

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ПРИОРИТЕТНЫЙ НАЦИОНАЛЬНЫЙ ПРОЕКТ «ОБРАЗОВАНИЕ»
КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. А.Н.ТУПОЛЕВА

А.Ю.АЛЕКСАНДРОВ

ПРОЕКТИРОВАНИЕ И АРХИТЕКТУРА ПРОГРАММНЫХ СИСТЕМ

Методические указания к лабораторным работам

КАЗАНЬ
2016

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1 ПОСТРОЕНИЕ ДИАГРАММ В ALLFUSION PROCESS MODELER	4
Цель работы	4
Принципы построения модели IDEF0	4
Диаграммы IDEF0	7
Работы (Activity)	8
Стрелки (Arrow)	11
Нумерация работ и диаграмм	16
Каркас диаграммы	17
Создание отчетов в Process Modeler	19
Задание на лабораторную работу	20
Создание контекстной диаграммы	20
 ЛАБОРАТОРНАЯ РАБОТА №2 ПОСТРОЕНИЕ МОДЕЛИ IDEF0 С ПОМОЩЬЮ ALLFUSION PROCESS MODELER	 23
Цель работы	23
Декомпозиция	23
Задание на лабораторную работу	25
Декомпозиция контекстной диаграммы	25
Декомпозиция работы "Сборка и тестирование компьютеров"	29
 ЛАБОРАТОРНАЯ РАБОТА №3 ПОСТРОЕНИЕ ДИАГРАММ DFD И IDEF3	 34
Цель работы	34
Диаграммы DFD	34
Принципы построения модели IDEF3	38
Диаграммы IDEF3	39
Задание на лабораторную работу	46
Декомпозиция работы "Продажи и маркетинг"	46

Построение DFD диаграммы	47
Создание диаграммы IDEF3	49
ЛАБОРАТОРНАЯ РАБОТА №4 СОЗДАНИЕ МОДЕЛИ ДАННЫХ С ПОМОЩЬЮ ALLFUSION ERWIN DATA MODELER	52
Цель работы	52
Отображение модели данных в ERwin	52
Физическая и логическая модель данных	52
Интерфейс ERwin. Уровни отображения модели	54
Подмножества модели и сохраняемые отображения	56
Создание логической модели данных	57
Уровни логической модели	57
Связи	63
Задание на лабораторную работу	77
ЛАБОРАТОРНАЯ РАБОТА №5 СОЗДАНИЕ ЛОГИЧЕСКОЙ МОДЕЛИ ДАННЫХ	78
Цель работы	78
Типы сущностей и иерархия наследования	78
Ключи	82
Нормализация данных	87
Домены	93
Задание на лабораторную работу	98
ЛАБОРАТОРНАЯ РАБОТА №6 СОЗДАНИЕ ФИЗИЧЕСКОЙ МОДЕЛИ ДАННЫХ	99
Цель работы	99
Уровни физической модели	99
Выбор сервера	99
Таблицы, колонки и представления (view)	100
Правила валидации и значения по умолчанию	109
Индексы	112
Триггеры и хранимые процедуры	115
Прямое проектирование	118

Лабораторная работа №1

Построение диаграмм в Allfusion Process Modeler

Цель работы

Знакомство с CASE-системой Allfusion Process Modeler, изучение основных принципов построения диаграмм IDEF0, разработка контекстной диаграммы.

Принципы построения модели IDEF0

В IDEF0 система представляется как совокупность взаимодействующих работ или функций. Под моделью в IDEF0 понимают описание системы (текстовое и графическое), которое должно дать ответ на некоторые заранее определенные вопросы.

Процесс моделирования какой-либо системы в IDEF0 начинается с определения контекста, т. е. наиболее абстрактного уровня описания системы в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель.

Под субъектом понимается сама система, при этом необходимо точно установить, что входит в систему, а что лежит за ее пределами, другими словами, мы должны определить, что мы будем в дальнейшем рассматривать как компоненты системы, а что как внешнее воздействие. На определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования - вопросы, на которые построенная модель должна дать ответ. Другими словами, первоначально необходимо определить область (Scope) моделирования. При формулировании области необходимо учитывать два компонента - широту и глубину. Широта подразумевает определение границ модели - мы определяем, что будет рассматриваться внутри системы, а что снаружи. Глубина определяет, на каком уровне детализации модель является завершенной.

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

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

Формулировка цели позволяет команде аналитиков сфокусировать усилия в нужном направлении. Примерами формулирования цели могут быть следующие утверждения: "Идентифицировать и определить текущие проблемы, сделать возможным анализ потенциальных улучшений", "Идентифицировать роли и ответственность служащих для написания должностных инструкций", "Описать функциональность предприятия с целью написания спецификаций информационной системы" и т. д.

Точка зрения (Viewpoint). Хотя при построении модели учитываются мнения различных людей, модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в Process Modeler следует выбрать пункт меню **Edit/Model Properties**, вызывающий диалог Model Properties (Рис. 1). В закладке Purpose следует внести цель и точку зрения, а в закладку Definition - определение модели и описание области.

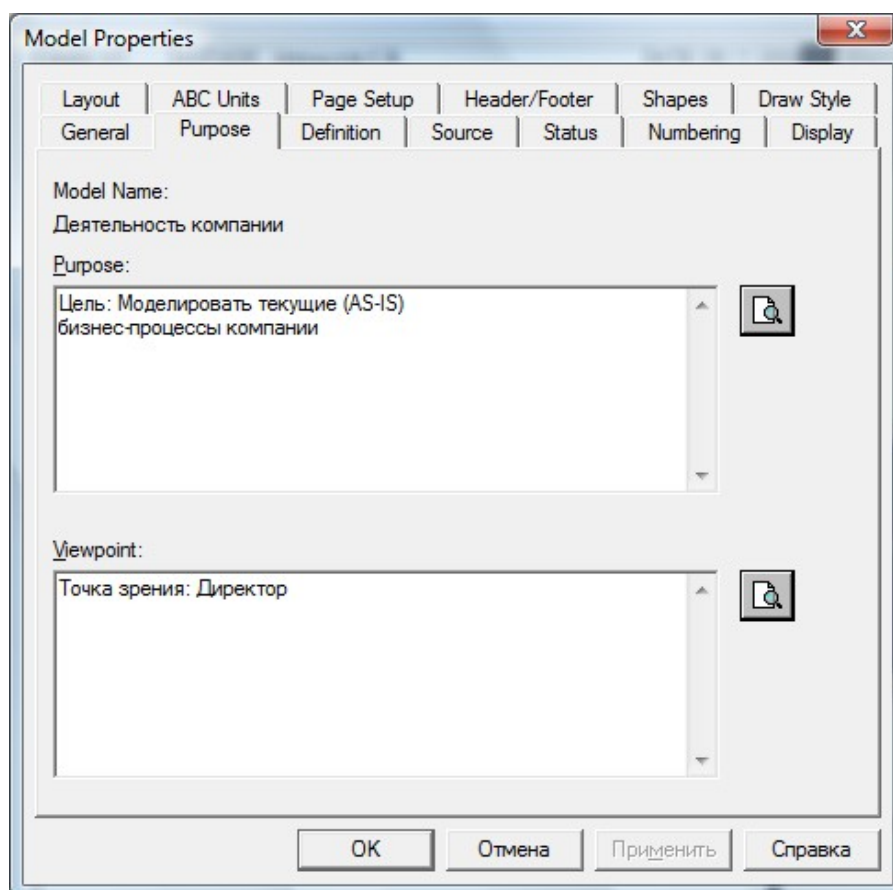


Рис. 1 Диалог задания свойств модели

В закладке Status того же диалога можно описать статус модели (черновой вариант, рабочий, окончательный и т. д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате). В закладке Source описываются источники информации для построения модели (например, "Опрос экспертов предметной области и анализ документации"). Закладка General служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок модели - AS-IS и TO-BE.

Результат описания модели можно получить в отчете Model Report. Диалог настройки отчета по модели вызывается из пункта меню **Report/Model Report**. В диалоге настройки следует выбрать необходимые поля, при этом автоматически отображается очередность вывода информации в отчет (Рис. 2).

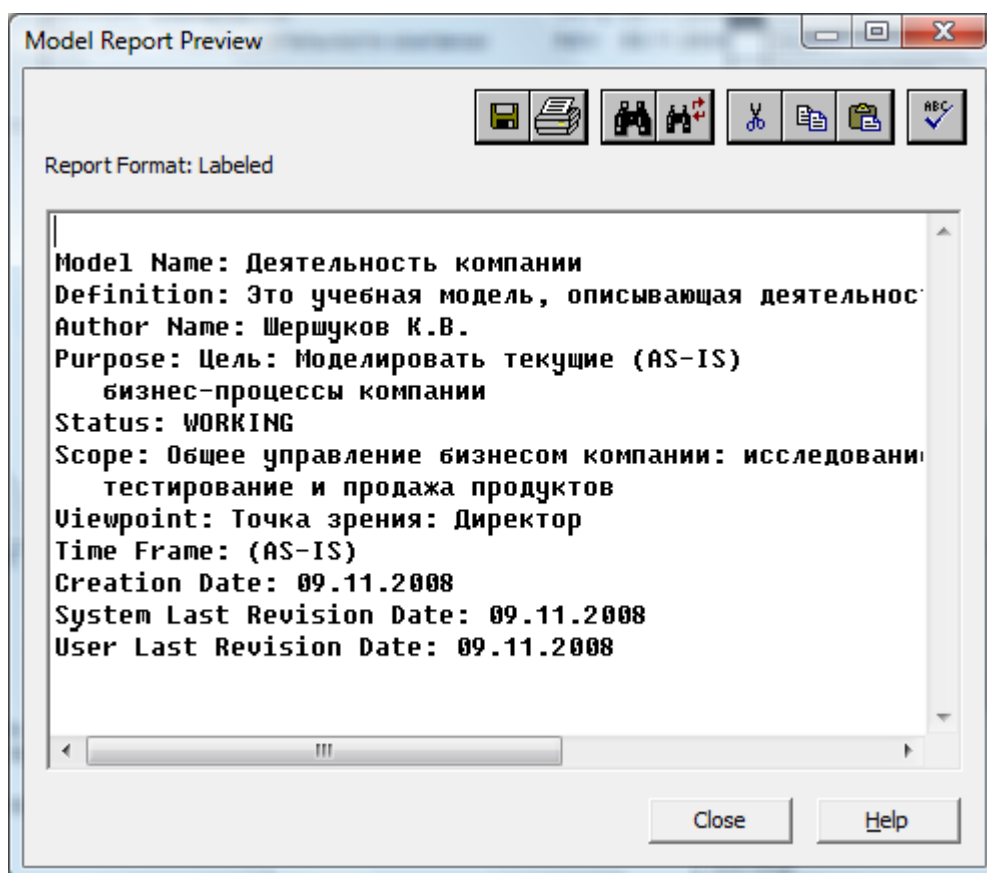


Рис. 2. Отчет по модели

Диаграммы IDEF0

Модель может содержать четыре типа диаграмм:

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

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого

большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколь угодно много, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

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

Работы (Activity)

Работы обозначают поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Работы изображаются в виде прямоугольников. Все работы должны быть названы и определены. Имя работы должно быть выражено отглагольным существительным, обозначающим действие (например, *"Изготовление детали"*, *"Прием заказа"* и т.д.). Работа *"Изготовление детали"* может иметь, например, следующее определение: "Работа относится к полному циклу изготовления изделия от контроля качества сырья до отгрузки готового упакованного изделия". При создании новой модели (меню File/New) автоматически создается контекстная диаграмма с единственной работой, изображающей систему в целом (Рис. 4).

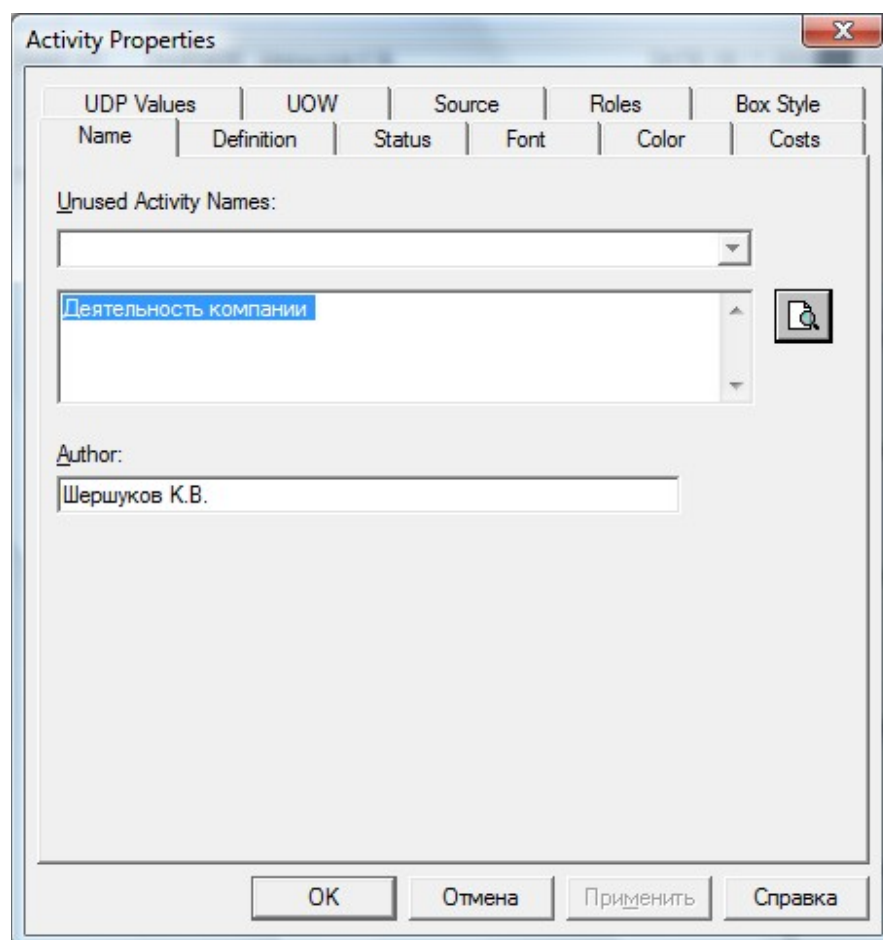


Рис. 3. Редактор задания свойств работы

Для внесения имени работы следует щелкнуть по работе правой кнопкой мыши, выбрать в меню Name Editor и в появившемся диалоге внести имя работы. Для описания других свойств работы служит диалог Activity Properties (Рис. 3).

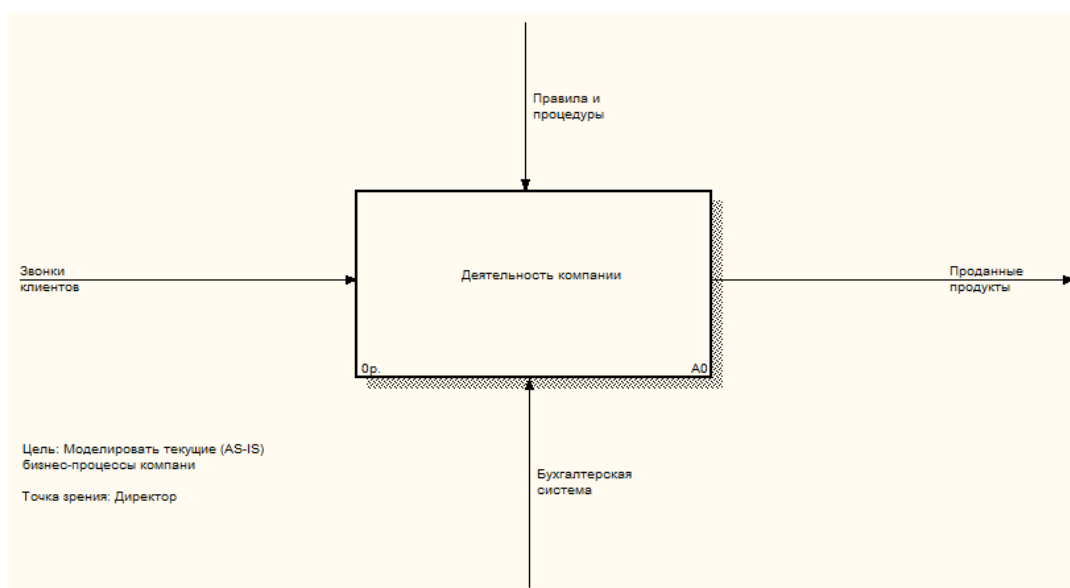


Рис. 4. Пример контекстной диаграммы

Диаграммы декомпозиции содержат родственные работы, т.е. дочерние работы, имеющие общую родительскую работу. Для создания диаграммы декомпозиции следует щелкнуть по кнопке

Возникает диалог Activity Box Count (Рис. 5), в котором следует указать нотацию новой диаграммы и количество работ на ней. Выберем нотацию IDEF0 и щелкнем на ОК. Появляется диаграмма декомпозиции (Рис. 6). Рекомендуемый интервал числа работ 3-6.

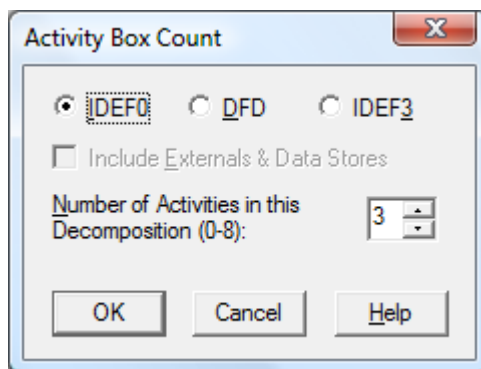


Рис. 5. Диалог Activity Box Count

Если оказывается, что количество работ недостаточно, то работу можно добавить в диаграмму, щелкнув сначала по кнопке

на палитре инструментов, а затем по свободному месту на диаграмме.

Работы на диаграммах декомпозиции обычно располагаются по диагонали от левого верхнего угла к правому нижнему.

Такой порядок называется порядком доминирования. Согласно этому принципу расположения в левом верхнем углу располагается самая важная работа или работа, выполняемая по времени первой. Далее вправо вниз располагаются менее важные или выполняемые позже работы. Такое расположение облегчает чтение диаграмм, кроме того, на нем основывается понятие взаимосвязей работ (см. ниже).

