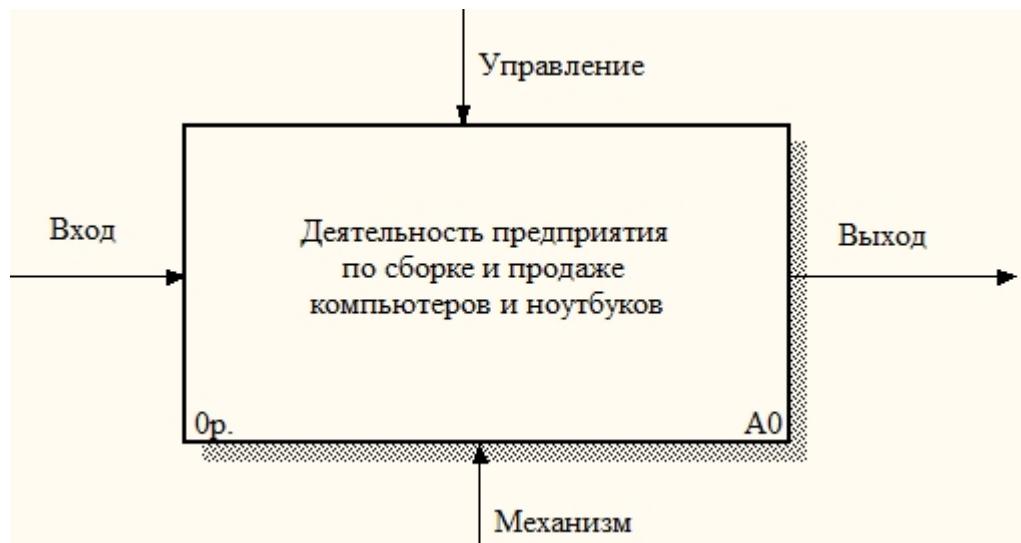


Рисунок 9. Работа в IDEF0

- Вход – это потребляемая или изменяемая работой информация или материал
- Выход – информация или материал, которые производятся работой
- Управление – процедуры, правила, стратегии или стандарты, которыми руководствуется работа
- Механизмы – ресурсы, которые выполняют работу (например, сотрудники, оборудование, устройства и т.д.)

Для рассматриваемого предприятия *входными стрелками* будут:

- Заказы клиентов - список компьютеров и их конфигурация, которые клиент желает приобрести
- Комплектующие от поставщиков - комплектующие, полученные от поставщиков, из которых собираются компьютеры и ноутбуки

Выходные стрелки:

- Готовая продукция - собранные компьютеры и ноутбуки
- Заказы поставщикам - список комплектующих, которые предприятие закупает у поставщиков
- Оплата за комплектующие - деньги поставщикам за комплектующие
- Маркетинговые материалы - прайс-листы, рекламки и т.п.

Стрелки управления:

- Законодательство - различные законодательные документы, которыми руководствуется предприятие в процессе своей деятельности
- Правила и процедуры - различные правила и процедуры, которыми руководствуется предприятие в процессе своей деятельности (например, правила сборки и тестирования компьютеров, процедура общения с клиентами и т.п.)

Стрелки механизмов:

- Бухгалтерская система
- Персонал

Итоговая контекстная диаграмма имеет вид (рис.10):

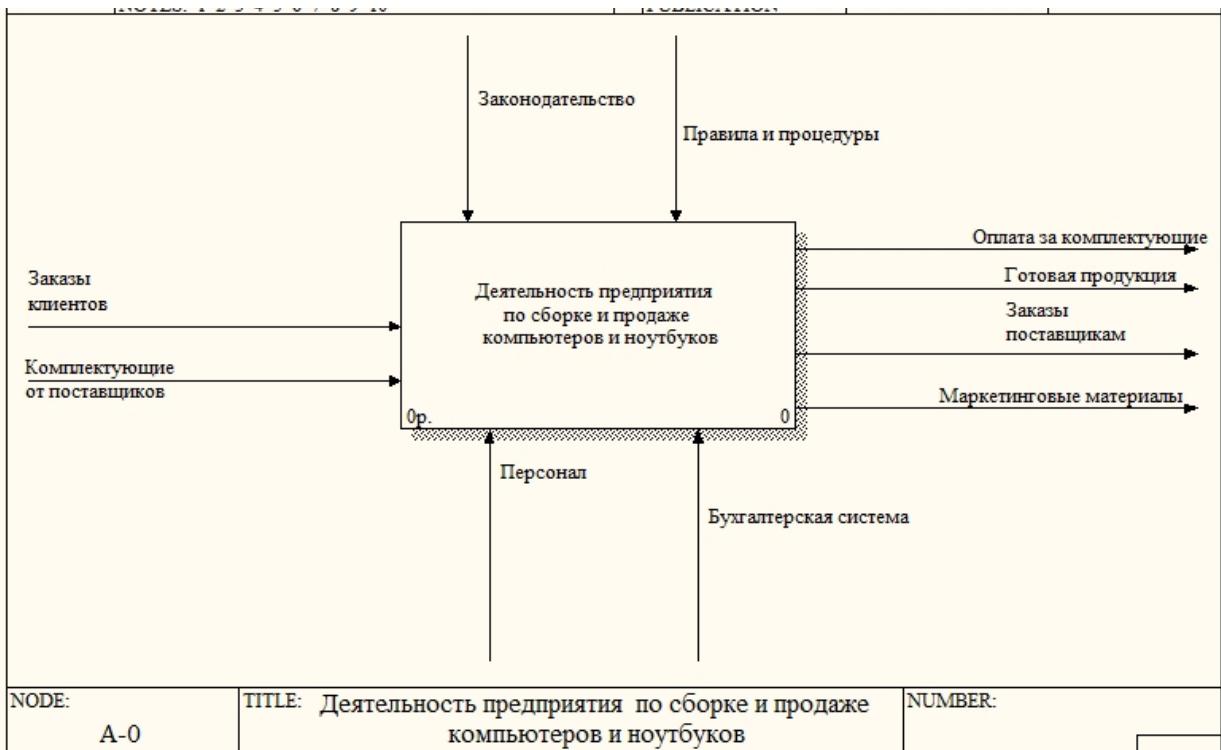


Рисунок 10. Итоговая контекстная диаграмма

Содержание отчета:

- вариант индивидуального задания
- краткое описание выбранной предметной области (чем предприятие занимается, как функционирует)
- описание контекста моделирования
- контекстная диаграмма

Контрольные вопросы:

- Для чего используется методология IDEF0.
- Объясните необходимость задания цели и точки зрения модели?
- Перечислите и расскажите назначения кнопок на панели инструментов.
- Перечислите этапы декомпозиции блока.
- Расскажите, каким образом на диаграмму добавить блок, дугу.
- Дайте определение ICOM-кодов.
- Для чего используются закладки General, Purpose, Definition, Status, Numbering, Display в диалоге Model Properties.

Лабораторная работа № 2

Построение диаграммы декомпозиции второго уровня в нотации IDEF0

Цель работы:

- построить диаграмму декомпозиции второго уровня в нотации IDEF0

В предыдущей работе была построена контекстная диаграмма, содержащая только одну работу, которая описывает деятельность предприятия в целом, без детализации составляющих этой работы. В данной работе будет построены диаграммы декомпозиции первого и второго уровней в нотации IDEF.

Декомпозиция - это разделение сложного объекта, системы, задачи на составные части, элементы.

С помощью диаграммы декомпозиции первого уровня покажем, из каких более мелких работ состоит работа "Деятельность предприятия по сборке и продаже компьютеров и ноутбуков". В данной работе были выделены следующие дочерние работы:

Управление	Данная работа включает в себя общее управление предприятием, финансами, кадрами, бухгалтерией и т.п.
Продажи и маркетинг	Работа с клиентами, презентации, выставки, реклама, маркетинговые исследования и т.д.
Сборка и тестирование компьютеров	Сборка и тестирование настольных компьютеров и ноутбуков
Отгрузка и снабжение	Снабжение предприятия необходимыми комплектующими, хранение и отгрузка готовой продукции

После создания дочерней диаграммы первым действием соединим граничные стрелки с работами . Стрелку "*Заказы клиентов*" соединим с работой "*Продажи и маркетинг*", стрелку "*Комплектующие от поставщиков*" - с "*Отгрузка и снабжение*". Выходом работы "*Управление*" будет "*Оплата за комплектующие*", выходом "*Продажи и маркетинг*" - "*Маркетинговые материалы*". Стрелки "*Заказы поставщикам*" и "*Готовая продукция*" - выход работы "*Отгрузка и снабжение*".

Стрелка "*Персонал*" будет являться входом механизма всех четырех работ, а стрелка "*Бухгалтерская система*" - работ "*Продажи и маркетинг*" и "*Отгрузка и получение*". Стрелка "*Правила и процедуры*" будет входом управления всех четырех работ.

Любую ветвь стрелки также можно декомпозировать и дать ей свое название. Покажем это на примере ветви стрелки "*Бухгалтерская система*" для работы "*Продажи и маркетинг*". Назовем ее "*Система оформления заказа*". В AllFusion Process Modeler для более четкого указаня какое название к какой стрелке относится существуют несколько механизмов, одним из которых является **Squiggle** - стрелка в виде молнии, соединяющая название со стрелкой. Воспользуемся им для, для чего щелкнем правой кнопкой по названию стрелки и выберем в выпадающем меню соответствующий пункт.

На данном этапе построения диаграммы выяснилось, что мы не учли такой важный фактор, как деньги, которые клиенты дают за готовую продукцию. Деньги клиентов - это вход работы "Деятельность предприятия по сборке и продаже компьютеров и ноутбуков". Добавим эту стрелку на диаграмму декомпозиции.

Если по каким-то причинам граничную стрелку дочерней диаграммы не следует показывать (например, на данной диаграмме она является несущественной, или чтоб не загромождать диаграмму), то ее можно просто удалить. Удалим стрелку "*Законодательство*".

Результат всех перечисленных действий показан на рис.1.

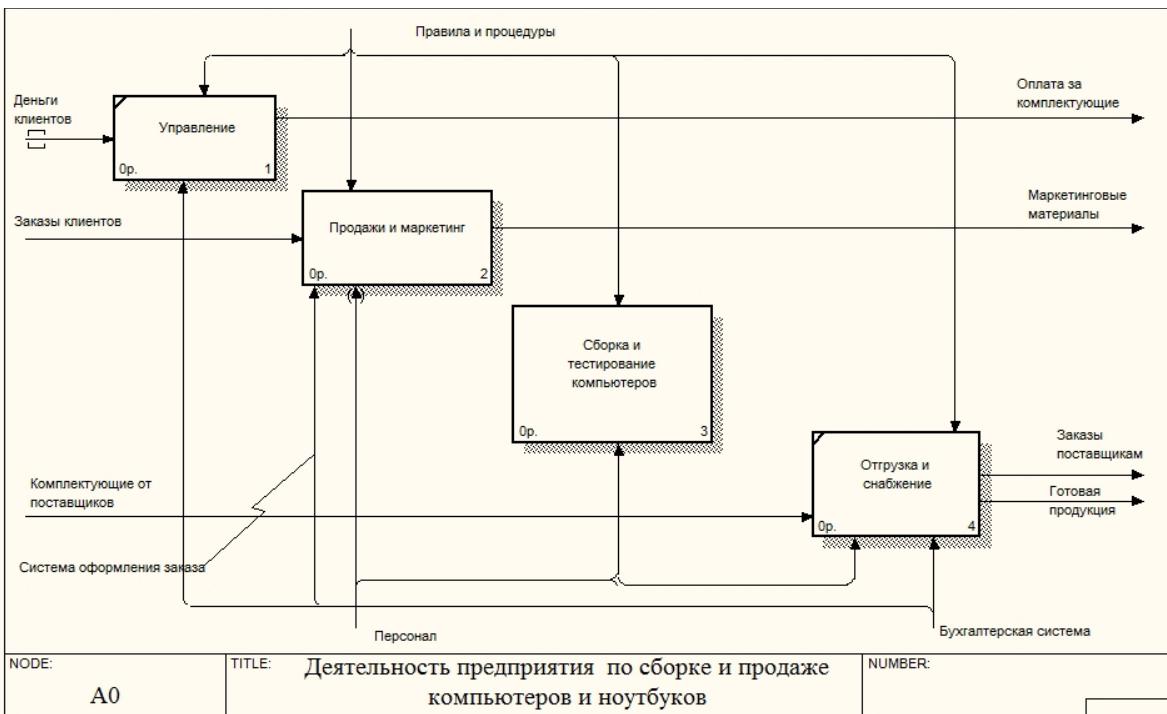


Рисунок 1

Если посмотреть на стрелку "Деньги клиентов" диаграммы декомпозиции и на стрелку "Законодательство" контекстной диаграммы, то видно, что они окружены небольшими квадратными скобками. Это означает, что данная граничная стрелка является новой на диаграмме и ее нет на дочерней диаграмме (как в случае со стрелкой "Законодательство"), или же данная стрелка является новой на дочерней диаграмме и ее нет на родительской (как в случае со стрелкой "Деньги клиентов"). От стрелок с квадратными скобками необходимо избавляться. Для этого есть два пути:

- добавить их на родительскую или дочернюю диаграмму, т.е. сделать граничной
- затоннелировать

Чтобы добавить такую стрелку на другую диаграмму или затоннелировать, нужно щелкнуть по квадратным скобкам правой кнопкой мыши и выбрать пункт меню "Arrow Tunnel". В появившемся окне следует выбрать один из двух вариантов: *Resolve it to border arrow* - сделать стрелку граничной, *Change it to resolved rounded tunnel* - затоннелировать стрелку. В данном случае мы решили обе стрелки затоннелировать (рис.2 и рис.3).

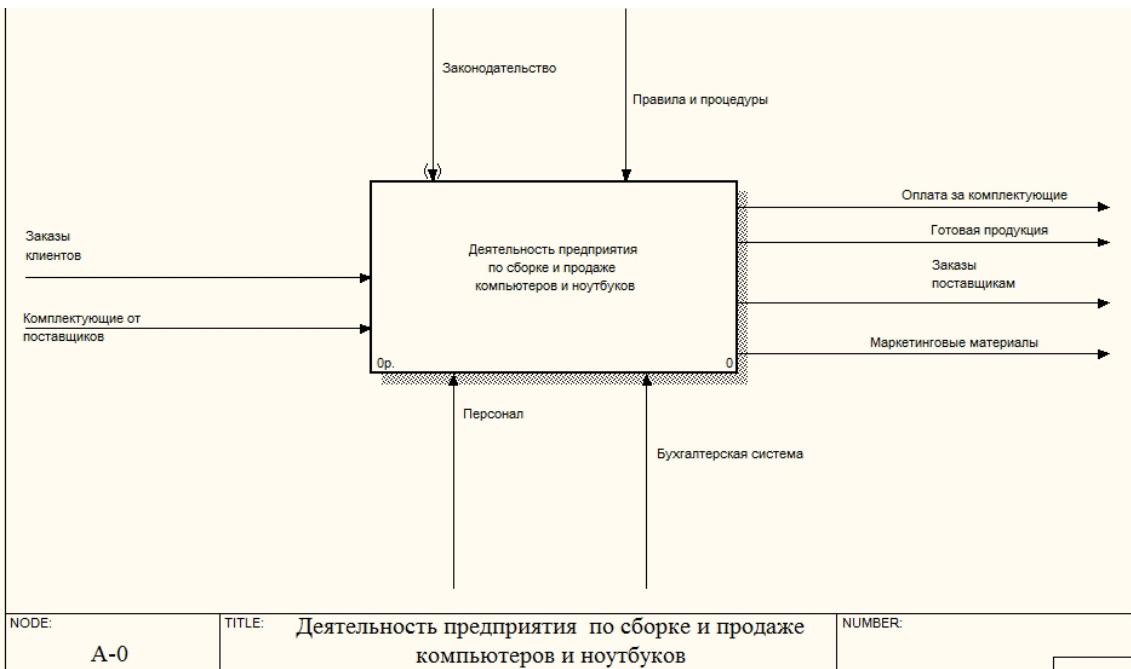


Рисунок 2. Контекстная диаграмма с затоннелированной границей стрелкой

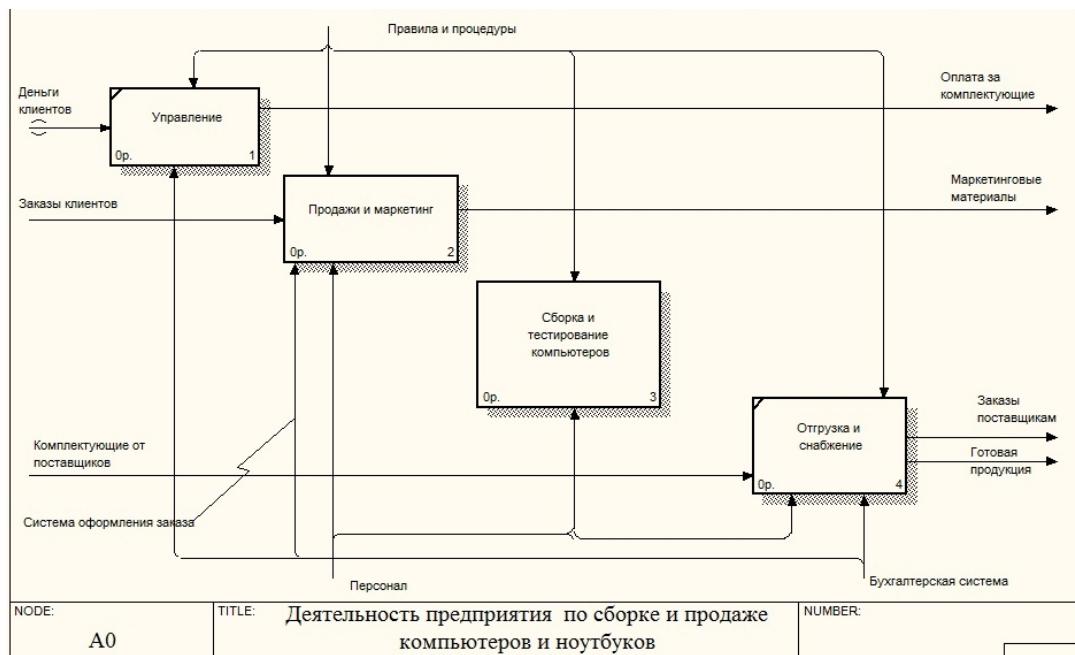


Рисунок 3. Диаграмма декомпозиции с затоннелированной границей стрелкой

После соединения граничных стрелок с работами следующим шагом соединим работы между собой. Поскольку работа "Управление" включает в себя общее управление предприятием, то одним из ее результатов будет являться "Управляющая информация", поступающая на вход управления всех остальных работ.

Работа "Продажи и маркетинг" получает на входе заказы клиентов (т.е. количество компьютеров и их конфигурация), информацию о которых она передает работе "Сборка и тестирование компьютеров" в качестве управляющей информации.

Работе "Сборка и тестирование" для своего функционирования необходимы комплектующие, которые она заказывает у работы "Отгрузка и снабжение" (выходная стрелка "Список необходимых комплектующих"). Собранные компьютеры она также передает работе "Отгрузка и снабжение" (выходная стрелка "Собранные компьютеры"). Информация о результатах сборки и тестирования необходима работе "Продажи и маркетинг" (выходная стрелка "Результаты сборки и тестирования").

Результатом работы "Отгрузка и снабжение" будут необходимые комплектующие, которые поступают на вход работы "Сборка и тестирование компьютеров".

Управление любого предприятия должно знать, что происходит на предприятии, чем занимается каждое подразделение и каковы результаты их работы, т.е. любая работа в идеале должна отчитываться о результатах своей деятельности перед управлением. Создадим стрелки выходов работ "Продажи и маркетинг", "Сборка и тестирование компьютеров" и "Отгрузка и снабжение" и соединим их со входом управления работы "Управление".

Результат соединения работ между собой показан на рисунке 3:

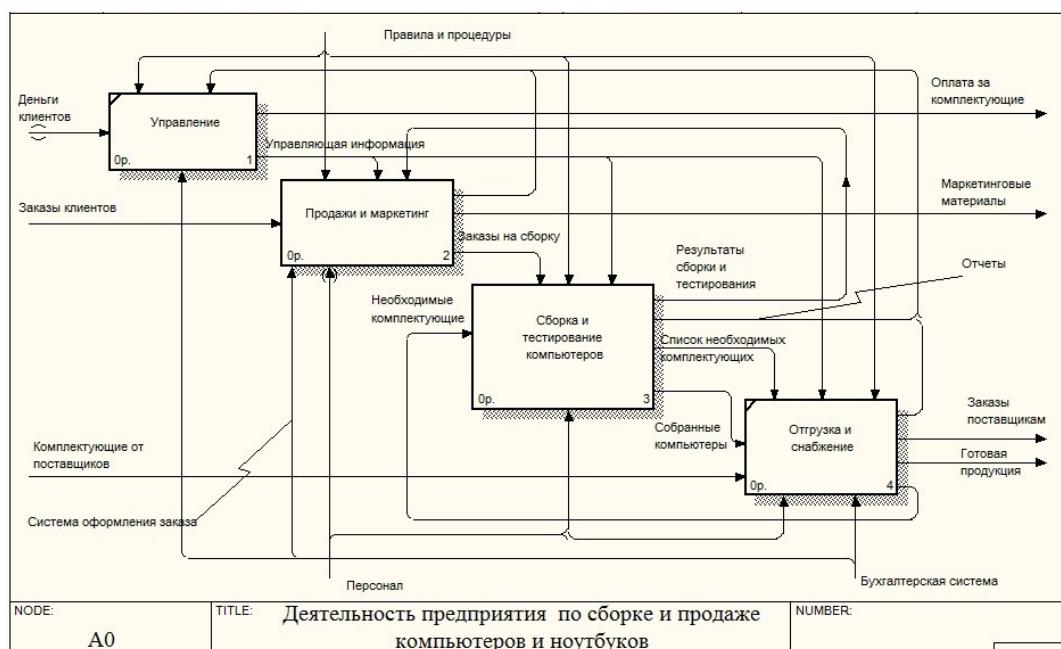


Рисунок 3

Если на диаграмме присутствует много работ и стрелок, то бывает затруднительно читать диаграмму. Для облегчения изучения диаграммы отдельные стрелки можно визуально выделить. Для зрительного выделения стрелки, соединяющей две работы, есть несколько механизмов:

- задать толщину стрелки
- поменять цвет стрелки
- добавить на стрелку дополнительные наконечники

Толщина и цвет стрелки задаются в окне свойств стрелки, вызываемое двойным щелчком по стрелке. Вкладка "Style" отвечает за стиль стрелки, в том числе и за ее толщину ("Thickness"), вкладка "Color" - за ее цвет. Для добавления на стрелку дополнительных наконечников следует щелкнуть правой кнопкой по стрелке и выбрать пункт меню "Extra Arrowhead".

Модифицируем диаграмму, визуально выделив некоторые стрелки. Итоговая диаграмма декомпозиции показана на рисунке 4:

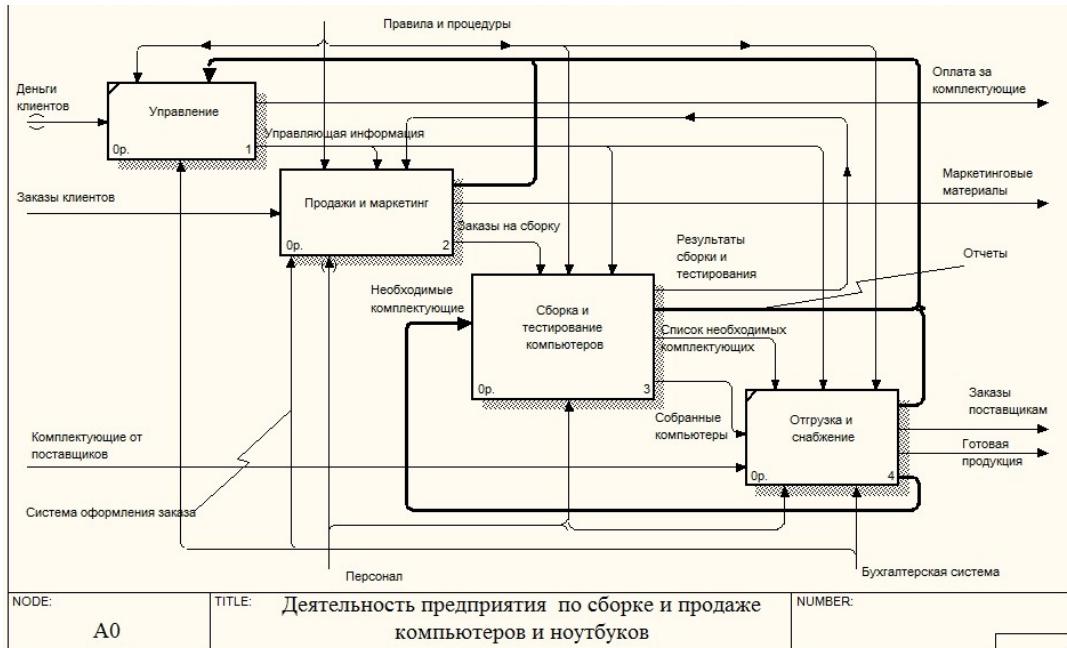


Рисунок 4. Итоговая диаграмма декомпозиции первого уровня

Содержание отчета:

- диаграмма декомпозиции второго уровня
- краткое описание каждой работы

Лабораторная работа № 3.

Последующая декомпозиция предметной области в нотации IDEF0

Цель работы:

- построить диаграмму декомпозиции следующего уровня в нотации IDEF0

В данной лабораторной работе построим еще одну диаграмму декомпозиции в нотации IDEF0 - декомпозицию работы "*Сборка и тестирование компьютеров*" диаграммы A0.

В результате проведения экспертизы получена следующая информация.

- Производственный отдел получает заказы клиентов от отдела продаж по мере их поступления.
- Диспетчер координирует работу сборщиков, сортирует заказы, группирует их и дает указание на отгрузку компьютеров, когда они готовы.
- Каждые 2 часа диспетчер группирует заказы - отдельно для настольных компьютеров и ноутбуков - и направляет на участок сборки.
- Сотрудники участка сборки собирают компьютеры согласно спецификациям заказа и инструкциям по сборке. Когда группа компьютеров, соответствующая группе заказов, собрана, она направляется на тестирование.

- Тестировщики testируют каждый компьютер и в случае необходимости заменяют неисправные компоненты. Тестировщики направляют результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров, соответствующих группе заказов, на отгрузку.

В данной работе мы выделили четыре дочерних работы: "*Отслеживание расписания и управление сборкой и тестированием*", "*Сборка настольных компьютеров*", "*Сборка ноутбуков*" и "*Тестирование компьютеров*". Как и в предыдущей работе начнем с соединения граничных стрелок с работами.

Стрелка "*Необходимые комплектующие*" - это вход работ "*Сборка настольных компьютеров*" и "*Сборка ноутбуков*".

Стрелки управления "*Управляющая информация*" и "*Заказы на сборку*" соединим с работой "*Отслеживание расписания и управление сборкой и тестированием*", поскольку именно данная работа управляет всем процессом сборки и тестирования, а стрелку управления "*Правила и процедуры*" - с остальными тремя работами.

Персонал принимает участие во всех выделенных дочерних работах, поэтому заводим стрелку "*Персонал*" на вход механизма всех работ (при этом указав, что в первой работе участвует диспетчер, а в четвертой - тестировщик).

Список необходимых комплектующих - это один из результатов работ "*Сборка настольных компьютеров*" и "*Сборка ноутбуков*". Результаты сборки и тестирования - это выходы работ "*Сборка настольных компьютеров*", "*Сборка ноутбуков*" и "*Тестирование компьютеров*". Компьютеры считаются собранными после того, как они успешно прошли тестирование, поэтому стрелка выхода "*Собранные компьютеры*" - выход работы "*Тестирование компьютеров*". Различные отчеты формирует работа "*Отслеживание расписания и управление сборкой и тестированием*".

Результат проделанных операций показан на рисунке 1:

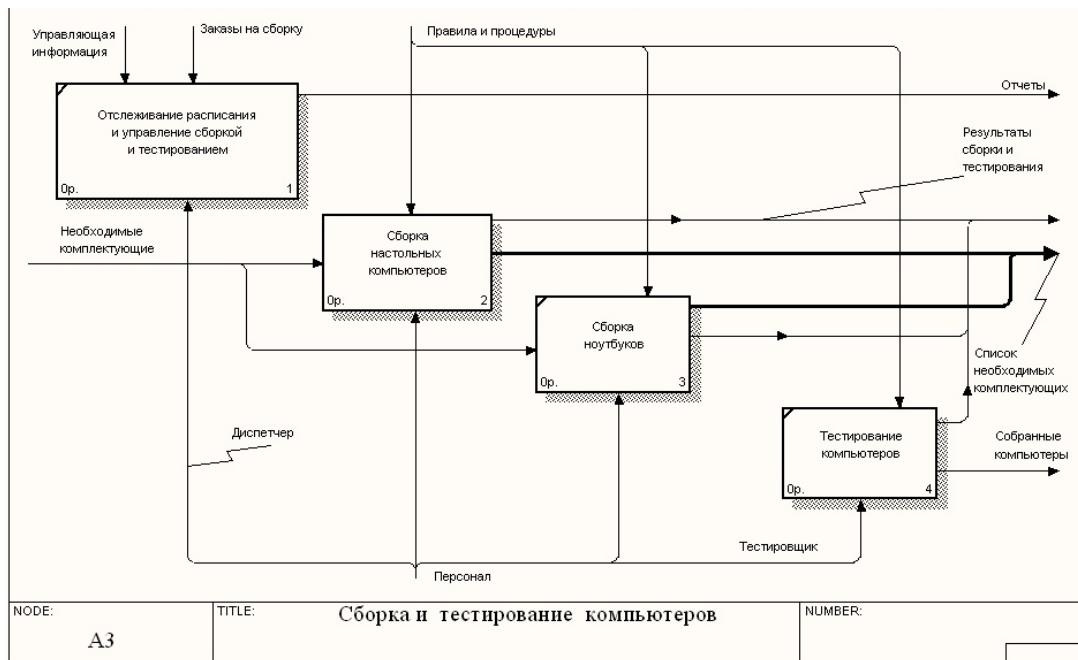


Рисунок 1

После соединения граничных стрелок с работами следующим шагом соединим работы между собой. Поступающие заказы на сборку сортируются диспетчером, после чего они поступают на вход управления работ "Сборка настольных компьютеров" и "Сборка ноутбуков"(стрелки Заказы на настольные компьютеры и "Заказы на ноутбуки", соответственно). Когда компьютеры собраны, диспетчер дает указание на их отгрузку (стрелка "Указание передать компьютеры на отгрузку").

Собранные компьютеры (выходы работ "Сборка настольных компьютеров" и "Сборка ноутбуков") должны быть протестираны, для чего они должны поступать на вход работы "Тестирование компьютеров" - стрелки "Настольные компьютеры" и "Ноутбуки".

После тестирования компьютеров отчет (стрелка "Результаты тестирования") направляется диспетчеру, который дает указание отгрузить компьютеры.

Итоговая диаграмма декомпозиции показана на рисунке 2:

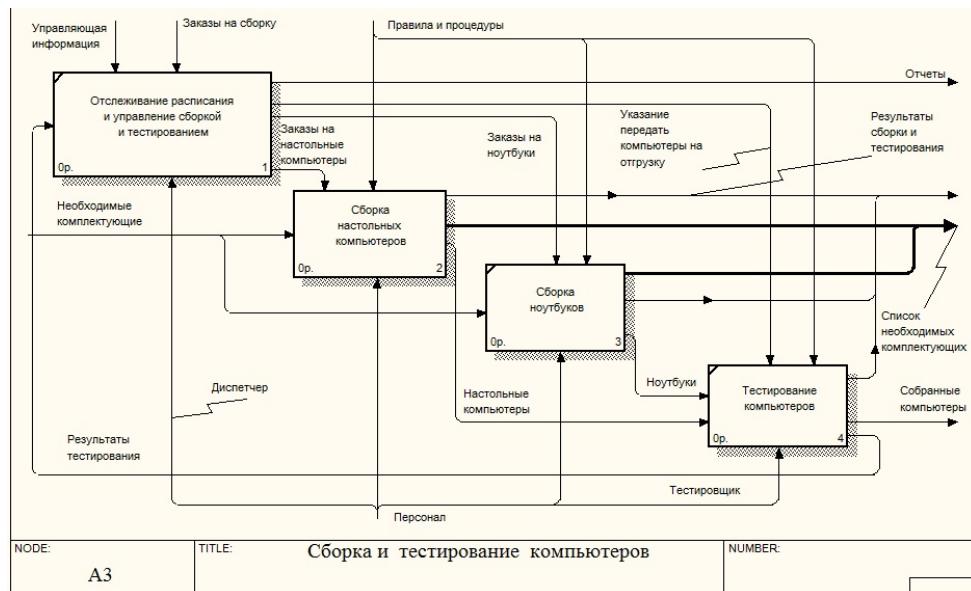


Рисунок 2

Содержание отчета:

- диаграмма декомпозиции
- краткое описание каждой работы

Лабораторная работа 4 BPwin 4.0 - Стоимостный анализ (Activity Based Costing).

Методика выполнения упражнения

- В диалоговом окне **Model Properties** (вызывается из меню **Mode/Model Properties**) во вкладке **ABC Units** (рисунок 9.1) установите единицы измерения денег - рубли и времени - часы.

