



ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И
ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Информационные системы в строительстве»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к проведению практических занятий
по дисциплине

«Методы и средства проектирования информационных систем и технологий. Структурный подход»

Авторы
Анисимова Г.Б.,
Позднышева Е.Е.

Ростов-на-Дону, 2016

Аннотация

Методические указания к практическим занятиям предназначены для студентов очной формы обучения направления бакалавров 09.03.02.

Авторы



к.ф.-м.н, ст.н.с., доцент
кафедра Информационные системы
в строительстве
Анисимова Галина Борисовна



ст. преподаватель
кафедра Информационные системы
в строительстве
Позднышева Екатерина Евгеньевна



Оглавление

1 case-ИНСТРУМЕНТ bрwin	5
1.1 Методические указания по применению CASE-инструмента BPWIN4.0	5
1.2 Основные характеристики.....	7
2 Практические задания	12
2.1 Упражнение 1. Создание контекстной диаграммы.....	12
2.2 Упражнение 2. Создание диаграммы декомпозиции A0.....	22
2.3 Упражнение 3. Создание диаграммы декомпозиции A2.....	28
2.4 Упражнение 4. Создание диаграммы узлов	33
2.5 Упражнение 5. Создание FEO диаграммы.....	35
2.6 Упражнение 6. Расщепление и слияние моделей	37
2.7 Упражнение 7. Создание диаграммы IDEF3.....	42
2.8 Упражнение 8. Создание сценария	45
2.9 Упражнение 9.	47
3 Erwin 53	
3.1 МОДЕЛЬ "СУЩНОСТЬ-СВЯЗЬ".....	53
4 Практические задания	69
4.1 СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ	69
4.2 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ.....	108
5 Список литературы	137

Это практическое руководство по созданию информационных систем с помощью эффективного CASE - инструмента анализа проектирования BPWIN 4.0. Данное пособие может быть полезным для студентов и аспирантов, изучающих основы системного анализа и проектирования ИС, т.к. в нем описывается конкретная технология разработки ИС

В настоящее время эффективная автоматизация любого вида деятельности предприятия может производиться только на основе применения информационных систем (ИС). Под автоматизацией можно понимать разработку ИС, или ее выбор на рынке с последующей адаптацией под специфику предприятия, а также процесс внедрения, обслуживания и ведения с целью модернизации.

Разработка ИС включает в себя несколько этапов, одним из первых и важнейших из которых является проведение изучения и анализ автоматизируемого процесса, а также его моделирование для определения путей возможного улучшения и оптимизации работы, которые будут реализованы в создаваемой ИС. При проведении такого анализ, в соответствии с требованиями системного подхода необходимо произвести точное описание процессов, деятельность которых требуется автоматизировать. Не сделав корректного описания автоматизируемых процессов, бессмысленно переходить к следующим стадиям анализа деятельности предприятия и тем более к его автоматизации.

1 CASE-ИНСТРУМЕНТ BPWIN

1.1 Методические указания по применению CASE-инструмента BPWIN4.0

CASE -средство верхнего уровня BPwin – это инструмент визуального моделирования ИС, позволяющий:

- наглядно описывать, анализировать и совершенствовать сложные бизнес-процессы, любую деятельность или структуру в виде модели, что позволяет значительно повысить эффективность работы предприятия;

- проверить модель на соответствие стандартам ISO9000. Для отечественных предприятий сертификация по ИСО 9000 – это пропуск на международный рынок, а также действенное средство для эффективного улучшения работы всего предприятия;

- спроектировать структуру информационных потоков, а соответственно, и модернизировать организационную структуру предприятия;

- четко выявить факторы, оказывающие влияние на бизнес: какие операции являются наиболее критичными, как повысить их эффективность, какие ресурсы требуются для этого;

- снизить издержки и повысить производительность;

- выявить и исключить лишние или неэффективные операции;

- повысить гибкость и эффективность.

Все это позволит получить целостное представление о том, как работает предприятие, начиная от структурного подразделения и заканчивая предприятием в целом. Если компания занимается системной интеграцией или поставкой готовых решений в области ИТ, модель бизнес-процессов - это наилучшее средство обосновать, как повлияют инвестиции в ИТ на эффективность деятельности предприятия.

BPwin входит в семейство продуктов AllFusion компании Computer Associates под именем **AllFusion Process Modeler** и предназначен для поддержки всех стадий жизненного цикла разработки ИС. – В линейку продуктов AllFusion Modeling Suite кроме BPwin для поддержки всех стадий разработки программного обеспечения, входят CASE-средств ERwin, BPwin, ModelMart, Paradigm Plus, ERwin Examiner и средства управления проекта-

ми. Совместное применение этих продуктов обеспечивает прочный фундамент для построения, развертывания и управления приложениями. При этом не накладываются ограничения на выбор базовых технологий, методов и платформ разработки. AllFusion Modeling Suite предлагает моделирование и управление процессами, проектами, изменениями, конфигурациями.

BPwin - инструмент моделирования, который используется не только для анализа и документирования, но и реорганизации сложных процессов. BPwin соответствует требованиям к инструментам для разработки ИС, так как позволяет четко документировать различные действия, которые необходимо предпринять, а также способы их осуществления и требуемые для этого ресурсы. BPwin является интуитивно понятным визуальным инструментом, позволяющим сформировать целостную картину деятельности предприятия: от моделей организации работы в маленьких отделах до сложных иерархических структур. В руках же системных аналитиков и разработчиков BPwin - мощное средство моделирования процессов при создании корпоративных информационных систем (КИС).

BPwin достаточно легок в освоении и может применяться практически во всех сферах деятельности, ориентирован на различные категории специалистов: от системных и деловых аналитиков до руководителей, от консультантов до специалистов по маркетингу и менеджеров по качеству. В первую очередь BPwin предназначен для облегчения труда и увеличения производительности системного аналитика на первом этапе разработки системы, т.к. является средством для сбора всей необходимой информации о работе предприятия и графического изображения этой информации в визуальной модели (рисунке).

BPwin является также средством системного анализа деловой и производственной активности, позволяющим отслеживать соответствие структуры бизнеса, документооборота, финансовых потоков требованиям современной экономики.

BPwin автоматизирует задачи, связанные с построением моделей, обеспечивая семантическую строгость, необходимую для гарантирования правильности результатов, а также целостность и непротиворечивость модели, которые гарантируются применением методологий IDEF 0, IDEF 1 X (Data Flow Diagram) и IDEF 3 (Work Flow Diagram). Каждая из этих трех нотаций, поддерживаемых в BPwin, позволяет рассмотреть различные стороны деятельности предприятия и комплексно описать предметную область с трех различных, но взаимосвязанных точек зрения :

- **функциональности системы** . В рамках методологии IDEF0 (Integration Definition for Function Modeling) процесс представляется в виде набора работ (функций), которые взаимодействуют между собой, а также показывается информационные, людские и производственные ресурсы, потребляемые каждой работой. С помощью функционального моделирования можно провести систематический анализ процессов, сосредоточившись на регулярно решаемых задачах/функциях, свидетельствующих об их правильном выполнении показателей, необходимых для этого ресурсах, результатах и исходных материалах/сырья. Правильно построенная функциональная модель адекватна на всех уровнях абстрагирования ;

- **потоков информации** (документооборота) в системе. Т.к. DFD не принадлежит к семейству IDEF, то более верно будет именовать ее нотацией Диаграммы DFD (Data Flow Diagramming) могут дополнить то, что уже отражено в модели IDEF3, поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией между функциями внутри системы. В тоже время диаграммы DFD оставляют без внимания взаимодействие между функциями. Моделирование потоков данных часто используется при разработке программного обеспечения, сосредоточенного вокруг потоков данных, передающихся между различными операциями, включая их хранение, для достижения максимальной доступности и минимального времени ответа. Такое моделирование позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

- **последовательности выполняемых работ** . Более точную картину можно получить, дополнив модель диаграммами IDEF3. Моделирование потоков работ позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход. Этот метод привлекает внимание к очередности выполнения событий. В IDEF3 включены элементы логики, что позволяет моделировать и анализировать альтернативные сценарии развития процесса.

1.2 Основные характеристики

В зависимости от корпоративного стандарта по проведению системного анализа процессов на начальном этапе проектирования системы, могут использоваться различные типы или

комбинации этих методологий моделирования. Если в процессе моделирования нужно осветить специфические стороны технологических процессов конкретного предприятия, то VPwin позволяет переключиться на любой вид модели созданный с помощью IDEF 3 или DFD или создать смешанную модель. Совместная работа в трех нотациях позволяет пользователю/аналитику более подробно описать деятельность исследуемого объекта, поскольку одного метода, как правило, бывает недостаточно.

На основе использования указанных выше методологий и нотаций визуального моделирования с помощью VPwin можно построить модель системы в виде иерархически упорядоченных диаграмм, которая может работать не только сегодня, но и в случае необходимости быстро и эффективно скорректировать систему в соответствии с новыми условиями. Посредством набора графических инструментов для отображения действий и объектов, VPwin позволяет легко построить схему процесса, на которой показаны исходные данные, результаты операций, ресурсы, необходимые для их выполнения, управляющие воздействия, взаимные связи между отдельными работами. Интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели.

Модель, созданная средствами VPwin, позволяет четко документировать различные аспекты деятельности - действия, которые необходимо предпринять, способы их осуществления, требующиеся для этого ресурсы и др. Модели VPwin дают основу для осмысления бизнес-процессов и оценки влияния тех или иных событий, а также описывают взаимодействие процессов и потоков информации в организации. Неэффективная, высокочатратная или избыточная деятельность может быть легко выявлена и, следовательно, усовершенствована, изменена или устранена в соответствии с общими целями организации.

- поддерживает три стандартные нотации: IDEF0, DFD, IDEF3, что обеспечивает комплексное описание предметной области
- обладает редакторами для описания операций/функций и связей
- имеет модуль функционально-стоимостного анализа **ABC** (ФСА) расчета затрат на выполнение исследуемого процесса. VPwin полностью поддерживает методы расчета себестоимости по всему объему хозяйственной деятельности
- обеспечивает создание структуры диаграмм, облегчающих последовательное уточнение элементов модели

- позволяет разрабатывать диаграммы:
 - контекстные - для описания границ системы, области действия, назначения объектов
 - декомпозиции - для описания особенностей взаимодействия различных процессов;
 - диаграммы **FEO** (For Exposition Only) для создания вариации модели или проблемной области, позволяющих произвести анализ вариантов, не внося изменений в основную модель;
 - диаграммы **Swim Lane** - для координации сложных процессов и функциональных ограничений, позволяя вам видеть процессы, роли и обязанности во всем их многообразии;
 - организационные. Организационные структуры оказывают огромное влияние на определение и выполнение процессов. VPwin поддерживает явное определение ролей, а это определяет и категоризирует задачи или работы, составляющие процессы. Основываясь на ролях, определенных пользователем, VPwin формирует организационные диаграммы;
- имеет расширенные возможности:
 - по представлению диаграмм. Графическое представление модели может быть изображено при помощи различных цветов, шрифтов и иных параметров представления, которые выделяют важные или, наоборот, тушируют незначительные аспекты модели. Такая возможность является ключевой при представлении и обсуждении модели с заказчиком или экспертами предметной области, т.к. правильно подобранное графическое представление позволяет им быстрее сориентироваться в модели. Различные варианты оформления с гибким использованием шрифтов, цвета и других средств форматирования придают документам большую наглядность;
 - по обеспечению логической четкости в определении и описании элементов диаграмм, необходимой для достижения корректных и согласованных результатов;
 - по проверке создаваемых моделей с точки зрения синтаксиса выбранной методологии;
 - по проверке ссылочной целостности, не допуская определения некорректных связей и гарантируя непротиворечивость отношений между объектами при моделировании между диаграммами. VPwin отслеживает связи в диаграммах при внесении изменений в модель. Динамическая "подсветка" объектов служит подсказкой при построении модели и предостерегает от повторения распространенных ошибок в моделировании.

VRwin обеспечивает коррекцию наиболее часто встречающихся ошибок при моделировании, таких, как "зависание" связей при переходе от диаграммы к диаграмме, нарушение ассоциации связей в различных диаграммах модели и т.п.;

- по поддержке пользовательские свойств, которые применяются к элементам диаграммы для описания специфических свойств, присущих данному элементу;

- выполняет ряд других проверок для создания правильной модели, а не просто рисунка (при этом сохраняются главные преимущества рисунка – простота создания и наглядность);

- обладает средствами объединения для параллельной разработки моделей и разграничения процессов. Модели некоторых процессов в масштабах всего предприятия могут оказаться очень сложными. VRwin предоставляет возможности, позволяющие нескольким проектным группам проводить анализ различных фрагментов деятельности, а затем создать глобальное представление. Иногда бывает необходимо более детально изучить определенную часть общей модели. VRwin позволяет разбить модель на фрагменты, поработать с ними, а затем вновь объединить их в одно целое;

- осуществляет экспорт моделей в средства имитационного моделирования (**Arena**). Имитационное моделирование – это создание компьютерной модели системы (физической, технологической, финансовой и т. п.) и проведение на ней экспериментов с целью наблюдения/предсказания в тех случаях, когда реальный (натурный) эксперимент проводить дорого, а зачастую опасно или невозможно. Использование имеющегося интерфейса к имитационному ПО позволяет использовать готовые модели для изучения изменяющегося во времени (динамического) взаимодействия бизнес-процессов. Распределение ресурсов и потоки могут быть оптимизированы для достижения эффективной загрузки. Имитационное моделирование позволяет в динамике проанализировать воздействие изменений. Прежде чем эти изменения будут произведены, можно проверить различные сценарии и обеспечить тем самым принятие оптимального решения.

- обеспечивает интеграцию и связь со средством проектирования баз данных **ERwin** (методология IDEF1X), которые позволяют сократить время проектирования и разработки сложных ИС. Интеграция VRwin с инструментом проектирования БД создавать комплексные системы, в которых ERwin служит для описания информационных объектов системы, а VRwin отражает

функциональные особенности предметной области. Механизмы экспорта-импорта позволяют синхронизировать модели на различных этапах разработки ИС. Хорошо спроектированная модель процессов является не только фундаментом для построения концепции ИС, но также и основой для создания структуры данных приложения. Связывая сущности и атрибуты модели данных с информацией о выполняемых действиях можно продолжить анализ процессов на новом уровне с одновременной перекрестной проверкой моделей процессов и данных;

- обеспечивает интеграцию и связь с **ModelMart**, поддерживающим мощный набор инструментальных программных средств, обеспечивающих совместное (групповое) проектирование и разработку программных систем, включая механизмы объединения моделей и анализа изменений, контроль версий, возможность создания "компонент" модели и т.д.;

- обладает удобным интерфейсом пользователя;

- обеспечивает легкую навигацию по диаграммам;

- поддерживает свойства, задаваемые пользователем, что позволяет производить расширенное описание моделей, включая мультимедийные документы;

- поддерживает автоматическую настройку размеров диаграмм и возможность изменения масштабов изображения моделей;

- содержит собственный генератор отчетов, который может создавать отчеты в формате MS Excel и Word для последующей обработки и использования в других приложениях. Инструмент отчетов **Report Template Builder** позволяет создавать различные отчеты о модели. С его помощью можно также создавать шаблоны для отчетов, которые можно многократно использовать, а также преобразовывать отчеты в формат txt, HTML или RTF;

- пользователь может просматривать и распечатывать общее представление своей модели в виде древовидных диаграмм. Возможности настройки пользовательских палитр цветов позволяют легко адаптировать вид документов в соответствии с особенностями принтера или демонстрационного проектора без внесения изменений в саму модель.;

- позволяет облегчить сертификацию на соответствие стандартам качества ISO9000

-

2 ПРАКТИЧЕСКИЕ ЗАДАНИЯ

2.1 Упражнение 1. Создание контекстной диаграммы

CASE -средство верхнего уровня BPwin – это инструмент визуального моделирования ИС.

BPwin входит в семейство продуктов AllFusion компании Computer Associates под именем AllFusion Process Modeler и предназначен для поддержки всех стадий жизненного цикла разработки ИС.

В линейку продуктов AllFusion Modeling Suite кроме BPwin для поддержки всех стадий разработки программного обеспечения, входят CASE-средств ERwin , BPwin , ModelMart , Paradigm Plus , ERwin Examiner и средства управления проектами.

BPwin соответствует требованиям к инструментам для разработки ИС, так как позволяет четко документировать различные действия, которые необходимо предпринять, а также способы их осуществления и требуемые для этого ресурсы. BPwin является интуитивно понятным визуальным инструментом, позволяющим сформировать целостную картину деятельности предприятия: от моделей организации работы в маленьких отделах до сложных иерархических структур. В руках системных аналитиков и разработчиков BPwin - мощное средство моделирования процессов при создании корпоративных информационных систем (КИС).

В первую очередь BPwin предназначен для системного аналитика на первом этапе разработки системы, т.к. является средством для сбора всей необходимой информации о работе предприятия и графического изображения этой информации в визуальной модели (рисунке).

BPwin автоматизирует задачи, связанные с построением моделей, обеспечивая семантическую строгость, необходимую для гарантирования правильности результатов, а также целостность и непротиворечивость модели, которые гарантируются применением методологий IDEF0, IDEF1X (Data Flow Diagram) и

IDEF3 (Work Flow Diagram). Каждая из этих трех нотаций, поддерживаемых в BPwin, позволяет рассмотреть различные стороны деятельности предприятия и комплексно описать предметную область с трех различных, но взаимосвязанных точек зрения:

- функциональности системы. В рамках методологии IDEF0 (Integration Definition for Function Modeling) процесс представляется в виде набора работ (функций), которые взаимодействуют между собой, а также показывается информационные, людские и производственные ресурсы, потребляемые каждой работой. С помощью функционального моделирования можно провести систематический анализ процессов, сосредоточившись на регулярно решаемых задачах/функциях, свидетельствующих об их правильном выполнении показателей, необходимых для этого ресурсах, результатах и исходных материалах/сырья. Правильно построенная функциональная модель адекватна на всех уровнях абстрагирования;

- последовательности выполняемых работ. Более точную картину можно получить, дополнив модель диаграммами IDEF3. Моделирование потоков работ позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход. Этот метод привлекает внимание к очередности выполнения событий. В IDEF3 включены элементы логики, что позволяет моделировать и анализировать альтернативные сценарии развития процесса.

- потоков информации (документооборота) в системе. Диаграммы DFD (Data Flow Diagramming) могут дополнить то, что уже отражено в модели IDEF3, поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией между функциями внутри системы. В тоже время диаграммы DFD оставляют без внимания взаимодействие между функциями. Моделирование потоков данных часто используется при разработке программного обеспечения, сосредоточенного вокруг потоков данных, передающихся между различными операциями, включая их хранение, для достижения максимальной доступности и минимального времени ответа. Такое моделирование позволяет рассмотреть конкретный процесс, проанализировать операции, из которых он состоит, а также точки принятия решений, влияющих на его ход.

В качестве примера рассматривается деятельность вымышленной компании. Компания занимается в основном сборкой и продажей настольных компьютеров и ноутбуков. Компания не производит компоненты самостоятельно, а только собирает и тестирует компьютеры.

Основные процедуры в компании таковы:

- продавцы принимают заказы клиентов;
- операторы группируют заказы по типам компьютеров;
- операторы собирают и тестируют компьютеры;
- операторы упаковывают компьютеры согласно заказам;
- кладовщик отгружает клиентам заказы.

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

1. Запустите программу BPwin


2. В открывшемся при запуске окне внесите:

Name (*Имя модели*) - "*Деятельность компаний*"

Type – *IDFO*.

Нажмите .

3. Автоматически создается контекстная диаграмма.

4. Обратите внимание на кнопку  на панели инструментов. Эта кнопка включает и выключает *инструмент просмотра и навигации* –

Model Explorer (*появляется слева*).

Model Explorer имеет три вкладки - *Activities, Diagrams* и *Objects*.

Во вкладке *Activities* щелчок правой кнопкой по объекту позволяет редактировать его свойства.

5. Если вам непонятно, как выполнить то или иное действие, вы можете вызвать помощь - клавиша или меню *Help*.