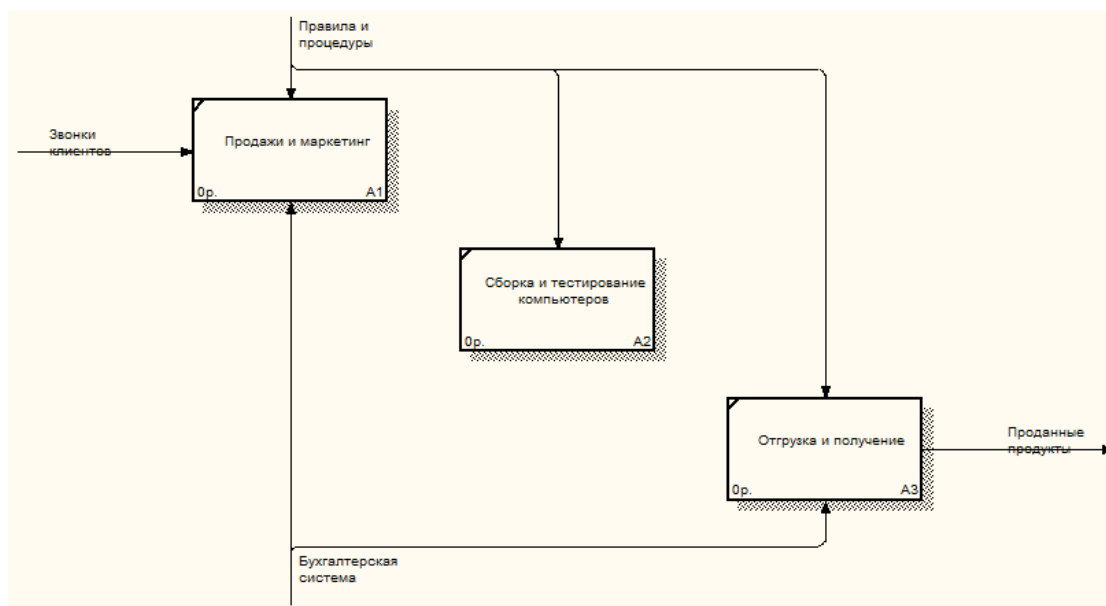


Рис. 6. Пример диаграммы декомпозиции

Каждая из работ на диаграмме декомпозиции может быть в свою очередь декомпозирована. На диаграмме декомпозиции работы нумеруются автоматически слева направо. Номер работы показывается в правом нижнем углу. В левом верхнем углу изображается небольшая диагональная черта, которая показывает, что данная работа не была декомпозирована.

Стрелки (Arrow)

Взаимодействие работ с внешним миром и между собой описывается в виде стрелок. Стрелки представляют собой некую информацию и именуются существительными (например, "*Заготовка*", "*Изделие*", "*Заказ*").

В IDEF0 различают пять типов стрелок:

Вход (Input) - материал или информация, которые используются или преобразуются работой для получения результата (выхода). Допускается, что работа может не иметь ни одной стрелки входа. Стрелка входа рисуется как входящая в левую грань работы.

Управление (Control) - правила, стратегии, процедуры или стандарты, которыми руководствуется работа. Каждая работа должна иметь хотя бы одну стрелку управления. Стрелка управления рисуется как входящая в верхнюю грань работы. На Рис. 4 стрелки "*Задание*" и "*Чертеж*" - управление


для работы **"Изготовление изделия"**. Управление влияет на работу, но не преобразуется работой.

Выход (Output) - материал или информация, которые производятся работой. Каждая работа должна иметь хотя бы одну стрелку выхода. Работа без результата не имеет смысла и не должна моделироваться. Стрелка выхода рисуется как исходящая из правой грани работы. На Рис. 4 стрелка **"Готовое изделие"** является выходом для работы **"Изготовление изделия"**.

Механизм (Mechanism) - ресурсы, которые выполняют работу, например персонал предприятия, станки, устройства и т. д. Стрелка механизма рисуется как входящая в нижнюю грань работы. На Рис. 4 стрелка **"Персонал предприятия"** является механизмом для работы **"Изготовление изделия"**. По усмотрению аналитика стрелки механизма могут не изображаться в модели.

Граничные стрелки. Стрелки на контекстной диаграмме служат для описания взаимодействия системы с окружающим миром. Они могут начинаться у границы диаграммы и заканчиваться у работы, или наоборот. Такие стрелки называются граничными.

Для внесения граничной стрелки входа следует:

щелкнуть по кнопке с символом стрелки 

в палитре инструментов перенести курсор к левой стороне экрана, пока не появится начальная штриховая полоска;

щелкнуть один раз по полоске (откуда выходит стрелка) и еще раз в левой части работы со стороны входа (где заканчивается стрелка);

вернуться в палитру инструментов и выбрать указатель 

щелкнуть правой кнопкой мыши на линии стрелки, во всплывающем меню выбрать Name Editor и добавить имя стрелки в закладке Name диалога IDEF0 Arrow Properties.

Стрелки управления, выхода, механизма и выхода изображаются аналогично.

Имена вновь внесенных стрелок автоматически заносятся в словарь (Arrow Dictionary).

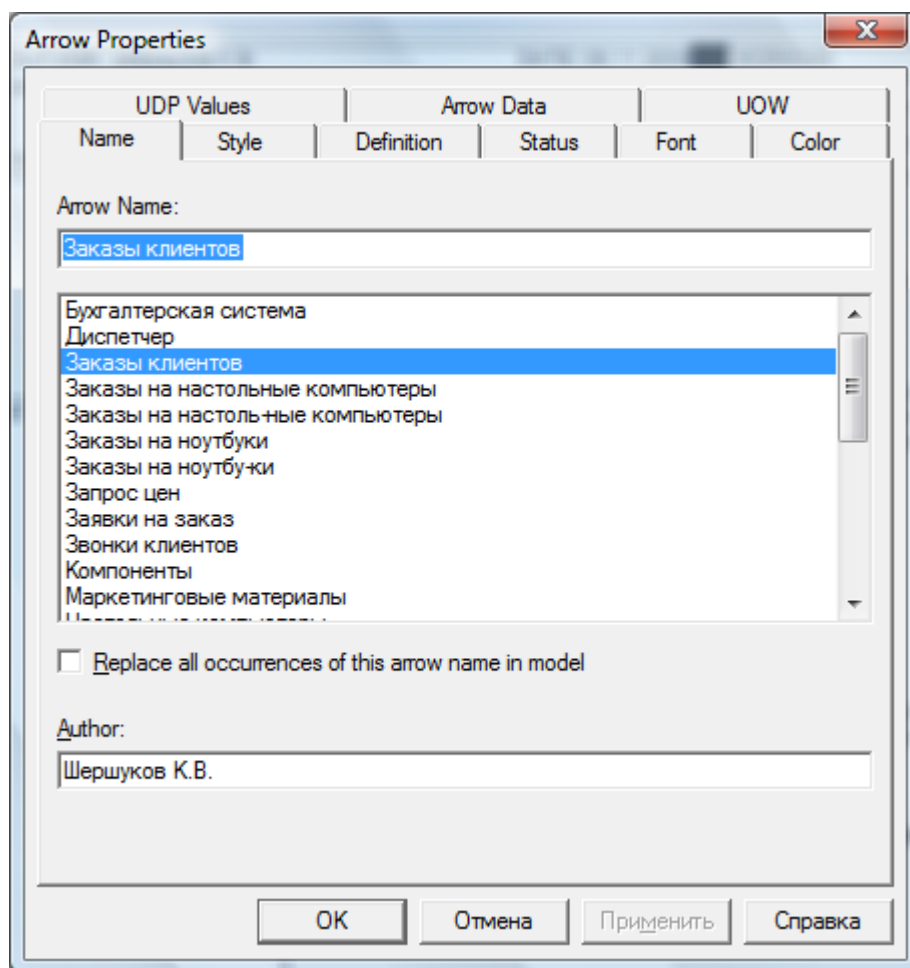


Рис. 7. Диалог IDEF0 Arrow Properties

ICOM-коды. Диаграмма декомпозиции предназначена для детализации работы. Работа на диаграмме верхнего уровня в IDEF0 - это не элемент управления нижестоящими работами. Работы нижнего уровня - это то же самое, что работы верхнего уровня, но в более детальном изложении. Как следствие этого границы работы верхнего уровня - это то же самое, что границы диаграммы декомпозиции. **ICOM** (аббревиатура от **I**nput, **C**ontrol, **O**utput и **M**echanism) - коды, предназначенные для идентификации граничных стрелок. Код **ICOM** содержит префикс, соответствующий типу стрелки (**I**, **C**, **O** или **M**), и порядковый номер (Рис. 8).

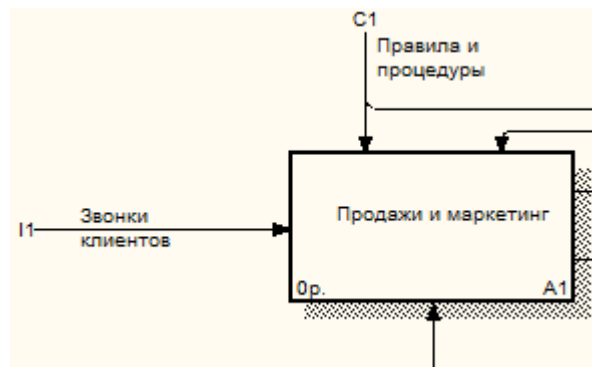


Рис. 8. Фрагмент диаграммы декомпозиции с ICOM-кодами (I1, C1)

Process Modeler вносит ICOM-коды автоматически. Для отображения ICOM-кодов следует включить опцию Show ICOM codes на закладке Presentation диалога Model Properties (меню Edit/Model Properties).

Словарь **стрелок** редактируется при помощи специального редактора Arrow Dictionary Editor, в котором определяется стрелка и вносится относящийся к ней комментарий. Словарь стрелок решает очень важную задачу. Диаграммы создаются аналитиком для того, чтобы провести сеанс экспертизы, т. е. обсудить диаграмму со специалистом предметной области. В любой предметной области формируется профессиональный жаргон, причем очень часто жаргонные выражения имеют нечеткий смысл и воспринимаются разными специалистами по-разному. В то же время аналитик - автор диаграмм должен употреблять те выражения, которые наиболее понятны экспертам. Поскольку формальные определения часто сложны для восприятия, аналитик вынужден употреблять профессиональный жаргон, а, чтобы не возникло неоднозначных трактовок, в словаре стрелок каждому понятию можно дать расширенное и, если это необходимо, формальное определение.

Содержимое словаря стрелок можно распечатать в виде отчета (меню Report/Arrow Report...) и получить тем самым толковый словарь терминов предметной области, использующихся в модели.

Несвязанные граничные стрелки (unconnected border arrow). При декомпозиции работы входящие в нее и исходящие из нее стрелки (кроме стрелки вызова) автоматически появляются на диаграмме декомпозиции (миграция стрелок), но при этом не касаются работ. Такие стрелки называются

несвязанными и воспринимаются в Process Modeler как синтаксическая ошибка.

Для связывания стрелок входа, управления или механизма необходимо перейти в режим редактирования стрелок, щелкнуть по конечнику стрелки и щелкнуть по соответствующему сегменту работы. Для связывания стрелки выхода необходимо перейти в режим редактирования стрелок, щелкнуть по сегменту выхода работы и затем по стрелке.

Внутренние стрелки. Для связи работ между собой используются внутренние стрелки, т.е. стрелки, которые не касаются границы диаграммы, начинаются у одной и кончаются у другой работы.

Для рисования внутренней стрелки необходимо в режиме рисования стрелок щелкнуть по сегменту (например, выхода) одной работы и затем по сегменту (например, входа) другой. В IDEF0 различают пять типов связей работ.

Связь по входу (output-input), когда стрелка выхода вышестоящей работы (далее - просто выход) направляется на вход нижестоящей.

Связь по управлению (output-control), когда выход вышестоящей работы направляется на управление нижестоящей. Связь по управлению показывает доминирование вышестоящей работы. Данные или объекты выхода вышестоящей работы не меняются в нижестоящей.

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

Обратная связь по управлению (output-control feedback), когда выход нижестоящей работы направляется на управление вышестоящей. Обратная связь по управлению часто свидетельствует об эффективности бизнес - процесса.

Связь выход-механизм (output-mechanism), когда выход одной работы направляется на механизм другой. Эта взаимосвязь используется реже

остальных и показывает, что одна работа подготавливает ресурсы, необходимые для проведения другой работы.

Явные стрелки. Явная стрелка имеет источником одну-единственную работу и назначением тоже одну-единственную работу.

Разветвляющиеся и сливающиеся стрелки. Одни и те же данные или объекты, порожденные одной работой, могут использоваться сразу в нескольких других работах. С другой стороны, стрелки, порожденные в разных работах, могут представлять собой одинаковые или однородные данные или объекты, которые в дальнейшем используются или перерабатываются в одном месте. Для моделирования таких ситуаций в IDEF0 используются разветвляющиеся и сливающиеся стрелки. Для разветвления стрелки нужно в режиме редактирования стрелки щелкнуть по фрагменту стрелки и по соответствующему сегменту работы. Для слияния двух стрелок выхода нужно в режиме редактирования стрелки сначала щелкнуть по сегменту выхода работы, а затем по соответствующему фрагменту стрелки.

Нумерация работ и диаграмм

Все работы модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс А. Контекстная (корневая) работа дерева имеет номер А0. Работы декомпозиции А0 имеют номера А1, А2, А3 и т. д. Работы декомпозиции нижнего уровня имеют номер родительской работы и очередной порядковый номер, например работы декомпозиции А3 будут иметь номера А31, А32, А33, А34 и т. д. Работы образуют иерархию, где каждая работа может иметь одну родительскую и несколько дочерних работ, образуя дерево. Такое дерево называют деревом узлов, а вышеописанную нумерацию - нумерацией по узлам.

Диаграммы IDEF0 имеют двойную нумерацию. Во-первых, диаграммы имеют номера по узлу. Контекстная диаграмма всегда имеет номер А-0, декомпозиция контекстной диаграммы - номер А0, остальные диаграммы декомпозиции - номера по соответствующему узлу (например, А1, А2, А21, А213 и т. д.). Process Modeler автоматически поддерживает нумерацию по

узлам, т. е. при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер. Кроме того, существует специальный номер - C-number, который должен присваиваться автором модели вручную. C-number - это произвольная строка, но рекомендуется придерживаться стандарта, когда номер состоит из буквенного префикса и порядкового номера, причем в качестве префикса используются инициалы автора диаграммы, а порядковый номер отслеживается автором вручную.

Каркас диаграммы

На Рис. 9 показан типичный пример диаграммы декомпозиции с граничными рамками, которые называются каркасом диаграммы.

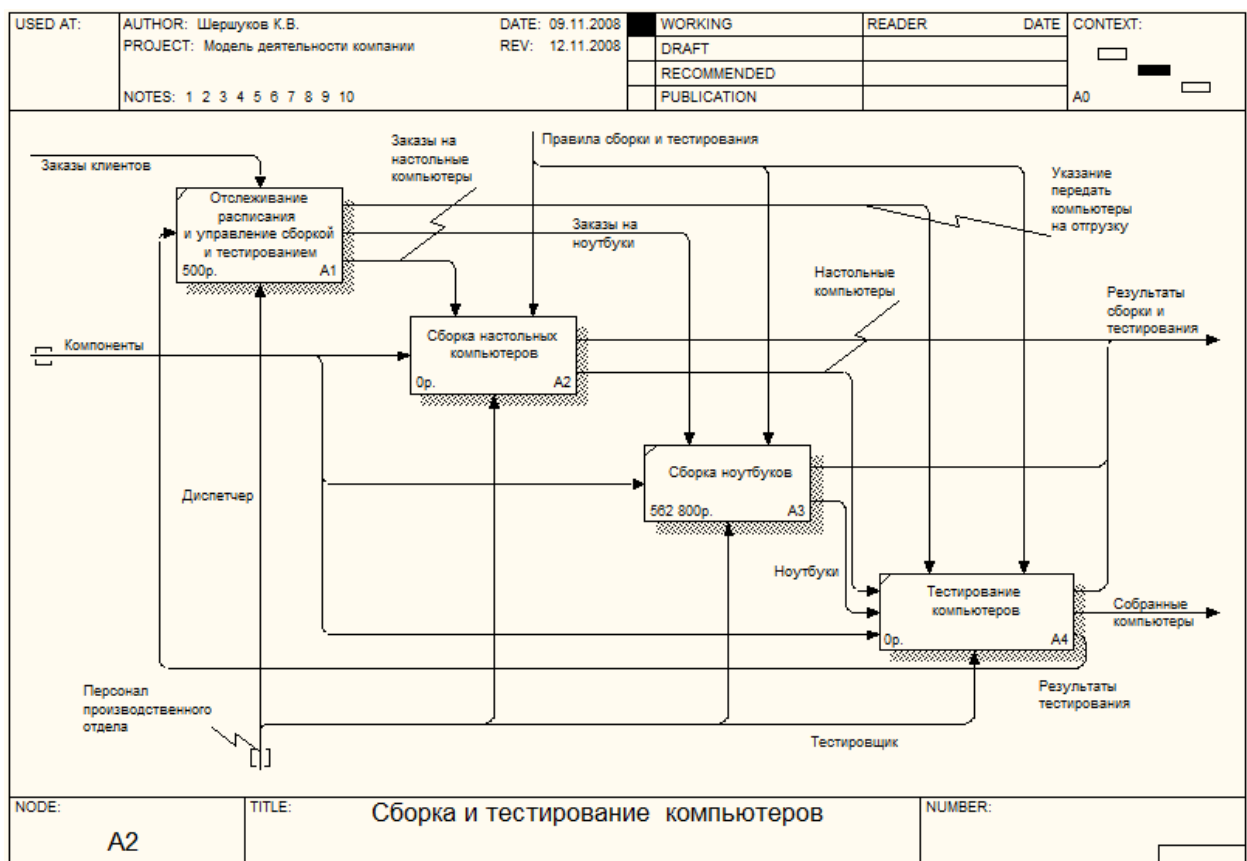


Рис. 9. Пример диаграммы декомпозиции с каркасом

Каркас содержит заголовок (верхняя часть рамки) и подвал (нижняя часть). Заголовок каркаса используется для отслеживания диаграммы в процессе моделирования. Нижняя часть используется для идентификации и позиционирования в иерархии диаграммы.

Смысл элементов каркаса приведен в Таблица 1 и Таблица 2.

Таблица 1. Поля заголовка каркаса (слева направо)

Поле	Смысл
Used At	Используется для указания на родительскую работу в случае, если на текущую диаграмму ссылались посредством стрелки вызова
Autor, Date, Rev, Prpject	Имя создателя диаграммы, дата создания и имя проекта, в рамках которого была создана диаграмма. REV-дата последнего редактирования диаграммы
Notes 1 2 3 4 5 6 7 8 9 10	Используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания
Status	Статус отображает стадию создания диаграммы, отображая все этапы публикации
Working	Новая диаграмма, кардинально обновленная диаграмма или новый автор диаграммы
Draft	Диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению
Recommended	Диаграмма и все ее сопровождающие документы прошли экспертизу. Новых изменений не ожидается
Publication	Диаграмма готова к окончательной печати и публикации
Reader	Имя читателя (эксперта)
Date	Дата прочтения (экспертизы)
Context	Схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На

	контекстной диаграмме (A-0) показана надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы.
--	--

Таблица 2. Поля подвала каркаса (слева направо)

Поле	Смысл
Node	Номер узла диаграммы (номер родительской работы)
Title	Имя диаграммы. По умолчанию - имя родительской работы
Number	C-Number, уникальный номер версии диаграммы
Page	Номер страницы, может использоваться как номер страницы при формировании палки

Создание отчетов в Process Modeler

Process Modeler имеет мощный инструмент генерации. Отчеты по модели вызываются из пункта меню Report. Всего имеется семь типов отчетов:

Model Report. Он включает информацию о контексте модели - имя модели, точку зрения, область, цель, имя автора, дату создания и др.