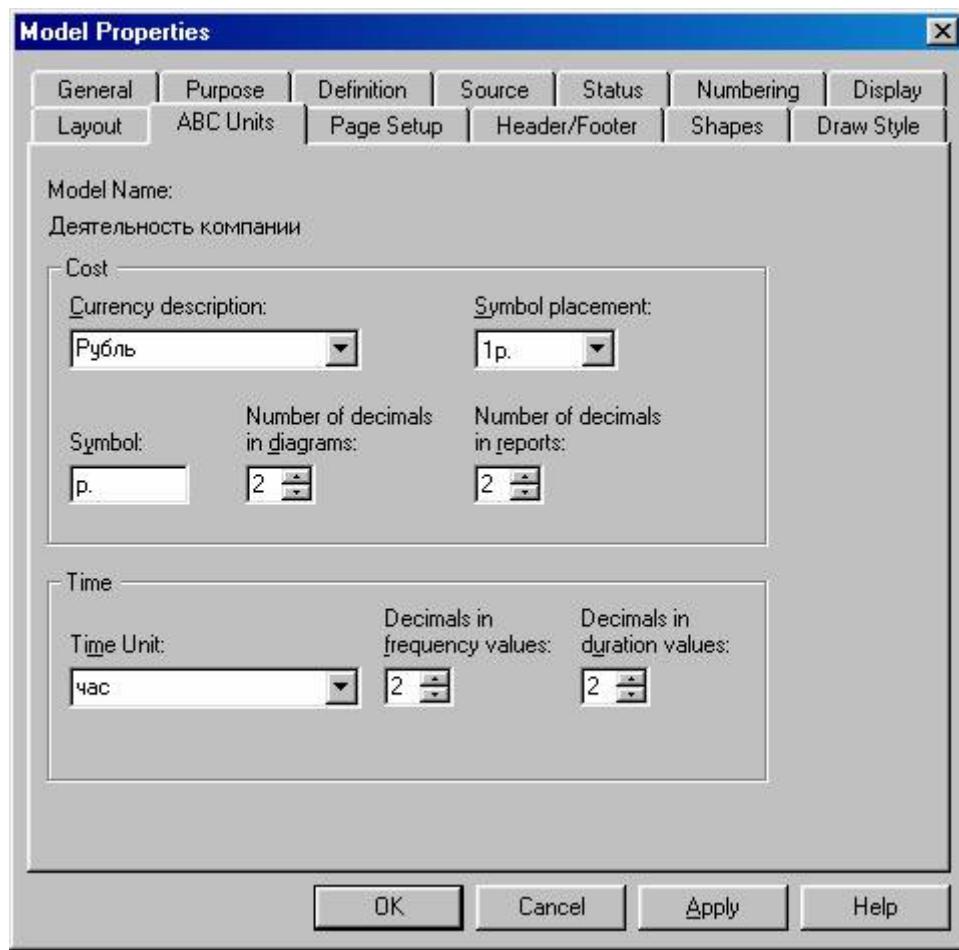


Рисунок 9.1- Вкладка ABC Units диалога Model Properties

- 2 Перейдите в меню **Dictionary/Cost Center** (Словарь/Центр Затрат) (рисунок 9.2) и в окне **Cost Center Dictionary** (Словарь Центра Затрат) (рисунок 9.3) внесите название и определение центров затрат (таблица 9.1). Вид окна **Cost Center Dictionary** после внесения название и определение центров затрат представлен на рисунке 9.4 (обратите внимание на то, что центры затрат упорядочились по алфавиту).

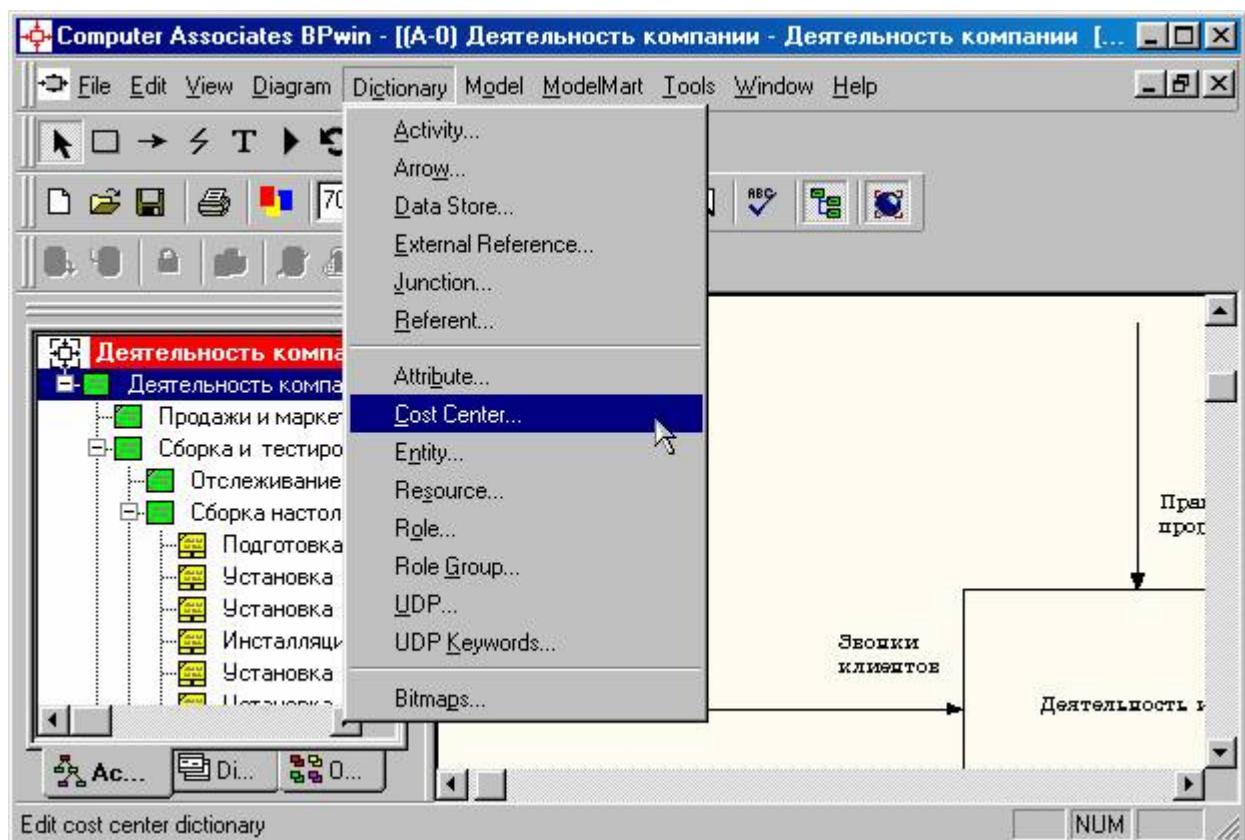


Рисунок 9.2- Выбор меню **Dictionary/Cost Center**

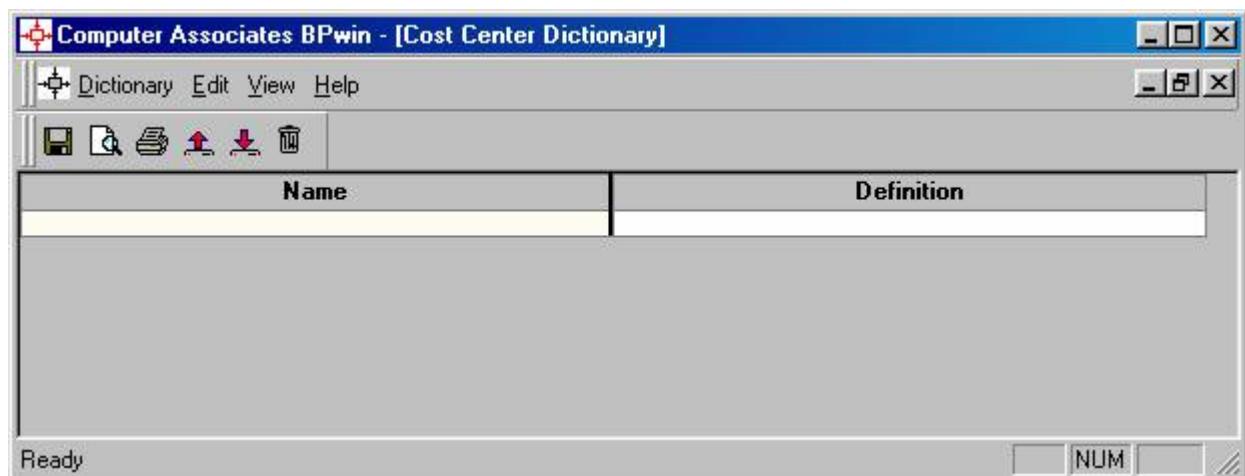


Рисунок 9.3 – Незаполненное окно **Cost Center Dictionary**

Таблица 9.1 - Центры затрат ABC

Центр затрат	Определение
Управление	Затраты на управление, связанные с составлением графика работ, формированием партий компьютеров, контролем над сборкой и тестированием
Рабочая сила	Затраты на оплату рабочих, занятых сборкой и тестированием компьютеров
Компоненты	Затраты на закупку компонентов

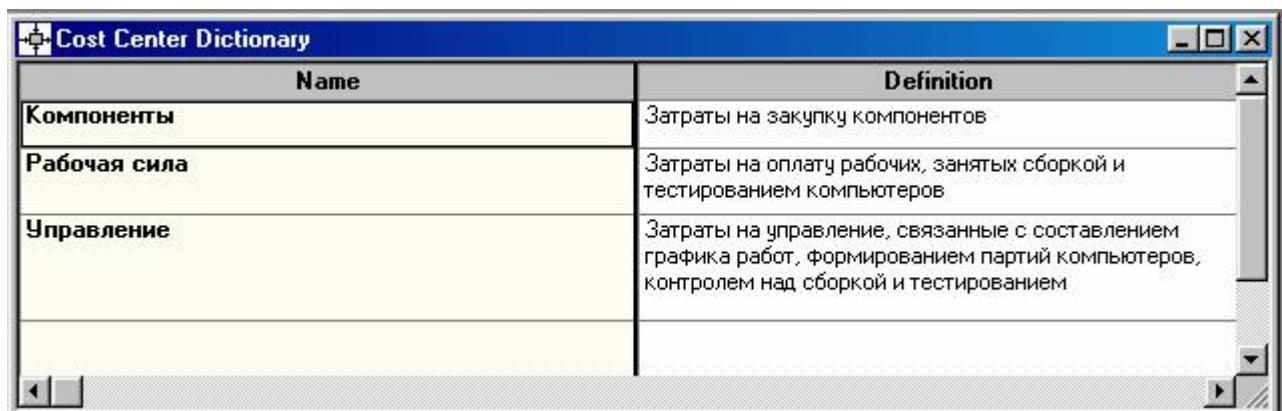


Рисунок 9.4- Заполненное окно Cost Center Dictionary

Для отображения стоимости каждой работы в нижнем левом углу прямоугольника перейдите в меню **Model/Model Properties** и во вкладке **Display** диалога **Model Properties** включите опцию **ABC Data** (рисунок 9.5).

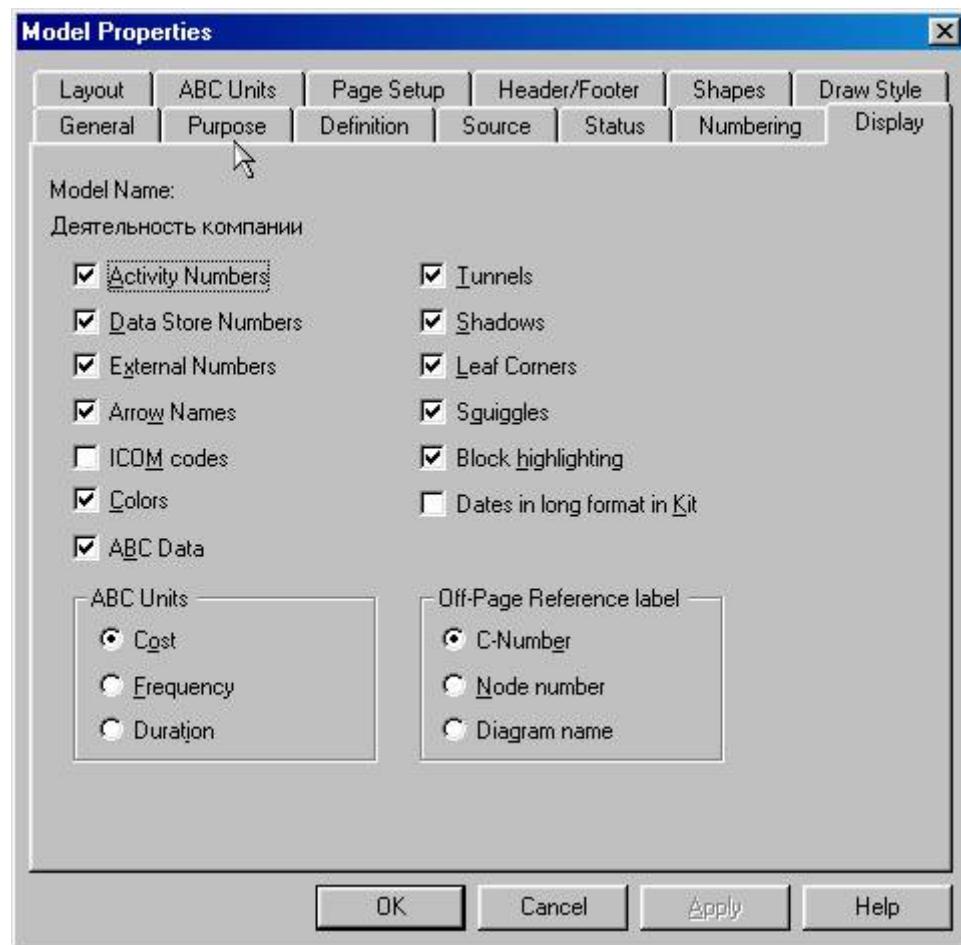


Рисунок 9.5 - Вкладка Display диалога Model Properties

Для отображения частоты или продолжительности работы переключите радиокнопки в группе **ABC Units**.

Для назначения стоимости работе "**Сборка настольных компьютеров**" следует на диаграмме A2 (рисунок 9.6) щелкнуть по ней правой кнопкой мыши и выбрать в контекстном меню **Cost** (рисунок 9.7).

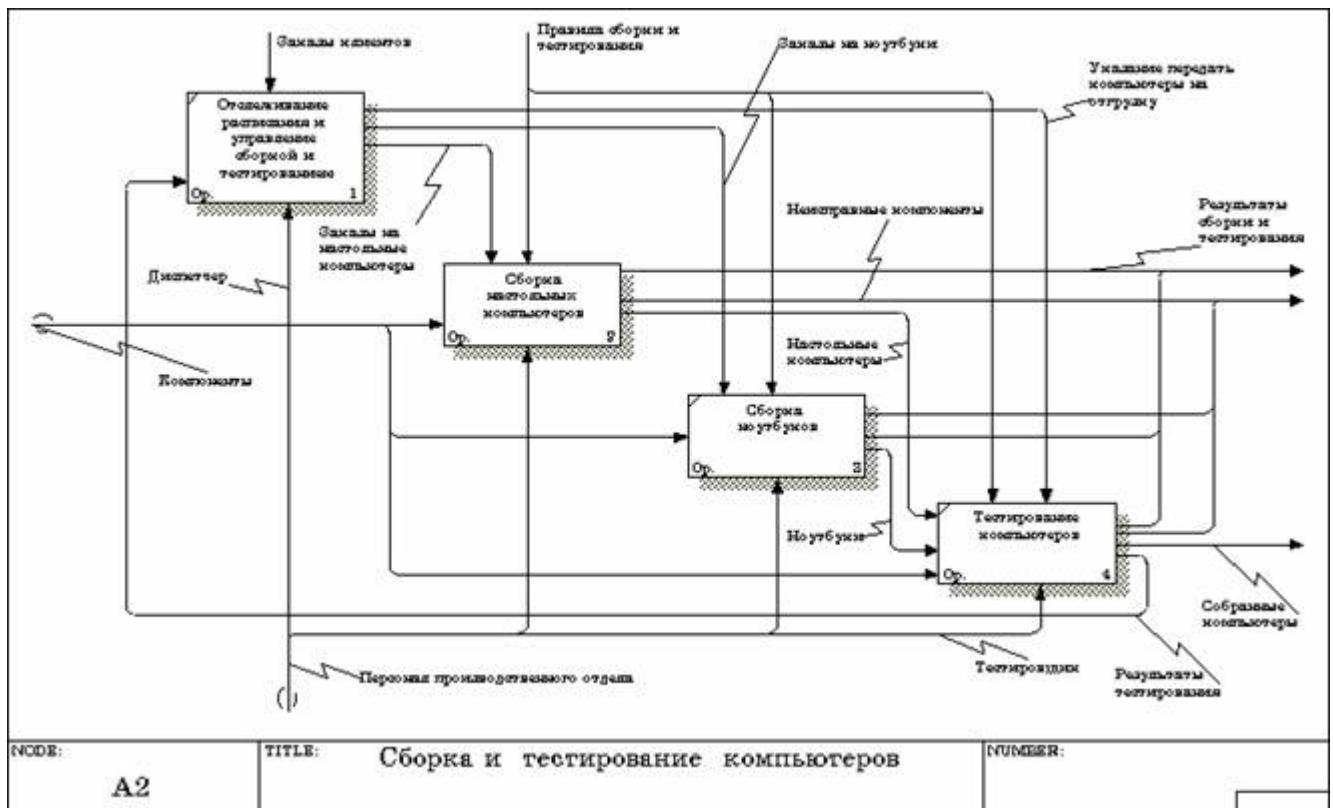


Рисунок 9.6 - Диаграмма А2

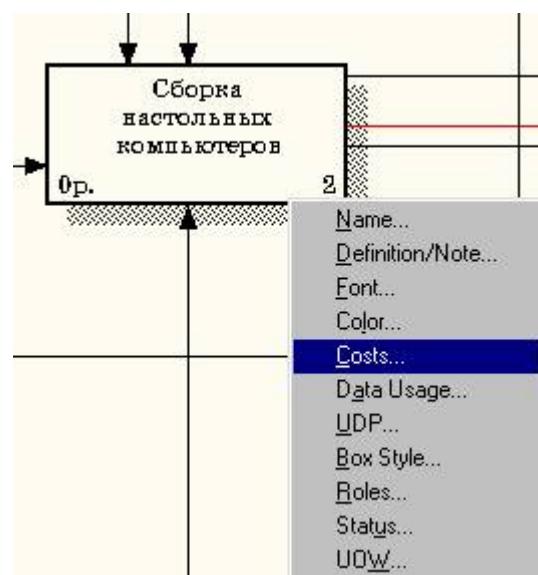


Рисунок 9.7 - Выбор в контекстном меню опции **Cost**

Откроется диалоговое окно **Activity Properties** (рисунок 9.10) в котором следует указать величины затрат (в рублях) на компоненты, рабочую силу, управление и временные характеристики работы – **Duration** (Продолжительность) и **Frequency** (Частоту) выполнения (см. таблицу 9.2).

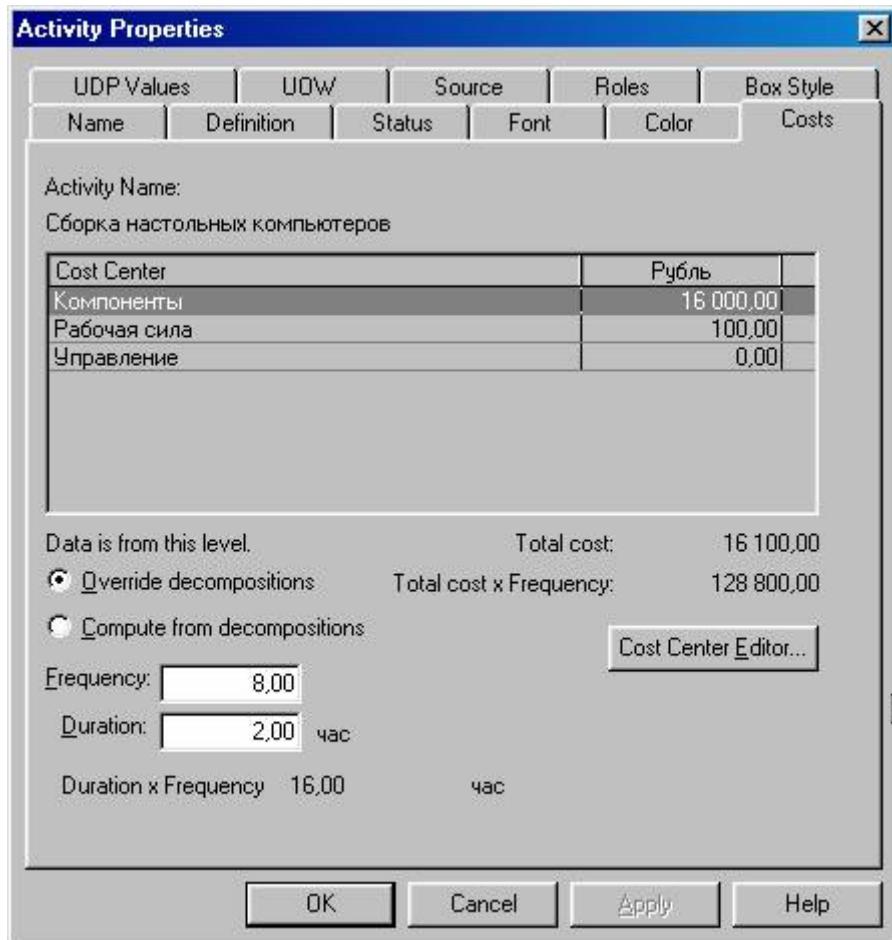


Рисунок 9.10 - Вкладка **Cost** диалога **Activity Properties**

3 Для работ на диаграмме А2 внесите параметры ABC (таблица 9.2).

Таблица 9.2 – Показатели стоимости работ на диаграмме А2

Activity Name	Cost Center	Cost Center Cost, руб.	Duration, час	Frequency
Отслеживание расписания и управление сборкой и тестированием	Управление	500,00	0,50	14,00
Сборка настольных компьютеров	Рабочая сила	100,00	2,00	8,00

	Компоненты	16000,00		
Сборка ноутбуков	Рабочая сила	140,00	4,00	6,00
	Компоненты	28000,00		
Тестирование компьютеров	Рабочая сила	60,00	1,00	14,00

Посмотрите результат - стоимость работы верхнего уровня (рисунок 9.11).



Рисунок 9.11 - Отображение стоимости в нижнем левом углу прямоугольника работы

- 4 Выбрав соответствующие опции меню (рисунок 9.12), сгенерируйте отчет **Activity Cost Report**.

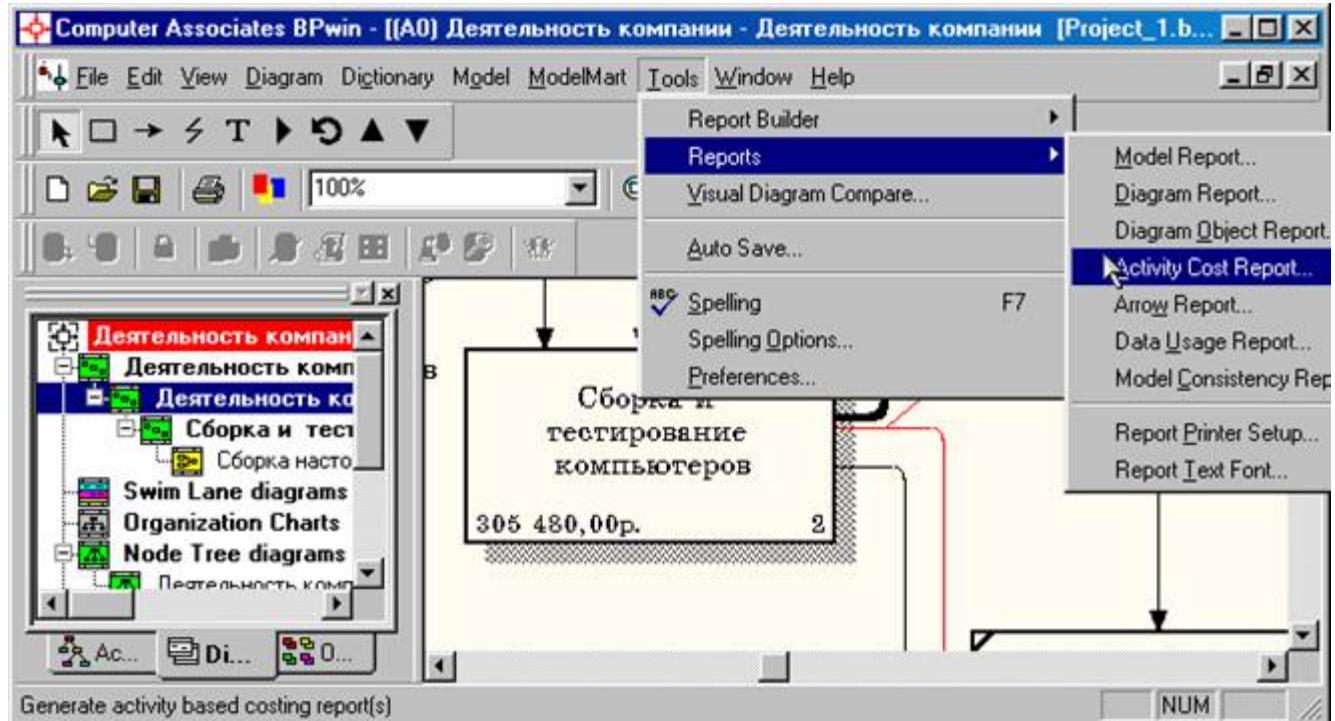


Рисунок 9.12 - Выбор опций меню для генерации отчета **Activity Cost Report**

В открывшемся диалоговом окне **Activity Based Costing Report** задайте параметры генерации отчета **Activity Cost Report** (рисунок 9.13).

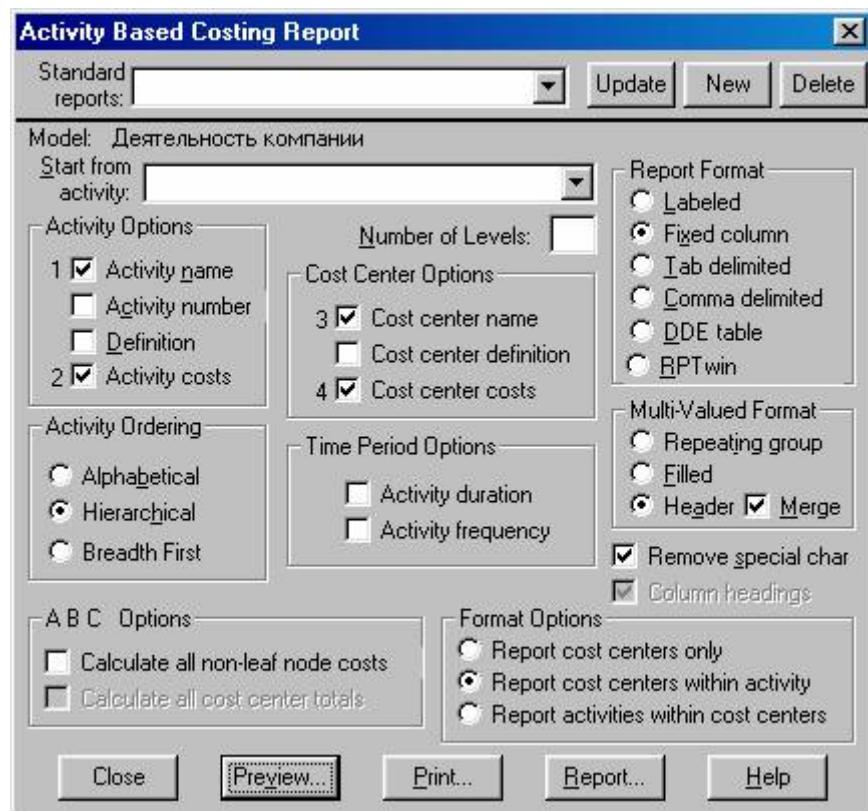


Рисунок 9.13 – Задание параметров генерации отчета **Activity Cost Report**

Activity Name	Activity Cost (Рубль)	Cost Center	Cost Center Cost (Рубль)
Деятельность компании	176 680,00	Компоненты	168 000,00
		Рабочая сила	1 680,00
		Управление	7 000,00
Продажи и маркетинг	0,00		
Сборка и тестирование компьютеров	176 680,00	Компоненты	168 000,00
		Рабочая сила	1 680,00
		Управление	7 000,00
Отслеживание расписания и управление сборкой и тестированием	500,00	Управление	500,00
Сборка ноутбуков	28 140,00	Компоненты	28 000,00
		Рабочая сила	140,00
Тестирование компьютеров	60,00	Рабочая сила	60,00

Рисунок 9.13 –Фрагмент отчета **Activity Cost Report**

Лабораторная работа № 5.

Построение диаграммы декомпозиции в нотации IDEF3

Цель работы:

- построить диаграмму декомпозиции в нотации IDEF3 одной из работ диаграмм IDEF0, построенных в предыдущих лабораторных работах

IDEF3 - методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. IDEF3 дает возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

Любая IDEF3-диаграмма может содержать работы, связи, перекрестки и объекты ссылок.

Работа (Unit of Work, activity). Изображается прямоугольником с прямыми углами (рис. 1) и имеет имя, выраженное отлагольным существительным,

обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы (например, «Изготовление изделия»). Все стороны работы равнозначны. В каждую работу может входить и выходить ровно по одной стрелке.



Рисунок 1. Работа IDEF3

Связи. Связи показывают взаимоотношения работ. Все связи в IDEF3 односторонние и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 возможны три вида связей:

Изображение стрелки	Название	Описание
→	Старшая (Precedence) стрелка	сплошная линия, связывающая единицы работ (UOW). Рисуется слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется
→→	Потоки объектов (Object Flow)	стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например когда объект порождается в одной работе и используется в другой
·---→	Стрелка отношения (Relational Link)	пунктирная линия, использующаяся для изображения связей между единицами работ (UOW), а также между единицами работ и объектами ссылок. Значение задается аналитиком отдельно для каждого случая

Перекрестки (Junction). Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления.

Типы перекрестков:

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Асинхронное «И» (Asynchronous AND)	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Синхронное «И» (Synchronous AND)	Все предшествующие процессы завершены одновременно	Все следующие процессы запускаются одновременно
	Асинхронное «ИЛИ» (Asynchronous OR)	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Синхронное «ИЛИ» (Synchronous OR)	Один или несколько предшествующих процессов завершены одновременно	Один или несколько следующих процессов запускаются

			одновременно
	Исключающее «ИЛИ» XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Объект ссылки. Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой. Они используются в модели для привлечения внимания читателя к каким-либо важным аспектам модели. При внесении объектов ссылок помимо имени следует указывать тип объекта ссылки (рис. 2).

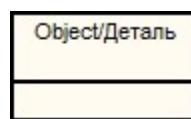


Рисунок 2. Объект ссылки

В данной лабораторной работе необходимо одну из работ, находящихся на диаграммах IDEF0, рассмотреть детально с помощью методологии IDEF3. При декомпозиции работы IDEF0 (и DFD) нужно учитывать, что стрелки на диаграммах IDEF0 или DFD означают потоки информации или объектов, передаваемых от одной работы к другой. На диаграммах IDEF3 стрелки могут показывать только последовательность выполнения работ, т.е. они имеют другой смысл, чем стрелки IDEF0 или DFD. Поэтому при декомпозиции работы IDEF0 или DFD в диаграмму IDEF3 стрелки не мигрируют на нижний уровень. Если необходимо показать на дочерней диаграмме IDEF3 те же объекты, что и на родительских диаграммах IDEF0 или DFD, необходимо использовать объекты ссылки.

Проведем декомпозицию работы *Сборка настольных компьютеров* диаграммы А3 "Сборка и тестирование компьютеров". Данная работа начинает

выполняться, когда поступают заказы на сборку. Первым действием проверяется наличие необходимых для сборки комплектующих и заказ со склада отсутствующих. Далее комплектующие подготавливаются для последующей сборки (освобождение от упаковки, снятие заглушек и т.п.). Следующим шагом начинается непосредственно сам процесс сборки: установка материнской платы в корпус и процессора на материнскую плату, установка ОЗУ и винчестера. Данные действия выполняются всегда, независимо от конфигурации компьютера. Далее по желанию клиента могут быть установлены некоторые дополнительные комплектующие - DVD привод, ТВ-тюнер, кардридер. На этом сборка компьютера завершается. Следующим шагом идет установка операционной системы. По желанию клиента также может быть установлено дополнительное программное обеспечение. Последним действием составляется отчет о проделанной работе.

Выделим работу *Сборка настольных компьютеров* диаграммы А3 "Сборка и тестирование компьютеров", нажмем на кнопку "Go to Child Diagram" панели инструментов и выберем нотацию IDEF3. Дочерние работы всегда можно добавить на диаграмму в процессе ее построения, поэтому число дочерних работ оставим по умолчанию. При создании дочерней диаграммы BPWin переносит граничные стрелки родительской работы, их необходимо удалить и заменить на объекты ссылок. Заменим стрелки "*Заказы на настольные компьютеры*", "*Необходимые комплектующие*", "*Список необходимых комплектующих*", "*Настольные компьютеры*" и "*Результаты сборки*" на объекты ссылок - кнопка "Referent" на панели инструментов, в появившемся окне выбрать переключатель "Arrow" и выбрать из списка нужное название (рис. 3):

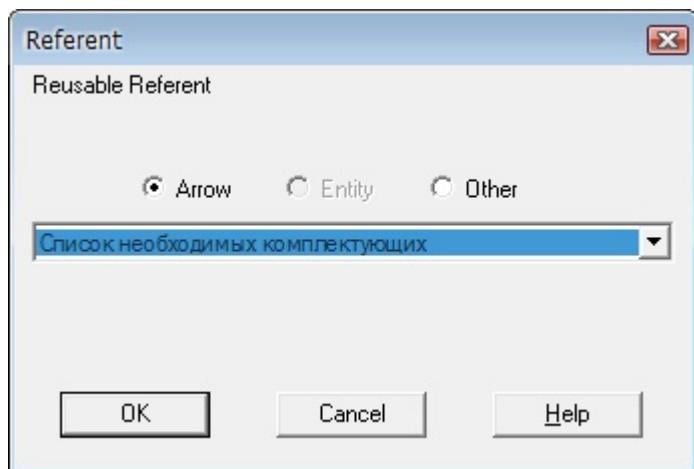


Рисунок 3. Добавление объекта ссылки

Далее начинаем располагать на диаграмме работы, отражающие указанные выше действия, выполняемые при сборке компьютеров. Итоговая диаграмма декомпозиции работы в нотации IDEF3 имеет вид:

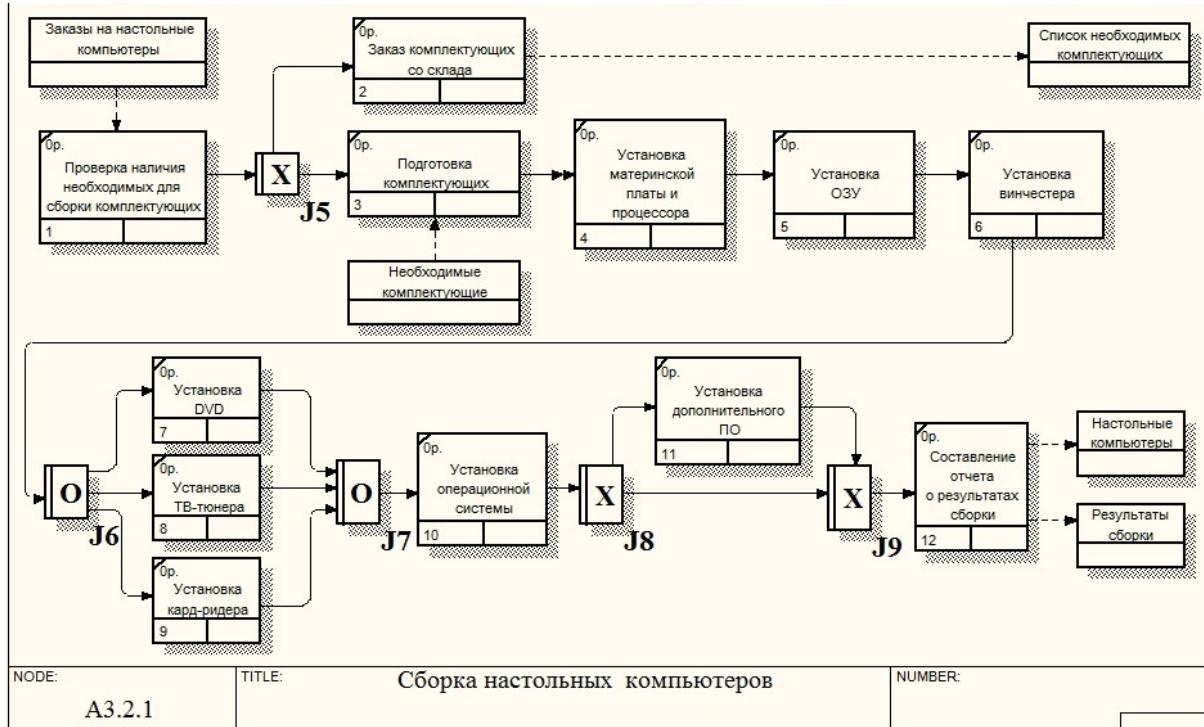


Рисунок 4. Диаграмма декомпозиции

Рассмотрим основные особенности этой диаграммы. После проверки наличия необходимых для сборки комплектующих возможно одно из двух действий - или заказ со склада недостающих комплектующих, или, если все комплектующие в наличии, их подготовка. Поэтому мы поставили перекресток разветвления типа "Исключающее ИЛИ". Работы "Подготовка комплектующих" и "Установка материнской платы и процессора" соединены связью "Поток объектов". Тем самым мы показываем, что между этими работами передаются объекты. Все последующие работы соединяются связями "старшая стрелка", поскольку они только показывают последовательность действий над одними и тем же объектами.

