МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное
образовательное учреждение
высшего профессионального образования
«Пензенский государственный университет (ПГУ)»

В. И. Горбаченко, Г. Ф. Убиенных,
Г. В. Бобрышева

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ с CA ERwin Modeling Suite 7.3

Рекомендовано учебно-методическим объединением
по образованию в области прикладной информатики в качестве
учебного пособия для студентов высших учебных заведений,
обучающихся по направлению «Прикладная информатика»
и другим экономическим специальностям

Пенза
Издательство ПГУ
2012

УДК 004.652.8

Г67

Р е ц е н з е н т ы :
кафедра «Информационные технологии и системы»
Пензенской государственной технологической академии
(заведующий кафедрой – доктор технических наук, профессор
М.Ю. Михеев);

кандидат технических наук, доцент,
заведующий кафедрой прикладной математики и информатики
Пензенского государственного университета
им. В.Г. Белинского
В.В. Дрождин

Горбаченко В. И.

Г67 Проектирование информационных систем с CA ERwin Modeling Suite 7.3 : учебное пособие / В. И. Горбаченко, Г. Ф. Убиенных, Г. В. Бобрышева – Пенза: Изд-во ПГУ, 2012. – 154 с.

ISBN 978-5-94170-459-0

Учебное пособие посвящено проектированию информационных систем на основе использования пакета CA ERwin Modeling Suite 7.3, представляющего собой интегрированный комплекс CASE-средств для моделирования баз данных, бизнес-процессов и компонентов программного обеспечения.

Приводятся краткие сведения о методологиях моделирования предметной области IDEF0, DFD и IDEF3, а также о методологии построения моделей "сущность-связь" IDEF1X. На конкретных примерах рассматривается построение функциональных моделей информационной системы в стандартах IDEF0, DFD и IDEF3 в среде CA ERwin Process Modeler 7.3, а также построение логической и физической моделей данных с помощью CASE-средства CA ERwin Data Modeler 7.3.

Пособие предназначено для подготовки специалистов, бакалавров и магистров по направлениям "Прикладная информатика", "Информатика и вычислительная техника" и «Информационные системы и технологии». Пособие может также использоваться студентами других направлений, осваивающих проектирование информационных систем.

УДК 004.652.8

ISBN 978-5-94170-459-0

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

CA ERwin Modeling Suite 7.3 представляет собой интегрированный комплекс CASE-средств для моделирования баз данных, бизнес-процессов и компонентов программного обеспечения. В его состав входят:

– CA ERwin Process Modeler 7.3;

– CA ERwin Data Modeler 7.3;

– CA ERwin Data Model Validator;

– CA ERwin Model Manager.

CA ERwin Process Modeler (старое название – BPwin) представляет собой мощное средство моделирования, которое поддерживает моделирование процессов (методология IDEF0), моделирование потоков данных (методология DFD) и моделирование технологических процессов (методология IDEF3).

CA ERwin Data Modeler 7.3 (старое название – ERwin) является CASE – средством моделирования модели данных. ERwin автоматически поддерживает согласованность логической и физической схем модели данных и обеспечивает автоматическую генерацию файлов БД в различных форматах: Oracle, DB2, Informix, Sybase, Microsoft SQL Server, Microsoft Access и др.

CA ERwin Data Model Validator (старое название – ERwin Examiner) анализирует структуру данных в схеме, ключи, индексы, поля и связи на предмет нарушения реляционной теории. Это средство генерирует графическую документацию всей структуры БД, включая перекрестные ссылки между столбцами и списки отношений.

CA ERwin Model Manager (старое название – ModelMart) представляет собой масштабируемую многопользовательскую среду моделирования, которая обеспечивает эффективную совместную работу специалистов по моделированию. Это средство обеспечивает централизованное хранение моделей, контроль доступа, управление версиями и службы создания отчетов для CA ERwin Process Modeler, а также для CA ERwin Data Modeler.

Пакет CA ERwin Modeling Suite 7.3 работает под управлением ОС Windows 2000/Windows XP/Windows 2003 Server.

Из-за ограниченности объема учебного пособия в нем рассматривается только работа в средах CA ERwin Process Modeler и CA ERwin Data Modeler. Для простоты далее будем использовать старые названия – BPwin и Erwin.

1. МЕТОДОЛОГИИ МОДЕЛИРОВАНИЯ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Функциональная методология IDEF0

Наиболее популярной методологией IDEF является методология IDEF0 [1 – 4]. Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique – Техника структурного анализа и дизайна) [5]. Исторически IDEF0 как стандарт был разработан в 1981 году в рамках обширной программы автоматизации промышленных предприятий, которая носила обозначение ICAM (Integrated Computer Aided Manufacturing). Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=Icam DEFinition), и последняя его редакция была выпущена в декабре 1993 года Национальным Институтом по Стандартам и Технологиям США (NIST). В России приняты официальные рекомендации по применению методологии IDEEF0 [6].

Целью методологии является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы. Другими словами, в IDEF0 моделируемая система представляется как совокупность взаимосвязанных работ (функций, *активностей*). Методология IDEF0 получила столь широкое распространение в бизнес–моделировании потому, что эта методология легко представляет такие системные характеристики, как управление, обратная связь, исполнители. Кроме того, методология IDEF0 имеет развитые процедуры поддержки коллективной работы.

В основе методологии лежат *четыре основных понятия*: функциональный блок, дуга (стрелка), декомпозиция, глоссарий.

Функциональный блок, или работа (Activity Box) представляет собой некоторую конкретную *функцию* (работу) в рамках рассматриваемой системы. Блок должен иметь название в глагольном наклонении (например, "Проверить документ" или "Проверка документа"). На диаграмме функциональный блок изображается прямоугольником (рис. 1.1). Каждая из четырех сторон функционального блока имеет свое определенное значение (роль) и определяет тип интерфейса, т. е. способ взаимодействия дуги с блоком:

– верхняя сторона имеет значение "Управление" (Control);

– левая сторона имеет значение "Вход" (Input);

– правая сторона имеет значение "Выход (Output);

– нижняя сторона имеет значение "Механизм" (Mechanism).

Дуга (Arrow) отображает элемент системы, который обрабатывается функциональным блоком или оказывает иное влияние на *функцию*, представленную данным функциональным блоком. Интерфейсные дуги часто называют потоками или стрелками.

С помощью дуг отображают различные объекты, которые передаются между блоками, обрабатываются блоками, определяют правила обработки и

механизмы обработки. Такими объектами могут быть элементы реального мира (детали, вагоны, сотрудники и т.д.) или потоки данных и информации (документы, данные, инструкции и т.д.). Стрелки снабжаются надписями – названиями.

В зависимости от того, к какой из сторон функционального блока подходит данная интерфейсная дуга, она носит название "входящей", "исходящей", "управляющей", "механизмом" или вызовом.

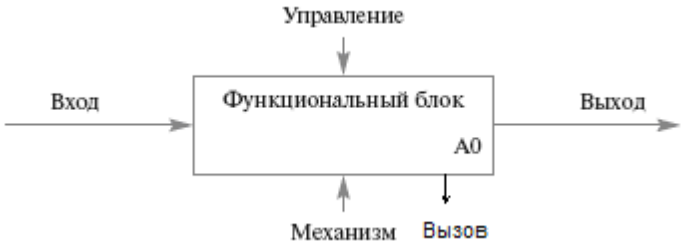

Рис. 1.1. Функциональный блок

В IDEF0 различают *пять типов стрелок.*

Вход (Input) – материальные объекты или информация, которые используются или преобразуются *работой* для получения результата (выхода). Допускается, что блок может не иметь ни одной стрелки входа.

При описании технологических процессов не возникает проблем определения входов. Вход – это нечто, что перерабатывается в блоке для получения результата. При моделировании информационной системы, когда стрелками являются не физические объекты, а данные, определение входа может вызвать трудности. Например, чтобы показать переработку данных блоком, целесообразно на входе указать "Документ", а на выходе – "Заполненный документ". Например, не может быть входом блока "Прием экзамена" стрелка "Студент", а выходом – стрелка "Экзаменационная ведомость", т. к. студент не перерабатывается в ведомость. В данном примере можно использовать входную стрелку "Не аттестованный студент" и выходную стрелку "Аттестованный студент". Очень часто сложно определить, являются ли данные входом или управлением. В этом случае подсказкой может служить информация о том, перерабатываются/изменяются ли данные в блоке или нет. Если изменяются, то, скорее всего, это вход, если нет – управление. Например, задание на курсовой проект является управлением, а не входом.

Управление (Control) – правила, стратегии, процедуры, стандарты, ограничения на бюджет и время, которыми руководствуется работа. Каждая *работа* должна иметь хотя бы одну *стрелку* управления. Управление влияет на *работу*, но не преобразуется *работой*. В случае возникновения неопределенности в статусе *стрелки* (управление или вход) рекомендуется рисовать *стрелку* управления.

Выход (Output) – материальный объект или информация, которые производятся работой. *Каждая* работа *должна иметь хотя бы одну стрелку выхода.* Работа *без результата не имеет смысла и не должна моделироваться.*

Механизм (Mechanism) – ресурсы, которые выполняют работу, например персонал предприятия, станки, устройства и т. д. *По усмотрению аналитика* стрелки *механизма могут не изображаться в модели.*

Вызов (Call) – специальная стрелка, указывающая на другую *модель* работы. Стрелка вызова используется при расщеплении модели и указывает, что некоторая работа представлена отдельной моделью. Расщепление моделей необходимо для коллективной работы над моделью. Руководитель проекта может создать декомпозицию верхнего уровня и провести расщепление модели на отдельные модели. Аналитики работают над отдельными моделями, а затем сливают отдельные модели в единую модель. Отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели.

Таким образом, любой функциональный блок по требованиям стандарта должен иметь, по крайней мере, одну управляющую дугу и одну исходящую. То есть каждый процесс должен происходить по каким-то правилам (отображаемым управляющей дугой) и должен выдавать некоторый результат (выходящая дуга), иначе его рассмотрение не имеет никакого смысла.

Декомпозиция (Decomposition) является основным понятием стандарта IDEF0. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его *функции*. Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Глоссарий (Glossary) – это набор соответствующих определений, ключевых слов, повествовательных изложений и т.д., которые характеризуют объект, отображенный данным элементом. Этот набор является описанием сущности данного элемента. Глоссарий дополняет наглядный графический язык, снабжая диаграммы необходимой дополнительной информацией.

Модель в методологии IDEF0 – это совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе. Модель может содержать четыре типа диаграмм:

1. *Контекстная диаграмма*, которая представляет всю систему как один блок и показывает *контекст* системы, т. е. связь системы с внешним миром. Модель может иметь только одну контекстную диаграмму.

2. *Диаграммы декомпозиции*, которые получаются в результате разбиения контекстной диаграммы на отдельные активности. Такой процесс называется *функциональной декомпозицией,* а диаграммы, получившиеся в результате декомпозиции, называются *диаграммами декомпозиции*. После декомпозиции контекстной диаграммы производится декомпозиция каждой получившейся диаграммы и т. д.

Декомпозиция продолжается до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся *сеансы экспертизы* – эксперты предметной области проверяют соответствие реальных бизнес–процессов созданным диаграммам. Найденные несоответствия ис-

правляются, и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес–процессам на каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

3. *Диаграммы дерева узлов* показывают иерархическую зависимость работ. То есть, в виде дерева показывается, какие активности получились в результате декомпозиции каждой активности. Диаграмм деревьев узлов может быть в модели несколько, поскольку дерево может быть построено на различную глубину и начиная с любой диаграммы (не обязательно с контекстной).

4. *Диаграммы только для экспозиции* (FEO – for exposition only) строятся для иллюстрации альтернативной точки зрения, для хранения старых версий. FEO – это просто картинка. Дело в том, что методология не поддерживает альтернативные варианты декомпозиции. Если необходимо хоть как-то сохранить альтернативный вариант декомпозиции, то применяют диаграмму только для экспозиции.

Рассмотрим подробнее различные виды диаграмм.

Модель IDEF0 всегда начинается с представления системы как единого целого – одной активности с дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одной активностью называется *контекстной диаграммой*. Дуги контекстной диаграммы должны описывать все основные связи моделируемой системы с внешним миром.

Методология IDEF0 подразумевает, что модель является не просто совокупностью диаграмм, а содержит всю необходимую информацию о моделируемой области. Информация о модели задается в свойствах модели. В BPwin информация задается в диалоге свойств модели (**Model Properties**) [18].

В *общих свойствах* (**General**) указываются имя модели, название проекта, автор модели, *временные рамки модели* (**Time Frame**) – **AS-IS** и **TO-BE**. Обычно сначала строится *модель существующей организации работы* – **AS-IS** (как есть). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес–процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса. Детализация бизнес–процессов позволяет выявить недостатки организации даже там, где функциональность на первый взгляд кажется очевидной. Найденные в модели **AS-IS** недостатки можно исправить при создании модели **TO-BE** (как будет) – модели новой организации бизнес–процессов. Иногда текущая **AS-IS** и будущая **TO-BE** модели различаются очень сильно, так что переход от начального к конечному состоянию становится неочевидным. В этом случае необходима третья модель, описывающая процесс перехода от начального к конечному состоянию системы, поскольку такой переход – это тоже бизнес–процесс.

Цель моделирования (**Purpose**) определяется из ответов на следующие вопросы:

– Почему этот процесс должен быть смоделирован?

– Что должна показывать модель?

– Что может получить клиент?

Точка зрения (**Viewpoint)** – это перспектива, с которой наблюдалась система при построении модели. Хотя при построении модели учитываются мнения различных людей, все они должны придерживаться единой точки зрения на модель. Точка зрения должна соответствовать цели и границам моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом.

Даются также определение модели (**Definition**) и описание области действия модели (**Scope**).

Указываются источники получения данных о модели (**Source**), например, "Опрос экспертов предметной области и анализ документации".

Статус модели (**Status**) – это рабочая версия (новая модель, не прошедшая экспертиз) – **WORKING**, черновой вариант (модель прошла первичную экспертизу) – **DRAFT**, рекомендовано (прошла экспертизу) – **RECOMMENDED**, публикация (окончательный вариант) – **PUBLICATION**.

Каждая активность может быть подвергнута *декомпозиции* на другой – "*дочерней*" диаграмме (Child Diagram). Каждая диаграмма нижнего уровня показывает "внутреннее" строение активности на родительской диаграмме (Parent Diagram). Каждая из активностей дочерней диаграммы может быть далее детализирована путем аналогичной декомпозиции. В каждом случае декомпозиции функционального блока все интерфейсные дуги, входящие в данный блок или исходящие из него, фиксируются на дочерней диаграмме. Этим достигается структурная целостность IDEF0-модели.

Чтобы сделать диаграммы удобочитаемыми, в стандарте IDEF0 приняты *ограничения сложности*: на диаграмме может быть от трех до шести активностей (в BPwin – от 2 до 8), при этом количество подходящих к одной активности и выходящих из одной активности дуг предполагается не более четырех.

Работы на диаграммах декомпозиции обычно располагаются в так называемом *порядке доминирования* – по диагонали от левого верхнего угла к правому нижнему. Согласно этому принципу расположения в левом верхнем углу располагается самая важная работа или работа, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже работы. Такое расположение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей работ.

Все активности модели нумеруются. *Номер* состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс **A**. Контекстная активность имеет номер **A0**. Активности, полученные в результате декомпозиции контекстной активности номера **A1, A2, A3** и т. д. Работы декомпозиции нижнего уровня имеют номер родительской

активности и очередной порядковый номер, например активности декомпозиции **A3** будут иметь номера **A31, A32, A33, A34** и т. д. Активности образуют иерархию, где каждая активность может иметь одну родительскую и несколько дочерних работ, образуя дерево. Такое дерево называют деревом узлов, а вышеописанную нумерацию – нумерацией по узлам.

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. Контекстная диаграмма всегда имеет номер **A-0**, декомпозиция контекстной диаграммы – номер **A0**, остальные диаграммы декомпозиции – номера по соответствующему узлу (например, **A1, A2, A21, A213** и т. д.). BPwin автоматически поддерживает нумерацию по узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер.

Чтобы отличать различные версии одной и той же диаграммы, используется специальный номер – **C-number**, который должен присваиваться автором модели вручную. **C-number** – это произвольная строка, но рекомендуется придерживаться стандарта, когда номер состоит из буквенного префикса и порядкового номера, причем, в качестве префикса используются инициалы автора диаграммы, а порядковый номер отслеживается автором вручную, например **GVI021**.

Если активность не повергалась декомпозиции, то левый верхний угол прямоугольника активности автоматически перечеркивается.

Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются *граничными*. Граничные стрелки мигрируют (переносятся) из родительской диаграммы в дочернюю диаграмму. Границы дочерней диаграммы соответствуют границам декомпозируемой активности. Поэтому входные стрелки располагаются на левой границе диаграммы декомпозиции и т. п. Для большего удобства граничные стрелки могут снабжаться так называемыми **ICOM**-кодами. **ICOM** (аббревиатура от **Input**, **Control**, **Output** и **Mechanism**) – коды, предназначенные для идентификации граничных стрелок. Код **ICOM** содержит префикс, соответствующий типу стрелки (**I, C, O** или **M**), и порядковый номер. Граничные стрелки необходимо соединить с соответствующими входами, выходами и т. п. активностей диаграммы декомпозиции.

Стрелки, соединяющие активности на диаграмме, называются *внутренними*. В IDEF0 различают *пять типов связей работ*.

Связь по входу (**output-input**), когда стрелка выхода вышестоящей работы (далее – просто выход) направляется на вход нижестоящей работы (рис. 1.2). На рис. 1.5 – 1.6 в основном показаны только рассматриваемые связи.

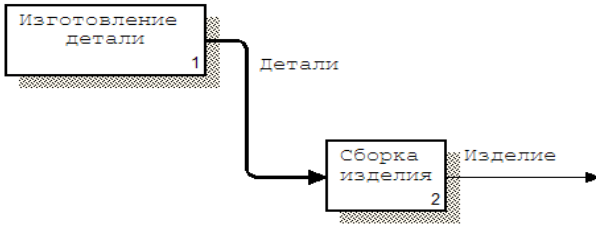

Рис. 1.2. Связь по входу

Связь по управлению (**output-control**), когда выход вышестоящей работы направляется на управление нижестоящей (рис. 1.3). Связь по управлению показывает доминирование вышестоящей работы.

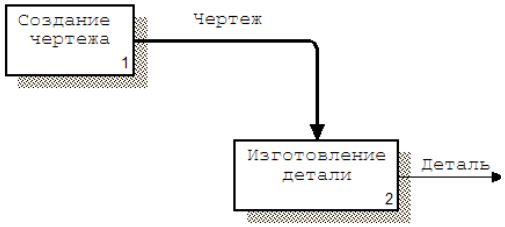

Рис. 1.3. Связь по управлению

Обратная связь по входу (**output-input feedback**), когда выход нижестоящей работы направляется на вход вышестоящей (рис. 1.4). Такая связь, как правило, используется для описания циклов.

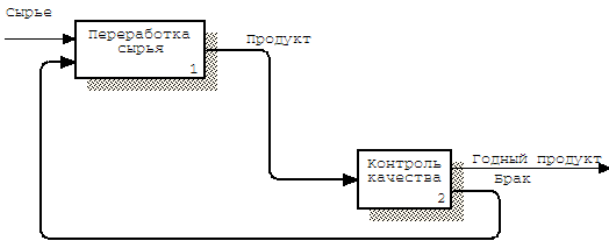

Рис. 1.4. Обратная связь по входу

Обратная связь по управлению (**output-control feedback**), когда выход нижестоящей работы направляется на управление вышестоящей (рис. 1.5). Обратная связь по управлению часто свидетельствует об эффективном управлении бизнес – процессами.

Связь выход-механизм (**output-mechanism**), когда выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже остальных и показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы (рис. 1.6).

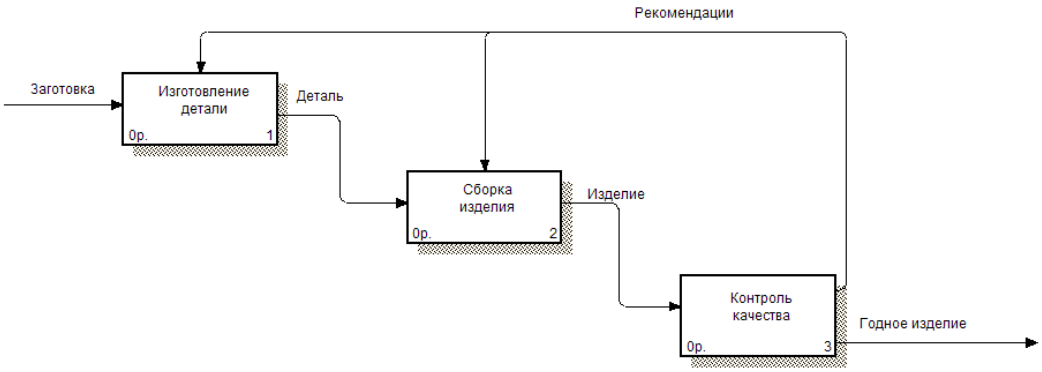

Рис. 1.5. Обратная связь по управлению

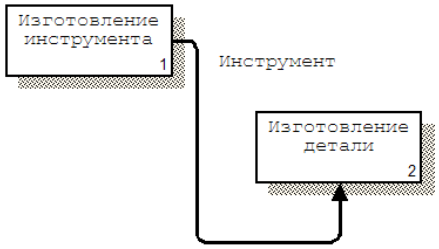

Рис. 1.6. Связь выход-механизм

Простейшим и наиболее распространенным видом стрелок является *явная стрелка*, которая имеет источником одну-единственную активность и назначением тоже одну-единственную активность. Одни и те же данные или объекты, порожденные одной активностью, могут использоваться сразу в нескольких других активностях. С другой стороны, стрелки, порожденные в разных активностях, могут представлять собой одинаковые или однородные данные или объекты, которые в дальнейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций в IDEF0 используются *разветвляющиеся* и *сливающиеся стрелки*. Смысл разветвляющихся и сливающихся стрелок передается именованием каждой ветви стрелок. Существуют определенные правила именования таких стрелок. Рассмотрим их на примере *разветвляющихся стрелок*. Если стрелка именована до разветвления, а после разветвления ни одна из ветвей не именована, то подразумевается, что каждая ветвь моделирует те же данные или объекты, что и ветвь до разветвления. Если стрелка именована до разветвления, а после разветвления какая-либо из ветвей тоже именована, то подразумевается, что эти ветви соответствуют именованию. Если при этом какая-либо ветвь после разветвления осталась неименованной, то подразумевается, что она моделирует те же данные или объекты, что и ветвь до разветвления. Недопустима ситуация, когда стрелка до разветвления не именована, а после разветвления не именована какая-либо из ветвей. Правила именования *сливающихся стрелок* полностью аналогичны – ошибкой будет считаться стрелка, которая после слияния не именована, а до слияния не именована какая-либо из ее ветвей. Для именования отдельной ветви разветвляющихся и сливающихся стрелок сле-

дует выделить на диаграмме только одну ветвь, после чего вызвать редактор имени и присвоить имя стрелке. Это имя будет соответствовать только выделенной ветви.

Иногда отдельные интерфейсные дуги высшего уровня не имеет смысла продолжать рассматривать на диаграммах нижнего уровня, или наоборот – отдельные дуги нижнего уровня отражать на диаграммах более высоких уровней – это будет только перегружать диаграммы и делать их сложными для восприятия. Для решения подобных задач в стандарте IDEF0 предусмотрено понятие *туннелирования*.

Вновь созданные на диаграмме декомпозиции граничные стрелки изображаются в квадратных скобках и автоматически не появляются на диаграмме верхнего уровня (рис. 1.7).

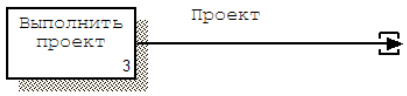

Рис. 1.7. Неразрешенная (unresolved) стрелка

Можно *разрешить миграцию* новой стрелки на диаграмму верхнего уровня или не разрешить такую миграцию. В последнем случае говорят, что стрелка будет *туннелирована*. В BPwin для этого нужно щелкнуть правой кнопкой мыши по квадратным скобкам граничной стрелки и в контекстном меню выбрать команду **Arrow Tunnel**. Появляется диалог **Border Arrow Editor**. Если щелкнуть по кнопке **Resolve Border Arrow**, стрелка мигрирует на диаграмму верхнего уровня, если по кнопке **Change To Tunnel** – стрелка будет туннелирована и не попадет на другую диаграмму. Туннельная стрелка изображается с круглыми скобками на конце.

Туннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые нецелесообразно отображать на диаграммах вышестоящего уровня, то следует туннелировать стрелки на самом нижнем уровне. Такое туннелирование называется *туннель "не-в-родительской-диаграмме"*. Другим примером туннелирования может быть ситуация, когда стрелка механизма мигрирует с верхнего уровня на нижний, причем на нижнем уровне этот механизм используется одинаково во всех работах без исключения. В этом случае стрелка механизма на нижнем уровне может быть удалена, после чего на родительской диаграмме она может быть туннелирована, острие стрелки на родительской диаграмме будет изображено в круглых скобках. В комментарии к стрелке или в словаре можно указать, что механизм будет использоваться во всех работах дочерней диаграммы декомпозиции. Такое туннелирование называется *туннель "не-в-дочерней-диаграмме"*.

Стандарт IDEF0 содержит набор процедур, позволяющих разрабатывать и согласовывать модель большой *группой людей*, принадлежащих к разным областям деятельности моделируемой системы. Обычно процесс разработки является итеративным и состоит из следующих условных этапов:

1. Создание модели группой специалистов, относящихся к различным сферам деятельности предприятия. Эта группа в терминах IDEF0 называется авторами (**`Authors`**). Построение первоначальной модели является динамическим процессом, в течение которого авторы опрашивают компетентных лиц о структуре различных процессов, создавая модели деятельности подразделений. При этом их интересуют ответы на следующие вопросы:

– Что поступает в подразделение "на входе"?

– Какие функции и в какой последовательности выполняются в рамках подразделения?

– Кто является ответственным за выполнение каждой из функций?

– Чем руководствуется исполнитель при выполнении каждой из функций?

– Что является результатом работы подразделения (на выходе)?

2. На основе имеющихся положений, документов и результатов опросов создается черновик (**`Model Draft`**) модели.

3. Распространение черновика для рассмотрения, согласований и комментариев. На этой стадии происходит обсуждение черновика модели с широким кругом компетентных лиц (в терминах IDEF0 – читателей) на предприятии. При этом каждая из диаграмм черновой модели письменно критикуется и комментируется, а затем передается автору. Автор, в свою очередь, также письменно соглашается с критикой или отвергает ее с изложением логики принятия решения и вновь возвращает откорректированный черновик для дальнейшего рассмотрения. Этот цикл продолжается до тех пор, пока авторы и читатели не придут к единому мнению.

4. Официальное утверждение модели. Утверждение согласованной модели происходит руководителем рабочей группы в том случае, если у авторов модели и читателей отсутствуют разногласия по поводу ее адекватности. Окончательная модель представляет собой согласованное представление о предприятии (системе) с заданной точки зрения и для заданной цели.

Наглядность графического языка IDEF0 делает модель вполне читаемой и для лиц, которые не принимали участия в проекте ее создания, а также эффективной для проведения показов и презентаций. В дальнейшем на базе построенной модели могут быть организованы новые проекты, нацеленные на производство изменений в модели.

1.2. Методология DFD

Целью методологии является построение модели рассматриваемой системы в виде *диаграммы потоков данных* (*Data Flow Diagram* – DFD). Диаграммы потоков данных предназначены прежде всего для описания *доку-*

ментооборота и *обработки информации*, хотя допускают и представление других объектов.

При создании диаграммы потоков данных используются *четыре основных понятия:*

– потоки данных,

– процессы (работы) преобразования входных потоков данных в выходные,

– внешние сущности,

– накопители данных (хранилища).

Потоки данных являются абстракциями, использующимися для моделирования передачи информации (или физических компонент) из одной части системы в другую. Потоки на диаграммах изображаются именованными стрелками, ориентация которых указывает направление движения информации.

Процессы (работы) служат для преобразования входных потоков данных в выходные. Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, «получить документы по отгрузке продукции»). Каждый процесс имеет уникальный номер для ссылок на него внутри диаграммы, который может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Хранилище (накопитель) данных моделирует данные, которые будут сохраняться в памяти между процессами. Информация, которую содержит хранилище, может использоваться в любое время после ее получения, при этом данные могут выбираться в любом порядке. Имя хранилища должно определять его содержимое и быть существительным.

Внешняя сущность представляет собой материальный объект вне контекста системы, являющейся источником или приемником данных. Ее имя должно содержать существительное, например, «склад товаров». Предполагается, что объекты, представленные как внешние сущности, *не должны участвовать ни в какой обработке*.

Кроме основных элементов, в состав DFD входят словари данных и миниспецификации.

Словари данных являются каталогами всех элементов данных, присутствующих в DFD, включая потоки данных, хранилища и процессы, а также все их атрибуты. *Миниспецификации обработки* – описывают DFD-процессы нижнего уровня. Фактически миниспецификации представляют собой алгоритмы описания задач, выполняемых процессами: множество всех миниспецификаций является полной спецификацией системы.

Диаграммы потоков данных строятся в виде иерархии. Сначала строится контекстная диаграмма. При проектировании относительно простых систем строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс, соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы. Перед

построением контекстной DFD необходимо проанализировать внешние события (внешние сущности), оказывающие влияние на функционирование системы. Количество потоков на контекстной диаграмме должно быть по возможности небольшим, поскольку каждый из них может быть в дальнейшем разбит на несколько потоков на следующих уровнях диаграммы.

Для сложных систем (признаками сложности могут быть наличие большого количества внешних сущностей (десять и более), распределенная природа системы или ее многофункциональность) строится *иерархия контекстных диаграмм*. При этом контекстная диаграмма верхнего уровня содержит не единственный главный процесс, а набор *подсистем*, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем.

Для проверки контекстной диаграммы можно составить *список событий*. Список событий должен состоять из описаний действий внешних сущностей (событий) и соответствующих реакций системы на события. Каждое событие должно соответствовать одному или более потокам данных: входные потоки интерпретируются как воздействия, а выходные потоки – как реакции системы на входные потоки.

Каждый процесс на DFD, в свою очередь, может быть детализирован при помощи DFD или (если процесс элементарный) спецификации. *Спецификация* процесса должна формулировать его основные функции таким образом, чтобы в дальнейшем специалист, выполняющий реализацию проекта, смог выполнить их или разработать соответствующую программу. Спецификация является конечной вершиной иерархии DFD. Решение о завершении детализации процесса и использовании спецификации принимается аналитиком исходя из следующих критериев:

– наличия у процесса относительно небольшого количества входных и выходных потоков данных (2-3 потока);

– возможности описания преобразования данных процессов в виде последовательного алгоритма;

– выполнения процессом единственной логической функции преобразования входной информации в выходную;

– возможности описания логики процесса при помощи спецификации небольшого объема (не более 20-30 строк).

В качестве языка спецификаций обычно используются структурированный естественный язык или псевдокод.

В методологии DFD используются две *нотации*: Йодана-Де Марко (Yourdan) и Гейна-Сарсона (Gane-Sarson) – табл. 1.1.

Следует отметить, что в BPwin формально используется нотация Гейна-Сарсона, но с рядом отступлений: отсутствуют миниспецификации, отличается изображение функций, контекстная диаграмма не может содержать подсистемы.

Таблица 1.1. Элементы методологии DFD в нотациях Гейна-Сарсона и Йодана-Де Марко

Элемент	Описание	Нотация Йордона-Де Марко	Нотация Гейна-Сарсона
Функция	Работа.	Имя функции Номер	Имя функции Номер
Поток данных	Объект, над которым выполняется работа. Может быть логическим или управляющим. (Управляющие потоки обозначаются пунктирной линией со стрелкой).	Имя объекта	Имя объекта (Понятие управляющего потока отсутствует)
Хранилище данных	Структура для хранения информационных объектов.	Имя объекта	Имя объекта
Внешняя сущность	Внешний по отношению к системе объект, обменивающийся с нею потоками.	Имя внешнего объекта	Имя внешнего объекта

Сравнивая методологии DFD и IDEF0, можно отметить, что в методологии DFD, кроме расширения изобразительных возможностей диаграмм (за счет хранилищ данных), изменяются правила интерфейсов для стрелок: стрелки могут входить и выходить с любой стороны блока.

1.3. Методология IDEF3

IDEF3 является стандартом документирования *технологических процессов*, происходящих на предприятии, и предоставляет инструментарий для наглядного исследования и моделирования их сценариев [7 – 9]. *Сценарием* (*Scenario*) называется описание последовательности изменений свойств объекта в рамках рассматриваемого процесса (например, описание последовательности этапов обработки детали в цехе и изменение её свойств после прохождения каждого этапа). Исполнение каждого сценария сопровождается соответствующим документооборотом, который состоит из двух основных потоков: документов, определяющих структуру и последовательность процесса (технологические карты, стандарты и т.д.), и документов, отображающих ход его выполнения (результаты тестов и экспертиз, отчеты о браке, и т.д.). Для эффективного управления любым процессом, необходимо иметь детальное представление об его сценарии и структуре сопутствующего документооборота.

Средства документирования и моделирования IDEF3 позволяют выполнять следующие задачи:

1. Документировать данные о технологии процесса.
2. Определять и анализировать точки влияния потоков сопутствующего документооборота на сценарий технологических процессов.
3. Определять ситуации, в которых требуется принятие решения, влияющего на жизненный цикл процесса, например, изменение конструктивных, технологических или эксплуатационных свойств конечного продукта.

4. Содействовать принятию оптимальных решений при реорганизации технологических процессов.

5. Разрабатывать имитационные модели технологических процессов, по принципу "КАК БУДЕТ, ЕСЛИ..." Такая возможность существует при использовании, например, системы имитационного моделирования ARENA.

Существуют *два типа диаграмм* в стандарте IDEF3, представляющих описание одного и того же сценария технологического процесса в разных ракурсах. Диаграммы, относящиеся к первому типу, называются *диаграммами потокового описания процесса* (*Process Flow Description Diagrams, PFDD*), а ко второму – *диаграммами сети изменения состояний объектов* (*Object State Transition Network, OSTN*).

Рассмотрим *пример* [9, 10]. Предположим, требуется описать процесс окраски детали в производственном цехе на предприятии. С помощью диаграмм PFDD документируется последовательность и описание стадий обработки детали в рамках исследуемого технологического процесса. Диаграммы OSTN используются для иллюстрации трансформаций детали, которые происходят на каждой стадии обработки. На следующем примере опишем, как графические средства IDEF3 позволяют документировать вышеуказанный производственный процесс окраски детали. В целом, этот процесс состоит непосредственно из самой окраски, производимой на специальном оборудовании и этапа контроля ее качества, который определяет, нужно ли деталь окрасить заново (в случае несоответствия стандартам и выявления брака) или отправить ее в дальнейшую обработку.

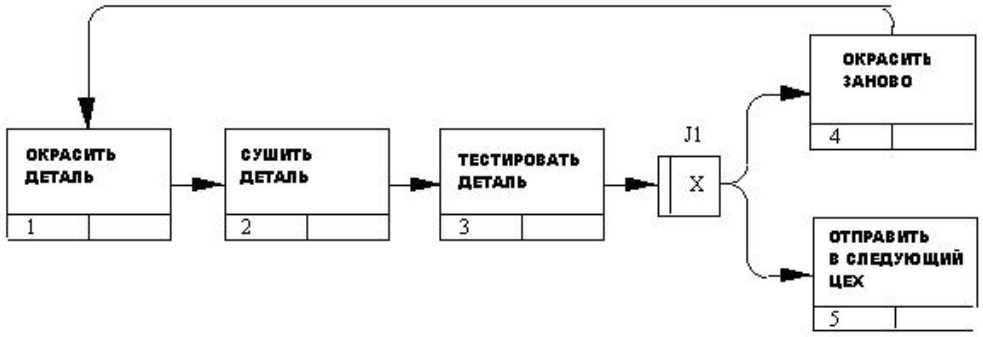

Рис. 1.8. Пример PFDD диаграммы

На рис. 1.8 изображена диаграмма PFDD, являющаяся графическим отображением сценария обработки детали. Прямоугольники на диаграмме PFDD называются *функциональными элементами* или *элементами поведения* (*Unit of Behavior, UOB*)[1] и обозначают событие, стадию процесса или принятие решения. Каждый *UOB* имеет свое имя, отображаемое в глагольном наклонении, и уникальный номер. Этот номер не используется вновь даже в том случае, если в процессе построения модели действие удаляется. В диа-

[1] В BPwin используется термин "*Единица работы*" (*Unit of Work – UOW*).

граммах IDEF3 номер действия, полученного в результате декомпозиции, обычно предваряется номером его родителя (рис. 1.9).

Рис. 1.9. Изображение и нумерация действия в диаграмме IDEF3

Существенные взаимоотношения между действиями изображаются с помощью *связей*. Все связи в IDEF3 являются однонаправленными, и хотя стрелка может начинаться или заканчиваться на любой стороне блока, обозначающего действие, диаграммы IDEF3 обычно организуются слева направо таким образом, что стрелки начинаются на правой и заканчиваются на левой стороне блоков. В табл. 1.2 приведены три возможных типа связей. Стандарт предусматривает и другие типы стрелок [8, 11], но они малоприменимы и не поддерживаются CASE-системами.

Таблица 1.2. Типы связей IDEF3

Изображение	Название	Назначение
	Временное предшествование (Temporal precedence)	Исходное действие должно завершиться, прежде чем конечное действие сможет начаться
	Объектный поток (Object flow)	Выход исходного действия является входом конечного действия (исходное действие должно завершиться, прежде чем конечное действие сможет начаться)
	Нечеткое отношение (Relationship)	Вид взаимодействия между исходным и конечным действиями задается аналитиком отдельно для каждого случая использования такого отношения

Связь типа "временное предшествование" показывает, что исходное действие должно полностью завершиться, прежде чем начнется выполнение конечного действия.

Связь типа "объектный поток" используется в том случае, когда некоторый объект, являющийся результатом выполнения исходного действия, необходим для выполнения конечного действия. Обозначение такой связи отличается от связи временного предшествования двойной стрелкой. Наименования потоковых связей должны четко идентифицировать объект, который передается с их помощью. Временная семантика объектных связей аналогична связям предшествования, это означает, что порождающее объектную связь исходное действие должно завершиться, прежде чем конечное действие может начать выполняться.

Связь типа "нечеткое отношение" используется для выделения отношений между действиями, которые невозможно описать с использованием связей предшествования или объектных связей. Обычно эти связи указыва-

ют, что между объектам существуют некоторые отношения, но на момент описания процесса они не определены.

Объект, обозначенный на рис. 1.8 как J1, называется *перекрестком* (*Junction*). Перекрестки используются для отображения логики взаимодействия стрелок (потоков) 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 *перекрестки для разветвления* (*Fan-out Junction*) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. При внесении перекрестка в диаграмму необходимо указать тип перекрестка. Классификация возможных типов перекрестков приведена в табл. 1.3.

Таблица 1.3. Типы перекрестков

Обозначение	**Наименование**	**Смысл в случае слияния стрелок (Fan-in Junction)**	**Смысл в случае разветвления стрелок (Fan-out Junction)**
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
	Synchronous AND	Все предшествующие процессы должны быть завершены одновременно	Все следующие процессы запускаются одновременно
	Asynchronous OR	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены
	Synchronous OR	Один или несколько предшествующих процессов завершаются одновременно	Один или несколько следующих процессов запускаются одновременно
	XOR (Exclusive OR)	Только один предшествующий процесс завершен	Только один следующий процесс запускается

Все перекрестки в *PFDD* диаграмме нумеруются, каждый номер имеет префикс "J".

Сценарий, отображаемый на диаграмме, можно описать в следующем виде. Деталь поступает в окрасочный цех, подготовленной к окраске. В процессе окраски наносится один слой эмали при высокой температуре. После этого производится сушка детали, после которой начинается этап проверки качества нанесенного слоя. Если тест подтверждает недостаточное качество нанесенного слоя (недостаточную толщину, неоднородность и т.д.), то деталь заново пропускается через цех окраски. Если деталь успешно проходит контроль качества, то она отправляется в следующий цех для дальнейшей обработки.

Описания процессов могут состоять из нескольких сценариев и содержать как диаграммы *PFDD*, так и *OSTN*. Для обозначения отношений и связей между *UOB* различных уровней *PFDD* и *OSTN* диаграмм и разных сценариев в IDEF3 используются специальные *ссылки* (*Referents*).

Ссылки могут использоваться:

− для обращения к ранее определенному функциональному модулю *UOB* без повторения его описания;

− для передачи управления или индикации наличия циклических действий при выполнении процесса;

– организации связи между диаграммами описания процесса *PFDD* и *OSTN* диаграммами.

Соответственно, выделяют следующие типы ссылок:

GOTO – циклический переход (в повторяющейся последовательности *UOW*), возможно, на текущей диаграмме, но не обязательно. Если все *UOW* цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую *UOW*. *GOTO* может ссылаться на перекресток.

UOB – экземпляр другого, ранее определенного *UOB*, выполняется в определенной точке. Например, *UOB* "Контроль качества" может быть использован в процессе "Изготовление редуктора" несколько раз, после каждой единичной операции.

SCENARIO – название сценария. Эта ссылка означает, что должна быть произведена активизация всех декомпозиций указанного сценария.

TS (*Transition Schematic*) – переход на схему. Это ссылка на соответствующую диаграмму, т. е. процесс, на который ссылаются, должен быть инициализирован.

NOTE (примечание) используется для документирования информации, относящейся к каким-либо графическим объектам на диаграмме. Элемент «примечание» может использоваться как в диаграммах описания процесса, так и объектных диаграммах *OSTN*. Этот элемент может быть применен к функциональному элементу *UOW*, перекрестку, связи, объекту или ссылке.

Отметим, что в BPwin используются немного другие ссылки [12].

Методология IDEF3 определяет *два вида ссылок по способу запуска*. *Ссылка "Вызвать и продолжить*" (*Call and Continue Referent*) указывает, что элемент, указанный в ссылке, должен быть активизирован до завершения выполнения действия модулем, к которому относится ссылка. Ссылка "*Вызвать и ждать*" (*Call and Wait Referent*), указывает, что элемент, указанный в ссылке, должен начать и закончить выполнение действия до завершения действия модулем, к которому относится ссылка.

Графические обозначения ссылок приведены на рис. 1.10.

В основном поле символа ссылки указывается её тип (*Referent Type*) "*UOB*", "*SCENARIO*", "*TS*" или "*GOTO*" и через дробь "*Label*" – уникальное наименование блока, сценария, схемы или функции узла, на который указывает ссылка. В поле "Locator" указывается уникальный идентификатор элемента, указанного в ссылке. Пример использования ссылок показан на рис. 1.11 [8].

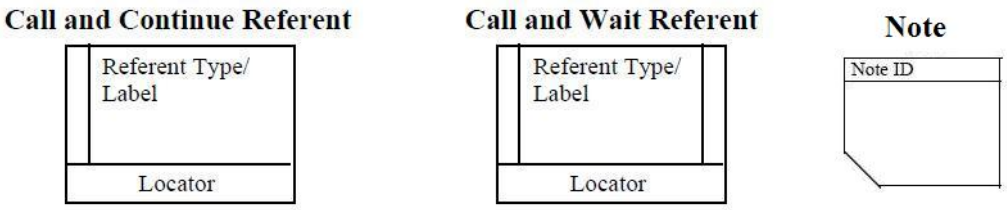

Рис. 1.10. Графическое обозначение ссылок

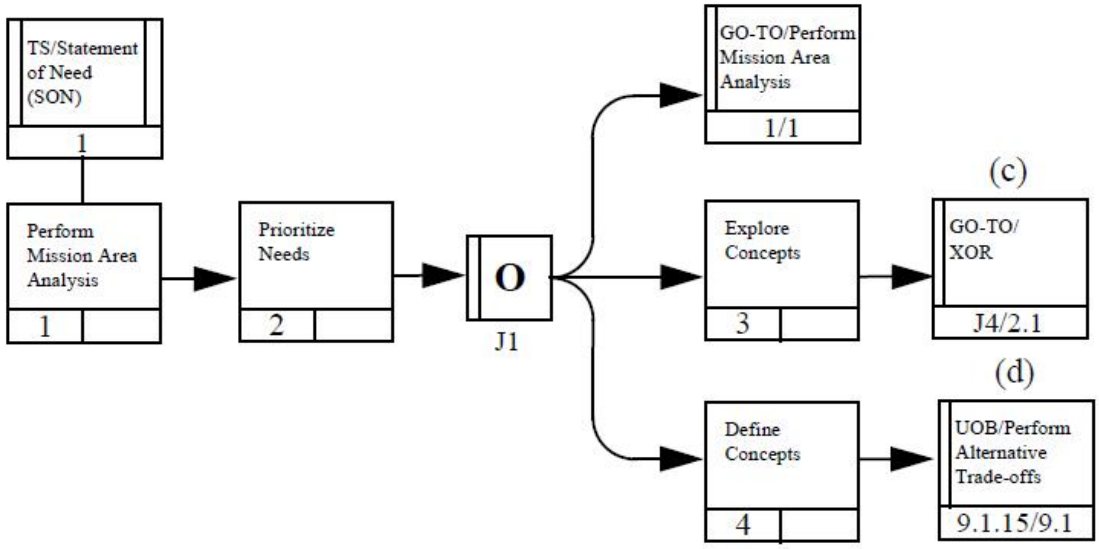

Рис. 1.11. Примеры использования ссылок

Каждый функциональный блок *UOB* может иметь последовательность *декомпозиций*, и, следовательно, может быть детализирован с любой необходимой точностью. Под декомпозицией мы понимаем представление каждого *UOB* с помощью отдельной IDEF3 диаграммы. Например, мы можем декомпозировать *UOB* "Окрасить Деталь", представив его отдельным процессом и построив для него свою *PFDD* диаграмму. При этом эта диаграмма будет называться *дочерней*, по отношению к изображенной на рис. 1.8, а та, соответственно, *родительской*. Номера *UOB* дочерних диаграмм имеют сквозную нумерацию, т.е., если родительский *UOB* имеет номер "1", то блоки *UOB* на его декомпозиции будут соответственно иметь номера "1.1", "1.2" и т.д. Применение принципа декомпозиции в IDEF3 позволяет структурировано описывать процессы с любым требуемым уровнем детализации. На рис. 1.12 приведен пример декомпозиции модулей (*UOB*) и принцип формирования их номеров. Для наглядности все модули представлены на одном рисунке, но в IDEF3 они отображаются в трех диаграммах.

Методология IDEF3 позволяет декомпозировать работу многократно, т. е. работа может иметь множество дочерних работ. Возможность множественной декомпозиции отражается в нумерации работ: номер работы состоит из номера родительской работы, номера декомпозиции и номера работы на

текущей диаграмме. На рис. 1.13 представлен пример двух вариантов декомпозиции родительского модуля [8].