Diagram Report. Отчет по конкретной диаграмме. Включает список объектов (работ, стрелок, хранилищ данных, внешних ссылок и т. д.).

Diagram Object Report. Наиболее полный отчет по модели. Может включать полный список объектов модели (работ, стрелок с указанием их типа и др.) и свойства, определяемые пользователем.

Activity Cost Report. Отчет о результатах стоимостного анализа. Будет рассмотрен ниже.

Arrow Report. Отчет по стрелкам. Может содержать информацию из словаря стрелок, информацию о работе-источнике, работе-назначении стрелки и информацию о разветвлении и слиянии стрелок.

DataUsage Report. Отчет о результатах связывания модели процессов и модели данных.

Model Consistency Report. Отчет, содержащий список синтаксических ошибок модели.

Задание на лабораторную работу

Разработать контекстную диаграмму, создать отчет по модели в соответствии с инструкциями приведенными ниже.


Создание контекстной диаграммы

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 ИС, которая позволяет оформить заказ, счет и отследить платежи по счетам.

Щелкните по кнопке . Появляется диалог I would like to. Внесите имя модели "Деятельность компании" и выберите Type - IDEF0. Нажмите ОК.

Автоматически создается контекстная диаграмма.

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.

Перейдите в меню Model/Model Properties. Во вкладке General диалога Model Properties следует внести имя модели "Деятельность компании", имя

проекта "Модель деятельности компании", имя автора и тип модели - Time Frame: AS-IS.

Во вкладке Purpose внесите цель - "Цель: Моделировать текущие (AS-IS) бизнес-процессы компании" и точку зрения (Viewpoint) - "Точка зрения: Директор".

Во вкладке Definition внесите определение "Это учебная модель, описывающая деятельность компании" и цель "Scope: Общее управление бизнесом компании: исследование рынка, закупка компонентов, сборка, тестирование и продажа продуктов".

Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по работе. В контекстном меню выберите Name. Во вкладку Name внесите имя "Деятельность компании".

Во вкладке Definition внесите определение "Текущие бизнес-процессы компании".

Создайте стрелки на контекстной диаграмме (Таблица 3).

Таблица 3. Стрелки контекстной диаграммы

<i>Имя стрелки (Arrow Name)</i>	<i>Определение стрелки (Arrow Definition)</i>	<i>Тип стрелки (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

1. С помощью кнопки **T** внесите текст в поле диаграммы - точку зрения и цель.

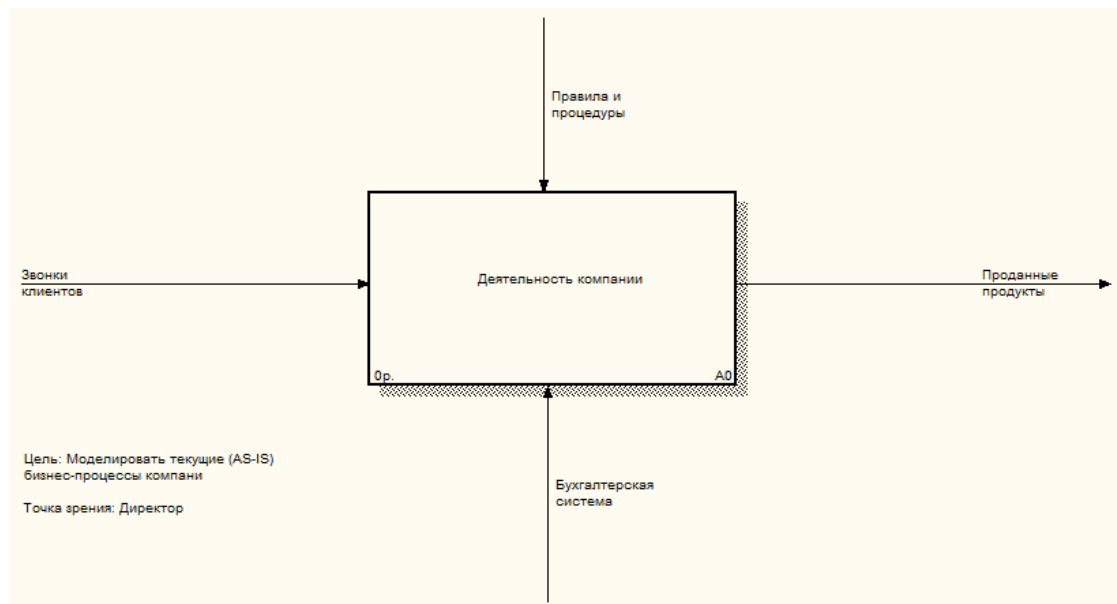


Рис. 10. Контекстная диаграмма

Результат выполнения упражнения 1 показан на Рис. 10. Создайте отчет по модели. Меню Tools/Reports/Model Report

Лабораторная работа №2

Построение модели IDEF0 с помощью Allfusion Process Modeler

Цель работы

Изучение приемов декомпозиции диаграмм IDEF0.

Декомпозиция

После описания системы в целом с помощью контекстной диаграммы проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы проводится декомпозиция каждого большого фрагмента системы на более мелкие и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы - эксперты предметной области указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступить к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели.

С помощью декомпозиции детализируется некоторая работа. Диаграммы IDEF0 могут быть декомпозированы как с помощью диаграмм IDEF0, так и с помощью диаграмм DFD и IDEF3.

В результате дополнения диаграмм IDEF0 диаграммами DFD и IDEF3 может быть создана смешанная модель, которая наилучшим образом описывает все стороны деятельности предприятия (Рис. 11). Иерархию работ в смешанной модели можно увидеть в окне Model Explorer. Работы в нотации IDEF0 изображаются зеленым цветом, IDEF3 - желтым, DFD - синим.

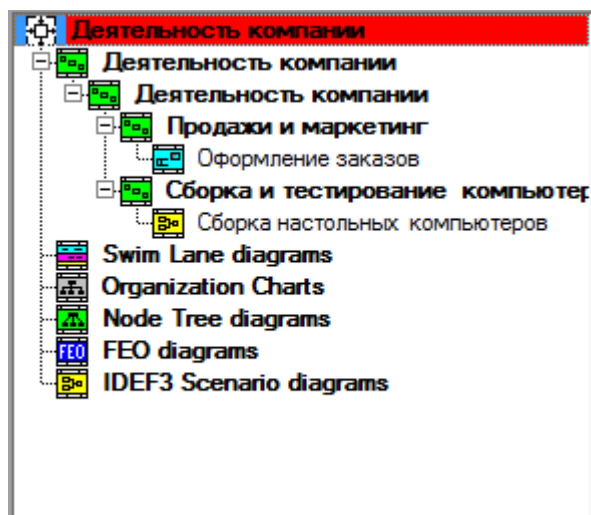


Рис. 11. Представление смешанной модели в окне Model Explorer

Рекомендации по рисованию диаграмм

В реальных диаграммах к каждой работе может подходить и от каждой может отходить около десятка стрелок. Если диаграмма содержит 6-8 работ, то она может содержать 30-40 стрелок, причем они могут сливаться, разветвляться и пресекаться. Такие диаграммы могут стать очень плохо читаемыми. В IDEF0 существуют соглашения по рисованию диаграмм, которые призваны облегчить чтение и экспертизу модели. Некоторые из этих правил Process Modeler поддерживает автоматически, выполнение других следует обеспечить вручную.

Прямоугольники работ должны располагаться по диагонали с левого верхнего в правый нижний угол (порядок доминирования). При создании новой диаграммы декомпозиции Process Modeler автоматически располагает работы именно в таком порядке. В дальнейшем можно добавить новые работы или изменить расположение существующих, но нарушать диагональное расположение работ по возможности не следует. Порядок доминирования подчеркивает взаимосвязь работ, позволяет минимизировать изгибы и пересечения стрелок.

Следует максимально увеличивать расстояние между входящими или выходящими стрелками на одной грани работы. Если включить опцию Line Drawing: Automatically space arrows на закладке Layout диалога Model Proper-

ties (меню Edit/Model Properties), Process Modeler будет располагать стрелки нужным образом автоматически.

Следует максимально увеличить расстояние между работами, поворотами и пересечениями стрелок.

Если две стрелки проходят параллельно (начинаются из одной и той же грани одной работы и заканчиваются на одной и той же грани другой работы), то по возможности следует их объединить и назвать единым термином.

Обратные связи по входу рисуются "нижней" петлей, обратная связь по управлению - "верхней". Process Modeler автоматически рисует обратные связи нужным образом.

Циклические обратные связи следует рисовать только в случае крайней необходимости, когда подчеркивают значение повторно используемого объекта.

Следует минимизировать число пересечений, петель и поворотов стрелок. Это ручная и, в случае насыщенных диаграмм, творческая работа

Задание на лабораторную работу

Декомпозируйте контекстную диаграмму и работу «Сборка и тестирование компьютеров в соответствии с инструкциями приведенными ниже.

Декомпозиция контекстной диаграммы

1. Выберите кнопку перехода на нижний уровень в палитре инструментов и в диалоге Activity Box Count установите число работ на диаграмме нижнего уровня - 3 - и нажмите ОК.

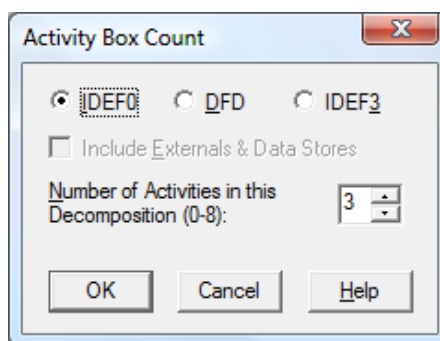




Рис. 12. Диалог Activity Box Count


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 Таблица 4.

Таблица 4. Работы диаграммы декомпозиции A0

<i>Имя работы (Activity)</i>	<i>Определение (Definition)</i>
Продажи и маркетинг	Телемаркетинг и презентации, выставки
Сборка и тестирование компьютеров	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.

Если описать имя и свойства работы в словаре, ее можно будет внести в диаграмму позже с помощью кнопки  в палитре 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  (Purge).

3. Перейдите в режим рисования стрелок. Свяжите граничные стрелки (кнопка  на палитре инструментов, так, как показано на Рис. 13.

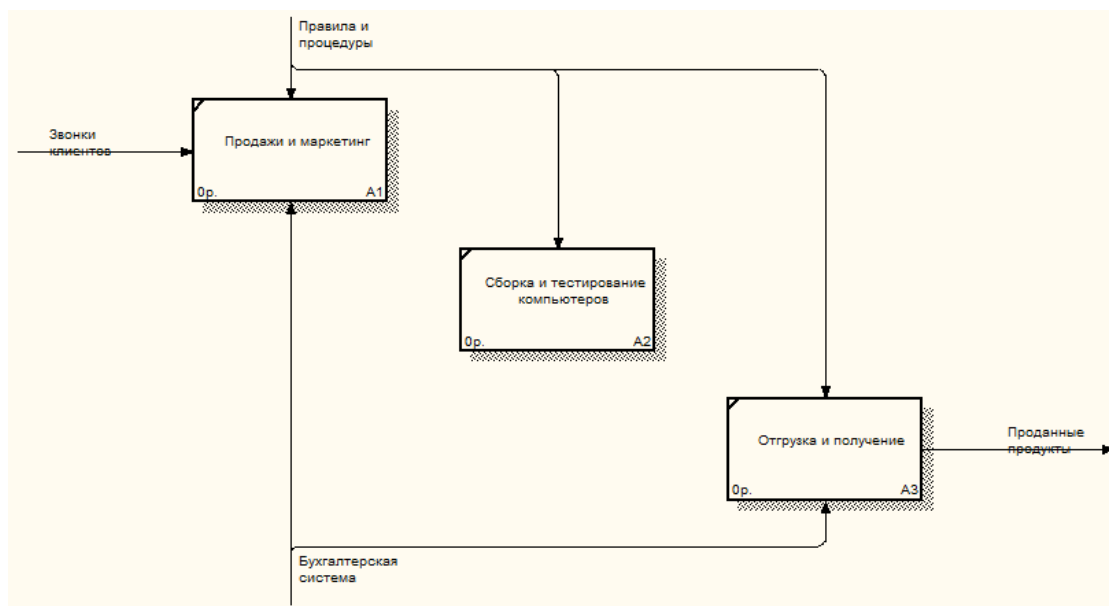


Рис. 13. Связанные граничные стрелки на диаграмме A0

4. Правой кнопкой мыши щелкните по ветви стрелки управления работы "Сборка и тестирование компьютеров" и переименуйте ее в "Правила сборки и тестирования".

Внесите определение для новой ветви: "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.

Создайте новые внутренние стрелки так, как показано на Рис. 14.

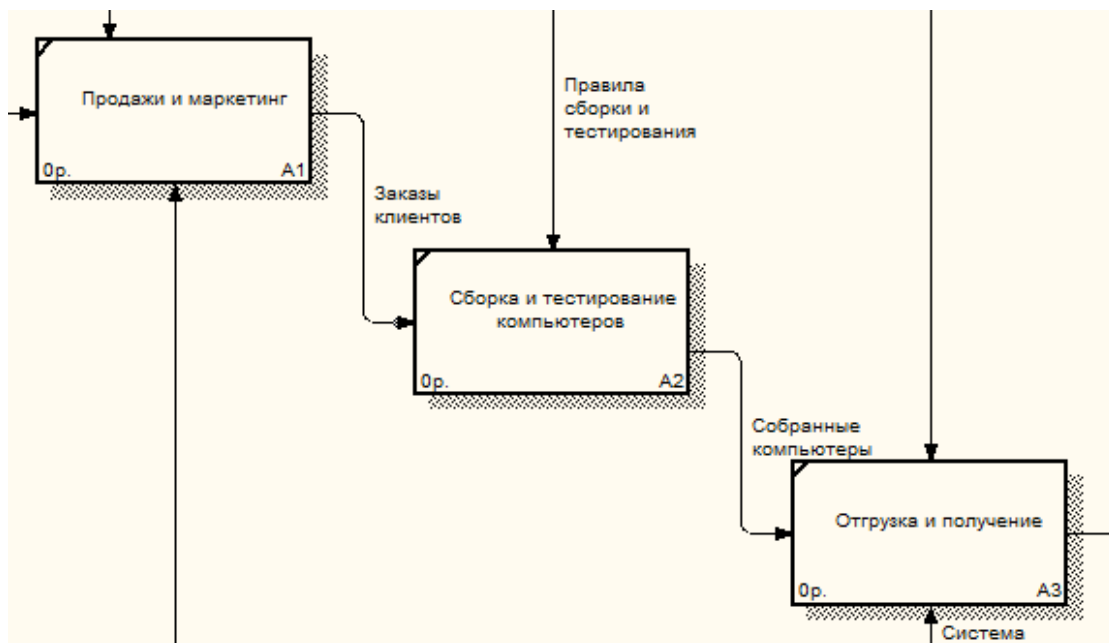


Рис. 14. Внутренние стрелки диаграммы АО

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. 15.

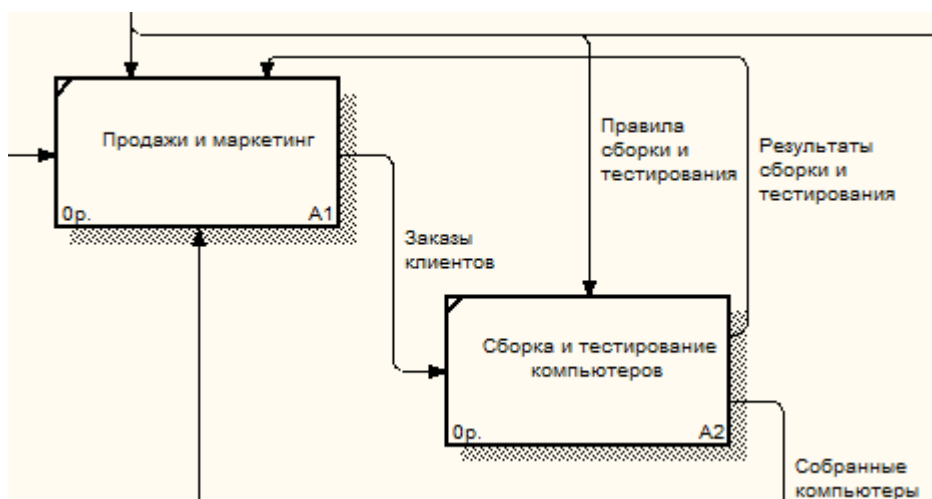


Рис. 15. Результат редактирования стрелок на диаграмме АО

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 конце. Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню Arrow Tunnel. В диалоге 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. 16.

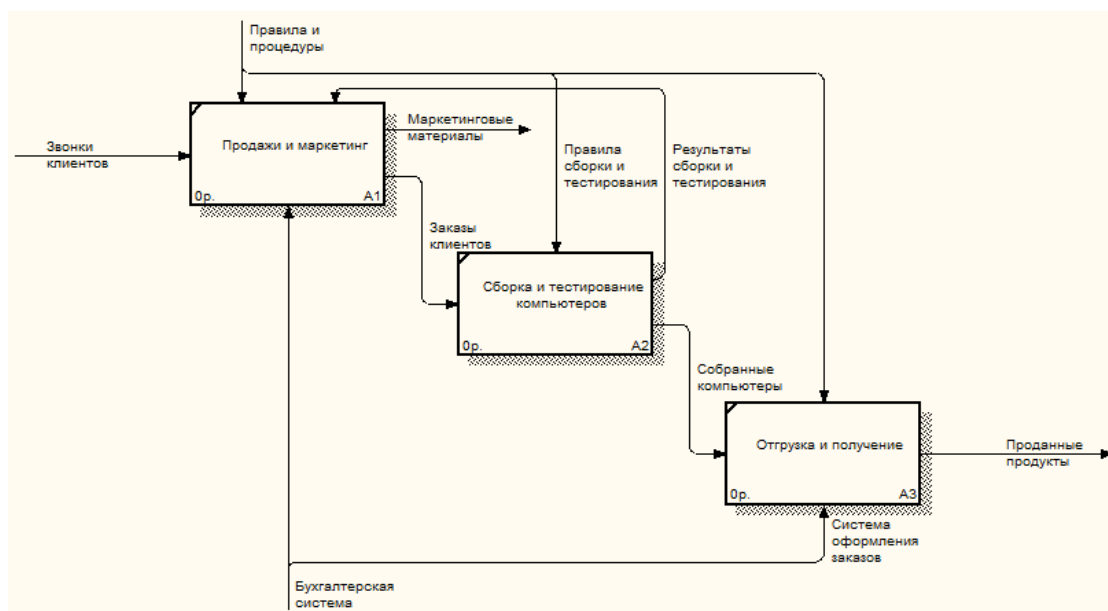


Рис. 16. Результат выполнения упражнения 2 - диаграмма АО

Декомпозиция работы "Сборка и тестирование компьютеров"

В результате проведения экспертизы получена следующая информация.