6. Перейдите в меню Model / Model Properties. Во вкладке *General* диалога Model Properties следует внести:

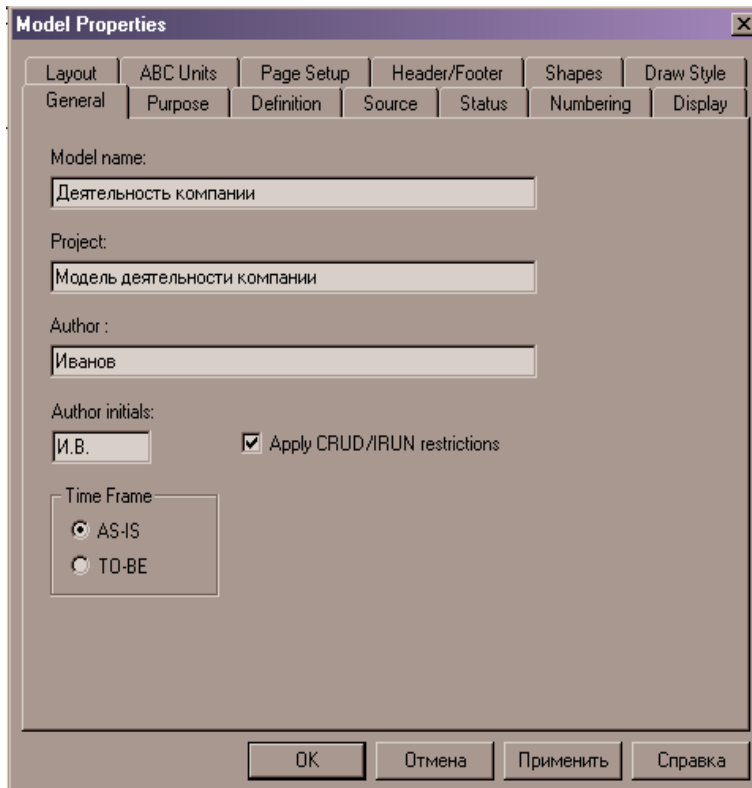
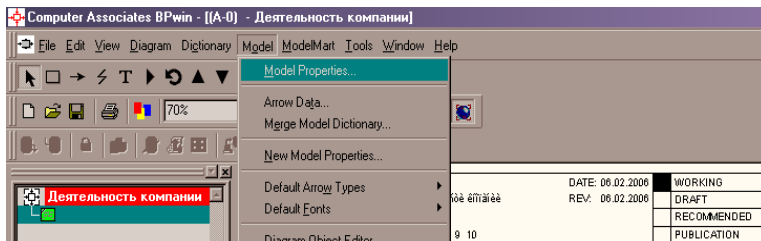
Name (*имя модели*): "*Деятельность компаний*",

Project (*имя проекта*): "*Модель деятельности компаний*",

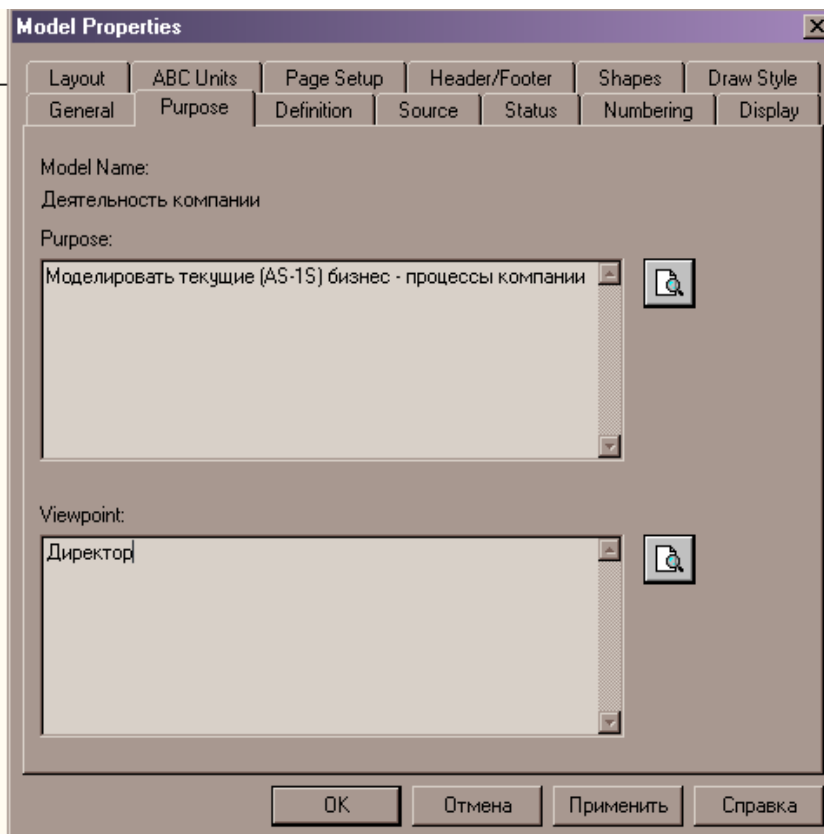
Author (*имя автора*): *Фамилия*

Time Frame (*тип модели*): *AS-1S (Как есть)*.

Информационные системы в строительстве



7. Во вкладке **Purpose** внесите:
 цель **Purpose: Моделировать текущие (AS-IS) бизнес - процессы компании**
 точку зрения **Viewpoint: Директор.**

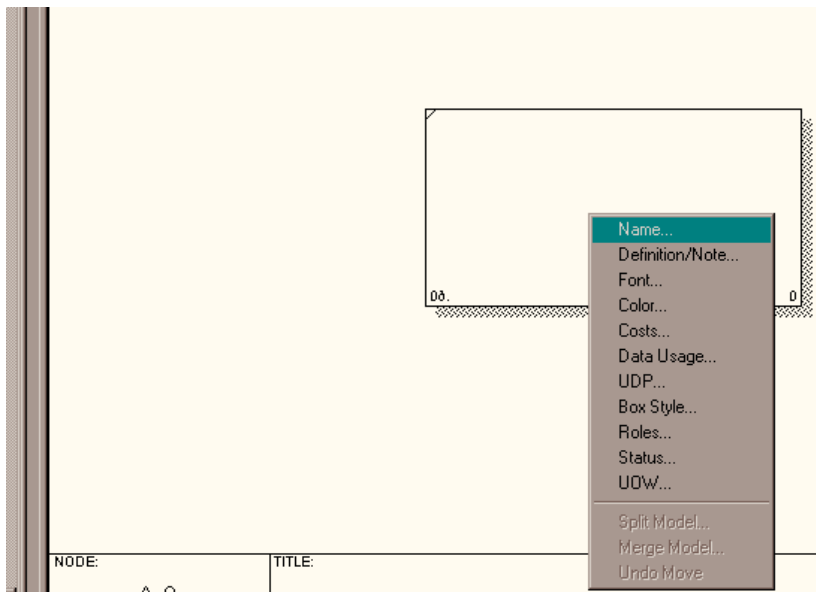


8. Во вкладке **Definition** внесите:

Definition (*Определение*) **"Это учебная модель, описывающая деятельность компании"**

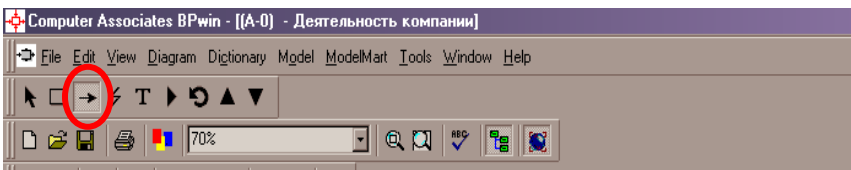
Score (*Цель*): **Общее управление бизнесом компании: исследование рынка, закупка компонентов, сборка, тестирование и продажа продуктов.**

9. Перейдите на контекстную диаграмму и **правой кнопкой мыши** щелкните по **работе**. В открывшемся контекстном меню выберите **Name**.

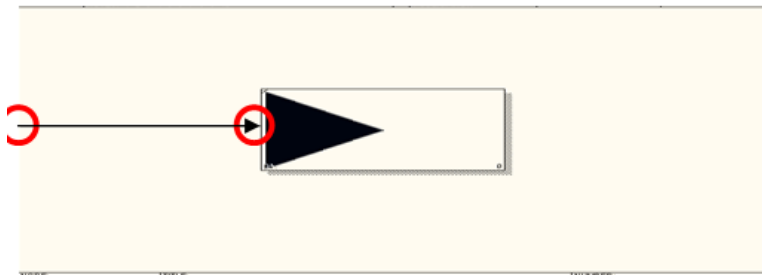


10. В открывшейся форме **Activity Properties**:
 во вкладке **Name** внесите имя - **Деятельность компании**
 во вкладке **Definition** внесите определение – **Текущие бизнес - процессы компании**

1. Создайте стрелки на контекстной диаграмме (см.табл.1).



Для этого используйте кнопку на панели инструментов далее, щелкните левой кнопкой мыши на левой границе окна и отпустив мышью протяните линию стрелки к правой стороне прямоугольника **Работы** и щелкните левой кнопкой мыши в точке соединения стрелки с левой стороной прямоугольника



Для **удаления** стрелки или другого объекта используйте **кнопку Pointer Tool** и далее нажимайте **Delete** на клавиатуре



те клави-
ре

Правильно ис-
ние стрелок обеспечи-

пользова-
вает

единство представления информации в диаграммах любого типа. По команде **Model / Default Arrow Types** производится задание стиля для новых стрелок данной модели, как стиль стрелок, устанавливаемый по умолчанию. Изменение стиля может производиться с помощью опций **Style** диалогового окна **Arrow Properties**. Стиль стрелки «по умолчанию» может быть изменен в любое время с помощью опций кнопки **Arrow tool**.

В VPwin используются следующие типы стрелок:

- **Precedence** – стрелка изображаемая сплошной линией. Это наиболее часто используемый тип стрелки для указания связи между двумя объектами диаграммы в любом направлении, в зависимости от методологии, по которой выполнена диаграмма

- **Relational** - стрелка изображаемая пунктирной линией. Используется для соединения ссылок в UOW.

- **Object Flow** –связь изображаемая сплошной линией с двойной стрелкой;

- **Bidirectional** – двунаправленная

стрелка указывающая на связь двух объектов только для потоко-



вых диаграмм

Щелчком правой кнопки мыши по стрелке открывается контекстное меню стрелки, содержащее следующие опции:

- **Name, Definition/Note, Font, Style, Color, UDP, Arrow Data, Status, UOW** – открытие диалогового окна **Arrow Properties** на соответствующей вкладке

- **Squiggle** – создание зигзага, соединяющего стрелку с ее названием

- **Trim** – уменьшение длины стрелки

- **Arrow Tunnel** – создание туннельной стрелки

- **Off Page Reference** – создание стрелки межстраничных

ССЫЛОК

- **External Reference** – внешняя ссылка

- **External Definition** – внешнее описание

- **Go to Reference** – переход по ссылке

В IDEF0 различают следующие виды стрелок.

Вход (Input) – материал или информация, которые используются работой для получения результата (выхода). Стрелка входа рисуется как входящая в левую грань работы. Вход – это нечто, что преобразуется/ изменяется работой.

Управление (Control) – правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Стрелка управления рисуется как входящая в верхнюю грань работы. Управление влияет на работу, но не преобразуется работой. Каждая работа должна иметь хотя бы одну стрелку управления.

Выход (Output) – материал, или информация, которые производятся работой. Стрелка рисуется как исходящая из правой грани работы. 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. стрелка механизма рисуется как входящая в нижнюю грань работы.

Информационные системы в строительстве

Стрелки контекстной диаграммы

<i>Arrow NAME</i>	<i>Arrow DEFINITION</i>	<i>Arrow Type</i>
Бухгалтерская система	Оформление счетов, оплата счетов, работа с заказами	Mechanism
Звонки клиентов	Запросы информации, заказы, техподдержка и т. д.	Input
Правила и процедуры	Правила продаж, инструкции по сборке, процедуры тестирования, критерии производительности и т. д.	Control
Проданные продукты	Настольные и портативные компьютеры	Output

2. С помощью кнопки **T** создайте текстовые поля и внесите в них текст в поле диаграммы - точку зрения и цель (рис. 1).

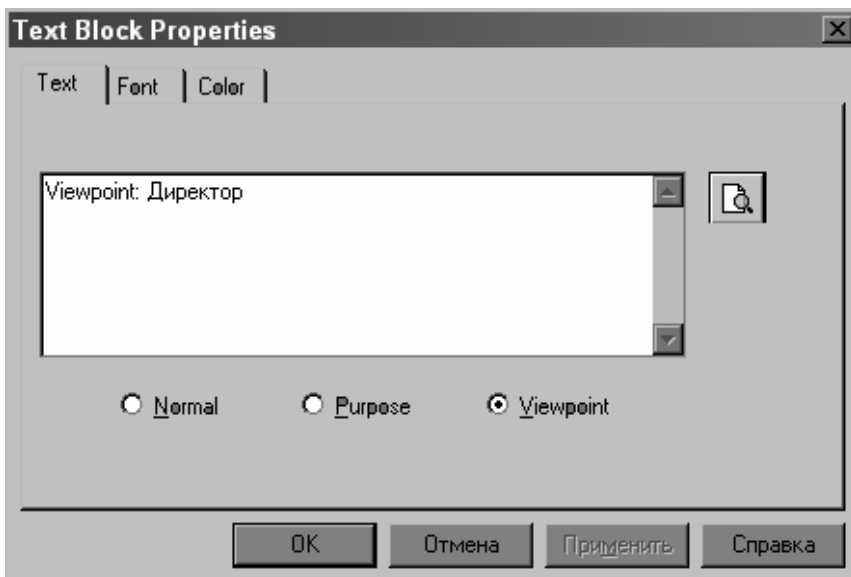


Рис. 1. Внесение текста в поле диаграммы

Результат выполнения упражнения 1 показан на рис. 2.

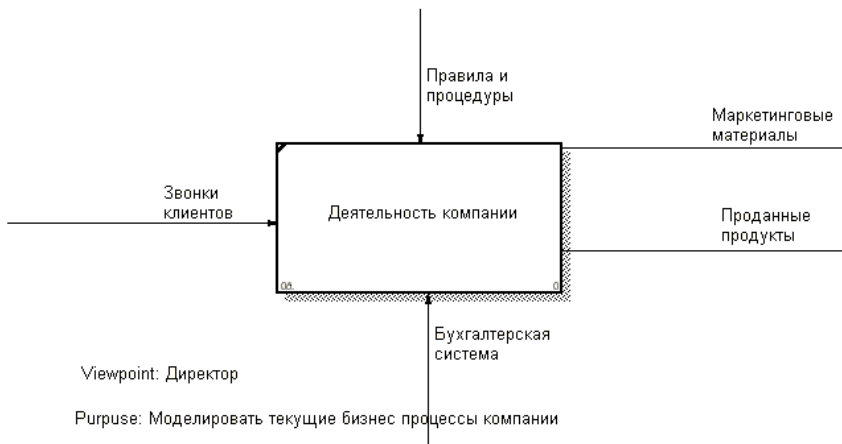


Рис. 2. Контекстная диаграмма

13. Создайте отчет по модели. Меню **Tools/Reports/Model Report** (рис. 3).

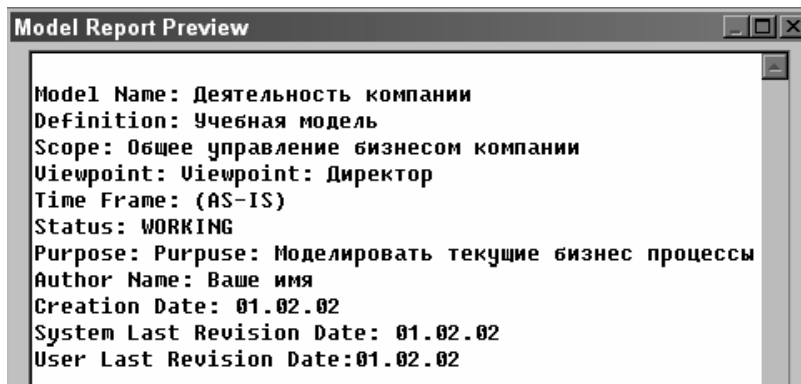



Рис. 3. Отчет

2.2 Упражнение 2. Создание диаграммы декомпозиции A0



1. На панели инструментов выберите кнопку перехода на нижний уровень . В диалоге Activity Box Count установите число работ на диаграмме нижнего уровня – 3. Нажмите **OK**.

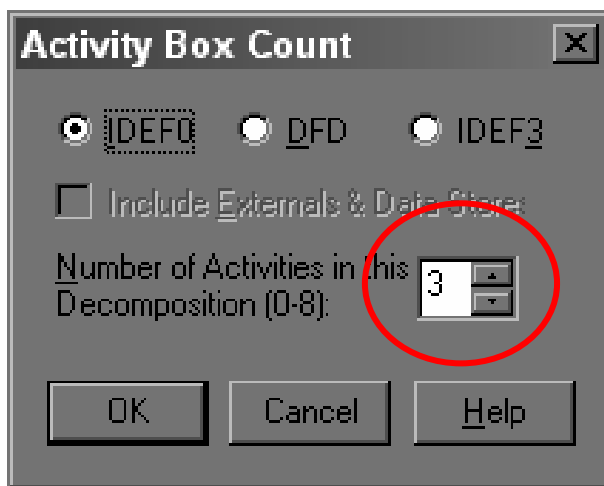


Рис. 1. Диалог Activity Box Count

Автоматически будет создана диаграмма декомпозиции. Правой кнопкой мыши щелкните по работе, выберите Name и внесите имя работы. Повторите операцию для всех трех работ. Затем внесите определение, статус и источник для каждой работы

согласно табл. 1.

Таблица 1. Работы диаграммы декомпозиции A0

Activity Name	Definition
Продажи и маркетинг	Телемаркетинг и презентации, выстав-
Сборка и тестирование ком- пьютеров	Сборка и тестирование настольных и портативных компьютеров
Отгрузка и получение	Отгрузка заказов клиентам и получение компонентов от поставщиков

2. Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем работ. Вызов словаря - меню Dictionary/Activity (рис.2).

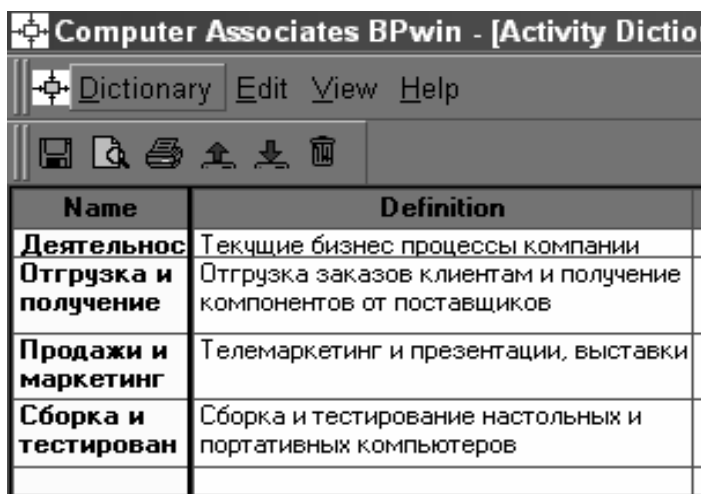





Рис. 2. Словарь Dictionary/Activity

Если описать имя и свойства работы в словаре, ее можно

будет внести в диаграмму позже с помощью кнопки  панели инструментов.

!!! Невозможно удалить работу из словаря, если она используется на какой-либо диаграмме. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для добавления работы в словарь необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства работы. Для удаления всех имен работ, не использующихся в модели, щелкните по кнопке 

3. Перейдите в режим рисования стрелок. Свяжите граничные стрелки (кнопка  на палитре инструментов так, как показано на рис. 3).

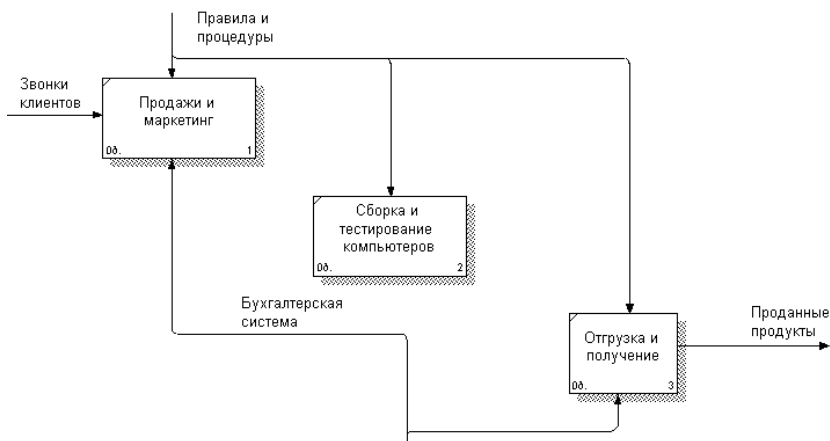


Рис. 3. Связанные граничные стрелки на диаграмме А0

1. Правой кнопкой мыши щелкните по ветви стрелки управления работы "Сборка и тестирование компьютеров" и переименуйте ее в Name - "Правила сборки и тестирования" (рис.4).
- 2.

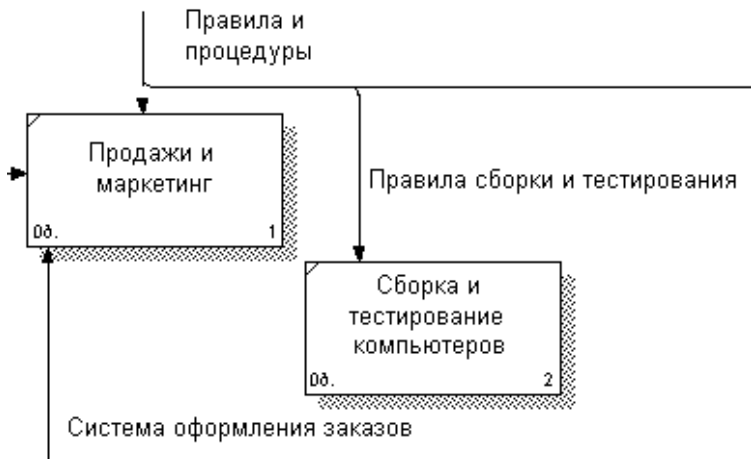


Рис. 4. Стрелка "Правила сборки и тестирования"

5. Внесите определение для новой ветви Definition: "Инструкции по сборке, процедуры тестирования, критерии производительности и т. д." Правой кнопкой мыши щелкните по ветви стрелки механизма работы "Продажи и маркетинг" и переименуйте ее в "Систему оформления заказов".

!!! Альтернативный метод внесения имен и свойств стрелок - использование словаря стрелок (вызов словаря - меню Dictionary/Arrow). Если внести имя и свойства стрелки в словарь, ее можно будет внести в диаграмму позже. Стрелку нельзя удалить из словаря, если она используется на какой-либо диаграмме. Если удалить стрелку из диаграммы, из словаря она не удаляется. Имя и описание такой стрелки может быть использовано в дальнейшем. Для добавления стрелки необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства стрелки.

Computer Associates BPwin - [Arrow Dictionary]		
Dictionary Edit View Help		
Name	Definition	Status
Бухгалтерская система	Оформление счетов, оплата счетов, работа с заказами	WORKIN
Звонки клиентов	Запросы информации, заказы, техподдержка	WORKIN
Маркетинговые материалы		WORKIN
Правила и процедуры	Правила продаж, инструкции по сборке, процедуры тестирования, кри	WORKIN
Правила сборки и тестирование	Инструкции по сборке, процедуры тестирования, критерии производ	WORKIN
Проданные продукты	Настольные и портативные компьютеры	WORKIN

6. Создайте новые внутренние стрелки так, как показано на рис. 5.