После установки винчестера возможна установка DVD привода, ТВ-тюнера, кард-ридерса или любая их комбинация. Поэтому мы поставили перекресток разветвления типа "Асинхронное ИЛИ". Такой же перекресток стоит и после

завершения этих работ. Далее после установки операционной системы может быть установлено дополнительное ПО, или же сразу формируется отчет, поэтому мы поставили перекресток разветвления типа "Исключающее ИЛИ". За перекрестком разветвления типа "Исключающее ИЛИ" может следовать только такой же перекресток слияния, поэтому перед работой "*Составление отчета о результатах сборки*" мы поставили такой же.

Содержание отчета:

- краткое описание декомпозируемой работы
- диаграмма декомпозиции

Лабораторная работа № 6.

Построение диаграммы декомпозиции в нотации DFD

Цель работы:

- построить диаграмму декомпозиции в нотации DFD одной из работ диаграмм IDEF0, построенных в предыдущих лабораторных работах

Диаграммы потоков данных (Data flow diagram, DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет моделируемую систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. Главная цель DFD - показать, как каждая работа преобразует свои входные данные в выходные, а также выявить отношения между этими работами.

Любая DFD-диаграмма может содержать работы, внешние сущности, стрелки (потоки данных) и хранилища данных.

Работы. Работы изображаются прямоугольниками с закругленными углами (рис. 1), смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0. Все стороны работы равнозначны. В каждую работу может входить и выходить по несколько стрелок.



Рисунок 1. Работа в DFD

Внешние сущности. Внешние сущности изображают входы в систему и/или выходы из нее. Одна внешняя сущность может одновременно предоставлять входы (функционируя как поставщик) и принимать выходы (функционируя как получатель). Внешняя сущность представляет собой материальный объект, например заказчики, персонал, поставщики, клиенты, склад. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что они находятся за пределами границ анализируемой системы. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы (рис. 2).



Рисунок 2. Внешняя сущность в DFD

Стрелки (потоки данных). Стрелки описывают движение объектов из одной части системы в другую (отсюда следует, что диаграмма DFD не может иметь граничных стрелок). Поскольку все стороны работы в DFD равнозначны, стрелки могут начинаться и заканчиваться на любой стороне прямоугольника. Стрелки могут быть двунаправлены.

Хранилище данных. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое (рис. 3). Хранилище данных - это абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь, причем способы помещения и извлечения могут быть любыми. Оно в общем случае является прообразом будущей базы данных, и описание хранящихся в нем данных должно соответствовать информационной модели (Entity-Relationship Diagram).

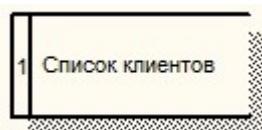


Рисунок 3. Хранилище данных в DFD

Декомпозиция работы IDEF0 в диаграмму DFD. При декомпозиции работы IDEF0 в DFD необходимо выполнить следующие действия:

- удалить все граничные стрелки на диаграмме DFD;
- создать соответствующие внешние сущности и хранилища данных;
- создать внутренние стрелки, начинающиеся с внешних сущностей вместо граничных стрелок;
- стрелки на диаграмме IDEF0 затоннелировать

Строго придерживаться правил нотации DFD не всегда удобно, поэтому BPWin позволяет создавать в DFD диаграммах граничные стрелки.

Построение диаграммы декомпозиции. Проведем декомпозицию работы *Отгрузка и снабжение* диаграммы А0 "Деятельность предприятия по сборке и продаже компьютеров и ноутбуков". В этой работе мы выделили следующие дочерние работы:

- снабжение необходимыми комплектующими - занимается действиями, связанными с поиском подходящих поставщиков и заказом у них необходимых комплектующих
- хранение комплектующих и собранных компьютеров
- отгрузка готовой продукции - все действия, связанные с упаковкой, оформлением документации и собственно отгрузкой готовой продукции

Выделим работу *Отгрузка и снабжение* диаграммы А0 "Деятельность предприятия по сборке и продаже компьютеров и ноутбуков", нажмем на кнопку "Go to Child Diagram" панели инструментов и выберем нотацию DFD. При создании дочерней диаграммы BPWin переносит граничные стрелки родительской работы, их необходимо удалить и заменить на внешние сущности. Стрелки механизмов, стрелки управления "Правила и процедуры", "Управляющая информация" и стрелку выхода "Отчеты" на дочерней диаграмме задействованы не будут, чтобы не загромождать диаграмму менее существенными деталями. Остальные стрелки заменим на внешние сущности - кнопка "External Reference Tool" на панели инструментов, в появившемся окне выбрать переключатель "Arrow" и выбрать из списка нужное название (рис. 4):

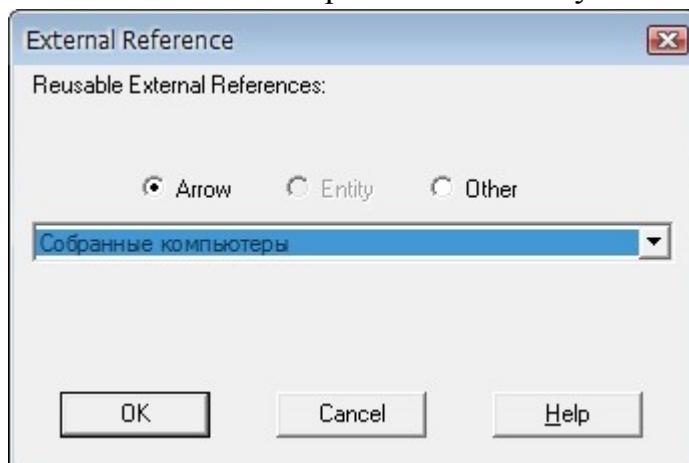


Рисунок 4. Добавление внешней сущности

Далее разместим дочерние работы, свяжем их со внешними сущностями и между собой (рис. 5):

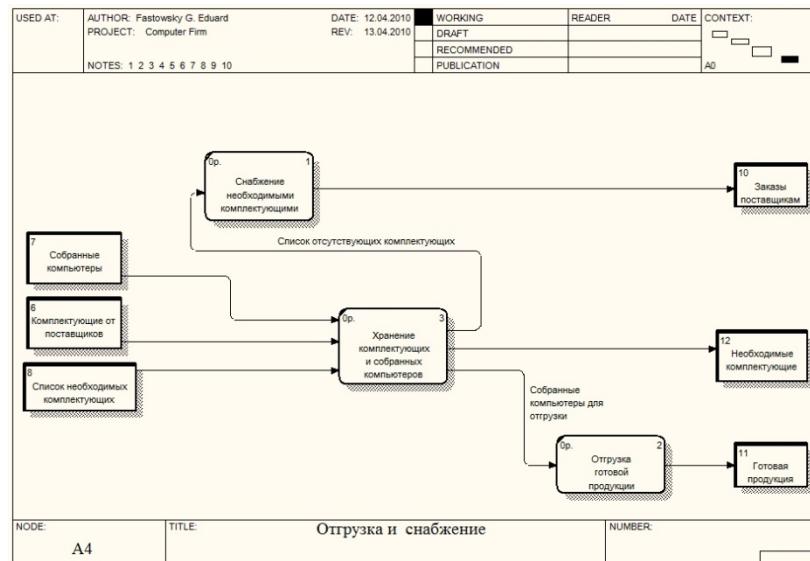


Рисунок 5. Работы и внешние сущности

Центральной здесь является работа "Хранение комплектующих и собранных компьютеров". На ее вход поступают собранные компьютеры и полученные от поставщиков комплектующие, а также список необходимых для сборки компьютеров комплектующих. Выходом этой работы будут необходимые комплектующие (если они есть в наличии), список отсутствующих комплектующих, передаваемый на вход работы "Снабжение необходимыми комплектующими" и собранные компьютеры, передаваемые на отгрузку. Выходами работ "Снабжение необходимыми комплектующими" и "Отгрузка готовой продукции" будут, соответственно, заказы поставщикам и готовая продукция.

Следующим шагом необходимо определить, какая информация необходима для каждой работы, т.е. необходимо разместить на диаграмме хранилища данных (рис. 6).

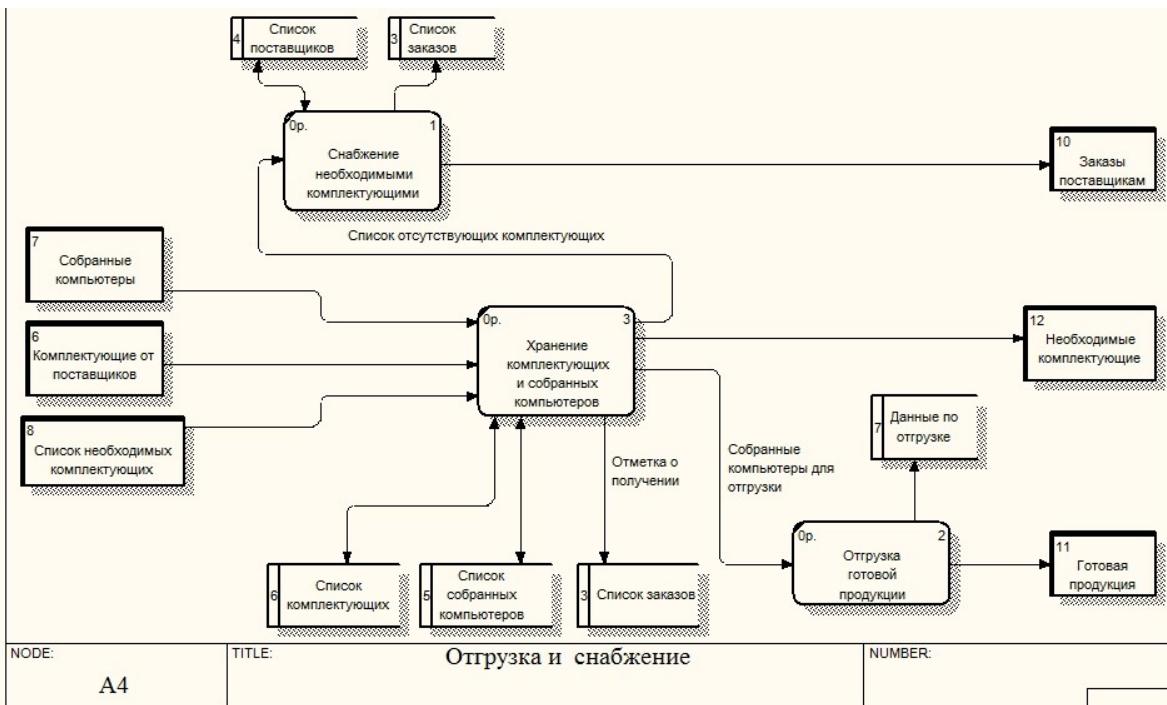


Рисунок 6. Итоговая диаграмма декомпозиции

Работа "Снабжение необходимыми комплектующими" работает с информацией о поставщиках и с информацией о заказах, сделанных у этих поставщиков. Стрелка, соединяющая работу и хранилище данных "Список поставщиков" двунаправленная, т.к. работа может как получать информацию о имеющихся поставщиках, так и вносить данные о новых поставщиках. Стрелка, соединяющая работу с хранилищем данных "Список заказов" односторонняя, т.к. работа только вносит информацию о сделанных заказах.

Работа "Хранение комплектующих и собранных компьютеров" работает с информацией о получаемых и выдаваемых комплектующих и собранных компьютерах, поэтому стрелки, соединяющие работу с хранилищами данных "Список комплектующих" и "Список собранных компьютеров" двунаправленные. Также эта работа при получении комплектующих должна делать отметку о том, что заказ поставщикам выполнен. Для этого она связана с хранилищем данных "Список заказов" односторонней стрелкой. Обратите внимание, что на DFD диаграммах одно и тоже хранилище данных может дублироваться.

Наконец, работа "Отгрузка готовой продукции" должна хранить информацию по выполненным отгрузкам. Для этого вводится соответствующее хранилище данных - "Данные по отгрузке".

Последним действием необходимо стрелки родительской работы затуннелировать (рис. 7):

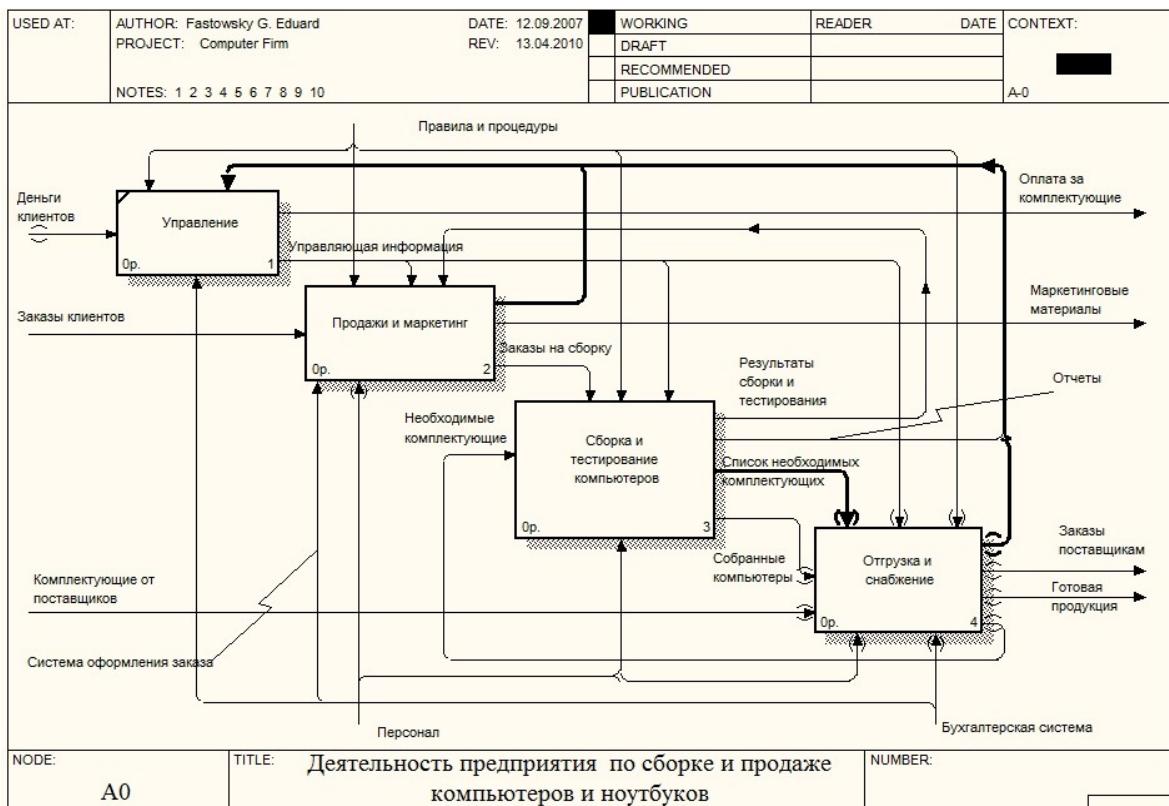


Рисунок 7. Диаграмма IDEF0 с затуннелированными стрелками работы "Отгрузка и снабжение"

Содержание отчета:

- краткое описание декомпозируемой работы
- диаграмма декомпозиции

Лабораторная работа № 7.

Построение FEO диаграмм и диаграмм дерева узлов

Цель работы:

- построить FEO диаграмму для одной из имеющихся диаграмм
- построить диаграмму дерева узлов

FEO диаграммы

FEO (For Exposition Only) диаграммы (другое название - диаграммы только для экспозиции, описания) используются для иллюстрации альтернативной точки зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. FEO диаграммы позволяют нарушить любое синтаксическое правило, поскольку эти диаграммы - фактически обычные картинки - копии стандартных диаграмм. Например, работа на FEO диаграмме может не иметь стрелок выхода или управления. AllFusion Process Modeler позволяет также строить FEO диаграммы для диаграмм в нотации DFD.

Для построения FEO диаграммы необходимо выбрать пункт меню *Diagram -> Add FEO Diagram* и в появившемся окне выбрать диаграмму, на базе которой будет строиться FEO диаграмма (рис. 1).

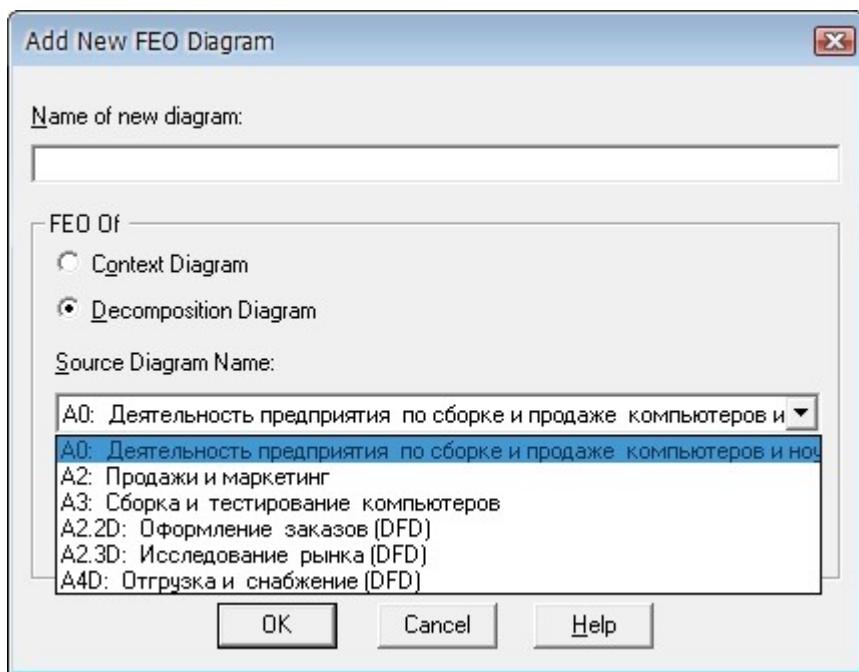


Рисунок 1. Добавление FEO диаграммы

Созданная диаграмма будет точной копией родительской диаграммы и будет иметь номер, равный номеру родительской диаграммы + буква F. После создания диаграммы ее можно изменять. При этом изменения не будут влиять на родительскую диаграмму.

Для просмотра списка имеющихся FEO диаграмм нужно выбрать в *Обозревателе Модели* (Model Explorer) вкладку *Diagrams* (рис.2).

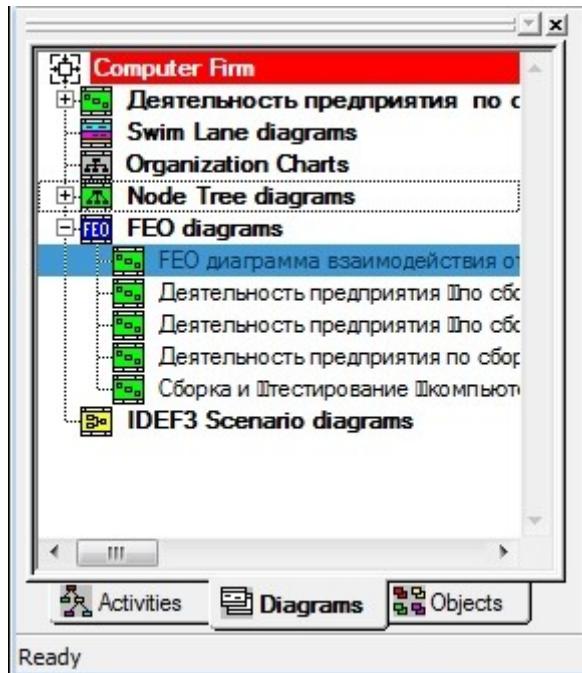


Рисунок 2. Просмотр списка имеющихся FEO диаграмм

Построим FEO диаграмму для диаграммы декомпозиции второго уровня А0 "Деятельность предприятия по сборке и продаже компьютеров и ноутбуков" и покажем на ней как дочерние работы связаны между собой. Для этого создаем диаграмму, как показано выше, и удаляем на ней все граничные стрелки. Итоговая FEO диаграмма показана на рис.3:

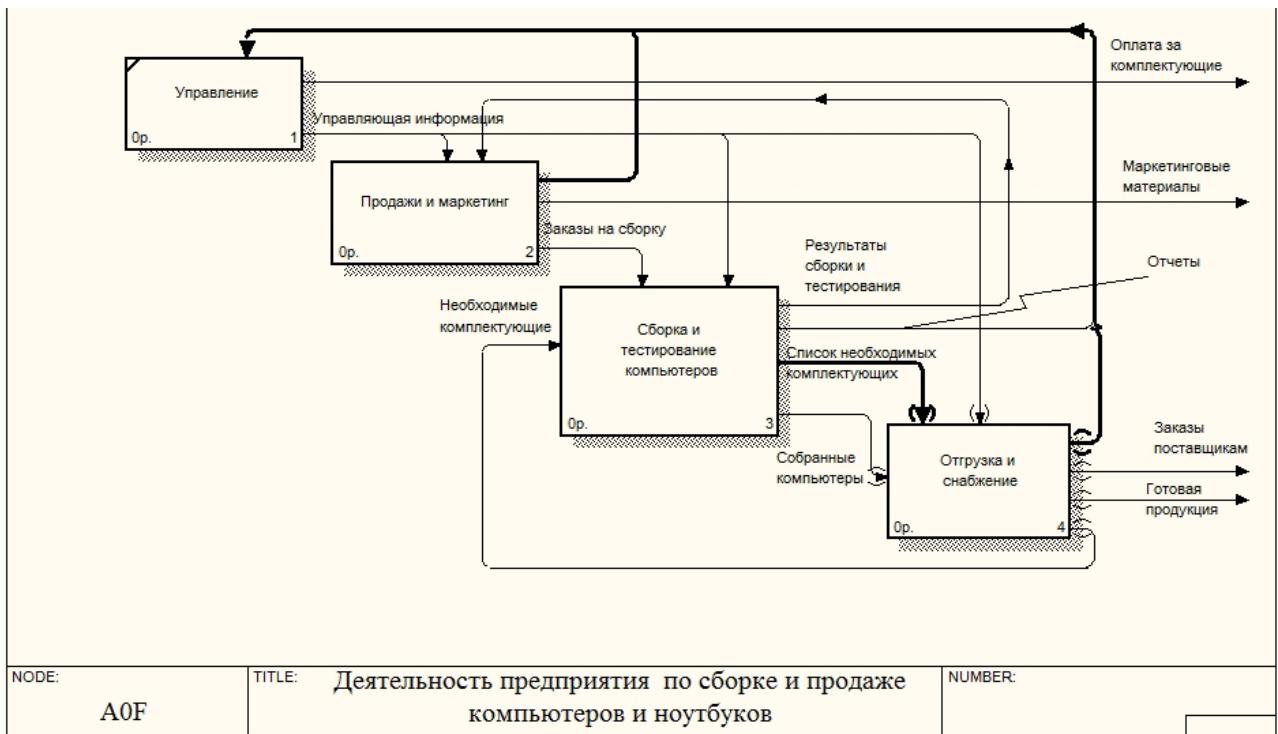


Рисунок 3. FEO диаграмма

Диаграммы дерева узлов

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. В одной модели диаграмм дерева узлов может быть множество, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

Для построения диаграммы дерева узлов необходимо выбрать пункт меню *Diagram -> Add Node Tree*. Появляется мастер, с помощью которого диаграмма будет создана. На первом шаге (рис.4) задается имя диаграммы дерева узлов, узел верхнего уровня и глубина дерева. Имя дерева узлов по умолчанию совпадает с именем работы верхнего уровня, а номер диаграммы генерируется автоматически как номер узла верхнего уровня + буква N.

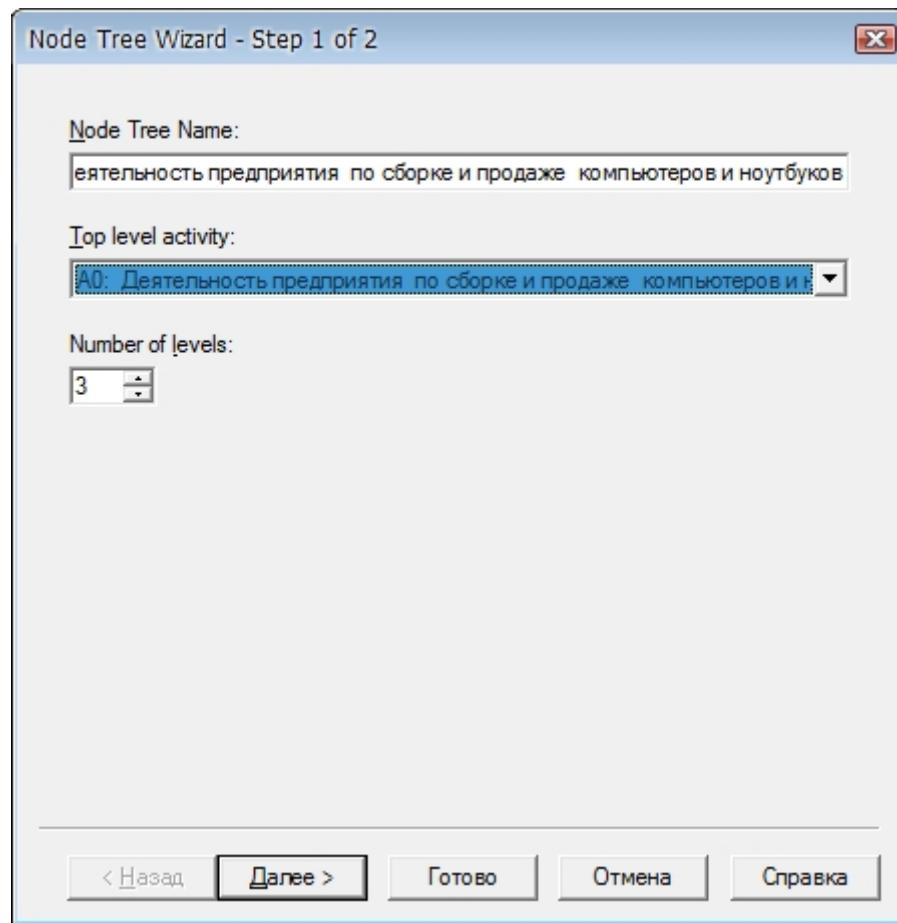


Рисунок 4. Создание диаграммы дерева узлов. Шаг 1

На втором шаге мастера (рис.5) задаются свойства диаграммы дерева узлов.

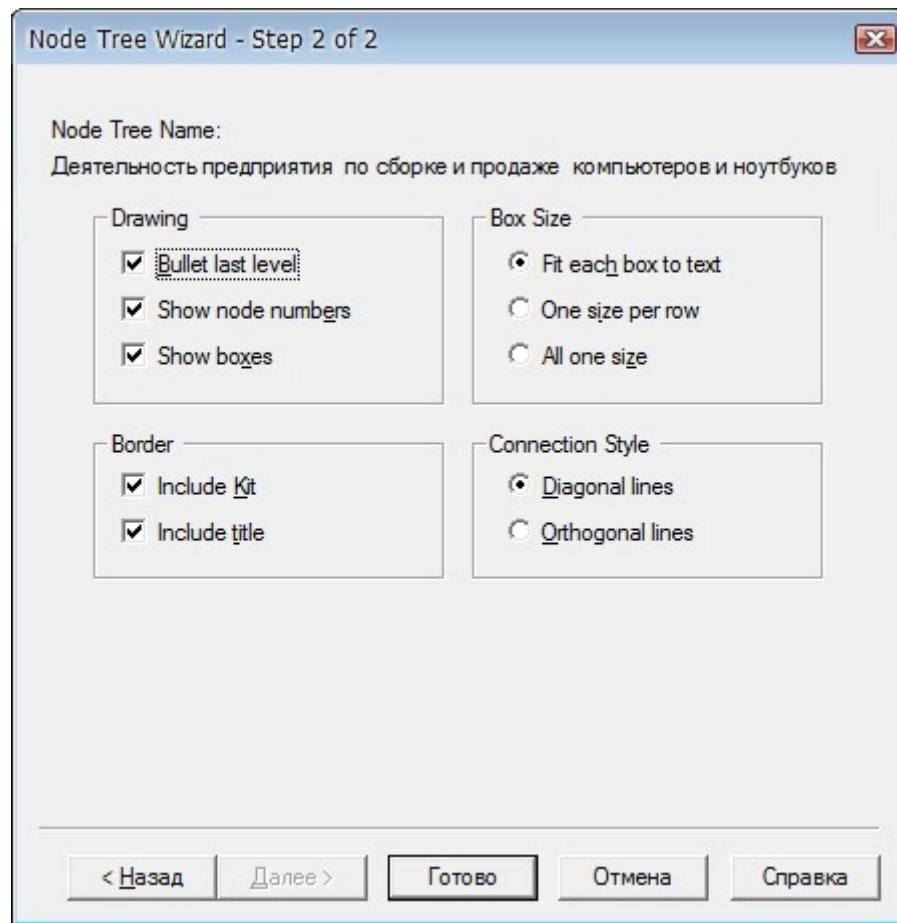


Рисунок 5. Создание диаграммы дерева узлов. Шаг 2

По умолчанию нижний уровень декомпозиции показывается в виде списка, остальные работы - в виде прямоугольников. Если необходимо отобразить все дерево в виде прямоугольников, то следует снять галочку возле опции "Bullet last level". Список всех созданных диаграмм дерева узлов можно посмотреть в *Обозреватели Модели*.

Диаграмма дерева узлов для всех узлов модели показана на рис. 6:

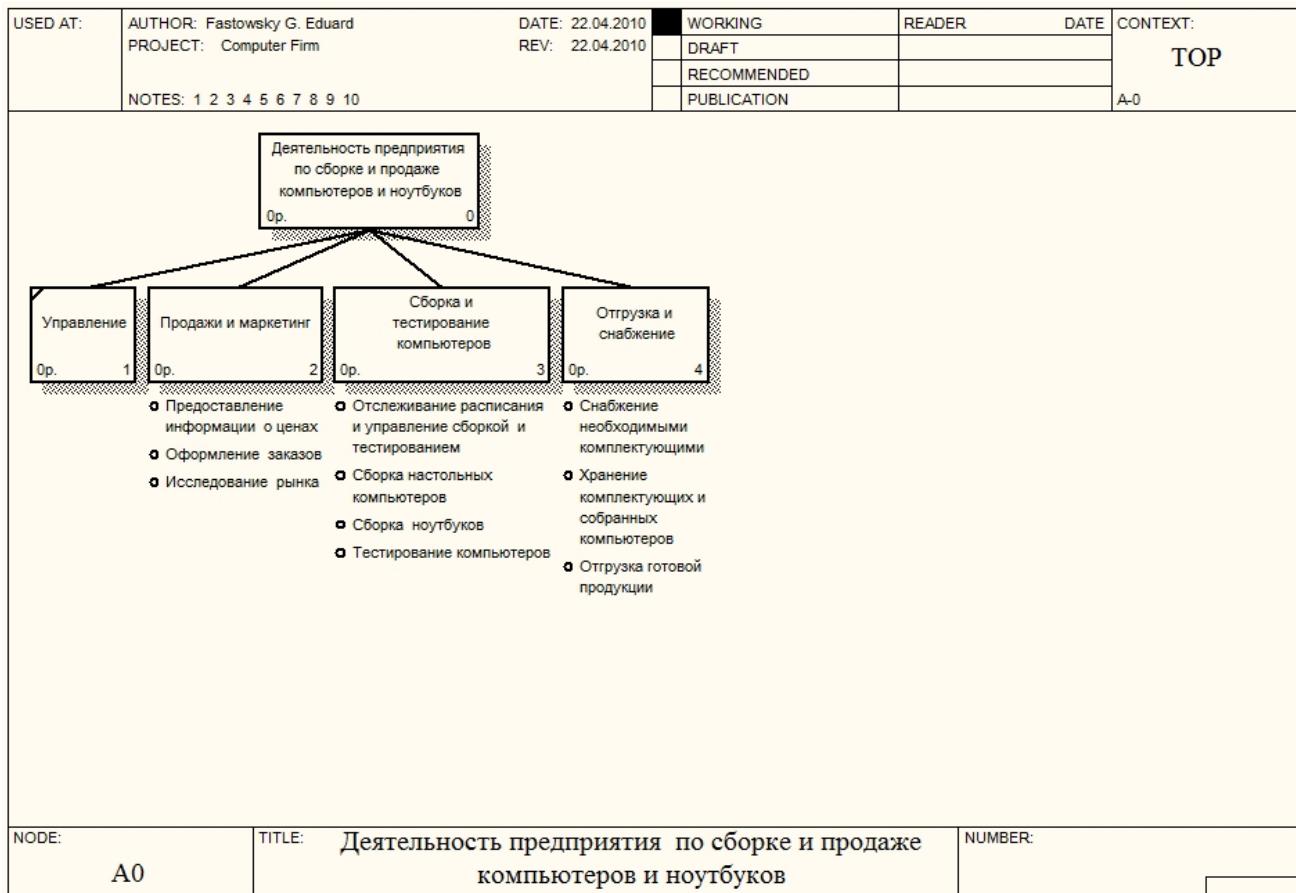


Рисунок 6. Диаграммы дерева узлов

Содержание отчета:

- FEO диаграмма
- диаграмма дерева узлов

Лабораторная работа № 8.