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

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

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

Сотрудники участка сборки собирают компьютеры согласно спецификациям заказа и инструкциям по сборке. Когда группа компьютеров, соответствующая группе заказов, собрана, она направляется на тестирование. Тестировщики тестируют каждый компьютер и в случае необходимости заменяют неисправные компоненты.

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

1. На основе этой информации внесите новые работы и стрелки (Таблица 5 и Таблица 6).

Таблица 5. Работы диаграммы декомпозиции A2

<i>Имя работы (Activity Name)</i>	<i>Определение работы (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 6. Стрелки диаграммы декомпозиции A2

Имя стрелки (Arrow Name)	Источник стрелки (Arrow Source)	Тип источника стрелки (Arrow Source Type)	Назначение стрелки (Arrow Dest.)	Тип назначения стрелки (Arrow Dest. Type)
Диспетчер	Персонал производственного отдела		Отслеживание расписания и управление сбор-	Mechanism
Заказы клиентов	Граница диаграммы	Control	Отслеживание расписания и управление сбор-	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	Input
			Сборка ноутбуков	Input
			Тестирование компьютеров	Input
Настольные компьютеры	Сборка настольных компьютеров	Output	Тестирование компьютеров	Input
Ноутбуки	Сборка ноутбуков	Output	Тестирование компьютеров	Input
Персонал производственного отдела	"Tunnel"	Mechanism	Сборка настольных компьютеров	Mechanism
			Сборка ноутбуков	Mechanism
Правила сборки и тестирования	Граница диаграммы		Сборка настольных компьютеров	Control
			Сборка ноутбуков	Control
			Тестирование компьютеров	Control

Результаты сборки и тестирования	Сборка настольных компьютеров	Output	Граница диаграммы	Output
	Сборка ноутбуков	Output		
	Тестирование компьютеров	Output		
Результаты тестирования	Тестирование компьютеров	Output	Отслеживание расписания и управление сборкой и тестированием	Input
Собранные компьютеры	Тестирование компьютеров	Output	Граница диаграммы	Output
Тестирующий	Персонал производственного отдела		Тестирование компьютеров	Mechanism
Указание передать компьютеры на отгрузку	Отслеживание расписания и управление сборкой и тестированием	Output	Тестирование компьютеров	Control

2. Тоннелируйте и свяжите на верхнем уровне граничные стрелки, если это необходимо. Результат выполнения упражнения 3 показан на Рис. 17.

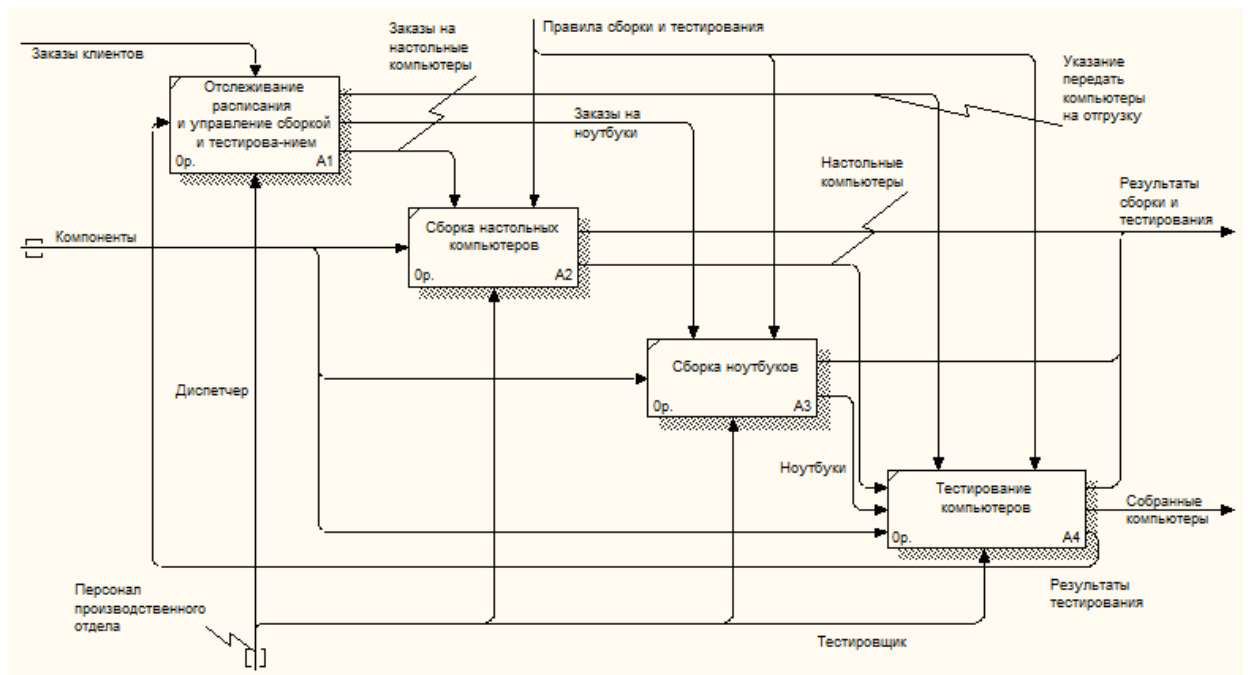


Рис. 17. Результат выполнения упражнения 3

Лабораторная работа №3

Построение диаграмм DFD и IDEF3

Цель работы

Изучение основных принципов построения диаграмм DFD и IDEF3, разработка диаграмм.

Диаграммы DFD

Диаграммы потоков данных (Data flow diagramming, DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывает:

функции обработки информации (работы);

документы (стрелки, arrow), объекты, сотрудников или отделы, которые участвуют в обработке информации;

внешние ссылки (external references), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;

таблицы для хранения документов (хранилище данных, data store).

В Process Modeler для построения диаграмм потоков данных используется нотация Гейна-Сарсона.

Для того чтобы дополнить модель IDEF0 диаграммой DFD, нужно в процессе декомпозиции в диалоге Activity Box Count “кликнуть” по радио-кнопке DFD. В палитре инструментов на новой диаграмме DFD появляются новые кнопки:

– добавить в диаграмму внешнюю ссылку (External Reference). Внешняя ссылка является источником или приемником данных извне модели;

– добавить в диаграмму хранилище данных (Data store). Хранилище данных позволяет описать данные, которые необходимо сохранить в памяти прежде, чем использовать в работах;

– ссылка на другую страницу. В отличие от IDEF0 инструмент offpage reference позволяет направить стрелку на любую диаграмму (а не только на верхний уровень).

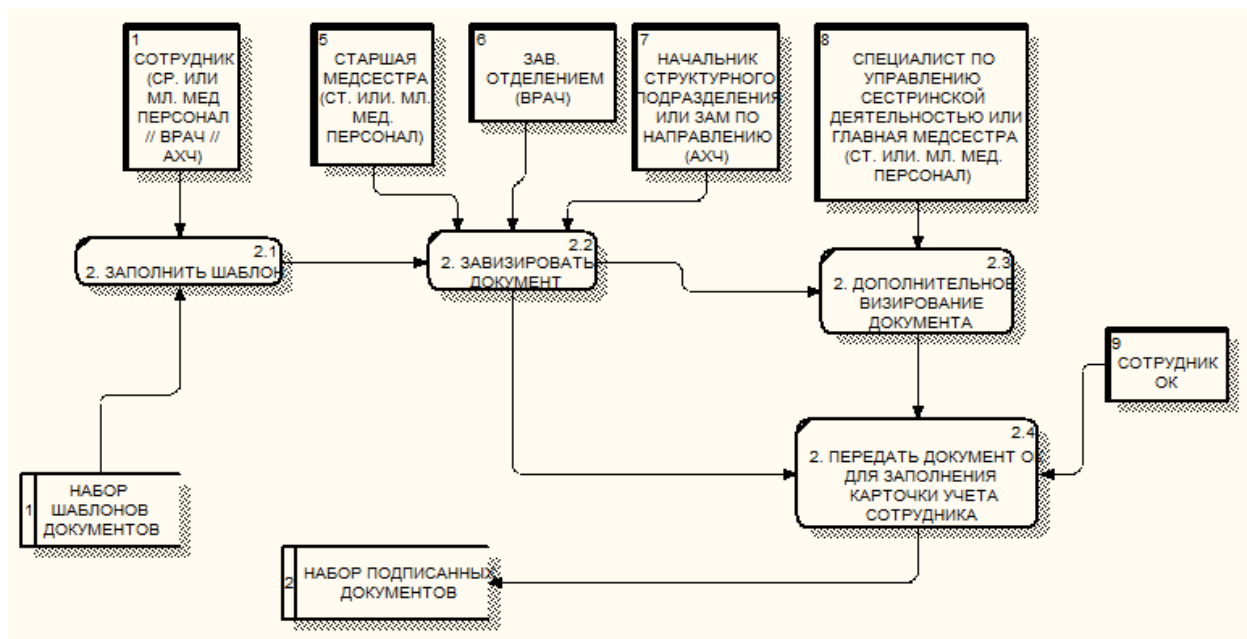


Рис. 18. Пример диаграммы DFD

В отличие от стрелок IDEF0, которые представляют собой жесткие взаимосвязи, стрелки DFD показывают, как объекты (включая данные) движутся от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на физические характеристики системы - движение объектов (data flow), хранение объектов (data stores), поставка и распространение объектов (external entities) (Рис. 18).

В отличие от IDEF0, где система рассматривается как взаимосвязанные работы, DFD рассматривает систему как совокупность предметов. Контекстная диаграмма часто включает работы и внешние ссылки. Работы обычно именуются по названию системы, например "Система обработки информации". Включение внешних ссылок в контекстную диаграмму не отменяет тре-

бования методологии четко определить цель, область и единую точку зрения на моделируемую систему.

Работы. В DFD работы представляют собой функции системы, преобразующие входы в выходы. Хотя работы изображаются прямоугольниками со скругленными углами, смысл их совпадает со смыслом работ IDEF0 и IDEF3. Так же как работы IDEF3, они имеют входы и выходы, но не поддерживают управления и механизмы, как IDEF0.

Внешние сущности. Внешние сущности изображают входы в систему и/или выходы из системы. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах. Обычно такой прием используют, чтобы не рисовать слишком длинных и запутанных стрелок.

Стрелки (Потоки данных). Стрелки описывают движение объектов из одной части системы в другую. Поскольку в DFD каждая сторона работы не имеет четкого назначения, как в IDEF0, стрелки могут подходить и выходить из любой грани прямоугольника работы. В DFD также применяются двунаправленные стрелки для описания диалогов типа "команда-ответ" между работами, между работой и внешней сущностью и между внешними сущностями (Рис. 19).

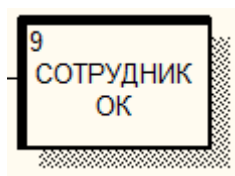


Рис. 19. Внешняя сущность

Хранилище данных. В отличие от стрелок, описывающих объекты в движении, хранилища данных изображают объекты в покое (Рис. 24).

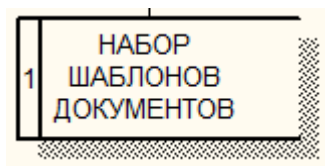


Рис. 20. Хранилище данных

В материальных системах хранилища данных изображаются там, где объекты ожидают обработки, например в очереди. В системах обработки информации хранилища данных являются механизмом, который позволяет сохранить данные для последующих процессов.

Слияние и разветвление стрелок. В DFD стрелки могут сливаться и разветвляться, что позволяет описать декомпозицию стрелок. Каждый новый сегмент сливающейся или разветвляющейся стрелки может иметь собственное имя.

Построение диаграмм DFD. Диаграммы DFD могут быть построены с использованием традиционного структурного анализа, подобно тому как строятся диаграммы IDEF0. Сначала строится физическая модель, отображающая текущее состояние дел. Затем эта модель преобразуется в логическую модель, которая отображает требования к существующей системе. После этого строится модель, отображающая требования к будущей системе. И наконец, строится физическая модель, на основе которой должна быть построена новая система.

Альтернативным подходом является подход, популярный при создании программного обеспечения, называемый событийным разделением (event partitioning), в котором различные диаграммы DFD выстраивают модель системы. Во-первых, логическая модель строится как совокупность работ и документирования того, что они (эти работы) должны делать.

Затем модель окружения (environment model) описывает систему как объект, взаимодействующий с событиями из внешних сущностей. Модель , окружения обычно содержит описание цели системы, одну контекстную диаграмму и список событий. Контекстная диаграмма содержит один прямоугольник работы, изображающий систему в целом, и внешние сущности, с которыми система взаимодействует.

Наконец, модель поведения (behavior model) показывает, как система обрабатывает события. Эта модель состоит из одной диаграммы, в которой каждый прямоугольник изображает каждое событие из модели окружения. Хра-

нилища могут быть добавлены для моделирования данных, которые необходимо запоминать между событиями. Потоки добавляются для связи с другими элементами, и диаграмма проверяется с точки зрения соответствия модели окружения.

Полученные диаграммы могут быть преобразованы с целью более наглядного представления системы, в частности работы на диаграммах могут быть декомпозированы.

Нумерация объектов. В DFD номер каждой работы может включать префикс, номер родительской работы (A) и номер объекта. Номер объекта -это уникальный номер работы на диаграмме. Например, работа может иметь номер A.12.4. Уникальный номер имеют хранилища данных и внешние сущности независимо от их расположения на диаграмме. Каждое хранилище данных имеет префикс D и уникальный номер, например D5. Каждая внешняя сущность имеет префикс E и уникальный номер, например E5.

Принципы построения модели IDEF3

Наличие в диаграммах DFD элементов для описания источников, приемников и хранилищ данных позволяет более эффективно и наглядно описать процесс документооборота. Однако для описания логики взаимодействия информационных потоков более подходит IDEF3, называемая также *workflow diagramming* - методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 - это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе.

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим процесс действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть задокументирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо задокументировать цель модели - те вопросы, на которые призвана ответить модель.


Диаграммы IDEF3

Диаграммы. Диаграмма является основной единицей описания в IDEF3. Важно правильно построить диаграммы, поскольку они предназначены для чтения другими людьми (а не только автором).

Единицы работы - Unit of Work (UOW). UOW, также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия,

одиноким или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы (например, *"Изготовление изделия"*). Часто имя существительное в имени работы меняется в процессе моделирования, поскольку модель может уточняться и редактироваться. Идентификатор работы присваивается при создании и не меняется никогда. Даже если работа будет удалена, ее идентификатор не будет вновь использоваться для других работ. Обычно номер работы состоит из номера родительской работы и порядкового номера на текущей диаграмме.

Связи. Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи, стиль которых устанавливается через меню Edit/Arrow Style:

Предшествование (Precedence)  - сплошная линия, связывающая единицы работ (UOW). Рисуются слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.


Отношения (Relational Link)

- пунктирная линия, используемая для изображения связей между единицами работ (UOW) а также между единицами работ и объектами ссылок.

Потоки объектов (Object Flow)


- стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой. Часто результатом работы-источника становится объект, необходимый для запуска работы-цели. В этом случае стрелку, обозначающую объект, изображают с двойным наконечником. Имя стрелки должно ясно идентифицировать отображаемый объект. Поток объектов имеет ту же семантику, что и стрелка предшествования.





Отношение показывает, что стрелка является альтернативой стрелке предшествования или потоку объектов в смысле задания последовательности выполнения работ - работа-источник не обязательно должна закончиться, прежде чем работа-цель начнется. Более того, работа-цель может закончиться прежде, чем закончится работа-источник.

Перекрестки (Junction). Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. Для внесения перекрестка служит кнопка  (добавить в диа-1рамму перекресток -Junction) в палитре инструментов. В диалоге Junction Type Editor необходимо указать тип перекрестка.


Смысл каждого типа приведен в Таблица 7.

Таблица 7. Типы перекрестков

Обозначение	Наименование	Смысл в случае слияния стрелок (Fan-in Junction)	Смысл в случае разветвления стрелок (Fan-out Junction)
	Asynchronous AND	Все предшествующие процессы должны быть завершены	Все следующие процессы должны быть запущены

	Synchronous AND	Все предше- ствующие про- цессы завершены одновременно	Все следующие процессы запус- каются одновремен- но
	Asynchronous OR	Один или несколько пред- шествующих процессов долж- ны быть заверше- ны	Один или несколько следую- щих процессов должны быть запу- щены
	Synchronous OR	Один или несколько пред- шествующих процессов завер- шены одновре- менно	Один или несколько следую- щих процессов запускаются од- новременно
	XOR (Exclusive OR)	Только один предшествующий процесс завер- шен	Только один следующий процесс запускается

Все перекрестки на диаграмме нумеруются, каждый номер имеет префикс J. Можно редактировать свойства перекрестка при помощи диалога Definition Editor. В отличие от IDEF0 и DFD в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки.

Объект ссылки. Объект ссылки в IDEF3 выражает некую идею, концепцию или данные, которые нельзя связать со стрелкой, перекрестком или работой (Рис. 21). Для внесения объекта ссылки служит кнопка  (добавить в диаграмму объект ссылки - Referent) в палитре инструментов. Объект ссылки изображается в виде прямоугольника, похожего на прямоугольник работы. Имя объекта ссылки задается в диалоге Referent (пункт всплывающего меню

Name Editor), в качестве имени можно использовать имя какой-либо стрелки с других диаграмм или имя сущности из модели данных. Объекты ссылки должны быть связаны с единицами работ или перекрестками пунктирными линиями. Официальная спецификация IDEF3 различает три стиля объектов ссылок - безусловные (unconditional), синхронные (synchronous) и асинхронные (asynchronous). Process Modeler поддерживает только безусловные объекты ссылок. Синхронные и асинхронные объекты ссылок, используемые в диаграммах переходов состояний объектов, не поддерживаются.

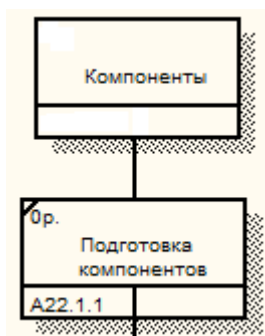


Рис. 21. Объект ссылки

При внесении объектов ссылок помимо имени рекомендуется указывать тип объекта ссылки. Типы объектов ссылок приведены в Таблица 8.