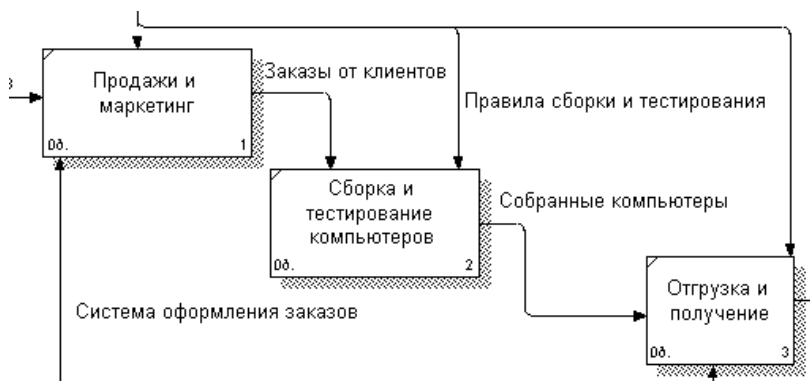


Рис. 5. Внутренние стрелки диаграммы АО

7. Создайте стрелку обратной связи (по управлению) "Результаты сборки и тестирования", идущую от работы "Сборка и тестирование компьютеров" к работе "Продажи и маркетинг". Измените стиль стрелки (толщина линий) и установите опцию Extra Arrowhead (из контекстного меню). Методом drag&drop перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите Squiggle (из контекстного меню). Результат изменений показан на рис. 6.

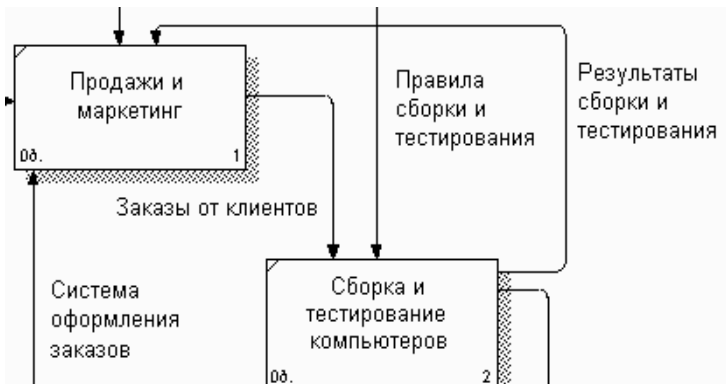


Рис. 6. Результат редактирования стрелок на диаграмме А0

8. Создайте новую граничную стрелку выхода "Маркетинговые материалы", выходящую из работы "Продажи и маркетинг". Эта стрелка автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на наконечнике. Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню Arrow Tunnel. В диалог Border Arrow Editor выберите опцию Resolve it to Border Arrow. Для стрелки "Маркетинговые материалы" выберите опцию Trim из контекстного меню. Результат выполнения упражнения 2 показан на рис. 7.



Рис. 7. Результат выполнения упражнения 2 - диаграмма А0.

2.3 Упражнение 3. Создание диаграммы декомпозиции A2

В результате проведения экспертизы получена следующая информация:

1. Производственный отдел получает заказы клиентов от отдела продаж по мере их поступления.

2. Диспетчер координирует работу сборщиков, сортирует заказы, группирует их и дает указание на отгрузку компьютеров, когда они готовы.

3. Каждые 2 часа диспетчер группирует заказы - отдельно для настольных компьютеров и ноутбуков - и направляет на участок сборки.

4. Сотрудники участка сборки собирают компьютеры согласно спецификациям заказа и инструкциям по сборке. Когда группа компьютеров, соответствующая группе заказов, собрана, она направляется на тестирование.

5. Тестировщики тестируют каждый компьютер и в случае необходимости заменяют неисправные компоненты.

6. Тестировщики направляют результаты тестирования диспетчеру, который на основании этой информации принимает решение о передаче компьютеров, соответствующих группе заказов, на отгрузку.

Задачи:

1. **Декомпозируем** работу "Сборка и тестирование компьютеров" на 4 блока (табл. 1).

2. На основе этой информации вносим **новые работы и стрелки** (табл. 1, 2).

3. **Туннелируем и связываем** на верхнем уровне граничные стрелки, если это необходимо.

4. **Отформатировать** все блоки диаграммы.

Таблица 1. Работы диаграммы декомпозиции A2

<i>Activity Name</i>	<i>Activity Definition</i>
Отслеживание расписания и управление сборкой и тестированием	Просмотр заказов, установка расписания выполнения заказов, просмотр результатов тестирования, формирование групп заказов на сборку и отгрузку

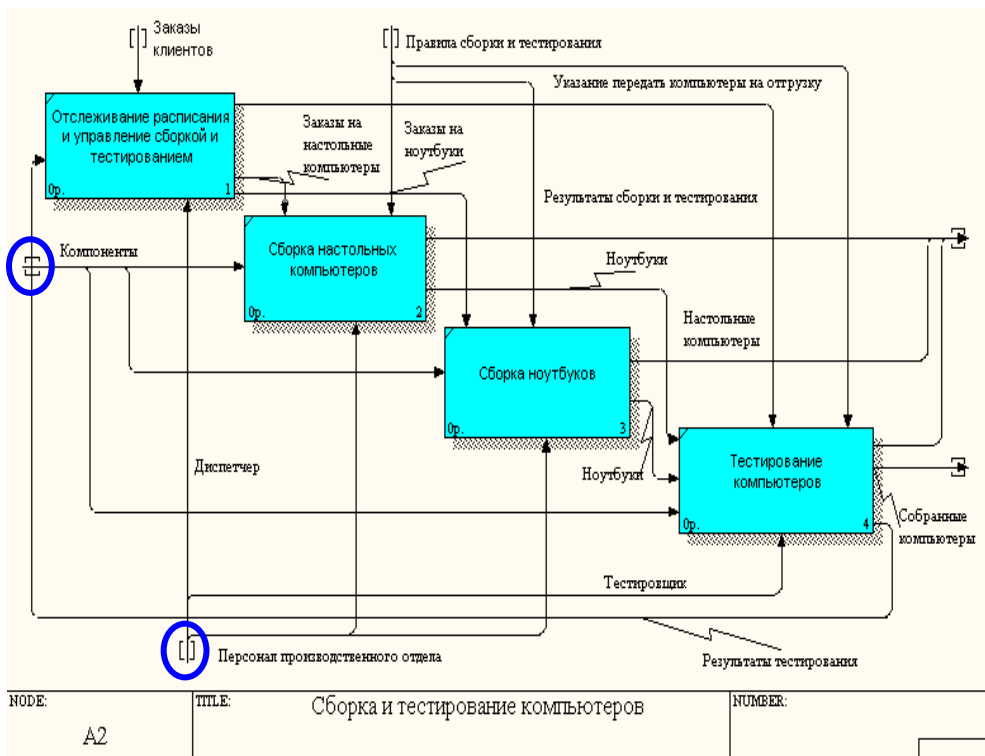
Сборка настольных компьютеров	Сборка настольных компьютеров в соответствии с инструкциями и указаниями диспетчера
Сборка ноутбуков	Сборка ноутбуков в соответствии с инструкциями и указаниями диспетчера
Тестирование компьютеров	Тестирование компьютеров и компонентов. Замена неработающих компонентов

Таблица 2. Стрелки диаграммы декомпозиции **A2**

<i>ARROW Nate</i>	<i>ARROW Source</i>	<i>ARROW Source Type</i>	<i>ARROW DEST.</i>	<i>ARROW DEST.Type</i>
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	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

Результат выполнения упражнения 3



позиции нижнего уровня изображаются в квадратных скобках и автоматически не появляются на диаграмме верхнего уровня. Для их перетаскивания наверх необходимо щелкнуть правой кнопкой мыши **по квадратным скобкам** и выбрать пункт меню **Arrow Tunnel**, а в диалоге **Border Arrow Editor** выбрать опцию **“Resolve it to Border Arrow”**.

Если в диалоге **Border Arrow Editor** установить опцию **“Change it to resolved rounded tunnel”** – стрелка будет затоннелирована и не попадет на другую диаграмму. Тоннельная стрелка изображается с круглыми скобками на конце. Тоннелирование применяется для изображения **малозначимых** стрелок.

После тоннелирования внесенных стрелок на диаграмме декомпозиции A2 (**Компоненты и Персонал**

производственного отдела) они автоматически появятся на диаграмме декомпозиции верхнего уровня (**АО**):

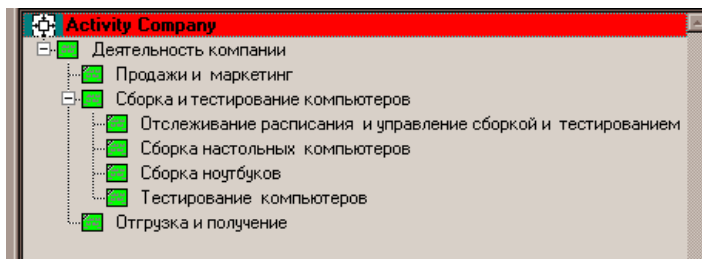
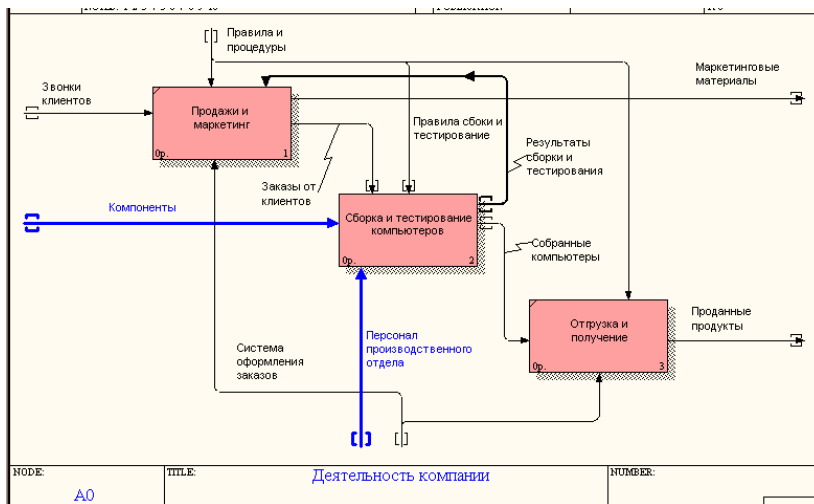
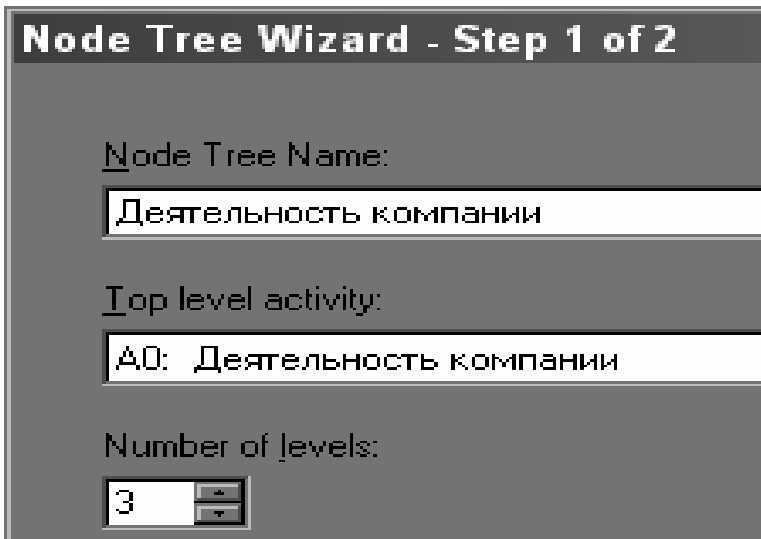


Рис. Структура модели после выполнения упражнения 3

2.4 Упражнение 4. Создание диаграммы узлов

1. Выберите меню **Diagram/Add Node Tree**. В первом диалоге гйда Node Tree Wizard внесите имя диаграммы, укажите диаграмму корня дерева и количество уровней (рис.1).



Node Tree Wizard - Step 1 of 2

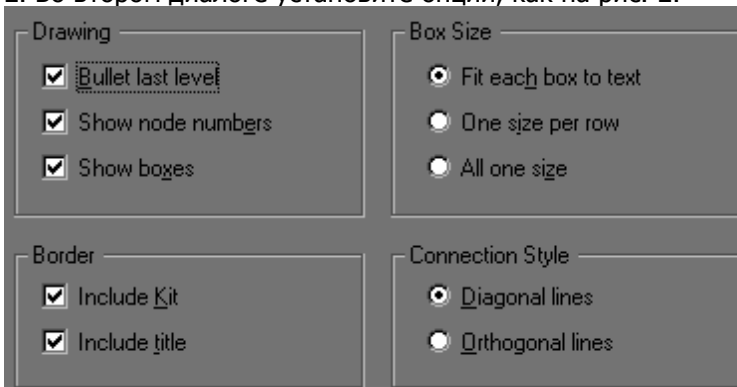
Node Tree Name:
Деятельность компании

Top level activity:
АО: Деятельность компании

Number of levels:
3

Рис. 1. Первый диалог Node Tree Wizard

2. Во втором диалоге установите опции, как на рис. 2.



Drawing

- Bullet last level
- Show node numbers
- Show boxes

Box Size

- Fit each box to text
- One size per row
- All one size

Border

- Include Kit
- Include title

Connection Style

- Diagonal lines
- Orthogonal lines

Рис. 2. Второй диалог Node Tree Wizard

3. Щелкните по **Finish**. Создается диаграмма дерева узлов. Результат можно посмотреть на рис. 3.

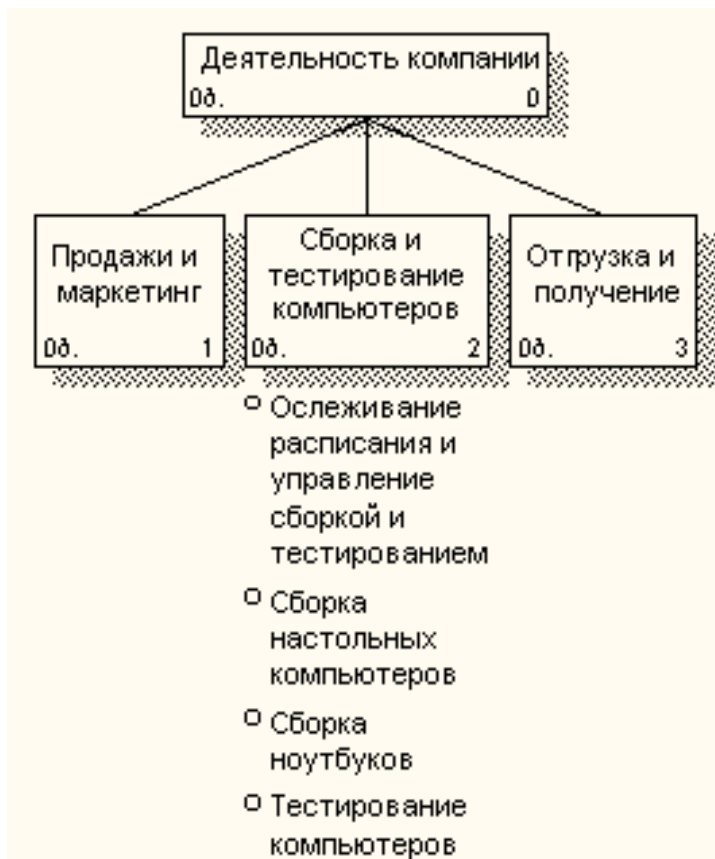


Рис. 3. Диаграмма дерева узлов

4. Диаграмму дерева узлов можно модифицировать. Нижний уровень может быть отображен не в виде списка, а в виде прямоугольников, так же как и верхние уровни.

Для модификации диаграммы правой кнопкой мыши щелкните по свободному месту, не занятому объектами, выберите меню **Node Tree Diagram Properties** и во вкладке **Style** диалога **Node Tree Properties** отключите опцию **Bullet Last Level**. Щелкните по **ОК**.

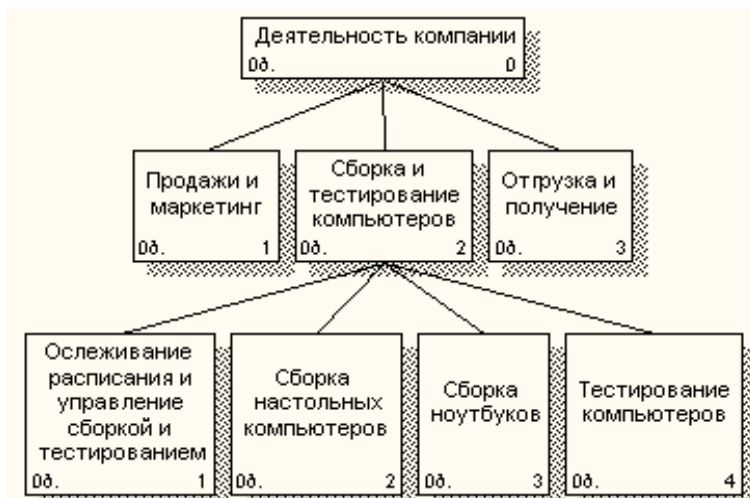


Рис. 4. Результат выполнения упражнения 4

2.5 Упражнение 5. Создание FEO диаграммы

Предположим, что при обсуждении бизнес - процессов возникла необходимость детально рассмотреть взаимодействие работы **"Сборка и тестирование компьютеров"** с другими работами. Чтобы не портить диаграмму декомпозиции, создайте FEO - диаграмму, на которой будут только стрелки работы **"Сборка и тестирование компьютеров"**.

FEO-ДИАГРАММА – это диаграмма-иллюстрация отдельных фрагментов модели и/или для иллюстрации альтернативной точки зрения, либо для специальных целей, которые не поддерживаются явно синтаксисом IDEFO. Это графическое описание, используемое, для сообщения специфических фактов о диаграмме IDEFO.

В любое время работы над диаграммой пользователь может добавлять диаграмму «только для экспозиции» (FEO), которая **позволяет** иллюстрировать различные сценарии, показывать различные точки зрения, отображать отдельные детали, которые явно не поддерживаются синтаксисом IDEFO. или подсвечивать иные функциональные особенности, не изменяя созданных ранее диаграмм модели.

Обычно FEO –диаграммы применяются как специальное средство объяснения с различных точек зрения отдельных составных частей проекта или для дополнительного изучения функциональных деталей, которые не получили должного исследова-

ния из-за особенностей синтаксиса IDEF0.

При **создании диаграмм** экспозиции FEO допускается нарушать синтаксические правила IDEF0, поскольку они являются просто копиями стандартных диаграмм и не включаются в анализ синтаксиса. Например, функция на диаграмме FEO может не иметь стрелок управления и выхода. С целью обсуждения определенных аспектов модели с экспертом предметной области может быть создана диаграмма только с одной функцией и одной стрелкой, поскольку стандартная диаграмма декомпозиции содержит множество деталей, не относящихся к теме обсуждения и дезориентирующих эксперта. Но если **FEO** используется для иллюстрации альтернативных точек зрения (альтернативный контекст), то рекомендуется все-таки придерживаться синтаксиса **IDEFO**.

В контекстной диаграмме или диаграммах декомпозиции могут существовать неограниченное количество FEO-диаграмм. Диаграммы этого типа могут сопровождаться различного рода текстовыми примечаниями, синтаксис которых также может не соответствовать правилам методологии IDEF0.

FEO-диаграмма внешне выглядит также, как и оригинальная диаграмма, за исключением имени, которое присваивается FEO-диаграмме. Например, если FEO-диаграмма была добавлена к диаграмме с номером A1.3, то FEO-диаграмма получит номер A1.3F.

Выполните последовательно следующие действия:

1. Выберите пункт меню **Diagram/Add FEO Diagram**.

2. В диалоге **Add New FEO Diagram** выберите:

Source Diagram Name: «**АО Деятельность компании**».

FEO Of: «**Decomposition Diagram**»

Name of New Diagram: «**Взаимодействие работы "Сборка и тестирование компьютеров" с другими работами**»

Щелкните по **ОК**.

3. Для определения диаграммы перейдите в **Diagram/Diagram Properties** и во вкладке **Diagram Text** внесите определение: «**Собранные компьютеры, прошедшие тестирование передаются на отгрузку и получение**».

4. Удалите лишние стрелки на диаграмме **FEO**. Результат показан на рис. 1.

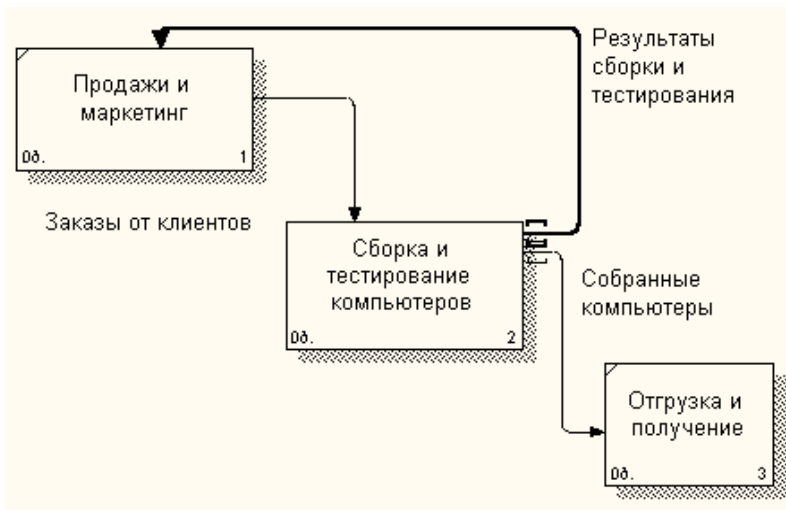


Рис. 1. Диаграмма FEO



Используйте кнопку на палитре инструментов для перехода между

стандартной диаграммой, деревом узлов и FEO

2.6 Упражнение 6. Расщепление и слияние моделей

Возможность слияния и расщепления моделей обеспечивает коллективную работу над проектом. Так, руководитель проекта может создать декомпозицию верхнего уровня и дать задание аналитикам продолжить декомпозицию каждой ветви дерева в виде отдельных моделей. После окончания работы над отдельными ветвями все подмодели могут быть слиты в единую модель. С другой стороны, отдельная ветвь модели может быть отщеплена для использования в качестве независимой модели, для доработки или архивирования.