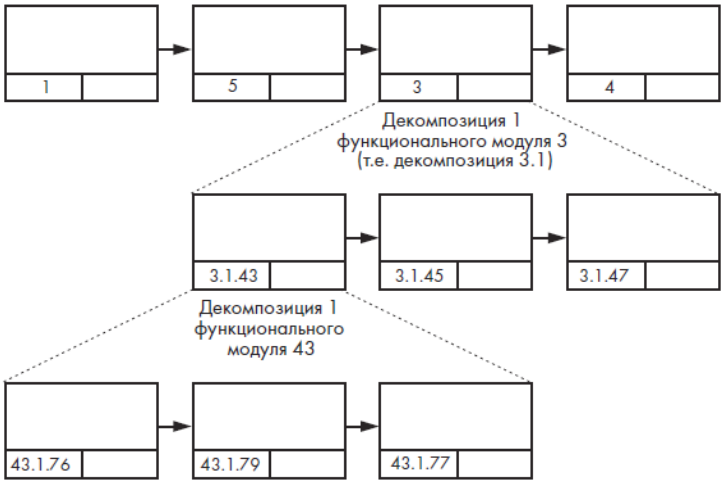

Рис. 1.12. Декомпозиция функциональных блоков

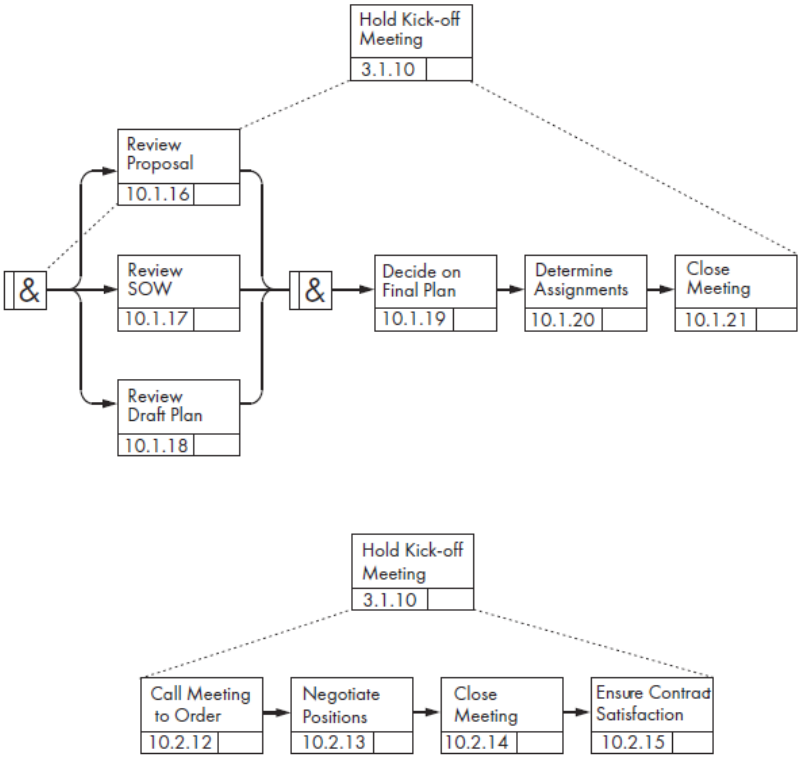

Рис. 1.13. Пример двух вариантов декомпозиции модуля

Если диаграммы *PFDD* описывают технологический процесс "с точки зрения наблюдателя", то другой класс диаграмм *OSTN – диаграммы сети изменения состояний объектов* (не поддерживаются в BPwin) позволяет рассматривать тот же самый процесс "*с точки зрения объекта*". С ее помощью можно графически представить, как одни виды объектов преобразуются в

другие или изменяют свое состояние в ходе выполнения рассматриваемого процесса.

На *OSTN* состояния объектов изображаются окружностями с именем объекта внутри, а изменения состояний − соединительными линиями. Состояние объекта описывается фактами и ограничениями, которые должны выполняться, чтобы объект находился в данном состоянии. Требования для перехода объекта в заданное состояние определяются условиями входа. Условия выхода говорят о ситуации, в которой объект выходит из заданного состояния. Эти ограничения описываются в списке свойств. Связи переходов состояний задают возможные способы изменения состояний объектов.

Для изображения последовательностей переходов объектов из одного вида в другой и изображения перехода одного и того же объекта из одного состояния в другое в диаграммах *OSTN* используются *связи переходов* (*Transition Links*), которые бывают *слабыми* (*Weak Transition Link*) и *сильными* (*Strong Transition Link*). Слабые связи переходов изображаются сплошными одинарными стрелками (рис. 1.14) и показывают, что объекту вида B предшествует объект вида A или что состоянию B некоторого объекта предшествует его состояние A.

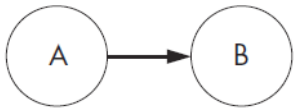

Рис. 1.14. Пример слабой связи переходов

Сильные связи переходов изображаются двойными однонаправленными стрелками (рис. 1.15) и подчеркивают, что объекту вида B должен предшествовать объект вида A или что состояние B объекта достижимо только из состояния A.

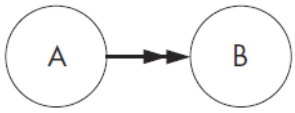

Рис. 1.15. Пример сильной связи переходов

В диаграммах *OSTN* используются те же виды *ссылок*, что и в диаграммах *PFDD*. Исключение составляет лишь ссылка типа *GOTO*, которая используется только в диаграммах потоковых процессов *PFDD*. Ссылки могут относиться как к символу объекта, так и к связи перехода. Соответственно, они интерпретируются как действия, которые необходимо осуществлять для поддержания объекта в данном виде или состоянии, или как действия, которые необходимы для преобразования вида или состояния объекта. Так как процессы поддержания объекта в определенном состоянии и его преобразования могут быть сложными, то допускается использование нескольких ссылок к любому элементу *OSTN* диаграммы.

На диаграммах *OSTN* могут использоваться перекрестки. Перекресток изображается кружком, внутри которого содержится условное обозначение логической функции, реализуемой перекрестком В качестве логических функций могут использоваться И (&), ИЛИ (*O*) и ИСКЛЮЧАЮЩЕЕ ИЛИ (X). Как и на диаграммах *PFDD*, узлы перехода могут означать слияние и разветвление. Но на диаграммах *OSTN* перекрестки не делятся на асинхронные и синхронные. На рис. 1.16 показан пример использования узла разветвления с логической функцией ИЛИ.

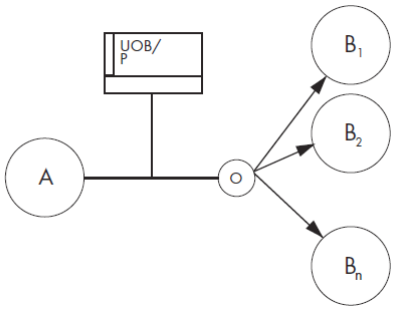

Рис. 1.16. Пример перекрестка с логической функцией ИЛИ

Диаграмма рис. 1.16 означает, что под действиями *UOB* с именем *P* объект из состояния *A* может перейти в одно или сразу несколько состояний из множества возможных: $B_1, B_2, \ldots, B_n$. Если бы в качестве логической функции использовалась функция ИСКЛЮЧАЮЩЕЕ ИЛИ, то это говорило бы, что возможен переход только в одно из возможных состояний $B_1, B_2, \ldots, B_n$. Использование же функции И в перекрестке отображало бы переход объекта из состояния *A* сразу во все состояния $B_1, B_2, \ldots, B_n$.

На рис.1.17 представлено отображение процесса окраски с точки зрения OSTN диаграммы.

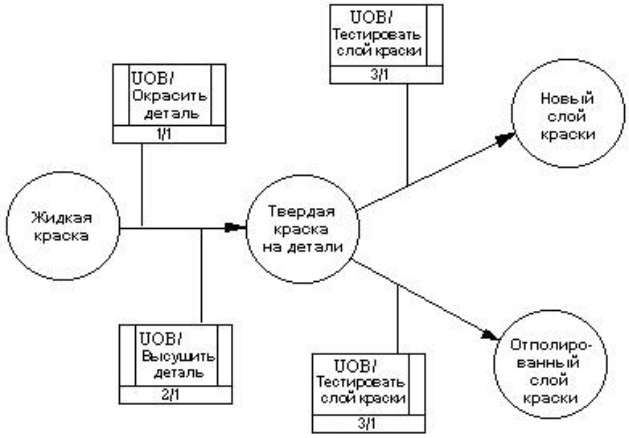

Рис. 1.17. Пример OSTN диаграммы

BPwin имеет возможность преобразования диаграмм IDEF3 в имитационную модель популярной системы моделирования Arena [13 – 14]. Идея преобразования описана в [2, 3], подробное описание дано в фирменной документации [12].

2. СОЗДАНИЕ МОДЕЛИ В СТАНДАРТЕ IDEF0

2.1. Создание контекстной диаграммы

Создание модели рассмотрим на следующем примере [2]. Компания занимается сборкой и продажей настольных компьютеров и ноутбуков. Основными процедурами в компании являются следующие:

– прием продавцами заказов клиентов;

– группировка операторами заказов по типам компьютеров;

– сборка и тестирование операторами компьютеров (под операторами понимаем технический персонал);

– упаковка операторами компьютеров;

– отгрузка кладовщиком заказов клиентам.

Компания использует систему компьютерной бухгалтерии, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Запускаем программу: заходим в **Пуск>Все программы>CA> ERwin>ERwin Process Modeler r7.3>Process Modeler**.

Начинаем создание модели [2, 18]. Описание и скриншоты приведены для ERwin Process Modeler Release 7.3.3.1773. После запуска появляется диалог **I would like to** – Я хочу…(рис. 2.1). Если диалог не появился (он может быть отключен), выберите пункт **File>New** или кнопку создания нового файла.

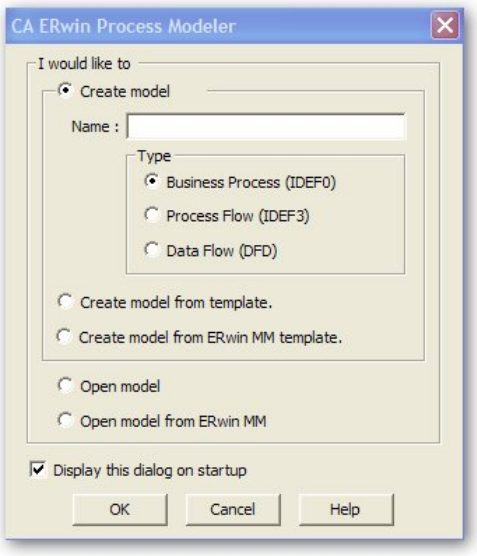

Рис. 2.1. Диалог **I would like to**

В этом окне нам предлагается выбрать, с чего мы хотим начать работу:

– **Create model from template** – создать новую модель по шаблону;

– **Open model** – открыть существующую модель

– **Open model from ERwin MM; Create model from ERwin template** – открыть или создать модель из среды групповой разработки CA ERwin Model Manager.

Выберем создание новой модели по шаблону в методологии IDEF0. Введем имя модели: "Деятельность компании" и нажмем кнопку **OK**.

Далее выводится диалог свойств модели – **Properties for New Models** (рис. 2.2).

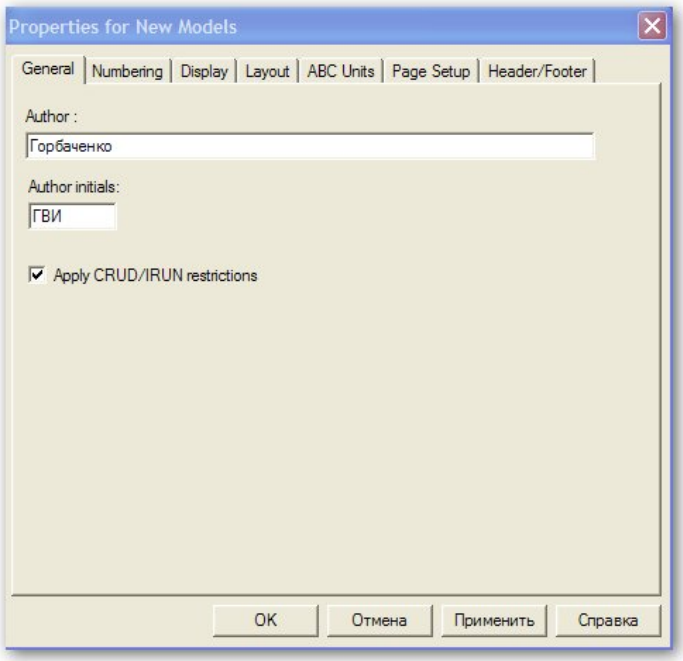

Рис. 2.2. Диалог свойств модели

На вкладке **General** вводим имя и инициалы автора. Включим опцию **Apply CRUD/IRUN Restrictions** – **Применить ограничения CRUD/IRUN**, хотя для нашей задачи эта опция не важна, так как мы не создаем модели данных для нашего процесса. Эти ограничения применяются для связывания модели процессов и модели данных [2]. Данные не могут использоваться работами произвольно. Например, входные данные, представленные стрелкой входа, могут только преобразовываться в выход или потребляться. Рассматриваемые ограничения контролируют использование данных. Аббревиатура CRUD означает Create, Retrieve, Update, Delete и применяется для сущностей, а IRUN означает Insert, Retrieve, Update, Nullify (сделать неопределенным) и применяется для атрибутов сущностей.

На вкладке **Numbering** задаются опции нумерации элементов модели (рис. 2.3).

Activity – группа параметров, отвечающая за нумерацию функциональных блоков:

– **Number prefix** – символ, предшествующий номеру блока (по умолчанию "A");

– **Show prefix** – отображение префикса;

– **Use persistent numbers** – использование постоянной нумерации для диаграмм IDEF0 (если опция включена, то в случае перемещения блока нумерация будет сохранена, если же опция отключена, то нумерация будет автоматически изменена в соответствии со стандартом IDEF0). На диаграммах IDEF3 и DFD используется только постоянная нумерация;

– **Numbering convention** – задание параметров нумерации:

– **1, 2, 3, …** – задание последовательной нумерации;

– **Use diagram numbering format** – нумерация каждой функции содержит в себе уровень декомпозиции;

– **None** – скрытие нумерации функциональных блоков.

Next Numbers – зона, содержащая опции, отвечающие за значения, с которых начнется нумерация:

– **Data Store** – задание первого значения для хранилищ данных на DFD-диаграммах;

– **External** – задание первого значения для внешней ссылки на DFD-диаграммах;

– **UOW** – задание первого значения для UOW на IDEF3-диаграммах.

Оставим на вкладке все без изменения.

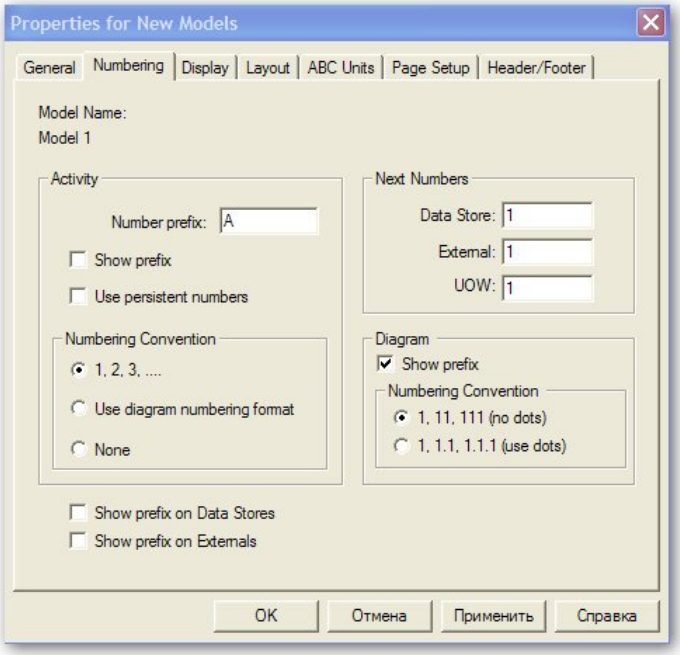

Рис. 2.3. Опции нумерации

На вкладке **Display** (рис. 2.4) определяется, что будет отображаться на диаграммах:

Activity Numbers – отображение нумерации функциональных блоков;

Data Store Numbers – отображение нумерации хранилищ данных в нотации DFD;

External Numbers – отображение нумерации внешних сущностей;

Arrow Names – отображение названий стрелок;

ICOM codes – отображение ICOM-кодов. ICOM (аббревиатура от Input, Control, Output и Mechanism) – коды, предназначенные для идентификации граничных стрелок. Код ICOM содержит префикс, соответствующий типу стрелки (I, C, O или M), и порядковый номер;

Colors – отображение цветов;

ABC Data – отображение данных стоимостного анализа;

Tunnels – отображение квадратных и круглых скобок на стрелках при обозначении незатуннелированных и затуннелированных стрелок соответственно;

Shadows – отображение теней объектов;

Leaf Corners – отображение риски на левом верхнем углу блока, говорящей об отсутствии декомпозиции соответствующей функции.

Squiggles – отображение сноски на названия стрелок;

Block highlighting – отображение подсветки выделенного блока;

Dates in long format in Kit – отображение даты в длинном формате (короткий формат – 10/1/11, длинный формат – October 1, 2011).

Группа переключателей **ABC Units** отвечает за содержимое поля стоимостного анализа, а именно: **Cost** – стоимость, **Frequency** – частота, **Duration** – длительность.

В группе **Off-Page Reference label** указывается маркер межстраничной ссылки. В качестве маркера может выступать пользовательский номер C-number-диаграммы (**C-number**), номер диаграммы по узлу (**Node number**) и имя диаграммы (**Diagram name**).

И на этой вкладке оставим все по умолчанию.

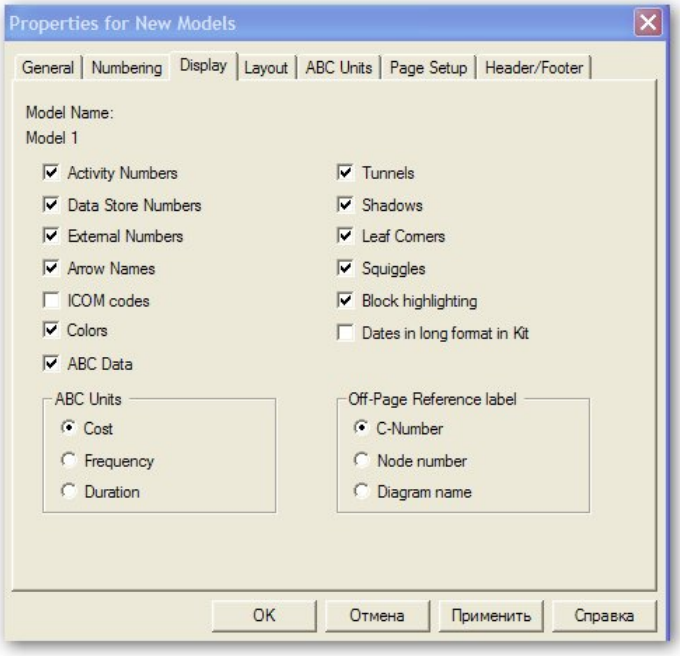

Рис. 2.4. Вкладка **Display**

Параметры вкладки **Layout** – *схема размещения* (рис. 2.5) отвечают за размещение объектов на диаграмме.

Группа **Diagram Objects** отвечает за объекты диаграмм:

– **Allow Box to be moved** – возможность перемещения объектов диаграмм вручную;

– **Allow Box to be resized** – возможность изменения размеров объектов диаграмм вручную.

Подгруппа **Fit Name in Box** позволяет задать опции размещения текста в блоке, а именно:

– **Do not resize or wrap** – текст будет вписан в блок без учета размеров;

– **Wrap text to fit box** – текст будет подогнан по размеру блока;

– **Automatically resize box to fit text** – размер блока будет подогнан по тексту.

Подгруппа **Arrows** отвечает за размещение стрелок:

– **Automatically space arrows** – автоматическое размещение стрелок на диаграмме (например, при задании новой стрелки или изменении размера функционального блока, стрелки на границах блока будут автоматически размещены на одинаковом расстоянии друг от друга);

– **Sort arrows** – минимизация пересечений стрелок на диаграммах IDEF3 и DFD;

– **Break arrows at intersection** – эта опция позволяет создать разрывы стрелок, что облегчает восприятие диаграмм. Можно задать разрыв горизонтальных (**Break horizontal arrow**) и вертикальных (**Break vertical arrow**) стрелок.

Зададим опции, как показано на рис. 2.5.

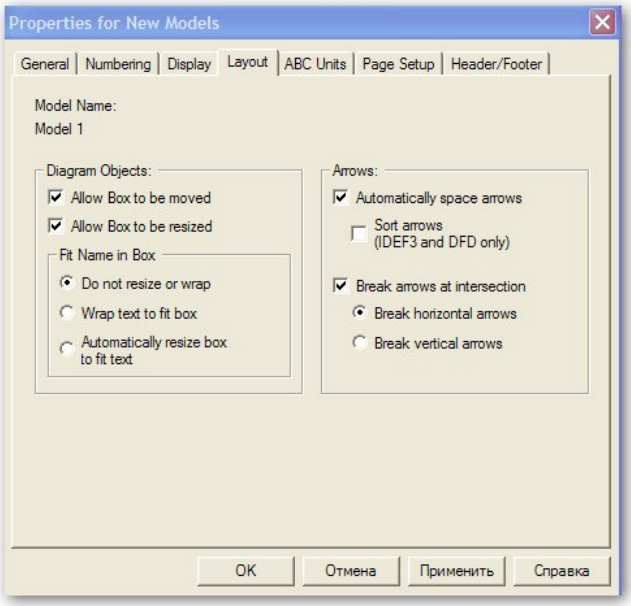

Рис. 2.5. Вкладка **Layout**

Вкладка **ABC Units** (рис. 2.6) отвечает за содержание единиц стоимостного анализа.

Cost – зона задания параметров отображения стоимости:

– **Currency description** – определение валюты. Рублей в выпадающем списке нет, но можно просто вписать нужную валюту;

– **Symbol placement** – определение положения знака валюты относительно числа;

– **Symbol** – определение знака валюты, по умолчанию берется из настроек Windows, можно вписать новое обозначение;

– **Number of decimals in diagrams** – определение числа десятичных знаков при отображении на диаграмме;

– **Number of decimals in reports** – определение числа десятичных знаков при составлении отчетов.

Time – зона задания параметров отображения времени:

– **Time Unit** – определение единиц измерения времени;

– **Decimals in frequency values** – определение количества знаков при задании частоты;

– **Decimals in duration values** – определение количества знаков при задании длительности.

Зададим значения, как показано на рис. 2.6.

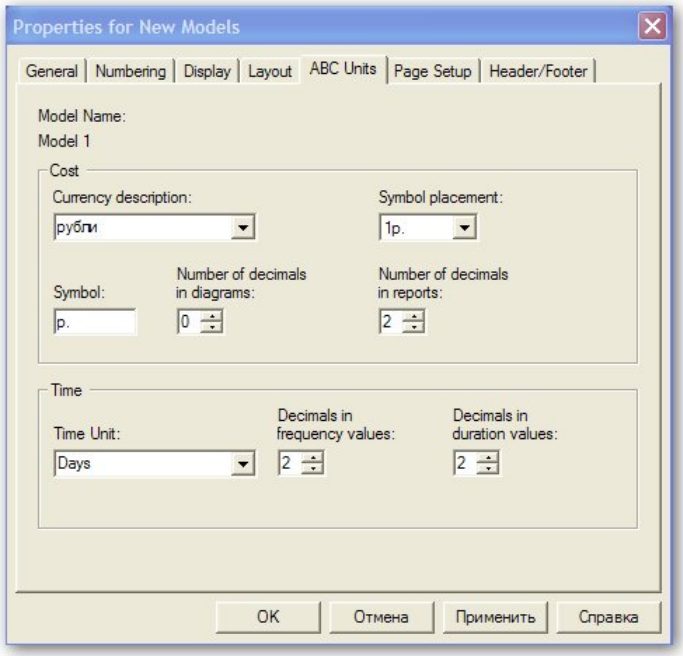

Рис. 2.6. Вкладка **ABC Units**

На вкладке **Page Setup** (рис. 2.7) задаются опции просмотра и печати страницы:

Units – определение единиц измерения (дюймы или миллиметры);

Sheet size – определение размеров листа;

Header – определения способа заполнения заголовка диаграммы (**IDEF Kit** – в соответствии с методологией IDEF, **Custom Header** – заданный пользователем на вкладке **Header/Footer**);

Footer – определение способа заполнения нижнего колонтитула (**IDEF Kit** – в соответствии с методологией IDEF, **Custom Header** – заданный пользователем на вкладке **Header/Footer**).

Примем значения по умолчанию.

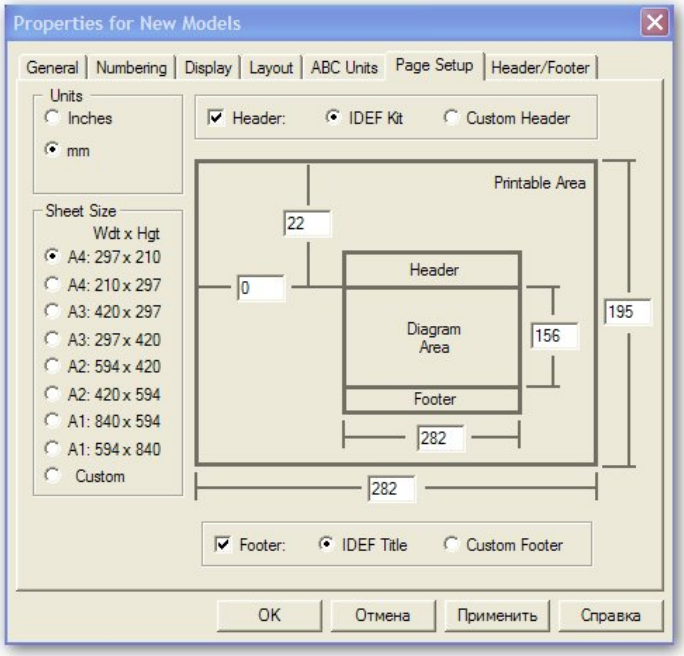

Рис. 2.7. Вкладка **Page Setup**

На вкладке **Header/Footer** (рис. 2.8) определяется пользовательский вид заголовка и нижнего колонтитула.

Доступна следующая информация для заполнения полей:

- **Author** – фамилия автора,
- **Author Initials** – инициалы автора,
- **C-Number** – С-номер,
- **Creation Date** – дата создания,
- **Date and Time** – дата и время,
- **Date (long)** – дата в полном формате записи,
- **Date (short)** – дата в кратком формате записи,
- **Diagram Name** – имя диаграммы,
- **Diagram Number** – номер диаграммы,
- **Diagram Status** – статус диаграммы,
- **Model Name** – имя модели,
- **Model Status** – статус модели,
- **Page Number** – номер страницы,
- **Project Name** – имя проекта,

– **Revision Date** – дата пересмотра.

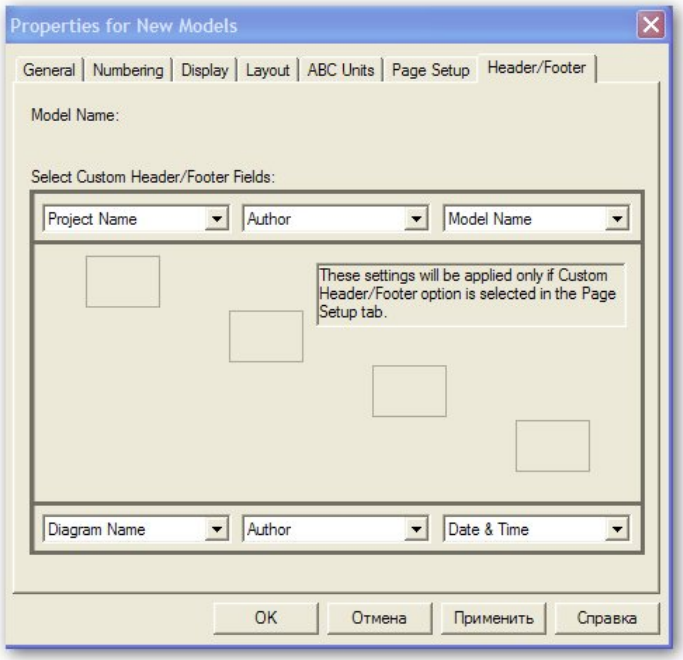

Рис. 2.8. Вкладка **Header/Footer**

Заполним вкладку и подтвердим сделанные изменения – нажмем кнопку **OK**. Сразу создается заготовка для контекстной диаграммы (рис. 2.9).

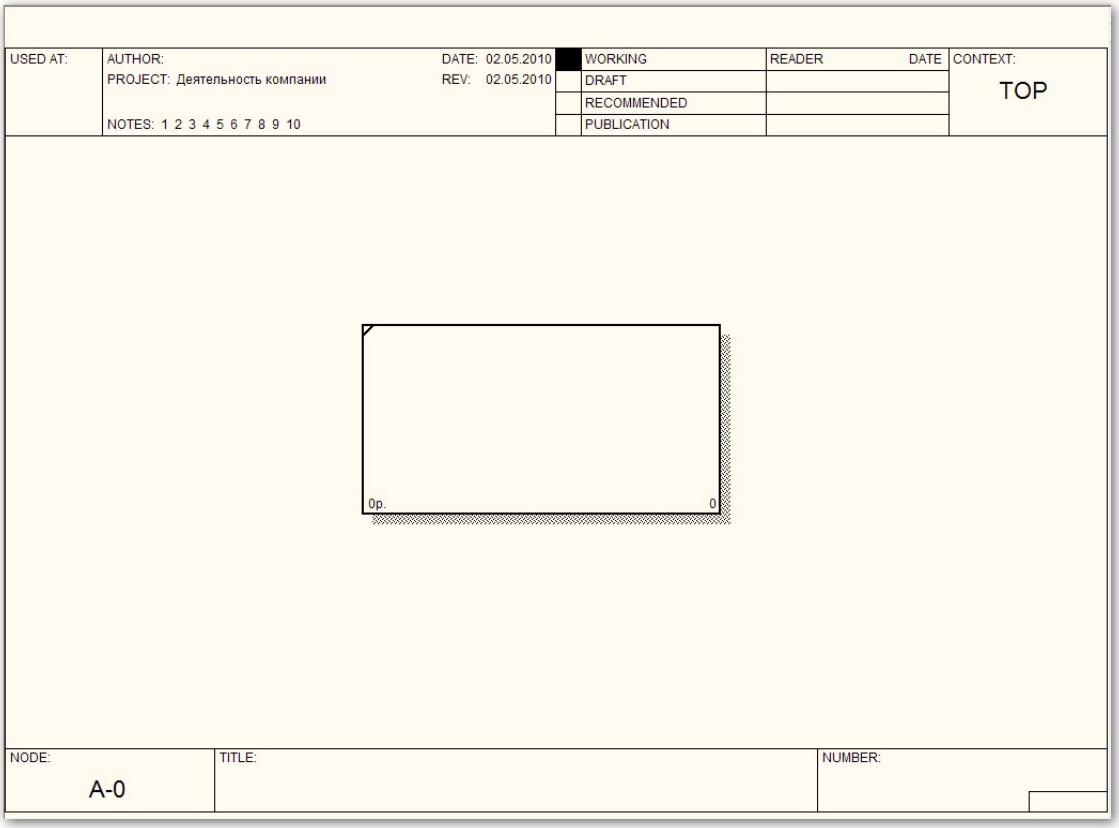

Рис. 2.9. Заготовка контекстной диаграммы

Граничные рамки диаграммы называются каркасом.

Каркас содержит заголовок (верхняя часть рамки) и подвал (нижняя часть) – табл. 2.1 – 2.2.

Таблица 2.1. Поля заголовка каркаса (слева направо)

Поле	Смысл
Used At	Используется для указания на родительскую *работу* в случае, если на текущую диаграмму ссылались посредством *стрелки* вызова
Author, Date, Rev, Project	Имя автора диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма, последнего редактирования диаграммы
Notes 123456789 10	Число замечаний – используется при проведении сеанса экспертизы. Эксперт на бумажной копии диаграммы указывает число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания
Status	Статус – отображает стадию создания диаграммы, включая все этапы публикации: Working – новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы. Draft – диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению. Recommended – диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается. Publication – диаграмма готова к окончательной печати и публикации.
Reader	Имя читателя (эксперта)
Date	Дата прочтения (экспертизы)
Context	Схема расположения диаграммы в иерархии диаграмм. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На *контекстной диаграмме* (A-0) показана надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы: 

Таблица 2.2. Поля подвала каркаса (слева направо)

Поле	Смысл
Node	Номер узла диаграммы (номер родительской *работы*)
Title	Имя диаграммы. По умолчанию – имя контекстной *работы*
Number	C-Number – задаваемый автором уникальный номер версии диаграммы
Page	Номер страницы

Построение модели включает не только построение диаграмм, но и задание свойств модели и ее объектов. Имя заготовке контекстной диаграммы можно задать, выбрав свойства модели (меню **Model>Model Properties**…), свойства диаграммы – двойной клик мыши на свободном поле диаграммы, или пункт меню **Diagram Properties**…, или контекстное меню на свободном поле диаграммы. Свойства активности

можно задать двойным кликом мыши на активности или из контекстного меню на этой активности. Задание свойств других объектов рассмотрим позднее.

Зададим свойства модели. На вкладке **General** (рис. 2.10) зададим информацию о модели. Временные рамки **Time Frame** примем **AS-IS**. Это означает, что рассматриваются существующие процессы.

На вкладке **Purpose (Цель)** внесем цель моделирования **Purpose**: "Моделировать текущие бизнес–процессы компании" и точку зрения, с которой строится модель **Viewpoint**: "Директор".

На вкладке **Definition (Определение)** задаем определение модели **Definition:** "Учебная модель, описывающая деятельность компании" и границы (рамки) модели **Scope**: "Общее управление бизнесом компании".

На вкладке **Source** (**Источник**) признаемся, что данные взяты из книги [2]. На вкладке **Status** зададим статус всей модели: **Working** – рабочий вариант. На вкладке **Shapes** (**Формы**) задается отображение объектов диаграммы.

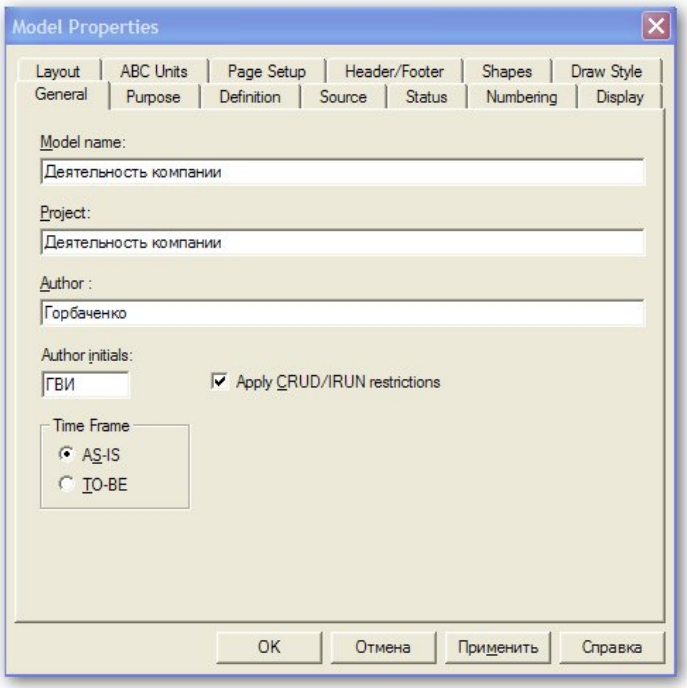

Рис. 2.10. Вкладка **General** диалога **Model Properties**

На вкладке **Draw Style** – **Стиль рисования** (рис. 2.11) задаются параметры графического отображения.

Группа **Style for Diagrams** – задает опции отображения объектов диаграммы:

Methodology Specific – в соответствии с методологией (IDEF0, IDEF3, DFD);

Bitmap – изображения в формате BMP;

Shape – в определенном виде;

Shape and Bitmap – в определенном виде и изображения в формате BMP;

Defer To Diagram – в соответствии с параметрами, заданными на вкладке **Draw Style** диалогового окна **Diagram Properties**;

Show Name – отображать имя;

Show Number – отображать номер;

Show ABC Data – отображать данные по стоимостному анализу.

Группа **Style for Organization charts** – задает опции отображения объектов организационной диаграммы:

Standard – с стандартном виде;

Bitmap – изображения в формате BMP;

Shape – как форму;

Defer To Diagram – в соответствии с параметрами, заданными на вкладке **Draw Style** диалогового окна **Organization Chart Properties**.

Остальные вкладки уже были рассмотрены.

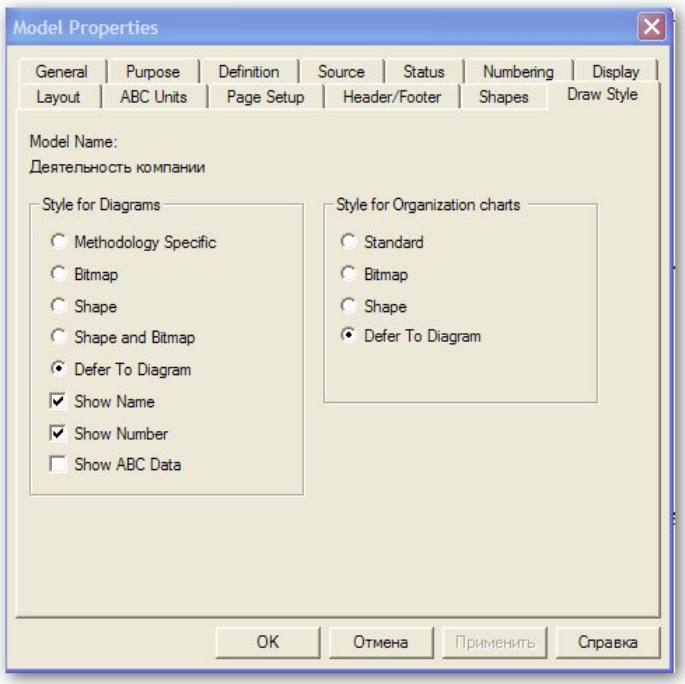

Рис. 2.11. Вкладка **Draw Style**

Теперь в диалоге **Activity Properties** (рис. 2.12) зададим свойства активности контекстной диаграммы. Вкладки **Name, Definition, Status, Source, Box Style** нам уже знакомы. На вкладке **Font** указываются настройки шрифта для отображения надписи на блоке. На вкладке **Color** задаются цветовые настройки.

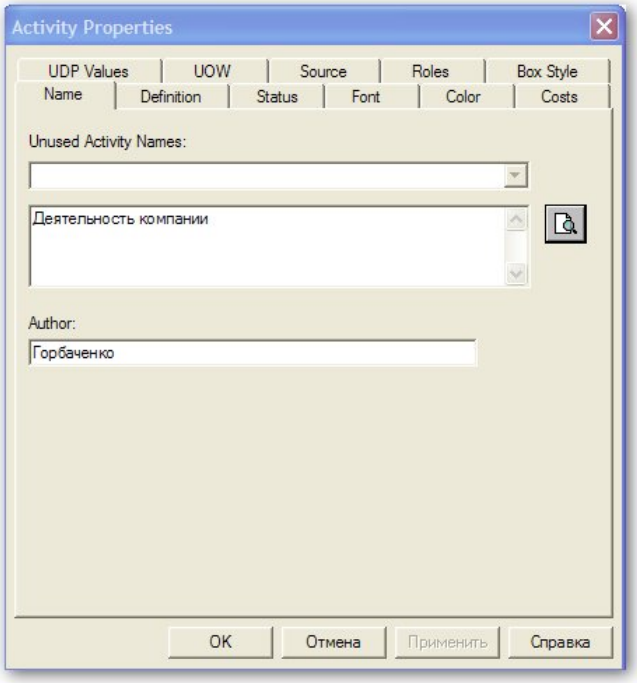

Рис. 2.12. Диалог **Activity Properties**

На вкладке **Cost** (рис. 2.13) задаются статьи затрат (**Cost Center**) для расчета стоимости выполнения данной активности. В верхней части вкладки расположено поле ввода стоимости функции по центрам затрат. В левом столбце представлены доступные центры затрат, их можно определить с помощью кнопки **Cost Center Editor** или в библиотеке центров затрат. В правом столбце можно ввести стоимость выполнения функции в соответствии с определенным центром затрат. Ниже имеется два переключателя: **Override decompositions** – не учитывать данные, введенные ниже по декомпозиции (стоимость определяется вручную); **Compute from decompositions** – вычислить на основе данных, введенных ниже по декомпозиции. Поле **Frequency** определяет кратность выполнения данной функции в цикле. Поле **Duration** определяет длительность выполнения функции.

Оставим все значения по умолчанию.

Вкладка **UDP Values** позволяет задавать значения свойств, определенных пользователем. Если тех свойств, которые поддерживает BPwin, недостаточно, то можно воспользоваться возможностью задания произвольных свойств с помощью словаря свойств, определенных пользователем.

Вкладка **UOW** позволяет задать дополнительные свойства активностей. Как правило, эти данные заполняются для функций IDEF3 диаграммы, но могут быть заданы для любого объекта.

На вкладке **Roles** (**Роли**) можно задать роли, которые будут выполнять эту активность. Например, определенную активность выполняет менеджер. Роли нужно задать в словаре ролей, но учтите, что добавить роли можно

только в определенную группу ролей, так что сначала нужно задать группы ролей в соответствующем словаре.

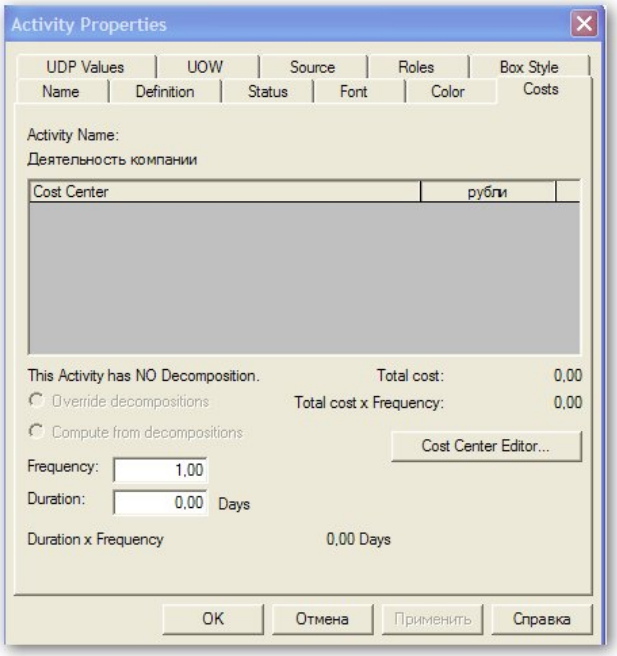

Рис. 2.13. Вкладка **`Cost`**

Подтвердим сделанные изменения – нажмем кнопку **`OK`**. Теперь на контекстной диаграмме изображена активность с заданным именем (рис. 2.14). Кликнув по активности, можно увидеть и редактировать свойства активности.

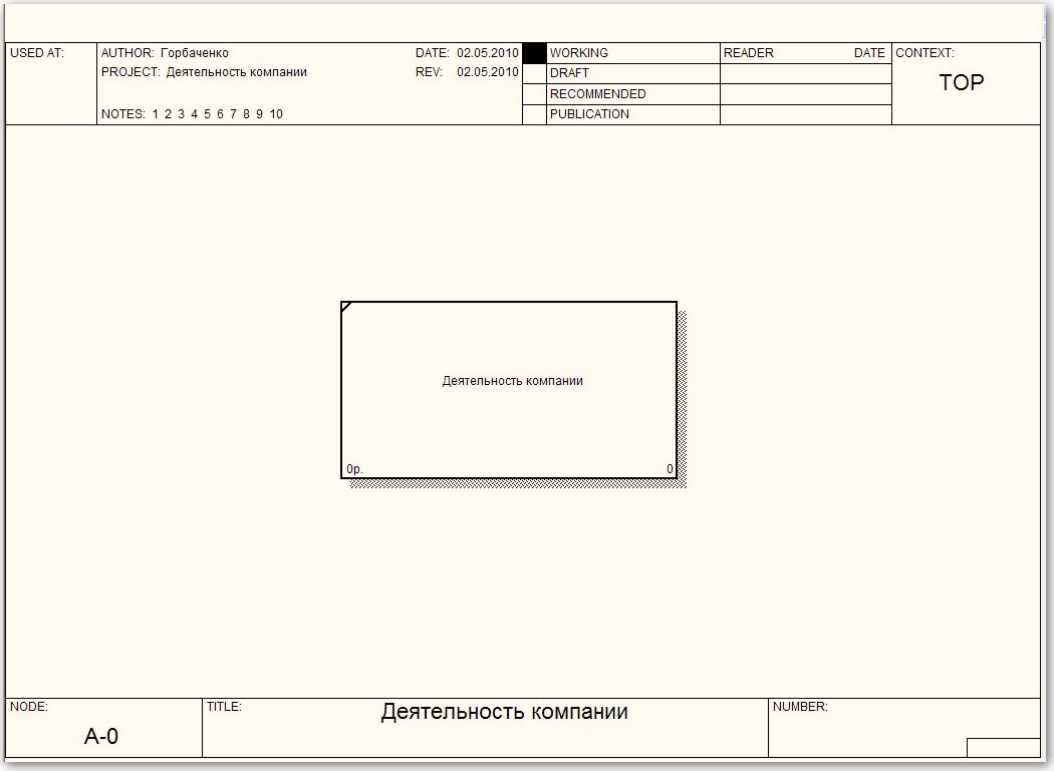

Рис. 2.14. Контекстная диаграмма после задания свойств активности

Дополним контекстную диаграмму стрелками. Чтобы нарисовать стрелку, необходимо выбрать соответствующую кнопку в палитре (рис. 2.15) и первым кликом левой клавиши мыши обозначить начало стрелки, будь-то граница области диаграммы или граница функционального блока, а вторым кликом левой клавиши мыши – конец.

Рис. 2.15. Кнопка рисования стрелок

Нарисуем стрелки в соответствии с рис. 2.16. Чтобы добавить стрелке имя, необходимо дважды кликнуть левой клавишей мыши на стрелке или выбрать пункт **Name** в контекстном меню. В открывшемся окне (рис. 2.17) надо ввести имя стрелки. (Внимание! Никогда не подписывайте стрелки с помощью инструмента **Text**, так как в этом случае информация не попадет в словарь стрелок). После нажатия на кнопку **Применить** появляется окно полного диалога свойств стрелки (рис. 2.18).