Таблица 8. Типы объектов ссылок

Тип объекта ссылки	Цель описания
OBJЕС	Описывает участие важного объекта в работе
GOTO	Инструмент циклического перехода (в повторяющейся последовательности работ), возможно на текущей диаграмме, но не обязательно. Если все работы цикла присутствуют на текущей диаграмме, цикл может также изображаться стрелкой, возвращающейся на стартовую работу. GOTO может ссылаться на перекресток

UOB (Unit of behavior)	. Применяется, когда необходимо подчеркнуть множественное использование какой-либо работы, но без цикла. Например, работа "Контроль качества" может быть использована в Процессе "Изготовления изделия" несколько раз, после каждой единичной операции. Обычно этот тип ссылки не используется для моделирования автоматически запускающихся работ
NOTE	Используется для документирования важной информации, относящейся к каким-либо графическим объектам на диаграмме. NOTE является альтернативой внесению текстового объекта в диаграмму
ELAB (Elaboration)	Используется для усовершенствования графиков или их более детального описания. Обычно употребляется для детального описания разветвления и слияния стрелок на перекрестках

Декомпозиция работ. В IDEF3 декомпозиция используется для детализации работ. Методология IDEF3 позволяет декомпонировать работу многократно, т. е. работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки. Возможность множественной декомпозиции предъявляет дополнительные требования к нумерации работ. Так, номер работы состоит из номера родительской работы, версии декомпозиции и собственного номера работы на текущей диаграмме (Рис. 22).

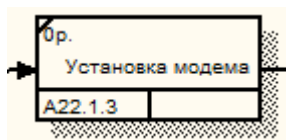


Рис. 22. Номер единицы работы (VOW)

Рассмотрим процесс декомпозиции диаграмм IDEF3, включающий взаимодействие автора (аналитика) и одного или нескольких экспертов предметной области.

Описание сценария, области и точки зрения. Перед проведением сеанса экспертизы у экспертов предметной области должны быть задокументированы сценарии и рамки модели для того, чтобы эксперт мог понять цели декомпозиции. Кроме того, если точка зрения моделирования отличается от точки зрения эксперта, она должна быть особенно тщательно задокументирована.

Возможно, что эксперт самостоятельно не сможет передать необходимую информацию. В этом случае аналитик должен приготовить список вопросов для проведения интервью.

Определение работ и объектов. Обычно эксперт предметной области передает аналитику текстовое описание сценария. В дополнение к этому может существовать документация, описывающая интересующие процессы. Из всей этой информации аналитик должен составить список кандидатов на работы (отглагольные существительные, обозначающие процесс, одиночные или в составе фразы) и кандидатов на объекты (существительные, обозначающие результат выполнения работы), которые необходимы для перечисленных в списке работ.

В некоторых случаях целесообразно создать графическую модель для представления ее эксперту предметной области. Графическая модель может быть также создана после сеанса сбора информации для того, чтобы детали форматирования диаграммы не смущали участников.

Поскольку разные фрагменты модели IDEF3 могут быть созданы разными группами аналитиков в разное время, IDEF3 поддерживает простую схему нумерации работ в рамках всей модели. Разные аналитики оперируют разными диапазонами номеров, работая при этом независимо. Пример выделения диапазона приведен в Таблица 9.

Таблица 9. Диапазоны номеров работ

Аналитик	Диапазон номеров IDEF3
Иванов	1-999
Петров	1000-1999
Сидоров	2000-2999

Последовательность и согласование. Если диаграмма создается после проведения интервью, аналитик должен принять некоторые решения, относящиеся к иерархии диаграмм, например сколько деталей включать в одну диаграмму. Если последовательность и согласование диаграмм неочевидны, может быть проведена еще одна экспертиза для детализации и уточнения информации. Важно различать подразумевающее согласование (согласование, которое подразумевается в отсутствие связей) и ясное согласование (согласование, ясно изложенное в мнении эксперта).

Работы, перекрестки и документирование объектов. IDEF3 позволяет внести информацию в модель различными способами. Например, логика взаимодействия может быть отображена графически в виде комбинации перекрестков. Та же информация может быть отображена в виде объекта ссылки типа ELAB (Elaboration). Это позволяет аналитику вносить информацию в удобном в данный момент времени виде. Важно учитывать, что модели могут быть реорганизованы, например для их представления в более презентабельном виде. Выбор формата для презентации часто имеет важное значение для организации модели, поскольку комбинация перекрестков занимает значительное место на диаграмме и использование иерархии перекрестков затрудняет расположение работ на диаграмме.

Задание на лабораторную работу

Провести декомпозицию работы «Продажи и маркетинг», построить DFD-диаграмму для работы «Оформление заказов». Провести декомпозицию работы «Сборка настольных компьютеров» с помощью диаграммы IDEF3.

Декомпозиция работы "Продажи и маркетинг"

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

На основе этой информации декомпозируйте работу "Продажи и маркетинг" (IDEF0).

Создайте следующие работы:

Предоставление информации о ценах.

Оформление заказов.

Исследование рынка.

Результат декомпозиции представлен на Рис. 23.

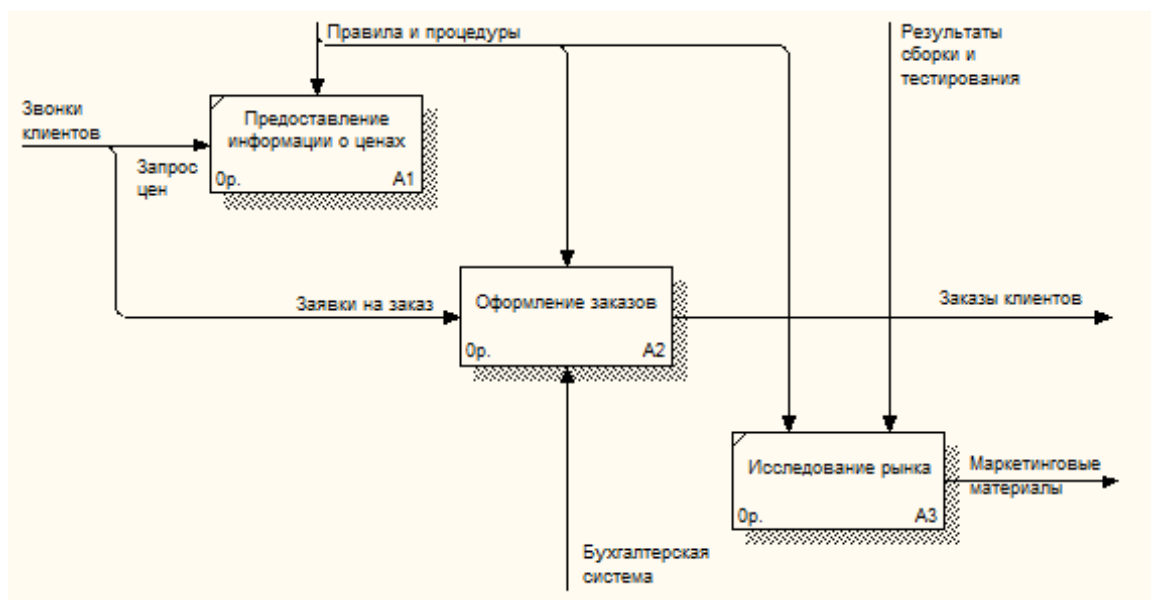


Рис. 23. Результат выполнения пятой части упражнения 14 - диаграмма A2

Построение DFD диаграммы

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 клиентов может просматриваться и редактироваться. Заказ должен включать как информацию о клиенте, так и ин-

формацию 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" на диаграмме A2.

В диалоге Activity Box Count выберите количество работ 2 и нотацию DFD (Рис. 24).

Щелкните по ОК и внесите в новую диаграмму, DFD A22, имена работ:

Проверка и внесение клиента.

Внесение заказа.

4. Используя кнопку данных  на палитре инструментов, внесите хранилища

Список клиентов;

Список продуктов;

Список заказов.

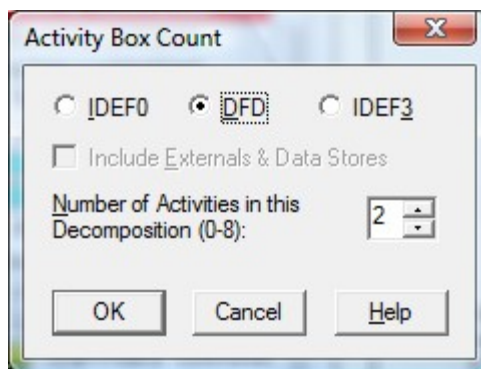



Рис. 24. Выбор нотации DFD в диалоге Activity Box Count

Удалите граничные стрелки с диаграммы DFD A22.

Используя кнопку  на палитре инструментов, внесите внешнюю ссылку: Звонки клиентов.

7. Создайте внутренние ссылки согласно Рис. 25. При именовании стрелок используйте словарь.

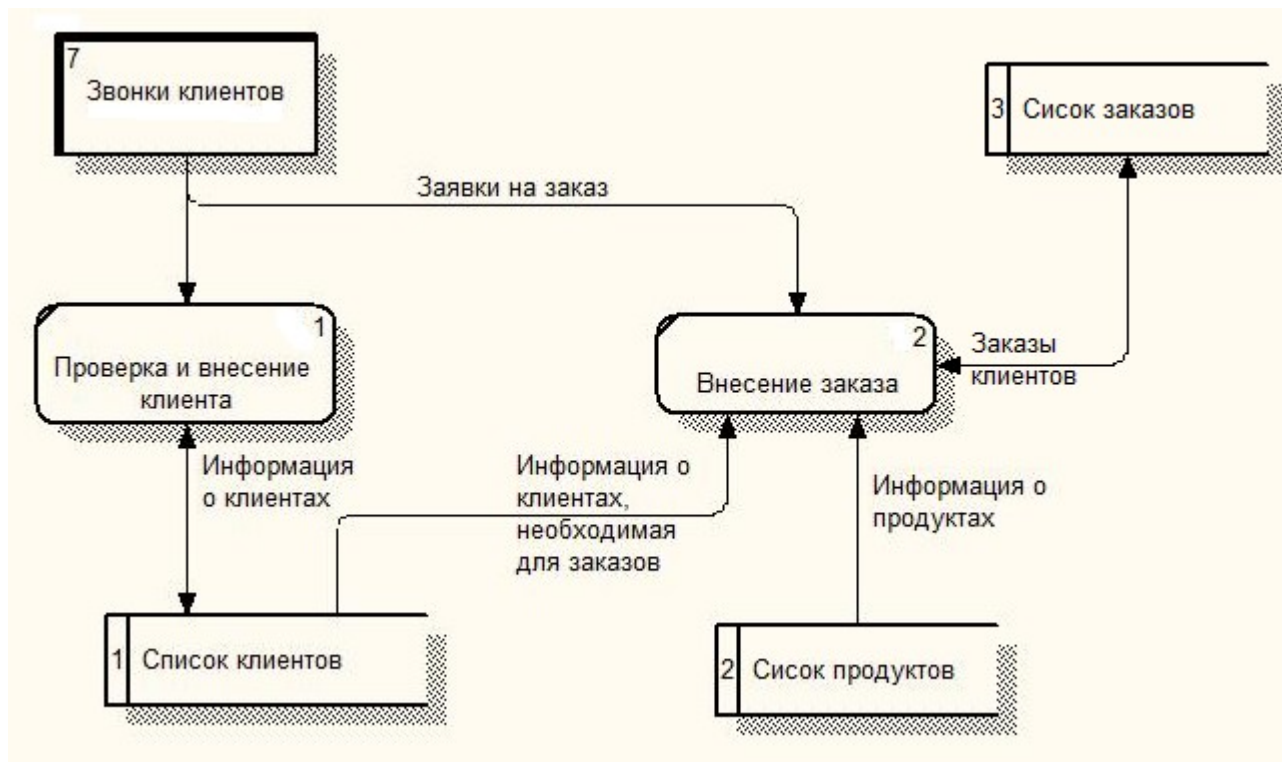


Рис. 25. Диаграмма A22

Обратите внимание, что стрелки *"Информация о клиентах"* и *"Заказы клиентов"* двунаправленные. Для того чтобы сделать стрелку двунаправленной, щелкните правой кнопкой по стрелке, выберите в контекстном меню пункт *Style* и во вкладке *Style* выберите опцию *Bidirectional*.

Создание диаграммы IDEF3

1. Перейдите на диаграмму A2 и декомпозируйте работу *"Сборка настольных компьютеров"*. В диалоге *Activity Box Count* (Рис. 26) установите число работ 4 и нотацию IDEF3.

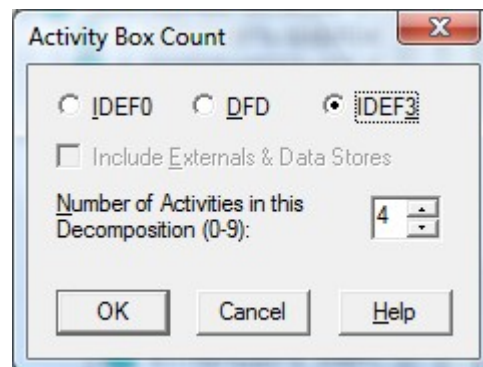



Рис. 26. Выбор нотации IDEF3 в диалоге Activity Box Count

Возникает диаграмма IDEF3, содержащая работы (UOW). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню Name и внесите имя работы "Подготовка компонентов". Затем во вкладке Definition внесите определение "Подготавливаются все компоненты компьютера согласно спецификации заказа".

2. Во вкладку UOW внесите свойства работы (Таблица 10).

Таблица 10. Свойства UOW

<i>Objects</i>	Компоненты: винчестеры, корпуса, материнские платы, видеокарты, звуковые карты, дисководы CD-ROM и флоппи, модемы, программное обеспечение
<i>Facts</i>	Доступные операционные системы: Windows 98, Windows NT, Windows 2000
<i>Constrains</i>	Установка модема требует установки дополнительного программного обеспечения

3. Внесите в диаграмму еще три работы (кнопка ). Внесите имена следующих работ:

Установка материнской платы и винчестера.


Установка модема.

Установка дисковод CD-ROM.

Установка флоппи-дисковод.

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 компонентов".

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, хотя BPwin показывает отсутствие имени как ошибку.

6. С помощью кнопки  на палитре инструментов внесите два перекрестка типа асинхронного "или" и свяжите работы с перекрестками, как показано на Рис. 27.

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, как показано на Рис. 27.

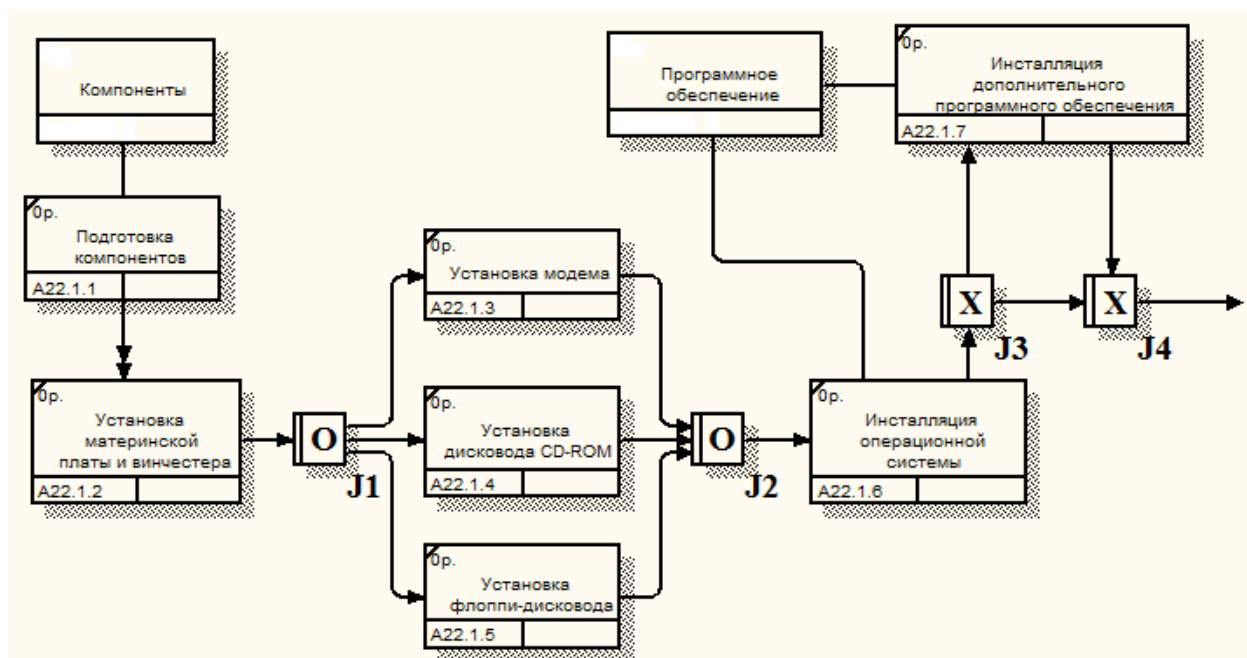


Рис. 27. Результат выполнения упражнения 7

Лабораторная работа №4

Создание модели данных с помощью Allfusion ERwin Data Modeler

Цель работы

Знакомство с CASE-системой Allfusion ERWin Data Modeler, изучение основных принципов построения логической модели данных, разработка модели.

Отображение модели данных в ERwin

Физическая и логическая модель данных

ERwin имеет два уровня представления модели - логический и физический.

Логический уровень - это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например "Постоянный клиент", "Отдел" или "Фамилия сотрудника". Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами (подробнее о сущностях и атрибутах будет рассказано ниже). Логическая модель данных может быть построена на основе другой логической модели, например, на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Физическая модель данных, напротив, зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах - таблицах, колонках,

индексах, процедурах и т. д. Разделение модели данных на логические и физические позволяет решить несколько важных задач.

Документирование модели. Многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов - пробела и т. п.). Зачастую разработчики ИС имеют дело с нелокализованными версиями СУБД. Это означает, что объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т. е. нельзя назвать таблицу предложением - только одним словом). Разделение модели на логическую и физическую позволяет решить эту проблему. На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы - имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Такое соответствие позволяет лучше задокументировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Масштабирование. Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели, проектировщик может выбрать необходимую СУБД и ERwin автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется прямым проектированием (Forward Engineering). Тем самым достигается масштабируемость - создав одну логическую модель данных, можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД. С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать физическую и логическую модель данных (Reverse Engineering). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД и затем сгенерировать ее системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на

другой. Например, можно перенести структуру данных с Oracle на Informix (или наоборот).

Для переключения между логической и физической моделью данных служит список выбора в левой части панели инструментов Erwin (Рис. 28).







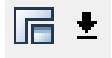
Рис. 28. Переключение между логической и физической моделью











При переключении, если физической модели еще не существует, она будет создана автоматически.