Основы работы с программным продуктом

AllFusion ERwin Data Modeler

CA ERwin Data Modeler (далее ERwin) - CASE-средство для проектирования и документирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных.

Работа с программой начинается с создания новой модели, для которой нужно указать тип и целевую СУБД (рис.1).

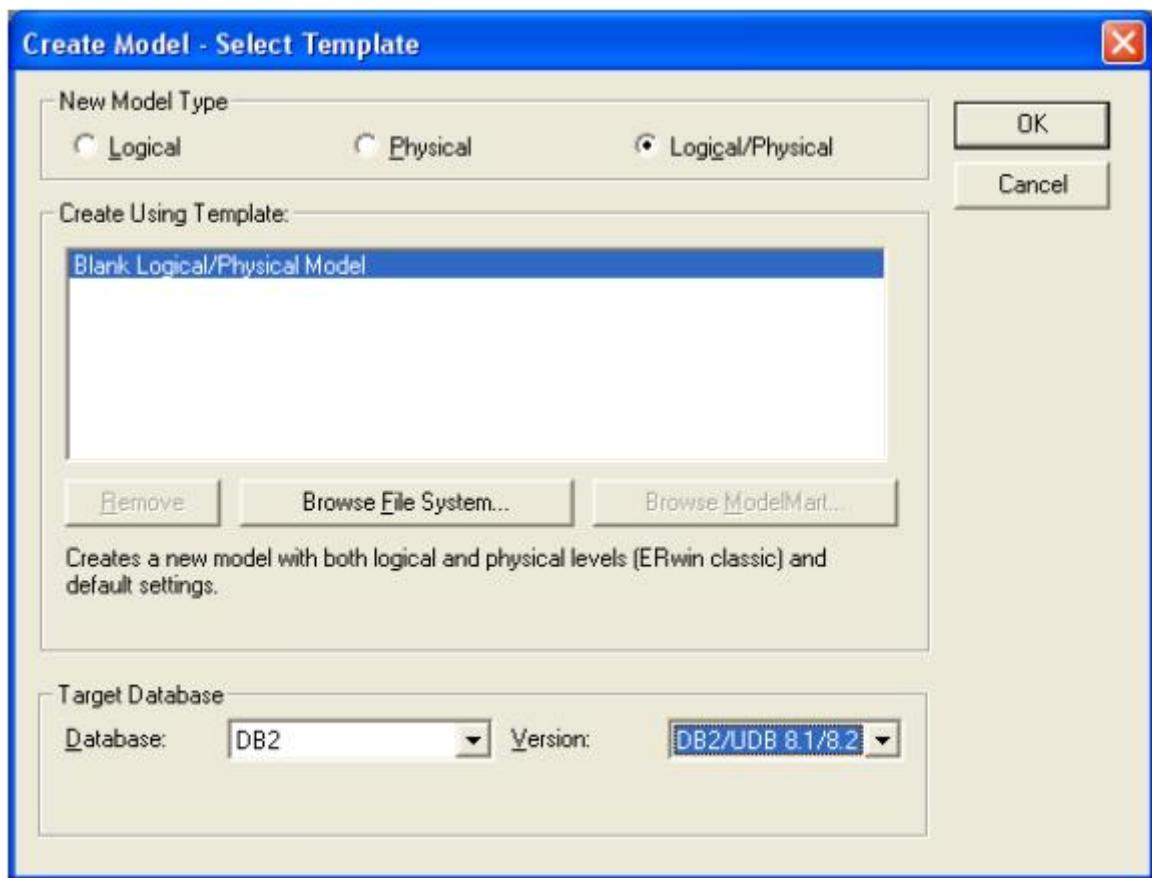


Рисунок 1.Создание новой модели

ERwin позволяет создавать логическую, физическую модели и модель, совмещающую логический и физический уровни.

Логический уровень - это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире (например "Постоянный клиент", "Отдел" или "Заказ").

Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физический уровень зависит от конкретной СУБД. В физической модели содержится информация о всех объектах БД. Физическая модель зависит от конкретной реализации СУБД. Одной и той же логической модели могут соответствовать несколько разных физических моделей.

На логическом уровне ERwin поддерживает две нотации (IE и IDEF1X), на физическом - три (IE, IDEF1X и DM). Далее будет рассматриваться работа с ERwin в нотации IDEF1X.

Переключение между логической и физической моделями данных осуществляется через список выбора на стандартной панели (рис.2).

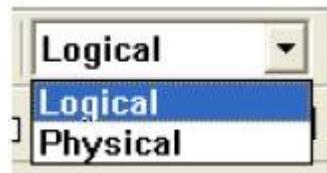


Рисунок 2.Переключение между уровнями

Примечание. В созданной модели с настройками по умолчанию некорректно отображаются русские символы. Чтобы устранить этот недостаток, необходимо подкорректировать используемые в модели шрифты. Для этого необходимо зайти в меню Format -> Default Fonts & Colors, последовательно пройтись по всем вкладкам, в качестве шрифта выбрав любой шрифт, название которого заканчивается на CYR (например, Arial CYR), и выставив переключатель Apply To в значение All Objects.

Логический уровень модели данных

Для создания на логическом уровне сущностей и связей между ними предназначена панель *Toolbox*:



Рисунок 3.Панель Toolbox

Вид кнопки	Назначение кнопки
	Создание новой сущности. Для этого нужно щелкнуть по кнопке и затем по свободному месту на модели
	Создание категории. Для установки категориальной связи нужно щелкнуть по кнопке, далее - по сущности-родителю, и затем - по сущности-потомку.
	Создание идентифицирующей связи. Для связывания двух сущностей нужно щелкнуть по кнопке, далее - по сущности-родителю, затем - по сущности-потомку.
	Создание связи "многие ко многим"
	Создание неидентифицирующей связи

После создания сущности ей нужно задать атрибуты. Для этого нужно дважды щелкнуть по ней или в контекстном меню выбрать пункт *Attributes* (рис.4).

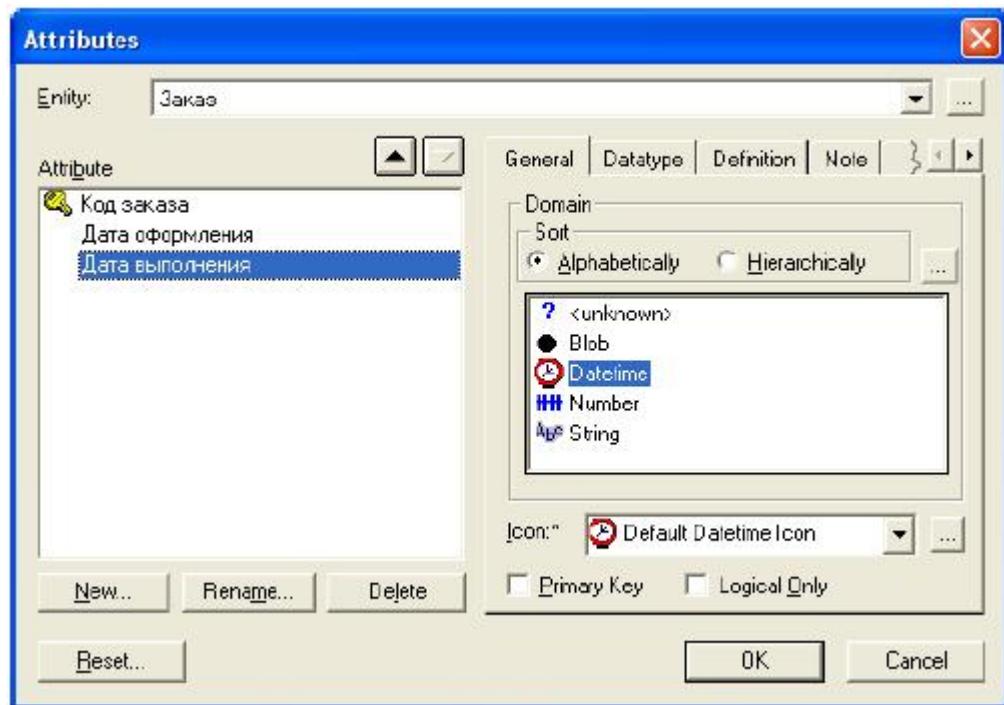


Рисунок 4.Окно атрибутов выбранной сущности

В появившемся окне можно просмотреть и отредактировать информацию о созданных атрибутах, создать новые. Здесь же задается первичный ключ. Для создания нового атрибута следует нажать кнопку *New*. В появившемся окне можно выбрать тип атрибута (BLOB, дата/время, число, строка), задать имя атрибута (*Attribute Name*) и имя столбца (*Column Name*), который будет соответствовать атрибуту на физическом уровне (рис.5).

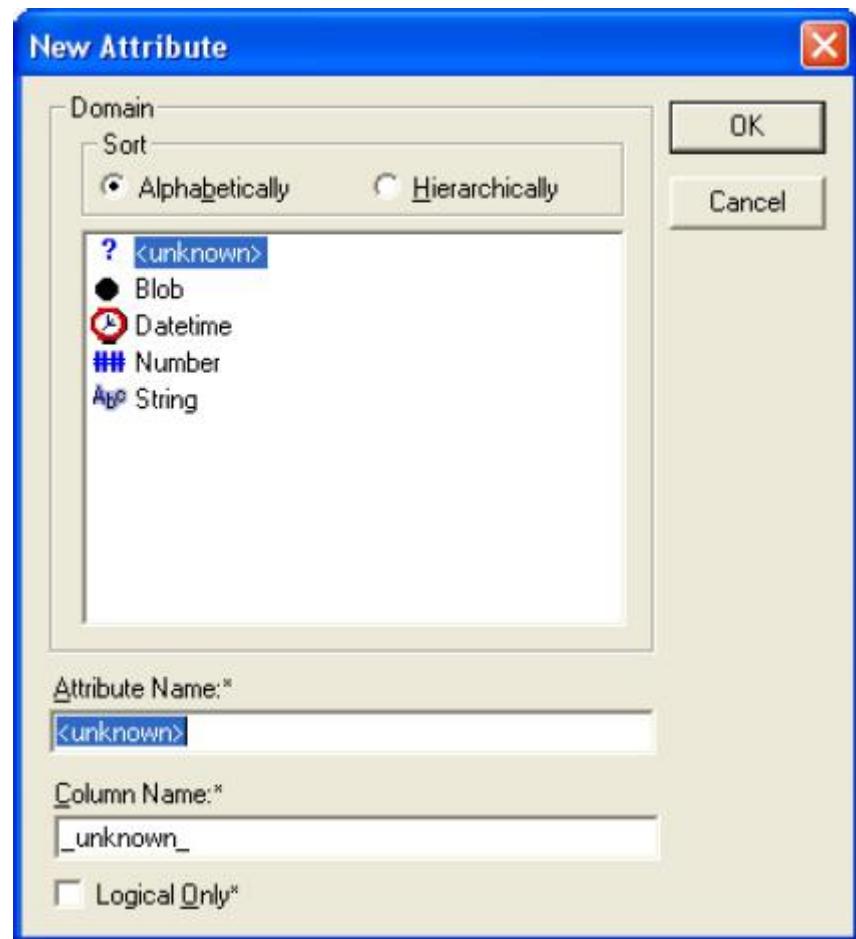


Рисунок 5.Окно создания атрибута

После создания сущностей создаются связи между ними. При создании идентифицирующей связи атрибуты, составляющие первичный ключ сущности-родителя, мигрируют в состав первичного ключа сущности-потомка, при создании неидентифицирующей связи - просто в состав атрибутов сущности-потомка. Задать свойства связи или поменять ее тип можно дважды щелкнув по ней или выбрав в контекстном меню пункт *Relationship Properties* (рис. 6). Здесь во вкладке *General* можно задать имя связи (в направлении родитель-потомок и потомок-родитель), мощность связи (ноль, один или больше; один и больше (P); ноль или один (Z); точно (конкретное число)), поменять тип связи. Во вкладке *RI Action* можно задать ограничения целостности.

Пример логической модели базы данных приведен на рис. 7.

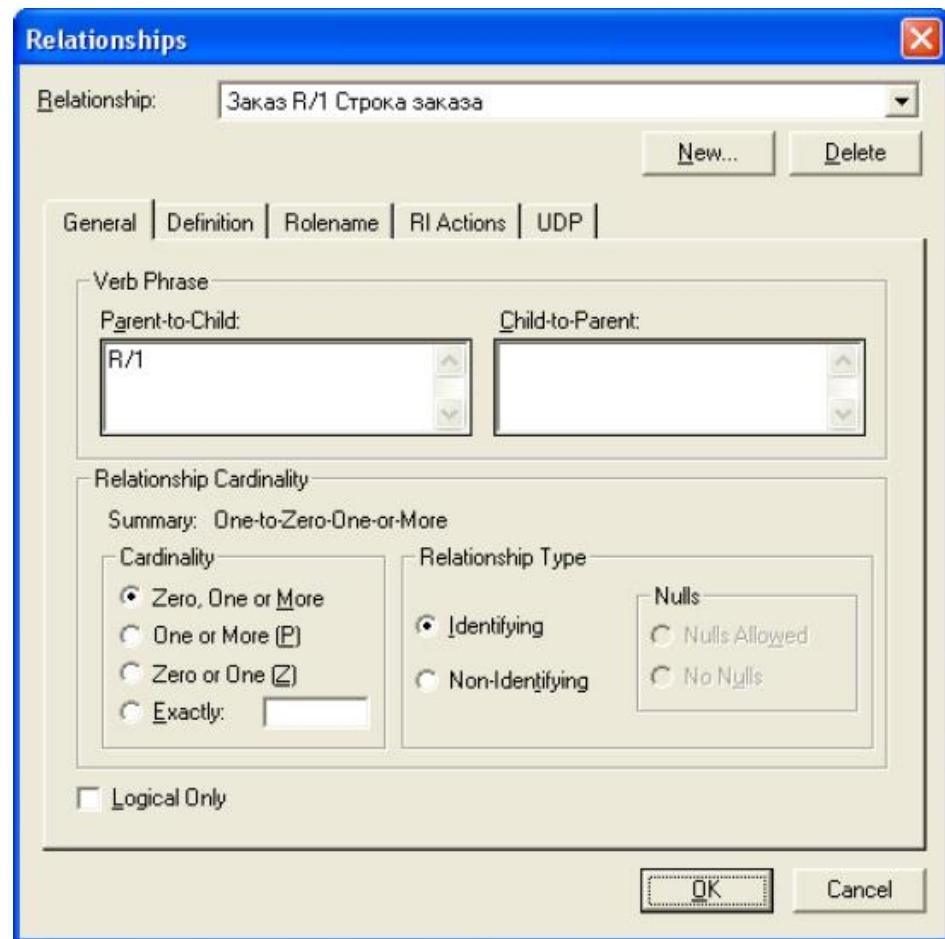


Рисунок 6.Окно свойств связи



Рисунок 7.Пример логической схемы БД

Физический уровень модели данных

При переключении с логического уровня на физический автоматически будет создана физическая схема базы данных (рис.8)

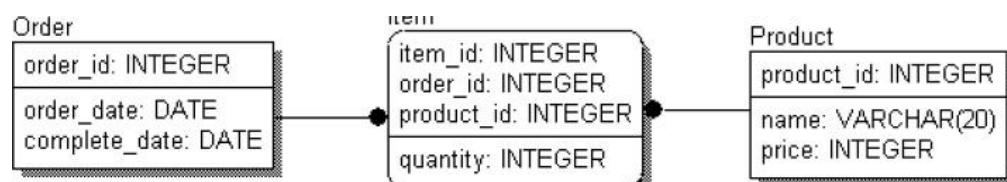


Рисунок 8.Автоматически созданная физическая схема БД

Ее можно дополнить, отредактировать или изменить. Принципы работы с физической схемой аналогичны принципам работы с логической схемой.

По готовой физической схеме можно сгенерировать скрипты для выбранной СУБД. Для этого предназначен пункт меню *Tools -> Forward Engineering/Schema Generation* (рис.9).

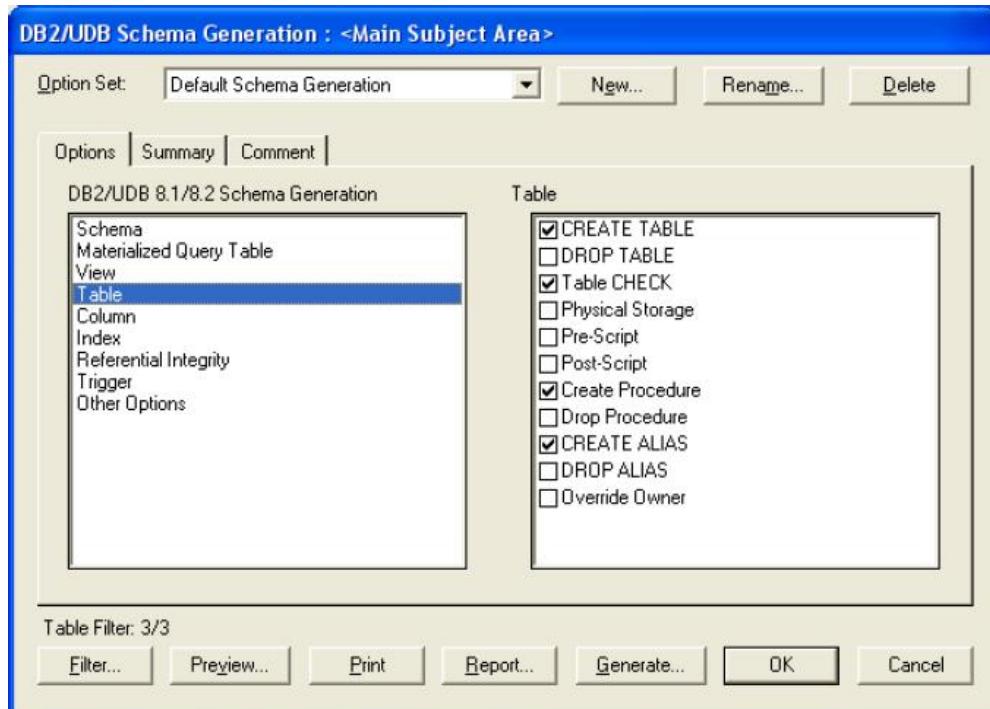


Рисунок 9.Окно генерации SQL-скриптов для целевой СУБД

Здесь можно указать, какие именно скрипты следует генерировать, предварительно просмотреть их и непосредственно сгенерировать (при этом ERwin произведет подключение к целевой СУБД и в автоматическом режиме выполнит все SQL-скрипты).

Данная лабораторная работа будет засчитываться вместе с лабораторной работой № 9 "Построение логической модели данных предметной области".

Лабораторная работа № 9.

Построение логической модели данных предметной области

Цель работы:

- построить логическую модель данных выбранной предметной области в нотации IDEF1X

В данной лабораторной работе необходимо построить в нотации IDEF1X в CASE-средстве ERwin Data Modeler логическую схему данных предметной области, бизнес-процессы которой моделировались в предыдущих лабораторных работах.

Примечание. При построении модели можно ограничиться 5-6 сущностями.

IDEF1X

IDEF1X основан на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. Нотация Чена и сам процесс построения диаграмм сущность-связь изучалась в курсе "Организация баз данных и знаний", поэтому здесь мы рассмотрим только отличия IDEF1X от нотации Чена.

Сущность (Entity) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Атрибут (Attribute) - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе.

Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области.

В методе IDEF1X все сущности делятся на зависимые и независимые от идентификаторов. Сущность является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Независимая сущность изображается в виде обычного прямоугольника, зависимая - в виде прямоугольника с закругленными углами.

В IDEF1X существуют следующие виды мощностей связей:

- N мощность - каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка (по умолчанию);
- P мощность - каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- Z мощность - каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- конкретное число - каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. По умолчанию мощность связи принимается равной N. Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае — неидентифицирующей. Идентифицирующая связь изображается сплошной линией, неидентифицирующая — пунктирной линией.

В ERwin'e при установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ (FK). При установке неидентифицирующей связи атрибуты первичного ключа родительской сущности мигрируют в состав неключевых полей дочерней сущности.

Построение логической модели данных предприятия по сборке и продаже компьютеров и ноутбуков. Построение модели данных начинается с выделения сущностей данной предметной области. В нашем случае были выделены следующие сущности:

- клиент - человек, который покупает компьютеры
- заказ - список компьютеров, которые покупает клиент
- компьютер
- комплектующие - то, из чего собирают компьютеры
- сотрудник - сотрудник предприятия, собирающий конкретный компьютер

Далее рассмотрим связи между сущностями:

- Клиент - Заказ. Один клиент может делать несколько заказов. При этом если данные о клиенте имеются в базе данных, то он сделал минимум один заказ. Поэтому мощность связи - Р. Связь идентифицирующая, т.к. заказ без клиента существовать не может;
- Заказ - Компьютер. В рамках одного заказа клиент может заказать несколько компьютеров, но как минимум заказ должен состоять из одного компьютера. Поэтому мощность связи - Р. Связь идентифицирующая, т.к. компьютер без заказа существовать не может;
- Компьютер - Комплектующие. В состав одного компьютера входит много различных комплектующих; один и тот же тип комплектующего может входить в состав разных компьютеров. Мощность связи - много ко многим. В IDEF1X такой тип связи отсутствует, поэтому вводим промежуточную (ассоциативную) сущность - Конфигурация. Мощность связи между сущностями Компьютер и Конфигурация - Р, поскольку у любого компьютера должна быть конфигурация, мощность между сущностями Комплектующие и Конфигурация - N, поскольку какие-то комплектующие еще могут быть не установлены ни в один компьютер. Связь в обоих случаях идентифицирующая, т.к. конфигурация компьютера не может существовать без привязки к самому компьютеру и к комплектующим;
- Комплектующие - Тип комплектующих. Поскольку перечень типов комплектующих, которые могут быть установлены в компьютер, ограничен, но используется очень часто, то мы приняли решение создать еще одну сущность - Тип комплектующих. Мощность связи - Р. Связь идентифицирующая;
- Компьютер - Сотрудник. Каждый компьютер собирается каким-то одним сотрудником. Какие-то сотрудники могут собирать множество компьютеров. Мощность связи - N. Тип связи - неидентифицирующая, поскольку экземпляр сущности Компьютер уже может существовать, но за ним еще может быть не закреплен ни один сотрудник. Именно из этих же соображений в свойствах этой связи мы выбрали переключатель "Nulls Allowed" (на диаграмме это отображается в виде незакрашенного ромбика со стороны сущности-родителя).

Итоговая диаграмма показана на рис. 1:

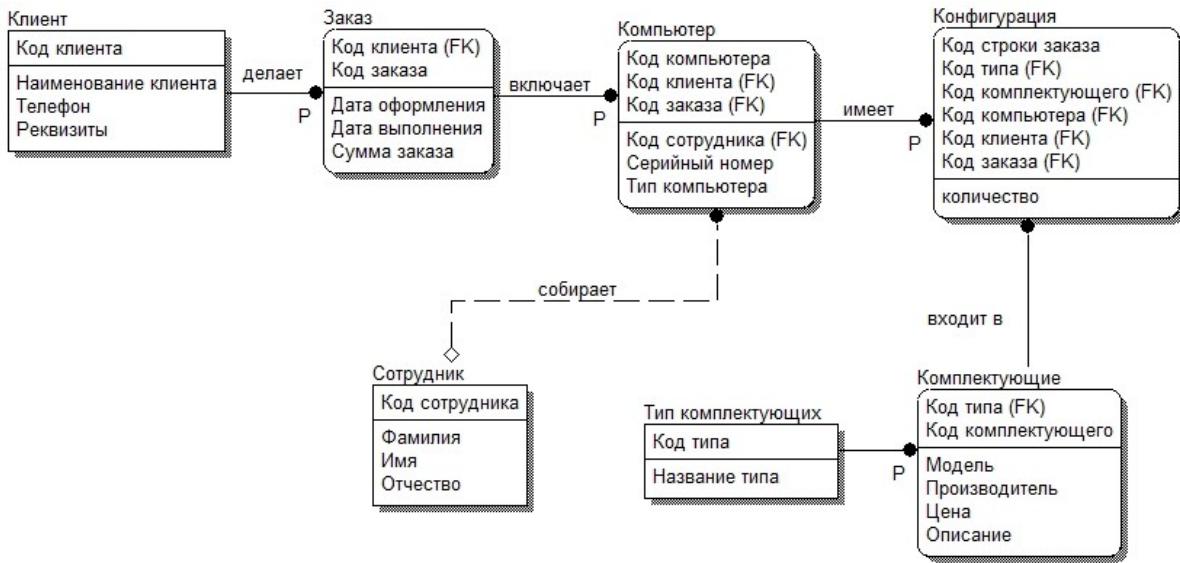


Рисунок 1. Логическая модель данных предприятия по сборке компьютеров и ноутбуков

Содержание отчета:

- логическая модель данных предметной области
- краткое описание каждой сущности

Лабораторная работа № 9

Соответствие логической модели ERwin и модели процессов BPwin

Цель работы: изучить на практических примерах соответствие между логической моделью Erwin и моделью процессов Bpwin и генерацию отчетов в Bpwin.

Задание: Экспортировать данные из ERwin в Bpwin, привязать в Bpwin полученные данные к одной из работ и к связанным с ней стрелкам и сгенерировать отчет в Bpwin, содержащий данные о привязанных сущностях и атрибутах. Дополнить в Bpwin словарь новой сущностью и связанными с ней атрибутами и экспортить данные из словаря сущностей из Bpwin в Erwin, сгенерировать отчет по сущностям и атрибутам Bpwin. Заполнить данные, характеризующие модель Bpwin в целом и одну из диаграмм и сгенерировать отчеты, включающие эту информацию. Создать в одной из работ Bpwin пояснения к работе и относящимся к ней стрелкам и сгенерировать отчет, включающий эти пояснения.

Ход работы:

Этап экспорта данных из ERwin в Bpwin и связывания полученных данных с работами и стрелками.

Из модели данных командой экспорта создаем файл экспорта данных из ERwin. В процессе экспортации данных из Erwin создается сообщение о результатах экспортации (Рис.1). В реальной задаче следует создавать сущности и атрибуты на русском языке.

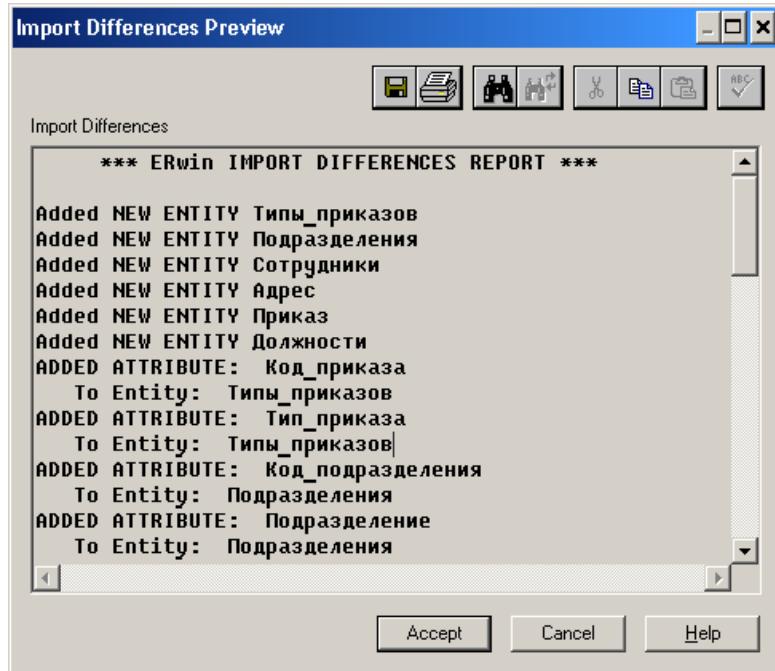


Рис.1. Экспортированные данные из ERwin в Bpwin

Далее импортируем данные в Bpwin и привязываем полученные данные к одной из работ и к связанным с ней стрелкам. Генерируем отчет DataUsage Report, содержащий данные о привязанных сущностях и атрибутах (Рис.2).

Report Format: Column		
Entity Name	Entity Definition	Attribute Name
Адрес		Город
		Дом
		Квартира
		Код_адреса
		Улица
Должности		Должность
		Код_должности
		Оклад
Подразделения		Код_подразделения
		Подразделение
Приказ		Дата
		Код_приказа
		Номер_приказа
		Табельный_номер
Сотрудники		Возраст
		Код_адреса

Рис.2. Отчет DataUsage Report

Этап дополнения словаря в Bpwin новой сущностью и атрибутами и экспортта данных из словаря сущностей из Bpwin в Erwin.

Новая импортированная в Erwin сущность не имеет первичного ключа и не связана с другими сущностями. Назначение атрибутов первичным ключом и связывание сущностей можно провести только средствами Erwin; другими словами, сущности и атрибуты, созданные в BPwin и затем импортированные в Erwin, можно рассматривать как заготовку для создания полноценной модели данных, а не как готовую модель.

Дополним словарь Bpwin новой сущностью и связанными с ней атрибутами. На рисунках 3 и 4 показан пример создания новых сущностей и атрибутов в словаре BPwin.

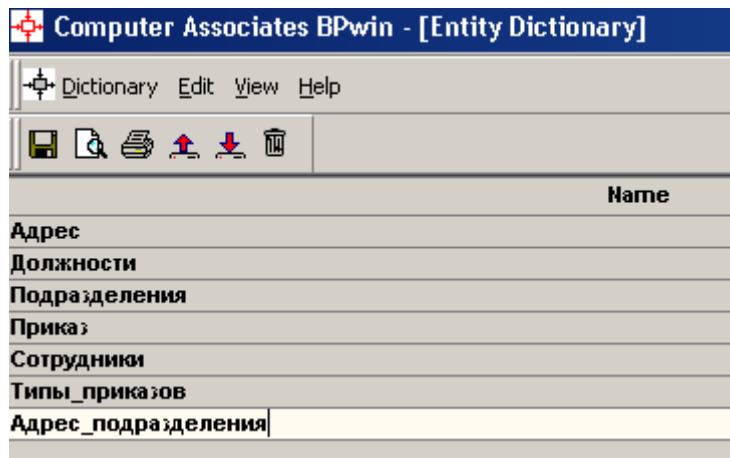


Рис.3. Добавление новой сущности

Computer Associates BPwin - [Attribute Dictionary]			
Name	Definition	Exchange with ERwin	Entity
Возраст		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Город		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес
Дата		<input checked="" type="checkbox"/> Exchange with ERwin	Приказ
Должность		<input checked="" type="checkbox"/> Exchange with ERwin	Должности
Дом		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес
Квартира		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес
Код_адреса		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Код_адреса		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес
Код_должности		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Код_должности		<input checked="" type="checkbox"/> Exchange with ERwin	Должности
Код_подразделения		<input checked="" type="checkbox"/> Exchange with ERwin	Подразделения
Код_подразделения		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Код_приказа		<input checked="" type="checkbox"/> Exchange with ERwin	Типы_приказов
Код_приказа		<input checked="" type="checkbox"/> Exchange with ERwin	Приказ
Номер_приказа		<input checked="" type="checkbox"/> Exchange with ERwin	Приказ
Образование		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Оклад		<input checked="" type="checkbox"/> Exchange with ERwin	Должности
Подразделение		<input checked="" type="checkbox"/> Exchange with ERwin	Подразделения
Табельный_номер		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Табельный_номер		<input checked="" type="checkbox"/> Exchange with ERwin	Приказ
Телефон		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Тип_приказа		<input checked="" type="checkbox"/> Exchange with ERwin	Типы_приказов
Улица		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес
ФИО		<input checked="" type="checkbox"/> Exchange with ERwin	Сотрудники
Дом_1		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес_подразделения
Улица_1		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес_подразделения
Телефон_1		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес_подразделения
		<input checked="" type="checkbox"/> Exchange with ERwin	Адрес_подразделения

Рис.4. Связывание атрибутов с новой сущностью

Командой экспорта экспортим данные из словаря сущностей из Bpwin в файл экспорта. При импорте данных в Erwin создается сообщение (Рис.5).

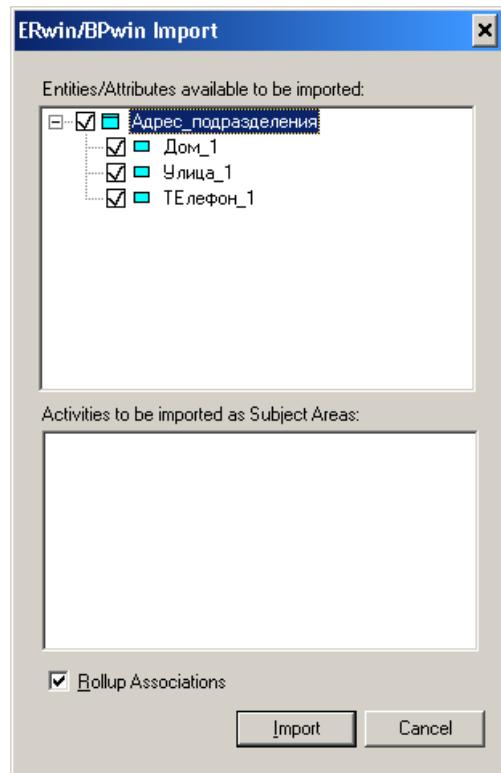


Рис.5. Импортирование данные из Bpwin в Erwin

Этап создания отчетов.

После заполнения данных, характеризующих модель Bpwin в целом (автор, цель и т.д.) и одну из диаграмм (описание диаграммы), генерируем отчеты Model Report (Рис.6) и Diagram Report (Рис.7).