1. Расщепление (разделение) модели

Расщепление (разделение) моделей

Для отщепления ветви от модели следует щелкнуть правой кнопкой мыши по декомпозированной работе (работа не должна иметь диагональной черты в левом верхнем углу) и выбрать во всплывающем меню пункт Split Model.

В появившемся диалоге Split Options следует указать имя создаваемой модели.

После подтверждения расщепления:
 в старой модели работа станет недекомпозированной
 (признак - диагональная черта в левом верхнем углу),
 будет создана стрелка вызова, причем ее имя будет
 совпадать с именем новой модели,
 будет создана новая модель, причем имя контекстной
 работы будет совпадать с именем работы, от которой была
 "оторвана" декомпозиция.

1. Перейдите на диаграмму **A0**. Правой кнопкой мыши щелкните по работе **"Сборка и тестирование компьютеров"** и выберите из контекстного меню - **Split Model**.

2. В диалоге **Split Option** внесите имя новой модели **"Сборка и тестирование компьютеров"**, установите опции, как на рисунке, и щелкните по **OK** (рис.1).

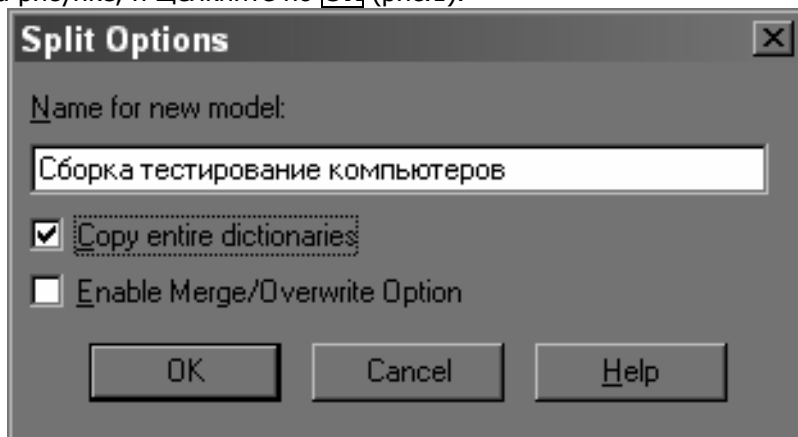
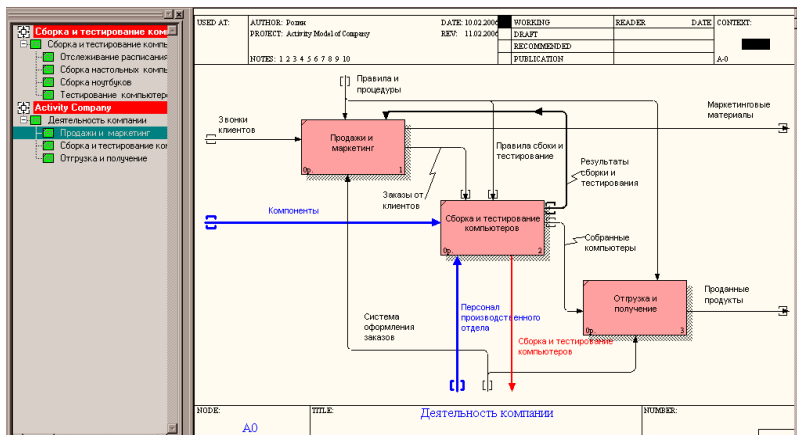
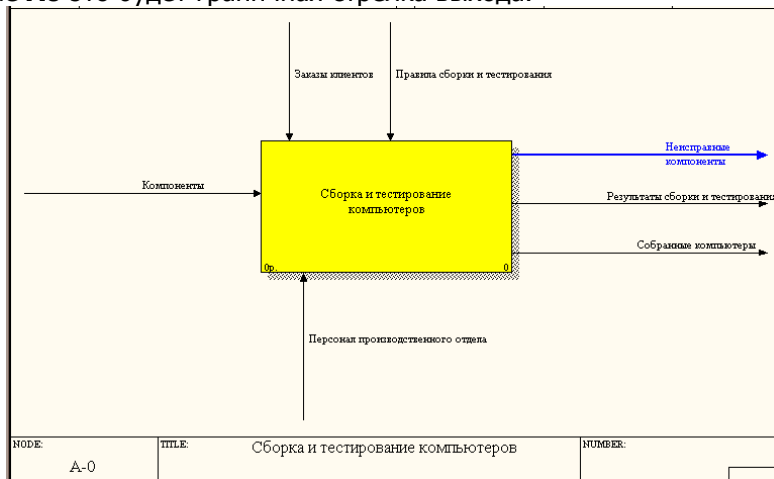


Рис. 1. Диалог **Split Option**

3. Посмотрите на результат: в **Model Explorer** появилась новая модель, а на диаграмме **A0** модели **"Деятельность компании"** появилась стрелка вызова **"Сборка и тестирование компьютеров"**.

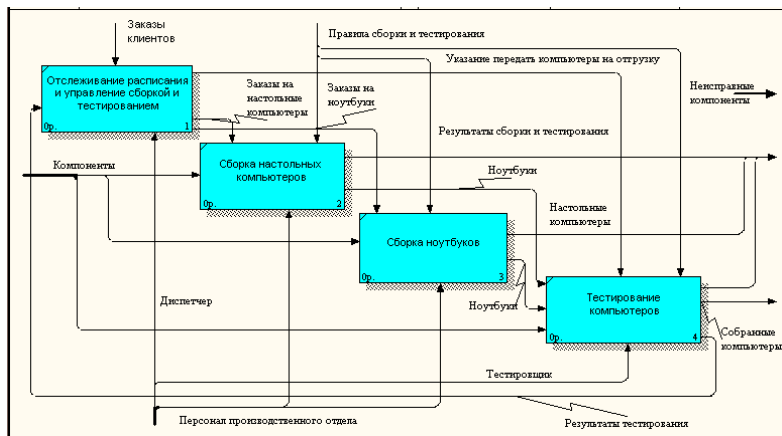


4. Создайте в модели "Сборка и тестирование компьютеров" новую стрелку "Неисправные компоненты". На диаграмме **A0** это будет граничная стрелка выхода.



На диаграмме **A0** - граничная стрелка выхода от работ "Сборка настольных компьютеров", "Тестирование компьютеров" и "Сборка ноутбуков".

Информационные системы в строительстве



2. Слияние модели

ВРwin применяет для слияния и разветвления моделей стрелки вызова.

Для слияния необходимо выполнить следующие условия:

обе сливаемые модели должны быть открыты в ВРwin;
имя модели-источника, которое присоединяют к модели-цели, должно совпадать с именем стрелки вызова работы в модели-цели;

стрелка вызова должна исходить из недекомпозируемой работы (работа должна иметь диагональную черту в левом верхнем углу);

имена контекстной работы подсоединяемой модели-источника и работы на модели-цели, к которой мы подсоединяем модель-источник, должны совпадать;

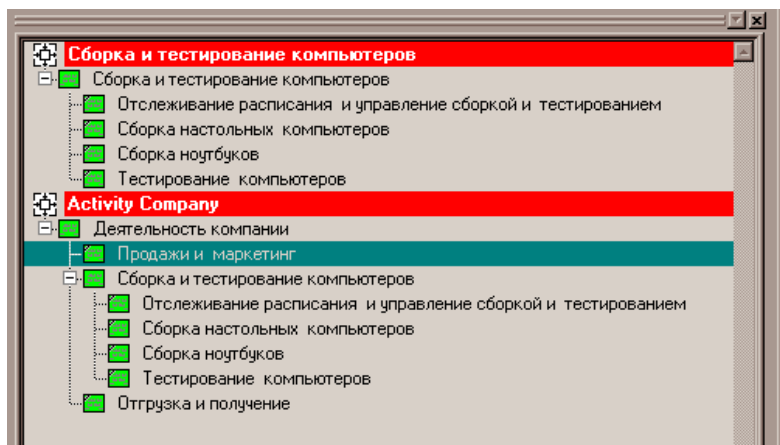
модель-источник должна иметь по крайней мере одну диаграмму декомпозиции.

1. Перейдите на диаграмму **АО** модели **"Деятельность компании"**.

2. **Правой** кнопкой мыши щелкните по работе **"Сборка и тестирование компьютеров"** и выберите **Merge model**.

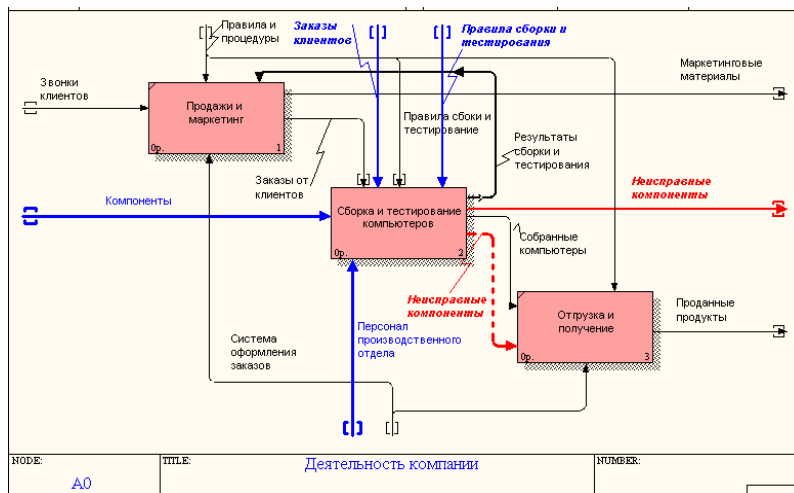
3. В диалоге **Merge Model** включите опцию **Cut/Paste entire dictionaries** и щелкните по **ОК**.

Посмотрите на результат. В **Model Explorer** видно, что **две модели слились**.



Модель "**Сборка и тестирование компьютеров**" осталась и может быть сохранена в отдельном файле.

На диаграмме А0 модели "**Деятельность компании**" исчезла стрелка вызова "**Сборка и тестирование компьютеров**". Появилась неразрешенная граничная стрелка "**Неисправные компоненты**". Направьте эту стрелку к входу работы "**Отгрузка и получение**".



Создание функциональной модели с помощью VPwin

2.7 Упражнение 7. Создание диаграммы IDEF3

Методология IDEF3 была создана для сбора и описания процессов в анализируемой системе.

ПРОЦЕСС - это упорядоченный набор функций, охватывающий различные сущности предприятия и завершающийся глобальной целью. В методологиях IDEF под «процессом» понимается набор операций, которые, взятые вместе, создают результат, имеющий ценность для потребителя.

СТАНДАРТ IDEF3 - это методология сбора данных о процессе, рассматривающая взаимодействие информационных потоков как логическую последовательность выполнения на основе причинно-следственных связей между ситуациями и событиями, предназначенная для разработки структурного представления знаний о системе, и описания изменения состояний объектов, являющихся составной частью описываемых процессов.

При помощи графической нотации IDEF3 описывается логика выполнения работ, очередность их запуска и завершения.

Т.о. IDEF3 предоставляет инструмент для моделирования сценариев действий сотрудников организации, отделов, цехов и т.п., например порядок обработки заказа или события, на которые необходимо реагировать за конечное время, выполнение действий по производству товара им т.д.

IDEF3-ДИАГРАММА (диаграмма потока, process flow diagram, IDEF3-diagram, workflow) - основная единица описания в IDEF3, являющаяся графическим представлением назначения системы или процесса и применяются для анализа завершенности процесса обработки информации (проверка модели ИС на целостность).

Обычно IDEF3-диаграммы являются дополнением к **IDEFO**-диаграммам, т.к. содержат все необходимые сведения для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

С помощью диаграмм IDEF3 можно **анализировать сценарии** из реальной жизни, например, как закрывать магазин в экстренных случаях или какие действия должны выполнить менеджер и продавец при закрытии.

1. Перейдите на диаграмму **A2** и декомпозируйте работу "**Сборка настольных компьютеров**". В диалоге **Activity Box Count** установите: число работ – **4**, нотацию - **IDEF3**.

Возникает диаграмма **IDEF3**, содержащая работы (**UOW**). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню:

щелкните **Name** и внесите имя работы "**Подготовка компонентов**".

щелкните **Definition** и внесите определение "**Подготавливаются все компоненты компьютера согласно спецификации заказа**".

2. Во вкладке **UOW** внесите свойства работы (табл. 1).


Таблица 1. Свойства **UOW**

Objects	Компоненты: винчестеры, корпуса, материнские платы, видеокарты, звуковые карты, дисководы СО-КОМ и флоппи, модемы, программное обеспечение
Facts	Доступные операционные системы: Windows
Constrains	Установка модема требует установки дополнительного программного обеспечения

3. Внесите в диаграмму еще 3 работы, кнопка

Внесите имена работ **NAME**:

- *Установка материнской платы и винчестера;*
- *Установка модема;*
- *Установка дисковода CD-ROM;*
- *Установка флоппи- дисковода;*
- *Инсталляция операционной системы;*
- *Инсталляция дополнительного программного обеспечения.*

4. С помощью кнопки  палитры инструментов создайте **объект ссылки**.

Внесите имя объекта внешней ссылки "**Компоненты**". Свяжите стрелкой объект ссылки и работу "**Подготовка компонент**". Установите стиль линии связи **Style\Referent**.

Объект ссылки в **IDEF3** обозначает данные, блок, которые **нельзя связать со стрелкой или работой**. Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы.

5. Свяжите стрелкой работы "**Подготовка компонентов**" (выход) и "**Установка материнской платы и винчестера**", измените стиль стрелки на **Object Flow**.

В **IDEF3** имя стрелки может отсутствовать.



6. С помощью кнопки на палитре инструментов внесите два перекрестка типа "**асинхронное или**" и свяжите работы с перекрестками.

Окончание одной работы может служить сигналом к началу нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении событий. Различают перекрестки для слияния и разветвления стрелок. Перекресток не может одновременно использоваться для слияния и разветвления. Смысл каждого типа перекрестка приведен в таблице 2. Все перекрестки в диаграмме нумеруются.

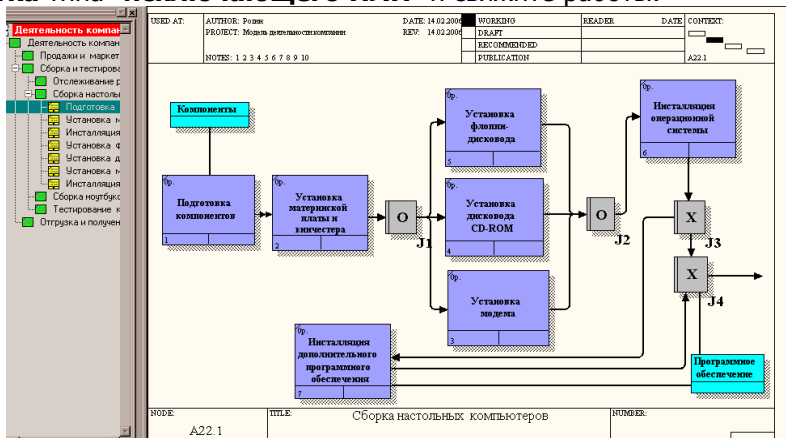
*В **IDEF3** стрелки могут сливаться и разветвляться только через перекрестки.*

Таблица 2.

Наименование	Смысл, в случае слияния стрелок	Смысл, в случае разветвления стрелок
Асинхронное И	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены
Синхронное И	Все предшествующие процессы должны быть завершены одновременно	Все следующие процессы запускаются одновременно
Асинхронное ИЛИ	Один или несколько предшествующих процессов должны быть завершены	Один или несколько следующих процессов должны быть запущены

Синхронное ИЛИ	Один или несколько предшествующих процессов должны быть завершены одновременно	Один или несколько следующих процессов запускаются одновременно
Исключающее ИЛИ	Только один предшествующий процесс завершен	Только один следующий процесс запускается

7. Правой кнопкой щелкните по перекрестку для разветвления (**fan-out**), выберите **Name** и внесите имя "**Компоненты, требуемые в спецификации заказа**". Создайте два перекрестка типа "**исключающего ИЛИ**" и свяжите работы.

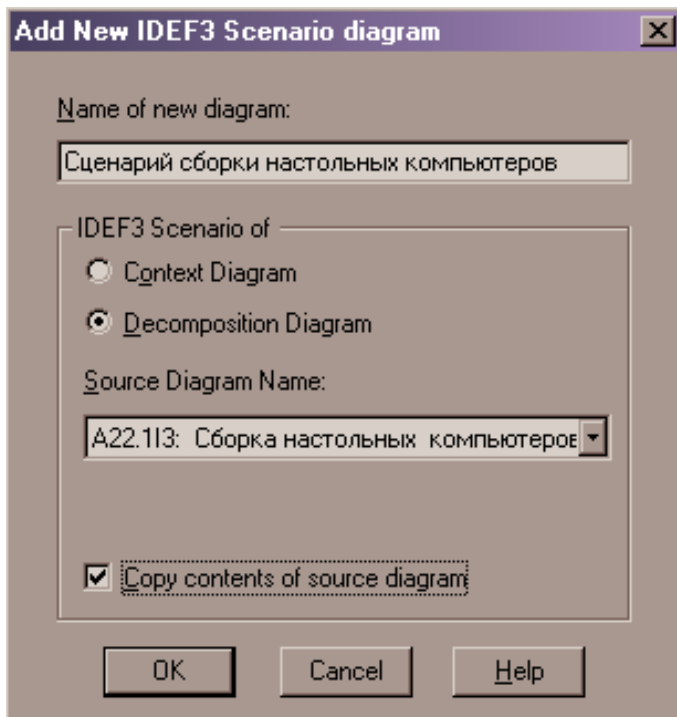


2.8 Упражнение 8. Создание сценария

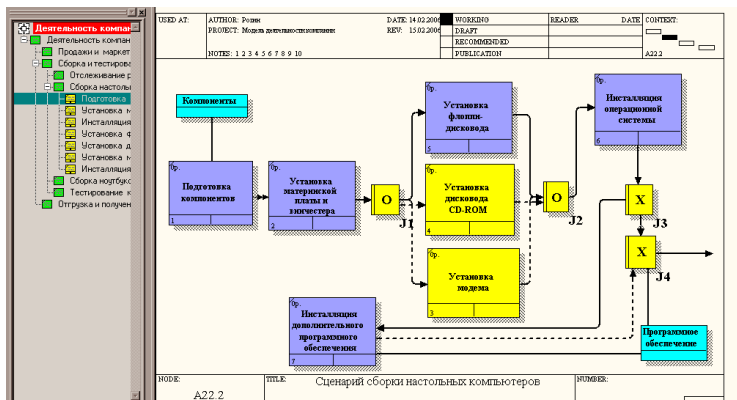
В **IDEF3** декомпозиция используется для детализации работ. Любая работа может содержать множество дочерних работ. Декомпозиция может быть **сценарием** или **описанием**.

Описание включает все возможные пути развития процесса. **Сценарий** является частным случаем описания и иллюстрирует только один путь реализации процесса. **По умолчанию** при декомпозиции на диаграмме **IDEF3** создается **описание**. Чтобы создать **сценарий**, необходимо перейти в меню **Diagram/Add IDEF3 Scenario**.

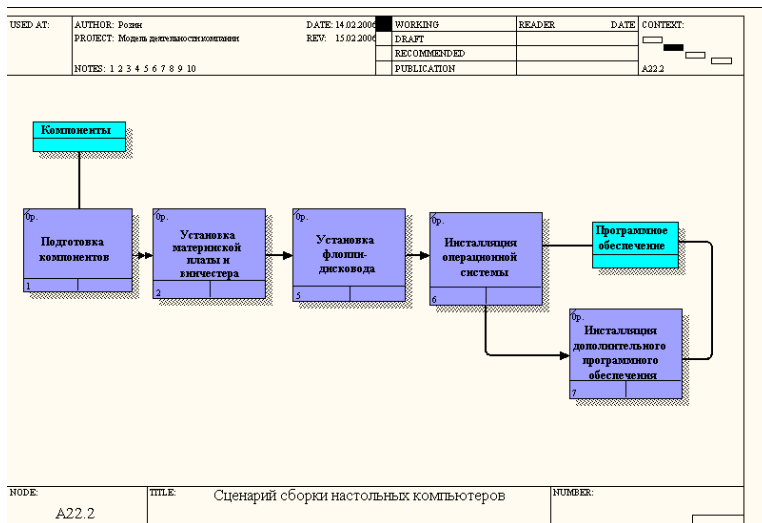
1. Выберите пункт меню **Diagram/Add IDEF3 Scenario**.
2. Создайте **диаграмму сценария** на основе диаграммы

IDEF3 "Сборка настольных компьютеров".


3. **Удалите** элементы, не входящие в сценарий (помечены желтым цветом).



4. Результат выполнения упражнения 8.



2.9 Упражнение 9.

Диаграммы потоков данных (Data flow diagrams, DFD) используются для описания документооборота и обработки информации. Они представляют модельную систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDF0 для более наглядного отображения текущих операций документооборота. DFD описывает: функции обработки информации (работы), документы (стрелки), внешние ссылки, которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы, таблицы для хранения данных (хранилища данных).

Предварительно выполните следующее упражнение:

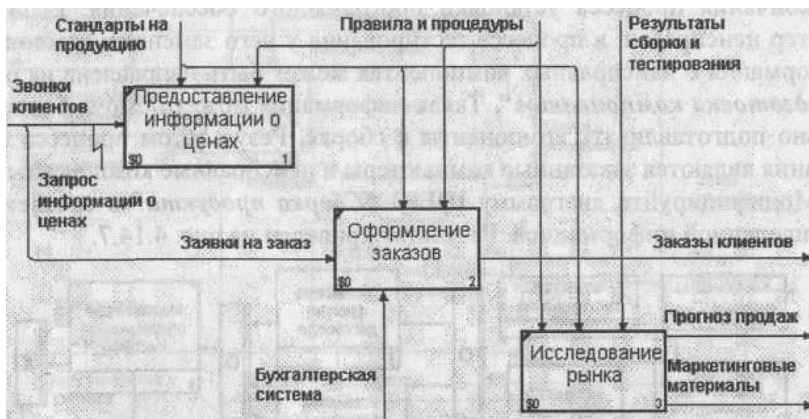
Работа по продажам и маркетингу заключается в ответах на телефонные звонки клиентов, предоставлении клиентам информации о ценах, оформлении заказов, внесении заказов в информационную систему и исследовании рынка.