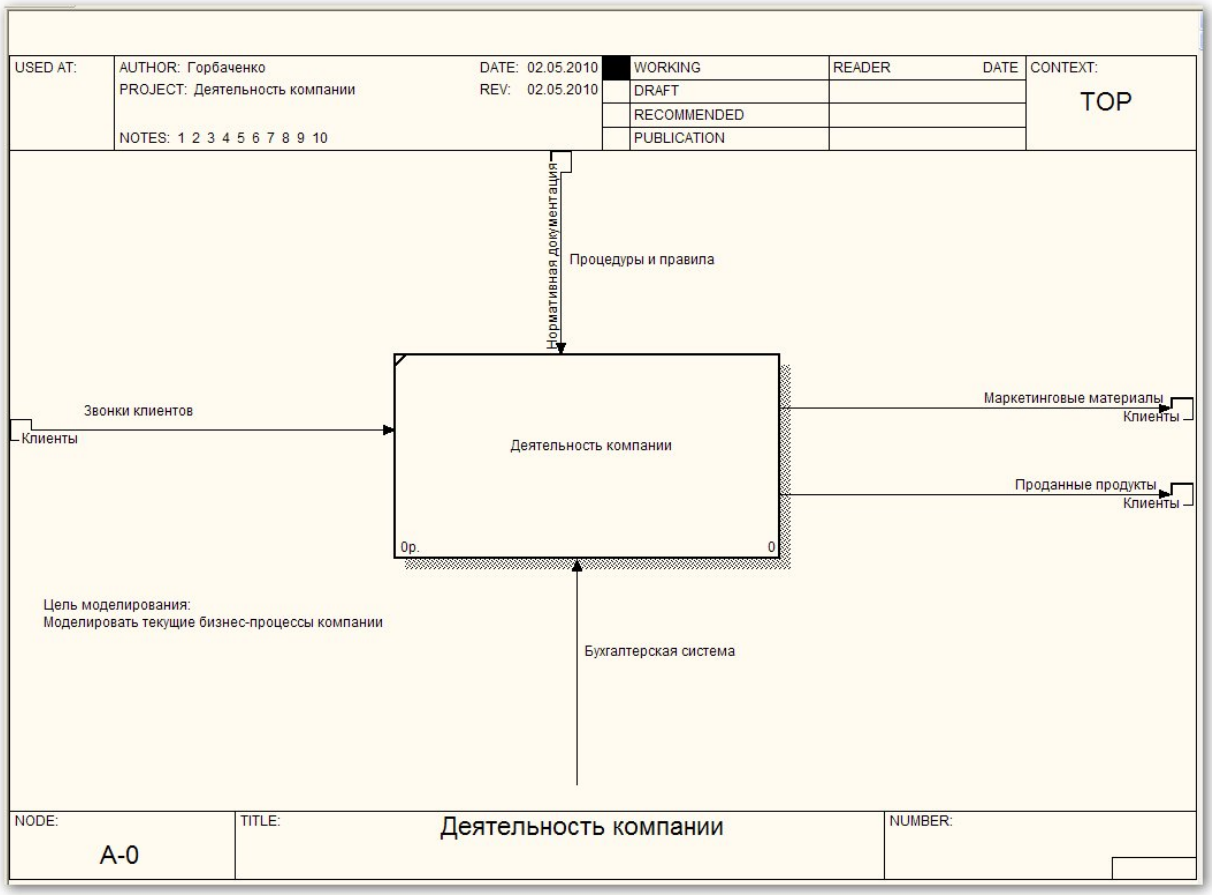

Рис. 2.16. Заполненная контекстная диаграмма

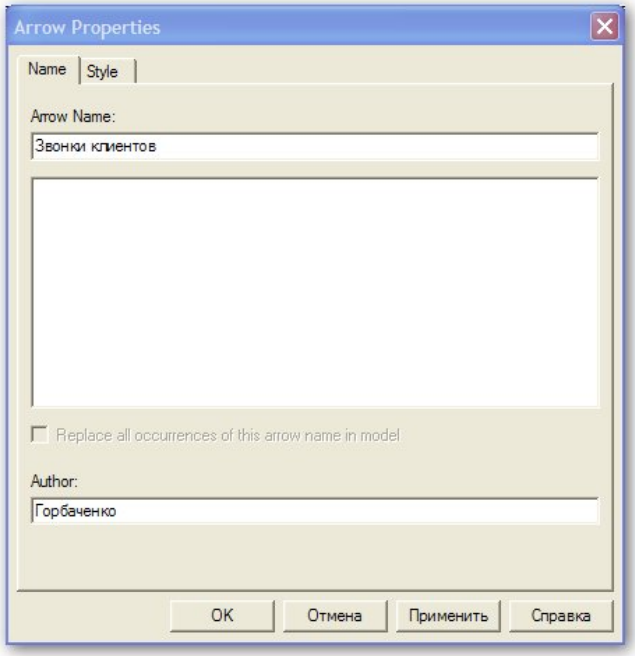

Рис. 2.17. Диалог задания свойств стрелки

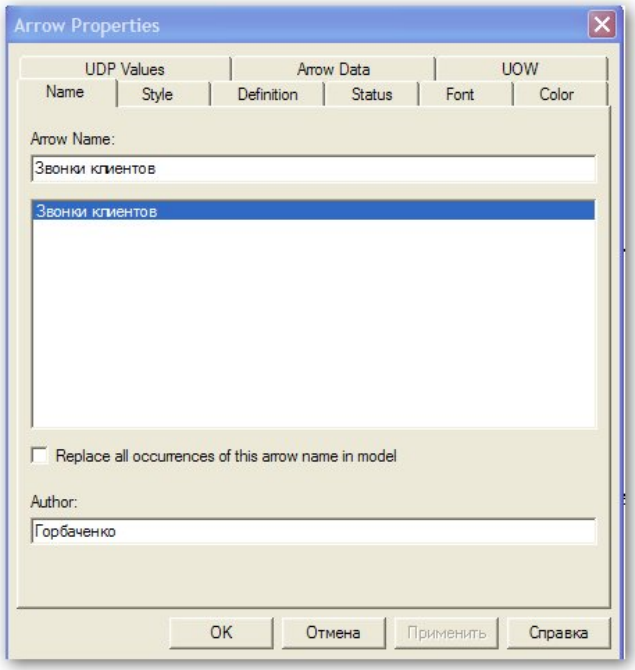

Рис. 2.18. Полный диалог задания свойств стрелки

На рис. 2.16 на концах стрелок у границ диаграммы присутствуют квадратики с подписями – это внешние ссылки (External Reference). Для того чтобы добавить внешнюю ссылку, нужно кликнуть правой клавишей мыши на конце стрелки у границы диаграммы и выбрать пункт **`External Reference`**. Открывается диалоговое окно (рис. 2.19), в котором можно ввести свое название внешней ссылки (оно автоматически будет добавлено в библиотеку), выбрать из уже добавленных или дать ей связь с соответствующей стрелкой.

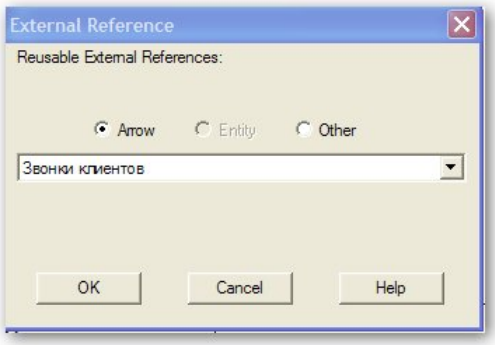

Рис. 2.19. Окно **External Reference**

Свойства активностей и стрелок автоматически помещаются в соответствующие словари. Для просмотра словаря активностей выберем пункт меню **Dictionary>Activity**... Результат представлен на рис. 2.20.

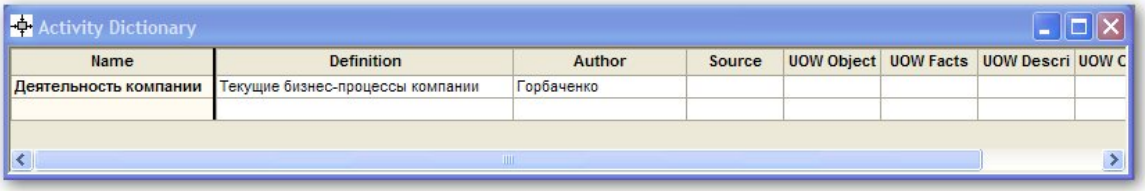

Рис. 2.20. Словарь активностей

Словарь стрелок можно вывести, выбрав пункт меню **Dictionary>Arrow**... (рис. 2.21).

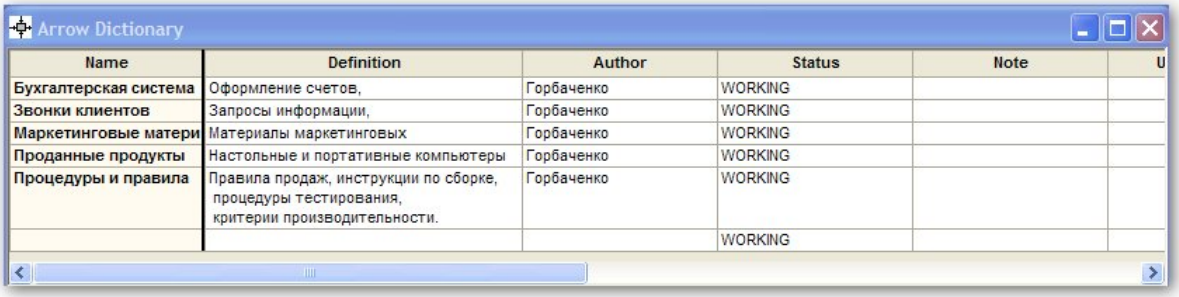

Рис. 2.21. Словарь стрелок

По модели можно создать отчет, содержащий выбранные пользователем свойства модели. В пункте меню **Tools>Reports>Model Report** вызывается диалог **Model report** (рис. 2.22), в котором отмечаются интересующие свойства (на рис. 2.22 отмечены все возможные свойства). Отчет может быть предварительно просмотрен – **Preview**... (рис. 2.23), выведен на печать (**Print**...) или сохранен как текстовый файл (**Report**...).

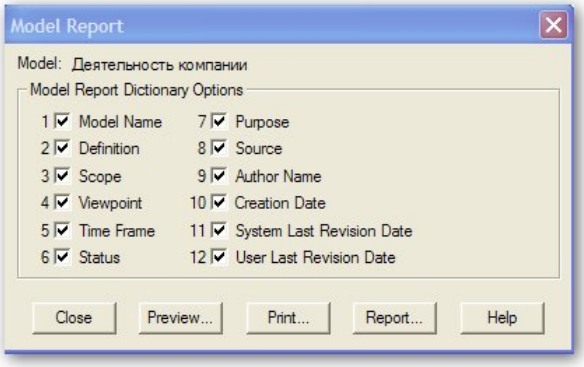

Рис. 2.22. Диалог **Model report**

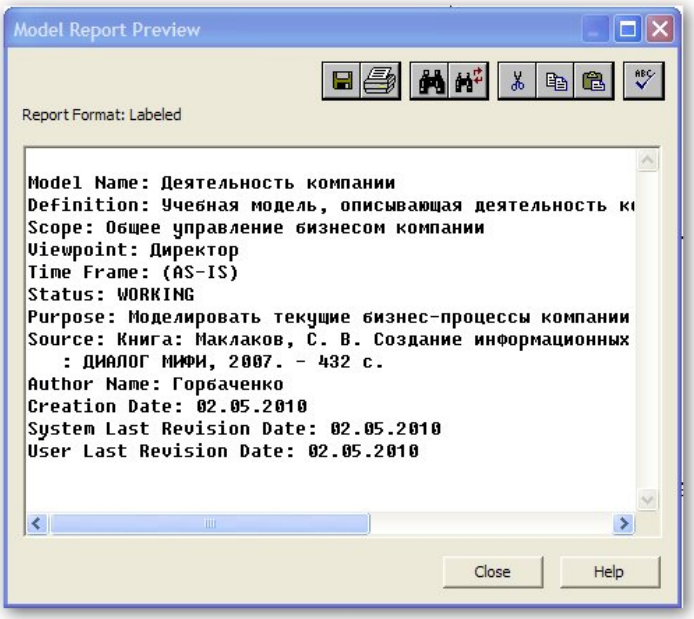

Рис. 2.23. Предварительный просмотр отчета по модели

В левом нижнем углу контекстной диаграммы (рис. 2.16) поместим надпись: «Цель моделирования: Моделировать текущие бизнес – процессы компании». Это необязательный элемент, но надписи помогают сделать модель более наглядной (как и использование цвета). Надписи добавляются с помощью кнопки **Текст** палитры инструментов (рис. 2.24).

Рис. 2.24. Кнопка **Текст**

Выбираем эту кнопку и кликаем левой клавишей мыши на свободном пространстве диаграммы, куда мы хотим добавить текст. Открывается диалоговое окно добавления текста (рис. 2.25).

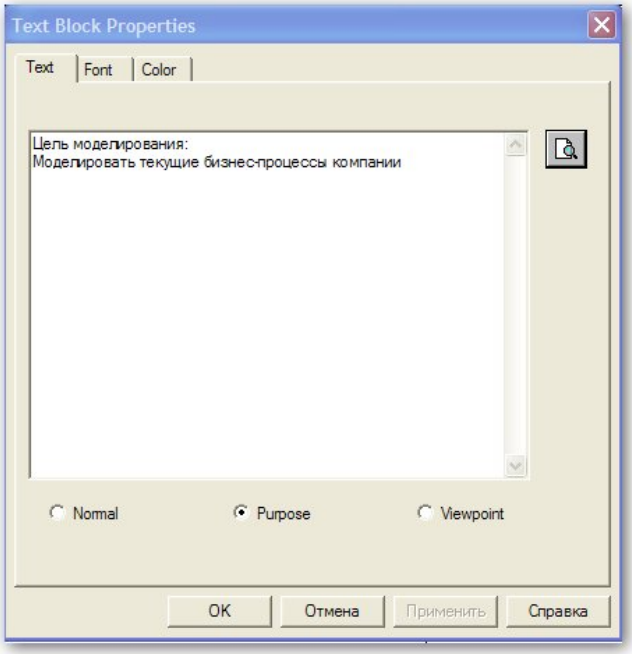

Рис. 2.25. Окно добавления текста

С помощью закладок **Font** и **Color** можно задать настройки шрифта и цвета. Внизу есть три варианта для задания заполнения для текстового блока:

Normal – текст, заданный пользователем;

Purpose – цель, которая была задана в окне свойств модели (цель моделирования была добавлена данным методом);

Viewpoint – точка зрения, которая была задана в окне свойств модели.

После нажатия кнопки **OK** элемент будет добавлен.

2.2. Создание диаграмм декомпозиции

Допустим, что в результате анализа бизнес–процессов выделены три активности, составляющие процесс "Деятельность компании":

– Продажи и маркетинг.

– Сборка и тестирование компьютеров.

– Отгрузка и получение.

Проведем декомпозицию контекстной диаграммы на три перечисленных активности. Декомпозицию можно произвести двумя способами.

Первый способ – выделить декомпозируемую активность, кликнув на ней мышкой (или выделив блок в навигаторе модели на вкладке **Activities**), нажать кнопку **Go to Child Diagram** (рис. 2.26) на панели инструментов.

Рис. 2.26. Кнопка **Go to Child Diagram**

Второй способ – кликнуть правой клавишей мыши в навигаторе модели на вкладке **Activities** на блоке, который необходимо декомпозировать и из выпадающего списка выбрать пункт **Decompose**.

Появляется диалог **Activity Box Count** (рис. 2.27), в котором выбираем тип диаграммы (декомпозицию можно провести в другой методологии) и число активностей на диаграмме декомпозиции.

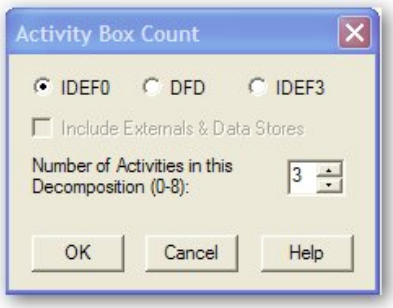

Рис. 2.27. Диалог **Activity Box Count**

Выберем методологию IDEF0, зададим число активностей, равное 3 и нажмем кнопку **OK**. Появляется диаграмма декомпозиции с несвязанными граничными стрелками (рис. 2.28). Обратите внимание на нумерацию диаграмм и активностей. Контекстная диаграмма имеет номер **A0**, а ее активность имеет номер 0. Диаграмма декомпозиции имеет номер декомпозируемой активности, в нашем случае **A0**, а ее активности имеют номера 1, 2, 3. У не подвергшейся декомпозиции активности перечеркнут левый верхний угол. В левом нижнем углу показывается стоимость работы. Так как мы не задавали стоимость работ, то в активностях показано **0р**.

Граничные стрелки мигрировали на диаграмму декомпозиции, но не касаются активностей. Кроме того, на диаграмме появились ICOM-коды: сокращения от Input, Control, Output, Mechanism. Для отображения ICOM-кодов в свойствах модели **Model Properties** на вкладке **Display** необходимо включить отображение ICOM-кодов. Несвязанные граничные стрелки необходимо связать с активностями, но предварительно надо задать имена и свойства активностей. Имена и свойства активностей можно задать из контекстного меню выбранной активности, дважды кликнув на активности.

Для связывания стрелок входа, управления и механизма необходимо перейти в режим редактирования стрелок (кнопка  на панели инструментов), щелкнуть по наконечнику стрелки и щелкнуть по соответствующей стороне прямоугольника активности. Для связывания стрелки выхода необходимо в режиме редактирования стрелок щелкнуть по правой стороне активности, а затем – по стрелке. Щелкая по стрелке и соответствующей стороне блока активности, можно построить разветвление стрелки, например,

стрелки правил. Для выхода из режима редактирования стрелок необходимо нажать кнопку указателя  на панели инструментов.

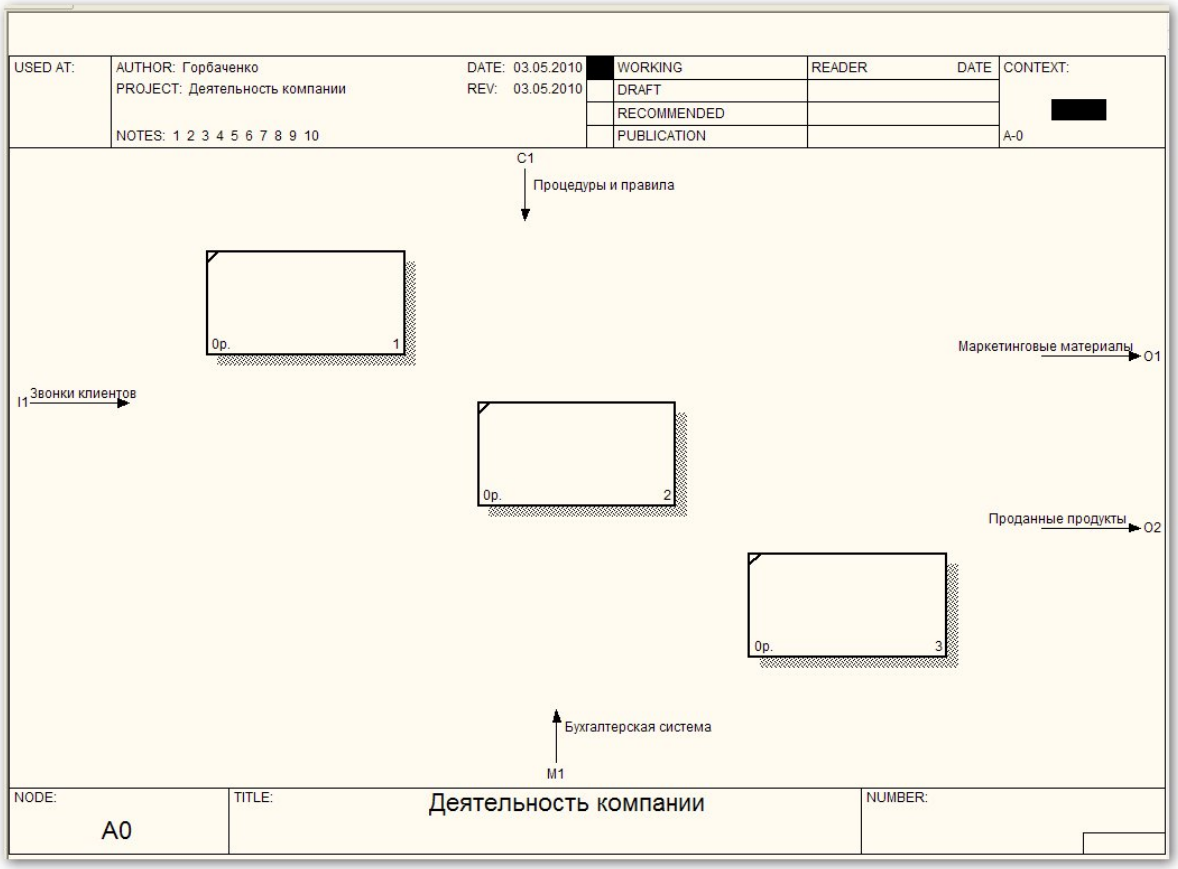

Рис. 2.28. Диаграмма декомпозиции с несвязанными граничными стрелками

В результате получаем диаграмму, показанную на рис. 2.29.

Далее создадим внутренние стрелки, как показано на рис. 2.30. Для рисования внутренней стрелки необходимо в режиме рисования стрелок щелкнуть на стороне блока активности, откуда выходит стрелка, затем по стороне блока активности, куда входит стрелка.

Необходимо задать имена и свойства стрелок. Установив указатель мыши на имя стрелки (при этом выделяется и сама стрелка), можно перемещать имя, изменять размер поля имени. Указатель в виде молнии (**Squiggle**) включается с помощью контекстного меню. Кликните правой клавишей мыши на той стрелке, где нужно добавить этот элемент и выберите из выпадающего меню **Squiggle**. Можно использовать соответствующую кнопку на панели инструментов.

На диаграмме рис. 2.30 введена обратная связь по управлению между активностями "Сборка и тестирование компьютеров" и "Продажи и маркетинг". В контекстном меню этой стрелки задана увеличенная толщина и дополнительная стрелка (**Extra Arrowhead**).

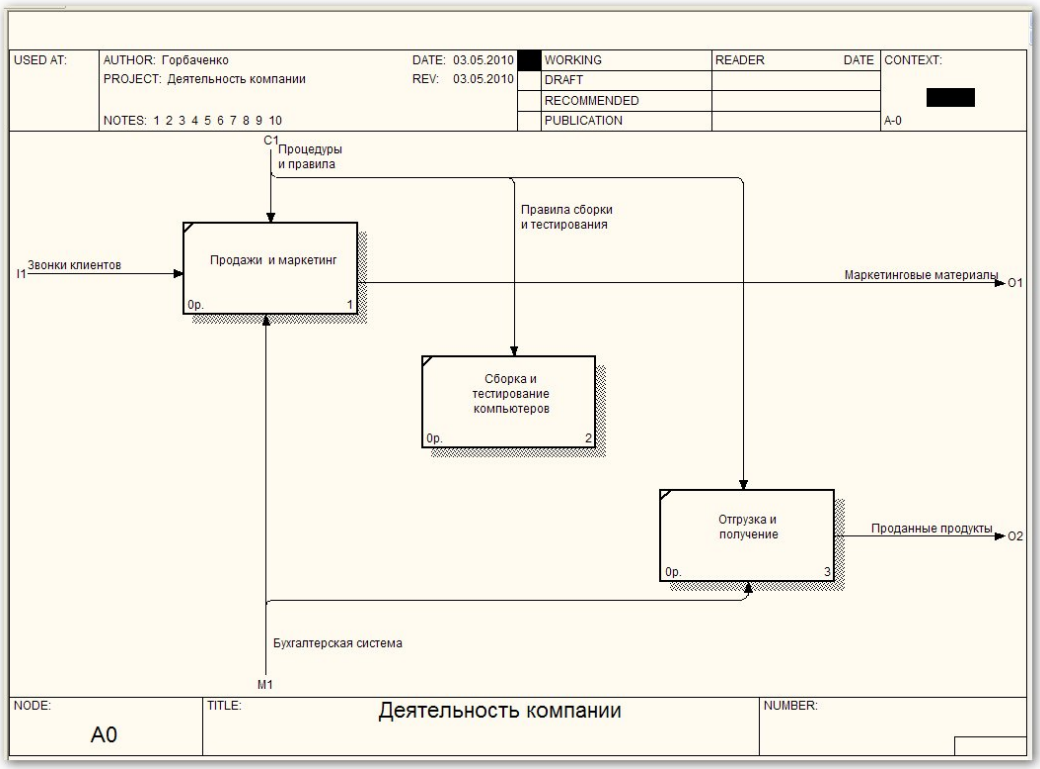

Рис. 2.29. Диаграмма со связанными граничными стрелками

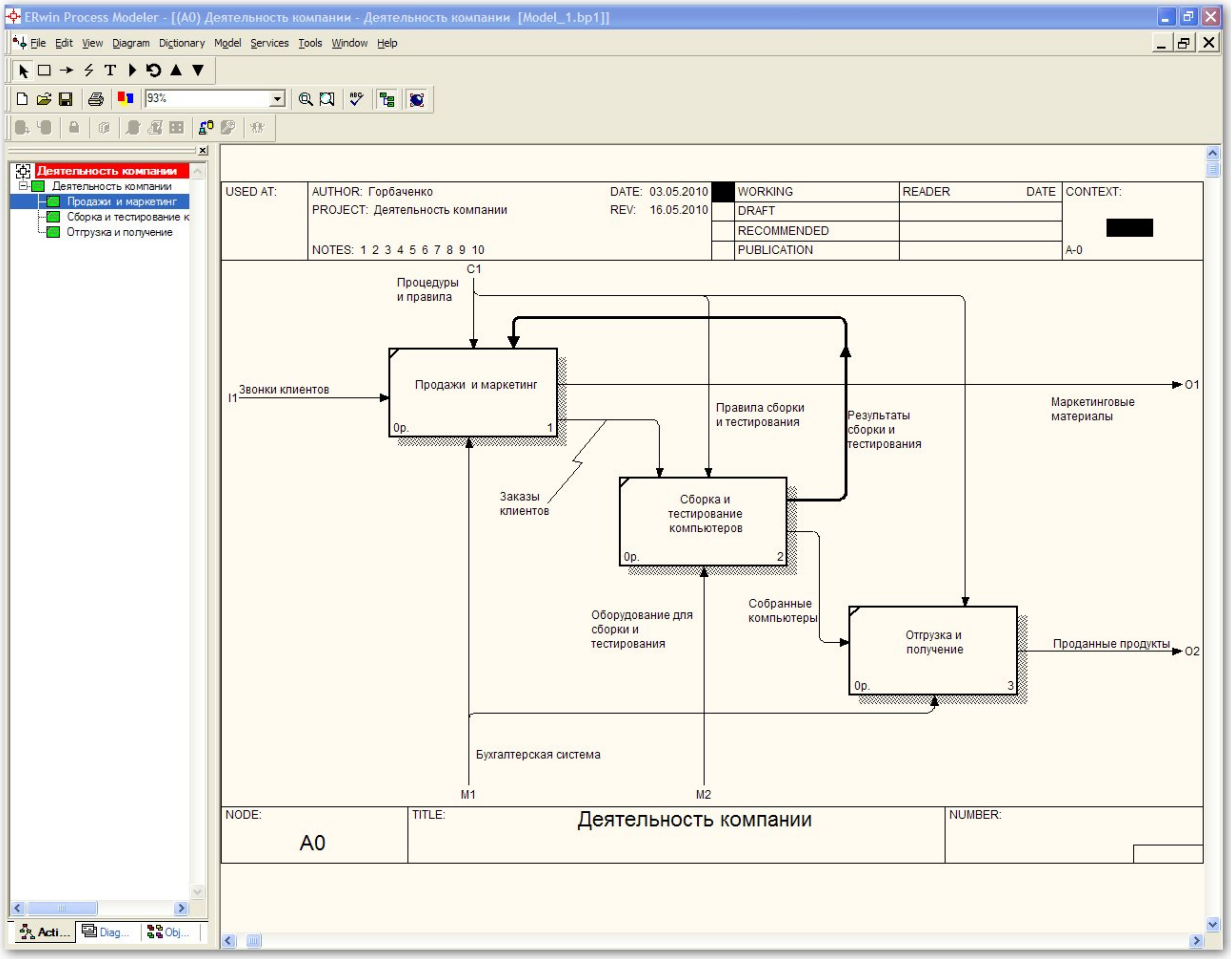

Рис. 2.30. Диаграмма с внутренними стрелками

Добавлена новая граничная стрелка "Оборудование для сборки и тестирования". Эта стрелка автоматически не попадает на диаграмму верхнего уровня, а ее начало заключено в квадратные скобки. Надо щелкнуть правой кнопкой мыши по квадратным скобкам и выбрать в контекстном меню пункт **Arrow Tunnel**. Появляется окно диалога **Border Arrow Editor** (рис. 2.31).

В этом окне выбираем **Resolve it border arrow** (**Разрешить граничную стрелку**). В этом случае граничная стрелка будет видна на всех диаграммах верхних уровней. Если выбрать пункт **Change it resolved rounded tunnel**, то стрелка будет помещена в туннель. Стрелка не будет видна на диаграммах верхних уровней, ее начало будет заключено в круглые скобки. На диаграммах нижних уровней стрелка будет видна в любом случае.

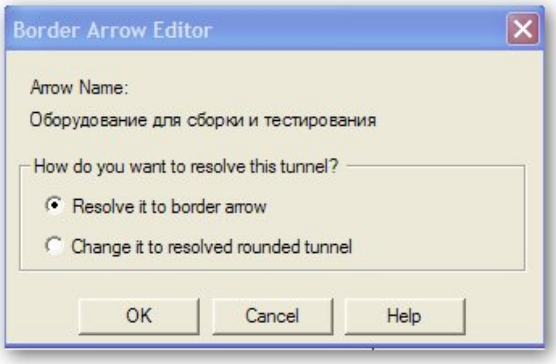

Рис. 2.31. Диалог **Border Arrow Editor**

Туннелирование может быть применено для изображения малозначимых стрелок. Если на какой-либо диаграмме нижнего уровня необходимо изобразить малозначимые данные или объекты, которые нецелесообразно отображать на диаграммах вышестоящего уровня, то следует туннелировать стрелки на самом нижнем уровне. Такое туннелирование называется туннель "не-в-родительской-диаграмме". Другим примером туннелирования может быть ситуация, когда стрелка механизма мигрирует с верхнего уровня на нижний, причем на нижнем уровне этот механизм используется одинаково во всех работах без исключения. В этом случае стрелка механизма на нижнем уровне может быть удалена, после чего на родительской диаграмме она может быть туннелирована, острие стрелки на родительской диаграмме будет изображено в круглых скобках. В комментарии к стрелке или в словаре можно указать, что механизм будет использоваться во всех работах дочерней диаграммы декомпозиции. Такое туннелирование называется туннель "не-в-дочерней-диаграмме".

Для перемещения между диаграммами разных уровней используйте кнопку с изображением треугольника острием вверх на панели инструментов или навигатор модели (рис. 2.32), расположенный слева от диаграммы.

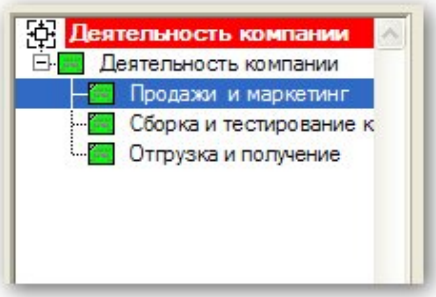

Рис. 2.32. Навигатор

Декомпозируем активность "Сборка и тестирование компьютеров". Эта активность описывается следующим образом [2]. 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 тестируют 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.

На основе этой информации на диаграмме декомпозиции создадим 4 активности (табл. 2.3) и стрелки (табл. 2.4).

Результат декомпозиции показан на рис. 2.33.

На диаграмме рис. 2.33 отключено отображение теней: в меню **Model>Model Properties>Display** отключено отображение теней **(Shadows)**. На диаграмме декомпозиции *A2* введены и затуннелированы (не попадают на диаграмму верхнего уровня) входная стрелка "Компоненты" и стрелка механизма "Персонал производственного отдела". Эта стрелка имеет разные имена после разветвления. Для более наглядного представления диаграммы использован разный цвет стрелок.

Таблица 2.3. Активности диаграммы декомпозиции *A2*

Имя работы (Activity Name)	**Определение работы (Activity Definition)**
Отслеживание расписания и управление сборкой и тестированием	Просмотр заказов, установка расписания выполнения заказов, просмотр результатов тестирования, формирование групп заказов на сборку и отгрузку
Сборка настольных компьютеров	Сборка настольных компьютеров в соответствии с инструкциями и указаниями диспетчера
Сборка ноутбуков	Сборка ноутбуков в соответствии с инструкциями и указаниями диспетчера
Тестирование компьютеров	Тестирование компьютеров и компонентов. Замена неработающих компонентов

Таблица 2.4. Стрелки диаграммы декомпозиции *A2*

Имя стрелки (Arrow Name)	**Источник стрелки (Arrow Source)**	**Тип источника стрелки (Arrow Source Type)**	**Назначение стрелки (Arrow Destination)**	**Тип назначения стрелки (Arrow Destination Type)**
Диспетчер	Персонал производственного отдела		Отслеживание расписания и управление сборкой и тестированием	Mechanism
Заказы клиентов	Граница диаграммы	Control	Отслеживание расписания и управление сборкой и тестированием	Control
Заказы на настольные компьютеры	Отслеживание расписания и управление сборкой и тестированием	Output	Сборка настольных компьютеров	Control
Заказы на ноутбуки	Отслеживание расписания и управление сборкой и тестированием	Output	Сборка ноутбуков	Control
Компоненты	"Tunnel"	Input	Сборка настольных компьютеров	Input
			Сборка ноутбуков	Input
			Тестирование компьютеров	Input
Настольные компьютеры	Сборка настольных Компьютеров	Output	Тестирование компьютеров	Input
Ноутбуки	Сборка ноутбуков	Output	Тестирование компьютеров	Input
Персонал производственного отдела	"Tunnel"	Mechanism	Сборка настольных компьютеров	Mechanism
			Сборка ноутбуков	Mechanism
Правила сборки и тестирования	Граница, диаграммы		Сборка настольных компьютеров	Control
			Сборка ноутбуков	Control
			Тестирование компьютеров	Control
Результаты сборки и тестирования	Сборка настольных Компьютеров	Output	Граница диаграммы	Output
	Сборка ноутбуков	Output		
	Тестирование компьютеров	Output		
Результаты тестирования	Тестирование компьютеров	Output	Отслеживание расписания и управление сборкой и тестированием	Input
Собранные компьютеры	Тестирование компьютеров	Output	Граница диаграммы	Output
Тестировщик	Персонал производственного отдела		Тестирование компьютеров	Mechanism
Указание передать компьютеры на отгрузку	Отслеживание расписания и управление сборкой и тестированием	Output	Тестирование компьютеров	Control

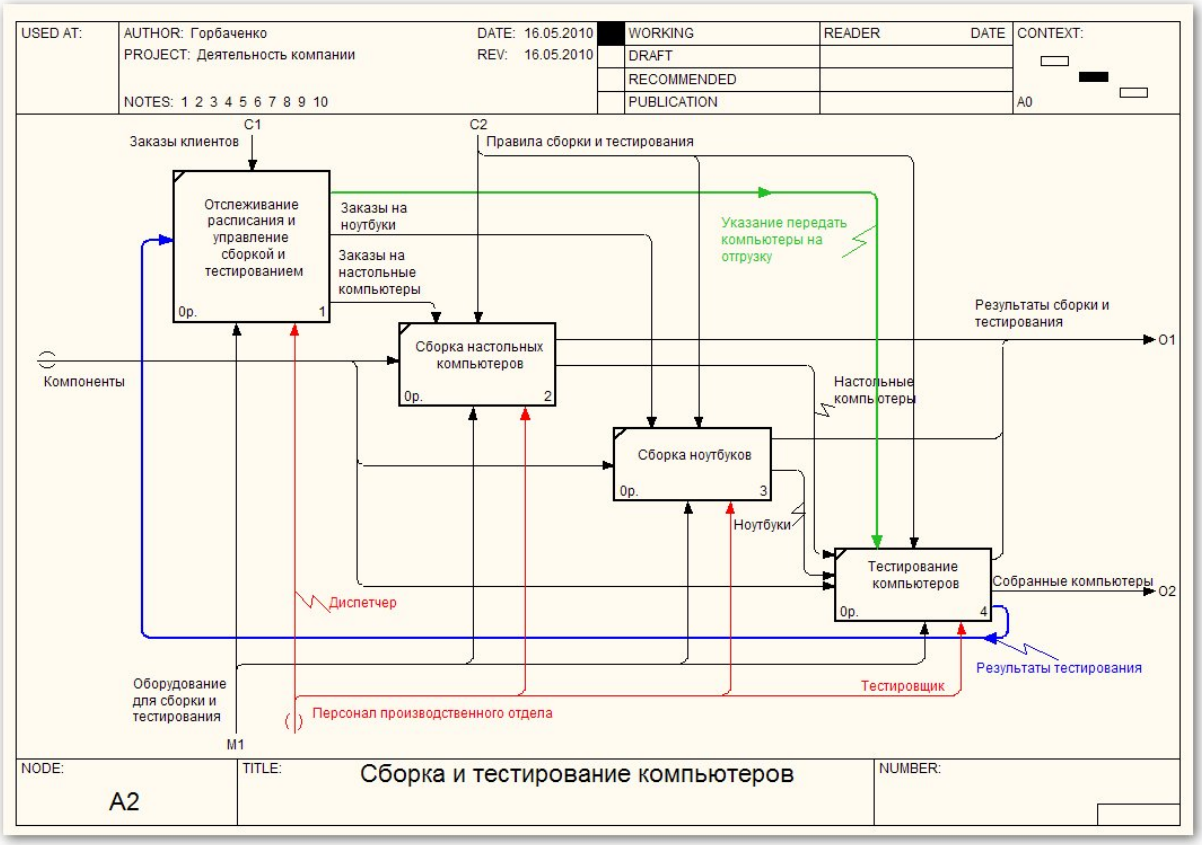

Рис. 2.33. Диаграмма декомпозиции *A2*

Для перемещения по модели целесообразно использовать навигатор, (**`Model Explorer`**), который после декомпозиции имеет вид рис. 2.34.

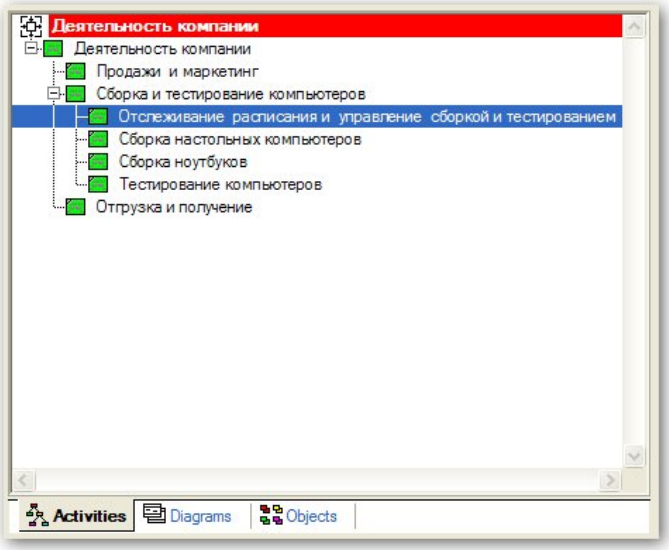

Рис. 2.34. Навигатор после декомпозиции

На рис. 2.34 отображена вкладка **`Activities`** (**`Активности`**), на которой видны иерархически упорядоченные активности модели. На вкладке **`Diagrams`** изображены иерархически упорядоченные диаграммы модели.

Вкладка **Objects** отображает имеющиеся в словаре, но не использованные активности (**Unused Activities**). Активность можно «перетащить» на диаграмму.

2.3. Создание диаграммы дерева узлов

Диаграмма дерева узлов показывает иерархию работ в модели. Для создания диаграммы выбираем в меню пункт **Diagram>Add Node Tree**. В первом окне визарда (мастера) построения дерева (рис. 2.35) необходимо ввести имя диаграммы, узел верхнего уровня (имя активности корня дерева) и число уровней (**Number of Levels**).

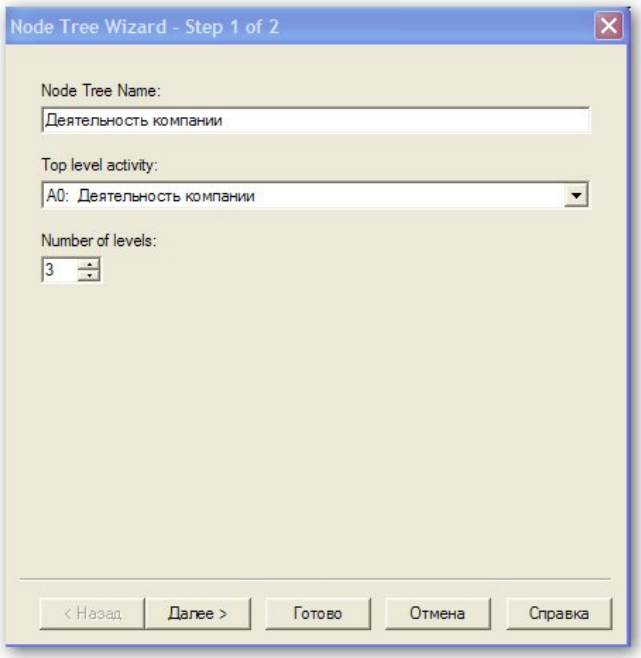

Рис. 2.35. Первый диалог построения диаграммы дерева узлов

В одной модели можно построить множество диаграмм дерева узлов, выбирая различные активности в качестве корня и задавая различное число уровней (глубину дерева). Имя и номер диаграммы дерева узлов по умолчанию совпадают с именем и номером корневой активности.

Во втором диалоге (рис. 2.36) задаются свойства диаграммы дерева узлов. Задание свойства **Bullet last level (Маркер последнего уровня)** означает, что последний уровень декомпозиции будет показан в виде списка (рис. 2.37). Группа свойств **Connection Style** (**Стиль соединения**) позволяет выбрать стиль соединительных линий – диагональные (по умолчанию) или ортогональные.

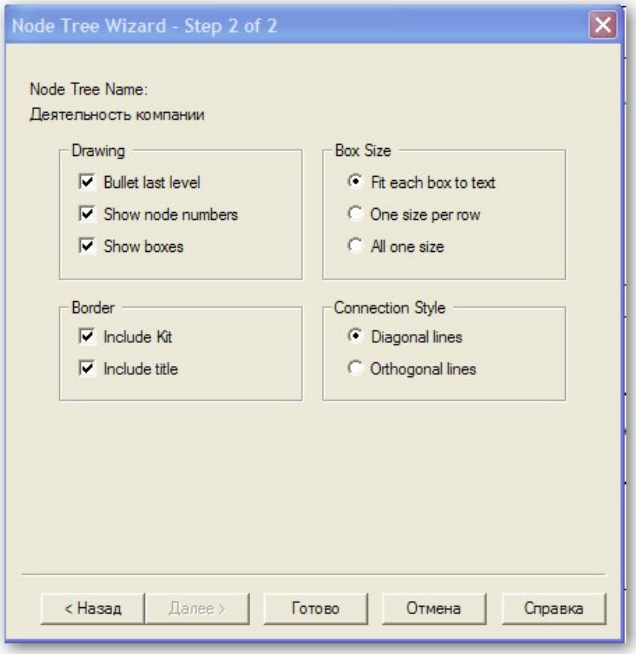

Рис. 2.36. Диалог настройки диаграммы дерева узлов

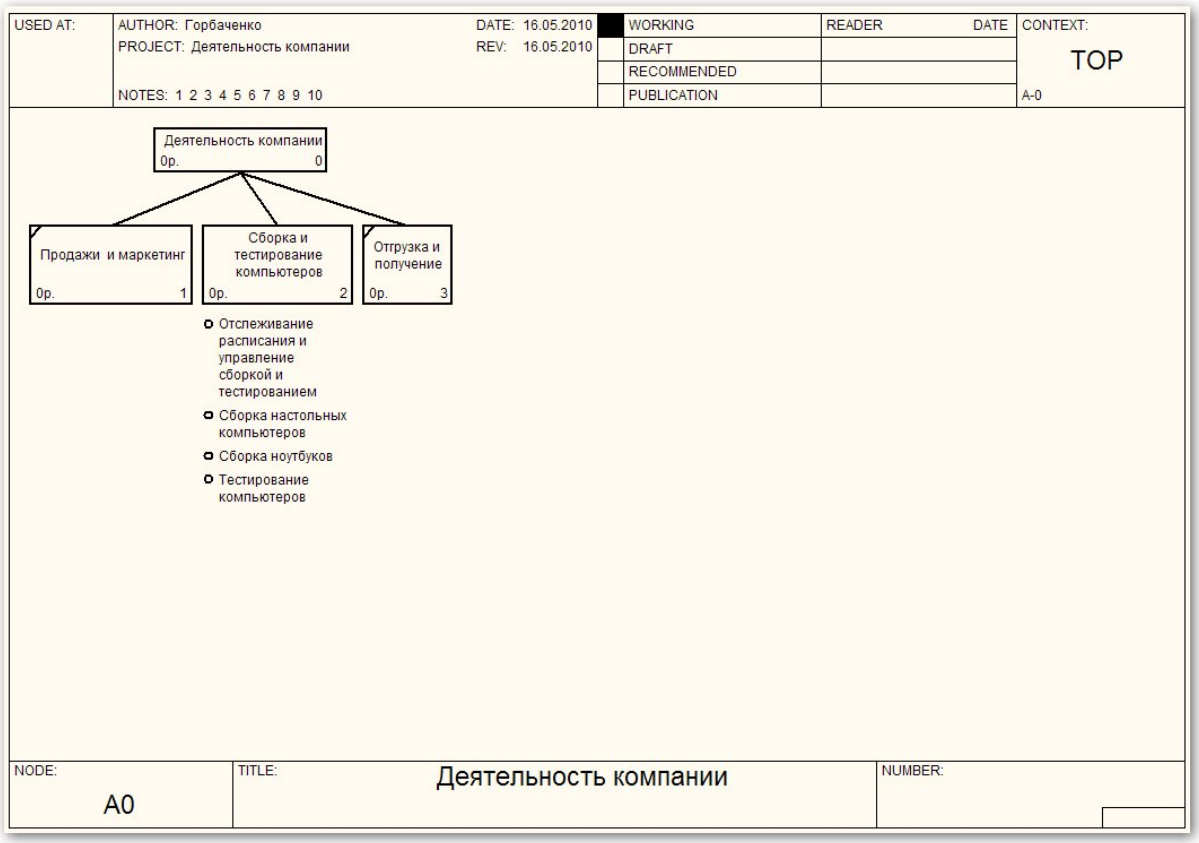

Рис. 2.37. Диаграмма дерева узлов

Для отображения нижнего уровня дерева узлов в виде прямоугольников необходимо отключить свойство **`Bullet last level`**. Полученное дерево показано на рис. 2.38.

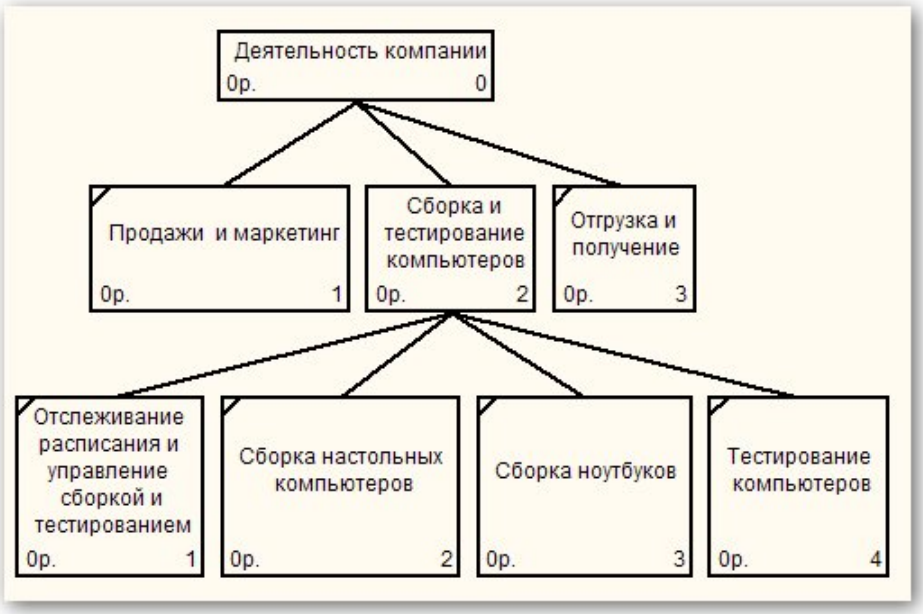

Рис. 2.38. Модифицированная диаграмма дерева узлов

2.4. Создание FEO-диаграммы

Диаграммы "только для экспозиции" (FEO) представляют собой просто картинки, отображающие альтернативные точки зрения, отдельные детали диаграммы и т. п. Эти диаграммы автоматически не поддерживают синтаксис стандарта IDEF0.