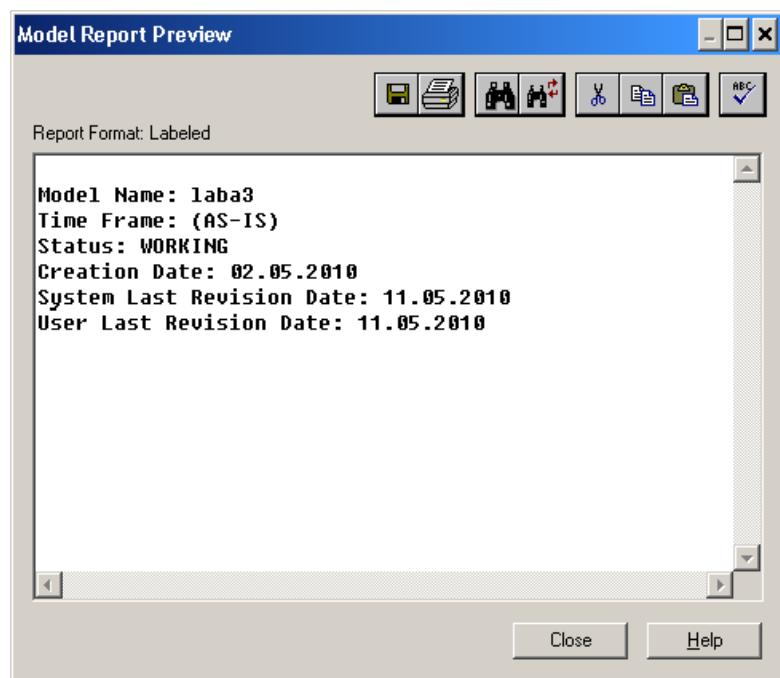


Рис.6. Отчет Model Report

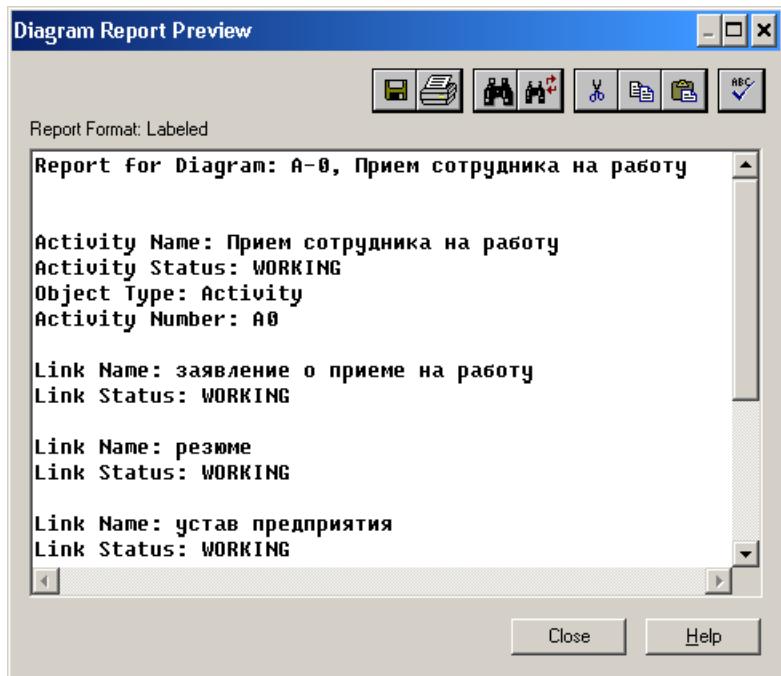


Рис.7. Отчет Diagram Report

Создаем в одной из работ Bpwin пояснения к работе и относящимся к ней стрелкам и генерируем отчет Diagram Object Report (Рис.8).

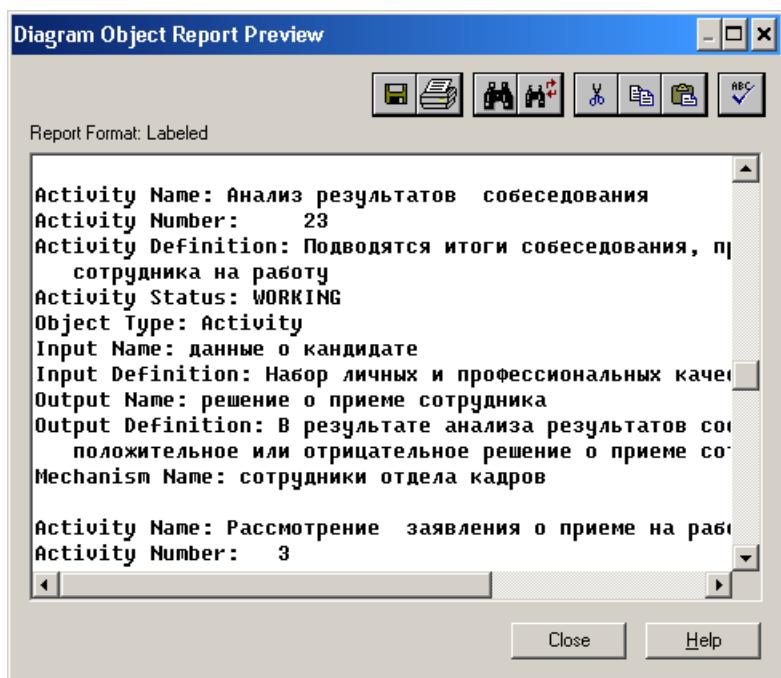


Рис.8. Отчет Diagram Object Report

Вывод: благодаря проделанной работе мы на практических примерах получили соответствие между логической моделью Erwin и моделью процессов Bpwin и технологию генерации отчетов в Bpwin.

Лабораторная работа № 11

Лабораторная работа № 7

ERwin. Прямое и обратное проектирование

Цель работы: овладеть навыками прямого и обратного проектирования в среде ERwin для «файл-серверных» и «клиент-серверных» СУБД.

Задание: Реализовать прямое проектирование в архитектуре «файл-сервер» Access. Изменить структуру БД и осуществить обратное проектирование. Реализовать прямое проектирование в архитектуре «клиент-сервер» (MS SQL Server), сгенерировать SQL – код создания базы данных на основе физической модели данных.

Ход работы:

Этап прямого проектирования в архитектуре «файл-сервер».

Рассмотрим исходные логические и физические модели данных (Рис.1, Рис.2).

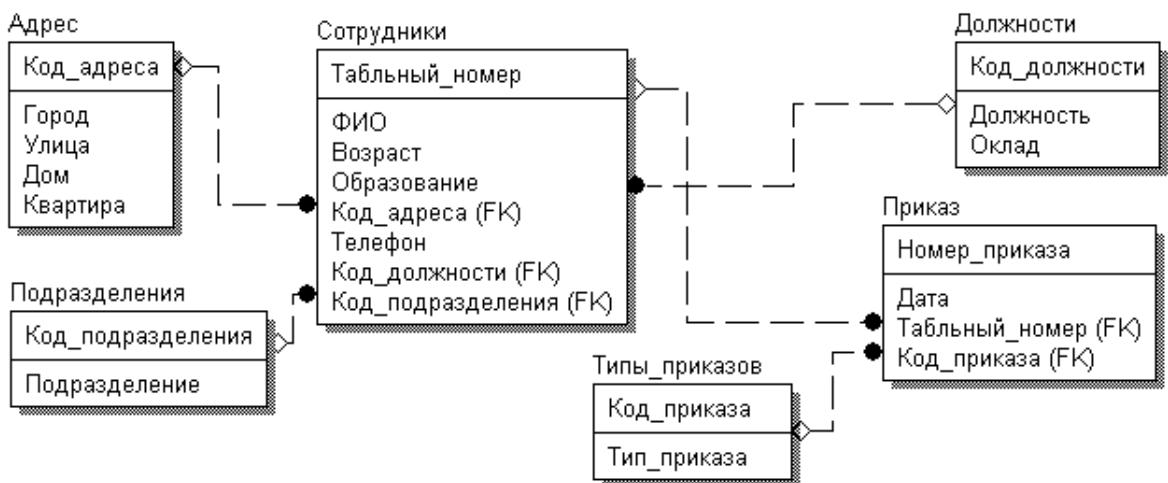


Рис.1. Логическая модель проектируемой ИС

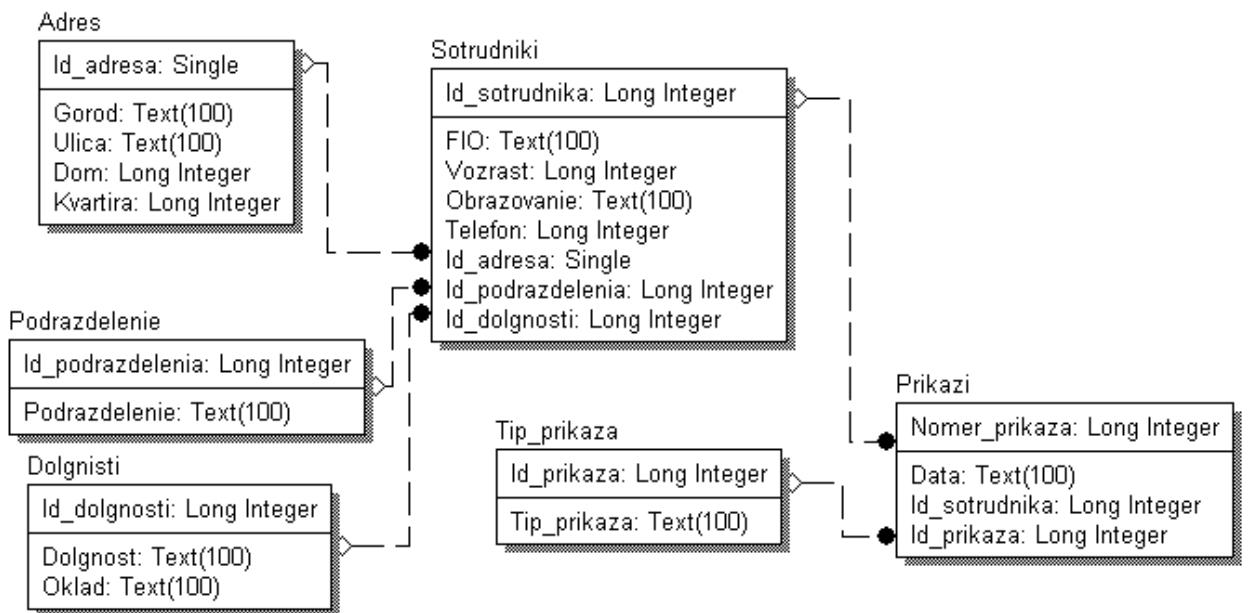


Рис.2. Физическая модель проектируемой ИС

Открываем физическую модель ИС и выбираем Access в качестве нужного типа СУБД, после чего типы данных в физической модели изменятся, так как по умолчанию она может быть настроена на другую СУБД.

Создаем пустую базу данных в Access и подключаемся к ней (Рис.3, Рис.4).

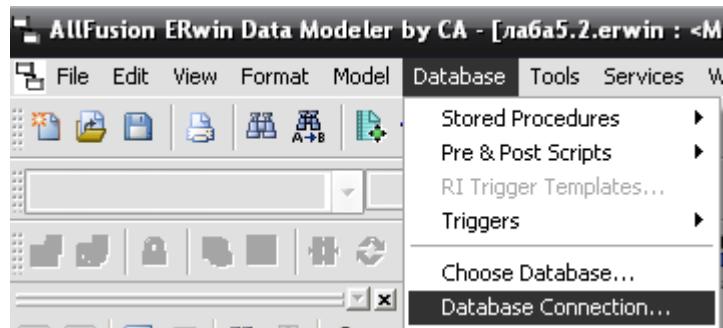


Рис.3. Подключение к СУБД Access

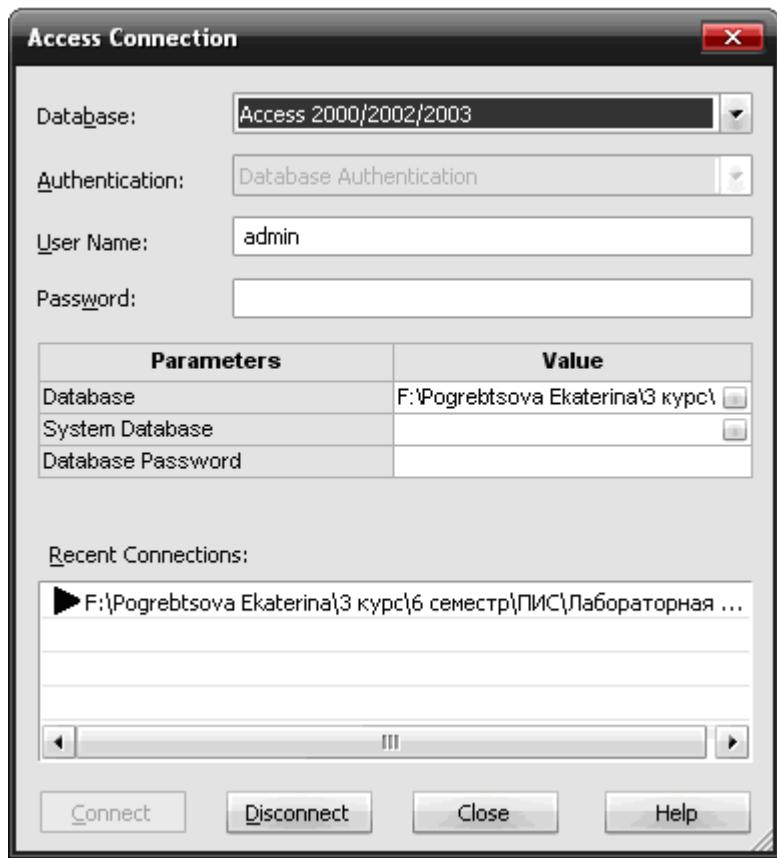


Рис.4. Выбор БД Access

Далее в меню выбираем Tools/ Forward Engineer/Shema Generation.

В открывшемся окне на вкладке Options в пункте Index поставили галочки напротив пунктов Primary Key и Foreign Key, отвечающих за генерацию первичных и внешних ключей (Рис.5).

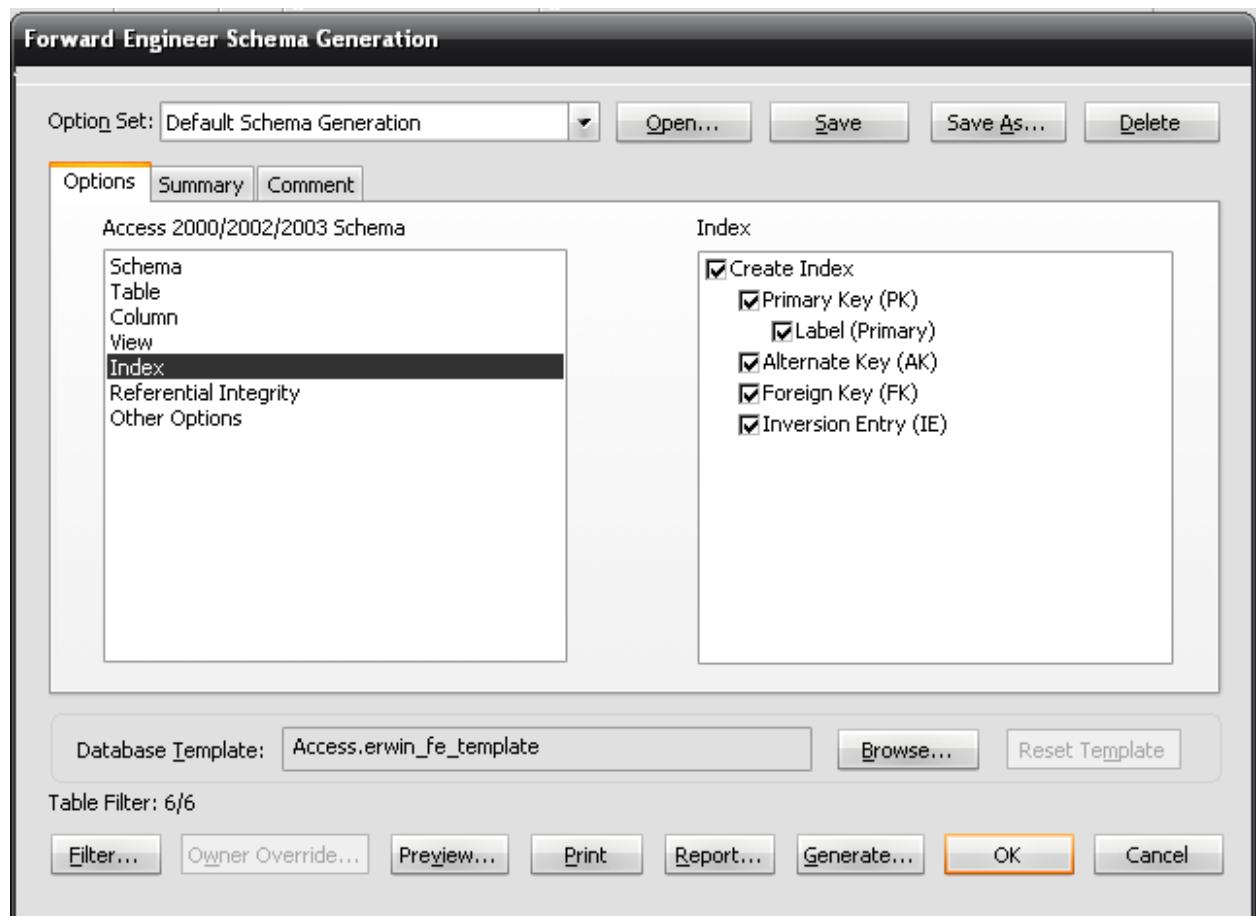


Рис.5. Установки по генерации схемы для базы данных Access

После завершения операции по переносу физической модели в Access заходим в полученную базу данных и проверяем результат (Рис.6).

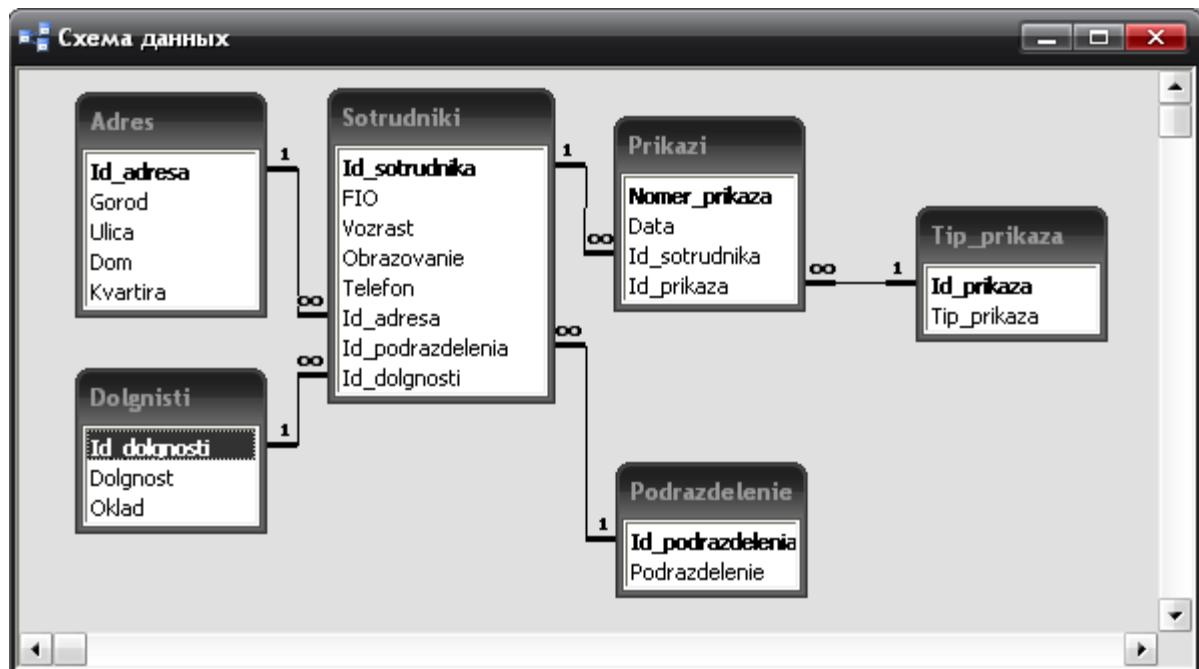


Рис.6. Схема данных в Access

Этап обратного проектирования.

В базе данных Access в таблице Адрес добавили поле e-mail и сохранили изменения. Далее зашли в Erwin и в меню выбрали Tools/ Reverse Engineer. В открывшемся окне выбрали тип новой модели - физическая, и СУБД из которой будем импортироваться физическая модель – Access (Рис.7).

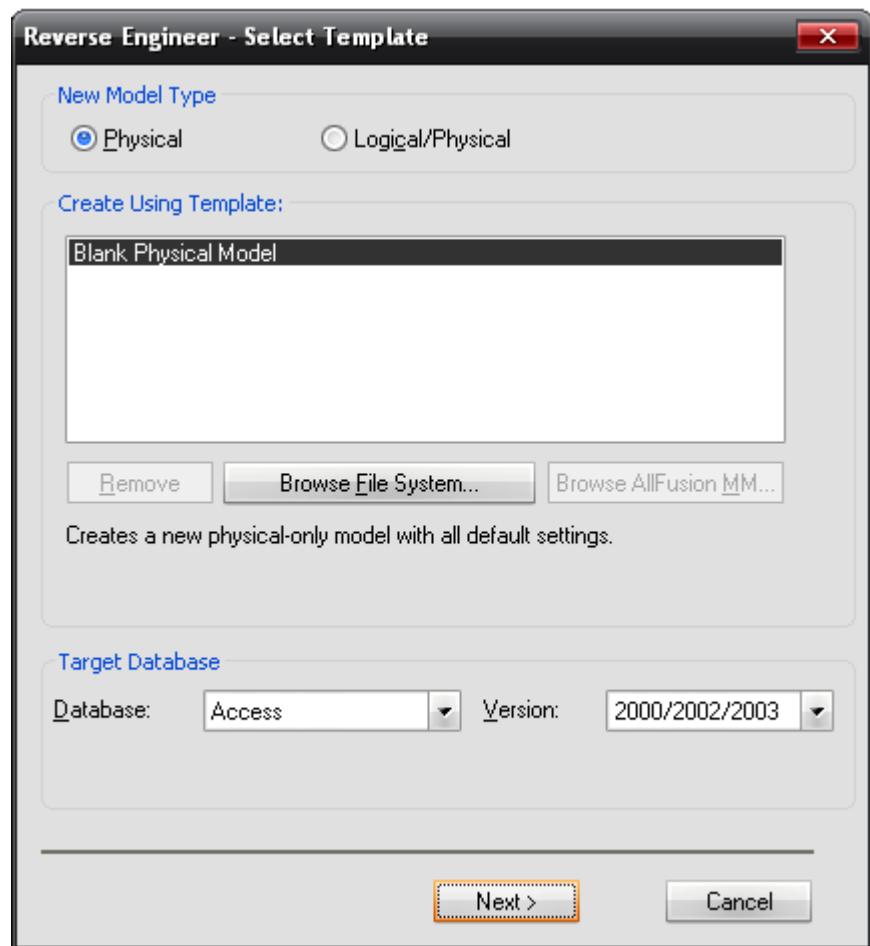


Рис.7. Установки обратного проектирования

Далее настраиваем параметры проектирования (Рис.8).

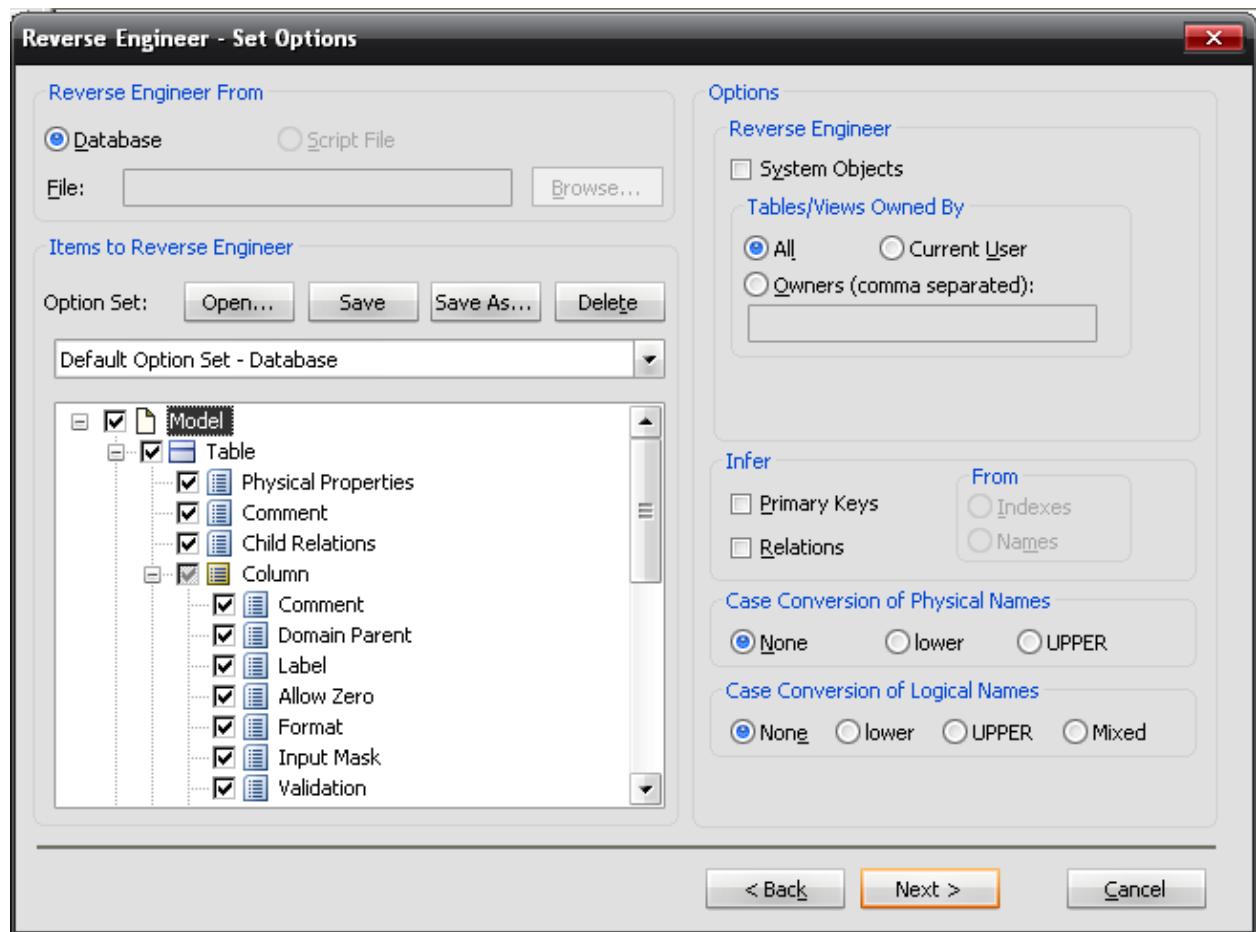


Рис.8. Установки по генерированию схемы для Erwin.

Подключение к Access аналогично режиму прямого проектирования.

Получаем физическую модель (Рис.9).

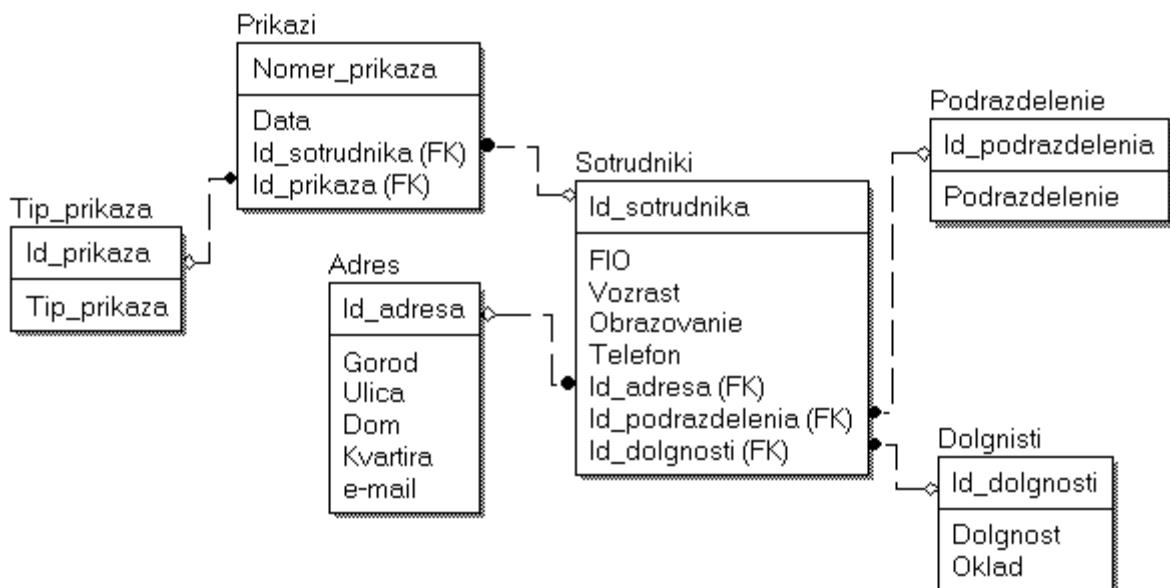


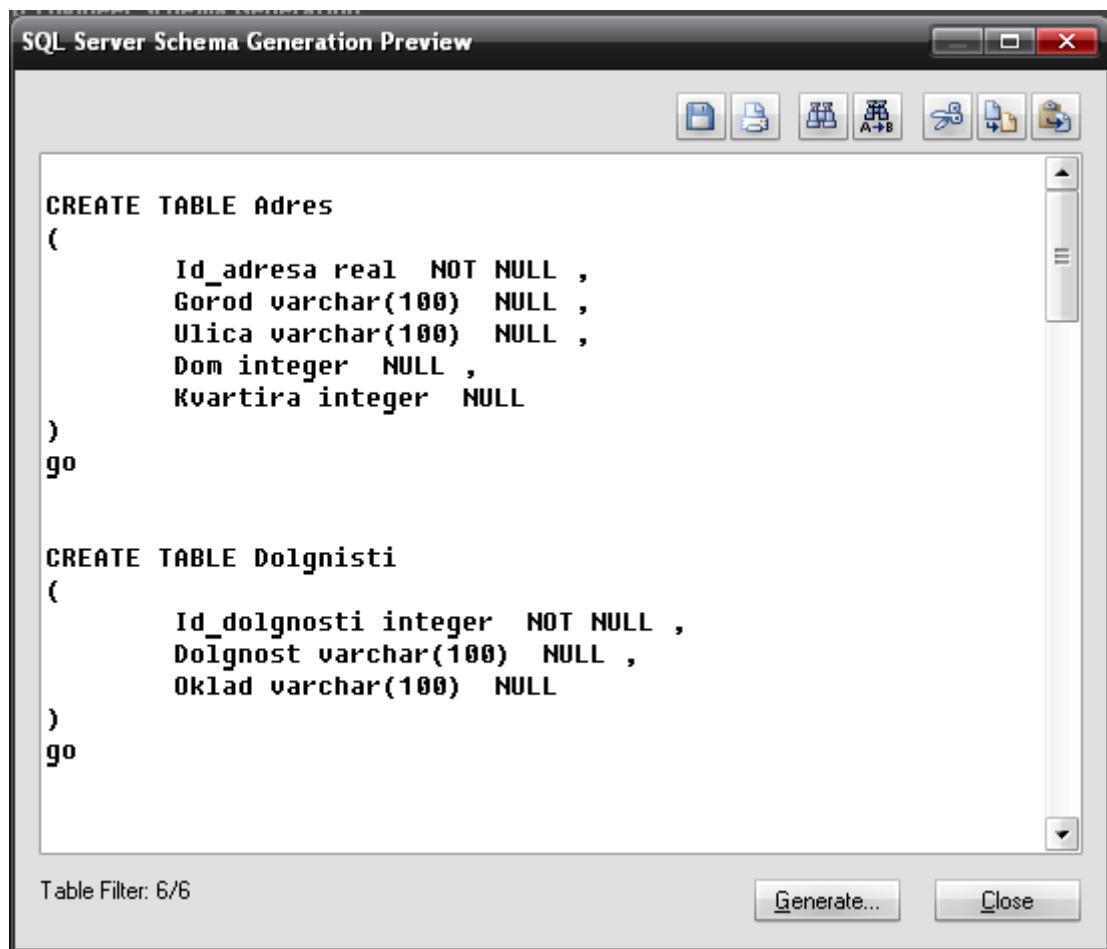
Рис.9. Физическая модель, полученная из БД Access

Этап проектирования БД для архитектуры “клиент-сервер”.

Проделываем действия как и для варианта с подключением к Access, а также сохраняем SQL-запрос на создание БД

В среде Erwin открыли физическую модель ИС, изменили тип СУБД на Microsoft SQL Server, в меню выбрали *Tools/ Forward Engineer/Schema Generation*.

В открывшемся окне на вкладке *Options* в пункте *Index* поставили галочки напротив пунктов *Primary Key* и *Foreign Key*, отвечающих за генерацию первичных и внешних ключей. Нажали кнопку *Preview* (Рис.10).



The screenshot shows the 'SQL Server Schema Generation Preview' dialog box. The main area contains the generated SQL code for creating two tables:

```
CREATE TABLE Adres
(
    Id_adresa real NOT NULL ,
    Gorod varchar(100) NULL ,
    Ulica varchar(100) NULL ,
    Dom integer NULL ,
    Kvartira integer NULL
)
go

CREATE TABLE Dolgnosti
(
    Id_dolgnosti integer NOT NULL ,
    Dolgnost varchar(100) NULL ,
    Oklad varchar(100) NULL
)
go
```

At the bottom left, it says 'Table Filter: 6/6'. At the bottom right, there are 'Generate...' and 'Close' buttons.

Рис.10. Генерация SQL-кода для MS SQL

Вывод: В процессе выполнения лабораторной работы получены навыки прямого и обратного проектирования в среде Erwin для «файл-серверных», «клиент-серверных» СУБД.

Лабораторная работа №9

Создание главного окна и главного меню клиентского приложения ИС средствами Microsoft Visual Studio. Net

Цель работы: Изучить основные элементы среды разработки Visual Studio при создании на языке C# приложений с графическим интерфейсом.

План проведения занятия

1. Изучить теоретический материал.
2. Создать Windows форму, на Windows форме создать кнопку "Приветствие", добавить в форму две кнопки, для которых задать различные цвета, написать для кнопок 1 и 2 обработчики, которые изменяют цвета кнопок, добавьте кнопку "Выход".
3. Протестировать работу приложения.

Порядок выполнения работы

Изучите теоретический материал из предложенной ниже литературы.

Задание 1.

Создать Windows форму.

В качестве приложения разработать простое приложение, пользовательского интерфейса которое будет содержать только главное окно. Для этого необходимо выполнить следующие шаги:

1. Создайте рабочую область, называемую также рабочей средой (проектирования), рабочим пространством и рабочей обстановкой нового проекта. Для создания каркаса приложения можно использовать мастер создания приложений - Application Wizard.