Декомпозируйте работу «**Продажи и маркетинг**» (IDF0, количество работ – 3).

Внесите имена работ: предоставление информации о ценах, оформление заказов, исследование рынка.

Внесите необходимые стрелки: звонки клиентов (граничная, вход для работ «Представление информации о ценах» - имя «запрос информации о ценах», «Оформление заказов» - имя «звонки

на заказ»), правила и процедуры – управление для всех блоков, стандарты на продукцию – управление для всех блоков, результаты сборки и тестирования – управление для «Исследования рынка», бухгалтерская система – механизм для «Оформления заказов», Прогноз продаж и Маркетинговые материалы – выход из «Исследования рынка».



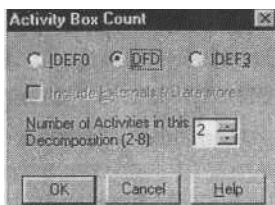
Создание диаграммы DFD.

Задача. При оформлении заказа важно проверить, существует ли такой клиент в базе данных и, если не существует, внести его в базу данных и затем оформить заказ. Оформление заказа начинается со звонка клиента. В процессе оформления заказа база данных может просматриваться и редактироваться. Заказ должен включать как информацию о клиенте, так и информацию о заказанных продуктах. Оформление заказа подразумевает чтение и запись информации о прочих заказах.

В процессе декомпозиции согласно правилам DFD необходимо преобразовать граничные стрелки во внутренние, начинающиеся и заканчивающиеся на внешних ссылках.

Декомпозируйте работу «**Оформление заказов**» (DFD, количество работ - 2)

В диалоге Activity Box Count выберите количество работ 2 и нотацию DFD.



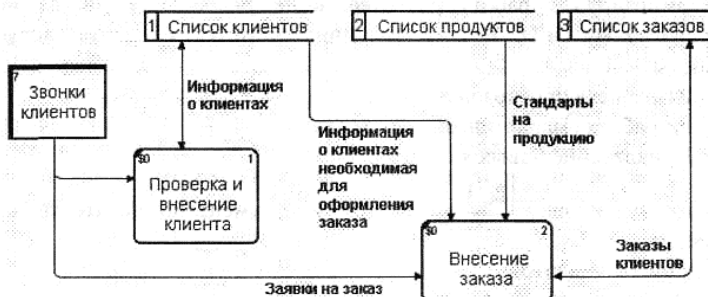
2. Внесите имена работ: проверка и внесение клиента, внесение заказа.

3. Внесите хранилища данных: список клиентов, список продуктов, список заказов. Используйте кнопку

4. Удалите граничные стрелки на диаграмме DFD.

5. Используя кнопку ссылки (с изображением прямоугольника) на палитре инструментов, внесите внешнюю ссылку: звонки клиентам.

6. Создайте стрелки: звонки клиентов – проверка и внесение клиента, внесение заказа; список клиентов – внесение заказа (имя – информация о клиентах для оформления заказов); список продуктов – внесение заказа (имя – стандарты на продукцию); список заказов – внесение заказа (имя – заказы клиентов), список клиентов – проверка и внесение клиентов (имя – информация о клиентах).



7. Сделайте стрелки «Информация о клиентах» и «Заказы клиентов» двунаправленными. Для того чтобы сделать стрелку двунаправленной, щелкните правой кнопкой по стрелке, выберите в контекстном меню пункт Style и во вкладке Style выберите опцию **Bidirectional**.

8. На родительской диаграмме выполните туннелирование стрелок (Change to Tunnel), подходящих к работе и исходящих из работы «Оформление заказов»: заявки на заказ, заказы клиентов, бухгалтерская система, стандарты на продукцию, правила и про-

цедуры.




Использование Off-Page Reference на диаграмме DFD.

Некоторые стрелки с диаграмм IDFO и DFD могут изображаться на диаграмме DFD. Для отображения таких стрелок используется инструмент Off-Page Reference.

Декомпозируйте работу **"Исследование рынка"** на диаграмме A2 на диаграмму DFD (DFD, количество работ – 3). Удалите граничные стрелки. Создайте следующие работы:

- Разработка прогнозов продаж;
- Разработка маркетинговых материалов;
- Привлечение новых клиентов.

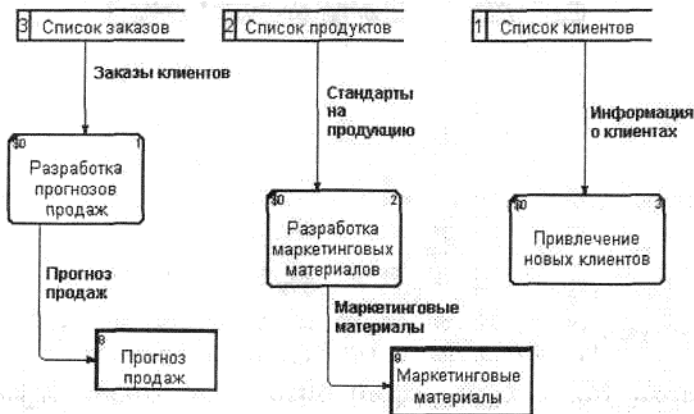
2. Используя кнопку  на палитре инструментов, внесите хранилища данных:

- Список клиентов;
- Список продуктов;
- Список заказов.

3. Добавьте две внешние ссылки:
Маркетинговые материалы;
Прогноз продаж.

4. Свяжите объекты диаграммы DFD стрелками, как показано на рис.

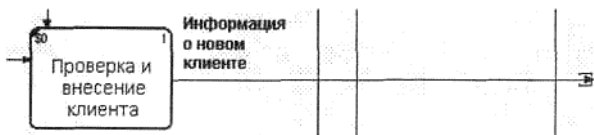
Информационные системы в строительстве



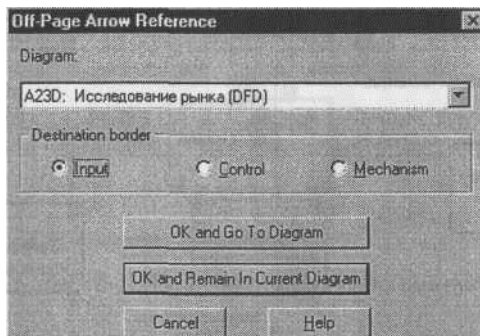
На родительской диаграмме A2 туннелируйте (Change to Tunnel) стрелки, подходящие к работе и исходящие из работы "Исследование рынка".

В случае внесения новых клиентов в работе "Проверка и внесение клиента" на диаграмме A22 "Оформление заказов" информация должна направляться к работе "Привлечение новых клиентов" диаграммы A23 "Исследование рынка". Для этого необходимо использовать инструмент Off-Page Reference.

На диаграмме A22 "Оформление заказов" создайте новую граничную стрелку, исходящую от работы "Проверка и внесение клиента", и назовите ее "Информацией о новом клиенте".



7. Правой кнопкой щелкните по наконечнику стрелки и выберите в меню Off-Page Reference. В появившемся диалоге Off-Page Arrow Reference выберите в качестве диаграммы A23D "Исследование рынка".



Перейдите в меню Model/Model Properties, далее - во вкладку Display. Установите опцию Off-Page Reference label – Node number.

Перейдите на диаграмму A23D "Исследование рынка" и направьте стрелку **"Информация о новом клиенте"** на вход работы **"Привлечение новых клиентов"**. Результат представлен на рис.

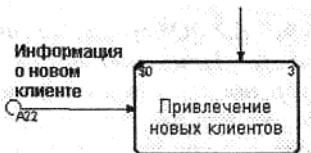


Рис. Межстраничная ссылка на диаграмме A23

3 ERWIN

3.1 МОДЕЛЬ "СУЩНОСТЬ- СВЯЗЬ"

3.1.1 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, выявлять бизнес-правила и позволять легко перейти к модели данных конкретной СУБД.

Таковыми моделями являются *модели "сущность-связь"*. Зачастую эти модели называют моделями данных. Известно несколько методологий построения моделей "сущность-связь". Наибольшее распространение получила методология IDEF1X. Рассмотрим построение моделей "сущность-связь", ориентируясь на продукт CA ERwin Data Modeler. 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 именуется 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. 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.

Студент

 Номер зачетки
Шифр группы
Фамилия
Имя
Отчество
Пол
Дата рождения
Адрес

Рис. 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 **ТАБЕЛЬ-НЫЙ_НОМЕР** транзитивно определяет **ТЕЛЕФОН**.

3.1.2 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. 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. 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. 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 отношении единственный экземпляр родительской сущности связан с множеством экземпляров подчиненной. Атрибуты первичного ключа родительской сущности мигрируют в подчиненную, чтобы стать там *неключевыми* атрибутами. На рис. 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. 3. Пример обязательного неидентифицирующего отношения

На примере рис. 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. 4 показан пример необязательно отношения: допускается, что сотрудник может не принадлежать ни одному из отделов.



Рис. 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). Для каждой категории необходимо указать *дискриминатор* - атрибут родительской сущности, показывающий, как отличить одну категориальную сущность от другой. На рис. 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. 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. 6 показан пример связи "многие-ко-многим", а на рис. 7 - пример использования ассоциативной сущности.

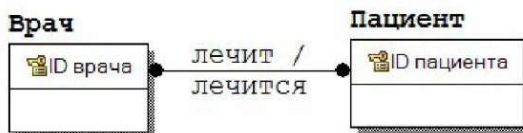


Рис. 6. Пример связи "многие-ко-многим"



Рис. 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. 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. 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.

3.1.3 Правила ссылочной целостности

Целостность данных является понятием базы данных.

ERwin позволяет в модели "сущность-связь" задать *правила ссылочной целостности*. 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 соответствующее 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.

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.

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 внешних 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.

3.1.4 Контрольные вопросы

1. Охарактеризуйте логический уровень представления модели в ERWin.
2. Охарактеризуйте физический уровень представления модели в ERWin.
3. Что такое: сущность, атрибут, домен?
4. Понятие первичного ключа, искусственного первичного ключа, внешнего ключа.
5. Правила выбора и описания атрибутов.
6. Свойства связей (отношений) между сущностями.
7. Что такое «степень отношения»?
8. Что такое «направленность отношений»?
9. Перечислите типы отношений между сущностями.
10. Что такое идентифицирующий тип отношений?
11. Что такое неидентифицирующее отношение?
12. Охарактеризуйте обязательность отношений.
13. Что такое типизирующие отношения?
14. Отношения «Многие-ко-многим».
15. Рекурсивное отношение. Пример.
16. Охарактеризуйте мощность отношений.
17. Характеристика правил ссылочной целостности.
18. Операции с кортежами родительского отношения.
19. Операции с кортежами дочернего отношения.
20. Стратегии поддержания ссылочной целостности (Restrict и Cascade).
21. Какие стратегии доступны при обновлении кортежа в родительском отношении?
22. Какие стратегии доступны при удалении кортежа в родительском отношении?
23. Какие стратегии доступны при вставке кортежа в дочернее отношение?
24. Какие стратегии доступны при обновлении кортежа в дочернем отношении?

4 ПРАКТИЧЕСКИЕ ЗАДАНИЯ

4.1 СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ

4.1.1 Начало работы с ERwin

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

При запуске ERwin открывается главное окно программы CA ERwin Data Modeler (рис. 1).

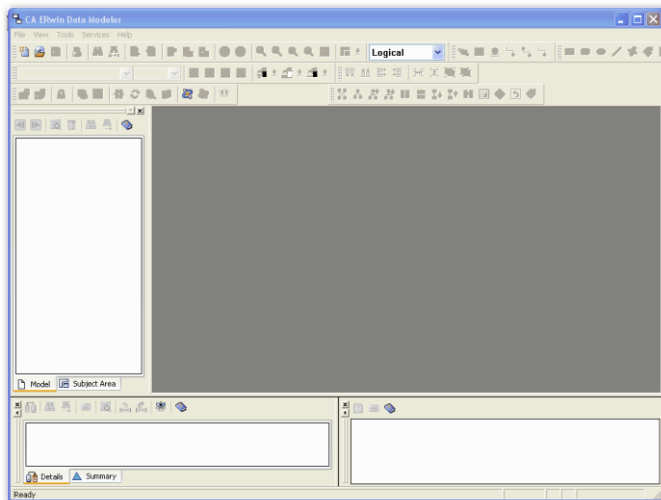



Рис. 1. Главное окно программы **CA ERwin Data Modeler**

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. 2), вызываемом через **File>New** или кнопку создания нового файла . При выборе физической или логико-

физической модели в диалоге **Create Model** необходимо также указать необходимую СУБД и номер ее версии (в полях **Database** и **Version**).

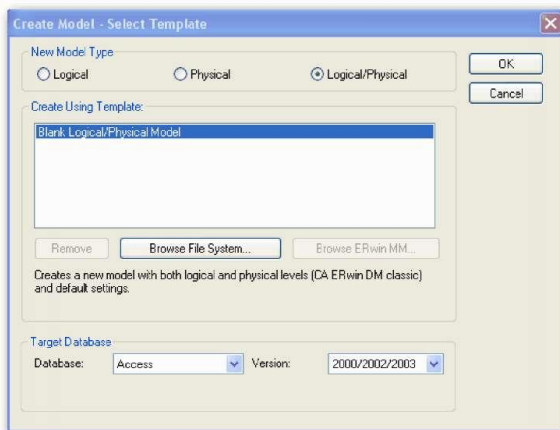


Рис. 2. Диалог **Create Model**

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

Browse File System...

В соответствии с выбранным действием при активизации кнопки **OK** открывается окно для построения новой модели данных (рис. 3) или окно редактирования модели, созданной ранее.

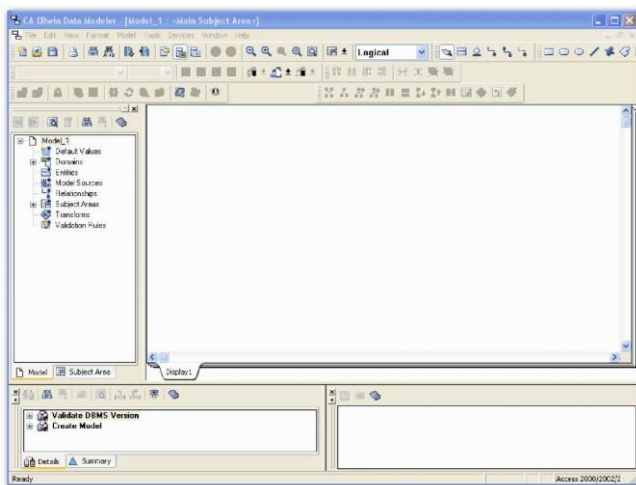


Рис. 3. Окно для построения новой модели данных

4.1.2 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. 4).



Рис. 4. Пункт **Logical** на панели инструментов

В противном случае тип модели необходимо выбрать из выпадающего меню данного пункта (рис. 5).



Рис. 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).

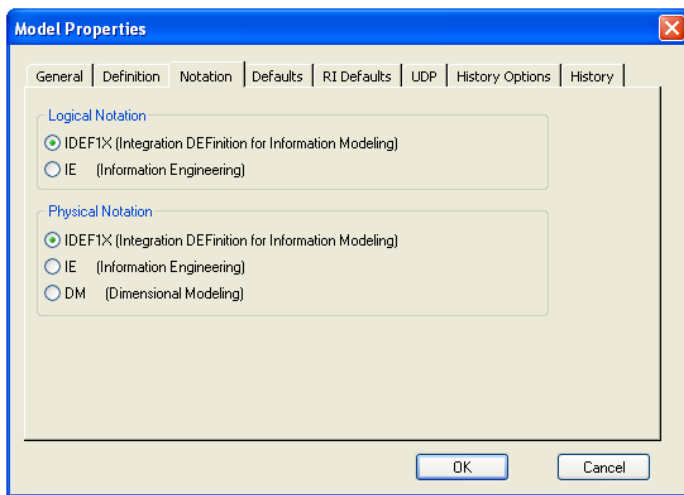








Рис. 6. Смена нотации на вкладке **Notation** диалога **Model Properties**

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. 1.

Таблица 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 шелкнуть левой кнопкой мыши по кнопке и один раз по свободному пространству на модели
	Создание категорин – для установления категоральной связи (специальный тип связи между сущностями) необходимо шелкнуть левой кнопкой мыши по кнопке категорин, затем один раз шелкнуть 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.

4.1.3 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 сохра-

няться.

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. 7).

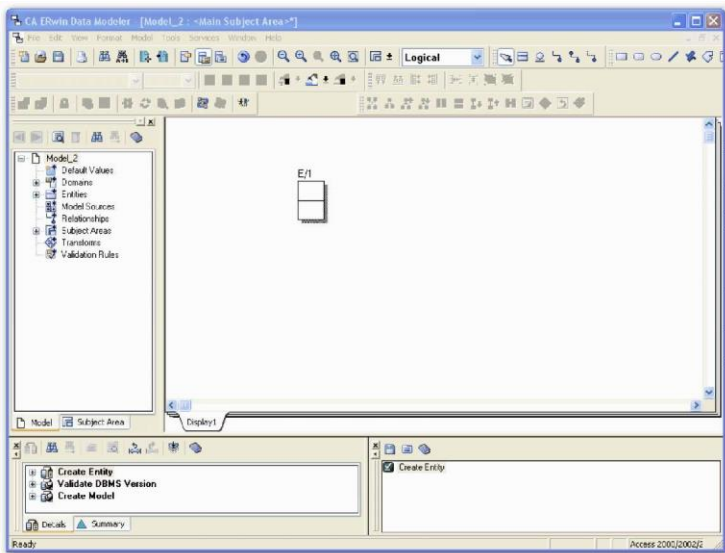


Рис. 7. Размещение в поле проектирования сущности с именем **E/1** по умолчанию

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. 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. 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 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. 9).

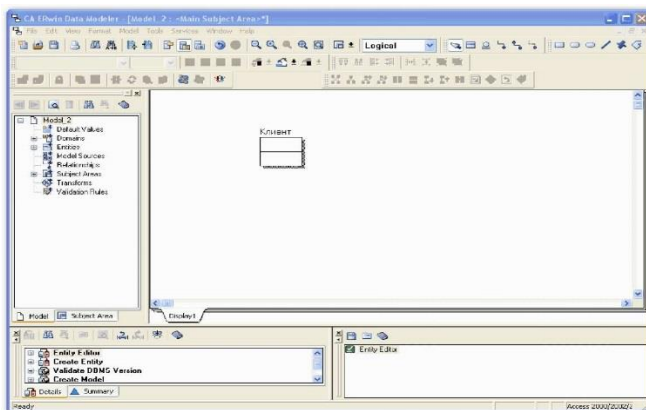


Рис. 9. Размещение в поле проектирования сущности с именем **Клиент**

При разработке ER-модели можно установить *параметры шрифтов и цветовой оформление сущностей* для облегчения обзора и понимания модели. При этом можно изменять установки параметров, назначенные по умолчанию, при добавлении нового объекта 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 от-

крывается с помощью пункта **Attributes...** контекстного меню и позволяет ввести данные об атрибутах выбранной сущности (рис. 10).

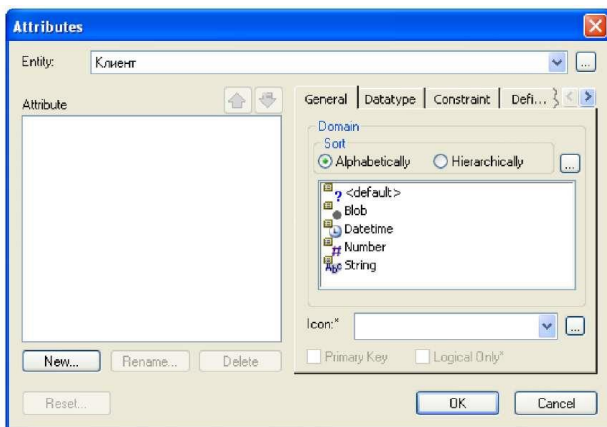


Рис. 10. Диалоговое окно **Attributes**

При активизации кнопки **New.** диалогового окна **Attributes** открывается диалог **New Attribute**, позволяющий добавить новый атрибут для выделенной сущности (рис. 11).



Рис. 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. 12).



Рис. 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. 13).

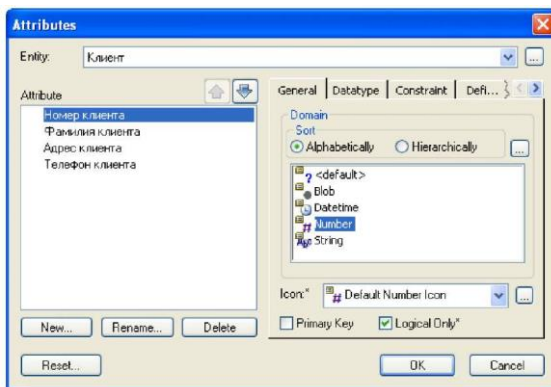


Рис. 13. Атрибуты сущности **Клиент**

Для определения первичного ключа выделенной сущности необходимо перейти на вкладку **General** и сделать пометку в окне выбора **Primary Key** (рис. 14).

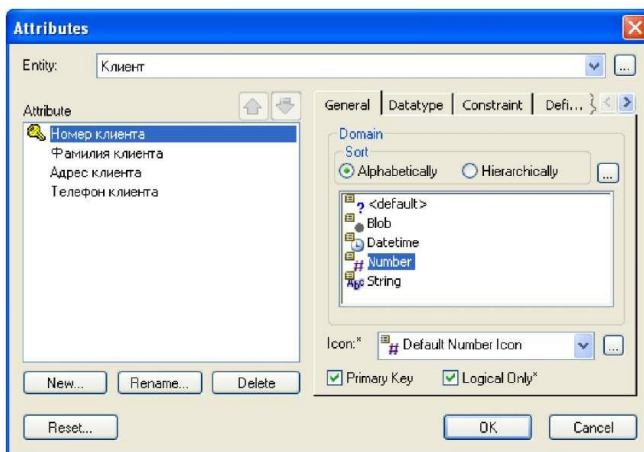


Рис. 14. Определение первичного ключа в окне выбора **Primary Key**

Вкладка **Datatype** позволяет с помощью выпадающего списка на странице уточнить тип данных для выделенного атри-

бута (рис. 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.