Для создания FEO-диаграммы выбираем пункт меню **`Diagram>Add FEO diagram`**. В первом диалоге (рис. 2.39) задаем имя диаграммы и выбираем, для какой диаграммы модели создается FEO-диаграмма.

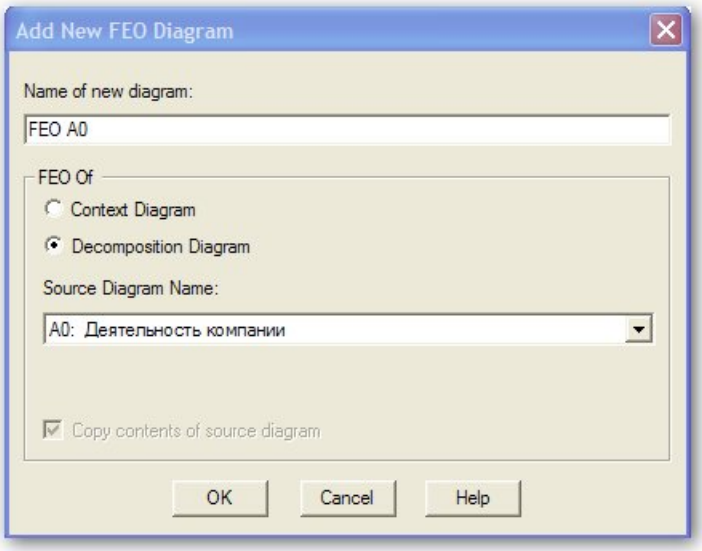

Рис. 2.39. Первый диалог создания FEO-диаграммы

По умолчанию на FEO-диаграмму копируется выбранная диаграмма модели. Удалим в скопированной диаграмме некоторые стрелки (рис. 2.40). Обратите внимание, что в диаграмме рис. 2.40 явно нарушены требования стандарта IDEF0.

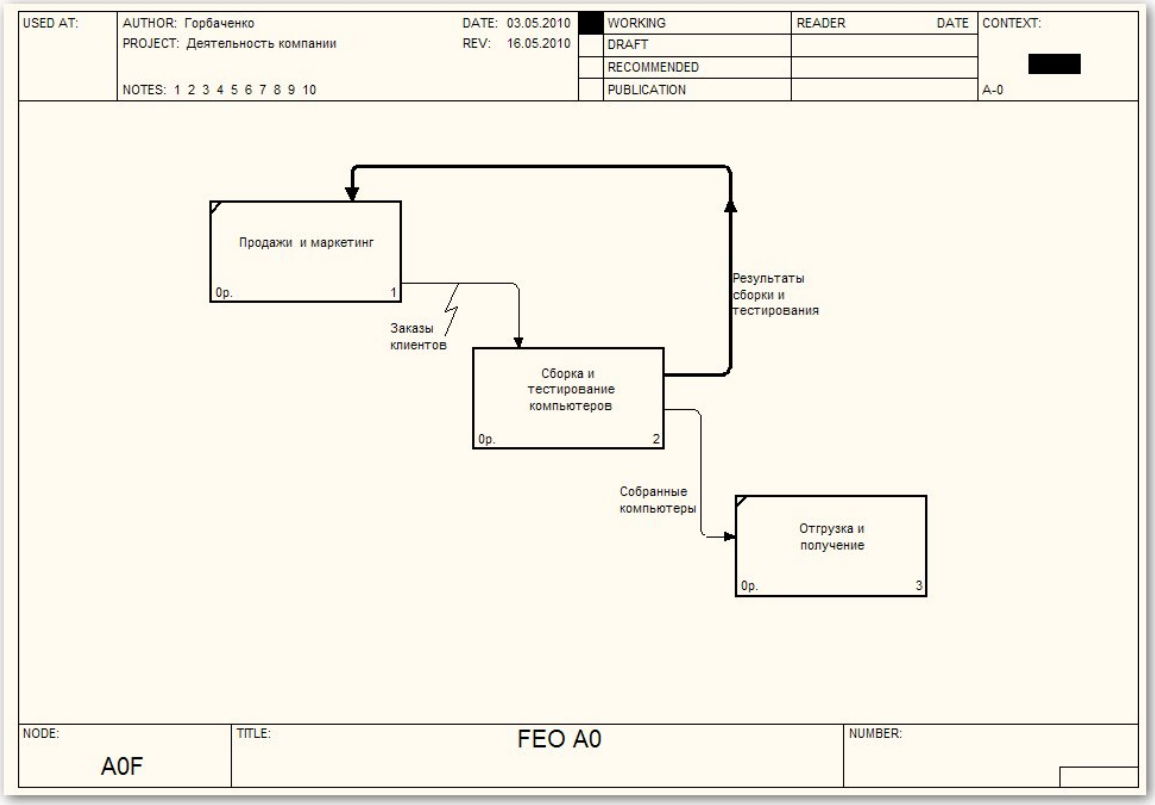

Рис. 2.40. FEO-диаграмма

Для перехода между стандартной диаграммой и FEO-диаграммой можно использовать навигатор и кнопку  . По нажатию на эту кнопку происходит переход к FEO-диаграмме и диаграмме дерева узлов на выбранном уровне модели.

2.5. Расщепление и слияние моделей

Слияние и расщепление моделей необходимо для коллективной работы над моделью. Руководитель проекта может создать декомпозицию верхнего уровня и провести расщепление модели на отдельные модели. Аналитики работают над отдельными моделями, а затем сливают отдельные модели в единую модель. Отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели.

Проведем расщепление активности "Сборка и тестирование компьютеров". На диаграмме *A0* правой кнопкой щелкаем на активности "Сборка и тестирование компьютеров" и выбираем из контекстного меню **`Split model`**. Возникает диалог **`Split Options`** (рис. 2.41).

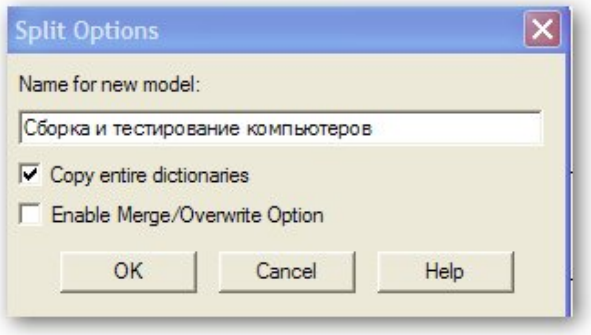

Рис. 2.41. Диалог **`Split Options`**

Задаем имя модели (лучше назвать по имени расщепляемой активности) и свойства. Зададим свойство **`Copy entire dictionaries`**, позволяющее копировать словари в отщепляемую модель.

После подтверждения в навигаторе появилась новая модель, на диаграмме *A0* модели "Деятельность компании" появилась стрелка вызова "Сборка и тестирование компьютеров".

Контекстная диаграмма модели "Сборка и тестирование компьютеров" имеет вид рис. 2.42. В старой модели активность "Сборка и тестирование компьютеров" стала недекомпозированной.

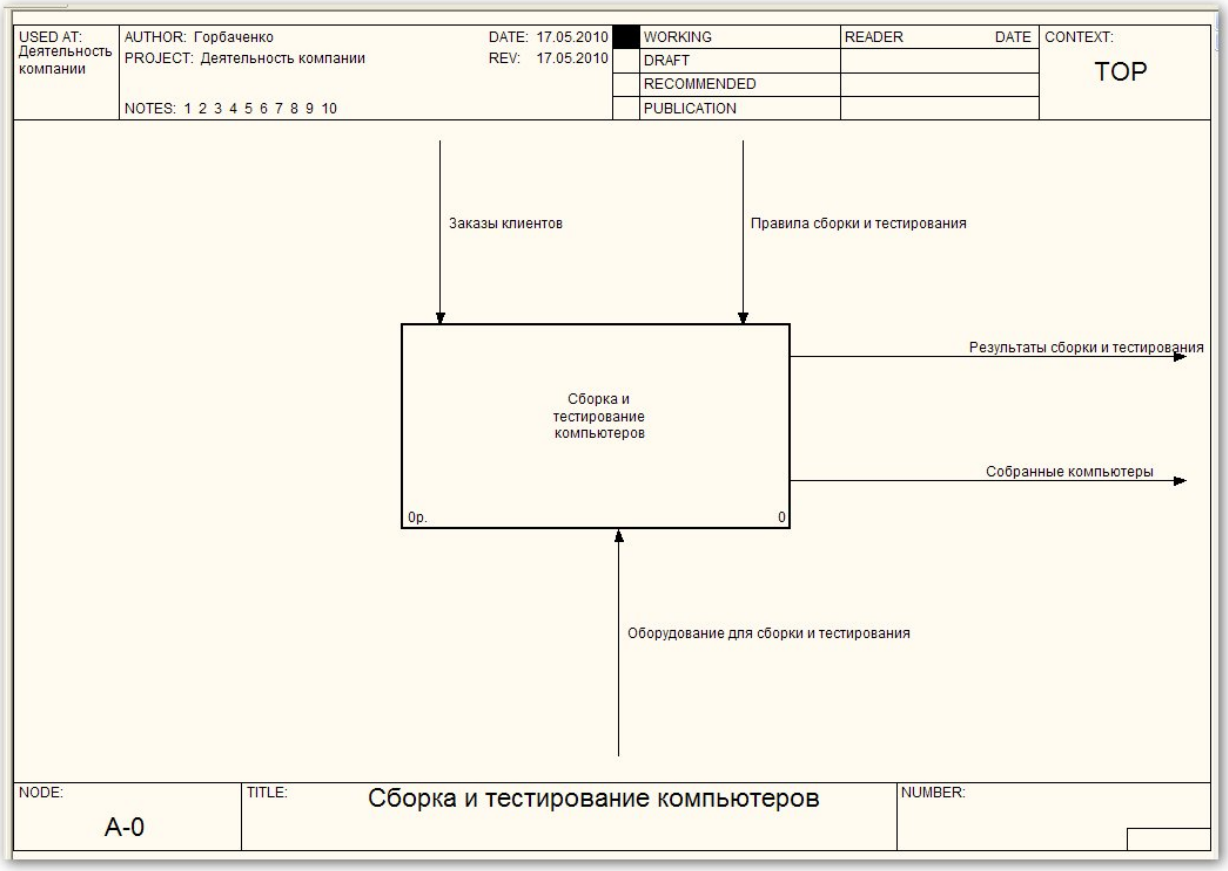

Рис. 2.42. Контекстная диаграмма модели "Сборка и тестирование компьютеров"

На контекстной диаграмме модели "Сборка и тестирование компьютеров" создадим новую стрелку "Неисправные компоненты". На диаграмме декомпозиции направим эту стрелку, как показано на рис. 2.43.

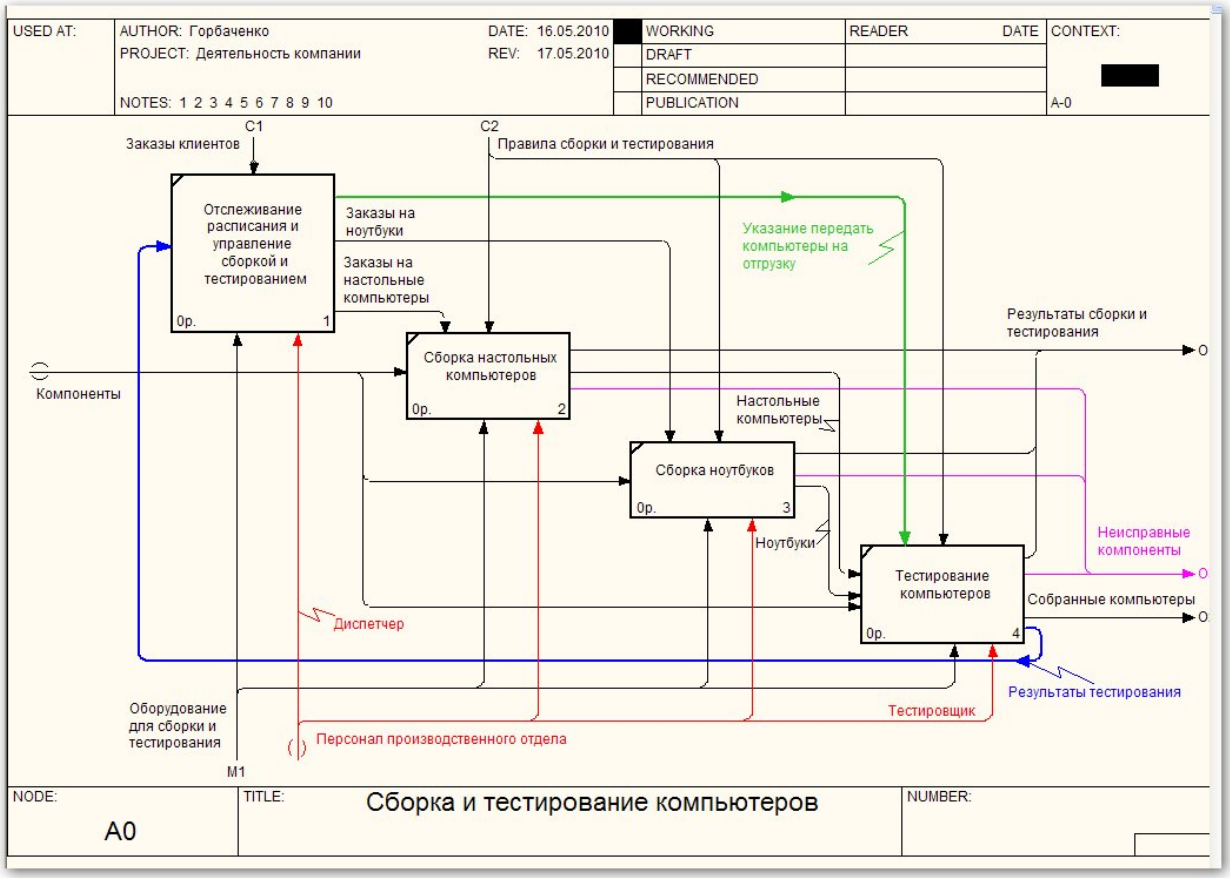

Рис. 2.43. Диаграмма декомпозиции модели "Сборка и тестирование компьютеров"

Теперь произведем слияние моделей. На диаграмме *A0* модели деятельность компании щелкнем правой кнопкой мыши по активности "Сборка и тестирование компьютеров" и из контекстного меню выберем **`Merge model`**. В диалоге слияния моделей (рис. 2.44) включаем опцию **`Cut/Paste entire dictionaries`** и щелкаем по кнопке **`OK`**.

Две модели слились. Модель "Сборка и тестирование компьютеров" осталась и может быть сохранена в отдельном файле. В модели "Деятельность компании" исчезла стрелка вызова. Появилась неразрешенная граничная стрелка "Неисправные компоненты". Эту стрелку туннелируем вручную. Полученная диаграмма декомпозиции модели "Деятельность компании" показана на рис. 2.45.

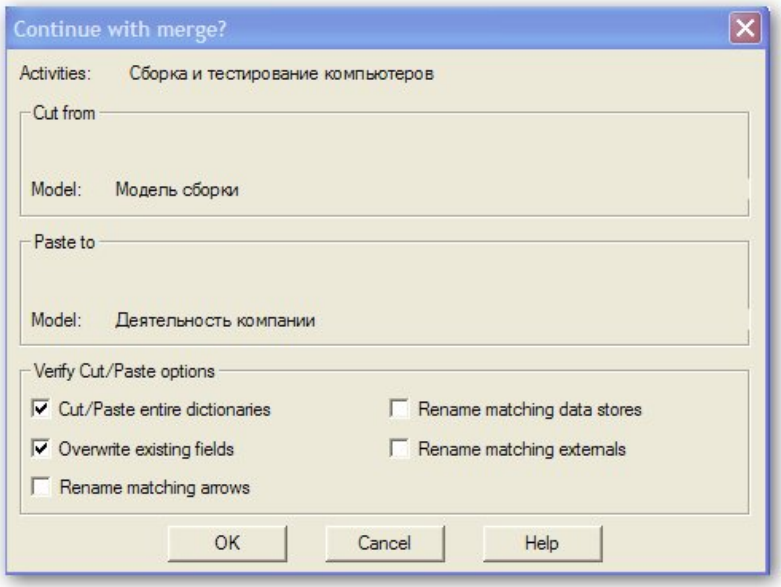

Рис. 2.44. Диалог слияния моделей

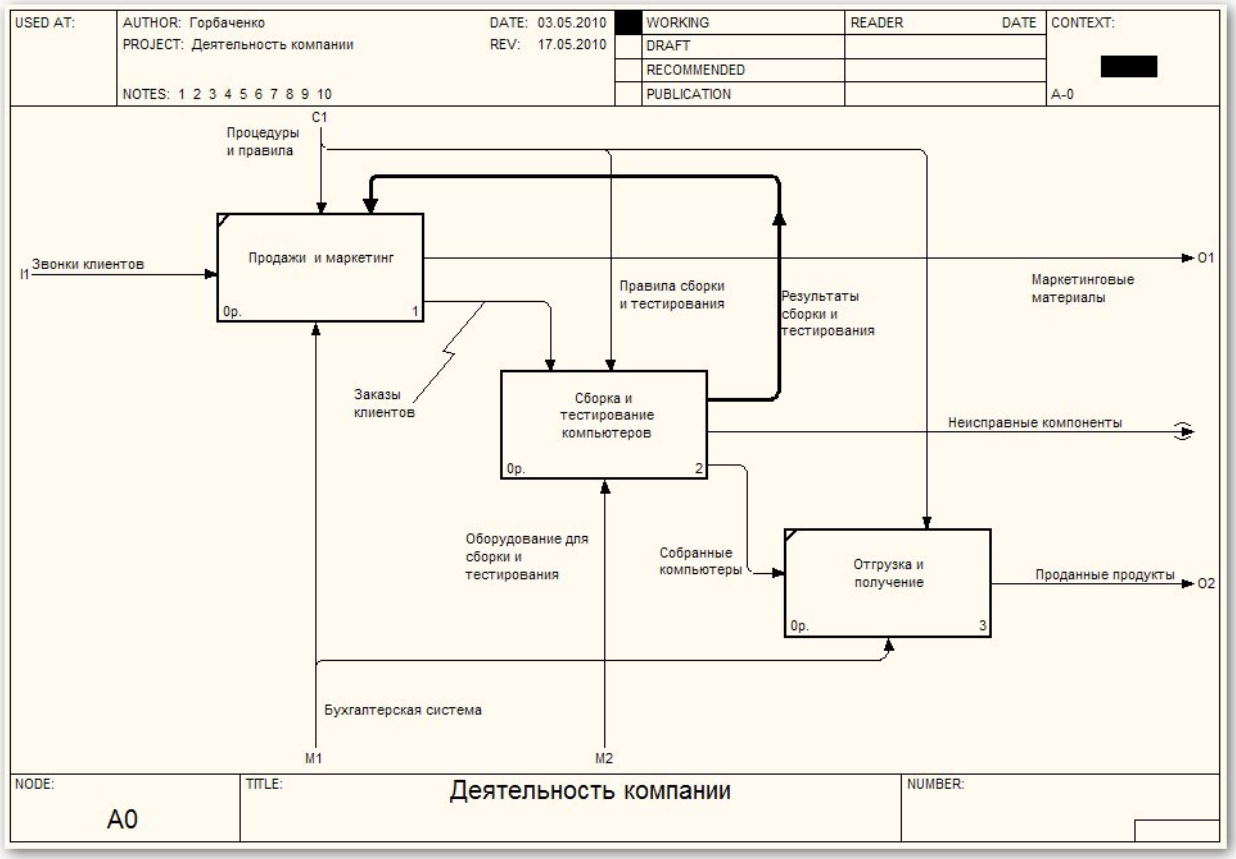

Рис. 2.45. Диаграмма декомпозиции модели "Деятельность компании"

2.6. Задание для самостоятельной работы

На основе методологии IDEF0 разработать модель работы системы «Поликлиника», обладающей следующими функциональными возможностями:

– хранение данных о пациентах, врачах, приемах, диагнозах, лечениях, лекарствах;

– корректировка данных о пациентах и врачах;

– поиск данных о пациентах и врачах по фамилии или адресу;

– добавление новых лекарств с описанием их свойств;

– поиск справочных данных по лекарствам;

– регистрация приемов на дому или в поликлинике.

– формирование статистической информации за отчетный период.

Входные данные системы:

– личные данные пациента;

– личные данные врача;

– сведения по приемам пациентов;

– описание лекарств;

– характеристика заболеваний.

Выходные данные системы:

– статистическая информация за отчетный период:

  – статистика осмотра пациентов на дому;

  – статистика заболеваний;

  – статистика приемов в поликлинике по специалистам;

– справочная информация:

  – сводные данные по врачу;

  – сводные данные по пациенту;

  – сведения о медицинских препаратах.

Модель должна включать в себя контекстную диаграмму, диаграммы декомпозиции, диаграмму дерева узлов, FEO-диаграмму.

3. СОЗДАНИЕ МОДЕЛИ В СТАНДАРТЕ DFD

3.1. Создание контекстной диаграммы

Методология DFD может быть использована *для создания новой модели и для декомпозиции работы*. Создадим новую модель работы "Оформление заказов". Для этого в диалоге создания модели (рис. 3.1) выберем тип модели DFD.

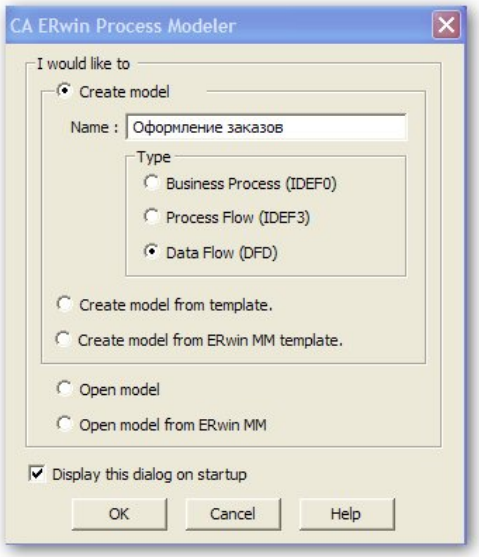

Рис. 3.1. Диалог создания модели

В открывшемся окне появляется единственная контекстная активность. Обратите внимание, что изображение активности немного отличается от ее изображения в методологии IDEF0: у активности закруглены углы. Построим контекстную диаграмму, как показано на рис. 3.2. Зададим имя и свойства активности.

Внесем две *внешние сущности*: источник и приемник. В нашем случае источником и приемником будет одна внешняя сущность "Клиенты" (рис. 3.3). С целью повышения наглядности покажем на диаграмме две внешние сущности с одинаковыми именами. Обратите внимание, что внешние сущности не участвуют в рассматриваемой работе и не подвергаются декомпозиции. Создается внешняя сущность с помощью кнопки .

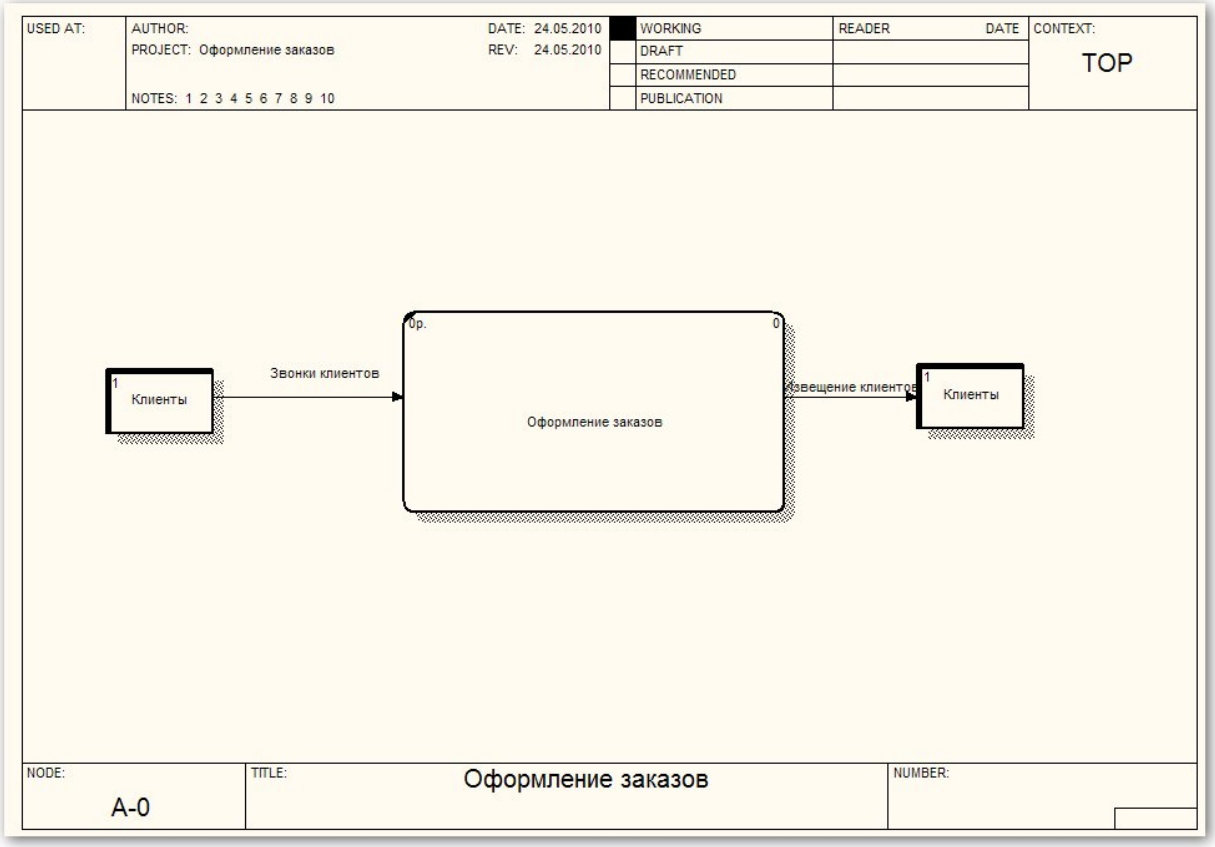

Рис. 3.2. Контекстная диаграмма в методологии DFD

Рис. 3.3. Изображение внешней сущности

3.2. Создание диаграммы декомпозиции

Произведем *декомпозицию* контекстной диаграммы. Оформление заказа начинается с телефонного звонка клиента. При оформлении заказа необходимо проверить, существует ли клиент в базе данных. Если клиента нет в базе, то необходимо занести данные о клиенте в базу клиентов. Далее производится оформление и внесение заказа в список заказов. При оформлении заказа используются как база клиентов, так и список продуктов. Заканчивается оформление заказа извещением по телефону клиента о результатах оформления заказа (можно было бы включить и уточнение заказа). Таким образом, в простейшем случае декомпозиция будет включать две активности: "Проверка и внесение клиента" и "Внесение заказа".

При создании диаграммы декомпозиции в диалоге **`Activity Box Count`** (рис. 3.4) следует выбрать тип диаграммы декомпозиции (IDEF0 выбрать нельзя).

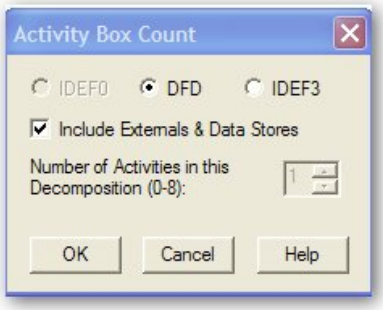

Рис. 3.4. Диалог **`Activity Box Count`** для методологии DFD

Свойство **`Include Externals & Data Stores`** означает, что на дочернюю диаграмму будут мигрировать внешние сущности и хранилища данных с родительской диаграммы. При этом родительская диаграмма копируется на дочернюю. На дочерней диаграмме надо удалить родительскую активность и создать необходимое число активностей. При удалении активности на дочерней диаграмме необходимо разрешить туннелированные стрелки на родительской диаграмме. Если это свойство не включено, то внешние сущности и хранилища не мигрируют на дочернюю диаграмму; можно также задать число сущностей на дочерней диаграмме. В нашем примере выберем миграцию внешних сущностей и хранилищ (на нашей родительской диаграмме нет хранилищ, но в принципе они возможны).

Построим диаграмму декомпозиции, как показано на рис. 3.5.

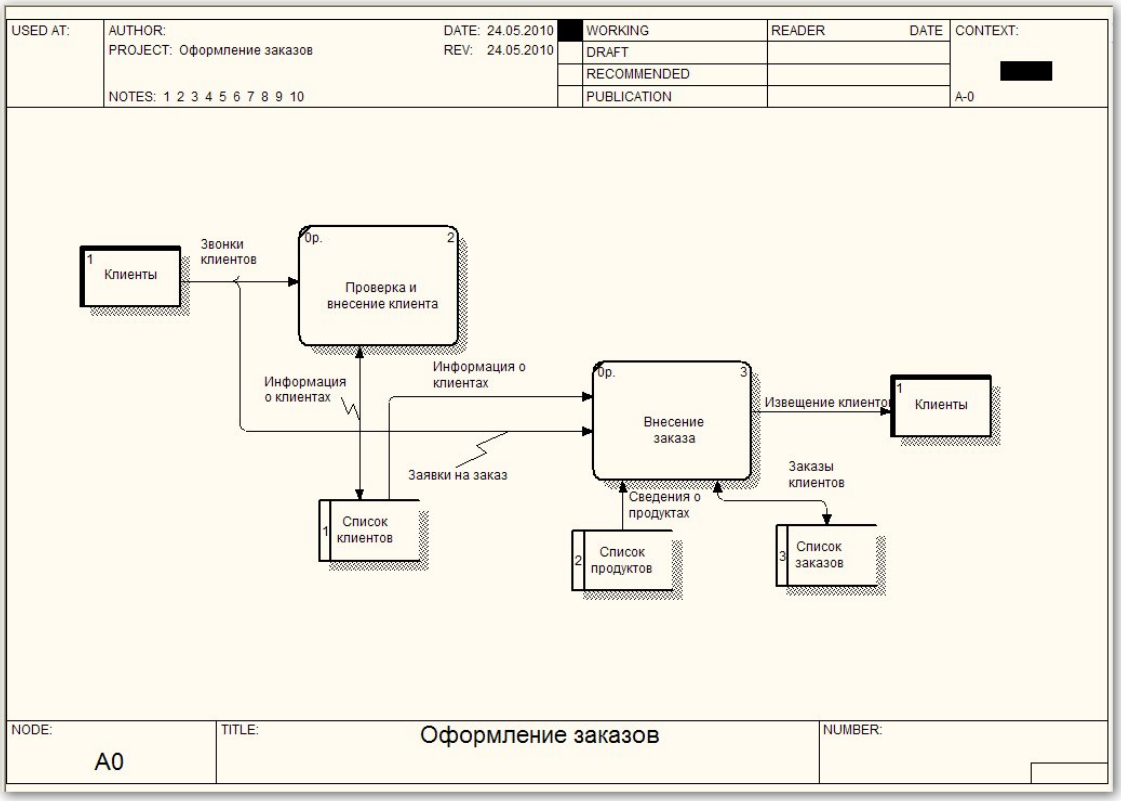

Рис. 3.5. Диаграмма декомпозиции в методологии DFD

На диаграмме декомпозиции присутствуют *хранилища* (рис. 3.6), которые создаются с помощью кнопки .

Рис. 3.6. Изображение хранилища

Некоторые стрелки на диаграмме декомпозиции являются *двунаправленными*. Сначала рисуется однонаправленная стрелка. Чтобы сделать стрелку двунаправленной, щелкните правой кнопкой мыши по стрелке, из контекстного меню выберите пункт **`Style`** и на вкладке **`Style`** меню свойств стрелки выберите вариант двунаправленной стрелки (**`Bidirectional`**).

3.3. Задание для самостоятельной работы

В модели, построенной по методологии IDEF0 (см. 2.6), декомпозируйте одну из активностей по методологии DFD. Учтите, что в методологии DFD нет стрелок управления и механизмов. При декомпозиции родительской активности по методологии DFD на диаграмму декомпозиции будут мигрировать стрелки с родительской диаграммы. Удалите их на диаграмме DFD, а на родительской диаграмме спрячьте в туннель (**`Change to Tunnel`**). На диаграмме декомпозиции нарисуйте внешние сущности и хранилища.

4. СОЗДАНИЕ МОДЕЛИ В СТАНДАРТЕ IDEF3

4.1. Создание диаграммы декомпозиции

В стандарте IDEF3 может быть создана контекстная диаграмма. Но обычно этот стандарт используют для декомпозиции активностей (по крайней мере, в BPwin). Проведем декомпозицию активности "Сборка настольных компьютеров". В диалоге **`Activity Box Count`** (рис. 4.1) выберем нотацию IDEF3 и число работ, равное 4.

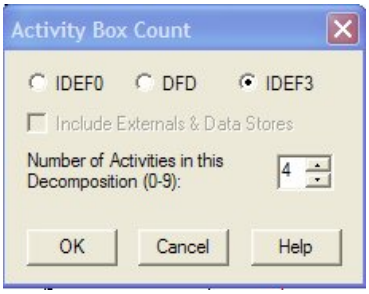

Рис. 4.1. Выбор нотации в диалоге **`Activity Box Count`**

Возникает диаграмма, содержащая 4 работы (UOW) – рис 4.2.

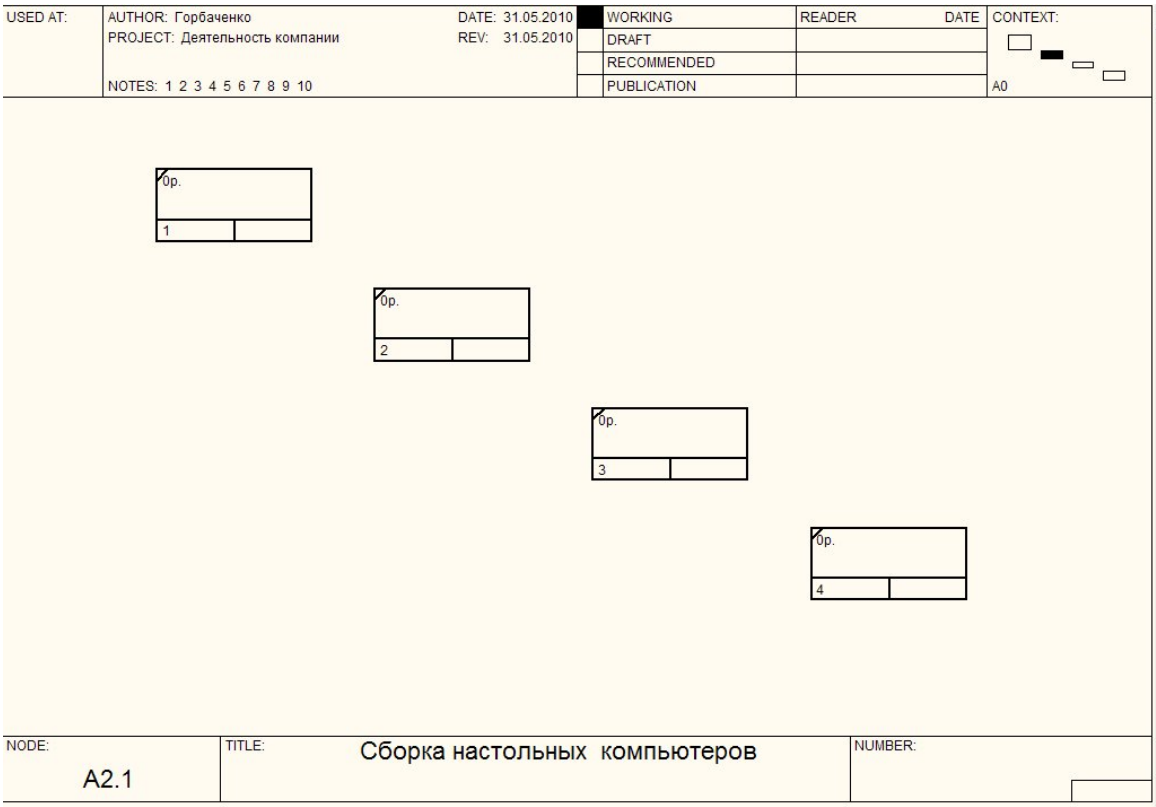

Рис. 4.2. UOW на диаграмме декомпозиции

Примем, что диаграмма декомпозиции должна содержать 7 работ:

– Подготовка компонентов.
– Установка материнской платы и винчестера.
– Установка карт-ридера.
– Установка DVD.
– Установка сетевой карты.
– Инсталляция операционной системы (ОС).
– Инсталляция дополнительного программного обеспечения (ПО).

С помощью кнопки  добавим к диаграмме еще три работы. В диалоге **Activity Properties** зададим имена и свойства работ. На вкладке **UOW** (рис. 4.3) зададим объекты (**Objects**), с которыми работает UOW. Например, для работы "Подготовка компонентов" можно указать подготавливаемые компоненты компьютера.

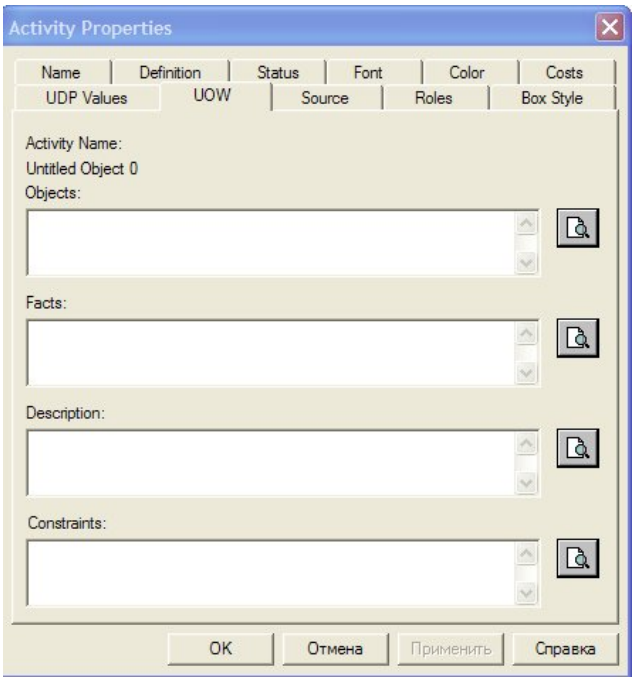

Рис. 4.3. Вкладка **UOW** в диалоге **Activity Properties**

Задаются факты (**Facts**) – данные, которые используются работой, например, какие операционные системы доступны. В ограничениях (**Constrains**) указываются ограничения на работу, например, порядок выполнения работ, необходимость в дополнительных работах.

С помощью кнопки  создадим объект "Ссылка". Зададим имя ссылки "Компоненты" и свяжем его стрелкой с работой "Подготовка компонентов". В контекстном меню свойств стрелки необходимо с помощью радиокнопки задать тип стрелки **Referent** (рис. 4.4).

Свяжем стрелкой работы "Подготовка компонентов" и "Установка материнской платы". Очевидно, что после подготовки компоненты передаются на сборку. Поэтому стиль стрелки изменим на **Object Flow** (Поток объек-

тов). В IDEF3 имена стрелок могут отсутствовать, но BPwin требует задавать имена.

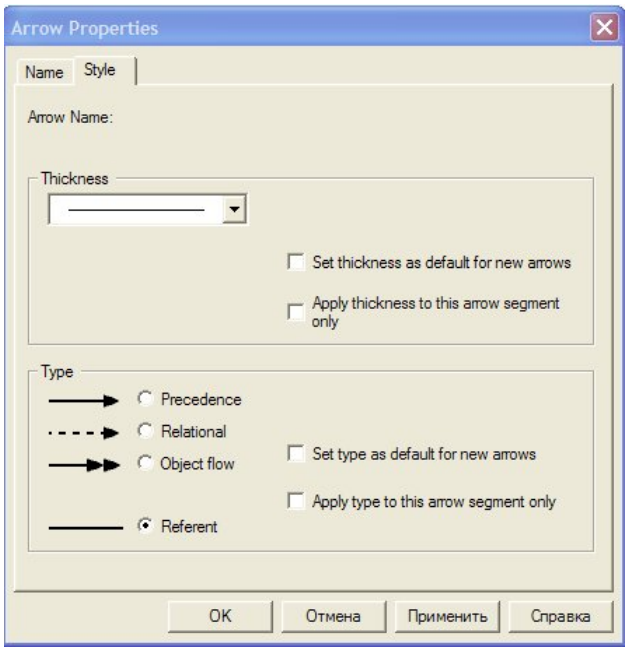

Рис. 4.4. Задание типа стрелки

Далее необходимо внести в диаграмму перекрестки. Окончательный вид диаграммы показан на рис. 4.5.

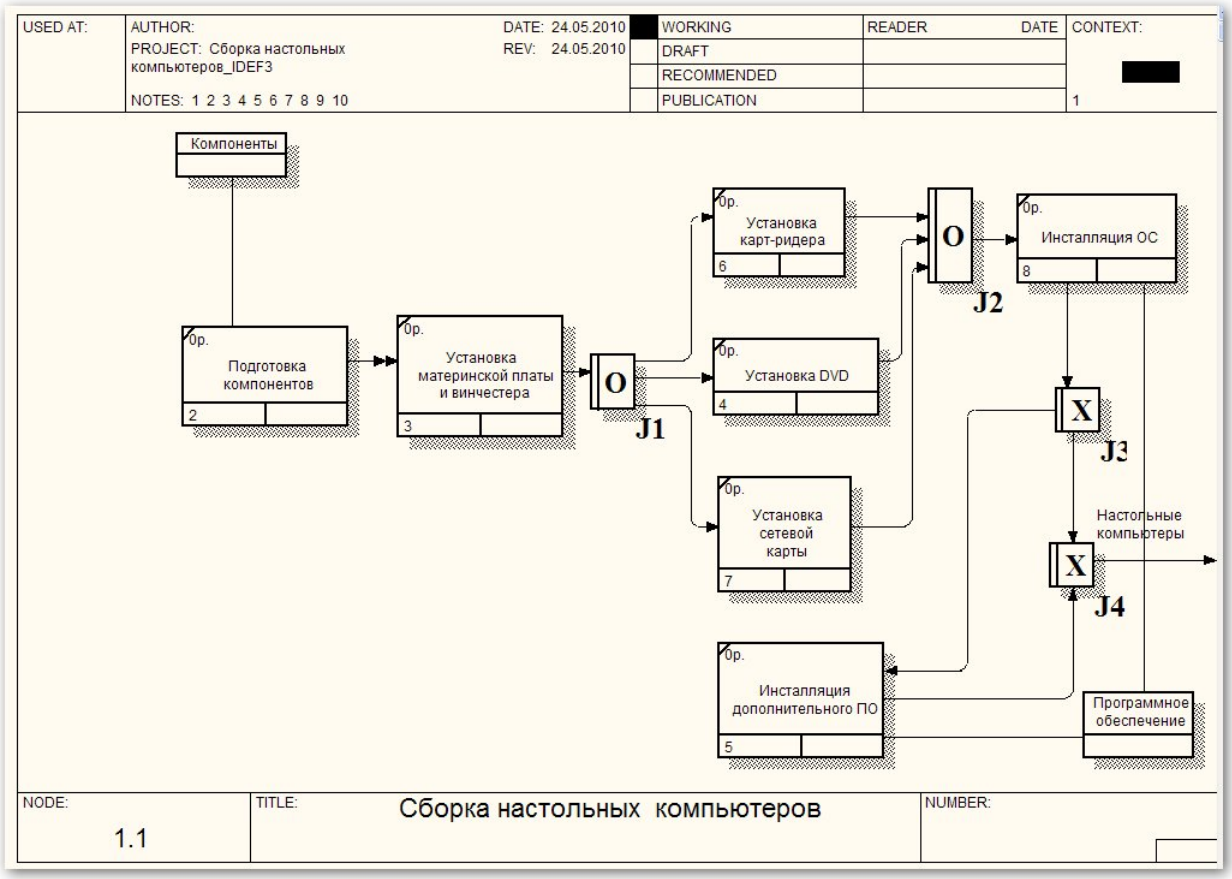

Рис. 4.5. Результат построения диаграммы декомпозиции

При выборе типа перекрестка необходимо руководствоваться свойствами перекрестков (табл. 1.3). Для создания перекрестка используется кнопка  . Левой клавишей мыши укажем на свободной области диаграммы место, где будет добавлен этот перекресток, откроется окно для выбора типа перекрестка (рис. 4.6).

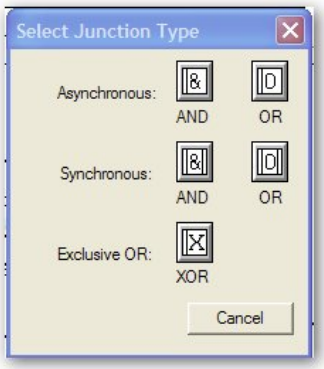

Рис. 4.6. Окно выбора типа перекрестка

4.2. Задание для самостоятельной работы

В модели, построенной по методологии IDEF0 (см. 2.6), декомпозируйте одну из активностей по методологии IDEF3.

5. МОДЕЛЬ "СУЩНОСТЬ-СВЯЗЬ"

5.1. Основные понятия модели "сущность-связь". Сущности и атрибуты

На этапе *инфологического проектирования* базы данных должна быть построена модель предметной области, не привязанная к конкретной СУБД, понятная не только разработчикам информационной системы, но и экономистам, менеджерам и другим специалистам. В то же время модель предметной области должна максимально точно отражать семантику предметной области [20], выявлять бизнес–правила и позволять легко перейти к модели данных конкретной СУБД.

Такими моделями являются *модели "сущность-связь"*. Зачастую эти модели называют моделями данных. Но этот термин неудачен, т. к. перекликается с термином "модель данных" [21]. Известно несколько методологий построения моделей "сущность-связь". Наибольшее распространение получила методология IDEF1X [2, 19, 22, 23]. Рассмотрим построение моделей "сущность-связь", ориентируясь на продукт CA ERwin Data Modeler [2, 24, 25]. Для простоты будем использовать старое название продукта: ERwin.

ERwin имеет два уровня представления модели:

– *Логический уровень*, соответствующий инфологическому этапу проектирования и не привязанный к конкретной СУБД. Модели логического уровня оперируют с понятиями сущностей, атрибутов и связей, которые на этом уровне именуются на естественном языке (в нашем случае – на русском) так, как они называются в реальном мире.