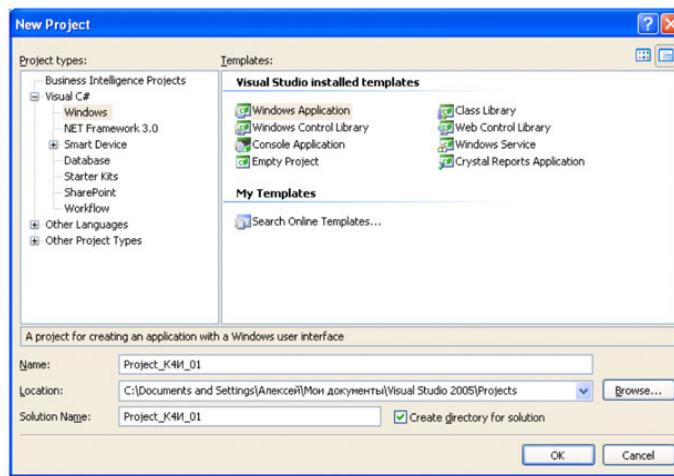


Рис. 9.1. Создания каркаса приложения

2. Создание рабочей среды нового проекта производится следующим образом:

Щелкните на ссылке *Project* (Создать новый проект) метки *Create* на начальной странице (*Start Page*) VS.NET. При этом откроется окно создания нового проекта *New Project*.

3. Для разметки окон приложения в соответствии с требованиями пользователя необходимо изменить свойства класса `Forms1`. Это можно сделать с помощью дизайнера окон (*Form Designer*), путем изменения свойств в окне Свойства (*Properties*) или в коде программы, аналогичным способом можно изменить и другие свойства окна.
4. На вкладке *Properties* измените значение в поле *Text* (Заголовок) на *Проект БИ*. При этом на форме изменится заголовок окна [рис. 9.2](#).

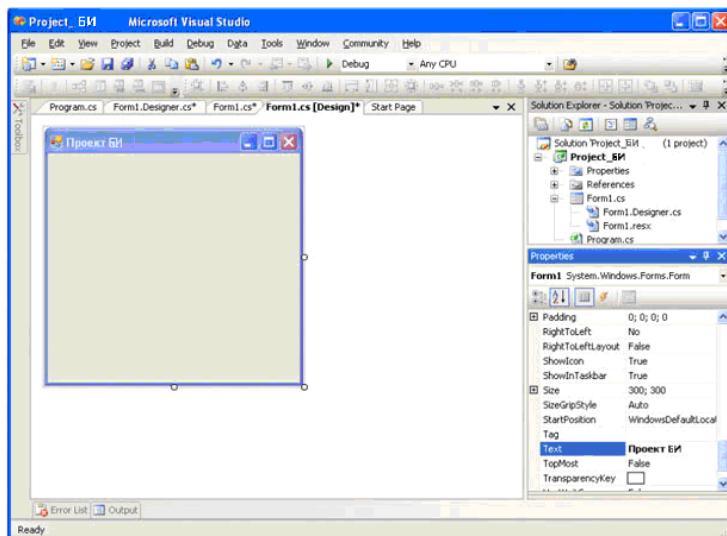


Рис. 9.2. Изменение значения в поле Text на вкладке Properties

5. Откомпилируйте приложение, выбрав из главного меню команду `Build :Build Project_BI`. В строке состояний должно появиться сообщение: `Build succeeded`
6. Для запуска приложения выберите из главного меню команду `Debug/Start (F5)`. Приложение запустится в отладочном режиме и на экране появится разработанное окно.

Задание 2.

На Windows форме создать кнопку "Приветствие".

1. Добавьте в главную форму элемент контроля - кнопку.
2. Для этого откроем вкладку *ToolBox* [рис. 9.3](#), и сначала щелкнем на элементе *Button* вкладки, а затем щелкнем на форме. В результате получим форму с кнопкой.

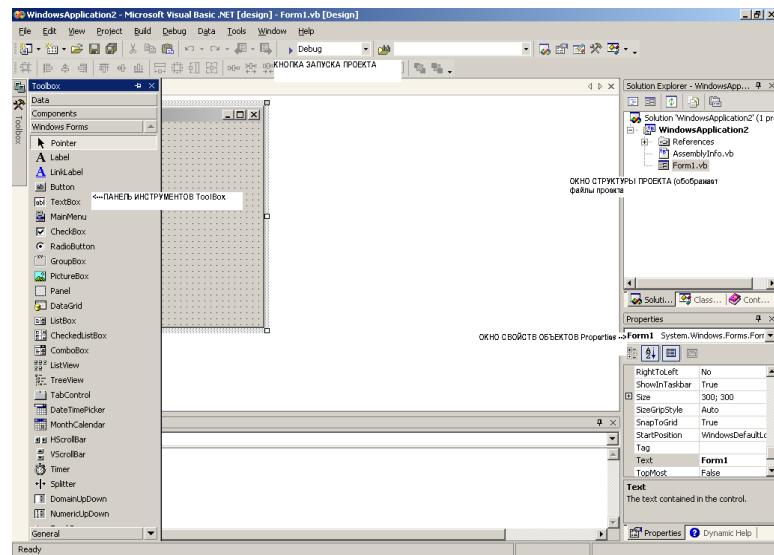


Рис. 9.3. Вкладка ToolBox

3. Для задания текста на кнопке выделите ее на форме и откройте вкладку *Свойства* и измените свойство *Text* на «Приветствие»

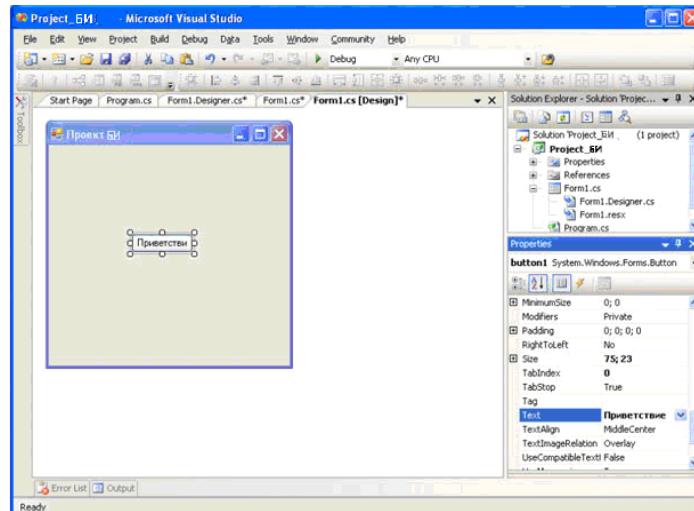


Рис. 9.4. Форма с измененным свойством Text кнопки

4. Для связывания функций кнопки с диалоговым окном необходимо создать обработчик события на нажатие кнопки. Для этого двойным щелчком мыши по кнопке откройте код приложения, в котором сформируется шаблон функции обработчика события Click для кнопки:

```
private void button1_Click(object sender, EventArgs e){}
```

5. В полученный шаблон добавьте функцию вывода диалогового окна с сообщением.

```
private void button1_Click(object sender, EventArgs e)
{
    // Сообщение
    MessageBox.Show("Поздравляю с первым проектом на C#");
}
```

6. После компиляции и запуска приложения получим следующее окно приложения [рис.9.5](#), а при нажатии кнопки будет выведено сообщение.

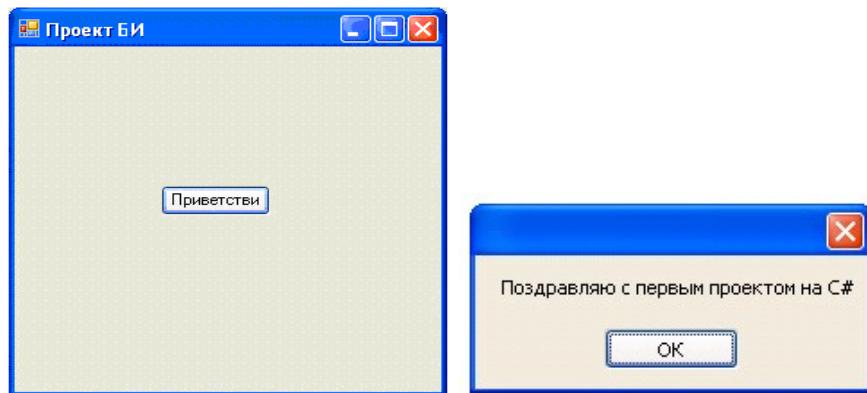


Рис. 9.5. Результат выполнения приложения

Задание 3.

Аналогично пункту 1 добавить в форму две кнопки (1 и 2), для которых задать различные цвета (свойство BackColor).

1. Добавьте на Windows форму 2 кнопки.
2. Напишите для них обработчики, которые изменяют цвета кнопок: при неоднократном нажатии любой кнопки цвета кнопок меняются (цвет кнопки 1 меняется на цвет кнопки 2 и наоборот).

Задание 4.

Добавьте кнопку "Выход"

1. Добавьте кнопку "Выход" Windows форму. Закрытие приложения обеспечивает метод Exit() класса Application.
2. Протестировать работу приложения.

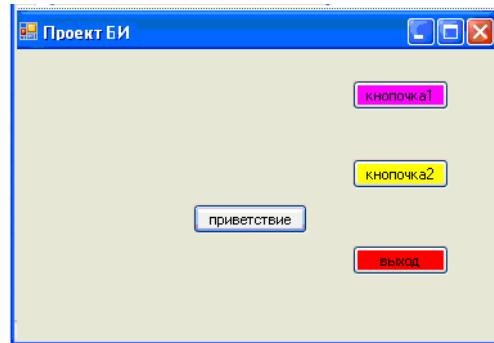


Рис. 9.6.

Содержание отчета

1. Создание Windows формы и создание кнопок на данной форме, написании для кнопок 1 и 2 обработчиков.
2. Тестирование работы приложения.
3. Выводы о проделанной работе.

Контрольные вопросы

1. Для создания кода на каких языках можно использовать Visual Studio?
2. Какие функциональные возможности в приложении имеет вкладка Properties?

3. Какой элемент должны стоять в C# при вызове функции за ее именем, даже если данной функции не передается ни один параметр?
4. Что необходимо учитывать при вводе программ, написанных на языке C#?
5. В какой элемент всегда заключается тело функции?

Литература [3-4, 11-12].

Лабораторная работа №10

Создание пользовательских диалоговых окон, панели инструментов, контекстного меню и строки состояния клиентского приложения ИС

Цель работы: Изучить основные способы разработки главного меню приложения. Получить практические навыки в создании главного меню приложения. Изучить основные способы разработки многооконных приложений, способы построения диалоговых окон, их параметры, способы построения панели инструментов и контекстного меню, их параметры

План проведения занятия

1. Изучить теоретический материал.
2. Создать главное меню для пунктов главного меню создать подпункты
3. Создать дочернее окно. В дочернее окно добавить пункты меню. Написать обработчик для вызова из главного меню дочернего окна.
4. Создать модальное диалоговое окно с помощью класса MessageBox. Пользовательское модальное диалоговое окно для пункта меню "О программе". Написать обработчики для вызова модальных окон.
5. Создать коды методов-заглушек для функций приложения.
6. Создать обработчики для вызова пунктов меню, панели инструментов и контекстного меню
7. Протестировать работу приложения.

Краткие теоретические сведения

Создание меню в режиме проектирования

Для построения в режиме проектирования главного меню и связанной с ним структуры достаточно перетащить на форму элемент управления, называемый MainMenu. (В Visual Studio 2005 элемент управления для создания меню называется ToolStrip, а для создания инструментальных панелей - ToolStrip.)

После перетаскивания метка с изображением этого элемента управления появляется ниже формы, а на форме появляется элемент меню с информационным полем, в котором можно задать название пункта меню, и двумя указателями на правого брата и старшего сына, позволяющими перейти к следующему пункту меню того же уровня или опуститься на нижний уровень. Технология создания меню вручную интуитивно ясна и не вызывает обычно никаких проблем

В пространстве имен *System.Windows.Forms* предусмотрено большое количество типов для организации ниспадающих главных меню (расположенных в верхней части формы) и контекстных меню, открывающихся по щелчку правой кнопки мыши.

Элемент управления ToolStrip представляет собой контейнер, используемый для создания структур меню, панелей инструментов и строк состояний.

Элемент управления MenuStrip - это контейнер для структур меню в приложении. Этот элемент управления наследуется от ToolStrip. Система меню строится добавлением объектов ToolStripMenuItem к menuStrip.

Класс ToolStripMenuItem служит для построения структур меню. Каждый объект ToolStripMenuItem представляет отдельный пункт в системе меню.

Для отображения диалогового окна использовался метод Show, передав ему через параметр текст сообщения "метод Undo". Прототип использованного метода Show следующий:

```
public static DialogResult Show(string message);
```

Когда пользователь щелкает кнопку *OK*, метод Show возвращает значение, равное DialogResult.OK

Прототип наиболее общего варианта метода MessageBox.Show, позволяющий реализовать практически все возможности диалогового окна *MessageBox*, приведен ниже

```
public static DialogResult Show
{
    string message, // текст сообщения
    string caption, // заголовок окна
    MessageBoxButtons bts, // кнопки, расположенные в окне
    MessageBoxIcon icon, // значок, расположенный в окне
    MessageBoxButtons defButton, // кнопка по умолчанию
    MessageBoxIcon opt // дополнительные параметры
};
```

Параметр caption позволяет задать текст заголовка диалогового окна *MessageBox*. С помощью параметра bts можно указать, какие кнопки необходимо расположить в окне диалогового окна. Этот параметр задается константами из перечисления MessageBoxButtons

Параметр icon метода MessageBox.Show позволяет выбрать один из нескольких значков для расположения в левой части диалогового окна. Он задается в виде константы перечисления MessageBoxIcon

Порядок выполнения работы

Задание 1.

Создание меню

1. Создайте стандартное ниспадающее меню, позволяющее пользователю выйти из приложения, выбрав пункт *Объект / Выход*. Для этого перетащите элемент управления MenuStrip на форму в конструкторе рис. 10.1.

Главное меню, должно включать следующие пункты: "Объект", "Справочник", "Справка".

При помощи графических средств можно настроить свойства любого элемента меню. Для пунктов меню задайте свойства:

"*Объект*" - свойство Name равным objektToolStripMenuItem,

"*Выход*" - exitToolStripMenuItem,

"*Справка*" - HelpToolStripMenuItem.

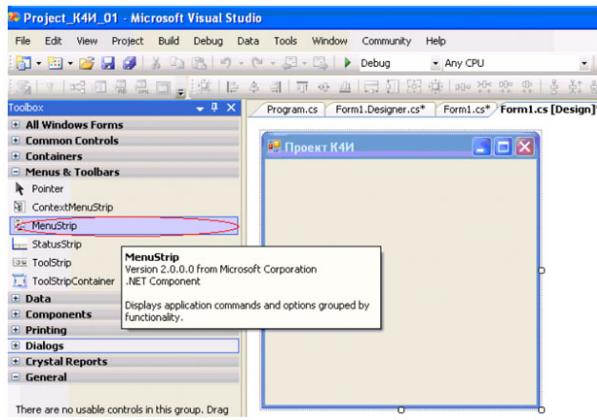


Рис. 10.1. Элемент управления ToolStrip

- Для пункта "Объект" создать следующие подпункты: "Сотрудник", "Клиент", "Договор", "Поручение", "Сделка", "Выход.

Элемент управления ToolStrip позволит вводить текст меню непосредственно в элементы меню рис. 10.2.

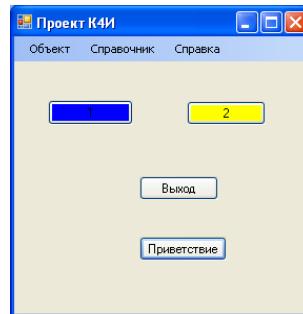


Рис. 10.2. Простое меню на форме

- Для пункта "Справочник" создать следующие подпункты: "Должность", "Страна", "Регион", "Город", "ИМНС". Для пункта "Справка" создать подпункт - "О программе".

Сделайте двойной щелчок на пункте меню "Выход" объект xitToolStripMenuItem. Visual Studio автоматически сгенерирует оболочку для обработчика события Click и перейдет в окно кода, в котором нам будет предложено создать логику метода (в нашем случае exitToolStripMenuItem_Click):

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    // Здесь мы определяем реакцию на выбор пользователем пункта меню
}
```

- На вкладке Свойства (Properties) при выводе окна событий, нажмите кнопку событию Click будет соответствовать метод menuItemExit_Click [рис. 10.3.](#)

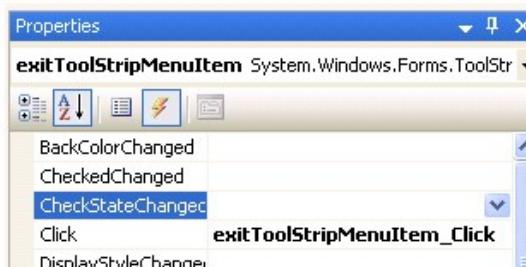


Рис. 10.3. Событие Click и обработчик события exitToolStripMenuItem_Click

5. Напишите код обработчика для корректного завершения приложения exitToolStripMenuItem_Click. Это можно сделать с помощью метода Exit класса Application:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

6. Протестируйте созданное меню, для этого создайте обработчик для пункта меню "Объект", который будет сообщать, что выбран именно этот пункт меню:

```
private void objektToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Пункт меню Объект");
}
```

При создании меню графическими средствами *Visual Studio* автоматически внесет необходимые изменения в служебный метод InitializeComponent и добавит переменные-члены, представляющие созданные элементы меню.

Задание 2.

Создание дочерней формы

Основа Интерфейса (*MDI*) приложения - *MDI* родительская форма. Это - форма, которая содержит *MDI* дочерние окна. Дочерние окна являются "подокнами", с которыми пользователь взаимодействует в *MDI* приложении.

1. Для определения главного окна (Form1), как родительской формы в окне *Свойств*, установите IsMDIContainer свойство - true. Это определяет форму как *MDI* контейнер для дочерних форм. Для того чтобы родительское окно занимало весь экран необходимо свойству WindowsState установить значение Maximized.
2. Создайте еще одно окно, которое будет дочерним (FormEmployee). Для этого выберите пункт меню *Project/Add Windows Form*. Это окно должно вызываться из пункта главного меню "*Сотрудник*".
3. Вставьте код, подобный следующему, чтобы создать новую *MDI* дочернюю форму, когда пользователь щелкает на пункте меню, например "*Сотрудник*" - имя объекта - employeeToolStripMenuItem (В примере ниже, указатель события обращается к событию Click для employeeToolStripMenuItem_Click).

Данный обработчик приведет к выводу на экран дочернего окна

```
private void menuItemEmploye_Click(object sender, System.EventArgs e)
{
    // Создать объект FEmployee класса FormEmployee
    FormEmployee FEmployee = new FormEmployee();
    // Установить родительское окно для дочернего
    FEmployee.MdiParent = this;
    // Вывести на экран дочернее окно FEmployee.Show();
}
```

Задание 3.

Создание меню в дочерней форме

1. Добавьте в дочернее окно пункт меню "Действие" (actionToolStripMenuItem) с подпунктами:

- "Отменить" (undoToolStripMenuItem),
- "Создать" (createToolStripMenuItem),
- "Редактировать" (editToolStripMenuItem),

- "Сохранить" (saveToolStripMenuItem) ,
 "Удалить" (removeToolStripMenuItem).
2. Перед пунктом удалить вставьте разделитель (Separator - name = toolStripSeparator1).
 3. Добавьте в дочернее окно еще пункт меню "Отчет"(reportToolStripMenuItem) с подпунктами "По сотруднику" (reportToolStripMenuItem1), "По всем сотрудникам" (reportToolStripMenuItem2). Дочернее окно будет иметь вид [рис. 10.4](#).

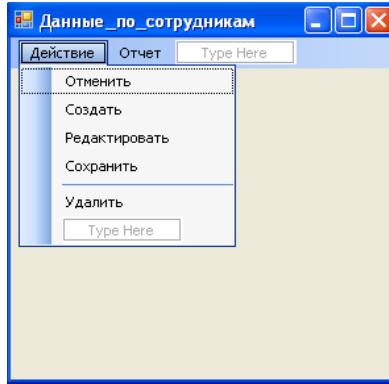


Рис. 10.4. Дочернее окно с меню

4. В главном меню родительской формы (Form1) имеются пункты: "Объект", "Справочник" и "Справка".

В дочерней форме (FormEmployee) сформированы пункты меню "Действие" и "Отчет". При загрузке дочерней формы меню родительской и дочерних форм должны были объединены и составлять следующую последовательность: "Объект", "Действие", "Отчет", "Справочник" и "Справка".

Объединение пунктов меню производится с помощью задания значений свойств `MegreAction` и `MegreIndex` для объектов `ToolStripMenuItem`.

5. Проверьте, чтобы в меню главного окна для объекта `objektToolStripMenuItem` свойство `MegreAction` было установлено `Append`, а `MegreIndex` было равно 0, а для объектов `dictionaryToolStripMenuItem` и `helpToolStripMenuItem` - соответственно 1 и 2. С учетом этого, в окне "Сотрудник" для объектов `actionToolStripMenuItem` (Действие) и "Отчет" (`reportToolStripMenuItem`) свойству `MegreAction` необходимо задать значение `Insert`, а свойству `MegreIndex` задаем порядковый номер который определяет позицию данного пункта меню обновленном главном меню, т.е. 1 (после объекта `objektToolStripMenuItem`).

6. После компиляции программы, запуска ее на выполнение и вызова пункта меню "Сотрудник" экран должен иметь вид, представленный на рисунке 10.5.

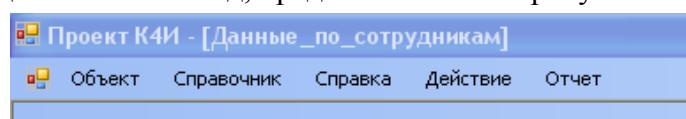


Рис.10.5. Дочернее окно с подключенным меню

Задание 4.

Создание обработчиков для меню дочерней формы

Созданные пункты меню для дочернего окна должны инициировать выполнение соответствующих функций (Отменить, Создать, Редактировать, Сохранить и Удалить) приложения в отношении объектов конкретного дочернего окна. Для дочернего окна "Данные по сотруднику" эти функции должны выполнять соответственно: отмену изменения данных по сотруднику (функция "Отменить"), создание новых данных по

сотруднику (функция "Создать"), внесение изменений в данные по сотруднику (функция "Редактировать"), сохранение созданных вновь или отредактированных данных по сотруднику (функция "Сохранить") и удаление данных по сотруднику (функция "Удалить").

Реализовать описанную функциональность целесообразно в программе в виде методов класса созданного класса FormEmployee.

1. В приложении необходимо создать следующие методы:
Undo-отменить; New-создать; Edit -редактировать; Save -сохранить; Remove-удалить

На начальных этапах проектирования, как правило, неясна реализация каждого метода, поэтому целесообразно их выполнять в виде методов-заглушек, которые только сообщают пользователю о своем вызове, а в дальнейшем необходимо написать реальный код.

2. Для создания метода Undo в коде файла FormEmployee.cs добавьте следующий метод:

```
private void Undo()
{
    MessageBox.Show("метод Undo");
}
```

3. Создайте обработчик события вызова пункта меню "Отменить". Для этого в дизайнере формы класса FormEmployee делаем двойной щелчок на пункте меню "Отменить". Инструментальная среда VS генерирует следующий код:

```
private void undoToolStripMenuItem_Click(object sender, EventArgs e){}
```

4. В код обработчика undoToolStripMenuItem_Click добавим вызов метода Undo:
private void undoToolStripMenuItem_Click(object sender, EventArgs e){Undo();}

5. Откомпилируйте приложение и протестируем вызов метода Undo. В результате выбора пункта меню "Отменить" должно быть выведено диалоговое окно с сообщением, приведенным на [рис. 10.6](#).

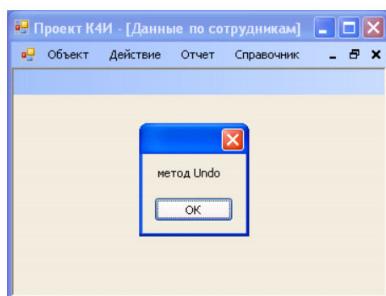


Рис. 10.6. Дочернее окно с подключенным меню

6. Аналогичным образом создайте методы-заглушки для функций "Создать", "Редактировать", "Сохранить" и "Удалить".

Задание 5.

Создать модальное диалоговое окно предупреждения

1. Измените метод Remove, добавив в него предупреждение перед удалением данных по сотруднику. Текст кода метода Remove должен иметь следующий вид:

```
private void Remove()
{
    DialogResult result = MessageBox.Show(" Удалить данные \n по сотруднику?
    ","Предупреждение", MessageBoxButtons.YesNo,
    MessageBoxIcon.Warning,MessageBoxDefaultButton.Button2);
    switch (result){case DialogResult.Yes:
```

```

{ //выполнить действия по удалению данных по сотруднику
    MessageBox.Show("Удаление данных");
}
case DialogResult.No:
{
    //отмена удаления данных по сотруднику
    MessageBox.Show("Отмена удаления данных");
    break;
}

```

В результате исполнения кода приложения и выбора пункта меню "Удалить" будет выводиться предупреждение рис. 10.7.

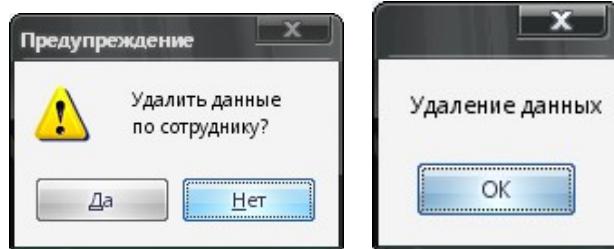


Рис. 10.7. Модальное диалоговое окно предупреждения

Диалоговое окно можно создать не только на основе класса MessageBox, но и с использованием Windows - формы.

- Создайте новую форму *FormAbout* для вывода справочной информации о разрабатываемом приложении, которое должно иметь вид, представленный на [рис. 10.8](#).

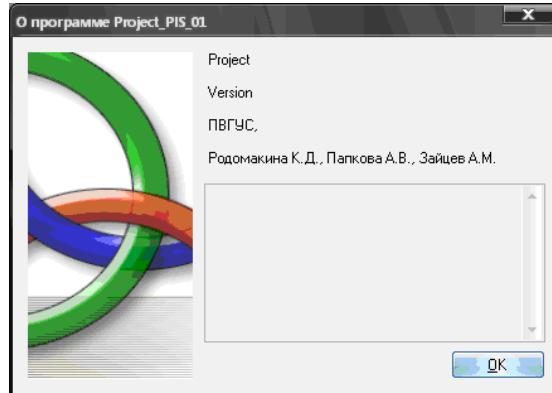


Рис. 10.8. Общий вид Windows - формы FormAbout

- Добавьте в проект новый компонент ([рис. 10.9.](#)), выбрав из списка AboutBox

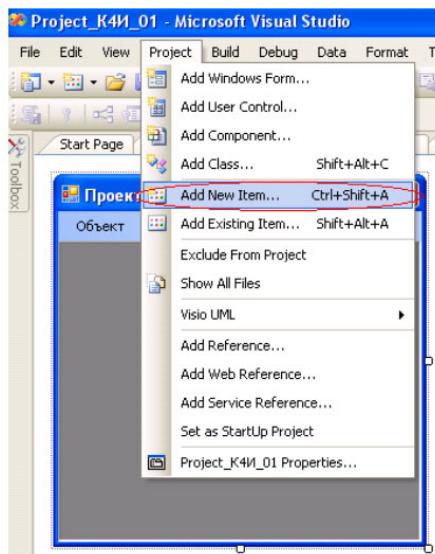


Рис. 10.9. Выбор режима добавления нового компонента в проект

Измените фрагмент кода конструктора класса AboutBox1 и введите собственную информацию.

4. Для класса AboutBox можно задайте логотип и дополнительную информацию. По умолчанию данный класс берет дополнительную информацию из метаданных сборки. Проверьте это.

5. Измените фрагмент кода конструктора класса AboutBox1 введем собственную информацию следующим образом:

```
public AboutBox1()
{
    InitializeComponent();

    this.Text = String.Format("О программе {0}", AssemblyTitle);
    this.labelProductName.Text = AssemblyProduct;
    this.labelVersion.Text = String.Format("Version {0}", AssemblyVersion);
    this.labelCopyright.Text = "ПВГУС, 2009";
    this.labelCompanyName.Text = "Иванченко Елена, БИ-401";
    this.textBoxDescription.Text = "Студенческий проект по курсу Проектирование ИС";
}
```

Для открытия пользовательского модального диалогового окна используется метод ShowDialog. В лабораторной работе диалоговое окно должно открываться при щелчке пользователем на пункте в меню "Справка / О программе". Код для открытия диалогового окна может выглядеть следующим образом:

```
// Открываем модальное диалоговое окно
private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    AboutBox1 aboutBox = new AboutBox1();
    aboutBox.ShowDialog(this); }
```

После компиляции и загрузки приложения, вызвав пункт меню «Справка» / «О программе» на дисплей будет выведено диалоговое окно, приведенное на рис. 10.10.

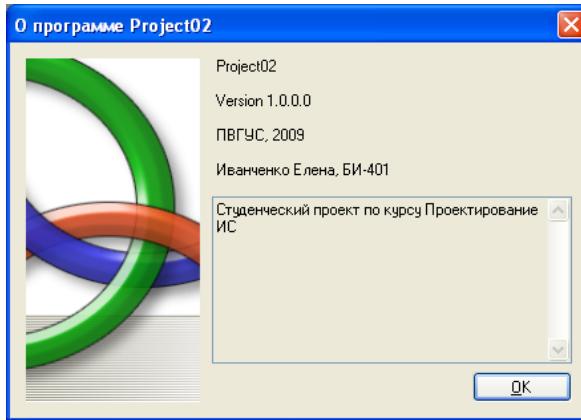


Рис. 10.10. Вызов модального окна

При нажатии на кнопку *OK* диалоговое окно будет автоматически закрыто и в ходе дальнейшего выполнения программы можно выяснить значение свойства *DialogResult*.

Задание 6.

Разработка панели инструментов

Элемент управления *ToolStrip* используется непосредственно для построения панелей инструментов. Данный элемент использует набор элементов управления, происходящих от класса *ToolStripItem*.

В *Visual Studio.NET* предусмотрены средства, которые позволяют добавить панель инструментов при помощи графических средств.

1. Откройте панель *Toolbox* и добавьте элемент управления *ToolStrip* на разрабатываемую форму *FormEmployee*.
2. Выберите элемент управления *button* – кнопка в выпадающем меню элемента управления *ToolStrip* на форме *FormEmployee* [рис. 10.11](#). При этом в панели инструментов добавится кнопка.

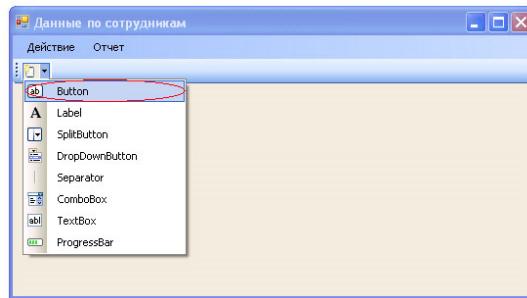


Рис.10.11. Окно свойств панели инструментов

3. Добавьте на панель инструментов кнопки с именами *toolStripButtonUndo*, *toolStripButtonNew*, *toolStripButtonEdit*, *toolStripButtonSave*, *toolStripButtonRemove*. В результате должна быть сформирована панель инструментов с кнопками [рис. 10.12](#).

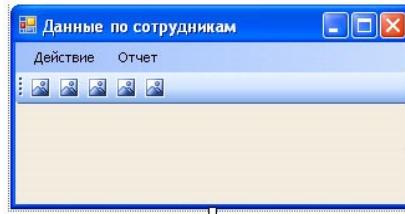


Рис. 10.12. Форма FormEmployee с панелью инструментов

4. Сформируйте графическое представление для кнопок панели инструментов. Задайте свойство *Image* соответствующей кнопке рис. 10.13.

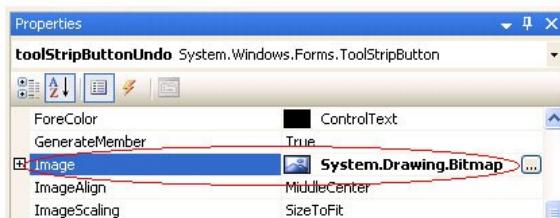


Рис. 10.13. Свойство *Image* для кнопки панели инструментов

5. При открытии коллекции свойства *Image* соответствующей кнопки, нажатии кнопки открывается окно мастера выбора графического ресурса
6. Добавьте ссылки на необходимые графические файлы с помощью кнопки *Import* в локальный ресурс, для формирования изображения кнопок. Результаты формирования графического представления кнопок панель инструментов.

Графические файлы расположены в папке Visual Studio 2005\VS2005ImageLibrary\bitmaps\commands\16color (для лабораторной работы графические файлы можно найти в папке Лабораторные работы).

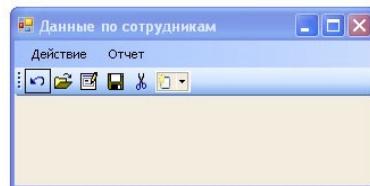


Рис. 10.14. Форма с панелью инструментов

Каждая кнопка панели инструментов, которая является объектом класса *toolStripButton*, может содержать текст, или изображение, или и то и другое.

Созданная панель инструментов содержит пять кнопок. По функциональности каждой из этих кнопок будут соответствовать пункты меню "Отменить", "Создать", "Редактировать", "Сохранить" и "Удалить".

Для удобства пользователя целесообразно снабдить кнопки панели инструментов всплывающими подсказками при фокусировке курсора на данной кнопке.

7. Задайте значение текстовой строки с содержанием подсказки свойству *ToolTipText* класса. На [рис.10.16](#). для кнопки "Отменить" (*toolStripButtonUndo*) строка подсказки *ToolTipText* соответствует строке "Отменить".

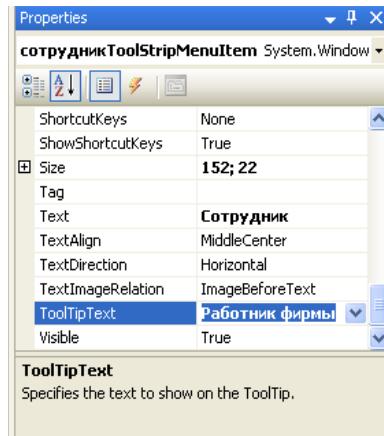


Рис. 10.15. Формирование подсказки для кнопки

На [рис.10.16](#) показан вывод подсказки при фокусировке курсора на кнопке панели инструментов.

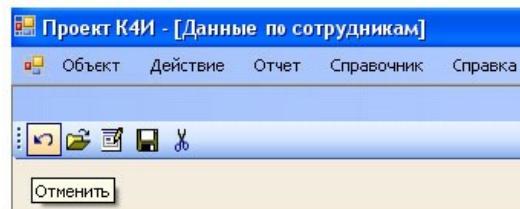


Рис. 10.16. Вывод подсказки для кнопки

- Создайте обработчик события для кнопок. Он необходим для распознавания реакции приложения при нажатии кнопок панели инструментов. При двойном нажатии на кнопку панели инструментов генерируется обработчик, в который нужно добавить вызов метода Undo. В этом случае обработчик нажатия кнопки панели инструментов будет иметь следующий вид:

```
private void toolStripButtonUndo_Click(object sender, EventArgs e)
{
    Undo();
}
```

Задание 7.

Создание контекстное меню

Класс ContextMenuStrip применяется для показа контекстного меню, или меню, отображаемого по нажатию правой кнопки мыши.

- Откройте панель *Toolbox* и добавьте элемент управления *contextMenuStrip* на форму *FormEmployee*. В результате получаем форму *FormEmployee* с контекстным меню.
- Сформируйте пункты контекстного меню. Формирование пунктов контекстного меню производится аналогично формированию пунктов главного меню [рис.10.17](#).

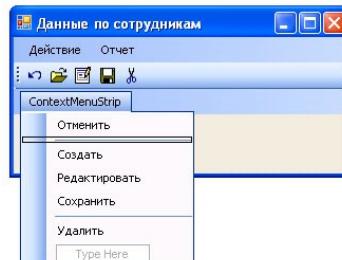


Рис. 10.17. Вид контекстного меню

- Установить значение созданного объекта `contextMenuStrip1` для подключения контекстного меню к форме `FormEmployee` на вкладке Свойства (*Properties*) строке, соответствующей свойству `ContextMenuStrip` рис.10.18.

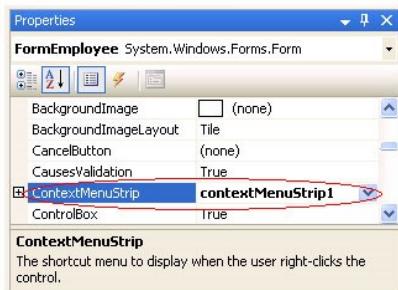


Рис. 10.18. Подключение контекстного меню к форме

Выполните компиляцию проекта и запустите приложения на выполнение, затем выберите из главного меню пункт "Сотрудник" и на появившейся форме щелкните правой кнопкой мыши. В результате на форме должно всплыть контекстное меню.

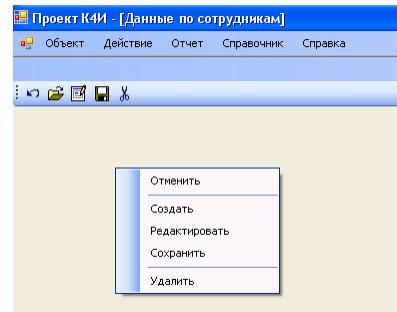


Рис. 10.19. Активизация контекстного меню

- Осуществите привязку пунктов контекстного меню к конкретным функциям путем создания кода обработчика событий для каждого пункта меню. Для формирования обработчика необходимо перейти в окно дизайнера формы `FormEmployee`, выделить на форме класс `ContextMenuStrip` и сделать двойной щелчок на соответствующем пункте меню, например "Отменить". В сгенерированном обработчике необходимо добавить вызов метода, для функции "Отменить" - метод `Undo()`. Листинг обработчика метода приведен ниже.