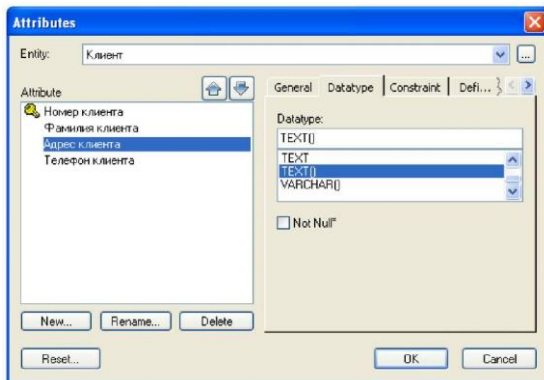


Рис. 15. Уточнение типа данных атрибута
Адрес клиента

Вкладка **Key Group** (рис. 16) предназначена для отображения атрибутов, входящих в группу ключевых атрибутов. В частности, на вкладке показано, что первичным ключом сущности **Клиент** является атрибут **Номер клиента**.

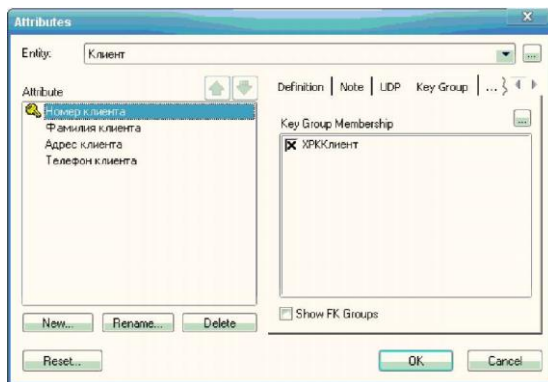
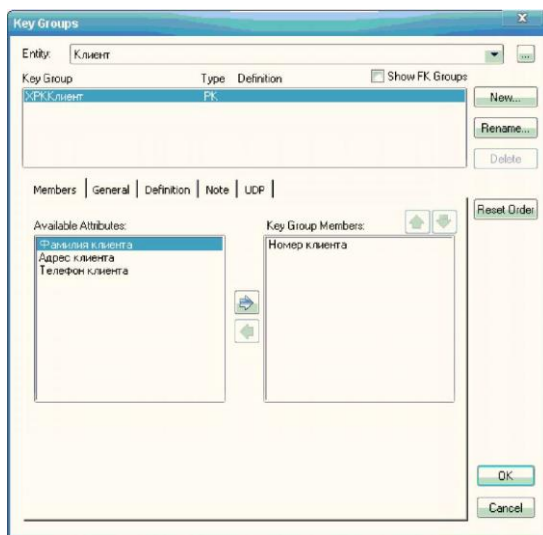


Рис. 16. Вкладка **Key Group** диалога **Attributes**

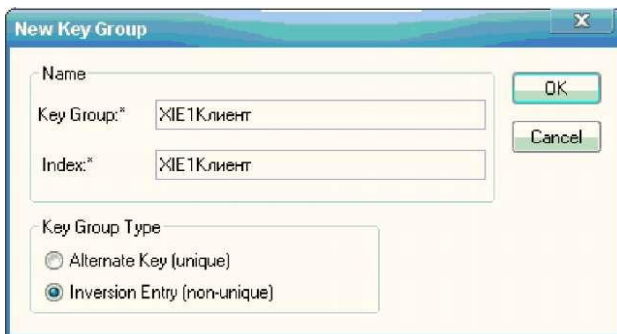
Сущность может иметь несколько возможных (*потенциальных*) *ключей*. Один потенциальный ключ объявляется первичным, а остальные могут быть объявлены *альтернативными ключами (Alternate Key)*.

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. 17).


 Рис. 17. Диалог **Key Groups**

Видно (рис. 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. 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. 19).

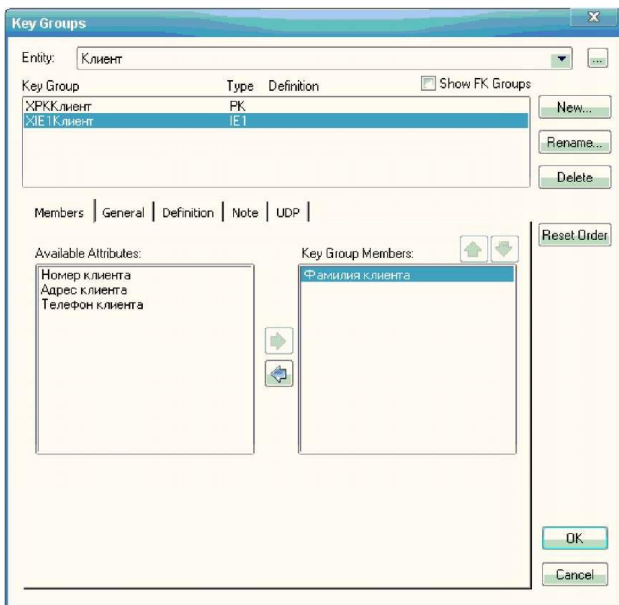



Рис. 19. Диалог **Key Groups** при построении инверсного входа

В верхнем окне **Key Group** вкладки выбираем ключевую группу **XIEКлиент**, 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. 20).

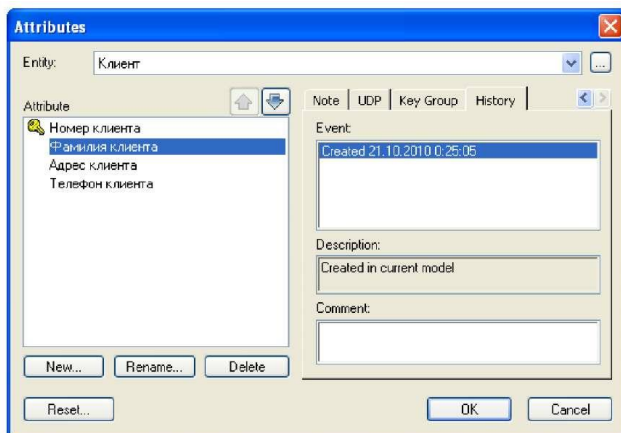


Рис. 6.20. История создания и изменения свойств атрибута
Фамилия клиента

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

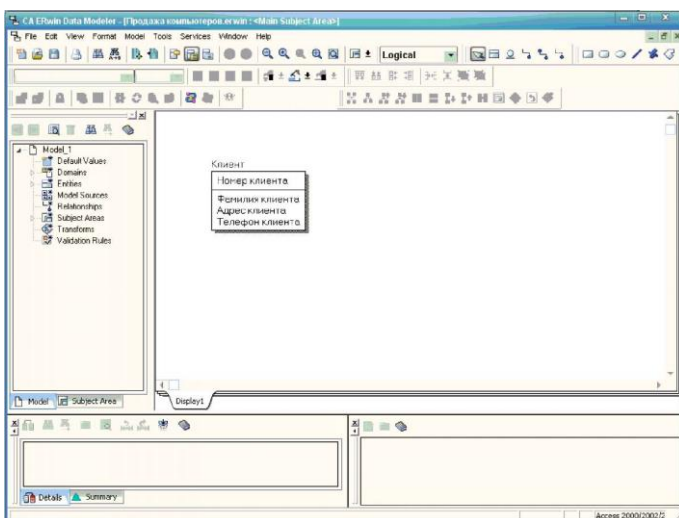
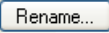


Рис. 21. Сущность **Клиент** с заданными атрибутами

Диалоговое окно **Attributes** позволяет пользователю ре-

дактировать 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  диалогового окна **Attributes** (рис. 22).

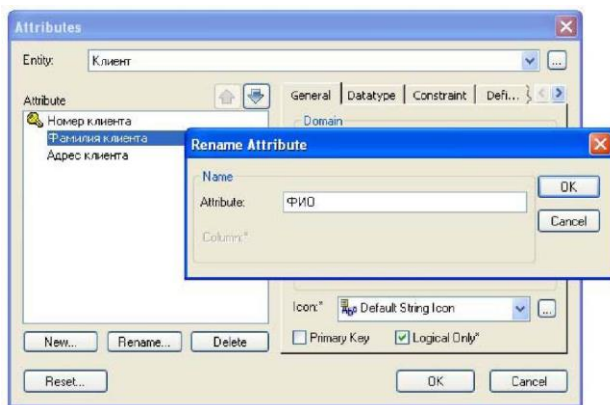



Рис. 22. Редактирование имени атрибута **Фамилия клиента**

При активизации кнопки  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 специальных 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. 23).

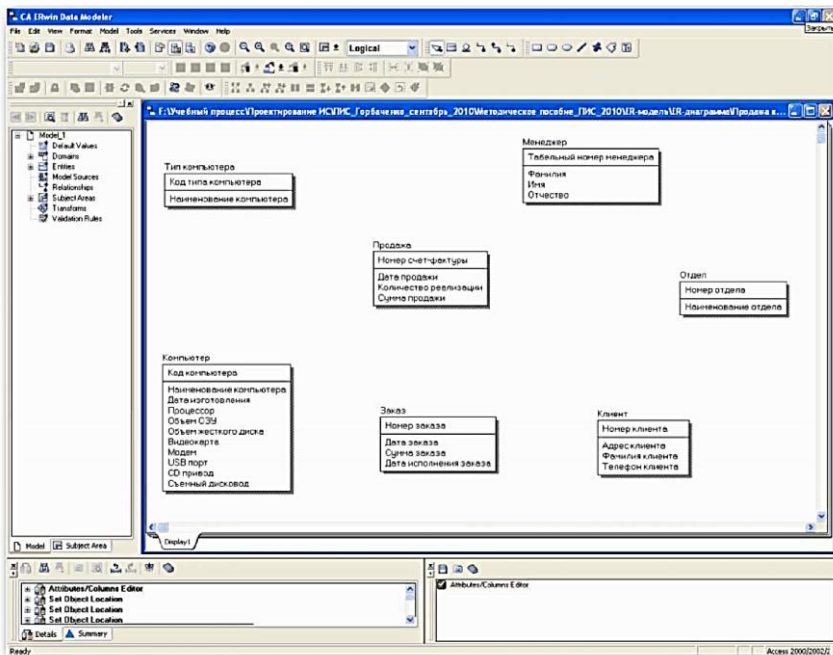


Рис. 23. Сущности, выделенные для описания объектов предметной области по реализации настольных компьютеров по заказам клиентов

4.1.4 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. 24), какой компьютер продан и какой менеджер оформил сделку продажи компьютера: каждый **Компьютер** < продается > **Продажа**; каждая **Продажа** < оформляет > **Менеджер**.

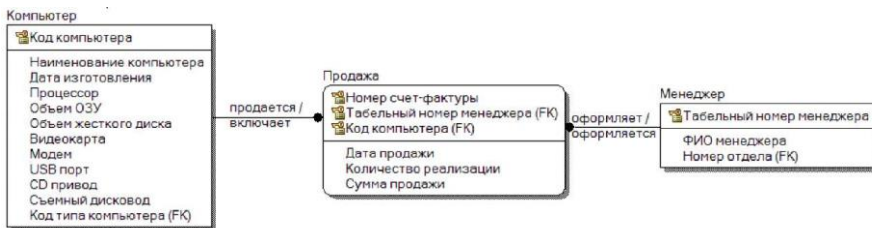


Рис. 24. Связь между сущностями **Компьютер**, **Продажа** и **Менеджер**

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. 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. 25).



Рис. 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. 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. 26).

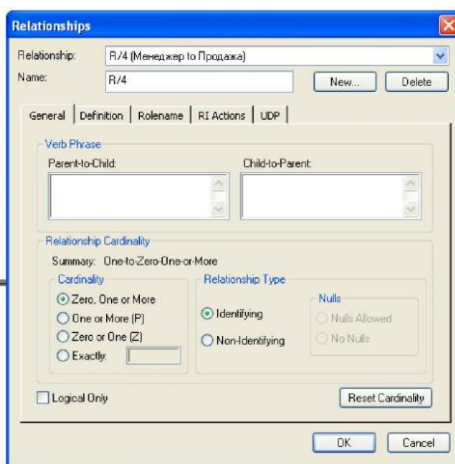


Рис. 26. Диалоговое окно **Relationship**

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. 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. исключены множественные значения;

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

—

Информационные системы в строительстве

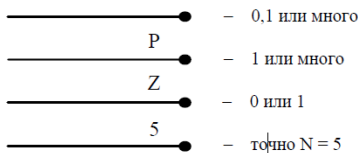


Рис. 27. Обозначение мощности связей

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

Пример определения мощности и имени связи между сущностями **Отдел** и **Менеджер** приведен на рис. 28.

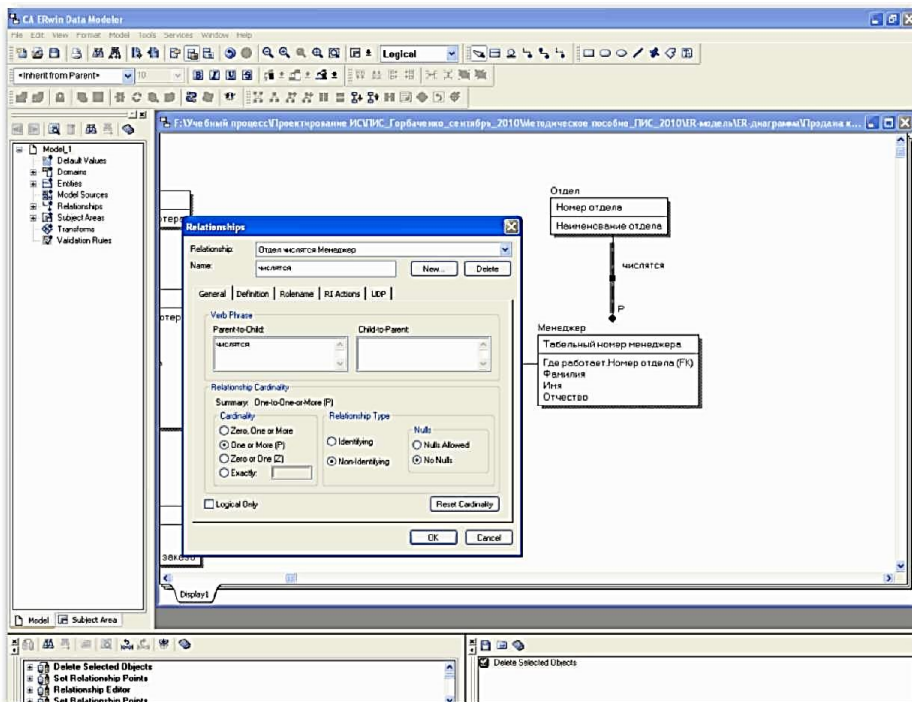


Рис. 28. Пример определения мощности и имени связи между сущностями **Отдел** и **Менеджер**

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. 29).



Рис. 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. 30).

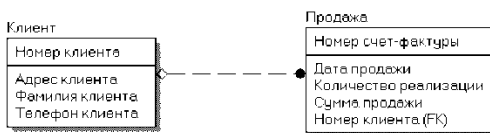


Рис. 30. Пример реализации необязательной неидентифицирующей связи

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. 31).

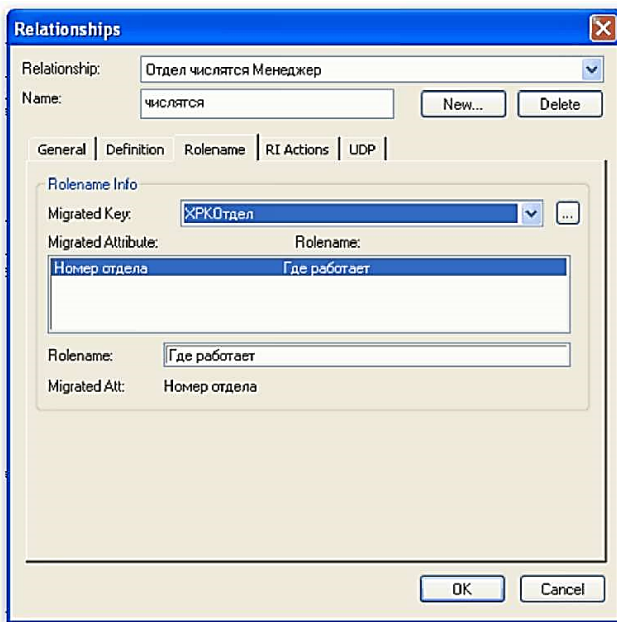


Рис. 31. Окно диалога **Relationships** вкладки **Rolename**

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 атрибута (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. 32).

Менеджер
Табельный номер менеджера
Где работает.Номер отдела (FK)
Фамилия
Имя
Отчество

Рис. 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. 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.

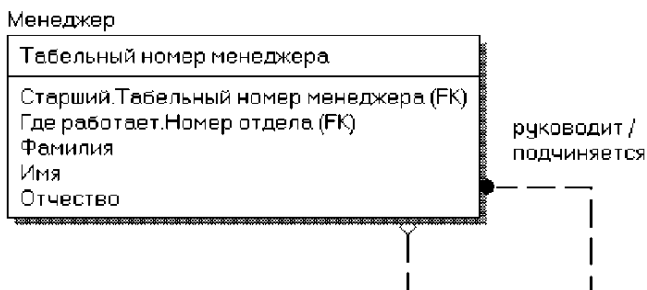


Рис. 33. Пример рекурсивной связи

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. 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. 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. 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 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. 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**.

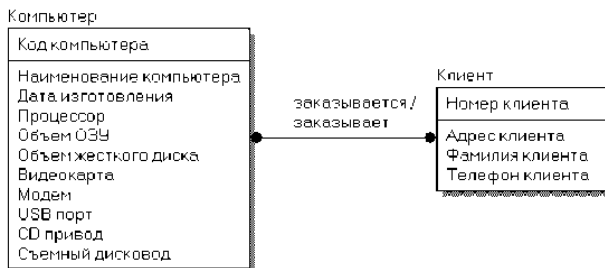


Рис. 35. Пример реализации связи "многие-ко-многим"

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. 36).

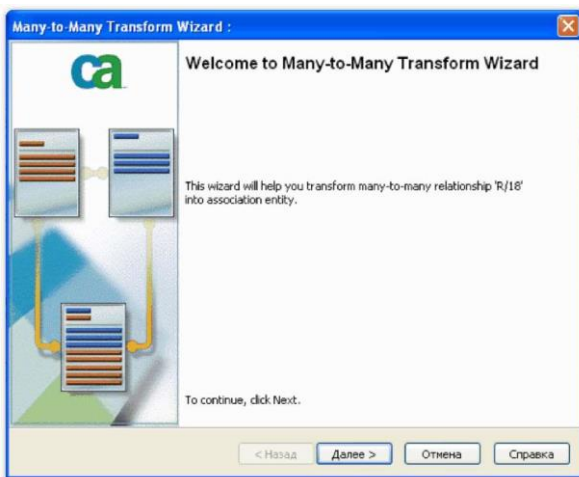


Рис. 36. Окно диалога **Many-To-Many Transform Wizard**

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 получают 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. 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**.

Информационные системы в строительстве

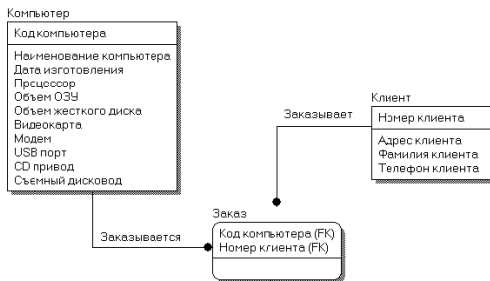


Рис. 37. Пример результата принудительного преобразования связи "многие ко многим"

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. 38).

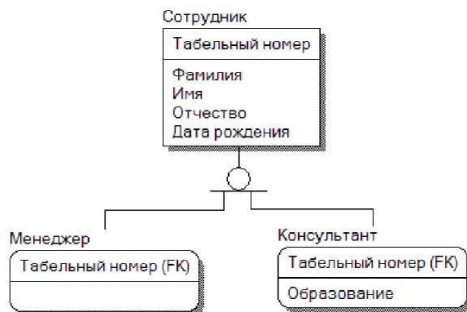



Рис. 38. Пример иерархии наследования (категориальной связи)

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. 39). Поля диалогового окна позволяют указать дискриминатор - атрибут категории (список **Discriminator**) и тип категории - полная/неполная (радиокнопки **Complete/Incomplete**).

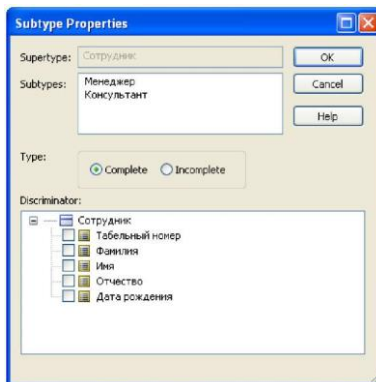


Рис. 39. Диалоговое окно **Subtype Properties**

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. 38). На рис. 40 представлен пример полной категории. Возможна также комбинация полной и неполной категорий.

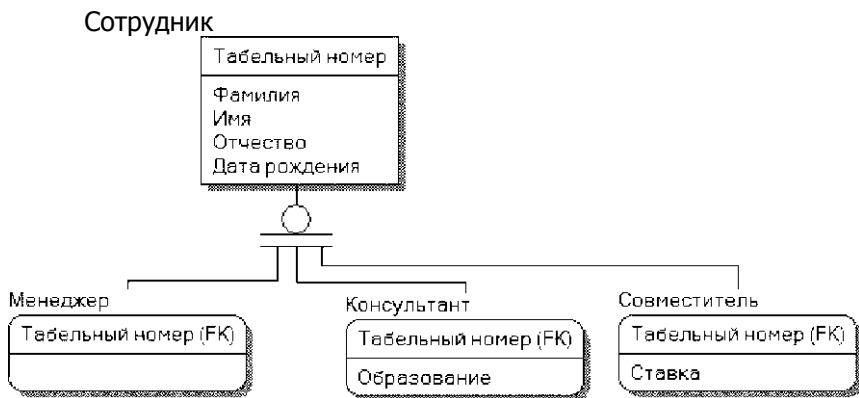


Рис. 40. Пример полной категории

Результат разработки логической модели данных системы "Реализация средств вычислительной техники", предназначенной для учета продаж настольных компьютеров по заказам клиентов приведен на рис. 6.41.