– *Физический уровень* – это отображение логической модели на модель данных конкретной СУБД. Одной логической модели может соответствовать несколько физических моделей. Причем, Erwin (как и другие CASE-системы проектирования баз данных) позволяет автоматизировать отображение логической модели на физическую.

Модель "сущность-связь" строится в виде диаграммы "сущность-связь", основными компонентами которой являются *сущности* (Entity) и *связи* (Relationship). Отсюда происходят часто используемые названия модели (ER-модель) и диаграммы (ER-диаграмма).

Сущность – это абстракция множества предметов или явлений реального мира, информацию о которых надо сохранить. Все *экземпляры* сущности имеют одинаковые характеристики и подчиняются одним и тем же правилам поведения. Например, можно выделить сущность **СТУДЕНТ**. Экземплярами сущности **СТУДЕНТ** будут данные о конкретных студентах. Сущность должна иметь *имя* – существительное в единственном числе.

Сущности обладают определенными свойствами – атрибутами. *Атрибут* – это абстракция *одной* характеристики объекта или явления реального мира. Каждый атрибут должен иметь *имя* – существительное в единственном числе, и получать значение из некоторого множества допустимых значений – *домена*.

У каждой сущности должен быть выделен идентификатор, или первичный ключ. *Первичный ключ* – это один или несколько атрибутов, однозначно определяющих каждый экземпляр сущности. Если первичный ключ состоит из нескольких атрибутов, то он называется *составным*. Первичный ключ не должен изменяться и принимать неопределенное значение (NULL). Ключ должен быть *компактным*, т. е. не содержать слишком много атрибутов. Сущность может иметь несколько *потенциальных ключей*, из которых должен быть выбран первичный ключ. Иногда приходится *использовать искусственный первичный ключ* (некоторый номер или код), когда ключ содержит слишком много атрибутов (в пределе каждый экземпляр сущности может определяться всем множеством атрибутов). Используется также понятие *внешнего ключа*. Внешний ключ – это первичный ключ другой сущности, который мигрирует (копируется) в сущность и служит для связи сущностей.

Пример сущности показан на рис. 5.1.

Каждая сущность должна сопровождаться *описанием*. Описание сущности должно объяснять ее смысл, а не то, как будет использоваться информация данной сущности. Описание должно быть ясным, полным и непротиворечивым, понятным специалистам предметной области.

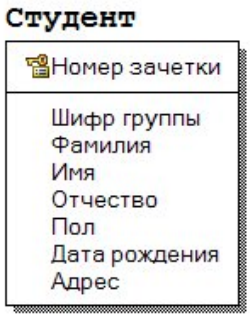

Рис. 5.1. Пример сущности

Сущности и атрибуты выделяются в результате анализа требований к системе. При выборе атрибутов целесообразно придерживаться следующих правил (не входящих в IDEF1X), позволяющих перейти к физической модели, находящейся в третьей нормальной форме:

1. Атрибуты должны быть неделимыми.

2. Каждый неключевой атрибут должен полностью зависеть от составного ключа, а не от части ключа.

3. Не должно существовать транзитивных зависимостей атрибутов от ключа. Например, если ключ **ТАБЕЛЬНЫЙ_НОМЕР** определяет атрибут **НОМЕР_ОТДЕЛА**, а **НОМЕР_ОТДЕЛА** определяет **ТЕЛЕФОН**, то **ТАБЕЛЬНЫЙ_НОМЕР** транзитивно определяет **ТЕЛЕФОН**.

5.2. Связи

Связи между объектами реального мира отражаются в виде связей (*отношений*, *ассоциаций*) между сущностями. *Отношение* – это ассоциация или "связь" между двумя сущностями. Отношение представляется в модели линией, соединяющей две сущности, и именем отношения – глагольной конструкцией, которая описывает, как две сущности зависят друг от друга. Имена сущностей, соединенные именем отношения, должны образовывать осмысленную фразу, описывающую бизнес-правило отношения. Например, **СТУДЕНТ <Обучается в> УЧЕБНАЯ ГРУППА**. В примере имя отношения показано в угловых скобках. Отношения двунаправлены, поэтому должны иметь имена в каждом направлении.

Отношение обладает следующими *свойствами*:

– степень,

– направленность,

– тип,

– мощность,

– обязательность.

Степень отношения представляет собой число сущностей, ассоциированных с отношением. Чаще всего используются *бинарные отношения*, связывающие две сущности. *Унарные*, или *рекурсивные отношения* представляют случаи, когда экземпляр сущности связан с другим экземпляром той же самой сущности. Часто унарные или рекурсивные отношения рассматриваются как бинарные рекурсивные отношения, связывающие экземпляр сущности с другим ее экземпляром.

Направленность отношения указывает на исходную сущность в отношении. Сущность, из которой отношение исходит, называется *родительской* сущностью. Сущность, в которой отношение заканчивается, называется *подчиненной* (*дочерней*) сущностью. Направленность отношения определяется взаимосвязью между сущностями и зависит от типа и мощности отношения (см. ниже). В отношении между независимой и зависимой сущностями отношение исходит из независимой сущности и заканчивается в зависимой сущности. Если обе сущности независимые, отношение симметрично. В отношении *один-ко-многим* родительской является сущность, входящая в отношение однократно. Отношения *многие-ко-многим* симметричны. Ключ родительской сущности *мигрирует* (повторяется) в дочерней сущности. Такой мигрировавший ключ в дочерней сущности называется *внешним ключом*. Как мы увидим далее, внешний ключ в зависимости от типа связи может стать частью составного ключа дочерней сущности или неключевым атрибутом дочерней сущности. С помощью внешнего ключа экземпляр дочерней сущности *ссылается* на соответствующий экземпляр родительской сущности.

В ERwin отношение между двумя сущностями, или сущности самой с собой, может принадлежать к одному из следующих *типов*:

– идентифицирующее отношение,

– неидентифицирующее отношение,
– типизирующее отношение,
– отношение многие-ко-многим,
– рекурсивное отношение.

Каждый тип отношений определяет поведение атрибутов первичного ключа, когда они мигрируют из родительской сущности в подчиненную. Ниже будут описаны особенности каждого из типов отношений.

Идентифицирующим является отношение между двумя сущностями, в котором каждый экземпляр подчиненной сущности идентифицируется значениями атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности зависит от родительской сущности и не может существовать без экземпляра родительской сущности. В идентифицирующем отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в атрибуты подчиненной, чтобы стать там атрибутами первичного ключа. На рис. 5.2 представлено идентифицирующее отношение между сущностями **Заказ** и **Состав заказа**. В сущности **Заказ** использован искусственный первичный ключ **ID заказа** (Идентификатор заказа), который мигрировал в дочернюю сущность **Состав заказа** и стал там первичным ключом. В реальной диаграмме атрибут **Заказчик**, скорее всего, будет являться мигрировавшим ключом сущности, описывающей заказчиков. Пример рис. 5.2 показывает, что дочерняя сущность не может существовать без родительской. Этот пример также демонстрирует, как показать переменный состав заказа: в заказ может входить произвольное количество товаров.

Рис. 5.2. Пример идентифицирующего отношения

Неидентифицирующим называется отношение между двумя сущностями, в котором каждый экземпляр подчиненной сущности не зависит от значений атрибутов родительской сущности. Это означает, что экземпляр подчиненной сущности не зависит от родительской сущности и может существовать без экземпляра родительской сущности. В неидентифицирующем от-

ношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в подчиненную, чтобы стать там *неключевыми* атрибутами. На рис. 5.3 показано неидентифицирующее отношение между сущностями **Группа** и **Студент**. Каждое из этих отношений имеет собственный первичный ключ. Первичного ключа родительской сущности **Группа** мигрировал в подчиненную сущность **Студент** и стал там неключевым атрибутом.

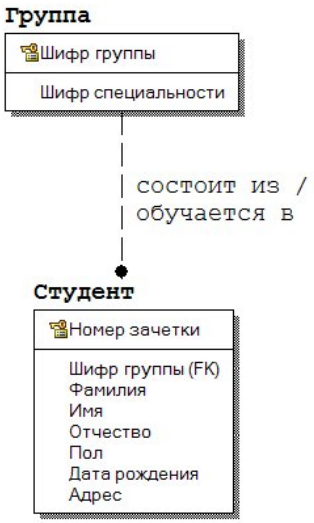

Рис. 5.3. Пример обязательного неидентифицирующего отношения

На примере рис. 5.3 рассмотрим также понятие *обязательности* отношения. Неидентифицирующее отношение называется *обязательным* (No Nulls), если все экземпляры дочерней сущности должны участвовать в отношении. Неидентифицирующее отношение называется *необязательным* (Nulls Allowed), если некоторые экземпляры дочерней сущности могут не участвовать в отношении. Очевидно, что студент должен принадлежать одной из учебных групп. На рис. 5.4 показан пример необязательно отношения: допускается, что сотрудник может не принадлежать ни одному из отделов.

Рис. 5.4. Пример необязательного неидентифицирующего отношения

Типизирующими называются отношения между родительской и одной или более подчиненными сущностями, когда сущности разделяют общие характеристики. Такие отношения называются еще *иерархией наследования* или *иерархией категорий*. Типизирующие отношения используются в том случае, когда экземпляр родительской сущности определяет различные наборы атрибутов в подчиненных сущностях. Например, имеются различные категории сотрудников, отличающиеся только небольшим количеством атрибутов (рис. 5.5). Для каждой категории необходимо указать *дискриминатор* – атрибут родительской сущности, показывающий, как отличить одну категориальную сущность от другой. На рис. 5.5 дискриминатором является атрибут **Тип**. На рис. 5.5 показана *полная категория*, т. е. каждый экземпляр сущности сотрудник относится к одной из перечисленных категорий. Возможна *неполная категория*, когда существуют экземпляры родительской сущности, не имеющие соответствующих экземпляров в дочерних сущностях (значок категории содержит одну горизонтальную линию).

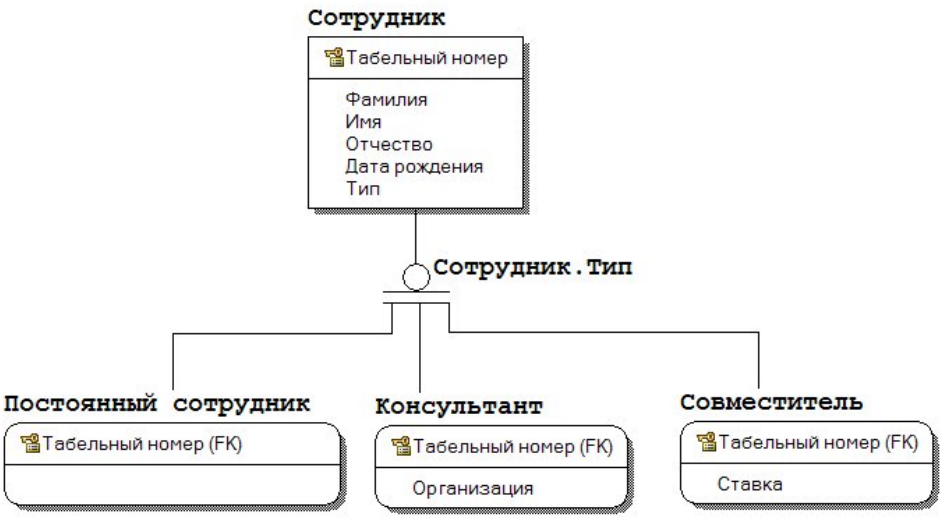

Рис. 5.5. Пример полной категории иерархии наследования

Отношения *многие-ко-многим* возникают тогда, где один экземпляр одной сущности связан с несколькими экземплярами другой, и один экземпляр этой другой сущности также связан с несколькими экземплярами первой сущности. Эти отношения также называют неспецифическими. Отношения многие-ко-многим *используются только на логическом уровне*. На физическом уровне эти отношения реализуются за счет использования *ассоциативной сущности*, содержащей ключи родительских сущностей и, возможно, дополнительные атрибуты. Для большей наглядности диаграммы желательно ввести ассоциативные сущности на логическом уровне. На рис. 5.6 показан пример связи "многие-ко-многим", а на рис. 5.7 – пример использования ассоциативной сущности.

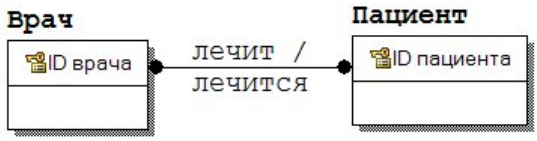

Рис. 5.6. Пример связи "многие-ко-многим"

Рис. 5.7. Пример использования ассоциативной сущности

Рекурсивное отношение – это *неидентифицирующее* отношение между двумя сущностями, которое указывает, что экземпляр сущности может быть связан с другим экземпляром той же самой сущности. При рекурсивном отношении родительская и подчиненная сущности совпадают. На рис. 5.8 показаны примеры двух реализаций рекурсивного отношения для сущности **Сотрудник**, с использованием *имени роли* и без него. Имя роли показывает, какую роль играет внешний ключ в сущности. В данном примере внешний ключ задает табельный номер. Обратите внимание, что ERwin "унифицирует" атрибуты внешнего ключа и первичного ключа, когда имя роли не используется. Использование имени роли приводит к размещению внешнего ключа в качестве неключевого атрибута.

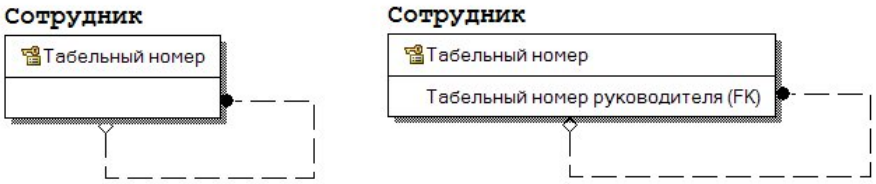

Рис. 5.8. Примеры реализации рекурсивных отношений
с использованием имени роли и без него в сущности **Сотрудник**

Мощность отношения задает максимальное число экземпляров одной сущности, которые могут быть связаны с экземплярами другой сущности. Мощность отношения определяется для обеих сторон отношения – для исходной и завершающей сущностей. Количество элементов определяет максимальное количество экземпляров сущностей, участвующих в отношении, в

то время как обязательность определяет минимальное число экземпляров. Количество элементов часто выражается как один или много. Один и много могут появляться в трех различных комбинациях:

Один-к-одному (1:1) – один и только один экземпляр сущности связан с одним и только одним экземпляром другой сущности.

Один-ко-многим (1:N) – один и только один экземпляр родительской сущности связан со многими экземплярами подчиненной сущности.

Многие-ко-многим (M:N) – много экземпляров одной сущности связаны со многими экземплярами другой сущности (также называется неспецифическим отношением).

В отношении *один-к-одному* один и только один экземпляр сущности связан с одним и только одним экземпляром другой сущности. Это редкий случай отношения, следует рассмотреть возможность объединения двух отношений в одно. Но, например, в отношении сущностей **Факультет** и **Сотрудник** целесообразнее установить связь один-к-одному, чем заносить данные о декане в сущность **Факультет.**

В отношении *один-ко-многим* один и только один экземпляр родительской сущности связан со многими экземплярами дочерней сущности. Это наиболее распространенное отношение.

Отношение *многие-ко-многим* уже было рассмотрено.

В ERwin отношение изображается линией с точкой на конце "много". Кроме общего случая мощности отношения 0, 1 или много можно задать частные случаи отношений: 1 или много (обозначается буквой **p**), 0 или 1 (обозначается буквой **z**), конкретное число экземпляров дочерней сущности (обозначается числом экземпляров, например, 6).

В ERwin реализована также методология IE (Information Engineering), которая принципиально не отличается от методологии IDEF1X [26].

5.3. Правила ссылочной целостности

Целостность данных является понятием базы данных [20, 21]. ERwin позволяет в модели "сущность-связь" задать *правила ссылочной целостности* (возможно также задание ограничений на значения атрибутов, но мы не рассматриваем эту возможность). При генерации схемы базы данных ERwin генерирует правила декларативной ссылочной целостности и триггеры. То есть, правила ссылочной целостности задаются в модели "сущность-связь", а реализуются в построенной по этой модели реляционной базе данных. Соответствие между моделью и базой данных легко устанавливается, учитывая, что сущностям модели соответствуют отношения реляционной базы данных, а экземплярам сущностей – кортежи отношений.

Так как внешние ключи кортежей дочернего отношения служат ссылками на соответствующие кортежи родительского отношения, то эти ссылки не должны указывать на несуществующие кортежи. Это определяет следующее *правило целостности внешних ключей* (правило ссылочной целостности): для каждого значения внешнего ключа должно существовать соответст-

вующее значение первичного ключа в родительском отношении.

Ссылочная целостность может нарушиться в результате операций с кортежами отношений. Таких операций три: вставка, обновление и удаление кортежей в отношениях. Рассмотрим эти операции для родительского и дочернего отношений [27].

При операциях с кортежами *родительского отношения* возможны следующие ситуации:

1. *Вставка кортежа в родительское отношение*. Так как допустимо существование кортежей родительского отношения, на которые нет ссылок из дочерних отношений, то *вставка кортежа в родительское отношение не нарушает ссылочной целостности*.

2. *Обновление кортежа родительского отношения*. При обновлении кортежа родительского отношения может измениться значение ключа. Если есть экземпляры дочернего отношения, ссылающиеся на обновляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. *Обновление кортежа в родительском отношении может привести к нарушению ссылочной целостности, если это обновление затрагивает значение ключа*.

3. *Удаление кортежа родительского отношения*. При удалении кортежа родительского отношения удаляется значение ключа. Если есть кортежи дочернего отношения, ссылающиеся на удаляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. *Удаление кортежа родительского отношения может привести к нарушению ссылочной целостности*.

При операциях с кортежами *дочернего отношения* возможны следующие ситуации:

1. *Вставка кортежа дочернего отношения может привести к нарушению ссылочной целостности, если вставляемое значение внешнего ключа некорректно*.

2. *Обновление кортежа дочернего отношения может привести к нарушению ссылочной целостности при некорректном изменении значения внешнего ключа.*

3. *При удалении кортежа дочернего отношения ссылочная целостность не нарушается*.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

1) обновление кортежа в родительском отношении;

2) удаление кортежа в родительском отношении;

3) вставка кортежа в дочернее отношение;

4) обновление кортежа в дочернем отношении.

Существуют *две основные стратегии поддержания ссылочной целостности*:

1) **RESTRICT** (**ОГРАНИЧИТЬ**) – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи дочернего отношения,

связанные с некоторыми кортежами родительского отношения.

2) **CASCADE** (**КАСКАД**) – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. Так как дочернее отношение может быть родительским для некоторого третьего отношения, то может потребоваться выполнение каскадной стратегии и для этой связи и т.д. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

ERwin реализует также дополнительные стратегии поддержания ссылочной целостности (если они реализованы в целевой СУБД):

1) **NONE** (**НИКАКОЙ**) – никаких операций по поддержке ссылочной целостности не выполняется. В этом случае в дочернем отношении могут появляться некорректные значения внешних ключей, и вся ответственность за целостность базы данных ложится на приложение.

2) **SET NULL** (**УСТАНОВИТЬ В NULL**) – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей заменять на неопределенные значения (null-значения). При этом кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения.

3) **SET DEFAULT** (**УСТАНОВИТЬ ПО УМОЛЧАНИЮ**) – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. При этом должен существовать кортеж родительского отношения, первичный ключ которого принят как значение по умолчанию для внешних ключей. Этот кортеж нельзя удалять из родительского отношения, и в этом кортеже нельзя изменять значение ключа. Кроме того, как и в предыдущем случае, кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения.

Рассмотрим *применение стратегии* поддержания ссылочной целостности [27].

При обновлении кортежа в родительском отношении допустимы следующие стратегии:

1) **RESTRICT** – не разрешать обновление, если имеется хотя бы один кортеж дочернего отношения, ссылающийся на обновляемый кортеж родительского отношения.

2) **CASCADE** – выполнить обновление и каскадно изменить значения внешних ключей во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж.

3) **SET NULL** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внеш-

них ключей на null-значение.

4) **SET DEFAULT** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

5) **NONE** – выполнить обновление, не обращая внимания на нарушения ссылочной целостности.

При удалении кортежа в родительском отношении допустимы стратегии:

1) **RESTRICT** – не разрешать удаление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на удаляемый кортеж.

2) **CASCADE** – выполнить удаление и каскадно удалить кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж.

3) **SET NULL** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на null-значение.

4) **SET DEFAULT** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

5) **NONE** – выполнить удаление, не обращая внимания на нарушения ссылочной целостности.

При вставке кортежа в дочернее отношение возможны стратегии:

1) **RESTRICT** – не разрешать вставку, если внешний ключ во вставляемом кортеже не соответствует ни одному значению потенциального ключа родительского отношения.

2) **SET NULL** – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

3) **SET DEFAULT** – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

4) **NONE** – вставить кортеж, не обращая внимания на нарушения ссылочной целостности

При обновлении кортежа в дочернем отношении возможны стратегии:

1) **RESTRICT** – не разрешать обновление, если внешний ключ в обновляемом кортеже становится не соответствующим ни одному значению потенциального ключа родительского отношения.

2) **SET NULL** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

3) **SET DEFAULT** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

4) **NONE** – обновить кортеж, не обращая внимания на нарушения ссылочной целостности.

6. СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ

6.1. Начало работы с ERwin

Запуск программы осуществляется через меню **Пуск>Все программы>CA>ERwin>ERwin Data Modeler r7.3>ERwin Data Modeler** или через ярлык на рабочем столе.

При запуске ERwin открывается главное окно программы CA ERwin Data Modeler (рис. 6.1).

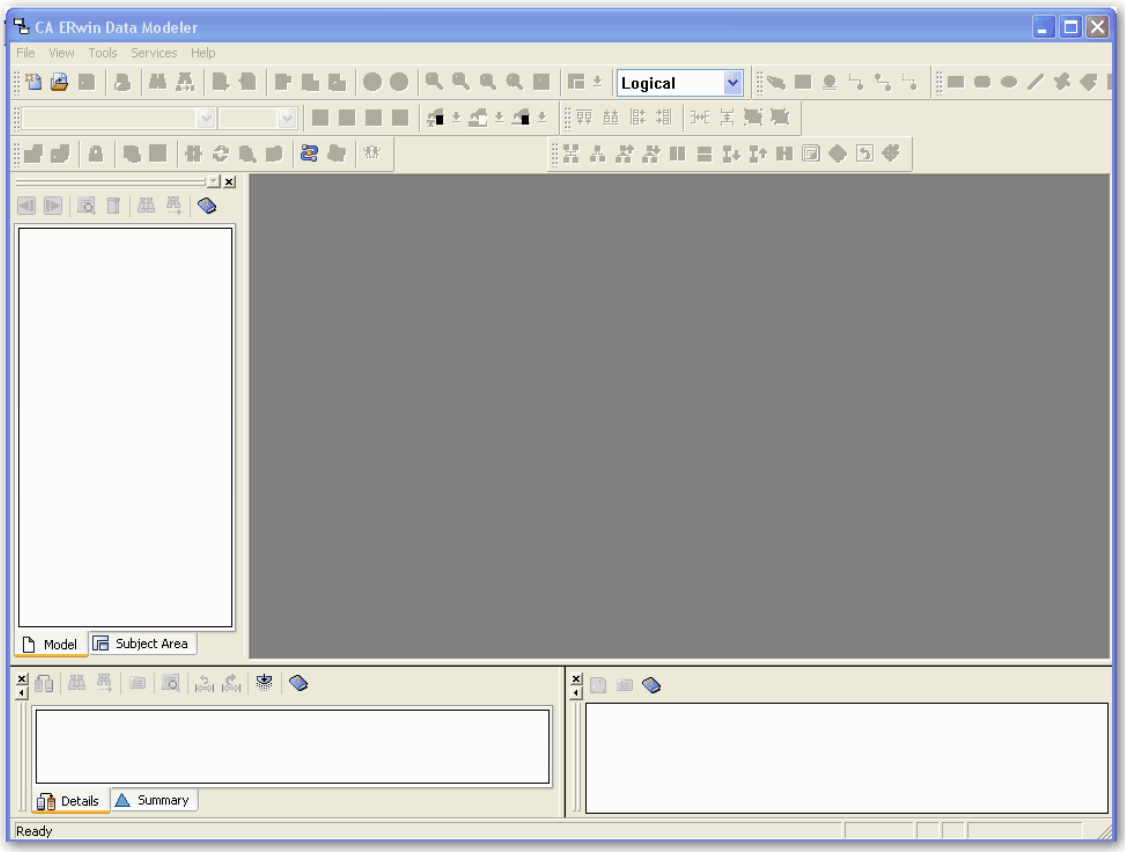

Рис. 6.1. Главное окно программы **CA ERwin Data Modeler**

Порядок построения модели данных в среде ERwin рассмотрим на примере автоматизированной информационной системы "Реализация средств вычислительной техники", предназначенной для учета продаж настольных компьютеров по заказам клиентов.

Создание новой модели начинается с выбора типа модели: логическая, физическая, логико-физическая. При проектировании базы данных целесообразнее выбирать логико-физическую модель.

Выбор типа новой модели осуществляется в диалоге **Create Model** (рис. 6.2), вызываемом через **File>New** или кнопку создания нового файла . При выборе физической или логико-физической модели в диалоге **Cre-**

ate Model необходимо также указать необходимую СУБД и номер ее версии (в полях **Database** и **Version**).

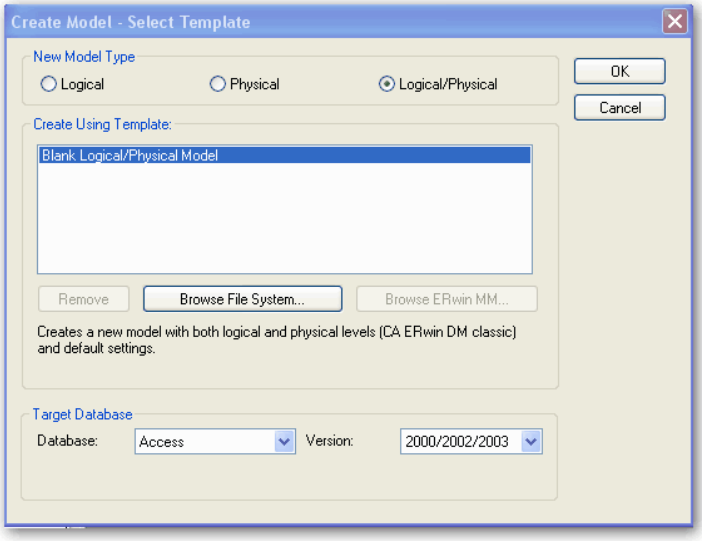

Рис. 6.2. Диалог **Create Model**

Данный диалог позволяет также открыть существующую модель данных указанного типа посредством активизации кнопки 

.

В соответствии с выбранным действием при активизации кнопки **OK** открывается окно для построения новой модели данных (рис. 6.3) или окно редактирования модели, созданной раннее.

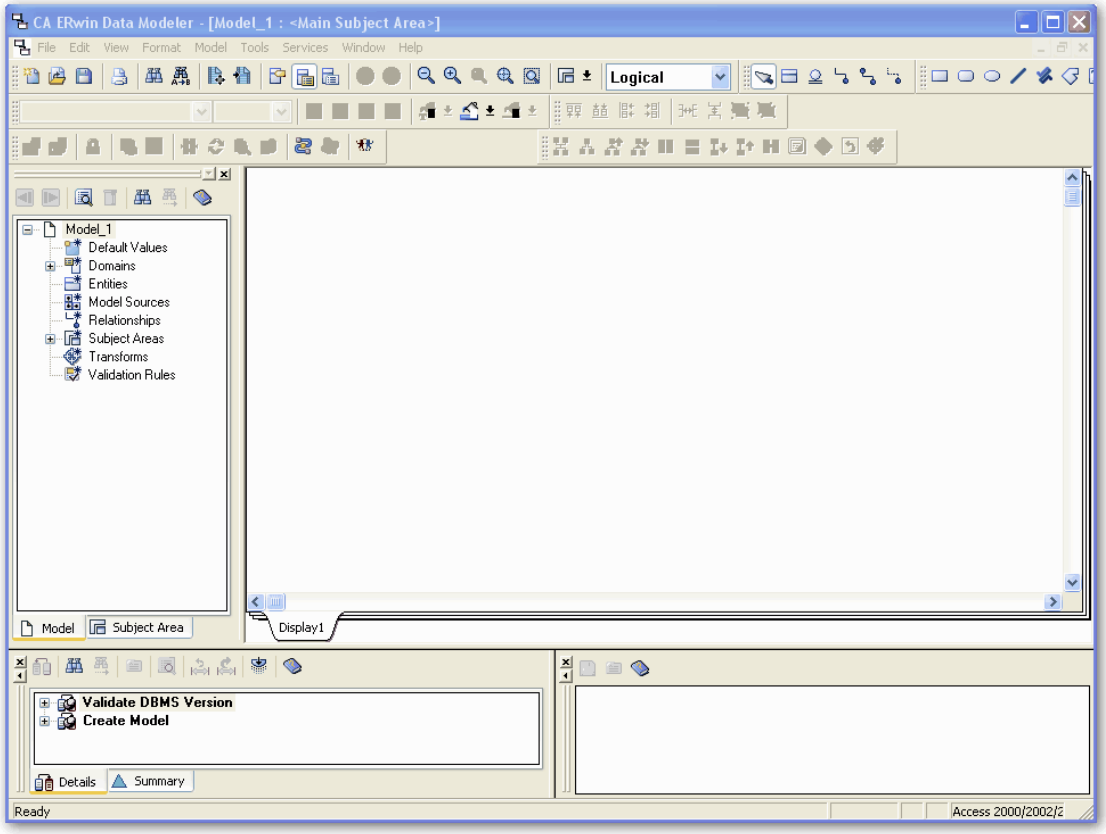

Рис. 6.3. Окно для построения новой модели данных

6.2. Подуровни логического уровня модели данных

Создание модели данных начинается с разработки логической модели, которая должна представлять состав сущностей предметной области с перечнем атрибутов и отношений между ними.

Если предварительно был выбран логико-физический уровень представления модели данных, то в *списке выбора для переключения между логической и физической моделью* (пункт **Logical**, расположенный на панели инструментов) автоматически будет выбрана логическая модель (рис. 6.4).

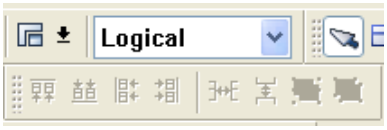

Рис. 6.4. Пункт **Logical** на панели инструментов

В противном случае тип модели необходимо выбрать из выпадающего меню данного пункта (рис. 6.5).

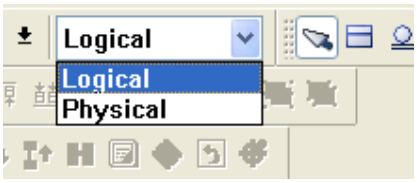

Рис. 6.5. Выпадающее меню пункта **Logical**

На логическом уровне ERwin поддерживает две нотации (IE и IDEF1X), на физическом уровне – три нотации (IE, IDEF1X и DM). По умолчанию используется нотация IDEF1X (*Integration DEFinition for information modeling*).

Смену нотации можно сделать на вкладке **Notation** диалога **Model Properties**, вызываемого через меню **Model>Model Properties** (рис. 6.6).

Для построения ER-модели используется панель инструментов, состав которой (условные графические обозначения) зависит от выбранного стандарта представления моделей и типа модели: логическая или физическая.

Поскольку стандарт IDEF1X используется по умолчанию, то на панели инструментов автоматически будут отображаться виды кнопок данного стандарта, описание которых приведено в табл. 6.1.

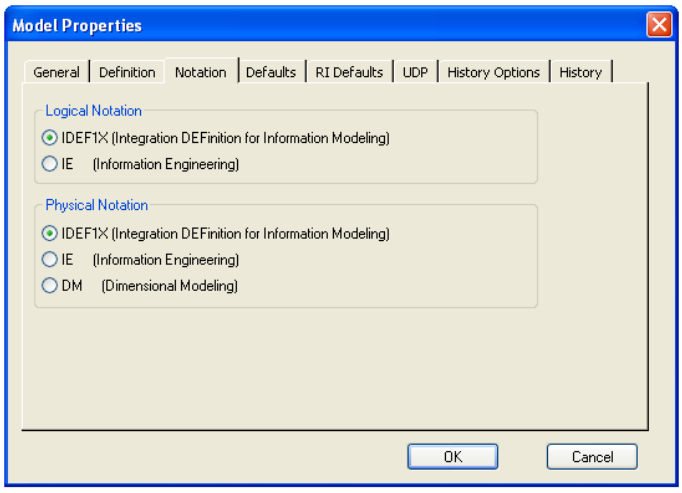

Рис. 6.6. Смена нотации на вкладке **Notation** диалога **Model Properties**

Таблица 6.1. Палитра инструментов логического уровня

ВИД КНОПКИ	НАЗНАЧЕНИЕ КНОПКИ
	Указатель (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели
	Создание новой сущности – для создания сущности необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание категории – для установления категориальной связи (специальный тип связи между сущностями) необходимо щелкнуть левой кнопкой мыши по кнопке категории, затем один раз щелкнуть по родительской сущности и затем один раз по сущности-потомку
	Создание идентифицирующей связи
	Создание связи «многие ко многим»
	Создание неидентифицирующей связи

В зависимости от глубины представления информации о данных различают 3 подуровня логического уровня модели данных:

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

Диаграмма сущность-связь включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области.

Модель данных, основанная на ключах – более подробное представление данных. Данная модель включает описание всех сущностей и первичных ключей, необходимых для подробного описания предметной области.

Полная атрибутивная модель данных – наиболее детальное представление структуры данных предметной области. Данная модель представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

6.3. Создание сущностей и атрибутов

Построение логической модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности и в конкретном атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которых должна сохраняться.

Например, сущность **Клиент** (но не **Клиенты**!) может иметь атрибуты **Номер клиента**, **Фамилия клиента**, **Адрес клиента** и **Телефон клиента**. На уровне физической модели данных данной сущности может соответствовать таблица (отношение) **Client** с колонками **Client_number**, **Client_name**, **Client_address** и **Client_telephone**.

Для *внесения сущности в модель* необходимо щелкнуть левой кнопкой мыши по кнопке сущности  , расположенной на панели инструментов, и затем щелкнуть один раз в поле проектирования модели на том месте, где необходимо расположить новую сущность. В результате в поле проектирования появится сущность с именем **E/1** по умолчанию (рис. 6.7).

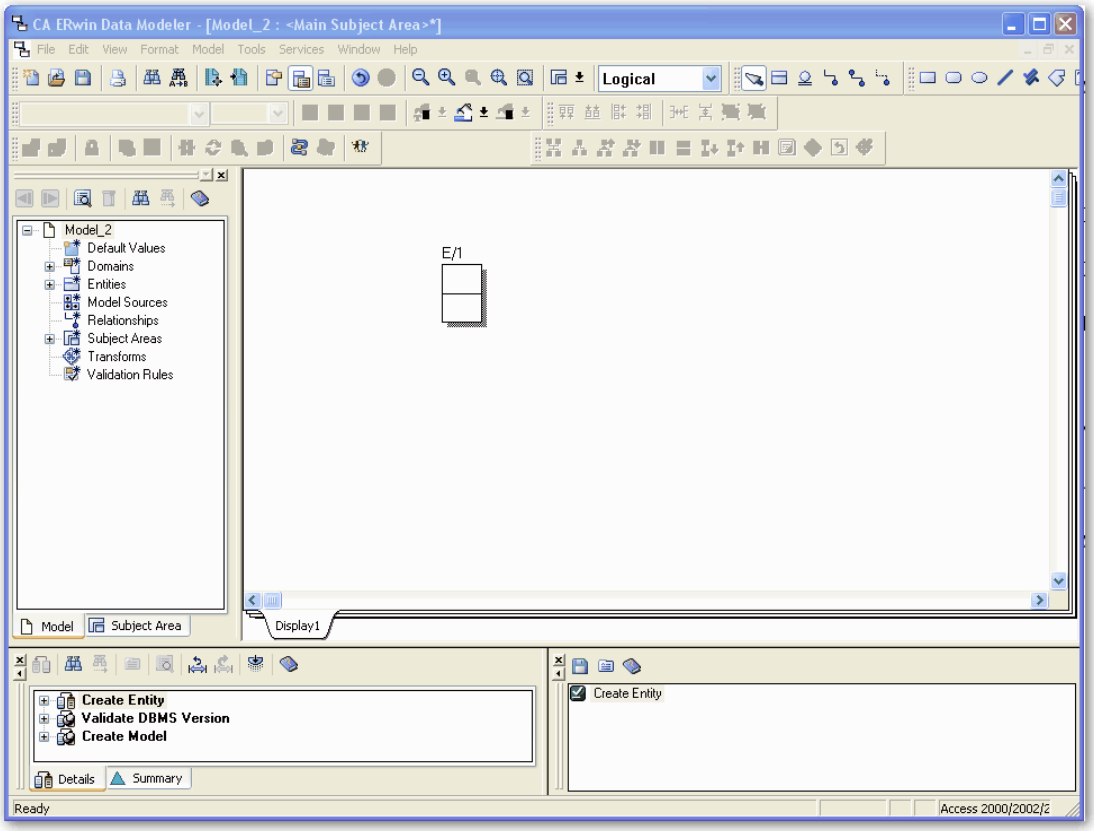

Рис. 6.7. Размещение в поле проектирования сущности с именем **E/1** по умолчанию

Определение имени сущности осуществляется через диалог **Entities**, который открывается через пункт **Entity Properties** (*свойства сущности*) контекстного меню по щелчку правой кнопкой мыши по выделенной сущности или пункта главного меню **Model/Entities**.

Диалог **Entities** (рис. 6.8) содержит вкладки, которые позволяют назначить и редактировать свойства сущности: имя (**Name**), определение (**Definition**), объем (**Volumetrics**), примечания (**Note, Note2, Note3**), определяемые пользователем свойства (**UDP** – *User definition properties*), иконки (**Icon**), историю (**History**).

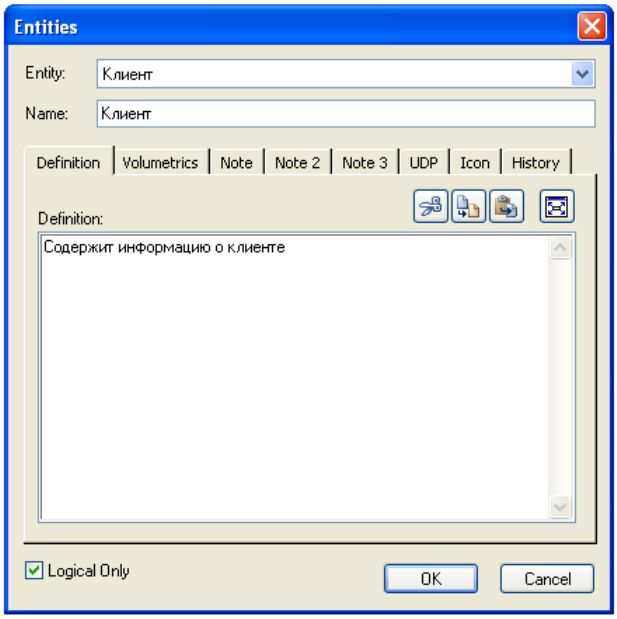

Рис. 6.8. Диалог **Entities**

Вкладка **Definition** используется для ввода определения сущности в виде ее текстового описания. На логическом уровне определения позволяют четче понять объект, а на физическом уровне – их можно экспортировать как часть схемы и использовать в реальной базе данных.

Вкладка **Volumetrics** позволяет указать предполагаемое начальное и максимальное количество экземпляров сущности и возможное их увеличение.

Вкладки **Note, Note2, Note3, UDP** служат для внесения дополнительных комментариев и определений к сущности, в частности:

– вкладка **Note** позволяет добавлять дополнительные замечания о сущности, которые не были отражены в определении (*Definition*), например, описать бизнес – правило или соглашение по организации диаграммы;

– вкладка **Note2** позволяет задокументировать некоторые возможные запросы, которые предполагается использовать по отношению к сущности в базе данных;

– вкладка **Note3** позволяет в произвольной форме вводить примеры данных для сущности;

– вкладка **UDP** позволяет пользователю определить свойства сущности и объем хранимых данных.

Вкладка **Icon** позволяет каждой сущности поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Вкладка **History** позволяет просмотреть историю всех изменений, связанных с сущностью и добавить комментарий к изменению в окне **Comment**.

По активизации кнопки **OK** на поле проектирования отобразится сущность с заданным именем и определенными свойствами (рис. 6.9).

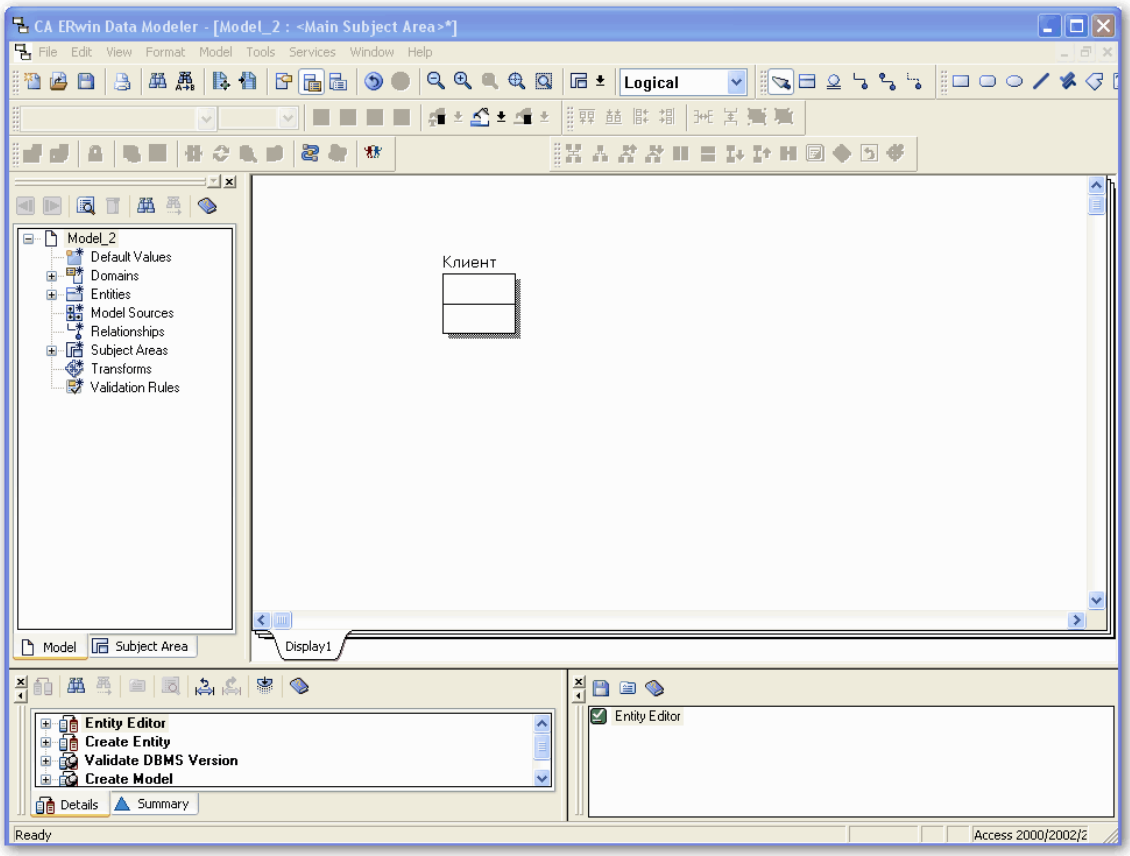

Рис. 6.9. Размещение в поле проектирования сущности с именем **Клиент**

При разработке ER-модели можно установить *параметры шрифтов и цветовое оформление сущностей* для облегчения обзора и понимания модели. При этом можно изменять установки параметров, назначенные по умолчанию, при добавлении нового объект на поле диаграммы, как для всех объектов диаграммы, так и для групп выделенных объектов или отдельных сущностей.

Параметры шрифта для названия сущности и цвет для заполнения поля блока редактируются с помощью пункта **Object Font & Color** контекстного меню (всплывающего меню по щелчку правой кнопкой мыши на поле выделенной сущности). Аналогичным образом изменяются параметры и для групп выделенных объектов диаграммы.

Определение атрибутов сущностей и их характеристик осуществляется в диалоговом окне **Attributes**, которое открывается с помощью пункта **Attributes**... контекстного меню и позволяет ввести данные об атрибутах выбранной сущности (рис. 6.10).

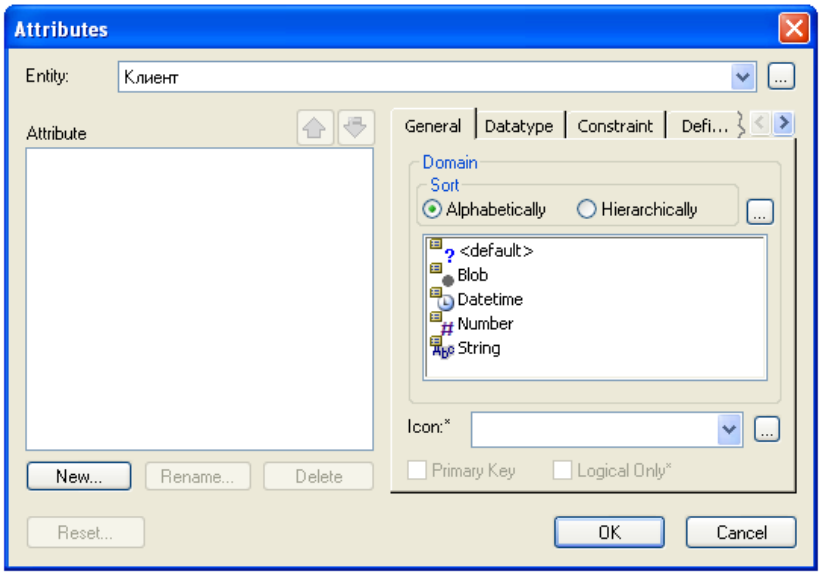

Рис. 6.10. Диалоговое окно **Attributes**

При активизации кнопки **New**... диалогового окна **Attributes** открывается диалог **New Attribute**, позволяющий добавить новый атрибут для выделенной сущности (рис. 6.11).

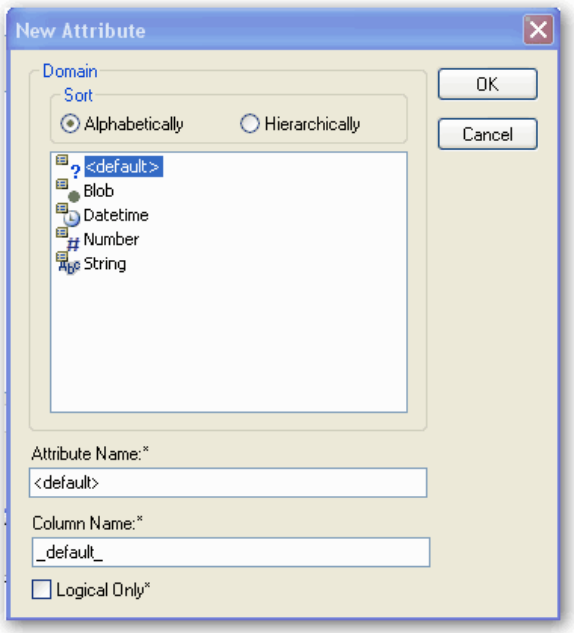

Рис. 6.11. Диалоговое окно **New Attribute**

В диалоговом окне **New Attribute** указывается имя нового атрибута (поле **Attribute Name**), имя, соответствующее ему в физической модели

колонки (поле **Column Name**) и домен (тип колонки на уровне физической модели).