```
private void undo1ToolStripMenuItem_Click(object sender, EventArgs e)
{Undo();}
```

Содержание отчета

- Краткое описание создания главного меню и панели инструментов, подпунктов для пунктов главного меню.
- Описание создания дочернего окна, добавления в него пунктов меню, написание обработчиков для вызова из главного меню дочернего окна.
- Краткое описание создания модального диалогового окна, кодов методов-заглушек для функций приложения, обработчиков для вызова пунктов меню
- Тестирование работы приложения.
- Выводы по проделанной работе

Контрольные вопросы

- Элемент управления для создания меню?

2. Что такое диалоговое окно?
3. Что представляет собой элемент управления ToolStrip?
4. Для чего предназначен метод Undo?

Литература [3-4, 11-12].

Лабораторная работа №11

Разработка Windows-форм с элементами контроля в среде Microsoft Visual Studio. Net

Цель работы: Изучить основные элементы управления Windows-форм, их свойства и методы, а также получить практические навыки в разработке Windows-форм с элементами контроля

План проведения занятия

1. Изучить теоретический материал.
2. Для формы *FormEmployee* создать требуемые элементы контроля.
3. Разработать методы для задания режимов "Просмотр", "Редактирование" для элементов контроля, для управления активностью пунктов главного меню формы, контекстного меню и кнопок панели инструментов.
4. Сформировать обработчик события Load.
5. Протестировать программу.

Краткие теоретические сведения

ADO.NET - это новая технология доступа к базам данных, специально оптимизированная для нужд построения рассоединенных (*disconnected*) систем на платформе .NET. ориентирована на архитектуру многоуровневых приложений.

Подробно смотрите в приведенной ниже литературе.

Порядок выполнения работы

Задание 1.

Для формы *FormEmployee* создать требуемые элементы управления. Для разработки проекта приложения форма *FormEmployee* должна содержать элементы управления, в соответствии с видом, приведенным на [рис. 11.1](#).

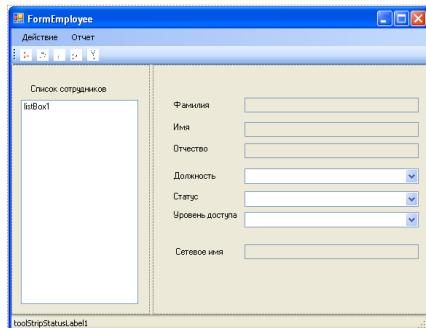


Рис. 11.1. Вид экранной формы FormEmployee

В этом задании сформируйте на данной форме элементы управления приведенные в табл.11.5.

Таблица 11.5

Элементы управления формы FormEmployee

Элемент контроля	Имя	Свойство Text/Items	Назначение
SplitContainer	splitContainerEmployee		Две панели с разделителем
Label	labelListEmployee	Список сотрудников	Надпись
listBox	listBoxEmployee		Список сотрудников
Label	labelSurname	Фамилия	Надпись
Label	labelName	Имя	Надпись
Label	labelPatronymic	Отчество	Надпись
Label	labelJobRole	Должность	Надпись
Label	labelStatus	Статус	Надпись
Label	labelAccess	Уровень доступа	Надпись
label	labelNetName	Сетевое имя	Надпись
textBox	textBoxNetName		Сетевое имя
textBox	textBoxSurname		Фамилия
textBox	textBoxName		Имя
textBox	textBoxPatronymic		Отчество
comboBox	comboBoxJobRole		Должность
comboBox	comboBoxStatus	Активен, выходной, в отпуске, болеет, не работает, помечен как удаленный	Статус
comboBox	comboBoxAccess	Оператор, старший оператор, начальник смены, администратор, аналитик	Уровень доступа
menuItem	menuItemAction	Действие	Пункт меню "Редактировать"
menuItem	menuItemUndo	Отменить	Подпункт меню "Отменить"
menuItem	menuItemNew	Создать	Подпункт меню "Новый"
menuItem	menuItemEdit	Изменить	Подпункт меню "Изменить"
menuItem	menuItemSave	Сохранить	Подпункт меню "Сохранить"
menuItem	menuItem	Удалить	Подпункт меню "Удалить"
menuItem	menuItemReport	Отчет	Пункт меню "Отчет"
menuItem	menuItemReport1	По сотруднику	Подпункт меню "Отчет по сотруднику"
menuItem	menuItemReport2	По всем сотрудникам	Подпункт меню "Отчет по

1. Создайте на форме элемент *SplitContainer* [рис. 11.2](#). На панели 1 создайте элементы управления *labelListEmployee* и *listBoxEmployee* [рис.11.3](#), а остальные элементы управления, приведенные в [табл. 11.5](#), - на панели 2 [рис. 11.1](#). После создания на форме *FormEmployee* элементов управления в соответствии табл. 11.1 необходимо настроить порядок перехода между ними при нажатии клавиши *Tab*.

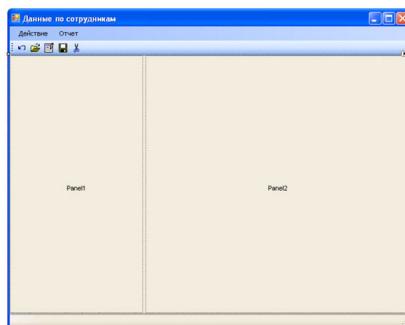


Рис. 11.2. Создание панелей на форме *FormEmployee*

Задайте последовательные номера свойству *TabIndex* элементов управления (в разрабатываемой форме это необходимо сделать для элементов управления *TextBox* и *ComboBox*) из окна *Properties* ([рис.11.4](#))

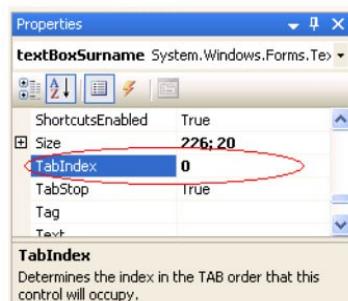


Рис. 11.4. Задание свойства *TabIndex* для элемента контроля

Так же для задания последовательных номеров свойству *TabIndex* элементов управления вызвать мастер *Tab Order Wizard* из меню *View/Tab Order* [рис.11.5](#). Задание последовательности значений свойству *TabIndex* производится щелчком мыши на элементах управления в заданной последовательности.

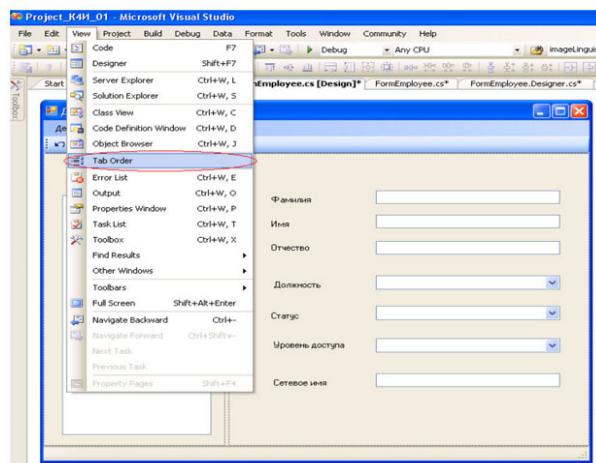


Рис. 11.5. Настройка перехода по элементам управления

Результат настройки порядка перехода между элементами управления при нажатии клавиши *Tab* приведен на [рис. 11.6](#).

2. Разработать методы для задания режимов "Просмотр", "Редактирование" для элементов контроля.

Создайте методы для работы с формой создайте методы, которые разрешают только просматривать форму (режим просмотра) и редактировать форму (режим редактирования).

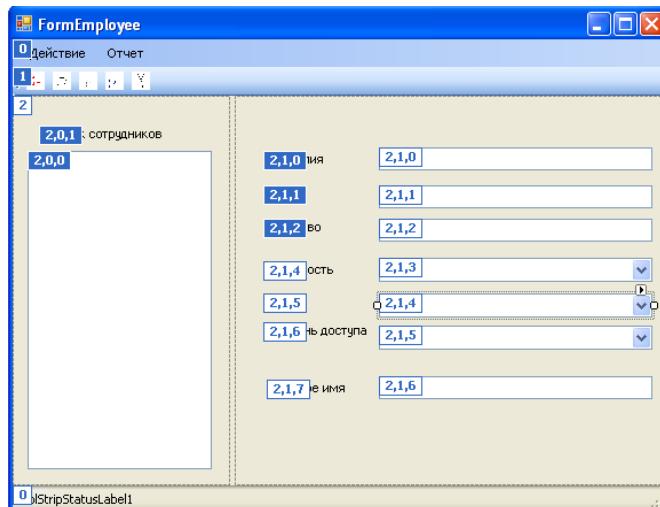


Рис. 11.6. Результат работы мастера Tab Order Wizard

3. Разработать методы для задания режимов "Просмотр", "Редактирование" для управления активностью пунктов главного меню формы, контекстного меню и кнопок панели инструментов.

Создадать метод для задания режима просмотра формы *DisplayReadOnly*. Метод *DisplayReadOnly* должен быть общедоступным, ничего не должен возвращать и не иметь параметров. Для задания режима просмотра (только для чтения) объекту класса *TextBox* свойству *ReadOnly* присвойте значение *true*, а для объекта класса *comboBox* - свойству *Enabled* значение *false*. Код метода *DisplayReadOnly* представлен далее:

```
public void DisplayReadOnly()
{
    this.textBoxSurname.ReadOnly = true;
    this.textBoxName.ReadOnly = true;
```

```
this.textBoxPatronymic.ReadOnly = true;
this.textBoxNetName.ReadOnly = true;
this.comboBoxJobRole.Enabled = false;
this.comboBoxStatus.Enabled = false;
this.comboBoxAccess.Enabled = false;
}
```

4. Сформируйте метод `DisplayEdit` аналогичным образом, который задает режим редактирования формы:

```
/// Задание режима редактирования
public void DisplayEdit()
{
    this.textBoxSurname.ReadOnly = false;
    this.textBoxName.ReadOnly = false;
    this.textBoxPatronymic.ReadOnly = false;
    this.textBoxNetName.ReadOnly = false;
    this.comboBoxJobRole.Enabled = true;
    this.comboBoxStatus.Enabled = true;
    this.comboBoxAccess.Enabled = true;
}
```

5. Сформировать обработчик события `Load`.

Для управления режимом доступности (только для чтения/редактирование) формы `FormEmployee` необходимо метод `DisplayReadOnly` вызывать при первоначальной загрузке формы (событие `Load`), при создании новых данных по сотруднику и при редактировании данных по сотруднику, а метод `DisplayEdit` - при сохранении данных по сотруднику и при отмене режима редактирования данных.

6. Проверьте правильность режима управления доступностью элементов управления формы `FormEmployee`.

Анализ кодов методов `DisplayReadOnly()` и `DisplayEdit()` показывает, что они могут быть объединены в один метод с параметром.

7. Необходимо самостоятельно написать объединенный метод, получив в результате метод `DisplayReadOnly(bool readOnly)`, в котором параметр `readOnly` определяет режим редактирования: если `readOnly` равен `true`, то режим только для просмотра, если равен `false`, то - редактирование.

В процессе работы приложения необходимо управлять доступом к пунктам меню формы `FormEmployee`.

При выборе в главном меню приложения пункта "Сотрудник" *Windows*-форма `FormEmployee` должна перейти в режим "Просмотр", что определяет доступ к пунктам меню "Создать", "Редактировать", "Удалить" и запрет доступа к подпунктам меню "Отменить", "Сохранить".

Если в режиме просмотр выбирается подпункт меню "Удалить", то в результате выполнения данной функции режим *Windows*-формы `FormEmployee` не должен измениться, т.е. форма должна оставаться в режиме "Просмотр".

Если в режиме просмотр выбирается подпункт меню "Изменить", то *Windows*-формы `FormEmployee` должна перейти в режим "Редактирование". Данный режим предполагает, что разрешается доступ к подпунктам меню "Отменить", "Сохранить" и запрещается доступа к подпунктам меню "Создать", "Редактировать", "Удалить".

Аналогичным образом интерпретируются переходы формы `FormEmployee` из одного режима в другой. Такие же режимы необходимо соблюдать для контекстного меню и кнопок панели инструментов.

Для управления доступом к пунктам главного меню создайте методы MenuItemEnabled(bool itemEnabled), для контекстного меню – MenuItemContextEnabled (bool itemEnabled) и для кнопок панели управления – StripButtonEnabled(bool itemEnabled).

Управление доступностью пунктов главного и контекстного меню осуществляется через свойство Enabled класса ToolStripMenuItem, а кнопок панели управления - через свойство Enabled класса ToolStripButton.

Проверьте правильность режима управления пунктов главного и контекстного меню, а также кнопок панели управления формы *FormEmployee*.

С учетом того, что установка режимов просмотра и редактирования экранной формы, а также управление доступом к пунктам меню должно выполняться при реализации нескольких функций программы целесообразно для избежания дублирования кода все методы управления режимами объединить в один метод DisplayForm.

```
private void DisplayForm(bool mode)
{
    DisplayReadOnly(mode);
    MenuItemEnabled(mode);
    MenuItemContextEnabled(mode);
    StripButtonEnabled(mode); }
```

Первоначальная установка режима "Просмотр" должна проводиться при первоначальной загрузке формы *FormEmployee*.

Содержание отчета

1. Создание требуемых элементов контроля для формы *FormEmployee*
2. Разработка методов для задания режимов "Просмотр", "Редактирование" для элементов контроля.
3. Разработка методов для задания режимов "Просмотр", "Редактирование" для управления активностью пунктов главного меню формы, контекстного меню и кнопок панели инструментов.
4. Формирование обработчиков события Load.
5. Тестирование программы.

Контрольные вопросы

1. Что представляет собой элемент управления TextBox ?
2. Для чего предназначен метод ButtonBase ?
3. Элемент управления CheckedListBox предназначен для?
4. Какой объект позволяет пользователю производить выбор заранее определенных элементов?
5. С помощью, каких объектов устанавливается дата?

Литература 3-4, 11-12].

Лабораторная работа №12

Работа с базой данных ИС средствами технологии ADO.NET

Цель работы: Изучить назначение и основные способы создания объектов ADO.NET при помощи Visual Studio IDE. Изучить основные приемы и способы отображения и связывания, данных объекта DataSet и элементов управления Windows - формы.

План проведения занятия

1. Изучить теоретический материал.
2. Создать класс DataSetEmployee и объекты dsEmployee, daJobTitle и daEmployee.
3. Разработать метод Fill для заполнения таблиц DataSet.
4. Осуществить привязку источника данных к элементам управления экранной формы.
5. Разработать необходимые методы для вывода информации из базы данных на экранную форму.
6. Протестировать программу.

Порядок выполнения работы

Задание 1.

Информация о базе данных

Разрабатываемое приложение предназначено для работы с базой данных сотрудников компании. На [рис. 12.1](#) представлена структура базы данных.

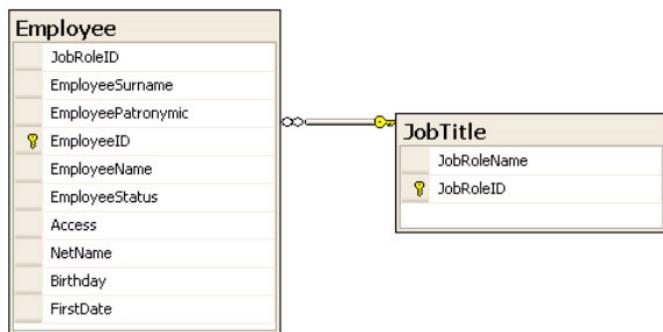


Рис. 12.1. Структура базы данных по сотрудникам компании

1. Создайте базу данных, включающую две таблицы:
 - сведения о сотрудниках - *Employee*;
 - справочник должностей - *JobTitle*.
2. Задайте значение атрибутам таблицы *Employee* приведенным в [табл. 12.2](#)

Таблица 12.2

Атрибуты таблицы Employee

Имя атрибута	Назначение	Тип
<i>EmployeeID</i>	Суррогатный ключ	smallint

<i>JobRoleID</i>	Внешний ключ	smallint
<i>EmployeeSurname</i>	Фамилия	varchar(50)
<i>EmployeeName</i>	Имя	varchar(20)
<i>EmployeePatronymic</i>	Отчество	varchar(20)
<i>EmployeeStatus</i>	Статус	int
<i>Access</i>	Уровень доступа	varchar(20)
<i>NetName</i>	Сетевое имя	varchar(20)
<i>Birthday</i>	Дата рождения	Smalldatetime
<i>FirstDate</i>	Дата приема на работу	smalldatetime

Суррогатный ключ EmployeeID, как и все остальные суррогатные ключи базы данных, генерируется сервером базы данных автоматически, т.е. для него задано свойство IDENTITY для СУБД *MS SQL Server* или AutoNumber для *MS Access*. Атрибут JobRoleID является внешним ключом, с помощью которого осуществляется связь с табл. й *JobTitle*.

Назначение атрибутов таблицы *JobTitle* приведено в [табл. 12.3](#).

Таблица 12.3

Атрибуты таблицы *JobTitle*

Имя атрибута	Назначение	Тип
<i>JobRoleID</i>	Суррогатный ключ	smallint
<i>JobRoleName</i>	Наименование должности	varchar(50)

В рассматриваемом приложении в качестве СУБД используется *MS SQL Server 2005*.

3. Создайте соединение проекта с базой данных. Для этого выберите пункт меню *Tools/Connect to Database*. Появляется окно *AddConnection* [рис.12.2](#).

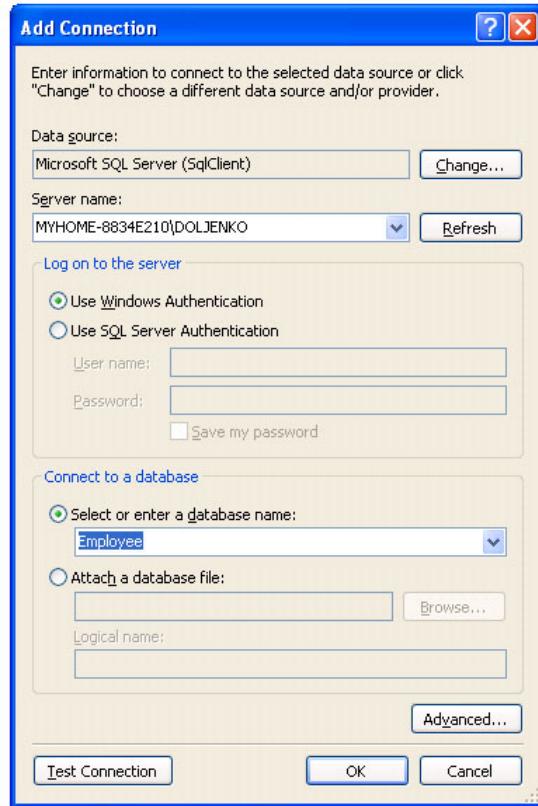


Рис. 12.2. Окно AddConnection

4. В пункте "Server name" задайте имя сервера, которое необходимо узнать у преподавателя (на [рисунок 12.2](#) MYHOME-8834E210\DOLJENKO). В пункте Select or enter database name - имя базы данных, которое определит преподаватель [рис. 2.2](#).
5. Для проверки правильности подключения к базе данных нажимаем клавишу "Test Connection". При правильном подключении появляется следующее сообщение [рис.12.3.](#)



Рис. 12.3. Окно Microsoft Data Link

При нормальном соединении с базой данных можно открыть навигатор Server Explorer из меню View/ Server Explorer или сочетанием клавиш ALT+CTRL+S ([рисунок 12.4.](#)).



Рис. 12.4. Окно навигатора Server Explorer

6. Добавьте в проект объект класса DataSet. Для этого выберем пункт меню *Project/Add New Item ...* (рис. 12.5).

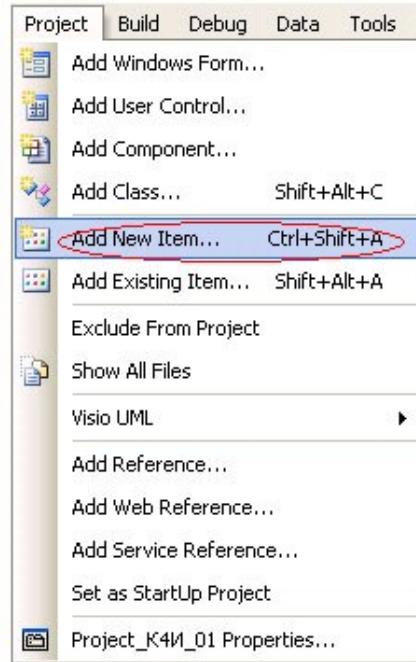


Рис. 12.5. Добавление в проект нового компонента

7. Выберете шаблон *DataSet* в окне *Add New Item* (рис. 12.6) и присвойте ему имя *DataSetEmployee*.

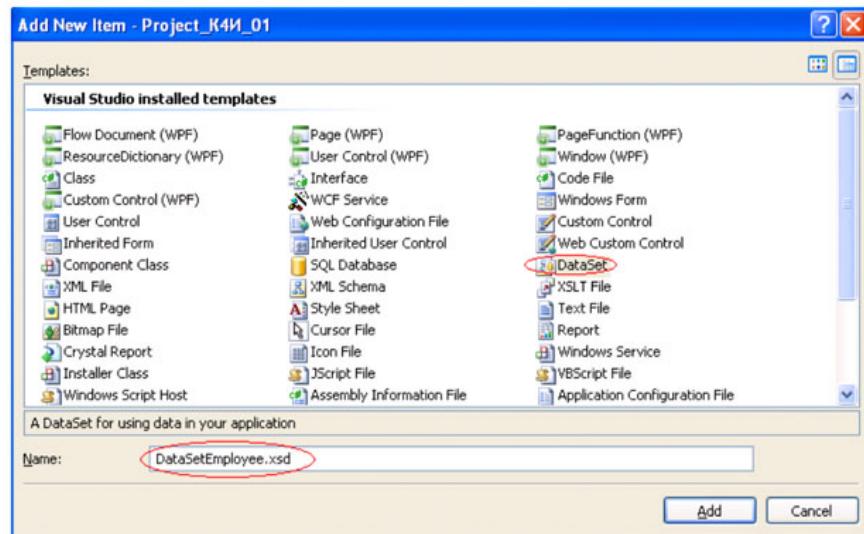


Рис. 12.6. Выбор нового компонента – DataSet

После нажатия кнопки *Add* система генерирует класс *DataSetEmployee*, который добавляется в решение проекта (рис. 12.7).

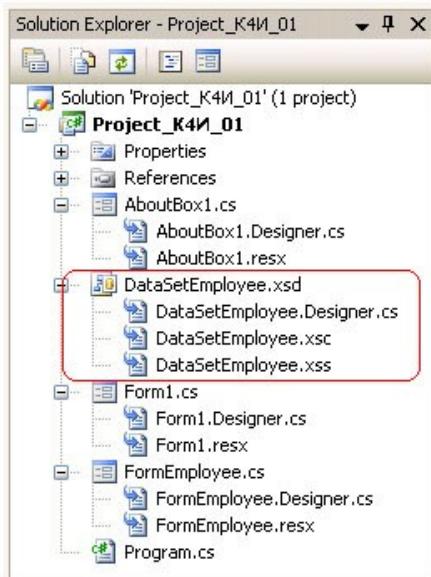


Рис. 12.7. Окно решения проекта с новым компонентом DataSet

8. Для добавления таблиц *Employee* и *JobTitle* к *DataSet* необходимо перетащить их из окна *Server Explorer* на поле графического дизайнера [рис.12.8](#).

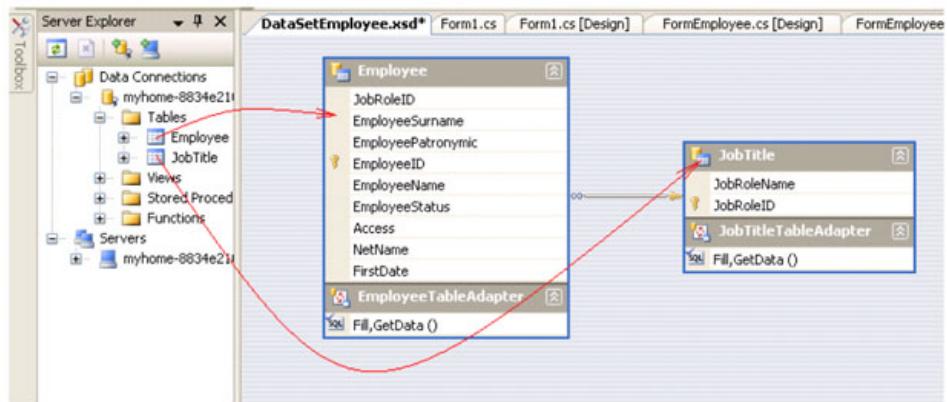


Рис. 12.8. Добавление таблиц к DataSet

В результате будут созданы классы таблиц, адаптеры и методы *Fill* и *GetData*.

При формировании класса *DataSetEmployee* необходимо учесть то, что первичные ключи таблиц *Employee* и *JobTitle* являются суррогатными и автоматически формируются (ключ со свойством *автоинкремент*) источником данных (например, *MS SQL Server*).

9. При формировании новых записей в приложении необходимо обеспечить уникальность первичных ключей для таблиц объекта *DataSetEmployee*. Это можно обеспечить, задав для ключевых колонок таблиц *Employee* и *JobTitle* следующие свойства:

```
AutoIncrement = true;
AutoIncrementSeed = -1;
AutoIncrementStep = -1;
```

Столбец со свойством *AutoIncrement* равным *true* генерирует последовательность значений, начинающуюся со значения *AutoIncrementSeed* и имеющую шаг *AutoIncrementStep*. Это позволяет генерировать уникальные значения целочисленного столбца первичного ключа. В этом случае при добавлении новой записи в таблицу будет генерироваться новое значение первичного ключа, начиная с *-1*, *-2*, *-3* и т.д., которое никогда не совпадет с

первичным ключом источника данных, т.к. в базе данных генерируются положительные первичные ключи. Свойства AutoIncrementSeed и AutoIncrementStep устанавливаются равными -1, чтобы гарантировать, что когда набор данных будет синхронизироваться с источником данных, эти значения не будут конфликтовать со значениями первичного ключа в источнике данных. При синхронизации DataSet с источником данных, когда добавляют новую строку в таблицу *MS SQL Server 2005* с первичным автоинкрементным ключом, значение, которое этот ключ имел в таблице *DataSet*, заменяется значением, сгенерированным СУБД.

10. Установите свойства AutoIncrement, AutoIncrementSeed и AutoIncrementStep для колонки первичного ключа EmployeeID таблицы *Employee* [рис. 12.9](#).

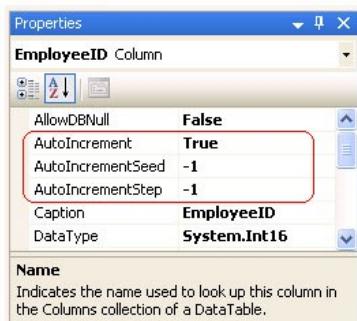


Рис. 12.9. Установка свойств для колонки EmployeeID

11. Сделайте для колонки JobTitleID таблицы *JobTitle*. установки свойств аналогичные AutoIncrement, AutoIncrementSeed и AutoIncrementStep
12. После создания класса DataSetEmployee и адаптера необходимо создать объекты этих классов, добавив следующий код к файлу FormEmployee.cs.

```
DataSetEmployee dsEmployee = new DataSetEmployee();
DataSetEmployeeTableAdapters.EmployeeTableAdapter daEmployee = new
Project_K4И_01.DataSetEmployeeTableAdapters.
EmployeeTableAdapter();
DataSetEmployeeTableAdapters.JobTitleTableAdapter daJobTitle = new
Project_K4И_01.DataSetEmployeeTableAdapters.
JobTitleTableAdapter();
```

13. Создайте метод для заполнения объекта dsEmployee из базы данных , после того, как созданы объекты адаптеров данных daEmployee и daJobTitle, а также объект класса DataSetEmployee - dsEmployee(в рассматриваемом примере база данных Employee, созданная в СУБД *MS SQL Server 2005*). Для заполнения данными dsEmployee из базы данных Employee создадим метод EmployeeFill():

```
public void EmployeeFill()
{
    daJobTitle.Fill(dsEmployee.JobTitle);
    daEmployee.Fill(dsEmployee.Employee);
    MessageBox.Show("Метод Fill отработал");}
```

В методе EmployeeFill() для объектов класса DataAdapter применяется метод Fill, который производит заполнение таблиц (*JobTitle* и *Employee*) объекта dsEmployee данными из базы данных. Метод Fill адаптера данных DataAdapter требует указания в качестве

параметров задания соответствующей таблицы *DataSet*, то есть *dsEmployee.JobTitle* и *dsEmployee.Employee*.

Метод *MessageBox.Show* введен в метод *EmployeeFill* для первоначального тестирования, после которого его нужно убрать.

Вызов метода *EmployeeFill* необходимо добавить в обработчик события *Load* для формы *FormEmployee*, возникающего при нажатии на пункт меню "*Сотрудник*".

Задание 2.

1. Свяжите элемент контроля *listBoxEmployee*, в котором должен отображаться список фамилий сотрудников, со столбцом *EmployeeSurname* таблицы *Employee*. Это можно сделать, добавив следующие строки кода в метод загрузки формы *FormEmployee_Load*.

```
this.listBoxEmployee.DataSource = this.dsEmployee1;  
this.listBoxEmployee.DisplayMember = "Employee.EmployeeSurname";
```

В разрабатываемом приложении на *Windows*-форме *FormEmployee* имеются четыре текстовых поля: *textBoxSurname*, *textBoxName*, *textBoxPatronymic* и *textBoxNetName*. Эти текстовые поля предназначены для отображения информации из одной записи таблицы *Employee* набора данных *dsEmployee*. Для того чтобы содержимое текстовых полей автоматически обновлялось при смене записи, их необходимо связать с соответствующими колонками набора данных *dsEmployee*. Связывание можно осуществить, используя свойство *DataBindings* элемента управления *TextBox*.

2. Создайте связь для элемента управления *textBoxSurname* с источником данных, добавив следующую строку кода в метод загрузки формы *FormEmployee_Load*.