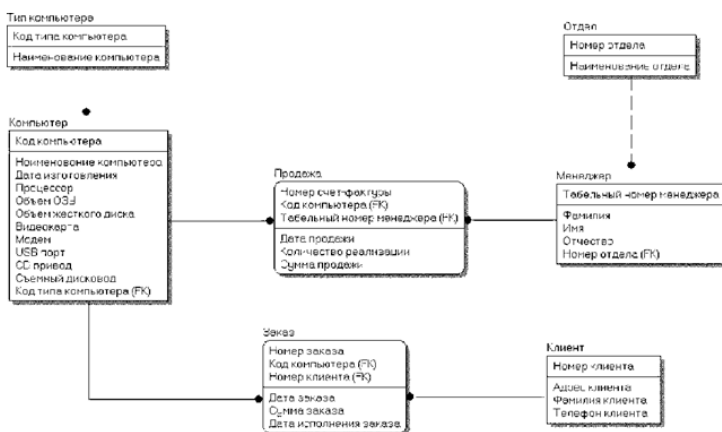


Рис. 41. Логическая модель данных системы "Реализация средств вычислительной техники"

Логический уровень модели данных является универсальным и никак не связан с конкретной реализацией СУБД. Одному и тому же логическому уровню модели могут соответствовать

вать несколько разных физических уровней различных моделей, где описывается вся информация о конкретных физических 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.

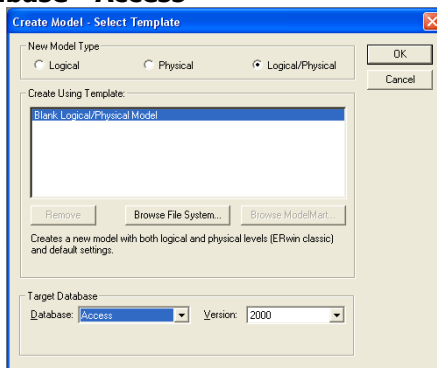
4.1.5 Упражнение 1.

1. Запустите программу **ERwin**
2. Создайте новую модель. **create a new model.**


Укажите, что создается **логически-физическая модель**.

Логически-физическая модель можно создавать для последующего создания базы данных в полностью русифицированных СУБД, например, Access (названия атрибутов и таблиц могут быть русскими). В остальных случаях логическую и физическую модель разделяют. В логической названия сущностей и атрибутов – русские. А в физической используется латиница.

Logical/Physical, Target Database - Access

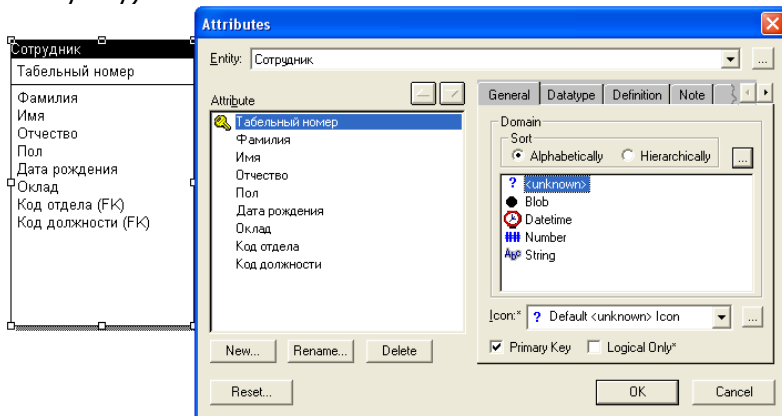


3. Автоматически создается модель. На панели инструментов выберите логическую модель. **Logical.**

4. При помощи кнопки  (entity) на панели инструментов создайте сущность "Сотрудник". Задайте для сущности "Сотрудник" следующие атрибуты:

1. Табельный номер
2. Фамилия
3. Имя
4. Отчество
5. Пол
6. Дата рождения
7. Оклад

Определите атрибут табельный номер как ключевой атрибут (Primary Key)



5. Создайте сущности "Отдел", "Должность", "Дети сотрудников".

Создайте для сущности "Отдел" следующие атрибуты:

1. Код отдела
2. Название отдела

Создайте для сущности "Должность" следующие атрибуты:

1. Код должности
2. Название должности

Создайте для сущности "Дети сотрудников" следующие атрибуты:

1. ID номер
2. Фамилия
3. Имя
4. Отчество
5. Пол
6. Дата рождения

Определите атрибуты "Код отдела", "Код должности", "ID номер" как ключевые атрибуты

6. Установите связи для сущностей:

Информационные системы в строительстве

- "Отдел" и "Сотрудник", "Должность" "Сотрудник" - Неидентифицирующие связи;
- "Сотрудник" и "Дети сотрудников" - идентифицирующая связь



В результате у сущности "Сотрудник" появились атрибуты "Код отдела" "Код должности", помеченные как Foreign Key (FK). А у сущности "Дети сотрудников" ключевой атрибут "Табельный номер" так же помеченный как(FK).

4.1.6 Контрольные вопросы

1. Охарактеризуйте диаграмму «Сущность-связь» (ERD), модель данных, основанную на ключах и полную атрибутивную модель данных.
2. Охарактеризуйте возможности диалога Entities (Сущность) (перечень и назначение всех вкладок)
3. Охарактеризуйте возможности диалога Attributes (Атрибуты) (перечень и назначение всех вкладок)
4. Какие правила существуют для имен атрибутов в рамках нотации IDEF1X?
5. Виды ключей для сущности в ERWin. Способы их создания.
6. Охарактеризуйте необходимость и возможность создания производных атрибутов для описания сущности.
7. Виды связей между сущностями. Правила именования связей.
8. Что такое зависимые и независимые сущности?
9. Что такое миграция атрибутов?
10. Что такое мощность связи? Типы мощности связей.
11. Что включает полное имя атрибута?
12. Когда применение имен ролей является обязательным?

13. Что такое иерархическая и сетевая рекурсия?
14. Что такое правила ссылочной целостности? Как они определяются? Шаблоны триггеров.
15. Как создается, обозначается и именуется связь «Многие-ко-многим»? Особенности реализации такой связи на физическом уровне. Принудительное преобразование связи.
16. Что такое иерархия наследования (категориальная связь)? Как создаются и редактируются такие связи?
17. Типы иерархии категорий.

4.2 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ

4.2.1 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 **CreateModel** через пункты основного меню **File>New** или кнопку нового файла , при этом в окне выбора нового типа модели **NewModelType** (рис. 1) следует выбрать физический уровень представления (радиокнопка **Physical**).



Рис. 1. Окно выбора нового типа модели **NewModelType**

Физический уровень представления модели зависит от конкретной реализации СУБД, поэтому необходимо предварительно осуществить ее выбор. CAERwinDataModeler поддерживает 17 наиболее распространенных СУБД. Выбор СУБД осуществляется в окне **TargetDatabase** диалога **CreateModel**. 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. 2).

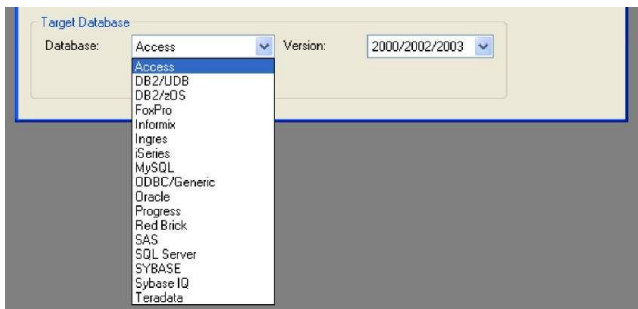


Рис. 2. Окно выбора СУБД **TargetDatabase**

При активизации кнопки **OK** диалога **CreateModel** 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. 3).



Рис. 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. 4).

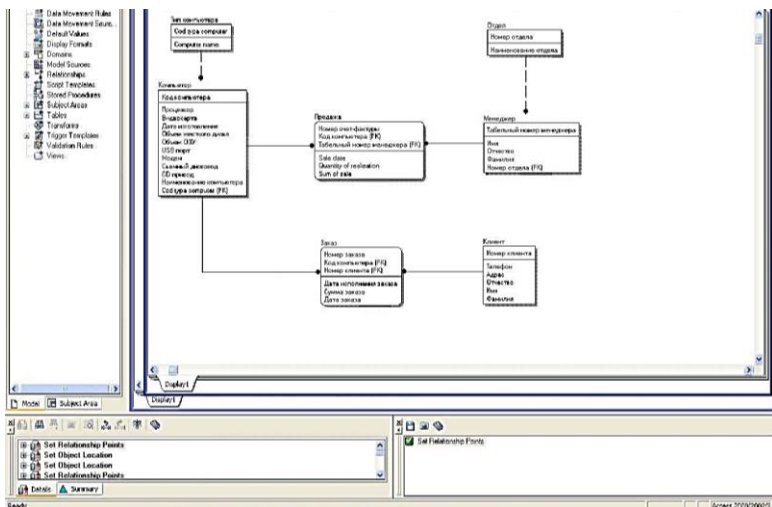








Рис. 4. Физический уровень логико-физической модели данных системы


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.

4.2.2 Добавление/редактирование таблиц

Для построения/редактирования ER-модели физического уровня используется панель инструментов, содержащая кнопки редактирования модели, условные графические обозначения которых приведены в табл. 1.

Таблица 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. 5, 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 **AccessTables** (рис. 7).

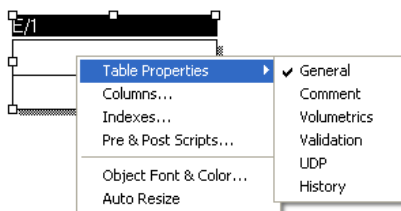
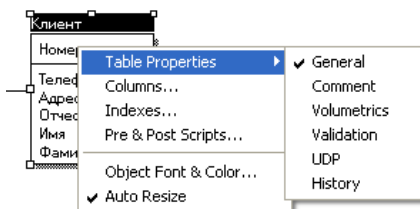
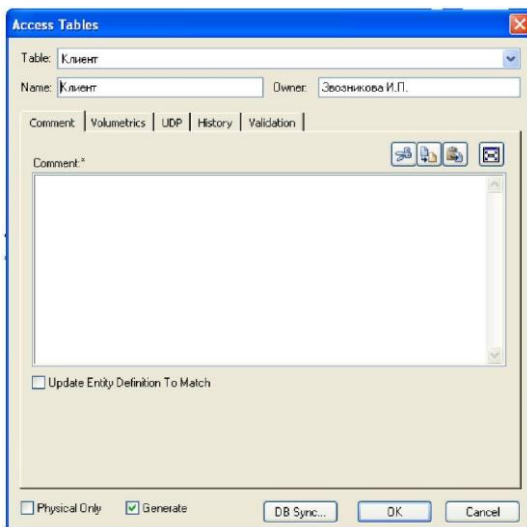


Рис. 5. Контекстное меню добавленной таблицы

Информационные системы в строительстве


 Рис. 6. Контекстное меню таблицы **Клиент**

 Рис. 7. Диалог **AccessTables**

Диалог **Tables** (в нашем случае **AccessTables**) 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. 8).

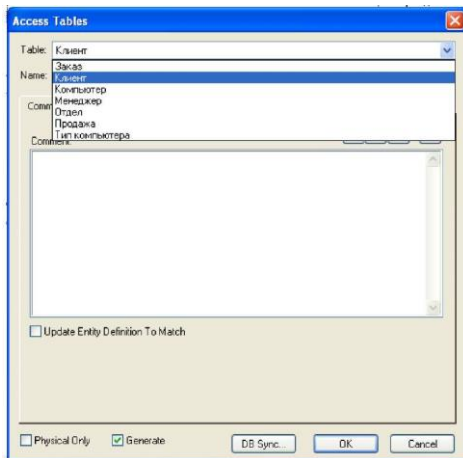


Рис. 8. Список выбора таблиц для редактирования окна **Table**

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 **CREATETABLE** (создать таблицу). Кнопка **DBSync** предназначена для синхронизации модели с системным каталогом базы данных (рис. 9).

Диалог **Tables** содержит следующие вкладки:

- **Comment** - предназначена для внесения комментария к таблице (рис. 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. 10).

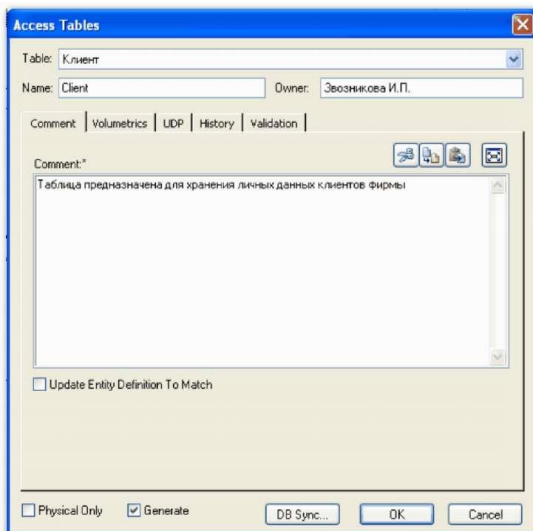


Рис. 9. Изменение имени и внесение имени владельца таблицы **Клиент**

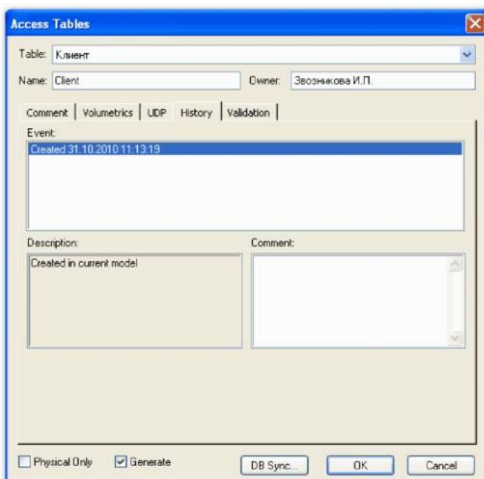


Рис. 10. История создания и изменения свойств таблицы **Клиент**

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.

4.2.3 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. 11).

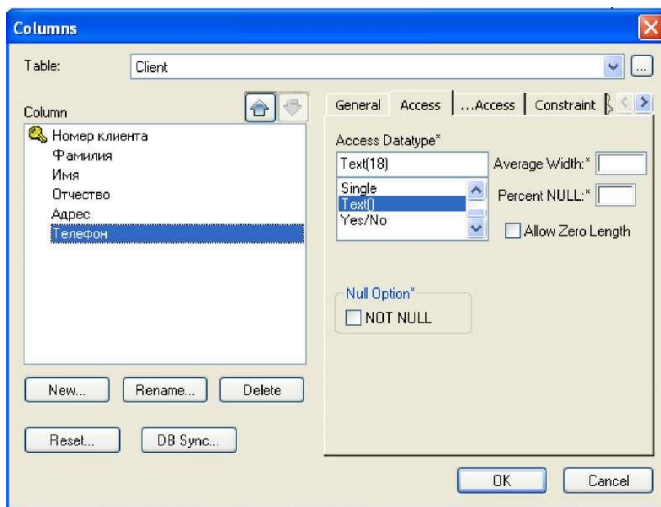


Рис. 11. Диалоговое окно **Columns**

По умолчанию ERwin присваивает пустые значения (**NULL**) всем неключевым колонкам, а для колонок первичного ключа и альтернативных ключей устанавливает режим **NOTNULL**. Режим **NOTNULL** не присваивается автоматически инверсным входам (**InversionEntry**).

Диалоговое окно **Columns** по своему внешнему виду напоминает диалоговое окно редактора свойств атрибутов **Attributes** (см. Л.р. 1 рис. 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. 12).

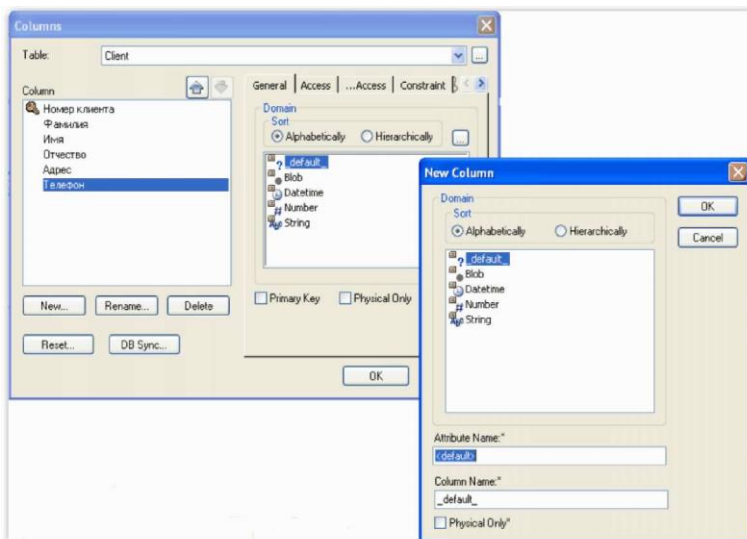
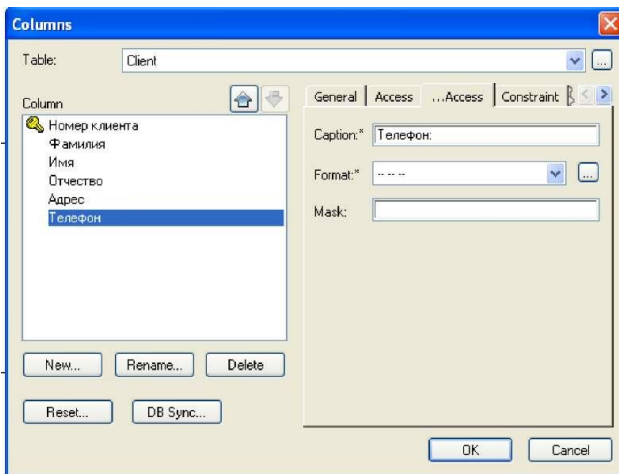
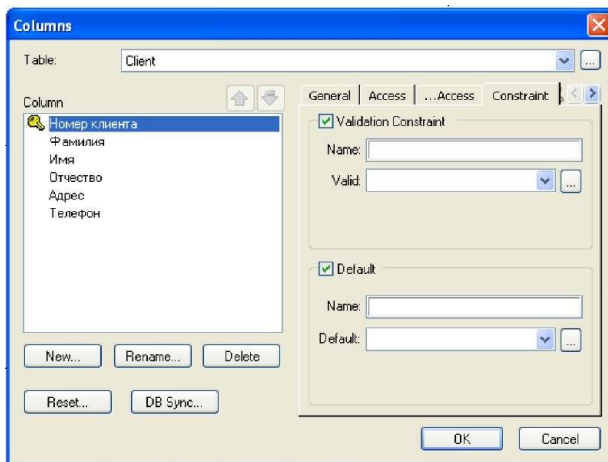


Рис. 12. Вкладка **General**

Вкладка, соответствующая выбранной СУБД, называется по имени СУБД и позволяет задать тип данных, опцию **NULL** и значение по умолчанию (см. рис. 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. 13).


 Рис. 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. 14).


 Рис. 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 **Q** 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. 15).

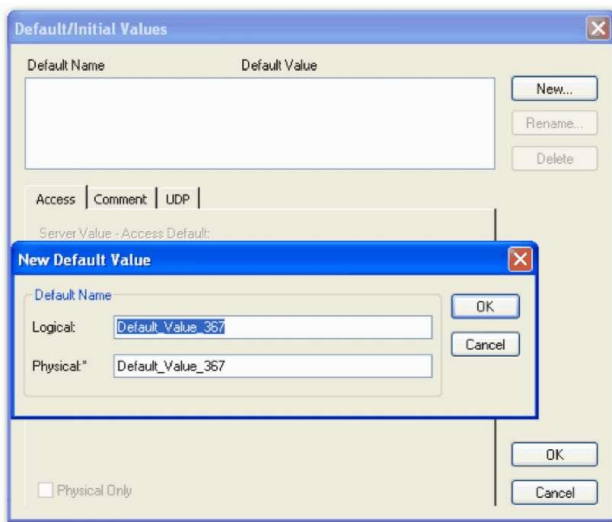
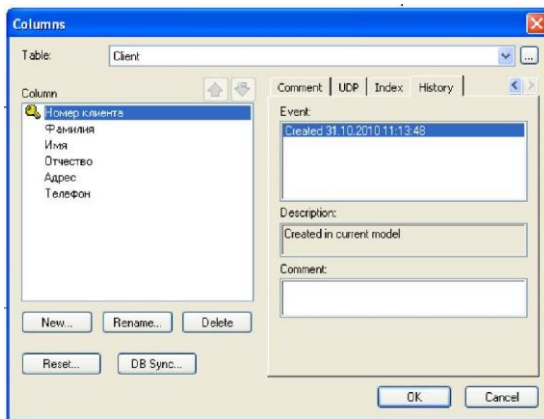




Рис. 15. Создания нового значения ограничения по умолчанию

Список всех заданных имен значений по умолчанию отображается в поле **DefaultName** 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. 16).


 Рис. 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 соответствующие 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 **DBSync** 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.

4.2.4 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 **NumbermanagerID** и XIF1 Менеджер - внешний ключ, в состав которого войдет колонка **DepartmentnumberFK**.

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. 17). Редактор **Indexes** позволяет изменить имя индекса и его определение, а также порядок сортировки данных.