Имена атрибутов можно задавать с помощью шрифтов типа "*Кириллица*" или "*Латиница*". При этом следует учитывать возможности СУБД, которые будут использоваться для сопровождения базы данных. Например, при использовании СУБД типа Access можно использовать имена сущностей и атрибутов для физической модели данных на русском языке (рис. 6.12).

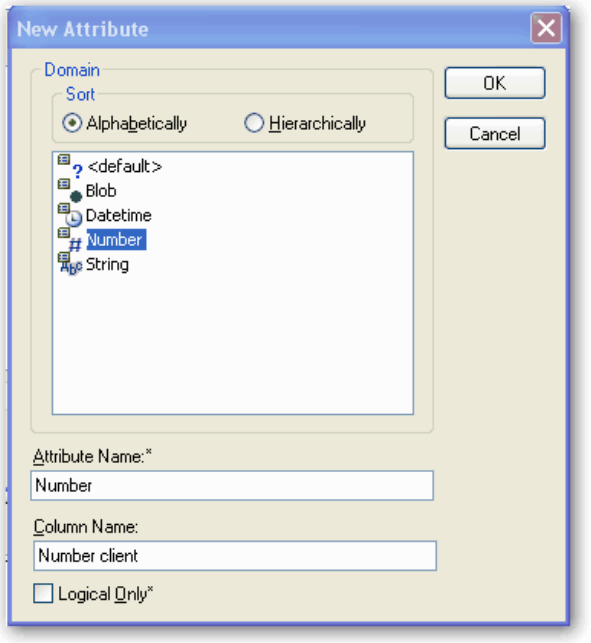

Рис. 6.12. Определение нового атрибута для сущности **Клиент**

Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение, что позволяет частично решить проблему нормализации данных на этапе определения атрибутов. Например, создание для сущности **Клиент** атрибута **Телефоны клиента** противоречит требованиям нормализации, т. к. атрибут должен быть атомарным, т.е. не содержать множественных значений.

Согласно синтаксису нотации IDEF1X имя атрибута должно быть уникально в рамках модели, поэтому новый атрибут, в случае совпадения его имени с уже существующим в рамках модели, должен быть переименован.

При активизации кнопки **OK** новый атрибут добавляется в список атрибутов выделенной сущности, отражающийся в поле **Attribute** диалогового окна **Attributes** (рис. 6.13).

Для определения первичного ключа выделенной сущности необходимо перейти на вкладку **General** и сделать пометку в окне выбора **Primary Key** (рис. 6.14).

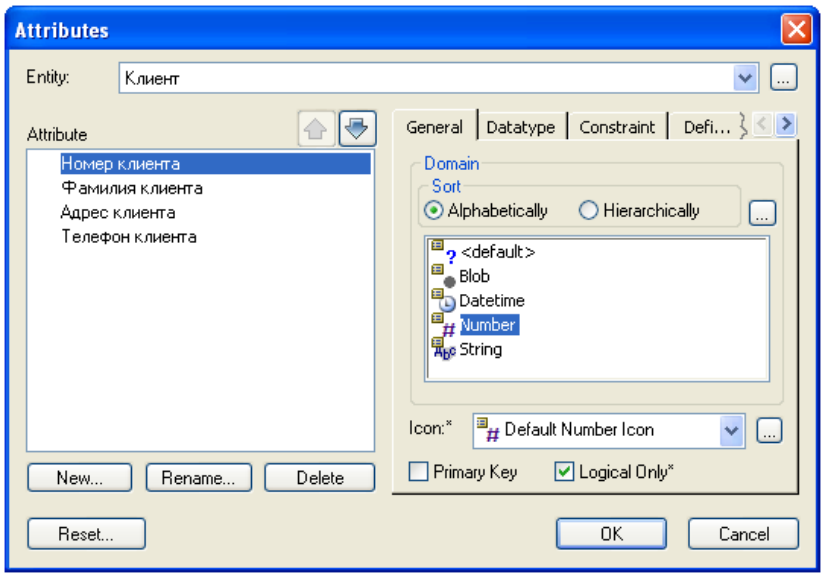

Рис. 6.13. Атрибуты сущности **Клиент**

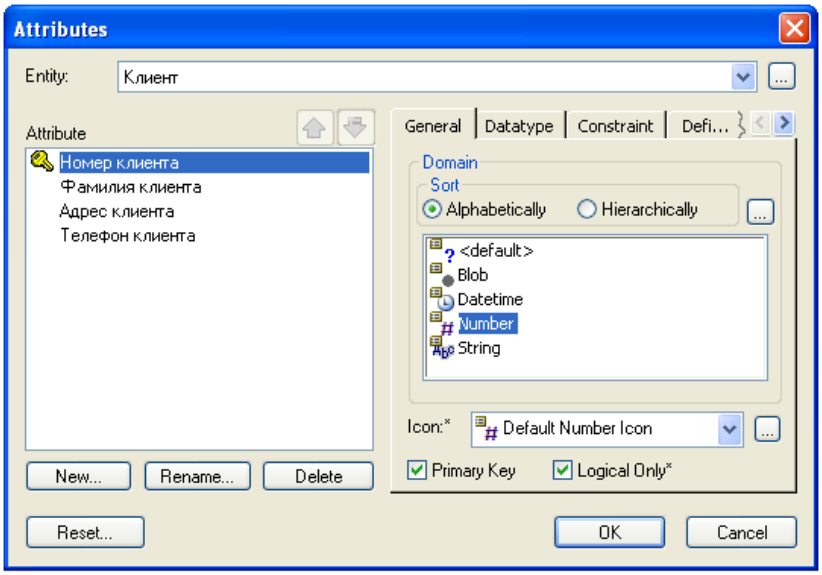

Рис. 6.14. Определение первичного ключа в окне выбора **Primary Key**

Вкладка **Datatype** позволяет с помощью выпадающего списка на странице уточнить тип данных для выделенного атрибута (рис. 6.15).

Вкладка **Constraint** позволяет задать ограничения для выделенного атрибута.

Вкладка **Definition** позволяет записать определения отдельных атрибутов.

Вкладка **Note** позволяет добавлять замечания об одном или нескольких атрибутах сущности, которые не вошли в определения.

Вкладка **UDP** предназначена для задания значений свойств, определяемых пользователем.

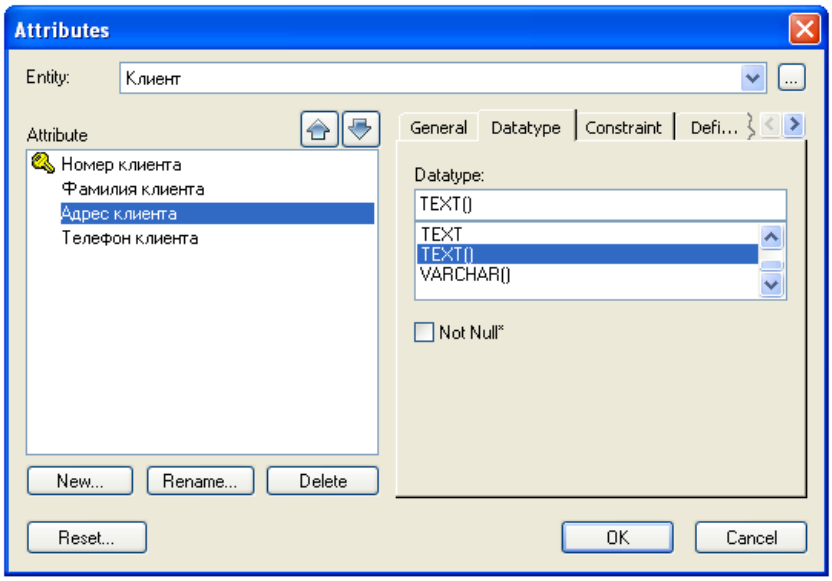

Рис. 6.15. Уточнение типа данных атрибута **Адрес клиента**

Вкладка **Key Group** (рис. 6.16) предназначена для отображения атрибутов, входящих в группу ключевых атрибутов сущности. В частности, на вкладке показано, что первичным ключом сущности **Клиент** является атрибут **Номер клиента**.

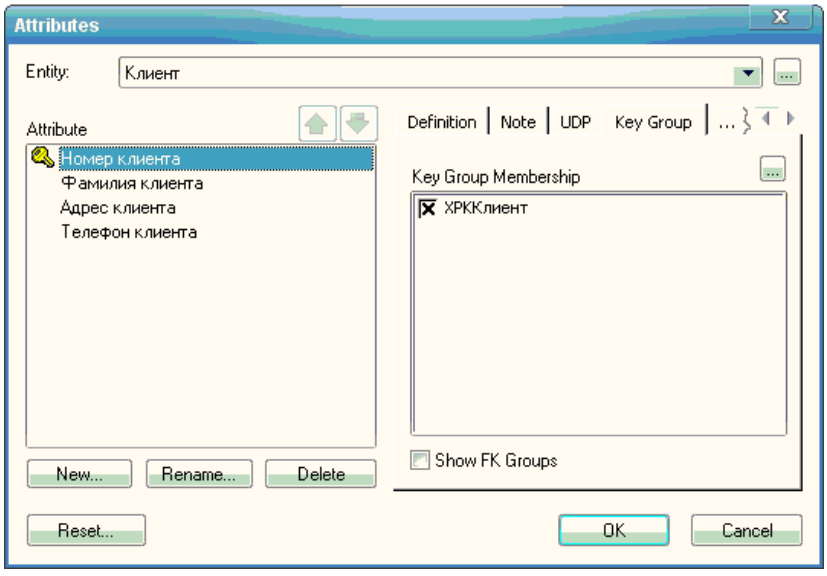

Рис. 6.16. Вкладка **Key Group** диалога **Attributes**

Сущность может иметь несколько возможных (*потенциальных*) *ключей*. Один потенциальный ключ объявляется первичным, а остальные могут быть объявлены *альтернативными ключами* (**Alternate Key**). ERwin по умолчанию на физическом уровне генерирует уникальные индексы по первичному и альтернативным ключам. ERwin позволяет также строить *неуникальные индексы* по атрибутам, называемым *инверсными входами* (**Inversion Entries**). Неуникальный индекс обеспечивает быстрый доступ не к

одной строке таблицы, а к нескольким, имеющим одинаковое значение инверсного входа. Это ускоряет доступ к данным.

Альтернативные ключи и инверсные входы создаются в диалоге **Key Groups**, который появляется после выбора соответствующего пункта контекстного меню (рис. 6.17).

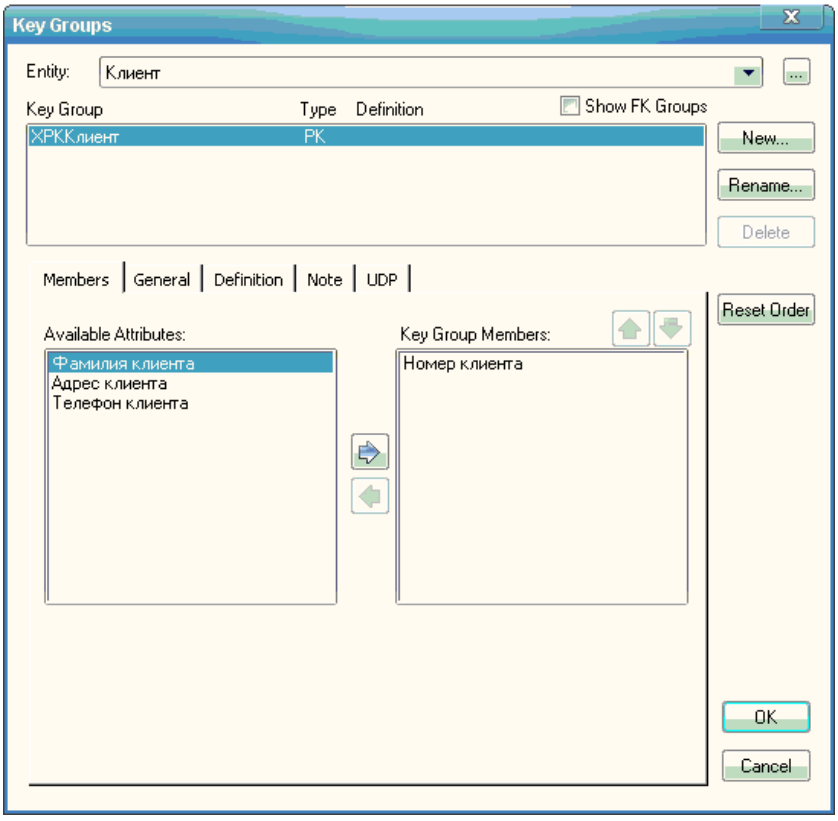

Рис. 6.17. Диалог **Key Groups**

Видно (рис. 6.17), что имеется первичный ключ (**PK**), включающий атрибут **Номер клиента**. Создадим инверсный вход по атрибуту **Фамилия клиента**. Для этого выделяем атрибут **Фамилия клиента** и нажимаем на кнопку **New**. Появляется диалог **New Key Group** (рис. 6.18).

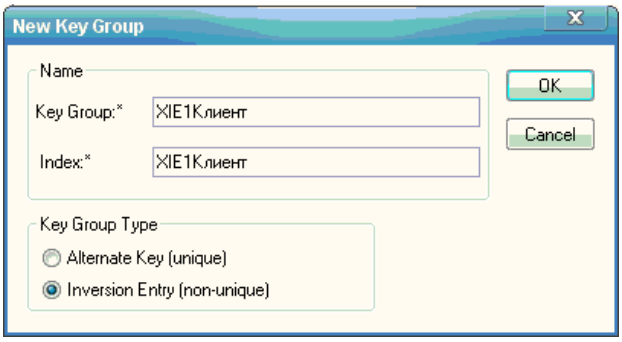

Рис. 6.18. Диалог **New Key Group**

В этом диалоге выбираем **Inversion Entry**. Имя ключевой группы (**Key Group**) и индекса (**Index**) присваиваются автоматически. Щелкаем по кнопке **OK** и возвращаемся в диалог **Key Groups** (рис. 6.19).

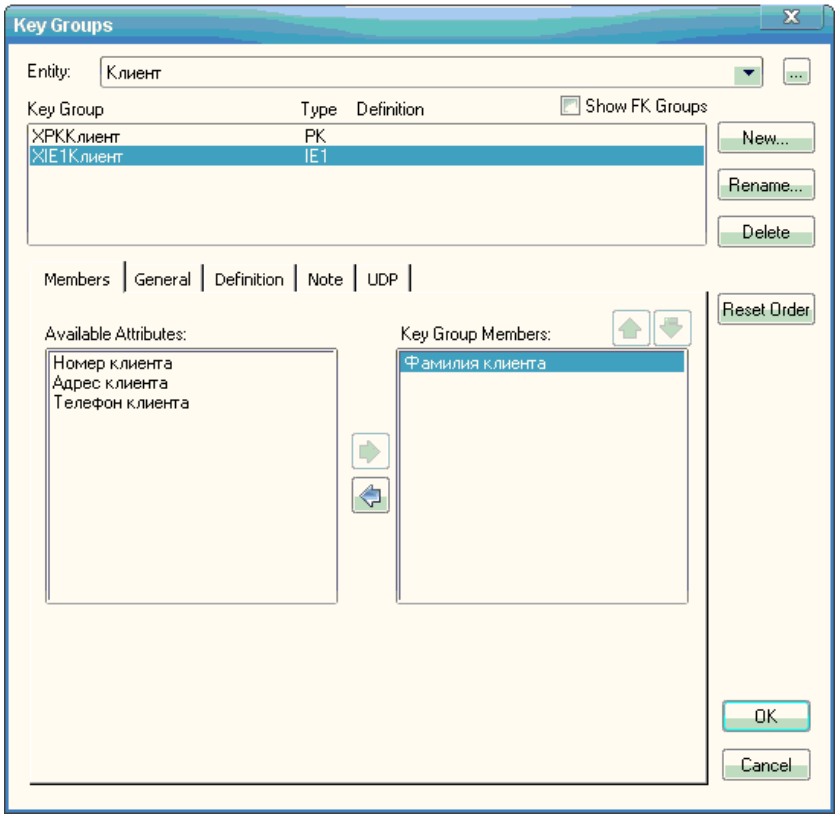

Рис. 6.19. Диалог **Key Groups** при построении инверсного входа

В верхнем окне **Key Group** вкладки выбираем ключевую группу **XIE1Клиент**, являющуюся инверсным входом (**IE**). В окне **Available Attributes** выбираем атрибут **Фамилия клиента** и с помощью кнопки  включаем его в состав ключевой группы.

Альтернативные ключи создаются аналогично.

Вкладка **History** диалога **Attributes** отображает историю создания и изменения свойств атрибута и позволяет добавить комментарии к изменению в окне **Comment** (рис. 6.20).

При нажатии на кнопку **OK** в поле проектирования модели отобразится сущность с атрибутами, определенными пользователем (рис. 6.21).

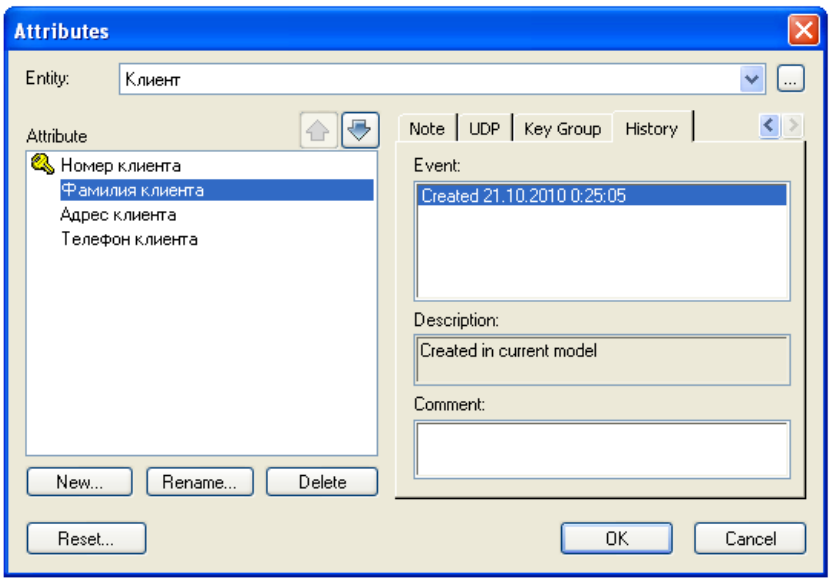

Рис. 6.20. История создания и изменения свойств атрибута **Фамилия клиента**

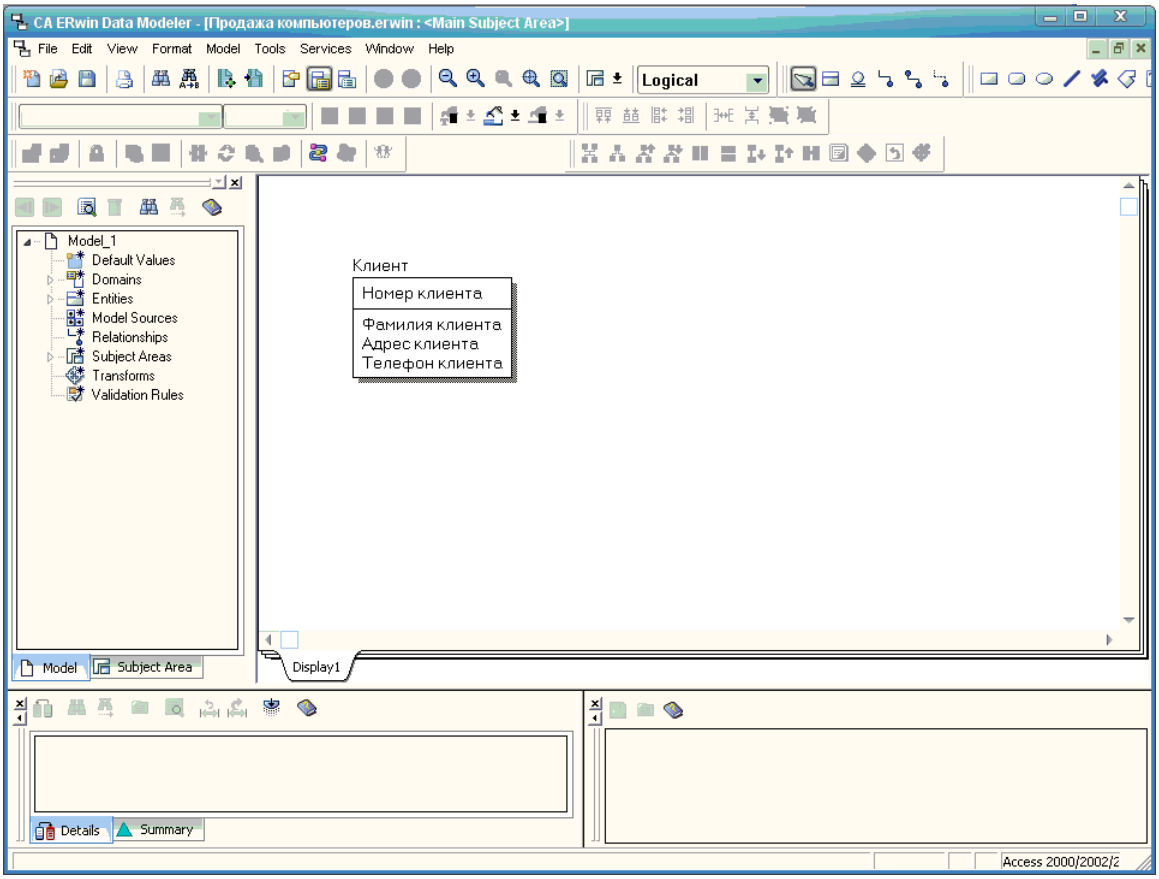

Рис. 6.21. Сущность **Клиент** с заданными атрибутами

Диалоговое окно **Attributes** позволяет пользователю редактировать имена атрибутов выделенной сущности и корректировать список атрибутов путем выполнения операции удаления ошибочно определенного атрибута.

Редактирование имени атрибута осуществляется в диалоге **Rename Attribute**, который открывается при активизации кнопки Rename... диалогового окна **Attributes** (рис. 6.22).

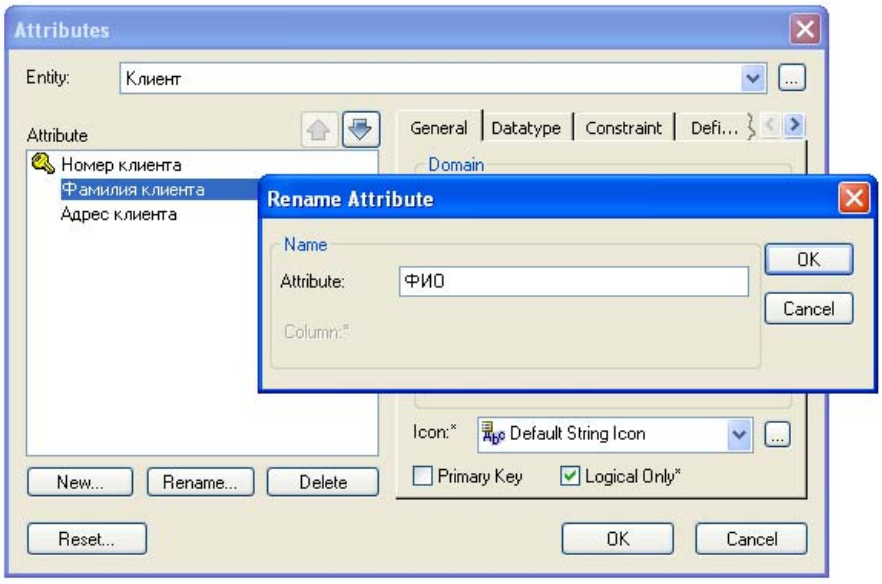

Рис. 6.22. Редактирование имени атрибута **Фамилия клиента**

При активизации кнопки Delete диалогового окна **Attributes** пользователю предоставляется возможность удаления выделенного атрибута из списка атрибутов. При этом необходимо внимательно относиться к данному действию, т.к. ERwin не требует подтверждения удаления атрибута.

Для описания свойств сущности часто приходится создавать *производные атрибуты*, т.е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить атрибут **Сумма заказа**, значение которого может быть вычислено из атрибутов **Количество заказанного товара** и **Цена за единицу товара**.

Многие производные атрибуты часто приводят к конфликтам. Например, значение производного атрибута **Возраст сотрудника** может быть вычислено из атрибута **Дата рождения сотрудника**. Возникновение конфликтной ситуации здесь возможно в случае несвоевременного обновления значения атрибута **Возраст сотрудника**, что приводит к противоречию значению атрибута **Дата рождения сотрудника**.

Производные атрибуты – ошибка нормализации. Нарушение синтаксиса нормализации допустимо в случаях необходимости повышения производительности системы. Например, хранение **Итоговой суммы заказа** позволяет повысить производительности системы при формировании документов на оплату заказа за счет непосредственного обращения к соответствующему атрибуту и исключения проведения предварительных вычислений, которые могут быть достаточно сложными, особенно при использовании спе-

циальных методик расчета с введением коэффициентов различного назначения.

ERwin позволяет *переносить атрибуты* внутри и между сущностями. Для этого необходимо щелкнуть правой кнопкой мыши по атрибуту. При этом указатель приобретает вид *кисти руки*, после чего можно перетащить выделенный атрибут на новое местоположение внутри сущности или в поле другой сущности диаграммы.

Для описания объектов предметной области по реализации настольных компьютеров по заказам клиентов выделены следующие сущности: **`Тип компьютера`**, **`Компьютер`**, **`Клиент`**, **`Заказ`**, **`Продажа`**, **`Менеджер`**, **`Отдел`** (рис. 6.23).

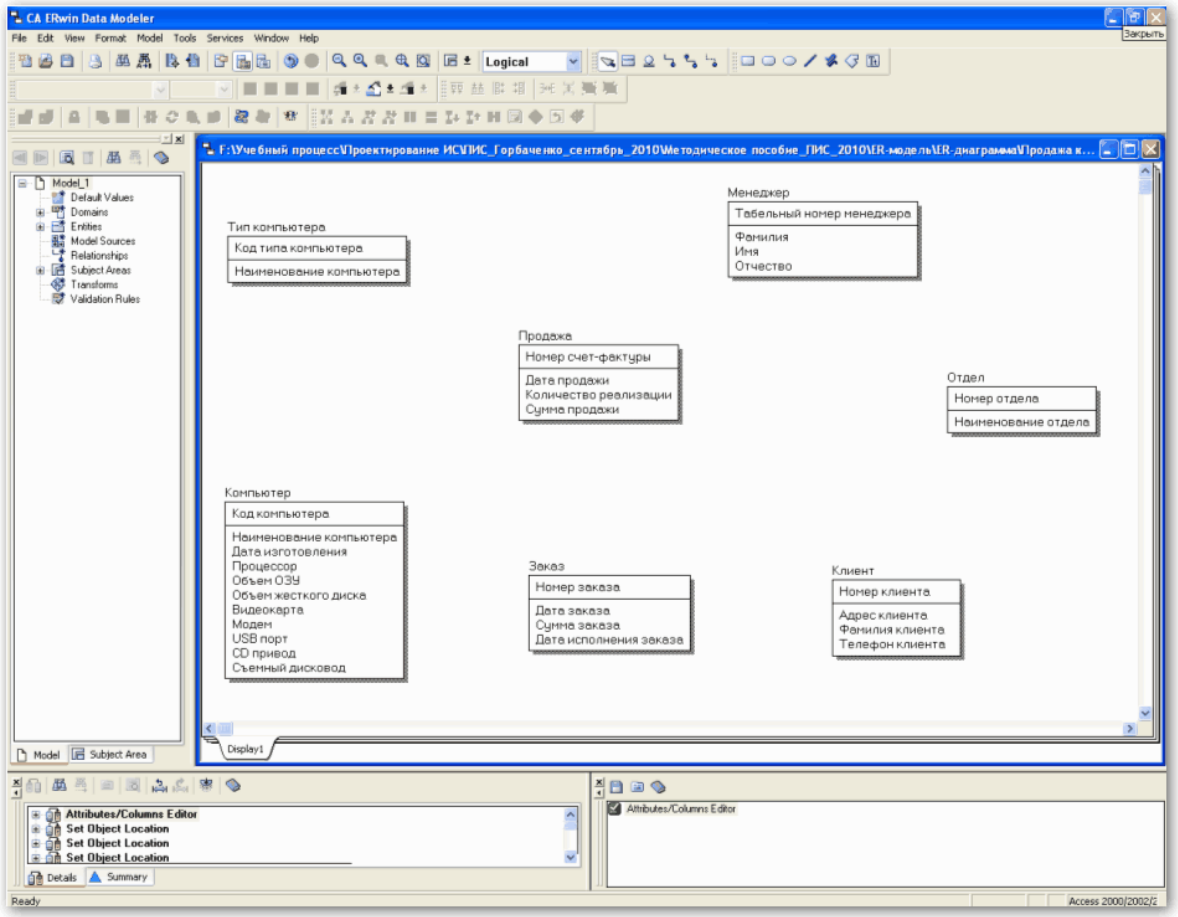

Рис. 6.23. Сущности, выделенные для описания объектов предметной области по реализации настольных компьютеров по заказам клиентов

6.4. Создание связей

Логическим соотношением между сущностями является *связь*. Каждому виду связи соответствует определенная кнопка, расположенная на палитре инструментов. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. Каждая связь должна именоваться

глаголом или глагольной фразой. Например, связь между сущностями **Компьютер**, **Продажа** и **Менеджер** показывает (рис. 6.24), какой компьютер продан и какой менеджер оформил сделку продажи компьютера:

– каждый **Компьютер** < продается > **Продажа**;

– каждая **Продажа** < оформляет > **Менеджер**.

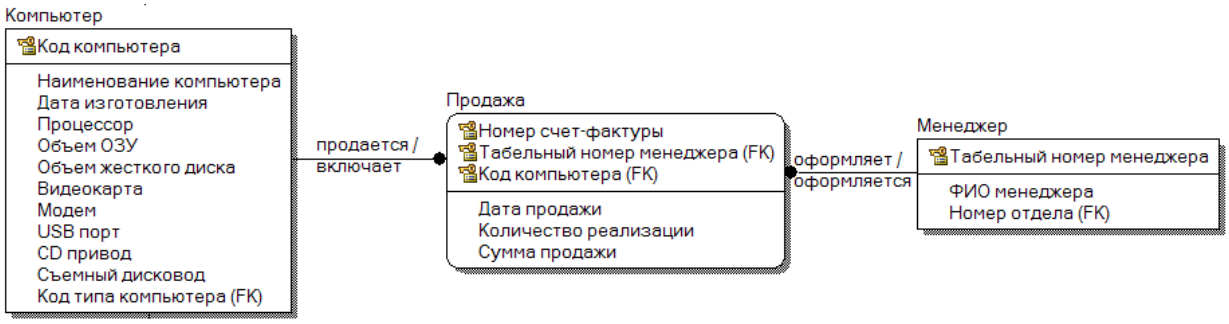

Рис. 6.24. Связь между сущностями **Компьютер**, **Продажа** и **Менеджер**

В нотации IDEF1X различают *зависимые* и *независимые* сущности. Тип сущности определяется ее связью с другими сущностями. *Идентифицирующая связь* устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями и показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи. При установлении идентифицирующей связи ERwin автоматически преобразует дочернюю сущность в зависимую сущность. Зависимая сущность на диаграмме изображается прямоугольником со скругленными углами, например, сущность **Продажа** (см. рис. 6.24). Экземпляр зависимой сущности определяется только через отношение к родительской сущности. Например, информация о продаже не может быть внесена и не имеет смысла без информации о проданном компьютере. При установлении *идентифицирующей связи* атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Операция дополнения атрибутов дочерней сущности при создании связи называется *миграцией атрибутов*. В дочерней сущности новые (мигрированные) атрибуты помечаются как внешний ключ (FK). В случае идентифицирующей связи при генерации схемы базы данных атрибутам внешнего ключа присваивается признак NOT NULL, что означает невозможность внесения записи в таблицу, соответствующей дочерней сущности, без идентификационной информации из таблицы, соответствующей родительской сущности. Например, невозможность внесения записи в таблицу продаж без информации об идентификационном номере проданного компьютера, определенного в таблице описания имеющихся в наличии компьютеров.

Неидентифицирующая связь показывается на диаграмме пунктирной линией с жирной точкой и служит для установления связи между независимыми сущностями. При установлении неидентифицирующей связи дочерняя

сущность остается независимой, а атрибуты первичного ключа мигрируют в состав неключевых атрибутов родительской сущности (рис. 6.25).

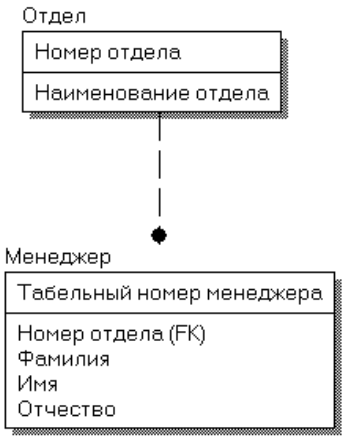

Рис. 6.25. Пример неидентифицирующей связи между независимыми сущностями **Менеджер** и **Отдел**

Для *создания новой связи* необходимо:

– установить курсор на кнопке, расположенной на палитре инструментов и соответствующей требуемому виду связи (см. табл. 6.1), и нажать левую кнопку мыши;

– щелкнуть левой кнопкой мыши сначала по родительской, а затем по дочерней сущности.

Изменить форму линии связи и ее местоположение между связанными сущностями можно путем захвата мышью выделенной линии связи и переноса ее с места на место, пока линия не примет необходимые местоположение и форму.

Редактирование свойств связи осуществляется в диалоговом окне **Relationship**, которое открывается через пункт **Relationship Properties** контекстного меню, активизируемого посредством нажатия правой кнопки мыши на выделенной связи (рис. 6.26).

Вкладка **General** диалогового окна **Relationship** позволяет задать мощность, имя и тип связи.

Мощность связи (*Cardinality*) служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней сущности. Различают 4 типа мощности связи (рис. 6.27):

– *общий случай* – не помечается каким-либо символом и соответствует ситуации, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности;

– *символом P помечается случай*, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности, т.е. исключено нулевое значение;

– *символом Z помечается случай*, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности, т.е. ис-

ключены множественные значения;

– *цифрой помечается случай* точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

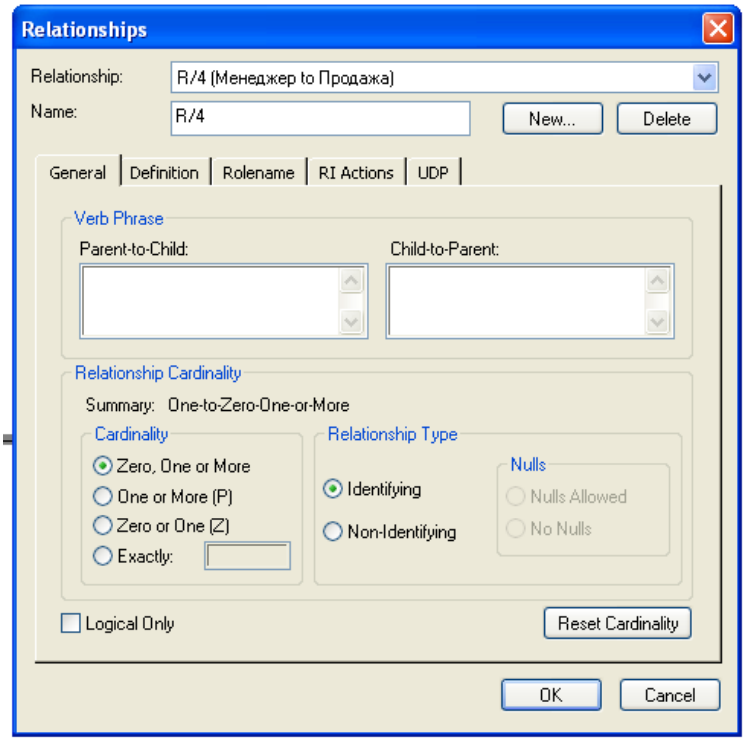

Рис. 6.26. Диалоговое окно **`Relationship`**

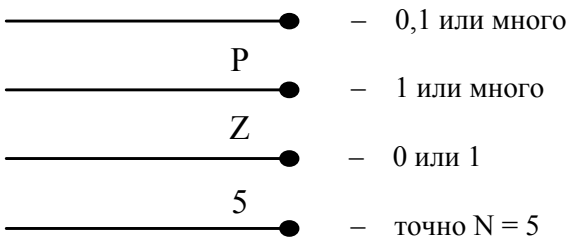

Рис. 6.27. Обозначение мощности связей

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения мощности связи следует включить опцию **`Cardinality`** пункта **`Relationship Display`** контекстного меню, которое появляется по щелчку правой кнопкой мыши по любому месту диаграммы, не занятому объектами модели.

Имя связи (*Verb Phrase*) – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи "один ко многим" (идентифицирующей или неидентифицирующей) достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (*Parent-to-Child*), а для связи "многие ко многим" необходимо указывать имя связи как от родительской к дочерней сущности (*Parent-to-Child*), так и от дочерней к роди-

тельской сущности (*Child-to-Parent*).

Пример определения мощности и имени связи между сущностями **Отдел** и **Менеджер** приведен на рис. 6.28.

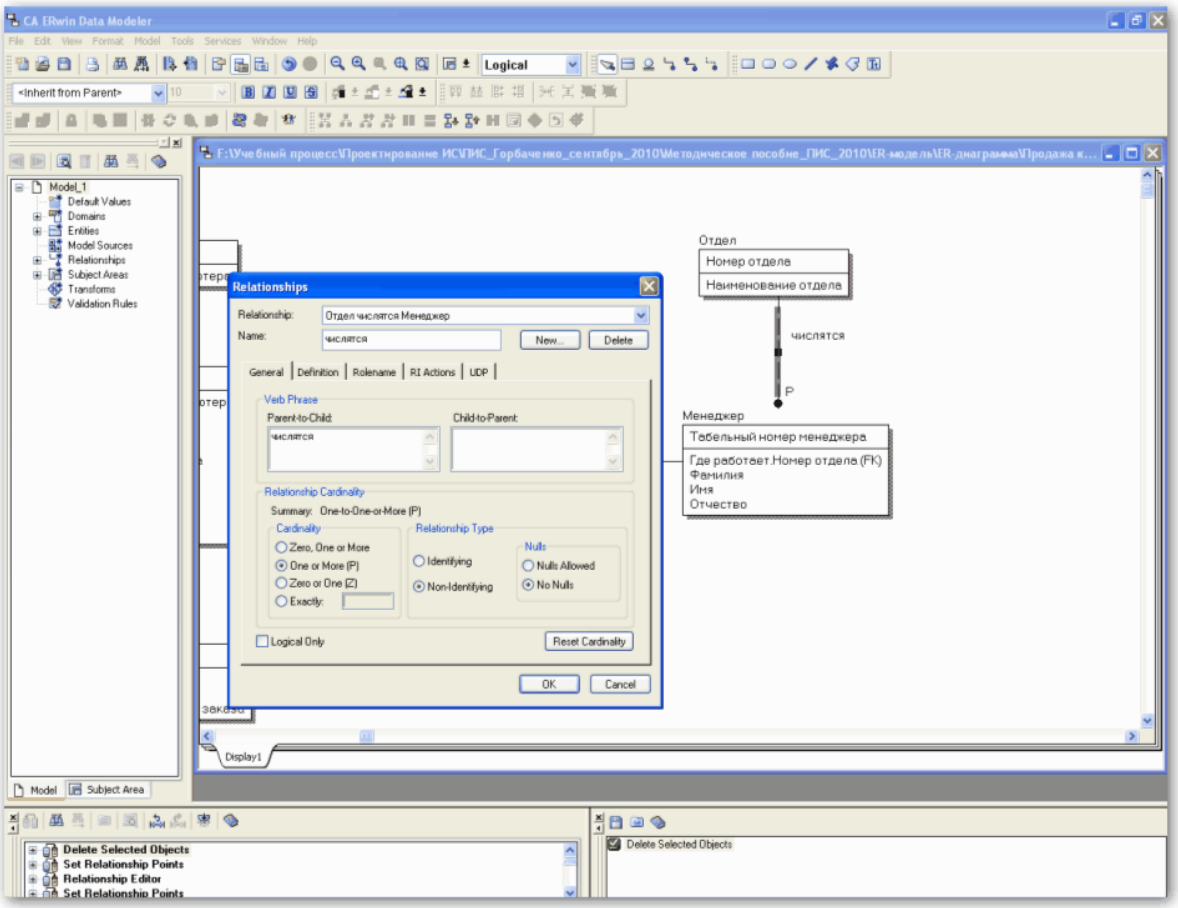

Рис. 6.28. Пример определения мощности и имени связи между сущностями **Отдел** и **Менеджер**

Тип связи (*Relationships Type*) позволяет обозначить идентифицирующую (*Identifying*) или неидентифицирующую (*Non-Identifying*) связь. Для неидентифицирующей связи необходимо указать обязательность связи (*Nulls Allowed или No Nulls*). В случае обязательной связи (*No Nulls*), несмотря на то, что внешний ключ не войдет в состав первичного ключа дочерней сущности, при генерации схемы базы данных атрибут внешнего ключа получит признак NOT NULL. В случае необязательной связи (*Nulls Allowed*) внешний ключ может принимать значение NULL. Это означает, что экземпляр дочерней сущности не будет связан ни с одним экземпляром родительской сущности. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности (рис. 6.29).

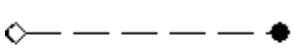

Рис. 6.29. Графическое представление необязательной неидентифицирующей связи

Например, при оформлении сделки по продаже компьютера информация о покупателе (клиенте) не всегда участвует в оформлении расходных документов (рис. 6.30).

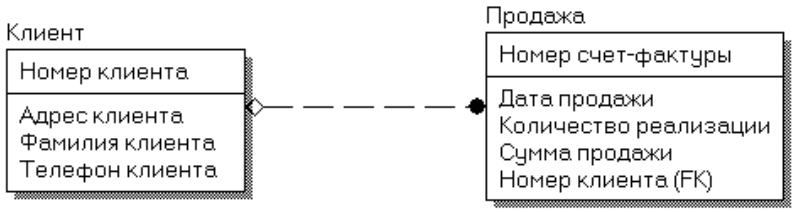

Рис. 6.30. Пример реализации необязательной неидентифицирующей связи

Вкладка **Definition** позволяет дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

Вкладка **Rolename** открывает окно диалога **Relationships**, которое позволяет задать имя роли атрибута внешнего ключа (рис. 6.31).

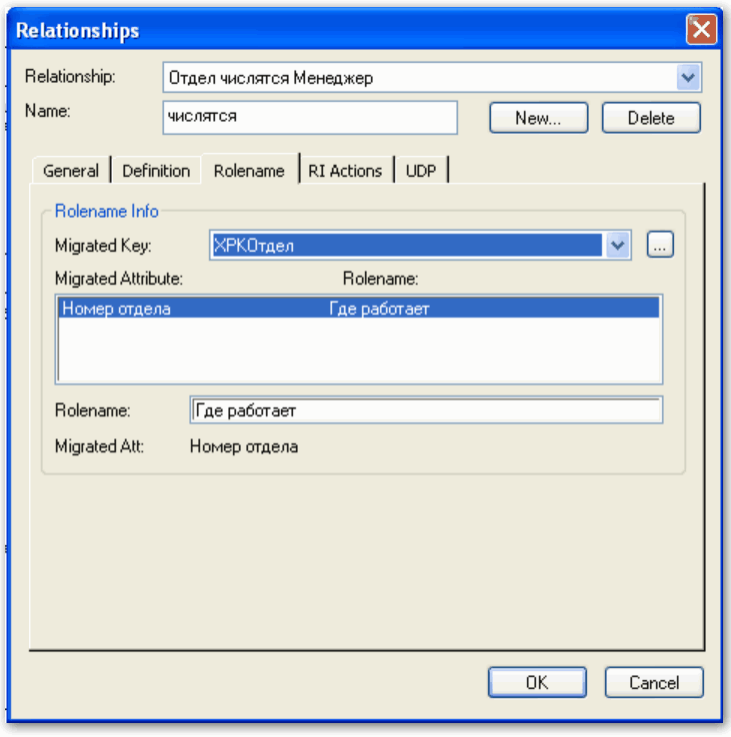

Рис. 6.31. Окно диалога **Relationships** вкладки **Rolename**

Имя роли (*Rolename*, функциональное имя) – это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности. Например, в сущности **Менеджер** внешний ключ **Номер отдела** имеет функциональное имя "*Где работает*", которое показывает, какую роль играет этот атрибут в данной сущности. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атри-

бута (как функционального имени, так и имени роли необходимо включить опцию **Rolename/Attribute** пункта **Entity Display** контекстного меню, которое появляется по щелчку правой кнопки мыши по любому месту диаграммы, не занятому объектами модели. В результате отобразится *Полное имя* атрибута, включающее функциональное имя и базовое имя, разделенные между собой точкой (рис. 6.32).

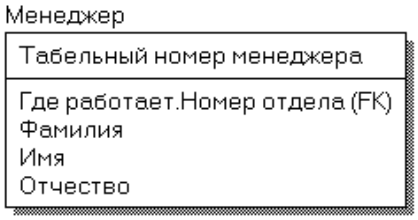

Рис. 6.32. Пример *Полного имени* внешнего атрибута **Номер отдела** сущности **Менеджер**

Обязательным является применение имен ролей в том случае, когда два или более атрибута одной сущности определены по одной и той же области, т. е. они имеют одинаковую область значений, но разный смысл. Другим примером обязательности присвоения имен ролей являются *рекурсивные связи* (иногда их называют "рыболовным крючком" – *fish hook*), когда одна и та же сущность является и родительской и дочерней одновременно. При задании рекурсивной связи атрибут мигрирует в качестве внешнего ключа в состав неключевых атрибутов той же сущности. Атрибут не может появиться дважды в одной сущности под одним именем, поэтому обязательно должен получить имя роли. Например, сущность **Менеджер** содержит атрибут первичного ключа **Табельный номер**. Информация о старшем менеджере (руководителе) содержится в той же сущности, поскольку старший менеджер работает в этой же организации. Чтобы сослаться на старшего менеджера, необходимо создать рекурсивную связь **руководит/подчиняется** и присвоить имени роли значение **старший** (рис. 6.33). Рекурсивная связь может быть только необязательной неидентифицирующей. В противном случае внешний ключ должен был бы войти в состав первичного ключа и получить при генерации схемы признак NOT NULL, что делает невозможным построение иерархии – у дерева подчиненности должен быть корень – менеджер, который никому не подчиняется в рамках данной организации.

Связь **руководит/подчиняется** позволяет хранить древовидную иерархию подчиненности сотрудников. Такой вид рекурсивной связи называется *иерархической рекурсией* (*hierarchical recursion*) и задает связь, когда руководитель (экземпляр родительской сущности) может иметь множество подчиненных (экземпляров дочерней сущности), но подчиненный имеет только одного руководителя.