Интерфейс ERwin. Уровни отображения модели

Интерфейс выполнен в стиле Windows-приложений, достаточно прост и интуитивно понятен. В дальнейшем будет описан интерфейс версии Erwin 3.5.2. Рассмотрим кратко основные функции ERwin по отображению модели, а также панель и палитру инструментов. Более подробно элементы интерфейса будут рассмотрены в последующих главах. Элементы панели инструментов описаны в Таблица 11.

Таблица 11. Основные команды и панели инструментов

Кнопки	Назначение кнопок
Стандартная панель	
	Создание, открытие, сохранение и печать модели
	Вызов диалога Report Templates для генерации отчетов
	Изменение уровня просмотра модели: уровень сущностей, уровень атрибутов и уровень определений
	Изменение масштаба просмотра модели
	Переключение между областями модели - Subject Area
Панель базы данных (Database, команды доступны только на уровне фи-	

зической модели)	
	Генерация схемы БД (Forward Engineer), построение модели по базе данных (Reverse Engineer), сравнение схемы с моделью, выбор сервера, валидация SQL запросов в модели
Панель инструментов (Toolbox)	
	Указатель
	Создание новой сущности
	Создание категории (отображается только на уровне логической модели)
	Создание нового представления (отображается только на уровне физической модели)
	Создание идентифицирующей связи
	Создание связи многие ко многим (отображается только на уровне логической модели)
	Создание связи между представлением и временной таблицей (отображается только на уровне физической модели)
	Создание неидентифицирующей связи
Панель рисования (Drawing)	
	Рисование прямоугольников, овалов, линий и т.д., размещение текстовых блоков

Палитра инструментов выглядит различно на разных уровнях отображения модели.

ERwin имеет несколько уровней отображения диаграммы: уровень сущностей, уровень атрибутов, уровень определений, уровень первичных ключей и уровень иконок. Переключиться между первыми тремя уровнями можно с

использованием кнопок панели инструментов. Переключиться на другие уровни отображения можно при помощи контекстного меню, которое появляется, если "кликнуть" по любому месту диаграммы, не занятому объектами модели. В контекстном меню следует выбрать пункт Display Level и затем необходимый уровень отображения. ERwin позволяет связать с сущностью большую и малую иконки. При переключении на уровень иконок показывается большая иконка. Для отображения малой иконки следует выбрать в контекстном меню пункт Display Options/Entities и в каскадном меню включить опцию Entity Icon. Малая иконка будет показана слева от имени сущности на всех уровнях отображения модели.

Подмножества модели и сохраняемые отображения

При создании реальных моделей данных количество сущностей и атрибутов может исчисляться сотнями. Для более удобной работы с большими моделями в ERwin предусмотрены подмножества модели (Subject Area), в которые можно включить тематически общие сущности. В подмножество модели может входить произвольный набор сущностей, связей и текстовых комментариев. Для создания, удаления или редактирования подмножеств модели нужно вызвать диалог Subject Area Editor (меню Edit/Subject Area), в котором указывается имя подмножества и входящие в нее сущности (Рис. 29). Все изменения, сделанные в любой Subject Area, автоматически отображаются на общей модели. Одна и та же сущность может входить в несколько Subject Area.

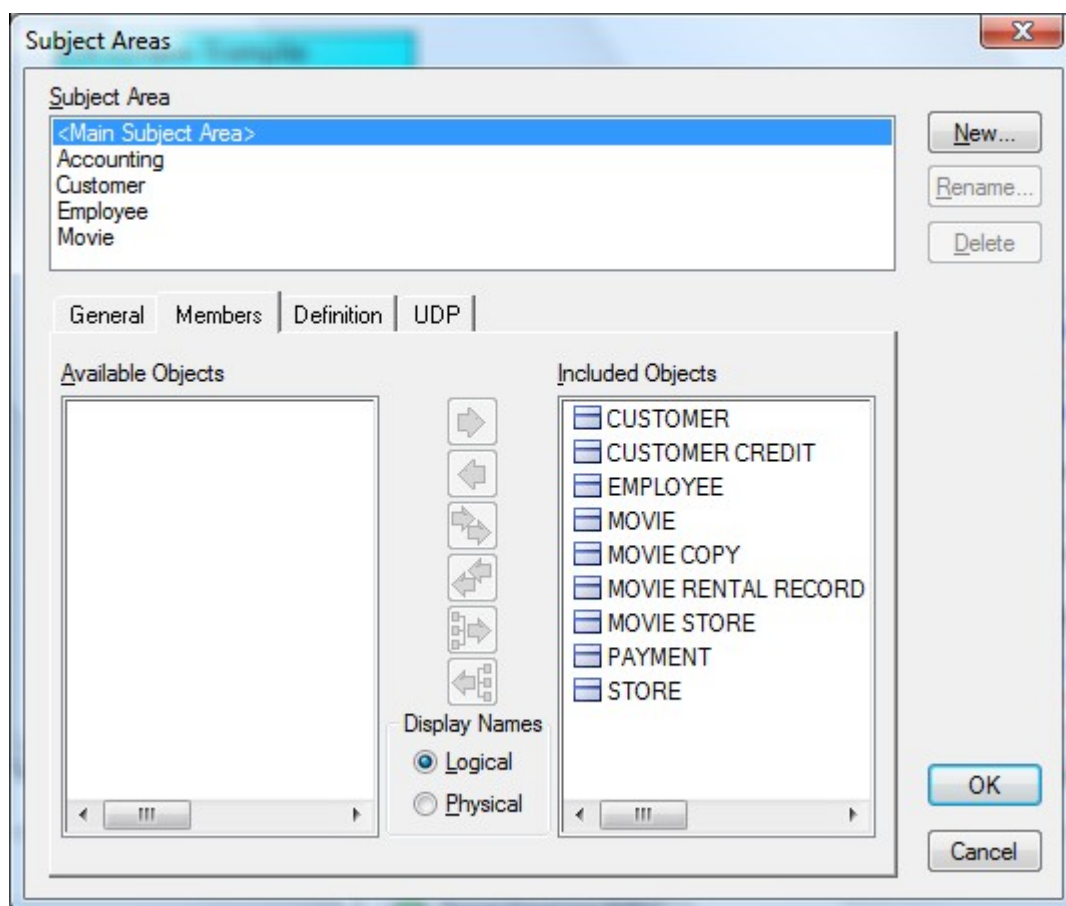


Рис. 29. Диалог *Subject Area Editor*

По умолчанию исходная модель получает имя Main Subject Area. При создании нового подмножества следует в диалоге Subject Area Editor указать ее имя и список входящих в него объектов. Subject Area можно создавать как в логической, так и в физической модели данных.

Создание логической модели данных

Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

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

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализиро-

вана, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, - более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель - наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Сущности и атрибуты

Основные компоненты диаграммы Erwin - это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности - строка в таблице, а атрибуту - колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которых должна сохраняться. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить "технических" наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущность *Заказчик* с атрибутами *Номер заказчика*, *Фамилия за-*

казчика и Адрес заказчика. На уровне физической модели ей может соответствовать таблица *Customer* с колонками *Customer_number*, *Customer_name* и *Customer_address*.

Каждая сущность должна быть полностью определена с помощью текстового описания в закладке Definition. Закладки Note, Note 2, Note 3, UDP (User Defined Properties - Свойства, определенные пользователем) служат для внесения дополнительных комментариев и определений к сущности.

Закладка Definition используется для ввода определения сущности. Эти определения полезны как на логическом уровне, поскольку позволяют понять, что это за объект, так и на физическом уровне, поскольку их можно экспортировать как часть схемы и использовать в реальной БД (CREATE COMMENT on entity_name).

Закладка Note позволяет добавлять дополнительные замечания о сущности, которые не были отражены в определении, введенном в закладке Definition. Здесь можно ввести полезное замечание, описывающее какое-либо бизнес-правило или соглашение по организации диаграммы.

В закладке Note 2 можно задокументировать некоторые возможные запросы, которые, как ожидается, будут использоваться по отношению к сущности в БД. При переходе к физическому проектированию, записанные запросы помогут принимать такие решения в отношении проектирования, которые сделают БД более эффективной.

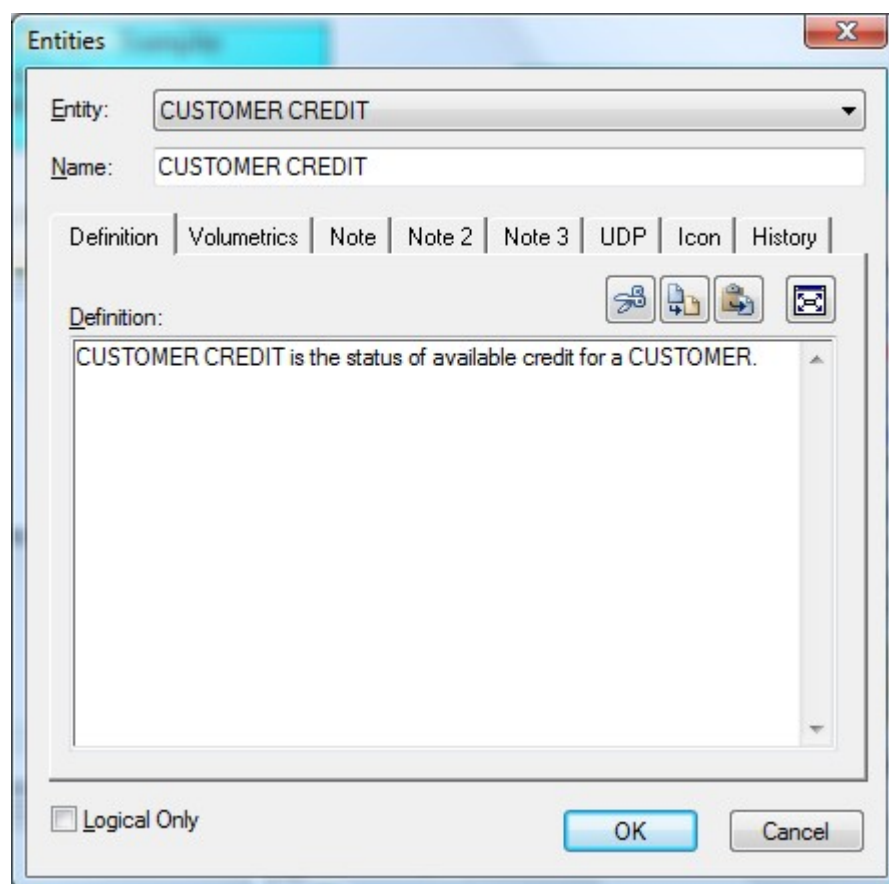


Рис. 30. Диалог *Entities*

Закладка Note 3 позволяет вводить примеры данных для сущности (в произвольной форме).

Как было указано выше, каждый атрибут хранит информацию об определенном свойстве сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом. Для описания атрибутов следует, "кликнув" правой кнопкой по сущности, выбрать в появившемся меню пункт Attribute Editor. Появляется диалог Attribute Editor (Рис. 31).

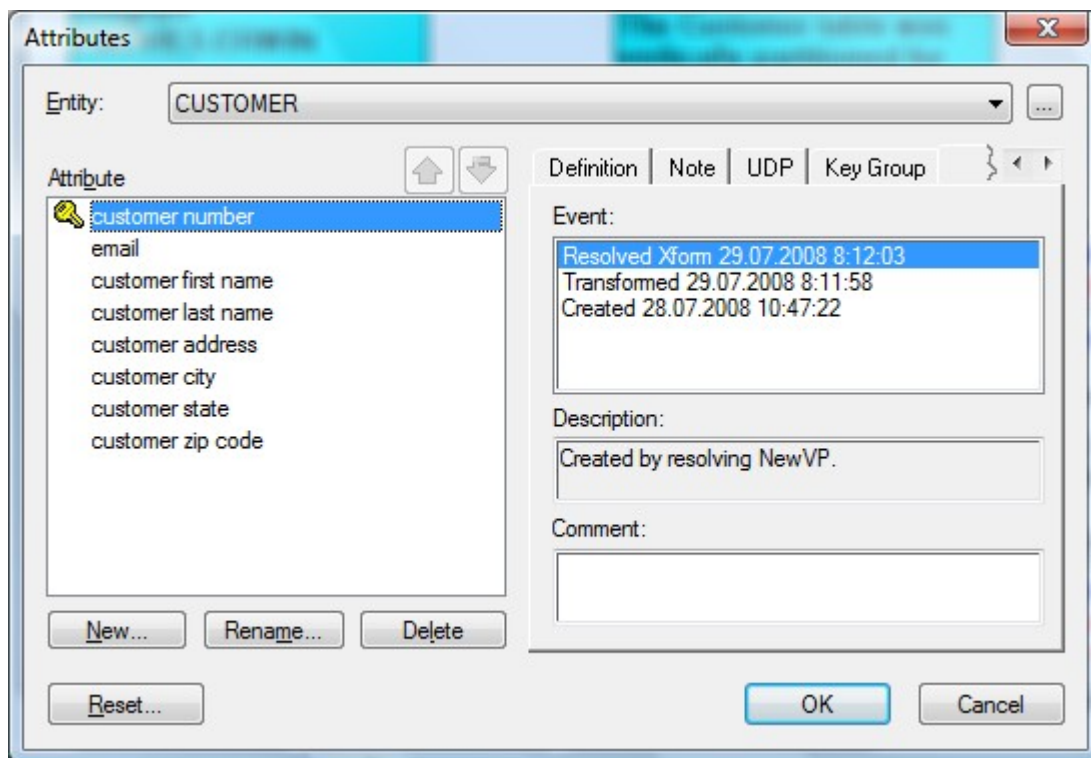


Рис. 31. Диалог *Attribute Editor*

Если щелкнуть по кнопке New, то в появившемся диалоге New Attribute () можно указать имя атрибута, имя соответствующей ему в физической модели колонки и домен. Домен атрибута будет использоваться при определении типа колонки на уровне физической модели.

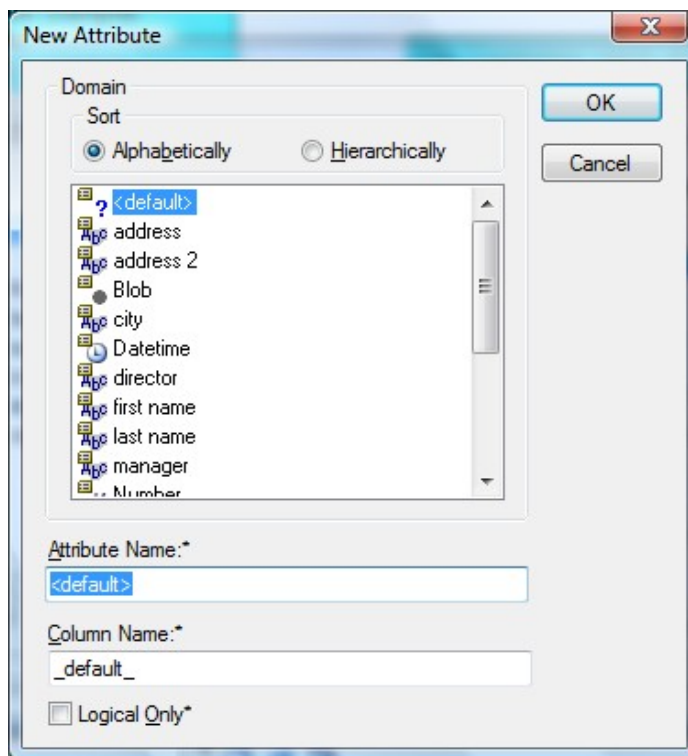


Рис. 32. Диалог *New Attribute*

Для атрибутов первичного ключа в закладке General диалога Attribute Editor необходимо сделать пометку в окне выбора Primary Key.

Закладка Definition позволяет записывать определения отдельных атрибутов. Определения атрибутов можно также сгенерировать как часть схемы (CREATE COMMENT on entity_name.attribute_name). Закладка Note позволяет добавлять замечания об одном или нескольких атрибутах сущности, которые не вошли в определения. Закладка UDP служит для задания значений свойств, определяемых пользователем. Предварительно эти свойства должны быть внесены в диалог User-Defined Property Editor как свойства атрибутов.

При установлении связей между сущностями атрибуты первичного ключа родительской сущности мигрируют в качестве внешних ключей в дочернюю сущность. Кнопка Migrate диалога Attribute Editor вызывает диалог Migrate Attribute Property, в котором можно задать свойства, сохраняемые при миграции.

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов. Например, создание в сущности *Сотрудник* атрибута *Телефоны сотрудника* противоречит требованиям нормализации, поскольку атрибут должен быть атомарным, т. е. не содержать множественных значений. Согласно синтаксису IDEF1X имя атрибута должно быть уникально в рамках модели (а не только в рамках сущности!). По умолчанию при попытке внесения уже существующего имени атрибута Erwin переименовывает его. Например, если атрибут *Комментарий* уже существует в модели, другой атрибут (в другой сущности) будет назван *Комментарий/2*, затем *Комментарий/3* и т. д.

Часто приходится создавать производные атрибуты, т. е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить *Возраст сотрудника*, который может быть вычислен из атрибута *Дата рождения сотрудника*. Такой атрибут может при-

вести к конфликтам; действительно, если вовремя не обновить значение атрибута *Возраст сотрудника*, он может противоречить значению атрибута *Дата рождения сотрудника*. Производные атрибуты - ошибка нормализации, однако их вводят для повышения производительности системы - если необходимо узнать возраст сотрудника, можно обратиться к соответствующему атрибуту, а не проводить вычисления (которые на практике могут быть значительно более сложными, чем в приведенном примере) по дате рождения.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (Verb Phrases) (Рис. 33). Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы, например:

Каждый КЛИЕНТ *<размещает>* ЗАКАЗы;

Каждый ЗАКАЗ *<выполняется>* СОТРУДНИКом.