```
textBoxSurname.DataBindings.Add("Text", dsEmployee,  
"Employee.EmployeeSurname");
```

3. Протестируйте добавленный в программу код.

4. Аналогично свяжите текстовые поля *textBoxName*, *textBoxPatronymic* и *textBoxNetName* с источником данных.

В таблице *Employee* значения для атрибута *Access* (доступ) задается в виде символьной строки, значение которой выбирается из списка элемента управления *comboBoxAccess*.

5. Задайте коллекцию выпадающего списка элемента управления *comboBoxAccess* можно задать следующей строкой кода:

```
this.comboBoxAccess.Items.AddRange(new object[] {"не задан",  
"администратор", "начальник смены", "старший оператор", "оператор",  
"аналитик"});
```

6. Для заданной записи источника данных (таблица *Employee*) значение столбца *Access* отобразите в элементе контроля *comboBoxAccess*. Сделайте это аналогично тому, как это делалось для элементов управления *TextBox*, задавая свойство *DataBindings*

7. Сформируйте коллекцию выпадающего списка: не задан, активен, выходной, в отпуске, болеет, не работает, помечен как удаленный. Для элемента управления *comboBoxStatus*

В таблице *Employee* значения для атрибута *EmployeeStatus* (статус) задается в виде целого числа (0, 1, 2, 3, 4, 6), однако статус сотрудника должен отображаться в элементе управления *comboBoxStatus* в виде строковых значений в соответствии со значениями его коллекции. В программе необходимо реализовать отображение целочисленных данных из *DataSet* в текстовые значения в элементе контроля *comboBoxStatus*. Для этого необходимо отслеживать изменение позиции в табл. источника данных *dsEmployee* и в соответствии

со значением столбца EmployeeStatus активизировать требуемый элемент (Item) списка comboBoxStatus.

8. Объявите объект bmEmployee класса BindingManagerBase в форме *ormEmployee*:
BindingManagerBase bmEmployee;
9. Создайте объект bmEmployee в конструкторе класса FormEmployee, применяя индексатор контента BindingContext включив в него связывание с таблицей Employee, и добавьте делегат для события, которое формируется при изменении позиции в данной таблице:

```
public FormEmployee( )  
{InitializeComponent();  
    bmEmployee = this.BindingContext[dsEmployee1, "Employee"];  
    // Добавляем делегата PositionChanged для события - изменение  
    //позиции в табл. Employee DataSet dsEmployee  
    bmEmployee.PositionChanged = new  
    EventHandler(BindingManagerBase_PositionChanged);}
```

10. Создайте обработчик для сформированного события, который на основе выбранной строки (pos) таблицы Employee будет задавать свойству Text списка comboBoxStatus значение из коллекции Items по индексу (sel), полученному из столбца EmployeeStatus Employee.

```
private void BindingManagerBase_PositionChanged(object sender, EventArgs e)  
{  
    int pos = ((BindingManagerBase)sender).Position;  
    int sel = (int)dsEmployee.Employee[pos].EmployeeStatus;  
    this.comboBoxStatus.Text = this.comboBoxStatus.Items[sel].ToString();}
```

11. Протестируйте добавленный в программу код.

12. Для задания в элементе контроля comboBoxJobRole должности сотрудника необходимо получить данные из родительской таблицы JobTitle, с которой таблица Employee связана внешним ключом JobRoleID. Фактически необходимо осуществить вывод данных из справочника (таблица JobTitle) по данным в основной табл. Employee.

Свойство DataBindings объекта ComboBox предоставляет доступ к коллекции ControlBindingsCollection. Метод Add этой коллекции добавляет в неё привязку.

Перегруженный вариант метода Add принимает три аргумента:

PropertyName - имя свойства элемента управления, к которому осуществляется привязка;

DataSource - имя привязываемого источника данных;

DataMember - имя свойства привязываемого источника данных.

13. Свяжите элемент контроля comboBoxJobRole с набором данных JobTitle (родительская таблица - справочник), в соответствии с кодом, приведенным ниже.

```
comboBoxJobRole.DataSource = this.dsEmployee.JobTitle;  
comboBoxJobRole.DisplayMember = "JobRoleName";  
comboBoxJobRole.ValueMember = "JobRoleID";
```

14. Свяжите comboBoxJobRole с полем JobRoleID основной таблицы Employee (дочерняя таблица), в соответствии со следующим кодом

```
comboBoxJobRole.DataBindings.Add("SelectedValue",  
    dsEmployee, "Employee.JobRoleID");
```

После компиляции и запуска приложения экранная форма будет иметь вид, приведенный на [рис. 12.10](#).

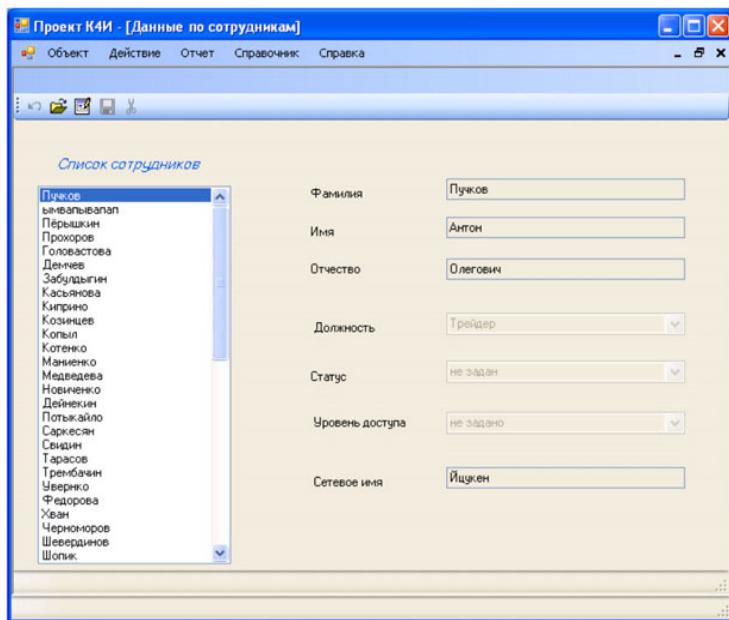


Рис. 12.10. Экранная форма в режиме просмотра (только для чтения)

Содержание отчета

- Описание процесса создание класса DataSetEmployee , объектов dsEmployee, daJobTitle и daEmployee, метода Fill.
- Описание создания базы данных, включающей две таблицы и соединение проекта с базой данных, формирование новых записей в приложении.
- Создание метода для заполнения объекта dsEmployee из базы данных.
- Описание процесса связывания элемента контроля listBoxEmployee, в котором должен отображаться список фамилий сотрудников, со столбцом EmployeeSurname таблицы Employee.
- Выводы по проделанной работе

Контрольные вопросы

- Что представляет собой ADO.NET?
- Для чего используется класс BindingContext?
- Для выполнения, каких задач предназначены все типы ADO.NET?
- Что представляет собой функция Data.Set?
- Какие уровни объектной модели ADO.NET можно выделить?

Литература [3-4, 11-12].

Примерные темы лабораторных работ

Вариант 1. Видеопрокат

Тема: Проектирование ИС. Программное обеспечение видеопроката

Пункт проката видео нуждается в компьютерной системе. Его ассортимент составляет около тысячи видеокассет и пятьсот видеодисков. В прокате имеются видеодиски разных форматов: DVD, MPEG4, Blu-Ray, HD-DVD. Фильмы закупаются у разных поставщиков. Обычно один заказ поставщику делается на несколько фильмов. База данных хранит обычную информацию о поставщиках: их адреса, телефонные номера и т. д. В каждом заказе поставщику указывается: перечень фильмов; их количество, форматы кассет/дисков; отпускная цена.

Каждый видеоноситель при поступлении от поставщика снабжается штрих-кодом (содержащим уникальный идентификационный номер) для того, чтобы сканер, интегрированный в систему, мог поддерживать операции выдачи и возврата видеофильмов.

Каждому клиенту при первом обращении в видеопрокат выдается клиентская карточка со штрих-кодом для автоматизации обработки его запросов. Данные о клиенте (ф. и. о., телефон, адрес) заносятся в базу данных.

При выдаче фильма в прокат устанавливается конкретный период проката (исчисляемый в днях). Плата за прокат вычисляется как произведение количества дней на цену одного дня проката. Цена зависит от видеоносителя: кассета или диск; формата диска. Плата за прокат взимается в момент выдачи. За кассеты и диски, возвращенные позже срока, взимается дополнительная плата за период, превышающий срок проката. Если кассета/диск задержаны более чем на два дня, клиента ежедневно уведомляют о задержке. После двух уведомлений о задержке одной и той же кассеты/диска, клиент заносится в список нарушителей. При следующем его обращении в видеопрокат работник проката решает: оставить клиента в списке нарушителей и отказать в обслуживании или удалить из списка нарушителей и обслужить. При порче видеоносителя клиентом с него взимается штраф.

Система должна обладать поисковым механизмом по базе видео. Работники проката должны иметь возможность быстро получить ответ, имеется ли фильм в наличии, в каком количестве и на каких носителях. Если все носители фильма выданы в прокат, то система сообщить ближайшую дату возврата.

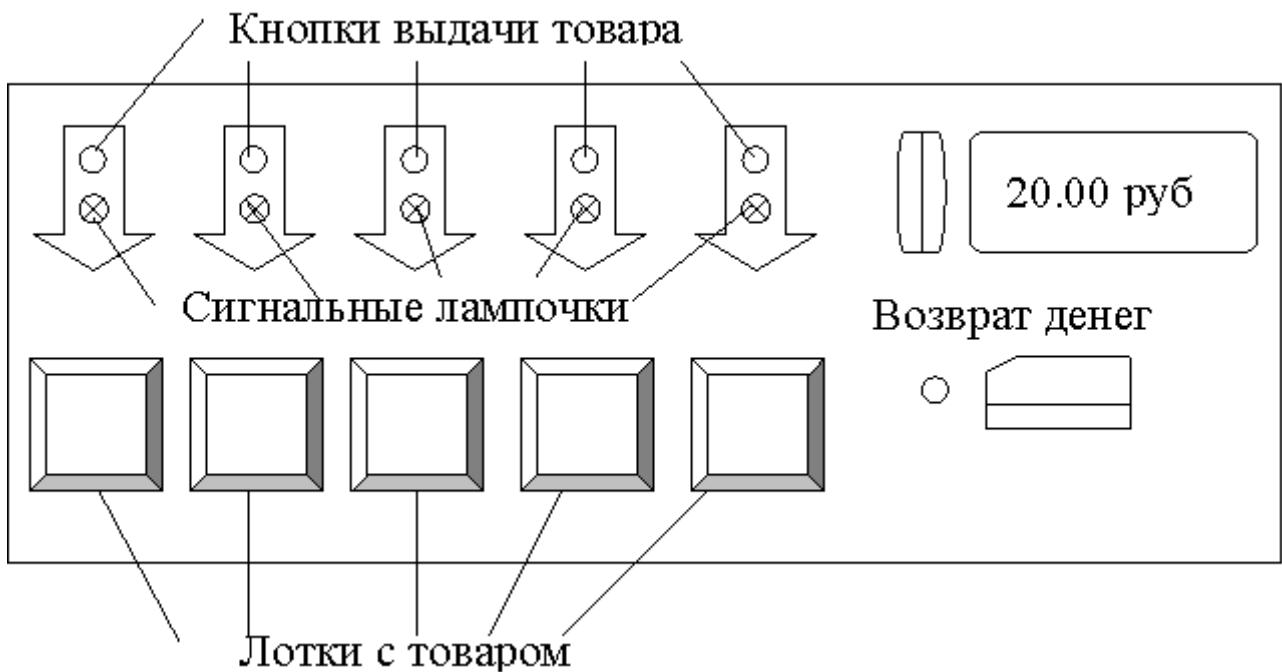
Постоянные клиенты (к ним относятся те, кто более десяти раз в течении 12 месяцев воспользовался услугами проката) могут оставлять заявки на фильмы, которых нет в прокате и которые не заказаны у поставщика. Фильмы из таких заявок включаются в следующий заказ поставщику, и в момент поступления фильмов от поставщика клиенты уведомляются о выполнении их заявок. Данные о выполненных заявках хранятся в течение 12 месяцев, после чего удаляются.

Клиенту одновременно могут быть выданы несколько кассет или дисков, однако каждому взявшему видеоноситель ставится в соответствие отдельная запись. Для каждого выдаваемого напрокат фильма фиксируются дата и время выдачи, стоимость проката, установленный и фактический срок возврата. При возврате запись о покате обновляется, чтобы отразить этот факт. Кроме того, запись хранит информацию о работнике, оформившем прокат. Записи хранятся в течение 12 месяцев, после чего удаляются.

Вариант 2. Торговый автомат

Тема: Проектирование ИС. Программное обеспечение торгового автомата

В автомате имеется пять лотков для хранения и выдачи товаров.



Внешний вид автомата изображен на рисунке.

Загрузка товаров на лотки осуществляется обслуживающим персоналом. Автомат следит за наличием товара. Если какой-либо товар распродан, автомат отправляет сообщение об этом на станцию обслуживания по линии связи и информирует покупателей (зажигается красная лампочка рядом с лотком данного товара).

Автомат принимает к оплате бумажные купюры и монеты. После ввода денег клиент выбирает товар нажатием на кнопку нужного лотка и нажимает на кнопку выдачи товара. Выдача товара производится только в том случае, если товар имеется в наличии и если введенная сумма денег не меньше цены товара. Если сумма превышает цену, клиенту выдается сдача. Товар выдается поштучно.

При нажатии на кнопку "Возврат" клиенту возвращаются все принятые от него к оплате деньги. Возврат денег не производился после выдачи товара. Автомат должен корректно работать при одновременном нажатии на кнопки выдачи товара и возврата денег.

На автомате имеется информационное табло, на котором высвечивается текущая сумма денег, принятых автоматом к оплате, и сообщения для клиентов, такие как: "введите деньги", "выберите товар", "нажмите кнопку выдачи", "введенной суммы недостаточно", "товара нет в наличии", "заберите покупку", "заберите сдачу", "заберите деньги".

В специальном отделении автомата, закрываемом замком, есть сервисная консоль, которая используется обслуживающим персоналом. С консоли производится управление доступом к ящику с деньгами для изъятия выручки, управление доступом к товарным лоткам для загрузки или замены товара, а также ввод данных о товарах на лотках в память автомата. Данные включают в себя цену, наименование товара, номер лотка, на котором находится товар и количество товара на лотке. Вариант задания включает в себя разработку схемы базы данных о товарах.

Вариант 3. Табло на станции метро

Тема: Проектирование ИС. Программное обеспечение табло на станции метро

Табло расположены на каждой станции метро. Они работают под управлением единого пункта управления (ПУ) информационной службы метро. Табло отображает текущее время (часы, минуты, секунды) и время, прошедшее с момента отправления последнего поезда (минуты, секунды). Момент прибытия и отправления поезда определяется при помощи датчиков, устанавливаемых на путях. Все табло метро синхронизованы, текущее время отсчитывается и устанавливается из центральной службы времени, находящейся на ПУ.

На табло высвечивается конечная станция назначения прибывающего поезда. Эти данные содержатся в расписании движения поездов, которое хранится в памяти табло и периодически обновляется с ПУ.

В "бегущей строке" табло отображается рекламная информация. Память табло хранит до 10 рекламных сообщений. Сообщения отображаются друг за другом с небольшими паузами, циклически. Содержание рекламных сообщений поступает с ПУ.

Дополнительная функция табло – по запросу с ПУ оно пересыпает данные о нарушениях расписания (преждевременных отправлениях поездов или опозданиях).

В ходе выполнения задания должна быть создана схема базы данных для хранения рекламных сообщений, расписания и сведений о нарушении расписаний.

Пояснение: в задании требуется разработать модель ПО только для табло, но не для пункта управления информационной службы.

Вариант 4. Онлайновая театральная касса

Тема: Проектирование ИС. Программное обеспечение онлайновой театральной кассы

Онлайновая театральная касса "Билетов.Нет" представляет собой web-сайт службы бронирования и доставки билетов на спектакли и концерты.

Перед тем как впервые воспользоваться услугами кассы клиент должен зарегистрироваться. В ходе регистрации он указывает данные о себе (ф. и. о., телефон, адрес электронной почты) и получает логин и пароль (логины и пароли разных клиентов не должны совпадать).

Войдя в систему, клиент может ознакомиться с афишами, выбрать интересующее его мероприятие, указав название, дату и место проведения. Получив от системы сведения о билетах имеющихся в наличии, пользователь может забронировать нужное ему количество билетов. Билеты бывают разных типов: партер, балкон, ложа, бельэтаж, 1-2-3 ярус, vip-места и т. п. Цена билета зависит от его типа и расположения зрительского места. Билеты могут быть выкуплены в течение трех суток с момента бронирования, но не позднее пяти суток до начала спектакля. Клиент может самостоятельно выкупить забронированные билеты, приехав в офис, или заказать доставку билетов курьером, сделав пометку в заявке и указав адрес доставки. Стоимость доставки зависит от дальности: центр / спальный район / дальний пригород. Клиент может получить информацию обо всех своих заявках с web-страницы онлайновой кассы.

Заявки клиентов хранятся в системе. В каждой указаны: сведения о клиенте, название спектакля, место и время проведения, количество и тип забронированных билетов, стоимость билетов, время создания заявки, время оплаты, вид доставки (самовывоз / курьер), адрес доставки, стоимость доставки, статус заявки (новая / рабочая / оплаченная / аннулированная). По истечении 12 месяцев с момента создания заявки данные автоматически удаляются из системы.

В обязанности работников онлайновой кассы входит внесение в систему сведений о мероприятиях и об имеющихся в продаже билетах. Данные о мероприятии – вид: концерт / шоу / спектакль; название; описание; место проведения; дата; – хранятся

в системе. Один и тот же спектакль может идти в разные дни и в разных местах, но разные спектакли не могут пересекаться по времени и месту проведения. Запись о билете содержит название спектакля, дата, время, место проведения, тип билета, зрительский ряд, зрительское место, цену билета, статус билета (есть в наличии / забронирован / продан / передан для реализации). По истечении 12 месяцев с даты, указанной в билете, данные автоматически удаляются из системы.

Работник кассы, получив новую заявку клиента, связывается с ним для подтверждения и уточнения места. Согласовав с клиентом зрительские места, работник делает пометку о бронировании билетов в системе (тем самым уменьшается количество билетов, имеющихся в наличии) и меняет статус заявки на "рабочая". После оплаты и/или доставки "рабочей" заявки билеты из заявки помечаются как проданные, а заявка – как оплаченная. За 5 суток до начала спектакля все не проданные билеты передаются для реализации в обычные кассы, в системе они автоматически помечаются как "передан для реализации", заявки на них аннулируются, клиенты, не успевшие оплатить заказанные билеты, информируются о снятии брони. Через 4 суток после создания "рабочие" заявки автоматически аннулируются, бронирование с билетов снимается, клиентам посыпается соответствующее сообщение. Также должна быть возможность аннулирования заявок вручную работниками онлайновой кассы. При аннулировании заявки вручную работник должен уведомить клиента, изменить статус заявки, снять бронирование билетов (количество билетов в наличии возрастает).

Вариант 5. Мини-АТС

Тема: Проектирование ИС. Программное обеспечение мини-АТС

Мини-АТС осуществляет связь между служащими учреждения. Каждый абонент подключен к ней линией связи. Мини-АТС соединяет линии абонентов (осуществляет коммутацию линий). Абоненты имеют номера, состоящие из трех цифр. Специальный номер "**9**" зарезервирован для внешней связи.

Телефонное соединение абонентов производится следующим образом. Абонент поднимает трубку телефона, и мини-АТС получает сигнал "**Трубка**". В ответ мини-АТС посылает сигнал "**Тон**". Приняв этот сигнал, абонент набирает телефонный номер (посыпает три сигнала "**Цифра**"). Мини-АТС проверяет готовность вызываемого абонента. Если абонент не готов (его линия занята), мини-АТС посылает вызывающему абоненту сигнал "**Занято**". Если абонент готов, мини-АТС посылает обоим абонентам сигнал "**Вызов**". При этом телефон вызываемого абонента начинает звонить, а вызывающий абонент слышит в трубке длинные гудки. Вызываемый абонент снимает трубку, и мини-АТС получает от него сигнал "**Трубка**", после чего осуществляет коммутацию линии. Абоненты обмениваются сигналами "**Данные**", которые мини-АТС должна передавать от одного абонента к другому. Когда один из абонентов опускает трубку, мини-АТС получает сигнал "**Конец**" и посылает другому абоненту сигнал "**Занято**". В любой момент разговора абонент может положить трубку, при этом мини-АТС получает сигнал "**Конец**". После получения этого сигнала сеанс обслуживания абонента завершается.

Если вызываемый абонент не подходит к телефону, то вызывающий абонент может, не дождавшись, повесить трубку. В этом случае мини-АТС получает сигнал "**Конец**" и завершает сеанс. Вызываемому абоненту посыпается сигнал "**Сброс**" для отмены вызова.

Если абонент желает соединиться с абонентом за пределами учреждения, то он набирает номер "**9**". Мини-АТС посылает по линии, соединяющей с внешней (городской) АТС, сигнал "**Трубка**" и в дальнейшем служит посредником между телефоном абонента и внешней АТС. Она принимает и передает сигналы и данные между ними, не внося никаких изменений. При завершении сеанса, получив от внешней

АТС сигнал "**Занято**" (в случае если вызываемый абонент первым повесил трубку), мини-АТС посыпает абоненту сигнал "**Занято**", ждет сигнала "**Конец**" для завершения обслуживания абонента и передает его внешней АТС. Если вызывавший абонент первым вешает трубку, то мини-АТС получает сигнал "**Конец**" и передает его городской АТС и завершает сеанс. Мини-АТС может получить сигнал "**Вызов**" от городской АТС. Это происходит, когда нет соединений с внешними абонентами. Сигнал "**Вызов**" от городской АТС передается абоненту с кодом "**000**". Только этот абонент может отвечать на внешние звонки. Если абонент "**000**" долго не отвечает на внешний вызов, от городской АТС может прийти сигнал "**Сброс**". Он передается абоненту "**000**", и сеанс завершается.

Вариант 6. Управление контактами с клиентами

Тема: Проектирование ИС. Программное обеспечение для управления контактами с клиентами

Компания, поставляющая оборудование, в рамках обеспечения своей коммерческой деятельности нуждается в системе управления контактами со своей клиентурой. Клиенты делятся на два вида: текущие – те, с которыми у компании заключены договора в текущий момент или ранее, и потенциальные.

Система управления контактами находится в распоряжении всех работников компании. Система поддерживает функции "постоянного контакта" с наличной и потенциальной клиентской базой, так, чтобы откликаться на ее нужды, получать новые контракты, обеспечивать выполнение старых. Система позволяет сотрудникам планировать задания, которые необходимо провести в отношении контактных лиц. Некоторые сотрудники должны иметь доступ к планированию заданий только для себя, другие – и для других сотрудников, и для себя.

Система хранит имена, номера телефонов и факсов, почтовые и электронные адреса и т. д. организаций и контактных лиц в этих организациях.

Каждое задание связано с каким-либо контактным лицом. Примерами заданий являются телефонный звонок, визит, отправка факса, отправка электронного сообщения, проведение презентации и т. д. Некоторые задания связаны с выполнением контракта, например, отправка оборудования, поставка, установка, гарантийный и послегарантийный ремонт. В таких заданиях указывается необходимая информация: номер контракта, серийный номер ремонтируемого оборудования. Каждое задание имеет дату создания – время внесения ее в систему. Некоторые задания имеют срок исполнения – период времени от начальной даты до финальной, другие являются бессрочными. Дата создания задания не может изменяться, а срок исполнения – может. По исполнении задания дата и время его завершения фиксируются.

Каждое задание имеет автора – сотрудника, который его создал. Исполнителем задания может быть сотрудник, не являющийся автором. Рядовые сотрудники не могут назначать задания кому-либо кроме себя. Менеджеры назначают задания себе или кому-либо из рядовых сотрудников. Менеджер в ходе выполнения созданного им задания может поменять исполнителя.

Просматривать задание, автором которого является менеджер, может либо автор, либо исполнитель задания. Просматривать задание, автором которого является рядовой сотрудник, может автор и любой менеджер. Задания сотрудника отображаются на экране его компьютера в виде страницы календаря (один день на страницу). Приоритет каждого задания (низкий, средний, высокий) визуально выделяется на экране. Каждый менеджер может помимо своего календаря просматривать календари рядовых сотрудников. Помечать задание как выполненное и указывать дату завершения может либо автор, либо исполнитель задания. Вносить какие-либо другие изменения в задание может только автор. После завершения задания внесение в него изменений не

допускается. По прошествии 12 месяцев после даты завершения задания сведения о нем удаляются из системы.

Администратор системы управляет доступом сотрудников: выдает логины и пароли пользователям, формирует две группы пользователей: менеджеров и рядовых сотрудников. Он также имеет доступ к специальным функциям, например, может изменить автора задания или внести изменения в завершенное задание.

Система имеет возможности для поиска в базе клиентов и контактных лиц по их атрибутам (названию, городу, имени контактного лица). Система генерирует отчет по исполнению заданий каким-либо сотрудником в течение периода времени, указываемого в параметре отчета. В отчете указывается: общее количество заданий для данного сотрудника в указанный период, сколько заданий завершено вовремя, сколько заданий завершено с нарушением срока исполнения, сколько заданий с истекшим сроком исполнения не завершено, и сколько не завершенных заданий, срок исполнения которых не истек.

Вариант 7. Банкомат

Тема: Проектирование ИС. Программное обеспечение банкомата

Банкомат – это автомат для выдачи наличных денег по кредитным пластиковым карточкам. В его состав входят следующие устройства: дисплей, панель управления с кнопками, приемник кредитных карт, хранилище денег и лоток для их выдачи, хранилище конфискованных кредитных карт, принтер для печати справок, сервисная консоль. Банкомат подключен к линии связи для обмена данных с банковским компьютером, хранящим сведения о счетах клиентов.

Обслуживание клиента начинается с момента помещения пластиковой карточки в банкомат. После распознавания типа пластиковой карточки, банкомат выдает на дисплей приглашение ввести персональный код. Персональный код представляет собой четырехзначное число. Затем банкомат проверяет правильность введенного кода, сверяя с кодом, хранящимся на карте. Если код указан неверно, пользователю предоставляются еще две попытки для ввода правильного кода. В случае повторных неудач карта перемещается в хранилище карт, и сеанс обслуживания заканчивается. После ввода правильного кода банкомат предлагает пользователю выбрать операцию. Клиент может либо снять наличные со счета, либо узнать остаток на его счету.

При снятии наличных со счета банкомат предлагает указать сумму (100, 200, 500, 1000, 5000, 10000 рублей). После выбора клиентом суммы банкомат запрашивает, нужно ли печатать справку по операции. Затем банкомат посыпает запрос на снятие выбранной суммы центральному компьютеру банка. В случае получения разрешения на операцию, банкомат проверяет, имеется ли требуемая сумма в его хранилище денег. Если он может выдать деньги, то на дисплей выводится сообщение "Выньте карту". После удаления карточки из приемника, банкомат выдает указанную сумму в лоток выдачи. Банкомат печатает справку по произведенной операции, если она была затребована клиентом.

Если клиент хочет узнать остаток на счету, то банкомат посыпает запрос центральному компьютеру банка и выводит сумму на дисплей. По требованию клиента печатается и выдается соответствующая справка.

Сервисная консоль, которая используется обслуживающим персоналом, находится в специальном отделении банкомата, закрываемом на замок. С консоли производится управление доступом к хранилищу денег для загрузки банкнот, управление доступом к хранилищу конфискованных карт, запуск самодиагностики банкомата, конфигурация сетевого соединения с банковским компьютером.

Вариант 8. Интернет-магазин

Тема: Проектирование ИС. Программное обеспечение Интернет-магазина

Магазин компьютеров предлагает возможность приобретения своих товаров через Интернет. Клиент может выбрать компьютер на web-странице магазина. Компьютеры подразделяются на серверы, настольные и портативные. Заказчик может выбрать стандартную конфигурацию из списка и детально ознакомиться с ней на отдельной web-странице. Если стандартная конфигурация ему не подходит, он может построить требуемую ему конфигурацию в диалоговом режиме. Компоненты конфигурации (такие, как оперативная память, процессор, жесткий диск и т. п.) представляются как список для выбора из доступных альтернатив. Для каждой новой конфигурации система может подсчитать цену.

Чтобы оформить заказ, клиент должен заполнить электронную форму с адресами для доставки товара и отправки счета-фактуры, а также деталями, касающимися оплаты. Оплата компьютеров осуществляется наличными курьеру, осуществляющему доставку, или банковским переводом на счет Интернет-магазина. После ввода заказа система отправляет клиенту по электронной почте сообщение с подтверждением получения заказа вместе с относящимися к нему деталями (стоимость, номер счета, банковские реквизиты для безналичной оплаты и т. п.). Пока клиент ожидает прибытия компьютера, он может проверить состояние заказа (поставлен в очередь / собран / отправлен). Работник магазина проверяет, поступила ли оплата (в случае безналичного расчета) и делает соответствующую пометку при поступлении денег. Если деньги не поступают в течение 5 банковских дней, заказ аннулируется. После оплаты или в случае оплаты наличными работник печатает счет-фактуру и отправляет ее на склад вместе с требованием заказанной конфигурации. Заказ помечается как поставленный в очередь. Собранный компьютер вместе со счетом-фактурой и накладной передается со склада в отдел доставки, при этом заказ помечается как собранный. Компьютер поставляется клиенту (статус заказа – отправлен). Если заказ оплачивается наличными, курьер по возвращении передает деньги в кассу, заказ помечается как оплаченный.

По окончании работы с заказом, он помечается в системе как выполненный. Заказы хранятся в системе в течение 15 месяцев с момента создания для составления годовых и квартальных отчетов, после чего автоматически удаляются.

Вариант 9. Библиотечная система

Тема: Проектирование ИС. Система автоматизации для библиотеки

Система поддержки управления библиотекой должна обеспечивать операции над данными о читателях (добавление, удаление и изменение). В регистрационном списке читателей хранятся следующие сведения: фамилия, имя и отчество читателя; номер его читательского билета и дата выдачи билета, дата последней перерегистрации.

Наряду с регистрационным списком системой должен поддерживаться каталог библиотеки, где хранится информация о книгах (наименованиях): название, список авторов, библиотечный шифр, год и место издания, название издательства, общее количество экземпляров книги в библиотеке и количество экземпляров, доступных в текущий момент. Система обеспечивает добавление, удаление и изменение данных каталога, а также поиск книг в каталоге на основании введенного шифра или названия книги или фамилии автора. Читатели имеют доступ только к каталогу книг (они могут осуществлять в нем только поиск и просмотр).

В системе поддерживается реестр экземпляров всех книг библиотеки. Каждый экземпляр имеет свой уникальный идентификационный номер, вообще говоря, не совпадающий с библиотечным шифром. В системе осуществляется регистрация взятых и возвращенных читателем книг. Про каждый выданный экземпляр в реестре хранится запись о том, кому и когда была выдана книга, и когда она должна быть возвращена.

При возврате книги в записи делается пометка, о том, что данный экземпляр находится в наличии и указывается, какой читатель пользовался этой книгой последним. Если экземпляр приходит в негодность, запись реестра о нем удаляется. Если от поставщиков приходят новые книги, записи о них добавляются в реестр и каталог.

При любом обращении читателя в библиотеку сначала осуществляется проверка, не является ли он нарушителем правил пользования. Нарушителем считается тот читатель, который не вернул по истечении срока какую-либо книгу. Нарушители библиотекой не обслуживаются, до тех пор не вернут книги и не заплатят штраф.

Перерегистрация читателей проходит раз в два года. Она необходима для поддержания списка читателей в актуальном состоянии. Если какой-либо читатель пропускает перерегистрацию, то по истечении полугода с момента перерегистрации его читательский билет аннулируется, сведения о нем удаляются из системы.

Система должна выдавать библиотекарям следующую справочную информацию:

- какие книги были выданы за данный промежуток времени;
- какие книги были возвращены за данный промежуток времени;
- какие книги находятся у данного читателя;
- имеется ли в наличии некоторая книга.

Вариант задания предусматривает разработку схемы базы данных, хранящей данные о читателях, каталоге книг, реестре экземпляров.

Вариант 10. Web-форум

Тема: Проектирование ИС. Программное обеспечение Web-форума

Web-форум состоит из нескольких разделов. В каждом разделе содержатся темы, обсуждаемые его пользователями. Темы в разделе упорядочены по убыванию даты последнего ответа в тему. Каждая тема открывается заглавным сообщением и представляет собой древовидную структуру сообщений. Верхний уровень иерархии составляют сообщения, открывающие новые темы, а подуровни составляют сообщения, полученные в ответ на них и т. д.

Сообщение состоит из текста и заголовка (который может не совпадать с заголовком темы). Каждое сообщение-ответ содержит ссылку на сообщение, ответом на которое оно является. Сообщения помечены именами их авторов и двумя датами (датой добавления сообщения и датой его последнего изменения).

Начальной страницей конференции является список разделов, на которой находятся ссылки на первые страницы разделов. Количество тем в разделе может быть большим, поэтому на первой странице раздела отображается список из первых 20 сообщений темы, на второй – следующие 20 и т. д. В списке отображаются только заголовки тем, их авторы и даты последних ответов. Просматривая список, пользователь может перейти на страницу заглавного сообщения темы. Помимо текста заглавного сообщения темы на этой странице отображается список (иерархический) сообщений являющихся ответами на заглавное, ответами на ответы и т. д. С этой страницы пользователь может перейти на страницу сообщения-ответа, на которой также отображается текст сообщения и дерево ответов. На всех страницах сообщений содержатся ссылки на заглавную страницу форума, на страницу текущего форума и на страницу заглавного сообщения темы.

Просматривать страницы форума могут любые пользователи Web. Зарегистрированные пользователи, осуществляют вход в форум, указывая имя и пароль. После входа пользователь может добавить ответ, заполнив форму на странице сообщения, также он может редактировать свои сообщения (в течение двух недель с момента их создания). Еще он имеет возможность начать новую тему, заполнив форму на странице раздела.

Регистрирует новых пользователей администратор форума. При регистрации пользователь заполняет специальную форму, содержимое которой затем пересыпается администратору и запоминается в базе пользователей. Администратор решает, регистрировать пользователя или нет, и отправляет свой ответ. Администратор может создавать, редактировать или удалять разделы.

Администратор управляет правами пользователей, он может назначить кого-либо из них модератором (ведущим) какого-либо раздела. У одного раздела может быть несколько ведущих. Модератор имеет право удалять любые сообщения из раздела, редактировать их, переносить темы в другие разделы. Он также может наказывать пользователей, нарушающих правила поведения в форуме, лишая на некоторое время возможности добавлять и редактировать сообщения.

Вариант задания включает в себя разработку схемы базы данных для хранения разделов, тем и сообщений форума, а также информации о зарегистрированных пользователях.

Вариант 11. Система должна описывать порядок подготовки к экзамену, предполагающий получение отличной оценки.

Вариант 12 Система должна описывать порядок выполнения практической работы по дисциплине «Проектирование ИС».

Вариант 13 Система должна описывать порядок получения водительских прав.

Вариант 14 Система должна описывать порядок организации городского спортивного соревнования.

Вариант 15 Система должна описывать порядок организации общегородского студенческого мероприятия.

Вариант 16 Система составления учебного графика дисциплин, изучаемых на факультете

Вариант 17 Система должна описывать порядок поставок товара в систему розничных киосков.

Вариант 18 Система должна описывать порядок обработки заказов в службе быта.

Вариант 19 Система должна описывать работу одного из участков автосалона .

Вариант 20 Система должна описывать работу приемного покоя в больнице .

Вариант 21 Система должна описывать порядок приема заявки на поставку продукции на хлебокомбинате.

Вариант 22 Система должна описывать процесс поставки сезонных товаров в оптовой фирме .

Вариант 23 Система должна описывать процесс работы торгового отдела .

Вариант 24 Система учета в видеопрокате.

Вариант 25 Система учета проката на лыжной базе.