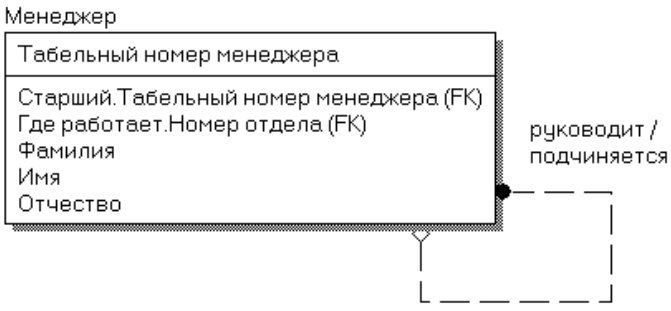

Рис. 6.33. Пример рекурсивной связи

Другим видом рекурсии является *сетевая рекурсия* (*network recursion*), когда руководитель может иметь множество подчиненных и, наоборот, подчиненный может иметь множество руководителей. Сетевая рекурсия задает паутину отношений между экземплярами родительской и дочерней сущностей. Это случай, когда сущность находится сама с собой в связи "многие-ко-многим". Для разрешения связи "многие-ко-многим" необходимо создать дополнительную сущность (подробнее связь "многие-ко-многим" рассматривается ниже).

Правила ссылочной целостности (*referential integrity* (*RI*)) – логические конструкции, которые выражают бизнес – правила использования данных и представляют собой правила вставки, замены и удаления. Определение правил ссылочной целостности осуществляется с помощью вкладки **RI Actions** контекстного меню **Relationships** (рис. 6.34). Заданные на вкладке **RI Actions** опции логической модели используются при генерации схемы базы данных для определения правил декларативной ссылочной целостности, которые предписываются для каждой связи, и триггеров, обеспечивающих ссылочную целостность.

На рис. 6.34 показан пример установленных по умолчанию правил ссылочной целостности для неидентифицирующей связи между сущностями **Отдел** и **Менеджер**. На удаление экземпляра родительской сущности (**Parent Delete**) устанавливается ограничение **RESTRICT**. Это означает, что экземпляр родительской сущности нельзя удалить, если с ней связан экземпляр дочерней. С экземпляром родительской сущности может быть не связан ни один экземпляр дочерней сущности. Поэтому при вставке экземпляра родительской сущности (**Parent Insert**) не проводится никакой проверки (ограничение **NONE**). При изменении ключевых атрибутов родительской сущности (**Parent Update**) нарушится связь с дочерними сущностями, так как внешний ключ дочерних сущностей не равен ключу родительской сущности. Поэтому по умолчанию такое изменение запрещено (ограничение **RESTRICT**). Отметим, что в данном случае можно применить ограничение **CASCADE**, приводящее к изменению внешних ключей дочерних сущностей при изменении ключа родительской сущности. При удалении экземпляра дочерней сущности (**Child Delete**) ссылочная целостность не нарушается, поэтому по умолчанию применено ограничение **NONE**. Если при

вставке экземпляра дочерней сущности (**Child Insert**) ее внешний ключ будет отличаться от ключа родительской сущности, то нарушится ссылочная целостность. Такая ситуация проверяется ограничением **RESTRICT**. Аналогично, ограничение **RESTRICT** не допускает присвоения внешнему ключу дочерней сущности значения, отличного от значения родительского ключа (**Child Update**).

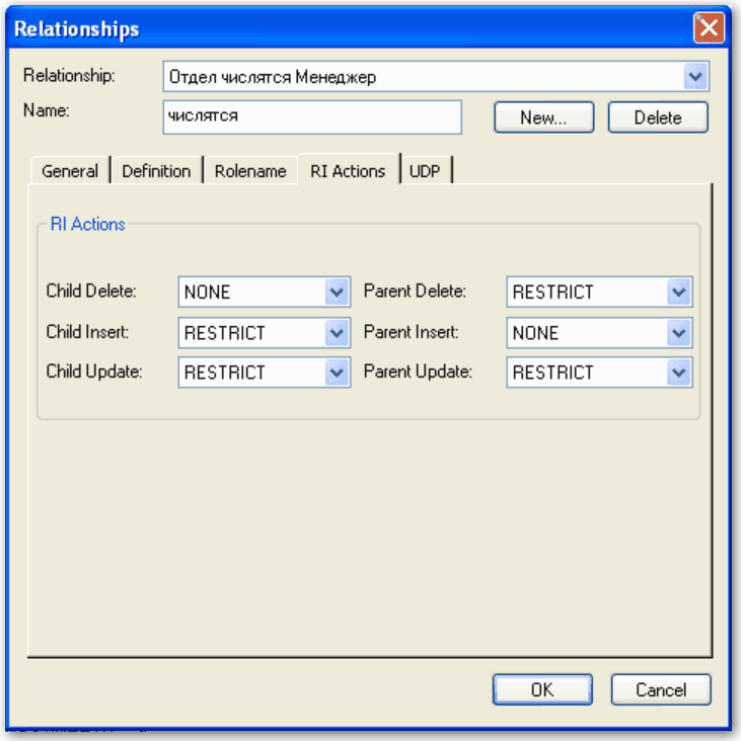

Рис. 6.34. Окно диалога **Relationships** вкладки **RI Actions**

ERwin автоматически присваивает каждой связи значение ссылочной целостности, устанавливаемой по умолчанию, прежде чем добавить ее в диаграмму. Режимы *RI*, присваиваемые ERwin по умолчанию, могут быть изменены на вкладке **RI Defaults** редактора **Model Properties** (меню **Model/Model Properties**). Правила ссылочной целостности можно изменить на вкладке **RI Actions** диалогового окна **Relationships.**

Правила ссылочной целостности реализуются специальными программами – *триггерами*, хранящимися в базе данных и выполняемыми всякий раз при выполнении команд вставки, замены или удаления (INSERT, UPDATE или DELETE). На логическом уровне ERwin создает шаблоны триггеров, построенные с использованием специальных макрокоманд. Шаблоны триггеров и расширенный код, зависящий от выбранной СУБД, можно увидеть на вкладке **Relationships Templates** контекстного меню **Relationships**. Подробнее о создании триггеров можно узнать в [2] и системе помощи ERwin.

Связь "многие-ко-многим" может быть создана только на уровне логической модели. Такая связь обозначается сплошной линией с двумя точками

на концах. Установление связи "многие-ко-многим" осуществляется при активизации соответствующей кнопки на палитре инструментов посредством щелчка левой кнопки мыши сначала по одной, а затем по другой сущности.

С целью облегчения чтения диаграммы связь "многие-ко-многим" должна иметь двунаправленное имя: от родительской к дочерней сущности (*Parent-to-Child*) и от дочерней к родительской сущности (*Child-to-Parent*).

Примером связи "многие-ко-многим" может служить связь между сущностями **Компьютер** и **Клиент** (рис. 6.35), т.к. один тот же вид компьютера может быть заказан многими клиентами (покупателями), а один и тот же клиент организации может заказать разные виды компьютеров:

- **Компьютер** < заказывается > **Клиент**;
- **Клиент** < заказывает > **Компьютер**.

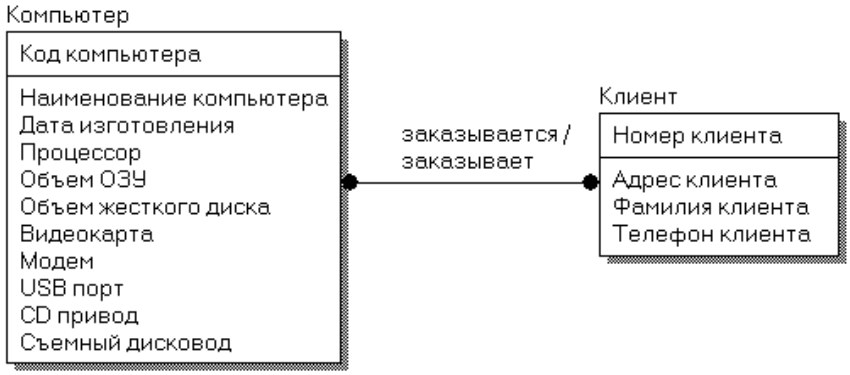

Рис. 6.35. Пример реализации связи "многие-ко-многим"

Нотация IDEF1X требует, чтобы на физическом уровне связь "многие-ко-многим" была преобразована, так как СУБД не поддерживают связь "многие- ко-многим". По умолчанию при переходе к физическому уровню ERwin автоматически не преобразует связь "многие-ко-многим". В этом случае на физическом уровне диаграмма выглядит так же, как и на логическом, однако при генерации схемы базы данных системы такая связь игнорируется. Поэтому уже на логическом уровне предпочтительно таких связей избегать. Кроме того, этого требует и синтаксис нормализации до третьей нормальной формы.

ERwin позволяет реализовать принудительное преобразование связи "многие-ко-многим", которое включает создание новой (связывающей) таблицы и двух новых связей "один-ко-многим" от старых таблиц к новой таблице. При этом имя новой таблице может присваиваться как автоматически, так и определяться пользователем. В случае автоматического присвоения имени связывающей таблице назначается имя, включающее имена обеих сущностей. Например, в случае автоматического разрыва связи "многие-ко-многим" между сущностями **Компьютер** и **Клиент** связывающая таблица получит имя **КомпьютерКлиент**.

Для осуществления принудительного преобразования связи "многие-

ко-многим" необходимо выбрать пункт контекстного меню **Create Association Entity**. В результате откроется диалог **Many-To-Many Transform Wizard** (рис. 6.36).

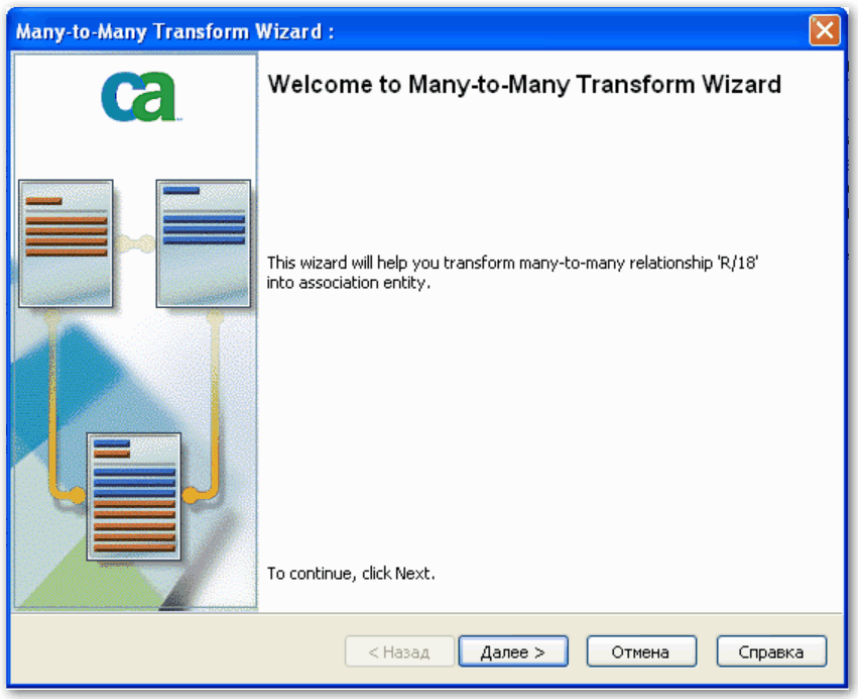

Рис. 6.36. Окно диалога **Many-To-Many Transform Wizard**

Диалог **Many-To-Many Transform Wizard** предлагает выполнить четыре шага для преобразования связи. Для перехода к следующему шагу необходимо щелкнуть по кнопке **Далее**. На втором и третьем шаге следует задать имя вновь создаваемой таблицы и имя преобразования. По окончании выполнения действий диалога **Many-To-Many Transform Wizard** на диаграмме будет добавлена новая таблица с заданным пользователем именем, а связь "многие-ко-многим" заменится идентифицирующими связями "один-ко-многим" от старых таблиц к новой таблице. При этом новые связи автоматически получат соответствующие имена от связи "многие-ко-многим".

Пример результата принудительного преобразования связи "многие-ко-многим" между сущностями **Компьютер** и **Клиент**, реализованного с помощью диалога **Many-To-Many Transform Wizard**, приведен на рис. 6.37.

Принудительного решения проблемы связи "многие-ко-многим" не всегда оказывается достаточно. Часто для описания сущности согласно бизнес – логике требуется добавление дополнительных атрибутов, позволяющих идентифицировать новую сущность. Например, для описания сущности **Заказ** необходимо добавить такие атрибуты, как **Дата заказа**, **Дата исполнения заказа** и **Сумма заказа**.

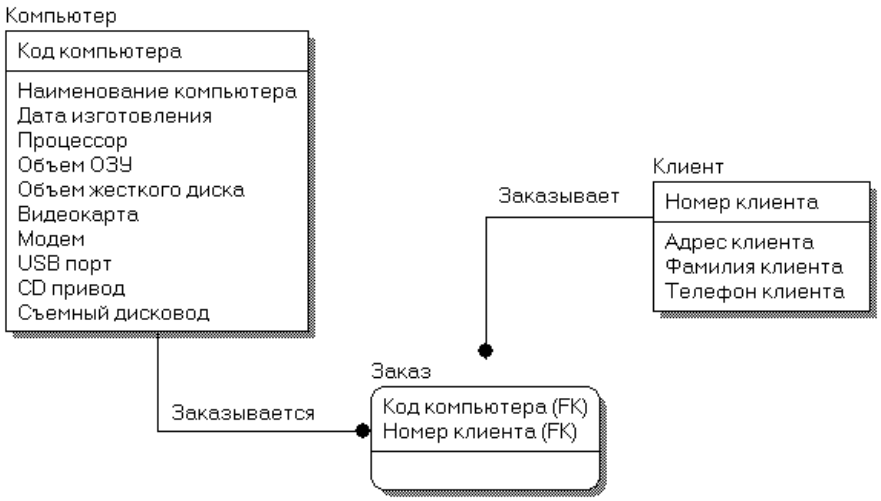

Рис. 6.37. Пример результата принудительного преобразования связи "многие ко многим"

Особым типом объединения сущностей является *иерархия наследования* (*иерархия категорий* или *категориальная связь*), согласно которой разделяются их общие характеристики. Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, или когда сущности имеют общие по смыслу связи, или когда это диктуется бизнес – правилами. Чтобы представить информацию, общую для всех типов экземпляров сущности, из их общих свойств можно сформировать обобщенную сущность (родовой предок). При этом специфическая для каждого типа экземпляра сущности информация будет располагаться в категориальных сущностях (потомках).

Например, в организации работают менеджеры и консультанты, которые имеют сходную по смыслу связь "**работает в**" с сущностью **Отдел**. Чтобы представить информацию, общую для всех типов служащих, из их общих свойств можно сформировать обобщенную сущность **Сотрудник**. При этом категориальными сущностями будут являться **Менеджер** и **Консультант** (рис. 6.38).

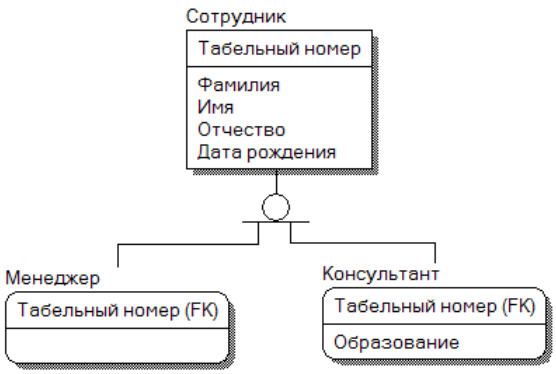

Рис. 6.38. Пример иерархии наследования (категориальной связи)

Для создания категориальной связи следует:

– установить курсор на кнопке , расположенной на палитре инструментов и нажать левую кнопку мыши;

– щелкнуть сначала по родовому предку, а затем – по потомку;

– установить вторую связь в иерархии категории, выбрав кнопку и щелкнув сначала по символу категории на диаграмме, затем – по второму потомку.

Редактирование категорий осуществляется в диалоговом окне **Subtype Properties**, которое открывается при выборе пункта **Subtype Properties**... контекстного меню, отображаемого по щелчку правой кнопкой мыши по символу категории (рис. 6.39). Поля диалогового окна позволяют указать дискриминатор – атрибут категории (список **Discriminator**) и тип категории – полная/неполная (радиокнопки **Complete/Incomplete**).

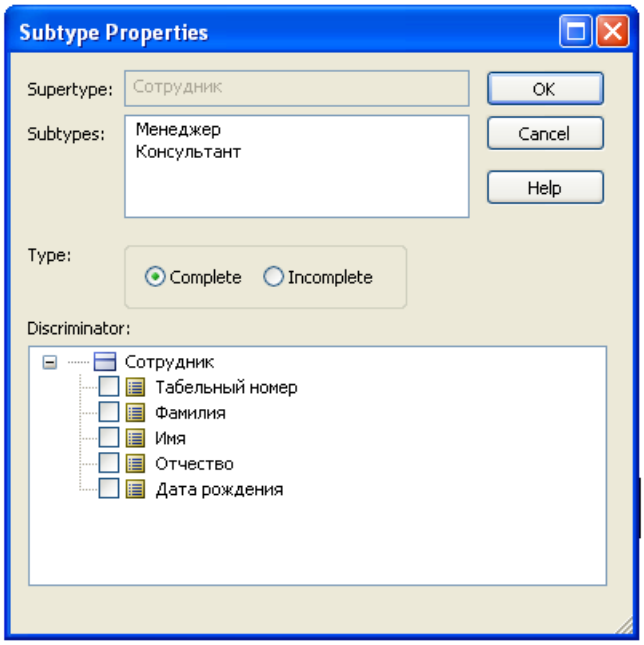

Рис. 6.39. Диалоговое окно **Subtype Properties**

Дискриминатор категории – это атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке. Например, сотрудник обязательно является либо менеджером, либо консультантом. Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. Например, сотрудник может быть не только менеджером или консультантом, но и совместителем. В случае неполной категории сущность **Совместитель** еще не внесена в иерархию наследования (рис. 6.38). На рис. 6.40 представлен пример полной категории. Возможна также комбина-

ция полной и неполной категорий.

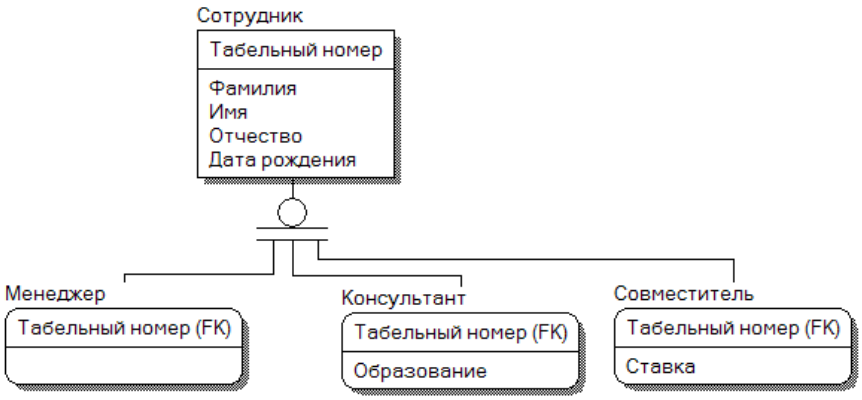

Рис. 6.40. Пример полной категории

Результат разработки логической модели данных системы "Реализация средств вычислительной техники", предназначенной для учета продаж настольных компьютеров по заказам клиентов приведен на рис. 6.40.

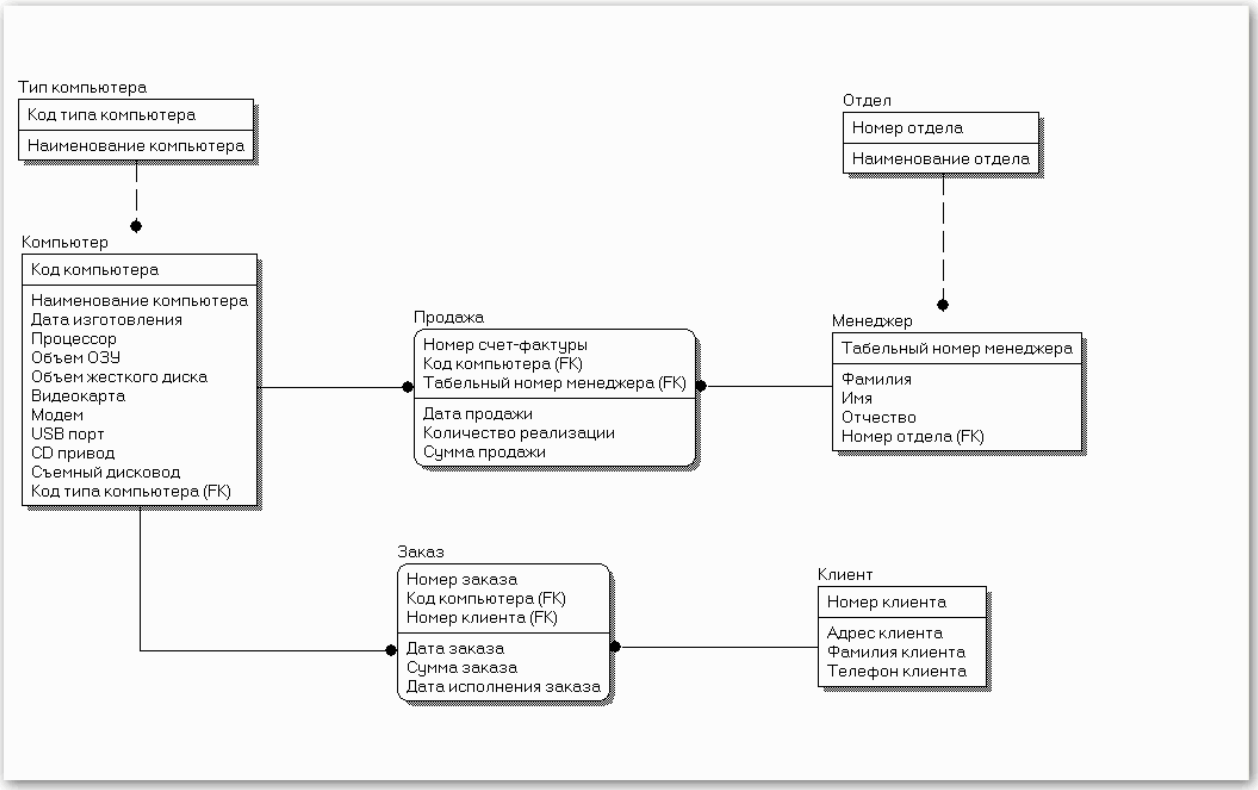

Рис. 6.40. Логическая модель данных системы
"Реализация средств вычислительной техники"

Логический уровень модели данных является универсальным и никак не связан с конкретной реализацией СУБД. Одному и тому же логическому уровню модели могут соответствовать несколько разных физических уровней различных моделей, где описывается вся информация о конкретных физиче-

ских объектах: таблицах, колонках, индексах, процедурах и т.д. При этом физические модели данных могут соответствовать СУБД разных производителей, например, Oracle, MS SQL Server, MySQL, SYBASE, Informix, MS Access, Progress и т.д. Возможность синхронизации логического уровня модели с несколькими моделями, имеющими разный физический уровень, позволяет повысить эффективность разработки гетерогенных информационных систем.

Созданная логическая модель данных системы является основанием для создания физической модели данных под выбранную СУБД.

6.5. Задание для самостоятельной работы

Разработать логическую модель данных системы «Поликлиника», обладающей следующими возможностями:

– хранение данных о пациентах, врачах, приемах, диагнозах, лечениях, лекарствах;

– корректировка данных о пациентах и врачах;

– поиск данных о пациентах и врачах по фамилии или адресу;

– добавление новых лекарств с описанием их свойств;

– поиск справочных данных по лекарствам;

– регистрация приемов на дому или в поликлинике.

– формирование статистической информации за отчетный период.

При создании логической модели данных системы в среде ERwin выбрать логико – физический тип модели.

7. СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ

7.1. Выбор физического уровня представления модели данных

Создание физической модели является вторым шагом построения модели данных системы. ERwin позволяет построить физическую модель как независимую, так и зависимую от логического уровня представления модели данных.

Построение физической модели, независимой от логического уровня представления модели данных, начинается с активизации диалога **Create Model** (рис. 6.2) через пункты основного меню **File>New** или кнопку нового файла  , при этом в окне выбора нового типа модели **New Model Type** (рис. 7.1) следует выбрать физический уровень представления (радиокнопка **Physical**).

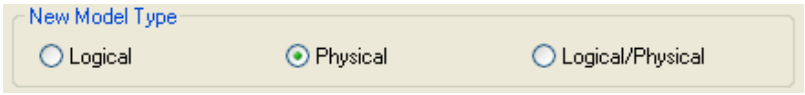

Рис. 7.1. Окно выбора нового типа модели **New Model Type**

Физический уровень представления модели зависит от конкретной реализации СУБД, поэтому необходимо предварительно осуществить ее выбор. CA ERwin Data Modeler 7.3 поддерживает 17 наиболее распространенных СУБД. Выбор СУБД осуществляется в окне **Target Database** диалога **Create Model**. Для выбора СУБД необходимо открыть выпадающий список с перечнем поддерживаемых СУБД и щелкнуть по соответствующему имени. При этом в поле **Version** отобразится версия выбранной СУБД (рис. 7.2).

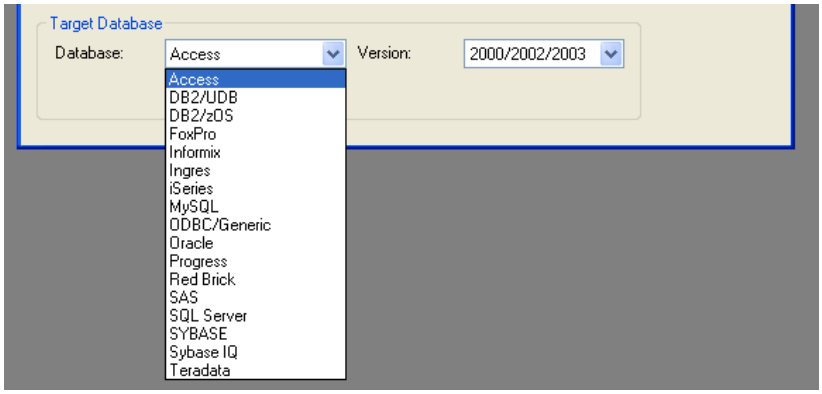

Рис. 7.2. Окно выбора СУБД **Target Database**

При активизации кнопки **OK** диалога **Create Model** откроется поле для построения физической модели данных.

При выборе логико-физической модели для проектирования базы данных преобразование логической модели в физическую модель осуществляется автоматически по переходу к пункту **`Physical`** в *списке выбора для переключения между логической и физической моделью*, расположенном на панели инструментов (рис. 7.3).

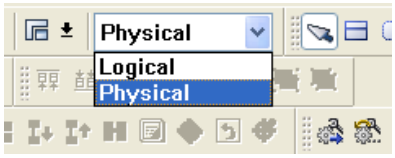

Рис. 7.3. Переход к физической модели через пункт **`Physical`** панели инструментов

В случае автоматического перехода к физической модели ERwin генерирует имена таблиц и колонок по умолчанию на основе имен соответствующих сущностей и атрибутов логической модели, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые выбранной СУБД. При этом правила ссылочной целостности, принятые на логическом уровне модели данных системы, также принимаются по умолчанию, т.е. сохраняются (рис. 7.4).

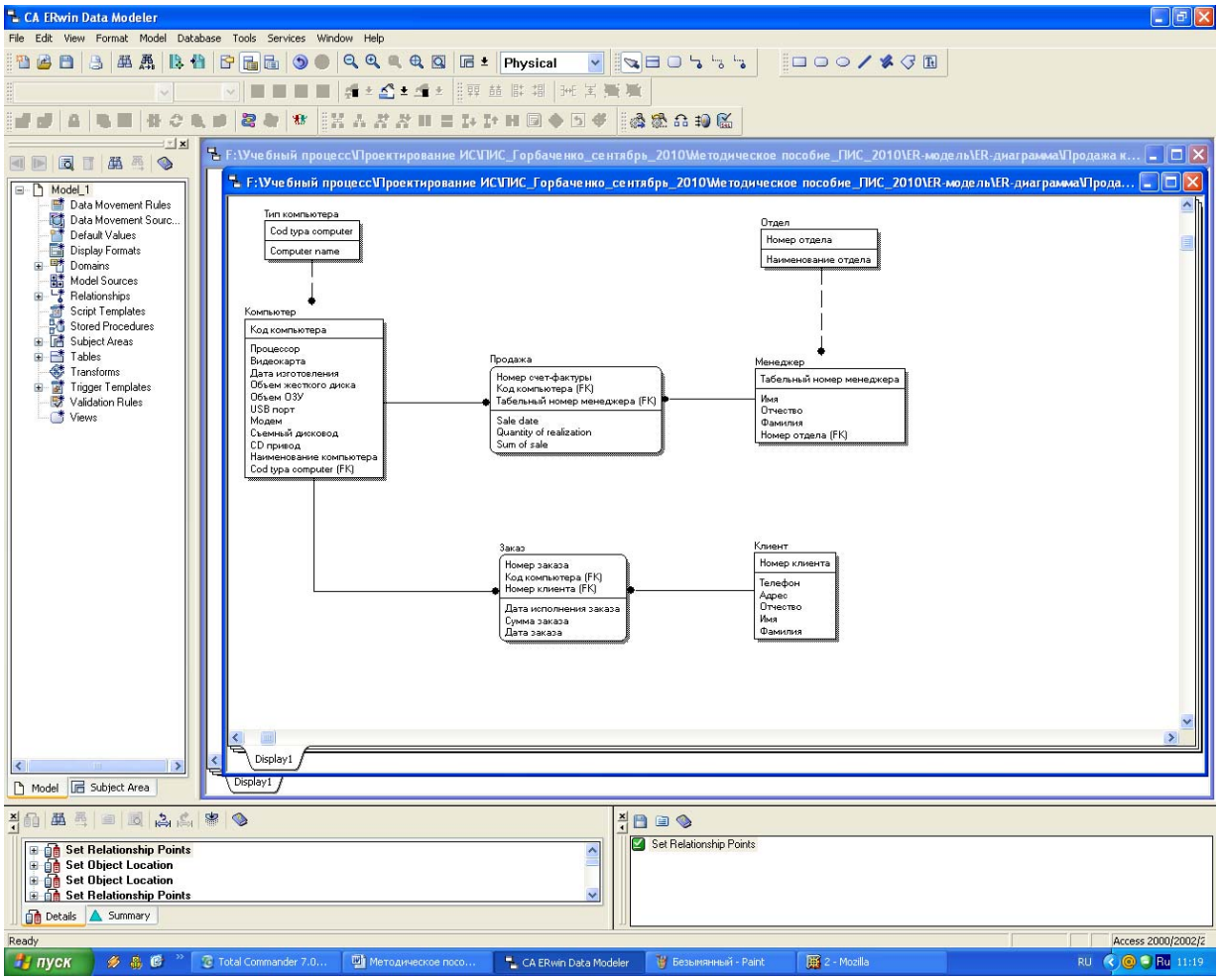

Рис. 7.4. Физический уровень логико-физической модели данных системы

При генерации имени таблицы или колонки по умолчанию ERwin по возможности (частично) преобразовывает их названия в соответствии с англоязычным представлением. При реализации базы данных с помощью СУБД Access допускается сохранять названия таблиц и колонок на русском языке.

7.2. Добавление/редактирование таблиц

Для построения/редактирования ER-модели физического уровня используется панель инструментов, содержащая кнопки редактирования модели, условные графические обозначения которых приведены в табл. 7.1.

Таблица 7.1. Палитра инструментов физического уровня

Вид кнопки	**Назначение кнопки**
	Указатель (режим мыши) – в этом режиме можно установить фокус на каком-либо объекте модели
	Создание новой таблицы – для создания таблицы необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание нового представления (view table) – для создания представления необходимо щелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание идентифицирующей связи
	Создание связи между представлением и временной таблицей
	Создание неидентифицирующей связи

Для внесения новой таблицы в модель на физическом уровне необходимо щелкнуть по кнопке  , расположенной на палитре инструментов, а затем на поле проектирования модели в том месте, где необходимо расположить новую таблицу. В результате в поле проектирования будет размещена таблица с именем по умолчанию **`E/1`**.

Определение/редактирование имени таблицы осуществляется через вкладку **`General`** пункта **`Table Properties`** (*свойства таблицы*) контекстного меню, отображаемого по щелчку правой кнопкой мыши по выделенной сущности (рис. 7.5, 7.6), или пункт главного меню **`Model/Tables`**....

При этом открывается диалог **`Tables`**, отражающий имя выбранной для реализации СУБД. Например, при выборе для реализации СУБД Access диалог **`Tables`** будет называться **`Access Tables`** (рис. 7.7).

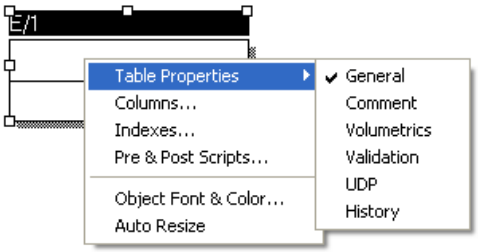

Рис. 7.5. Контекстное меню добавленной таблицы

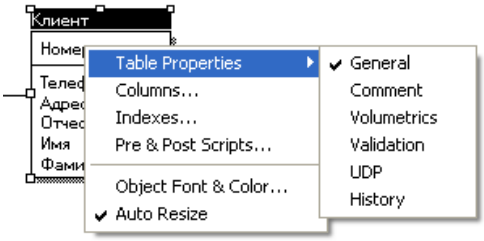

Рис. 7.6. Контекстное меню таблицы **Клиент**

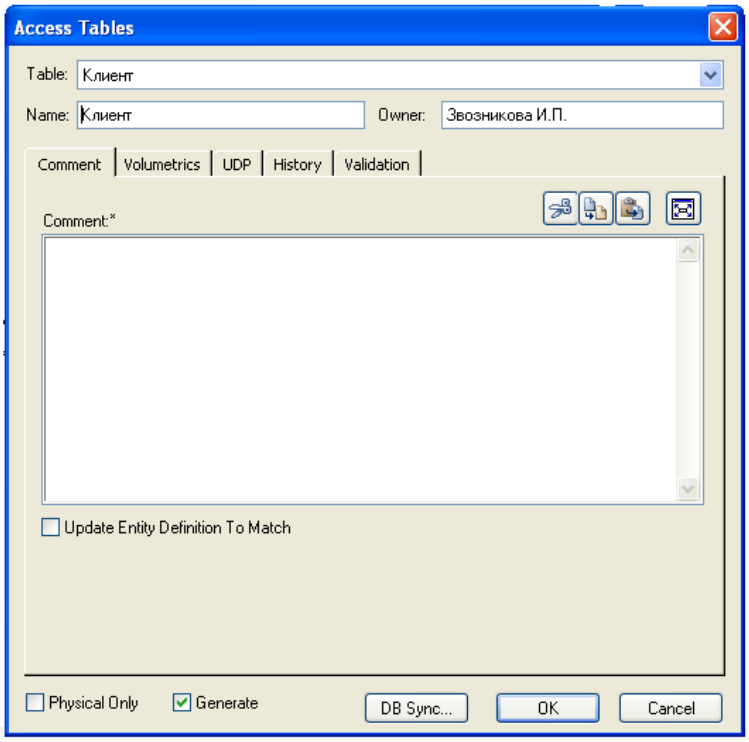

Рис. 7.7. Диалог **Access Tables**

Диалог **Tables** (в нашем случае **Access Tables**) позволяет переключаться с одной таблицы на другую и задать свойства любой таблицы модели, отличные от значения по умолчанию.

Переключиться на другую таблицу можно при помощи раскрывающегося списка выбора таблиц для редактирования окна **Table**, расположенного в верхней части диалога (рис. 7.8).

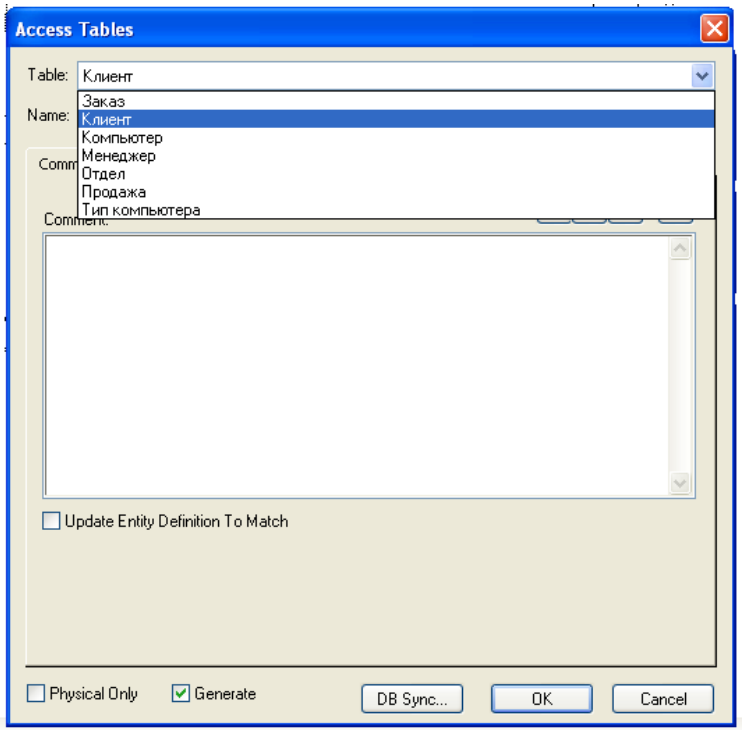

Рис. 7.8. Список выбора таблиц для редактирования окна **Table**

Окно **Name** служит для задания/изменения имени текущей таблицы. Окно **Owner** позволяет внести/изменить имя владельца таблицы, отличное от имени пользователя, производящего генерацию логической модели базы данных. Окно выбора **Physical Only** предназначено для создания объектов только на физическом уровне. Выбор опции **Generate** позволяет сгенерировать логическую модель базы данных путем выполнения команды **CREATE TABLE** (создать таблицу). Кнопка **DB Sync** предназначена для синхронизации модели с системным каталогом базы данных (рис. 7.9).

Диалог **Tables** содержит следующие вкладки:

– **Comment** – предназначена для внесения комментария к таблице (рис. 7.9);

– **Volumetrics** – предназначена для оценки размера БД;

– **UDP** – предназначена для задания свойств, определяемых пользователем;

– **Validation** – предназначена для задания правил валидации;

– **History** – отображает историю создания и изменения свойств таблицы и позволяет добавить комментарии к изменению в окне **Comment** (рис. 7.10).

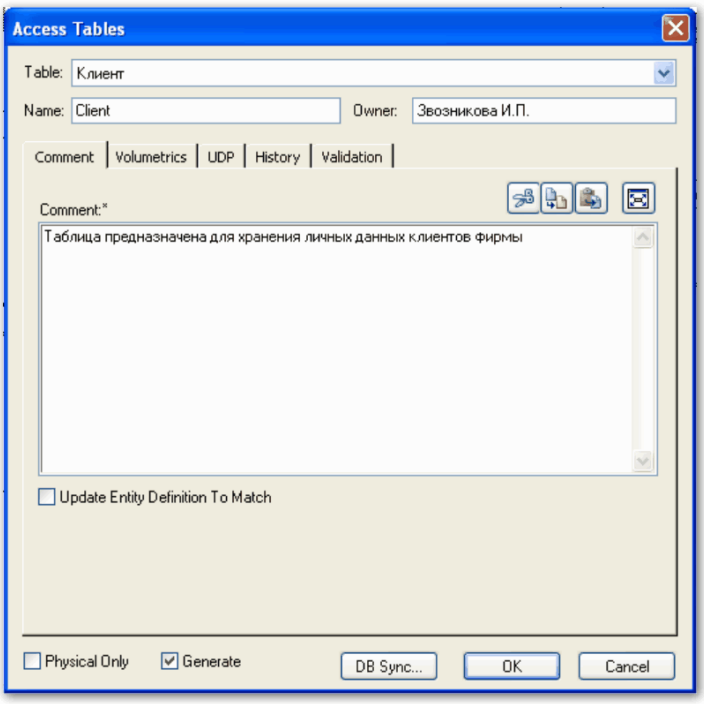

Рис. 7.9. Изменение имени и внесение имени владельца таблицы **Клиент**

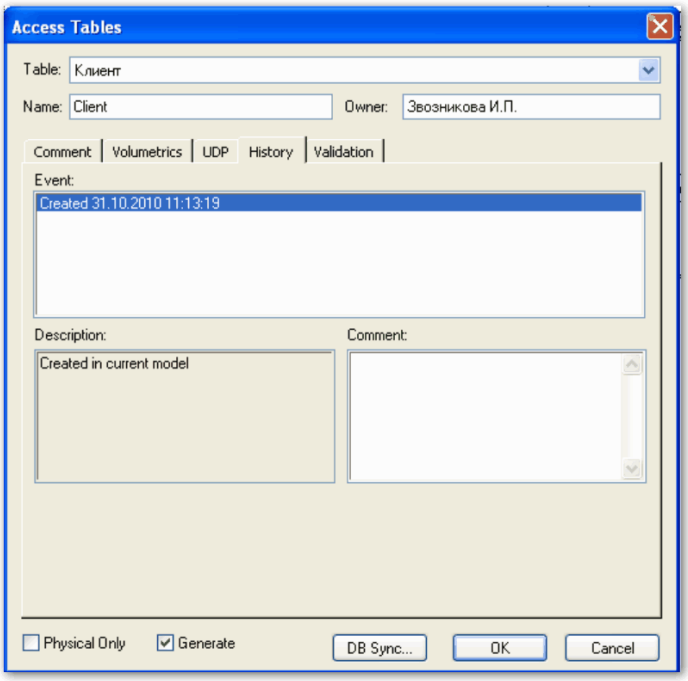

Рис. 7.10. История создания и изменения свойств таблицы **Клиент**

Таблицы физической модели данных определяются в соответствии с семантическим анализом предметной области. При этом каждому объекту (сущности) предметной области ставится в соответствие таблица, характеристикам объекта (атрибутам) соответствуют колонки (поля) таблицы, а идентификатору объекта соответствует ключ (ключевое поле) таблицы.

7.3. Определение свойств колонок таблицы

Задание/редактирование свойств колонок таблицы осуществляется с помощью редактора диалогового окна **Columns**, которое открывается через пункт **Columns** контекстного меню выделенной таблицы или пункт главного меню **Model/Columns**.... При этом редактор предлагает установить типы данных согласно выбранной СУБД (рис. 7.11).

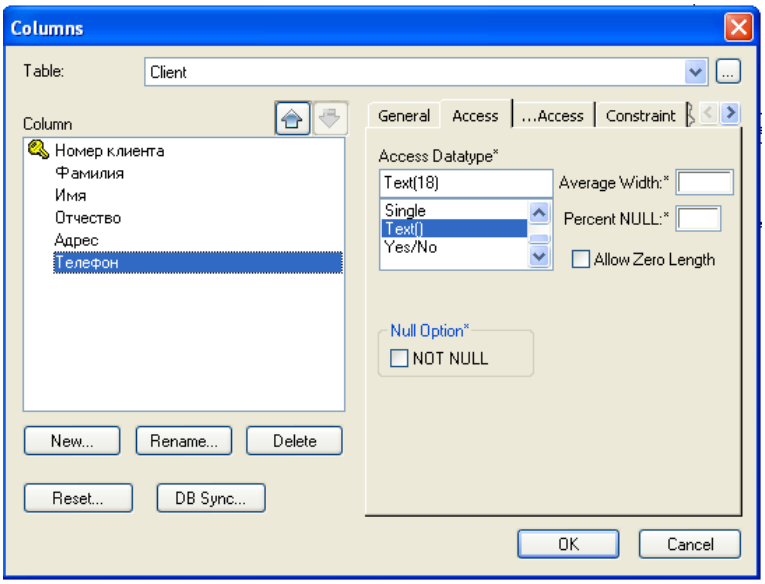

Рис. 7.11. Диалоговое окно **Columns**

По умолчанию ERwin присваивает пустые значения (**NULL**) всем неключевым колонкам, а для колонок первичного ключа и альтернативных ключей устанавливает режим **NOT NULL**. Режим **NOT NULL** не присваивается автоматически инверсным входам (**Inversion Entry**).

Диалоговое окно **Columns** по своему внешнему виду напоминает диалоговое окно редактора свойств атрибутов **Attributes** (см. рис. 6.10) и содержит окна и вкладки, позволяющие добавлять, редактировать и удалять колонки выделенной таблицы.

Вкладка **General** позволяет присвоить колонку определенному домену, создать колонку только на физическом уровне и включить ее в состав первичного ключа (рис. 7.12).

Вкладка, *соответствующая выбранной СУБД*, называется по имени СУБД и позволяет задать тип данных, опцию **NULL** и значение по умолчанию (см. рис. 7.11). Имя вкладки устанавливается автоматически и соответствует названию выбранной для реализации СУБД. Например, при выборе для реализации базы данных системы СУБД Access вкладка будет называться **Access**. Для СУБД Access, PROGRESS и Teradata создается дополнительная вкладка для задания свойств колонки (рис. 4.13).

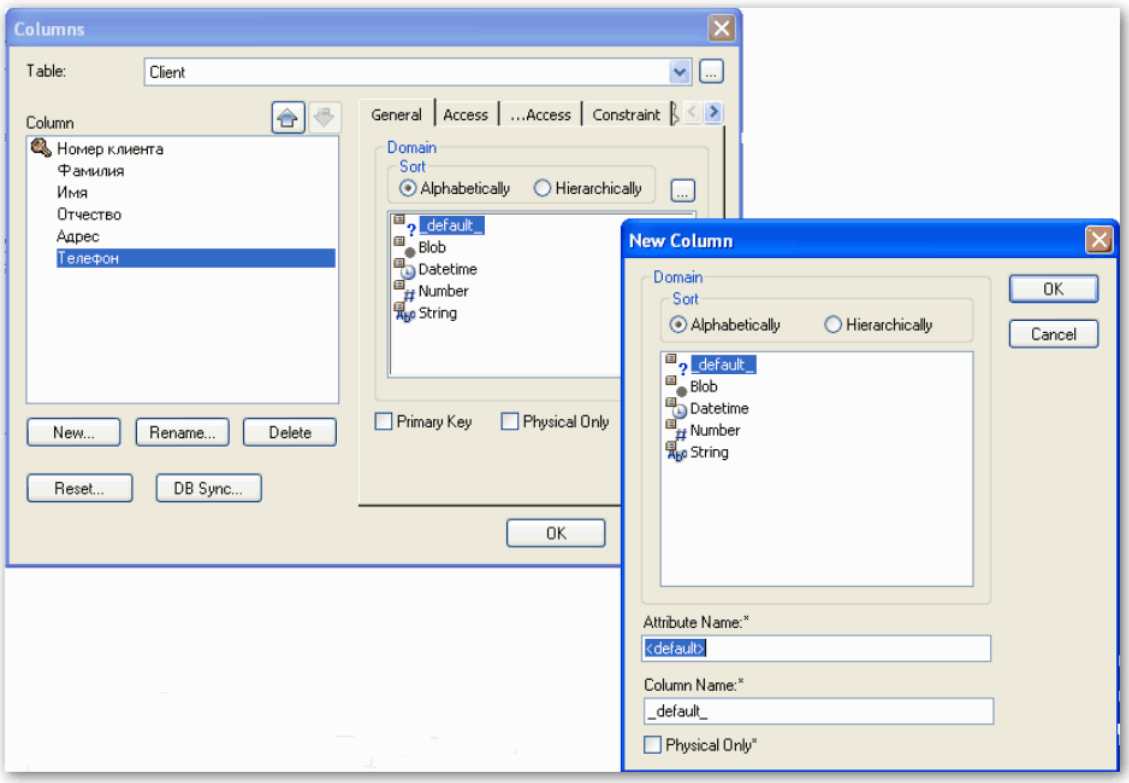

Рис. 7.12. Вкладка **General**

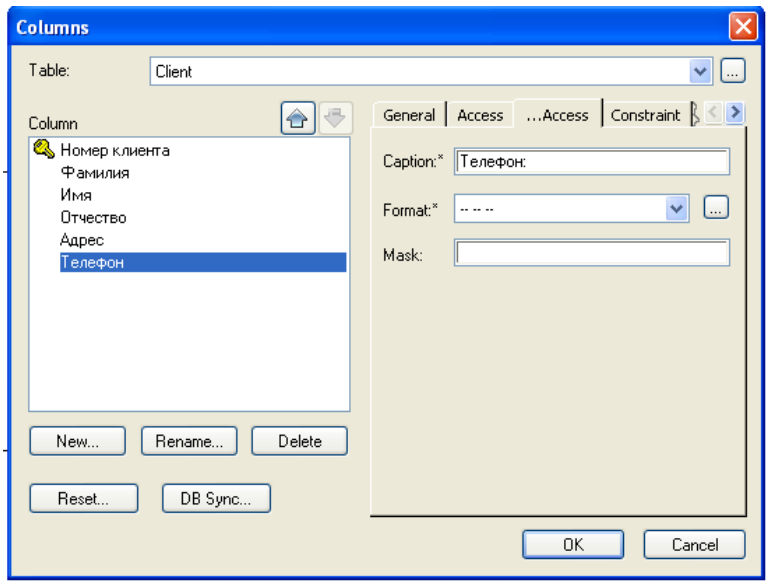

Рис. 7.13. Вкладка ... **Access**

Вкладка **Constraint** позволяет задать имена и значения ограничений, накладываемых на поле указанной колонки таблицы, включая и значение, которое присваивается в окне **Default** автоматически, т.е. по умолчанию (рис. 7.14).