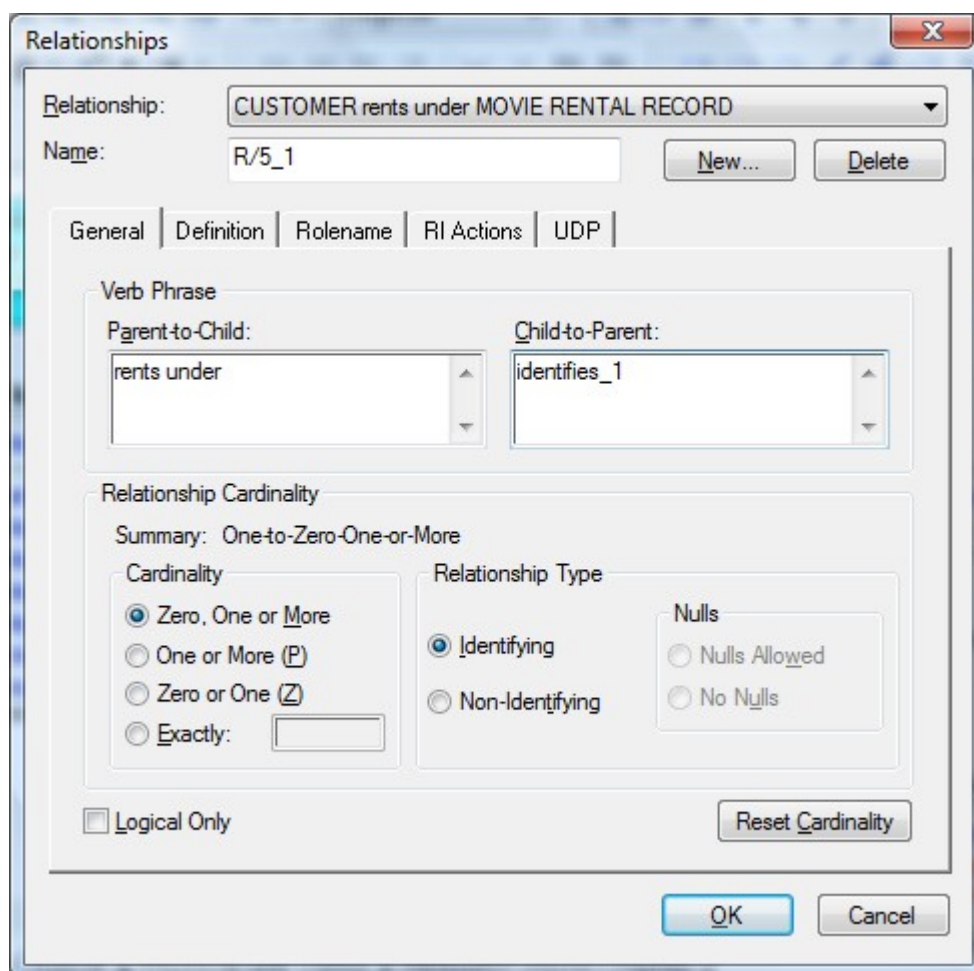


Рис. 33. Диалог Relationships

Связь показывает, какие именно заказы разместил клиент и какой именно сотрудник выполняет заказ. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт *Display Options/Relationship* и затем включить опцию *Verb Phrase*.

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (соответственно это кнопки слева направо в палитре инструментов).

В 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. Когда рисуется идентифициру-

ющая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (сущность *Заказ* на Рис. 34). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на Рис. 34 информация о заказе не может быть внесена и не имеет смысла без информации о клиенте, который его размещает. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности мигрировавшие атрибуты помечаются как внешний ключ - (FK).

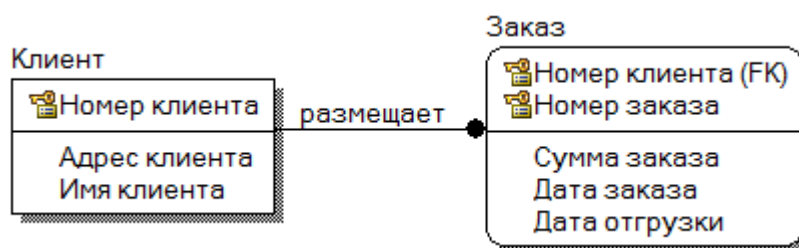


Рис. 34. Идентифицирующая связь между независимой и зависимой таблицей

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получают признак NOT NULL, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

При установлении неидентифицирующей связи (Рис. 35) дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

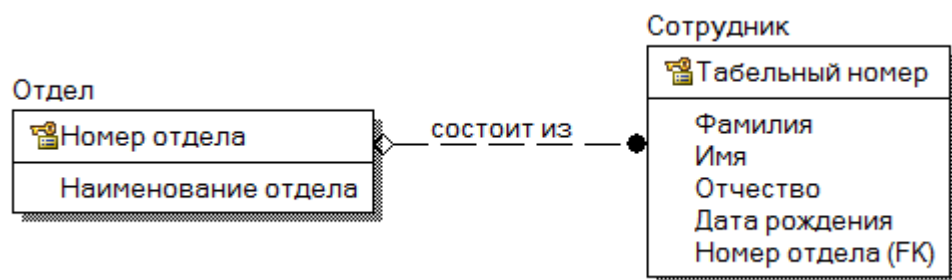


Рис. 35. Неидентифицирующая связь

Экземпляр сущности *Сотрудник* может существовать безотносительно к какому-либо экземпляру сущности *Отдел*, т. е. сотрудник может работать в организации, не числясь в каком-либо отделе.

Для создания новой связи следует:

установить курсор на нужной кнопке в палитре инструментов (идентифицирующая или неидентифицирующая связь) и нажать левую кнопку мыши (Рис. 35);

щелкнуть сначала по родительской, а затем по дочерней сущности.

Форму линии связи можно изменить. Для этого нужно захватывать мышью нужную линию связи и переносить ее с места на место, пока линия не начнет выглядеть лучше.

Для редактирования свойств связи следует "кликнуть" правой кнопкой мыши по связи и выбрать на контекстном меню пункт Relationship Editor.

В закладке General появившегося диалога можно задать мощность, имя и тип связи (Рис. 33).

Мощность связи (Cardinality) - служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Различают четыре типа мощности:

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

символом Р помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);

символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);

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

По умолчанию символ, обозначающий мощность связи, не показывается на диаграмме. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт *Display Options/Relationship* и затем включить опцию *Cardinality*.

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*). Для связи многие-ко-многим следует указывать имена как *Parent-to-Child* так и *Child-to-Parent*.

Тип связи (идентифицирующая/неидентифицирующая). Для неидентифицирующей связи можно указать обязательность (*Nulls*). В случае обязательной связи (*No Nulls*) при генерации схемы БД атрибут внешнего ключа получит признак *NOT NULL*, несмотря на то что внешний ключ не войдет в состав первичного ключа дочерней сущности. В случае необязательной связи (*Nulls Allowed*) внешний ключ может принимать значение *NULL*. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности (см. Рис. 35).

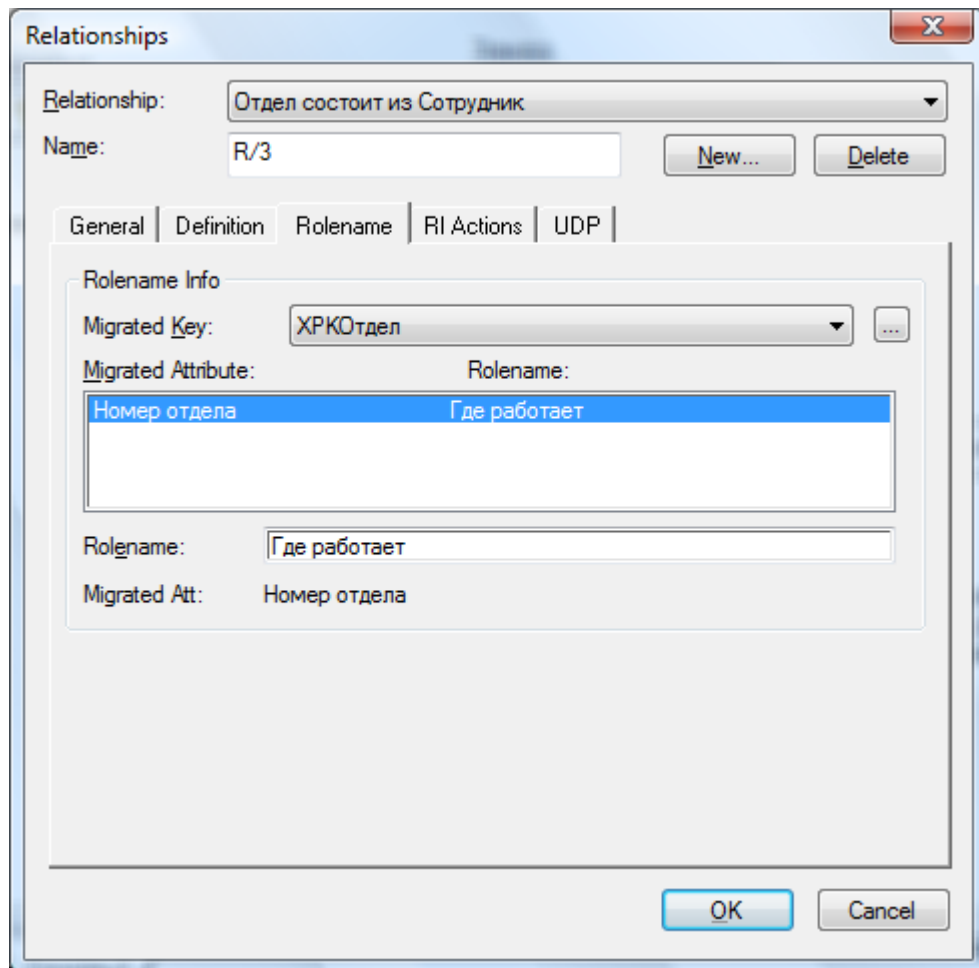


Рис. 36. Закладка Rolename диалога Relationships

В закладке Definition можно дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

В закладке Rolename можно задать имя роли и правила ссылочной целостности.

Имя роли (функциональное имя) - это синоним атрибута внешнего ключа, который показывает, какую роль играет атрибут в дочерней сущности.



Рис. 37. Имена ролей внешних ключей

В примере, приведенном на Рис. 37, в сущности *Сотрудник* внешний ключ *Номер отдела* имеет функциональное имя "Где работает", которое показывает, какую роль играет этот атрибут в сущности. По умолчанию в списке атрибутов показывается только имя роли. Для отображения полного имени атрибута (как функционального имени, так и имени роли) следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт Display Options/Entities и затем включить опцию Rolename/Attribute (Рис. 36). Полное имя показывается как функциональное имя и базовое имя, разделенные точкой (см. Рис. 37).

Обязательным является применение имен ролей в том случае, когда два или более атрибутов 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 "рыболовный крючок" - fish hook), когда одна и та же сущность является и родительской и дочерней одновременно. При задании рекурсивной связи атрибут должен мигрировать 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. На Рис. 37 сущность *Сотрудник* содержит атрибут первичного ключа *Табельный номер*. Информация о руководителе сотрудника содержится в той же сущности, поскольку руководитель работает в той же организации. Чтобы сослаться на руководителя сотрудника следует создать рекурсивную связь (на Рис. 37 связь руководит/подчиняется) и присвоить имя роли ("Руководитель"). Заметим, что рекурсивная связь может быть только неидентифицирующей. В противном случае внешний ключ должен был бы войти в состав первичного ключа и получить при генерации схемы признак NOT NULL. Это сделало бы невозможным построение иерархии - у дерева подчиненности должен быть корень - сотрудник, который никому не подчиняется в рамках данной организации.

Связь руководит/подчиняется на Рис. 37 позволяет хранить древовидную иерархию подчиненности сотрудников.

Правила ссылочной целостности (referential integrity, RI) - логические конструкции, которые выражают бизнес-правила использования данных и представляют собой правила вставки, замены и удаления. При генерации схемы БД на основе опций логической модели, задаваемых в закладке RI Actions (Рис. 38), будут сгенерированы правила декларативной ссылочной целостности, которые должны быть предписаны для каждой связи, и триггеры, обеспечивающие ссылочную целостность. Триггеры представляют собой программы, выполняемые всякий раз при выполнении команд вставки, замены или удаления (INSERT, UPDATE или DELETE). На Рис. 34 существует идентифицирующая связь между сущностями *Клиент* и *Заказ*. Что будет, если удалить клиента? Экземпляр сущности *Заказ* не может существовать без клиента (атрибут первичного ключа *Номер клиента* не может принимать значение NULL), следовательно, нужно либо запретить удаление клиента, пока за ним числится хотя бы один заказ (для удаления клиента сначала нужно удалить все его заказы), либо сразу удалять вместе с командой всех ее игроков. Такие правила удаления называются "ограничение" и "каскад" (Parent RESTRICT и Parent CASCADE, см. Рис. 38). Заметим, что если сущность *Заказ*, в свою очередь, связана идентифицирующей связью с какой-то еще сущностью, то в случае удаления каскадом клиента будут удалены все его заказы и все элементы третьей сущности, связанные с удаленными заказами. Выполнение команды на удаление одной строки реально может привести к удалению тысячи строк в БД, поэтому использовать правило удаления каскадом следует с осторожностью. В том случае, если установлено правило ограничения удаления, при попытке выполнить удаление клиента, у которого есть хотя бы один заказ, сервер реляционной СУБД возвратит ошибку.

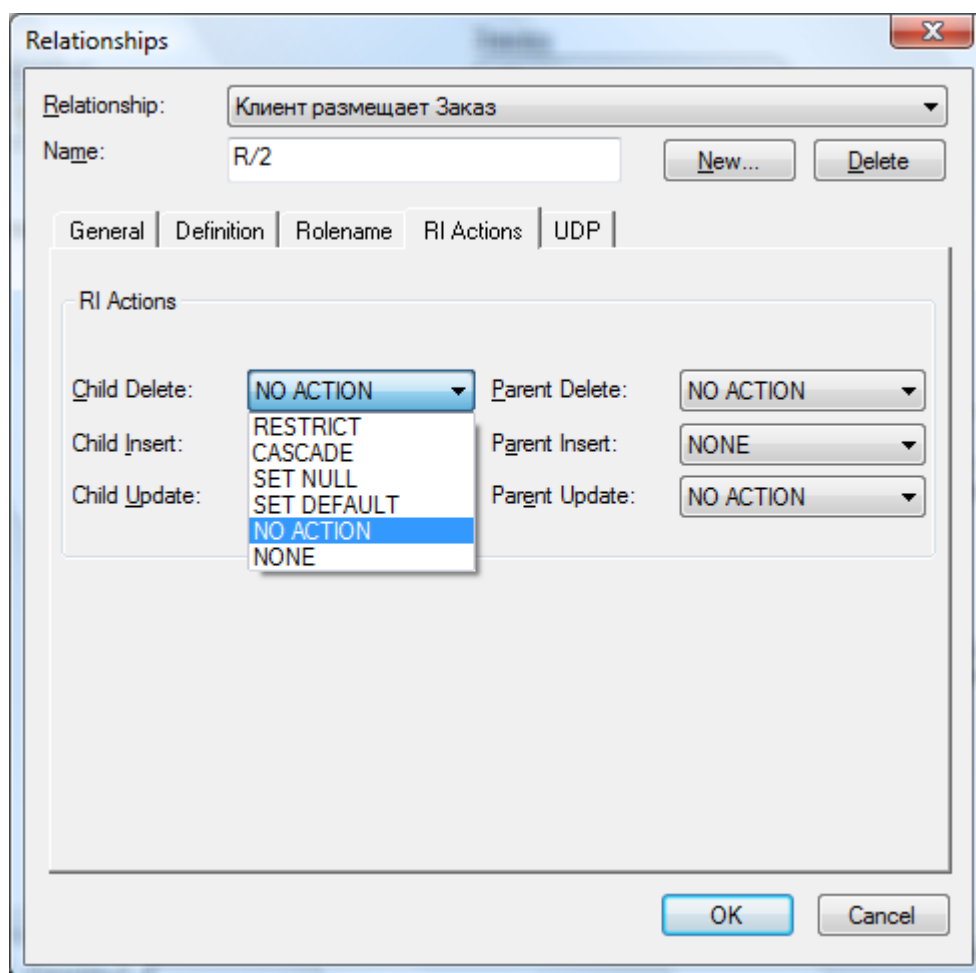


Рис. 38. Закладка *RI Actions* диалога *Relationships*

На рис. Рис. 37 установлена необязательная неидентифицирующая связь между сущностями *Отдел* и *Сотрудник*. Экземпляр сущности *Сотрудник* может существовать без ссылки на отдел (атрибут внешнего ключа *Где работает. Номер отдела* может принимать значение NULL). В этом случае возможно установление правила установки в нуль - SET NULL. При удалении отдела атрибут внешнего ключа сущности *Сотрудник* - *Где работает. Номер отдела* примет значение NULL. Это означает, что при удалении отдела сотрудник остается работать в организации не будучи приписан к какому-либо отделу и информация о нем сохраняется.

Возможна установка еще двух правил удаления (если таковые поддерживаются СУБД):

SET DEFAULT - при удалении атрибуту внешнего ключа присваивается значение по умолчанию. Например, при удалении клиента заказы могут быть переписаны на другого клиента (что, конечно, вряд ли имеет смысл).

NONE - при удалении значение атрибута внешнего ключа не меняется. Запись о заказе "повисает в воздухе", т. е. ссылается на несуществующего уже клиента.

Правила удаления управляют тем, что будет происходить в БД при удалении строки. Аналогично правила вставки и обновления управляют тем, что будет происходить с БД, если строки изменяются или добавляются. Например, можно установить правило, которое разрешает вносить новую команду только в том случае, когда в нее зачислен хотя бы один игрок. Желаемое поведение может быть достигнуто следующими действиями:

Задать мощность связи между сущностями *Команда* и *Игрок*, равную "One or more" - 1 или более (тип Р). Предполагается, что установлена идентифицирующая связь.

Присвоить действие RI-триггера "Parent Insert-CASCADE" для того, чтобы при создании новой строки в таблице *Команда* автоматически создавалась хотя бы одна строка в дочерней таблице *Игрок*.

Присвоить связи действие RI-триггера "Parent Delete-CASCADE" для того, чтобы при удалении строки из таблицы *Команда* соответствующая строка или строки из таблицы *Игрок* тоже удалялись.

ERwin автоматически присваивает каждой связи значение ссылочной целостности, устанавливаемой по умолчанию, прежде чем добавить ее в диаграмму. Режимы RI, присваиваемые ERwin по умолчанию (приведены в Таблица 12), могут быть изменены в редакторе Referential Integrity Default, который вызывается, если щелкнуть по кнопке RI Defaults диалога Target Server (меню Server/Target Server).

Таблица 12. Значения RI, присваиваемые в ERwin по умолчанию, а также возможные режимы для каждого типа связи

	<i>Идентифицирующая связь</i>	<i>Неидентифицирующая связь (Nulls Allowed)</i>	<i>Неидентифицирующая связь (No Nulls)</i>	<i>Категориальная связь</i>

Chi Id Delete Возмож- ные ре- жимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Chi Id Delete Режимы по умолча- нию	NONE	NONE	NONE	NONE
Chi Id Insert Возмож- ные ре- жимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Chi Id Insert Режимы по умолча- нию	RE- STRICT	SET NULL	RESTRICT	RESTR ICT
Chi Id Up- date Возмож- ные ре- жимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Chi Id Up-	RE-	SET NULL	RESTRICT	RESTR

date Ре- жимы по умолча- нию	STRICT			ICT
Par- ent Delete Возмож- ные ре- жимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Par ent Delete Режимы по умолча- нию	RESTRIC T	SET NULL	RESTRICT	CASC ADE
Par- ent In- sert Воз- можные режимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Par ent Insert Режимы по умолча-	NONE	NONE	NONE	NONE

нию				
Par- ent Up- date Возмож- ные ре- жимы	RE- STRICT, CAS- CADE, NONE	RESTRICT, CASCADE, NONE, SET NULL, SET DE- FAULT	RESTRICT, CASCADE, NONE, SET DE- FAULT	RE- STRICT, CASCADE, NONE
Par ent Update Режимы по умолча- нию	RESTRIC T	SET NULL	RESTRICT	CASC ADE

Связь многие-ко-многим возможна только на уровне логической модели данных. На Рис. 39 показан пример связи многие-ко-многим. Сотрудник может обслуживать много клиентов, клиент может обслуживаться несколькими сотрудниками. Такая связь обозначается сплошной линией с двумя точками на концах.