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

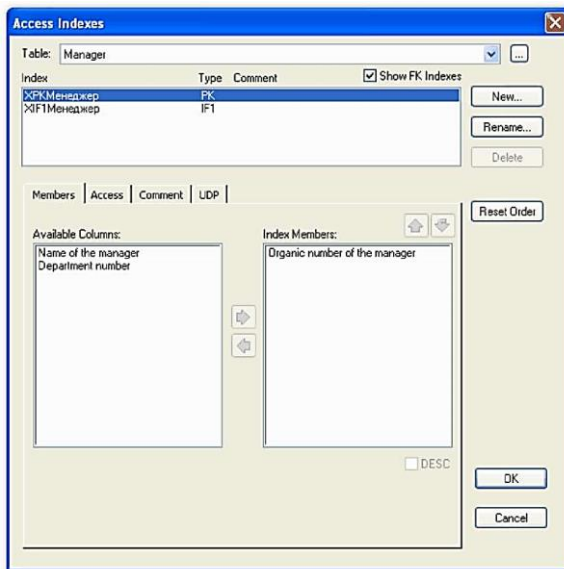


Рис. 17. Редактор **Indexes**

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

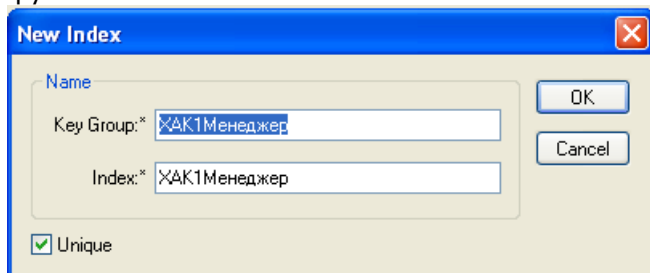


Рис. 18. Создание нового уникального индекса

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 **RenameIndex**, открывающийся по активизации кнопки **Rename** (рис. 19).

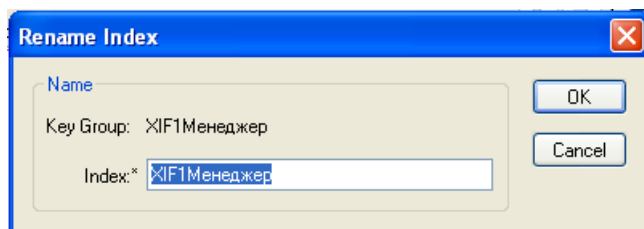


Рис. 19. Диалог **RenameIndex**

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, MSAccess, MSSQLServer, 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 **DepartmentnumberFK**. 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.

4.2.5 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 **ValidationRules**, который открывается через пункт главного меню **Model/ ValidationRules**, или через диалоговые окна, открывающиеся в следующем порядке:

- 1) активизировать кнопку "...", расположенную справа от раскрывающегося списка **Table** диалогового окна **Columns** (рис. 17);
- 2) в открывшемся окне выбрать закладку **Validation** и активизировать кнопку **ValidationConstraint**. (рис. 20).
- 3)

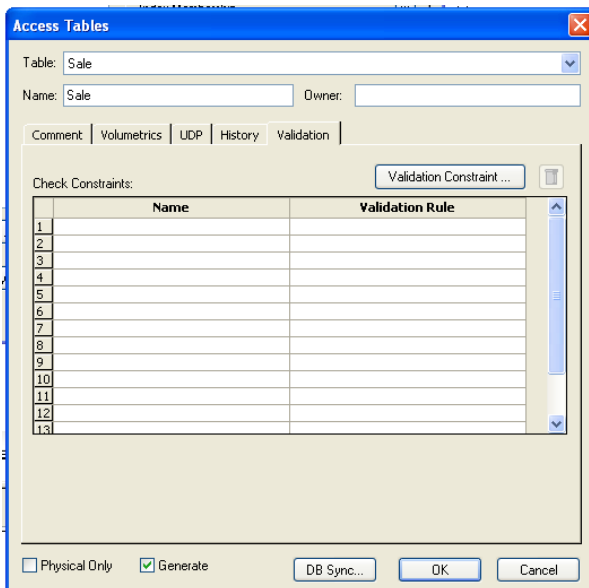


Рис. 7.20. ВыборзакладкиValidationиактивизациякнопки ValidationConstraint...

В результате открывается диалог **ValidationRules**, 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. 21).

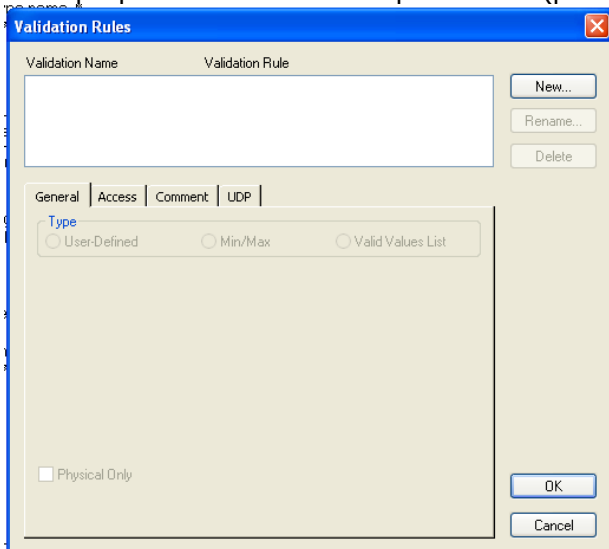


Рис. 7.21. Диалог **ValidationRules**

Для создания нового правила валидации необходимо активизировать кнопку **New**, ввести имя правила в поле **ValidationRuleName** диалога **NewValidationRule** и активизировать кнопку **OK** (рис. 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**.

Информационные системы в строительстве

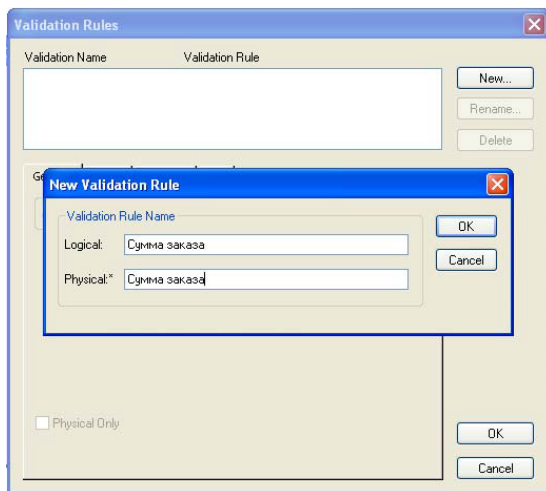


Рис. 22. Создание нового правила валидации

В результате в верхней части редактора **ValidationRules** отобразится список всех созданных правил валидации и представится возможность редактирования правил валидации (рис. 23).

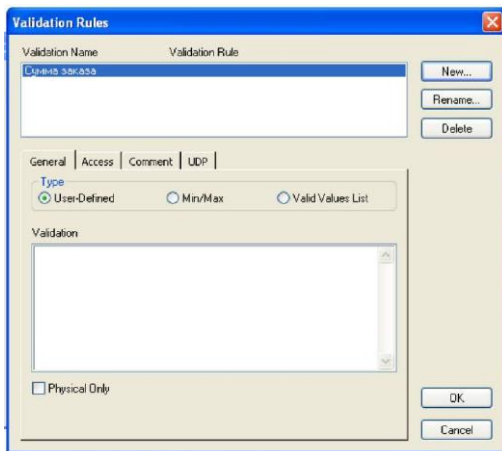


Рис. 23. Создание нового правила валидации

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 **ValidValuesList** позволяет задать список допустимых значений.

Пример правила валидации типа **Min/Max** для колонки **Сумма заказа** таблицы **Заказ** приведен на рис. 24.

Вкладка, соответствующая выбранной СУБД (на рис. 24 - **Access**) позволяет просмотреть фрагмент SQL - выражения, соответствующего правилу валидации, которое гарантирует проверку вводимых значений (рис. 25). В случае выхода за границы заданного диапазона СУБД выдает сообщение об ошибке.

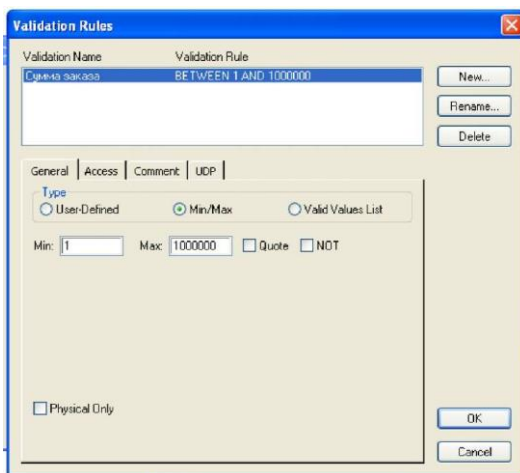


Рис. 24. Пример правила валидации типа **Min/Max**

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

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

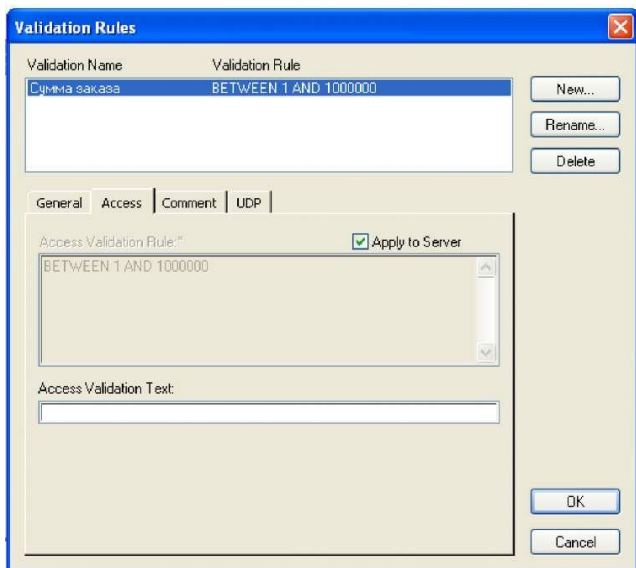


Рис. 25. Просмотр фрагмента SQL- выражения через вкладку **Access**

4.2.6 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. 26.

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

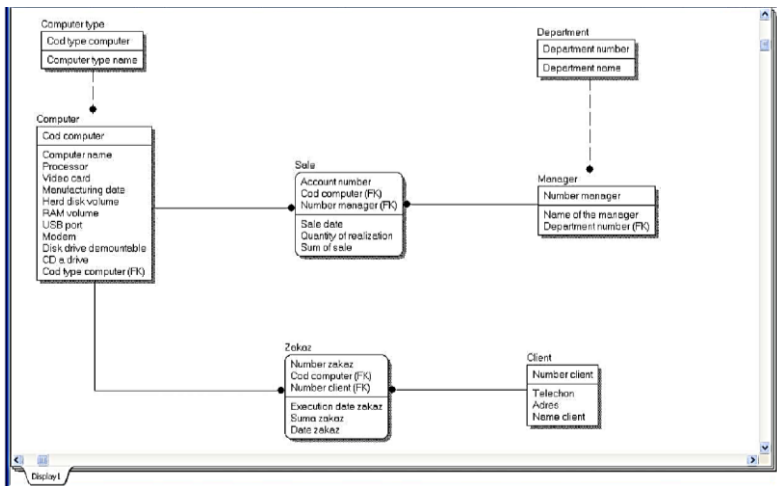


Рис. 7.26. Физическая модель данных системы "Реализация средств вычислительной техники"

Генерация кода создания базы данных

Для генерации кода создания базы данных можно использовать пункт главного меню **Tools/ForwardEngineer**. В результате откроется окно установки свойств генерируемой схемы данных (рис. 27).

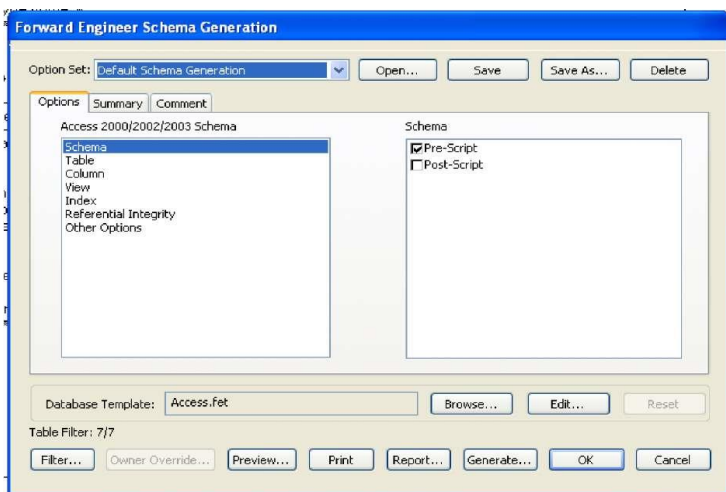


Рис. 27. Окно установки свойств генерируемой схемы данных

Для предварительного просмотра SQL-скрипта служит кнопка **Preview** (рис. 28), для генерации схемы - кнопка **Generate** (рис. 29).

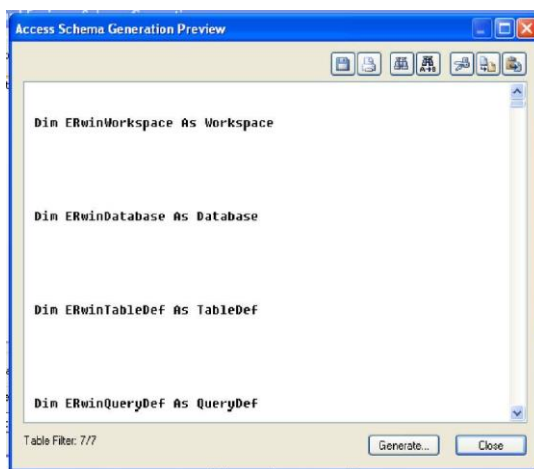


Рис. 28. Окно просмотра SQL-скрипта

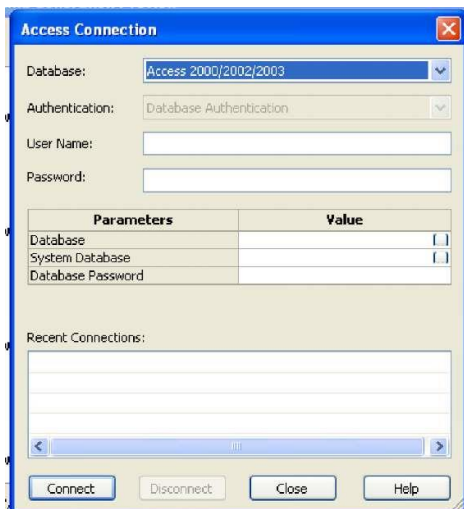


Рис. 29. Окно генерации схемы данных

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. 30).

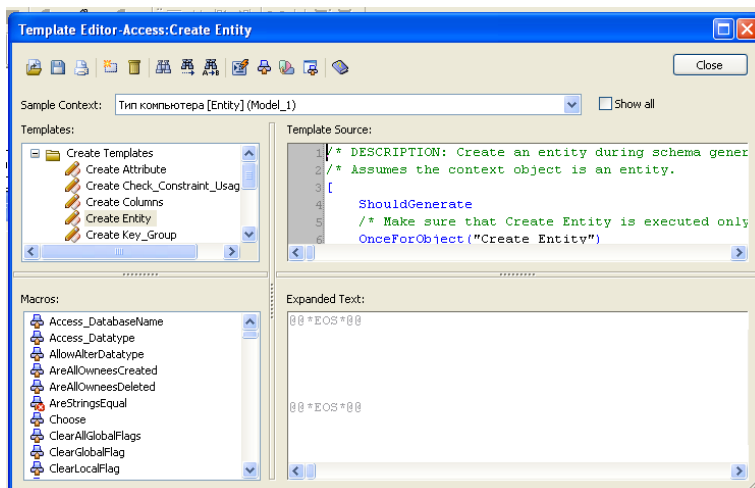


Рис. 30. Окно правки шаблона базы данных

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 CAERwinDataModeler 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.

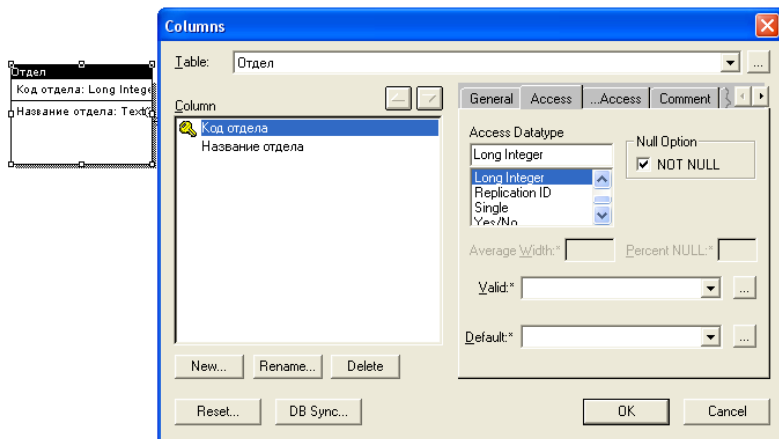
4.2.7 Упражнение 2: Создание физической модели с помощью ERwin

Откройте модель «Сотрудники»

1. На панели инструментов выберите **физическую** модель. **Physical.**

Для сущности "Отдел " задайте следующие типы атрибутов:

- Код отдела – Longinteger
- Название отдела – Text(30)
-



Для сущности " Должность " задайте следующие типы атрибутов:

- Код должности – Longinteger
- Название должности – Text(30)

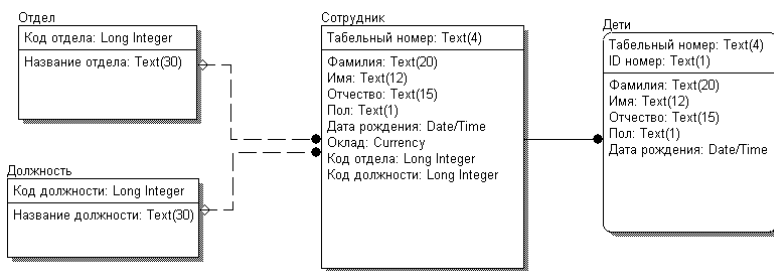
Для сущности " Дети сотрудников " задайте следующие типы атрибутов:

- ID номер– Text(1)
- Фамилия– Text(20)
- Имя– Text(12)
- Отчество– Text(15)

- Пол– Text(1)
- Дата рождения - Date/Time

Для сущности " Сотрудник" задайте следующие типы атрибутов:

- Табельный номер– Text(4)
- Фамилия– Text(20)
- Имя– Text(12)
- Отчество– Text(15)
- Пол– Text(1)
- Дата рождения - Date/Time
- Оклад – Currency



4.2.8 Упражнение 3

Создание базы данных


1. Запустите программу **Access**.

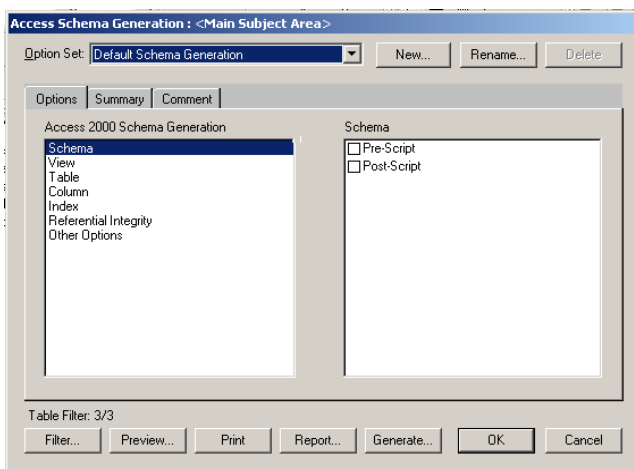
Создайте новую базу данных, и сохраните в собственной папке.

Закройте программу **Access**.

2. Запустите программу **ERwin**

Откройте физическую модель «Сотрудники» (Упражнение 2, Л.р. 2).

При помощи кнопки  (ForwardEngineer) на панели инструментов проведите прямое проектирование. В появившемся окне выберите Generate.



В окне AccessConnection в поле UserName установите имя пользователя admin. В поле Database укажите путь к созданному в Access файлу базы данных. Нажмите кнопку Connect.

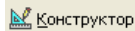


Начнется процесс генерации базы данных. В появившемся окне вы можете следить за ходом процесса. По окончании процесса генерации нажмите ОК.

3. Запустите программу **Access**. Откройте созданную вами базу данных.

Убедитесь в том, что на основе сущностей были созданы таблицы в базе данных.

Откройте таблицы в режиме конструктора (кнопка




панели инструментов), и убедитесь в том, что

- на основе атрибутов были созданы поля в таблицах;
- тип данных длина поля соответствует типу данных и

длине поля в ERD модели

– ключевые поля соответствует ключевым полям, выбранным в ERD модели

Откройте схему данных (кнопка  панели инструментов), и убедитесь в наличии связей между таблицами, связи соответствуют связям, установленным при создании модели в Erwin.

4.2.9 Контрольные вопросы

1. Охарактеризуйте построение физической модели данных. При каких условиях ERWin преобразование логической модели в физическую происходит автоматически?
2. Средства построения/редактирования ER-модели физического уровня. Назначение вкладок диалога Tables.
3. Технология определения свойств колонок таблицы физической модели данных. Назначение вкладок диалога Columns.
4. Индексы в физической модели. Принципы генерации имени индекса, созданного на основе ключа.
5. Уникальные индексы и повторяющиеся значения. Технология создания нового индекса.
6. Хешированный (кластеризованный) индекс.
7. Правила валидации колонок. Технология задания правил валидации.
8. Назначение закладок диалога ValidationRules. Типы правил.
9. Технология генерации кода создания базы данных.
10. Преимущества применения программного продукта CA ERwinDataModeler.

5 СПИСОК ЛИТЕРАТУРЫ

1. Смирнова Г.Н., Сорокин А.А., Тельнов Ю.Ф. «Проектирование экономических информационных систем», М.: Финансы и статистика, 2003. – 510 с.
2. Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite, М.: «Диалог МИФИ», 2003 427с.
3. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем, М.: Финансы и статистика, 1998. – 176 с.
4. Анисимова Г.Б., Романенко М.В. Выбор методологии проектирования информационных систем. I. Критерии. // Научное обозрение. 2014. № 12-2. с. 539-542.
5. Анисимова Г.Б., Романенко М.В. Выбор методологии проектирования информационных систем. II. Стандарты. // Научное обозрение. 2014. № 12-2. с. 543-547