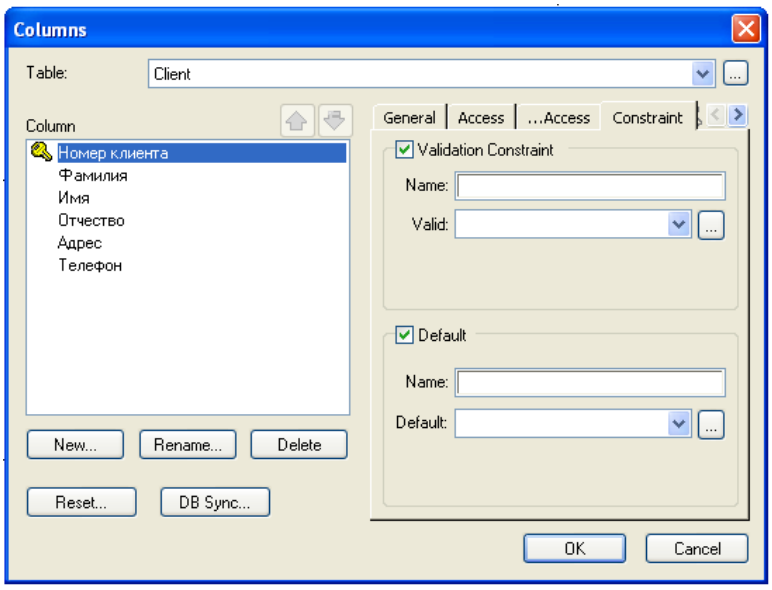

Рис. 7.14. Вкладка **Constraint**

Значение по умолчанию – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. Для создания нового значения ограничения по умолчанию необходимо перейти по кнопке … поля **Default** в диалоговое окно **Default/Initial Values**, по активизации кнопки **New** открыть диалог **New Default Value**, ввести в поле **Name** имя правила и щелкнуть по кнопке **OK** (рис. 7.15).

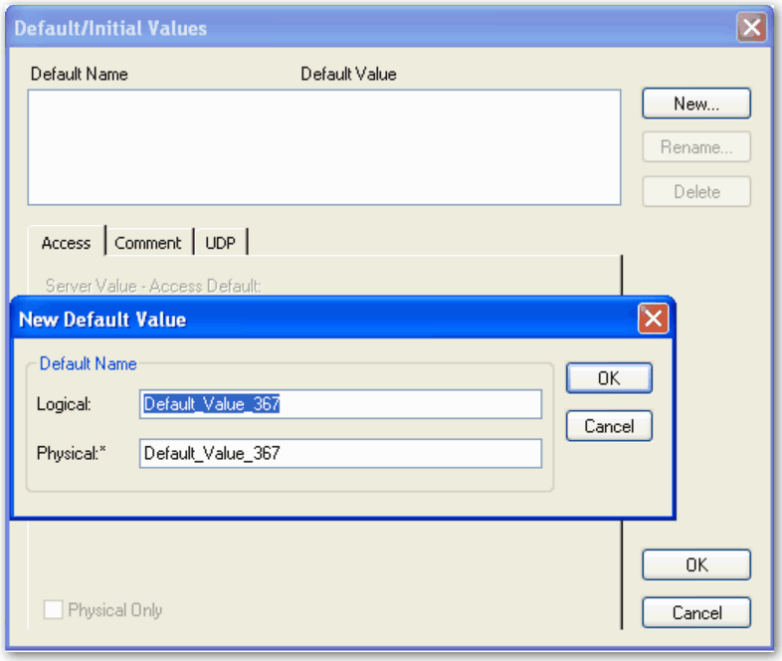

Рис. 7.15. Создания нового значения ограничения по умолчанию

Список всех заданных имен значений по умолчанию отображается в поле **Default Name** диалогового окна **Default/Initial Values**. Для

удаления и переименования значений по умолчанию используют соответственно кнопки **Delete** и **Rename**.

Вкладка **Comment** предназначена для внесения комментария к каждой колонке. Вкладка **UDP** позволяет задать свойства, определяемые пользователем. Вкладка **Index** служит для включения колонки в состав индексов. Вкладка **History** отображает историю создания и изменения свойств колонки и позволяет добавить комментарии к изменению в окне **Comment** (рис. 7.16).

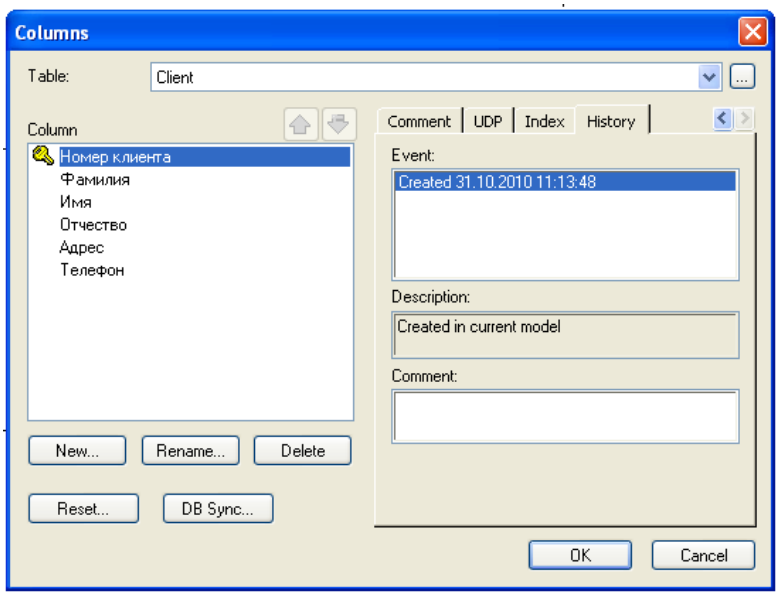

Рис. 7.16. Вкладка **History**

Упорядоченный список колонок таблицы отображается в поле **Column**, расположенный в левой части диалогового окна **Columns**. Для перемещения колонок в списке на позицию вверх или вниз используют соответственно кнопки  и  .

Кнопки **New**, **Rename** и **Delete** служат соответственно для создания, переименования и удаления колонки. При помощи кнопки **Reset** можно переустановить свойства колонки, заданные вручную, на значения по умолчанию. Кнопка **DB Sync** позволяет запустить процесс синхронизации модели с системным каталогом БД.

Связи между таблицами физической модели создаются так же, как и при создании логического уровня модели данных. При создании связи колонки первичного ключа родительской таблицы мигрируют в состав колонок дочерней таблицы в качестве внешнего ключа.

Изменения, внесенные в имена таблиц и колонок физической модели, не отражаются на именах сущностей и атрибутов, поскольку информация на логическом и физическом уровнях в ERwin хранится отдельно.

7.4. Индексы

Индекс – это особый объект, позволяющий решить проблему поиска данных в таблице БД.

Индекс содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки.

Возникновение *проблемы* поиска данных связано с тем, что многие реляционные СУБД имеют страничную организацию, при которой физически таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно, т.к. данные обычно хранятся в том порядке, в котором их ввели в таблицу.

Хотя такой способ хранения позволяет быстро вводить новые данные, но для того, чтобы найти нужную строку, необходимо просмотреть всю таблицу. В промышленных системах каждая таблица может содержать миллионы строк, поэтому простой перебор ведет к катастрофическому падению производительности ИС.

Использование индексов позволяет существенно повысить эффективность информационных систем по обработке хранимых данных. Индекс подобен содержанию книги, которое указывает на все номера страниц, посвященных конкретной теме. Например, если необходимо найти клиента по имени, то можно создать индекс по колонке **`ClientName`** таблицы **`Client`.** В индексе имена клиентов будут отсортированы в алфавитном порядке. Для имени индекс будет содержать ссылку, указывающую, в каком месте таблицы хранится эта строка. Индекс можно создать для всех колонок таблицы, по которым часто производится поиск. Для поиска клиента серверу направляется запрос с критерием поиска (**`ClientName`** ="Иванов"). При выполнении запроса СУБД просматривает индекс, а не все по порядку строки таблицы **`Client`**. Поскольку значения в индексе хранятся в определенном порядке, то просматривать нужно гораздо меньший объем данных, что значительно уменьшает время выполнения запроса.

При генерации схемы физической БД ERwin автоматически, по умолчанию создает отдельный индекс на основе первичного ключа каждой таблицы, а также на основе всех альтернативных ключей, внешних ключей и инверсных входов, поскольку эти колонки наиболее часто используются для поиска данных.

Для повышения производительности системы ERwin позволяет создать собственные индексы. При этом целесообразно создавать индексы с различными колонками и порядком сортировки, предварительно проанализировав наиболее часто выполняемые запросы.

ERwin автоматически генерирует имя индекса, созданного на основе ключа по принципу: "X" + имя ключа + имя таблицы, где имя ключа – "PK" для первичного ключа, "IFn" – для внешнего, "AKn" – для альтернативного, "IEn" – для инверсионного входа. Например, по умолчанию при создании

таблицы **Manager** будут созданы индексы XPKМенеджер – первичный ключ, в состав которого войдет колонка **Number manager ID** и XIF1Менеджер – внешний ключ, в состав которого войдет колонка **Department number FK.**

Изменить характеристики существующего индекса или создать новый индекс можно в редакторе **Indexes**, открывающийся при выборе пункта **Indexes**... контекстного меню, появляющегося по щелчку правой кнопки мыши на выделенной таблице (рис. 7.17). Редактор **Indexes** позволяет изменить имя индекса и его определение, а также порядок сортировки данных.

ERwin 7.3 позволяет создавать индексы, которые могут содержать либо повторяющиеся, либо только уникальные значения.

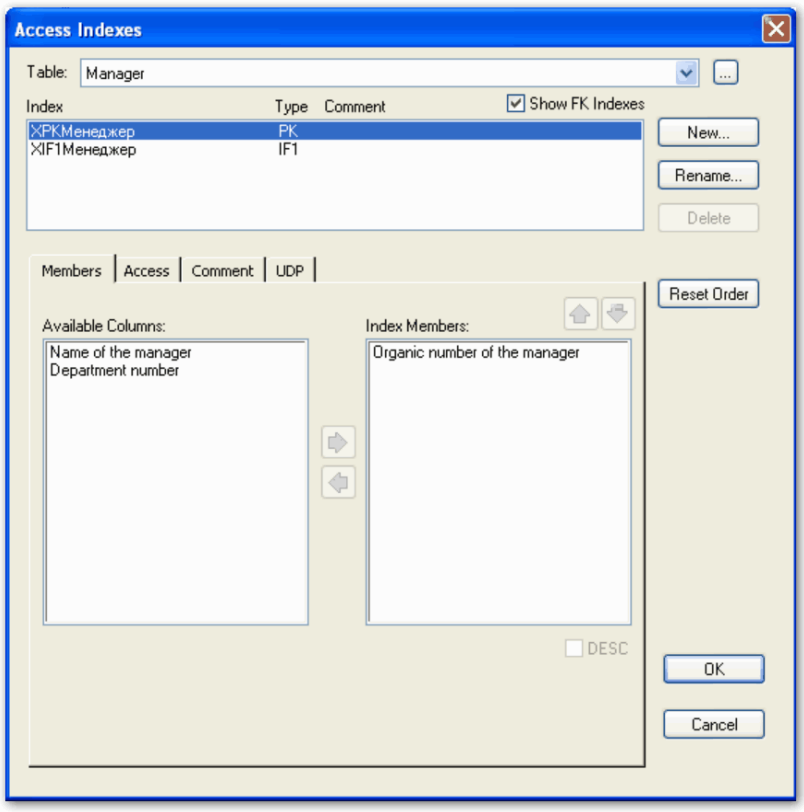

Рис. 7.17. Редактор **Indexes**

Создание нового уникального индекса осуществляется через диалог **New Index**, открывающийся по активизации кнопки **New**..., при включенной опции **Unique** (рис. 7.18).

Уникальные индексы генерируются на основе первичного и альтернативных ключей и предотвращают попытки вставить запись с неуникальным (повторяющимся) значением посредством извещения пользователя об ошибке и игнорирования на его действия по добавлению неверного (повторяющегося) значения индекса.

Для создания индекса с повторяющимися значениями опцию **Unique** следует выключить. Повторяющиеся значения индекса разрешаются, если

ожидается, что индексированная колонка будет с большой вероятностью содержать повторяющуюся информацию. Неуникальный индекс генерируется на основе внешнего ключа.

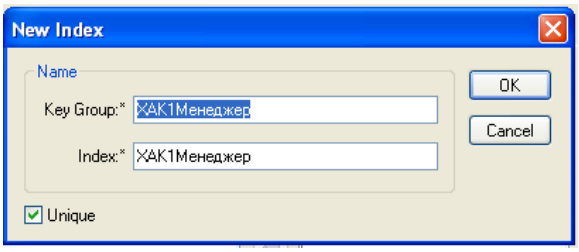

Рис. 7.18. Создание нового уникального индекса

При создании нового индекса ERwin автоматически создает альтернативный ключ для уникального индекса и инверсионный вход для неуникального индекса, а также дает соответствующее ключу имя индекса. Имя сгенерированного индекса в дальнейшем при необходимости можно изменить вручную через диалог **Rename Index**, открывающийся по активизации кнопки **Rename**... (рис. 7.19).

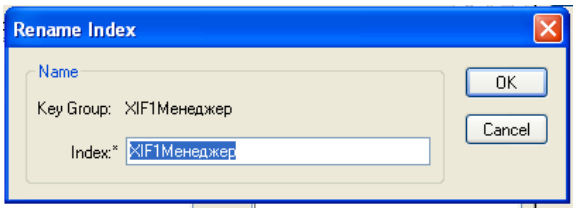

Рис. 7.19. Диалог **Rename Index**

Редактор **Indexes** содержит следующие вкладки:

– **Members** – позволяет включить колонки в состав индекса;

– вкладка, соответствующая выбранной СУБД (например, Access), задает свойства индекса, специфические для выбранной СУБД;

– **Comment** – содержит комментарий для каждого индекса;

– **UDP** – позволяет связать с индексом свойства, определяемые пользователем.

Такие СУБД, как DB2/MVS, DB2/390, HiRDB, INFORMIX, MS Access, MS SQL Server, SYBASE и SQLBase поддерживают кластеризованные или кластеризованные хешированные индексы. *Кластеризация индекса* – это специальная техника индексирования, при которой данные в таблице физически располагаются в индексированном порядке. Использование кластеризованного индекса значительно ускоряет выполнение запросов по индексированной колонке. *Хеширование* – альтернативный способ хранения данных в заранее заданном порядке с целью ускорения поиска. Кластеризованный или хешированный индекс значительно ускоряет операции поиска и сортировки, но добавление и удаление строк замедляется из-за необходимости реорганизации данных для соответствия индексу.

Для того чтобы сделать индекс кластеризованным, необходимо включить опцию **Cclustered** на вкладке, соответствующей выбранной СУБД. Например, можно создать кластеризованный индекс в таблице **Manager** по колонке **Department number FK**. В результате информация обо всех менеджерах одного отдела будет физически располагаться на диске рядом, что значительно повысит скорость выполнения запроса, который делает выборку данных по менеджерам определенного отдела. Поскольку данные в БД физически располагаются в индексированном порядке, то для каждой таблицы может существовать только один кластеризованный индекс.

Если СУБД поддерживает использование кластеризованного индекса, например, Access, то ERwin автоматически создает индекс первичного ключа кластеризованным, а при создании кластеризованного индекса не по первичному ключу автоматически снимает кластеризацию с индекса по первичному ключу.

7.5. Правила валидации колонок

Правило валидации задает список допустимых значений для конкретной колонки таблицы и/или правила проверки допустимых значений. Значение по умолчанию - значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных.

Задание правил валидации осуществляется через диалог **Validation Rules**, который открывается через пункт главного меню **Model/ Validation Rules**... или через диалоговые окна, открывающиеся в следующем порядке:

1) активизировать кнопку "...", расположенную справа от раскрывающегося списка **Table** диалогового окна **Columns** (рис. 7.17);

2) в открывшемся окне выбрать закладку **Validation** и активизировать кнопку **Validation Constraint**... (рис. 7.20).

В результате открывается диалог **Validation Rules**, в котором можно задать максимальное и минимальное число колонок для всех таблиц модели, а также тип валидации: где проверять – на сервере или в клиентском приложении (рис. 7.21).

Для создания нового правила валидации необходимо активизировать кнопку **New**..., ввести имя правила в поле **Validation Rule Name** диалога **New Validation Rule** и активизировать кнопку **OK** (рис. 7.22.). Наименование правила валидации может быть разным на логическом и физическом уровне. Чтобы переименовать имеющееся правило валидации, необходимо активизировать кнопку **Rename**. Для удаления правила валидации предназначена кнопка **Delete**.

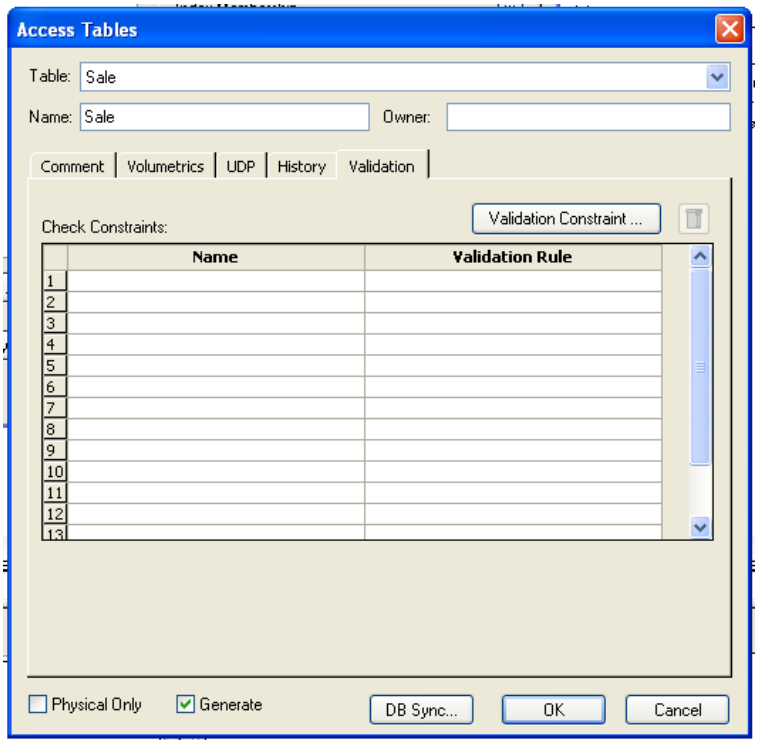

Рис. 7.20. Выбор закладки **Validation** и активизация кнопки **Validation Constraint**...

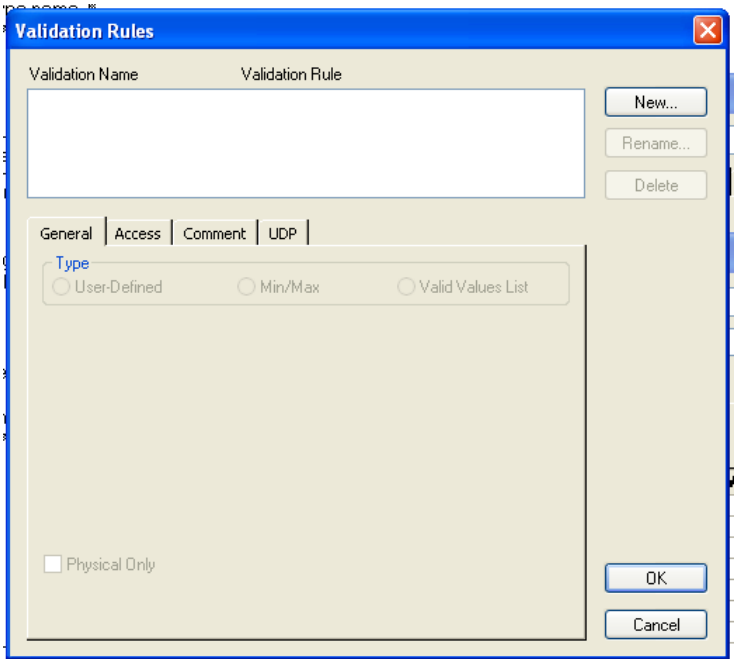

Рис. 7.21. Диалог **Validation Rules**

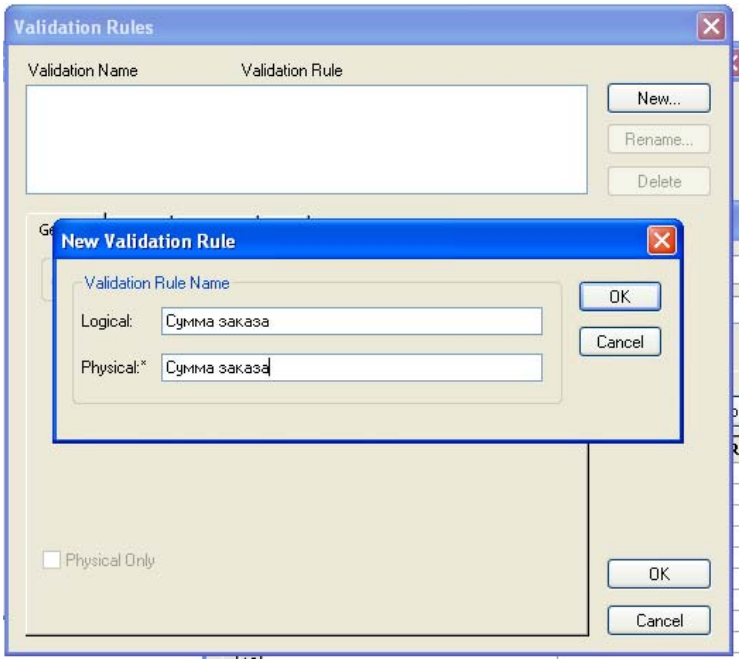

Рис. 7.22. Создание нового правила валидации

В результате в верхней части редактора **Validation Rules** отобразится список всех созданных правил валидации и представится возможность редактирования правил валидации (рис. 7.23).

Закладка **General** редактора задания правил валидации позволяет выбрать тип правила:

– тип **User-Defined** позволяет задать вручную фрагмент SQL – выражения, который соответствует правилу валидации и будет использоваться при генерации схемы базы данных;

– тип **Min/Max** позволяет задать минимальное и максимальное значение колонки, которое будет проверяться в базе данных на вхождение в заданный диапазон;

– тип **Valid Values List** позволяет задать список допустимых значений.

Пример правила валидации типа **Min/Max** для колонки **Сумма заказа** таблицы **Заказ** приведен на рис. 7.24.

Вкладка, соответствующая выбранной СУБД (на рис. 7.24 – **Access**) позволяет просмотреть фрагмент SQL – выражения, соответствующего правилу валидации, которое гарантирует проверку вводимых значений (рис. 7.25). В случае выхода за границы заданного диапазона СУБД выдает сообщение об ошибке.

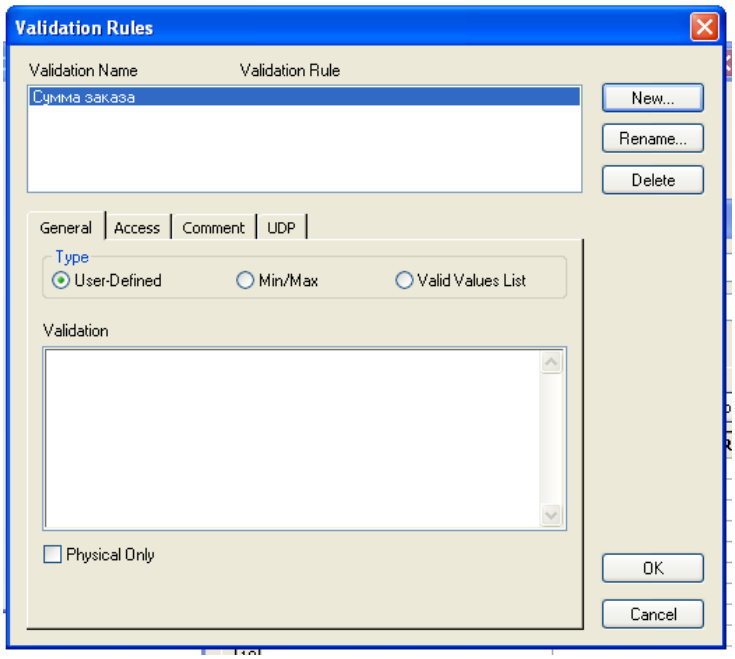

Рис. 7.23. Создание нового правила валидации

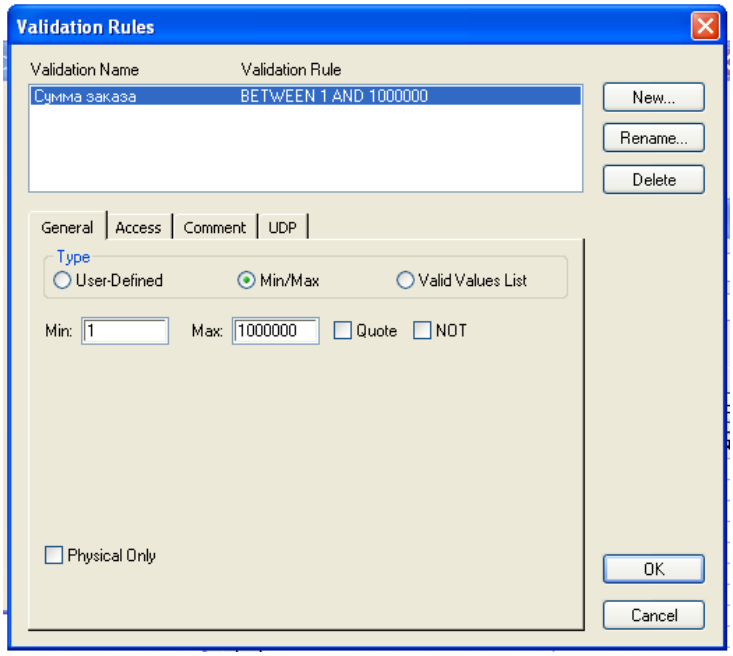

Рис. 7.24. Пример правила валидации типа **`Min/Max`**

Закладка **`Comment`** предназначена для внесения комментария к правилу валидации.

Закладка **`UDP`** позволяет для правила валидации задать свойства, определяемые пользователем.

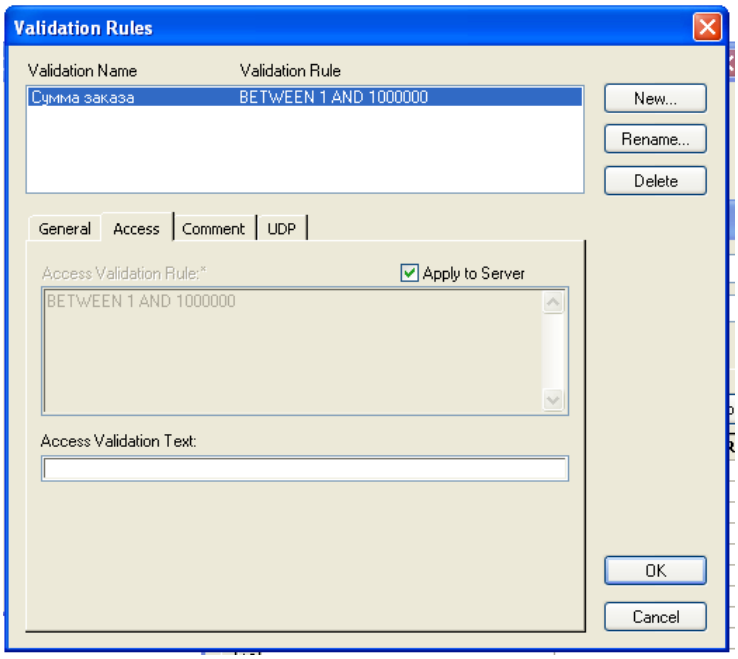

Рис. 7.25. Просмотр фрагмента SQL – выражения через вкладку **Access**

7.6. Пример физической модели данных

Построение физической модели данных для системы "Реализация средств вычислительной техники" осуществлено путем автоматического перехода от логической модели к физической модели, так как при создании логической модели данных системы был выбран логико – физический тип модели.

Физическая модель данных системы "Реализация средств вычислительной техники" приведена на рис. 7.26.

Созданная физическая модель данных предназначена для реализации базы данных в среде Access.

Для генерации кода создания базы данных можно использовать пункт главного меню **Tools/Forward Engineer**. В результате откроется окно установки свойств генерируемой схемы данных (рис. 7.27).

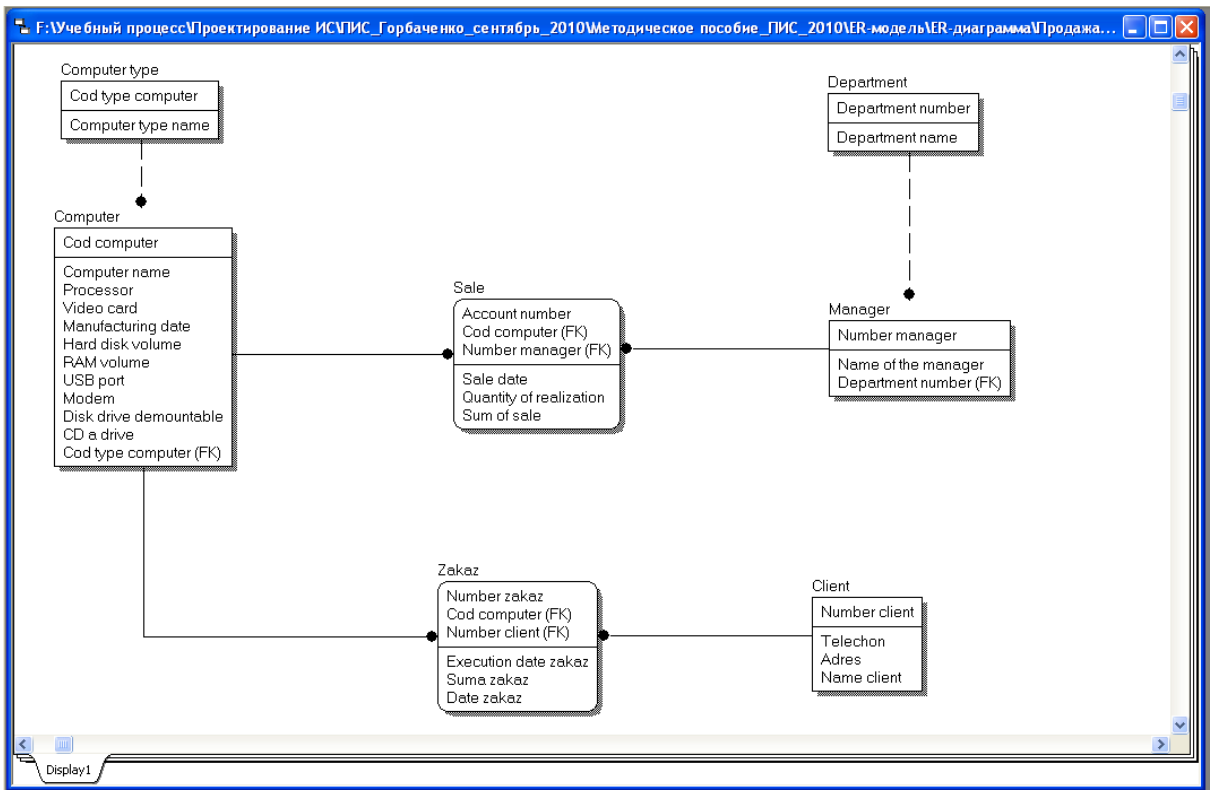

Рис. 7.26. Физическая модель данных системы

"Реализация средств вычислительной техники"

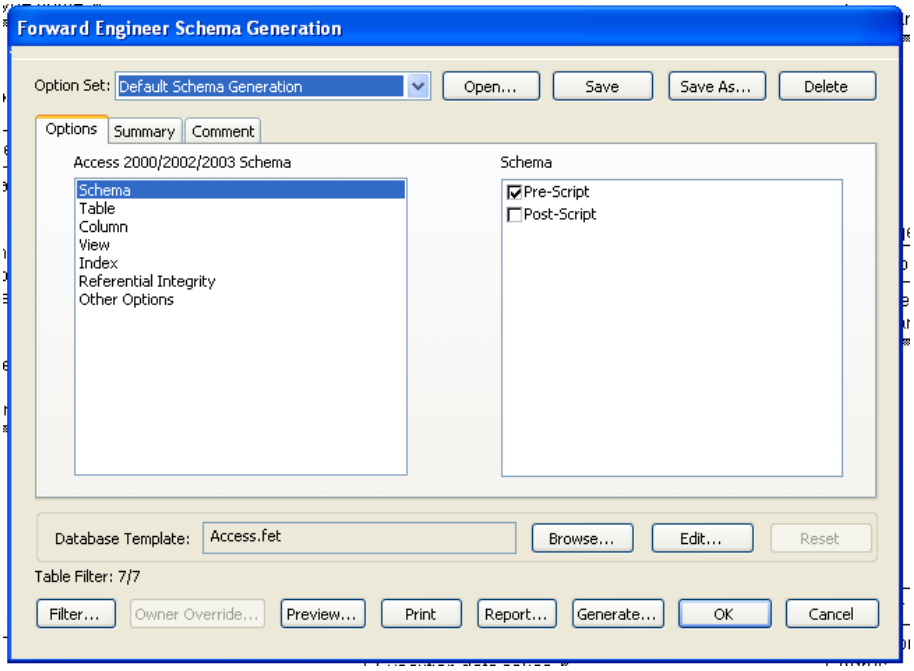

Рис. 7.27. Окно установки свойств генерируемой схемы данных

Для предварительного просмотра SQL-скрипта служит кнопка **`Preview`** (рис. 7.28), для генерации схемы – кнопка **`Generate`** (рис. 7.29).

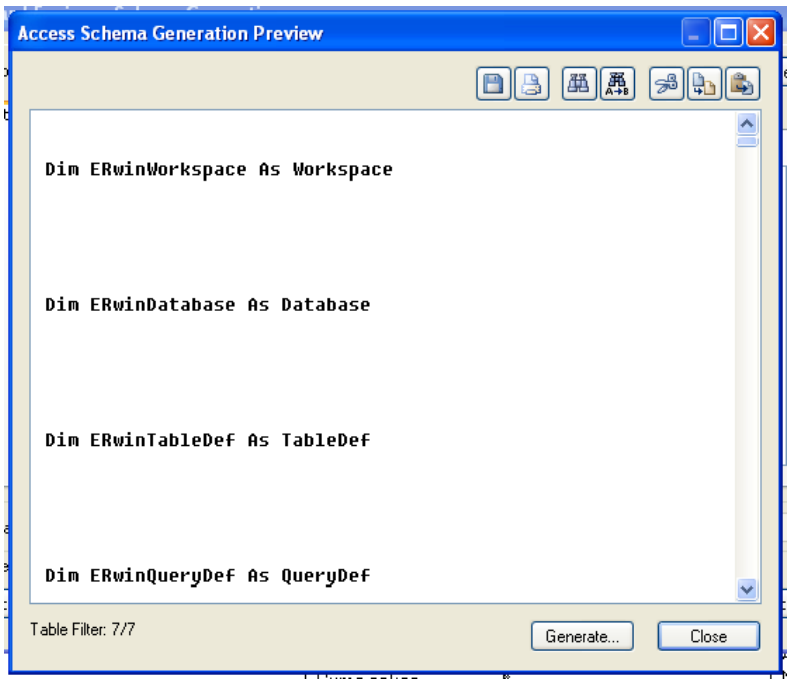

Рис. 7.28. Окно просмотра SQL-скрипта

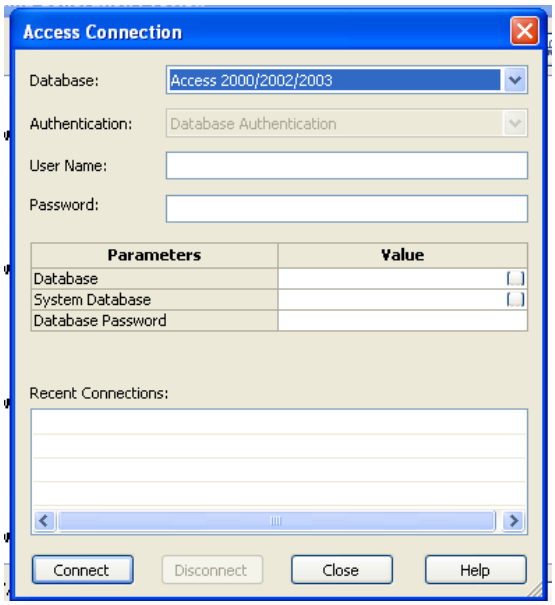

Рис. 7.29. Окно генерации схемы данных

В процессе генерации ERwin связывается с выбранной базой данных, выполняя SQL-скрипт.

Если в процессе генерации возникают какие-либо ошибки, то она прекращается, и открывается окно сообщений об ошибках.

Окно установки свойств генерируемой схемы данных позволяет также просмотреть и править шаблон базы данных путем активизации соответственно кнопки **Browse**... или **Edit**... (рис. 7.30).

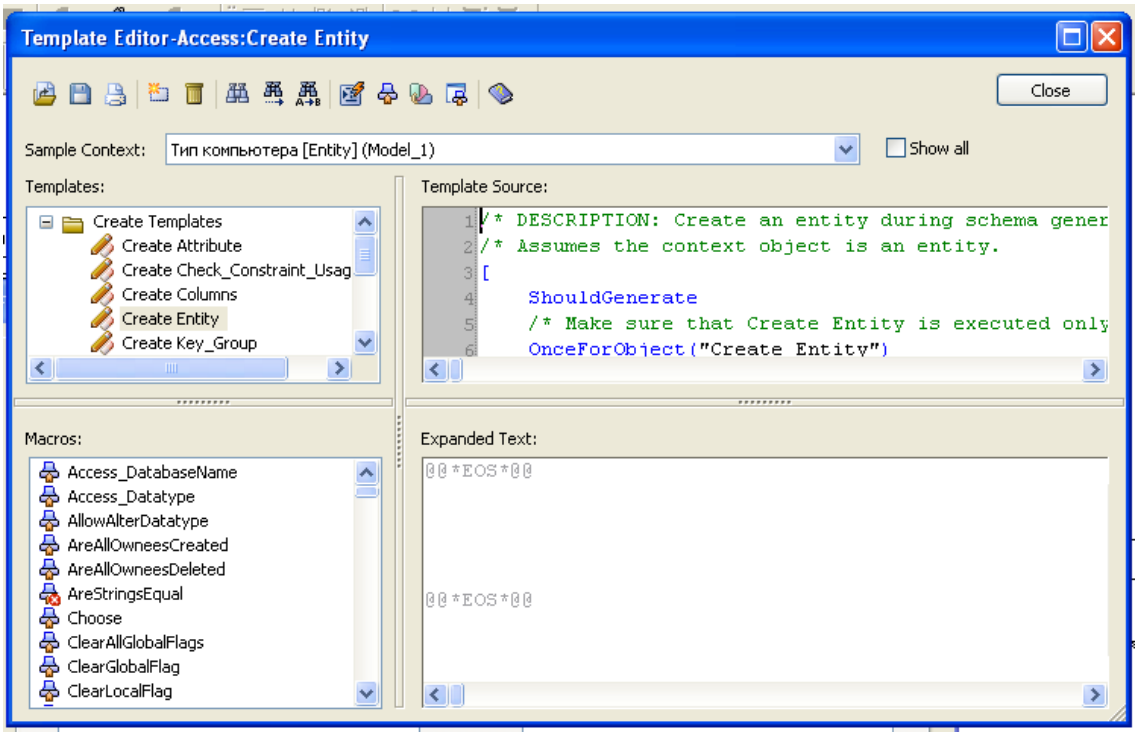

Рис. 7.30. Окно правки шаблона базы данных

Диаграмма физической модели данных является необходимым и очень удобным материалом для разработчиков программ. Физическое проектирование позволяет обеспечить безопасность и целостность данных в выбранной для реализации СУБД.

Применение программного продукта CA ERwin Data Modeler 7.3 существенно повышает эффективность деятельности разработчиков информационных систем за счет:

– существенного повышения скорости разработки базы данных с помощью мощного редактора диаграмм, возможности автоматической генерации базы данных и автоматической подготовки документации;

– наличия возможности автоматической подготовки SQL – предложений для создания базы данных;

– наличия возможности внесения изменений в модель при разработке и расширении системы;

– наличия возможности автоматической подготовки отчетов по базе данных, которые соответствуют реальной структуре базы данных;

– наличия возможности осуществления обратного проектирования, что позволяет документировать и вносить изменения в существующие информационные системы;

– поддержки однопользовательских СУБД, что позволяет использовать для персональных систем современные технологии и значительно упростить переход от настольных систем к системам с технологией клиент-сервер.

7.7. Задание для самостоятельной работы

На основе созданной ранее логической модели данных системы «Поликлиника» (см. 6.5) разработать физическую модель данных, предварительно выбрав целевую СУБД.

СПИСОК ЛИТЕРАТУРЫ

1. *Грекул, В. И.* Проектирование информационных систем / *В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина*. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2008. – 300 с.

2. *Маклаков, С. В*. Создание информационных систем с AllFusion Modeling Suite / *С. В. Маклаков*. – М.: Диалог – МИФИ, 2007. – 432 с.

3. *Маклаков, С. В*. Моделирование бизнес-процессов с AllFusion Process Modeler / *С. В. Маклаков*. – М.: Диалог – МИФИ, 2008. – 224 с.

4. *Дубейковский, В. И*. Практика функционального моделирования с AllFusion Process Modeler 4.1. Где? Зачем? Как? / *В. И. Дубейковский* – М.: Диалог – МИФИ, 2004. – 464 с.

5. *Марка, Д. А.* Методология структурного анализа и проектирования / *Д. А. Марка, К. Мак Гоуэн*. – М.: МетаТехнология, 1993. – 240 с.

6. Р 50.1.028-2001. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования. – М.: Госстандарт России, 2001. – 53 с.

7. IDEF3 – // http://en.wikipedia.org/wiki/IDEF3

8. *Mayer, R. J.* Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report / *R. J. Mayer, C. P. Menzel, M. K. Painter, P. S. deWitte, T. Blinn, B. Perakath*. – Knowledge Based Systems, Inc., 1995. – 236 p.

9. IDEF. Process Description Capture Method – //http://www.idef.com/ IDEF3. htm

10. *Верников, Г.* Основы IDEF3 / *Г. Верников.* – //http://www.cfin.ru/ vernikov/idef/idef3. shtml

11. *Беспалов, Р. С.* Инструментарий разработчика бизнес-процессов / *Р. С. Беспалов.* – // http://www.rb-erp.ru/Texts/Modeling_Bespalov.pdf

12. CA ERwin Process Modeler. Process Flow Modeling. Overview Guide r. 7.3. – 81 p.

13. *Кельтон В.* Имитационное моделирование. Классика CS. 3-е изд. / *В. Кельтон, А. Лоу.* – СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.

14. *Kelton, W. D.* Simulation with Arena / *W. D. Kelton, P. P. Sadowski, D. T. Sturrock*. – McGraw-Hill Science, 2009. – 636 p.

15. *Леоненков, А. В.* Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose / *А. В. Леоненков*. – М.: Интернет-университет информационных технологий-ИНТУИТ.ру; БИНОМ. Лаборатория знаний, 2006. – 320 с.

16. *Коберн, А*. Современные методы описания функциональных требований к системам / *А. Коберн*. – М.: ЛОРИ, 2002. – 226 с.

17. *Орлов, С. А.* Технологии разработки программного обеспечения / *С. А. Орлов*. – СПб.: Питер, 2002. – 464 с.

18. *Ампилогов, А.* Первые шаги с CA ERwin Process Modeler / *А. Ампилогов* – // http://www.interface.ru/home.asp?artId=22274

19. CA ERwin® Data Modeler – // http://erwin.com/

20. *Карпова, Т. С.* Базы данных: модели, разработка, реализация / *Т. С.Карпова*. – СПб.: Питер, 2001. – 304 с.

21. *Дейт, К. Дж*. Введение в системы баз данных. / *К. Дж*. *Дейт*. – М.: Вильямс, 2005. – 1328 с.

22. Integration Definition for Information Modeling (IDEF1X) – // http://www.itl.nist.gov/fipspubs/idef1x.doc

23. *Верников, Г.* Основы методологии IDEF1X / *Г. Верников*. – // http://www.interface.ru/fset.asp?Url=/ca/idef1x.htm

24. *Зайцев, С. Л.* Проектирование баз данных с ERwin / *С. Л. Зайцев*. – // http://www.interface.ru/fset.asp?Url=/ca/comp.htm

25. *Зайцев, С. Л*. Понятие отношения / *С. Л. Зайцев*. – // http://www.interface.ru/fset.asp?Url=/ca/ponatie.htm

26. *Козодоев, А.* Использование методик моделирования данных IDEF1X и IE в программном средстве AllFusion ERwin Data Modeler компании Computer Associates / *А. Козодоев*. – // http://www.interface.ru/ca/MethodsDM_ERwin.htm

27. *Пушников, А. Ю.* Введение в системы управления базами данных / *А. Ю. Пушников*. – // http://www.citforum.ru/database/dblearn/index.shtml