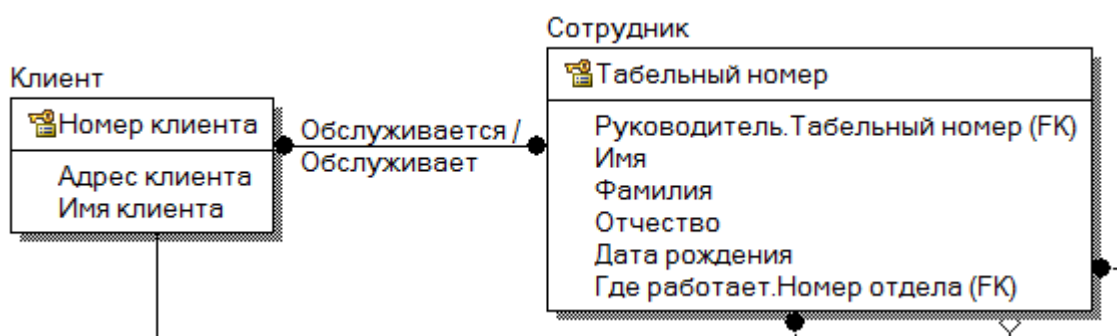


Рис. 39. Связь многие-ко-многим

Связь многие-ко-многим должна именоваться двумя фразами - в обе стороны (в примере "Обслуживается/Обслуживает"). Это облегчает чтение диаграммы.

ERwin может автоматически преобразовать связь многие-ко-многим (команда контекстного меню связи Create Association Entity), добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице (Рис. 40). При этом имя новой таблицы присваивается автоматически как “Имя1 Имя2”.

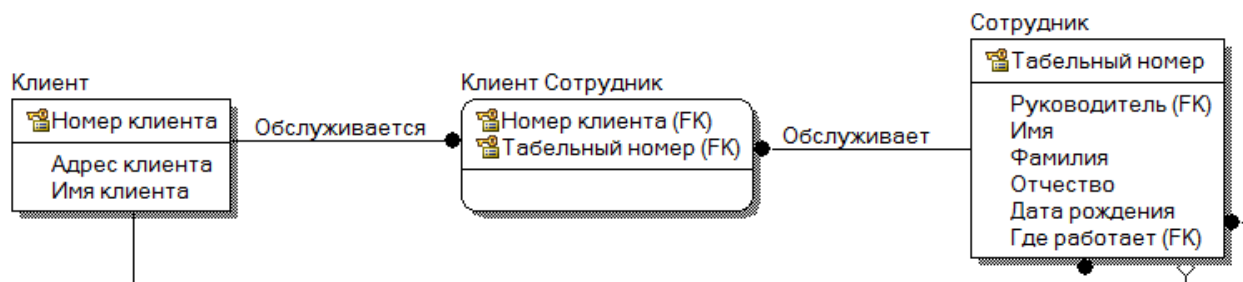


Рис. 40. Иллюстрация автоматического разрешения связи многие-ко-многим на уровне физической модели

Автоматического решения проблемы связи многие-ко-многим не всегда оказывается достаточно. В нашем примере связь между сотрудником появляется при оформлении заказа. Более того один и тот же клиент может оформить несколько заказов у одного и того же сотрудника. Поэтому для корректного моделирования этой связи следует использовать отношение Заказ.

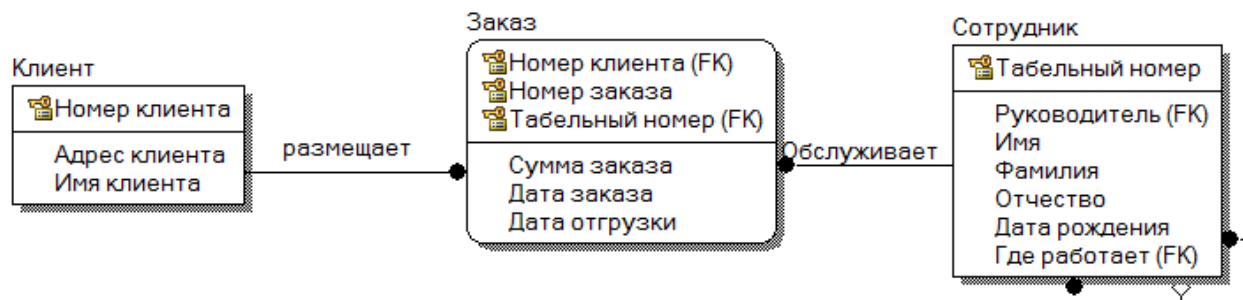


Рис. 41. Дополнение модели при разрешении связи многие-ко-многим на уровне физической модели

Задание на лабораторную работу

Построить логическую модель данных, включающую основные сущности и связи, для разработки информационной системы в следующих предметных областях:

- Учет заработной платы

- Учет кадров
- Учет студентов
- Составление расписания занятий
- Составление расписания консультаций и экзаменов
- Учет торговых операций
- Учет заказов на изготовление печатной продукции
- Учет заказов на сборку компьютеров
- Продажа билетов на поезда
- Инвентаризация компьютерной и оргтехники
- Аренда автомобилей
- Долевое строительство

Лабораторная работа №5

Создание логической модели данных

Цель работы

Развитие логической модели.

Типы сущностей и иерархия наследования

Как было указано выше, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей.

Характеристическая - зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.

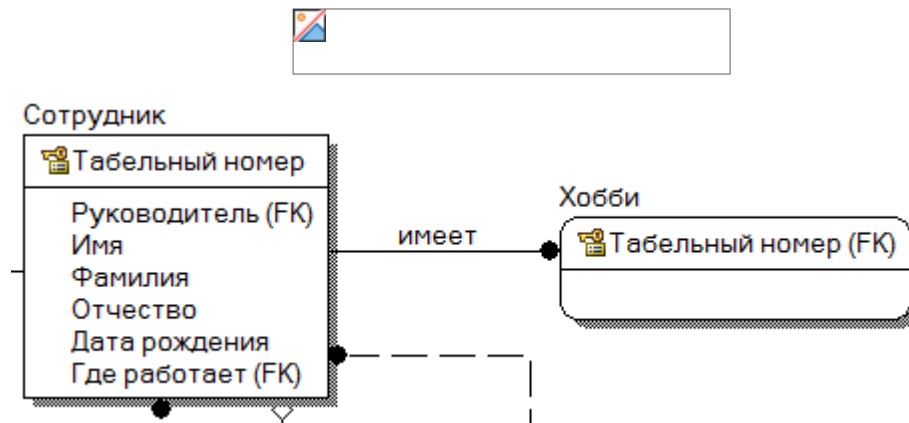


Рис. 42. Пример характеристической сущности "Хобби "

Ассоциативная - сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей. Примером ассоциативной сущности является *Заказ* на Рис. 43.

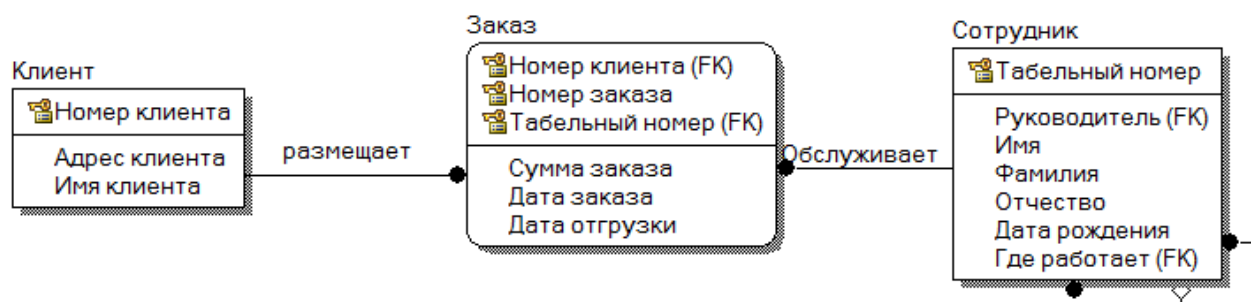


Рис. 43

Именуемая - частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

Категориальная - дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (постоянные служащие) и совместители. Из их общих свойств можно сформировать обобщенную сущность (родовой предок) *Сотрудник* (Рис. 44), чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных сущностях (потомках) *Постоянный сотрудник* и *Совместитель*.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи (например, если бы *Постоянный сотрудник* и *Совместитель* имели бы сходную по смыслу связь "работает в" с сущностью *Организация*), либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор - атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой (атрибут *Тип* на Рис. 44).

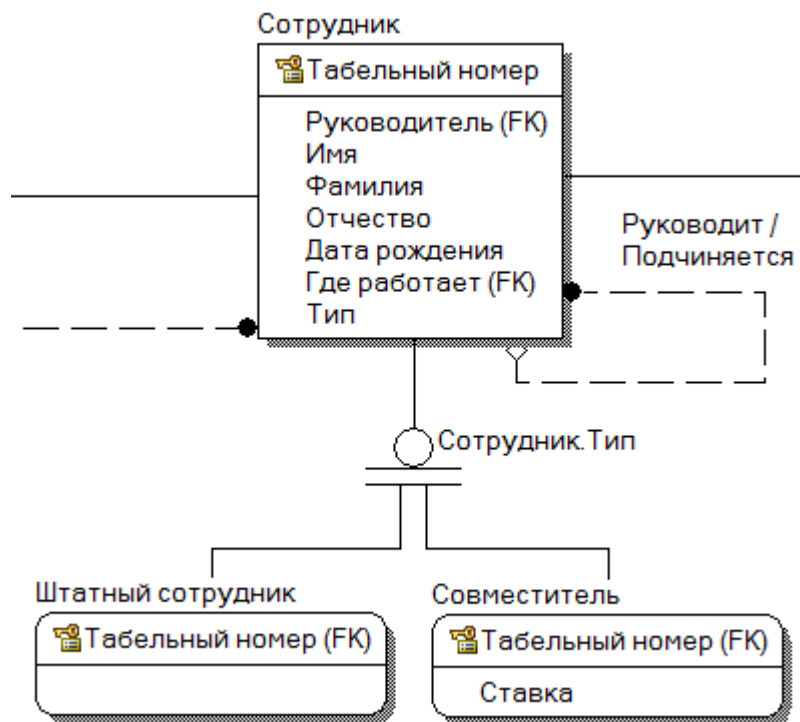



Рис. 44. Иерархия наследования.

Иерархии категорий делятся на два типа - полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке, т. е. в примере служащий обязательно является либо совместителем, либо постоянным сотрудником.

Для создания категориальной связи следует:

- установить курсор на кнопке  в палитре инструментов и нажать левую кнопку мыши;
- щелкнуть сначала по родовому предку, а затем по потомку;
- для установления второй связи в иерархии категории следует сначала щелкнуть по символу категории, затем по второму потомку.

Для редактирования категорий нужно щелкнуть правой кнопкой мыши по символу категории и выбрать в контекстном меню пункт Subtype Relationship Editor. В диалоге Subtype Relationship (Рис. 45) можно указать атрибут - дискриминатор категории (список Discriminator Attribute Choice) и тип категории - полная/неполная (радиокнопки Complete/Incomplete).

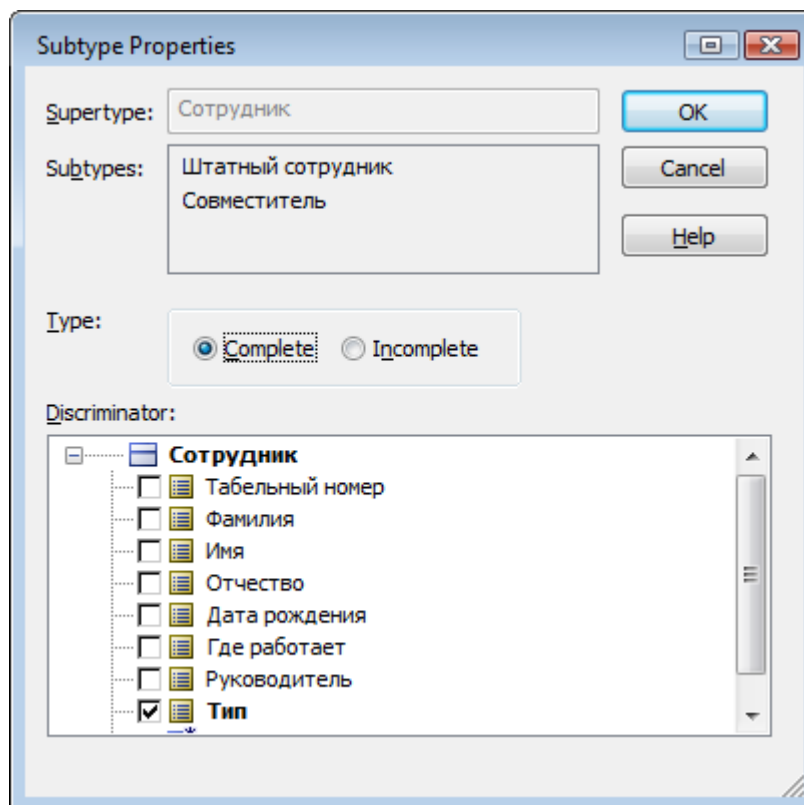


Рис. 45. Диалог Subtype Properties

Рассмотрим возможные стадии построения иерархии наследования. Определение сущностей с общими (по определению) атрибутами. Предположим, в процессе проектирования созданы сущности *Постоянный сотрудник* и *Совместитель*. Можно заметить, что часть атрибутов у этих сущностей (*Фамилия, Имя, Отчество, Дата рождения, Должность*) имеет одинаковый смысл.

Перенос общих атрибутов в сущность - родовой предок. В случае обнаружения совпадающих по смыслу атрибутов следует создать новую сущность (*Сотрудник*) - родовой предок и перенести в нее общие атрибуты (*Фамилия, Имя, Отчество, Дата рождения, Должность*).

Создание неполной структуры категорий. Создается категориальная связь от новой сущности - родového предка к старым сущностям - потомкам. Новая сущность дополняется атрибутом-дискриминатором категории (*Тип*) (см. Рис. 44).

Ключи

Каждый экземпляр сущности должен быть уникален и отличаться от других экземпляров.

Первичный ключ (primary key) - это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности. Атрибуты первичного ключа на диаграмме не требуют специального обозначения - это те атрибуты, которые находятся в списке атрибутов выше горизонтальной линии. При внесении нового атрибута в диалог Attribute Editor для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки General.

Выбор первичного ключа может оказаться непростой задачей, решение которой может повлиять на эффективность будущей ИС. В одной сущности могут оказаться несколько атрибутов или наборов атрибутов, претендующих на роль первичного ключа. Такие претенденты называются потенциальными ключами (candidate key).

Ключи могут быть сложными, т. е. содержащими несколько атрибутов. Сложные первичные ключи не требуют специального обозначения - это список атрибутов выше горизонтальной линии.

Рассмотрим кандидатов на первичный ключ сущности *Сотрудник*.

Здесь можно выделить следующие потенциальные ключи:

1. *Табельный номер*,
2. *Номер паспорта*;
3. *Фамилия + Имя + Отчество*.

Для того чтобы стать первичным, потенциальный ключ должен удовлетворять ряду требований:

Уникальность. Два экземпляра не должны иметь одинаковых значений возможного ключа. Потенциальный ключ № 3 (*Фамилия + Имя + Отчество*) является плохим кандидатом, поскольку в организации могут работать полные тезки.

Компактность. Сложный возможный ключ не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности. Для обеспечения уникальности ключа № 3 дополним его атрибутами *Дата рождения* и *Цвет волос*. Если бизнес-правила говорят, что сочетания атрибутов *Фамилия + Имя + Отчество + Дата рождения* достаточно для однозначной идентификации сотрудника, то *Цвет волос* оказывается лишним, т. е. ключ *Фамилия + Имя + Отчество + Дата рождения + Цвет волос* не является компактным.

При выборе первичного ключа предпочтение должно отдаваться более простым ключам, т. е. ключам, содержащим меньшее количество атрибутов. В примере ключи № 1 и 2 предпочтительней ключа № 3.



Атрибуты ключа не должны содержать нулевых значений. Если допускается, что сотрудник может не иметь паспорта или вместо паспорта иметь какое-либо другое удостоверение личности, то ключ № 2 не подойдет на роль первичного ключа. Если для обеспечения уникальности необходимо дополнить потенциальный ключ дополнительными атрибутами, то они не должны содержать нулевых значений. Дополняя ключ № 3 атрибутом *Дата рождения*, нужно убедиться в том, что даты рождения известны для всех сотрудников.

Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности. Сотрудница организации может выйти замуж и сменить как фамилию, так и паспорт. Поэтому ключи № 2 и 3 не подходят на роль первичного ключа.

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Некоторые сущности могут иметь более одного возможного ключа. Тогда один из них становится первичным, а остальные - альтернативными ключами. Альтернативный ключ (Alternate Key) - это потенциальный ключ, не ставший первичным. ERwin позволяет выделить атри-

буты альтернативных ключей, и по умолчанию в дальнейшем при генерации схемы БД по этим атрибутам будет генерироваться уникальный индекс.

При работе ИС часто бывает необходимо обеспечить доступ к нескольким экземплярам сущности, объединенным каким-либо одним признаком. Для повышения производительности в этом случае используются неуникальные индексы. ERwin позволяет на уровне логической модели назначить атрибуты, которые будут участвовать в неуникальных индексах. Атрибуты, участвующие в неуникальных индексах, называются Inversion Entries (инверсионные входы). Inversion Entry - это атрибут или группа атрибутов, которые не определяют экземпляр сущности уникальным образом, но часто используются для обращения к экземплярам сущности. ERwin генерирует неуникальный индекс для каждого Inversion Entry.

Создать альтернативные ключи и инверсионные входы можно в закладке Key Group диалога Attribute Editor (Рис. 46). Если щелкнуть по кнопке , расположенной в правой верхней части закладки, вызывается диалог Key Group Editor (Рис. 47). В верхней части диалога находится список ключей, в нижней - список атрибутов, доступных для включения в состав ключа (слева), и список ключевых атрибутов. Каждый вновь созданный ключ должен иметь хотя бы один атрибут. Для включения атрибута в состав ключа следует выделить его в левом списке и щелкнуть по кнопке .

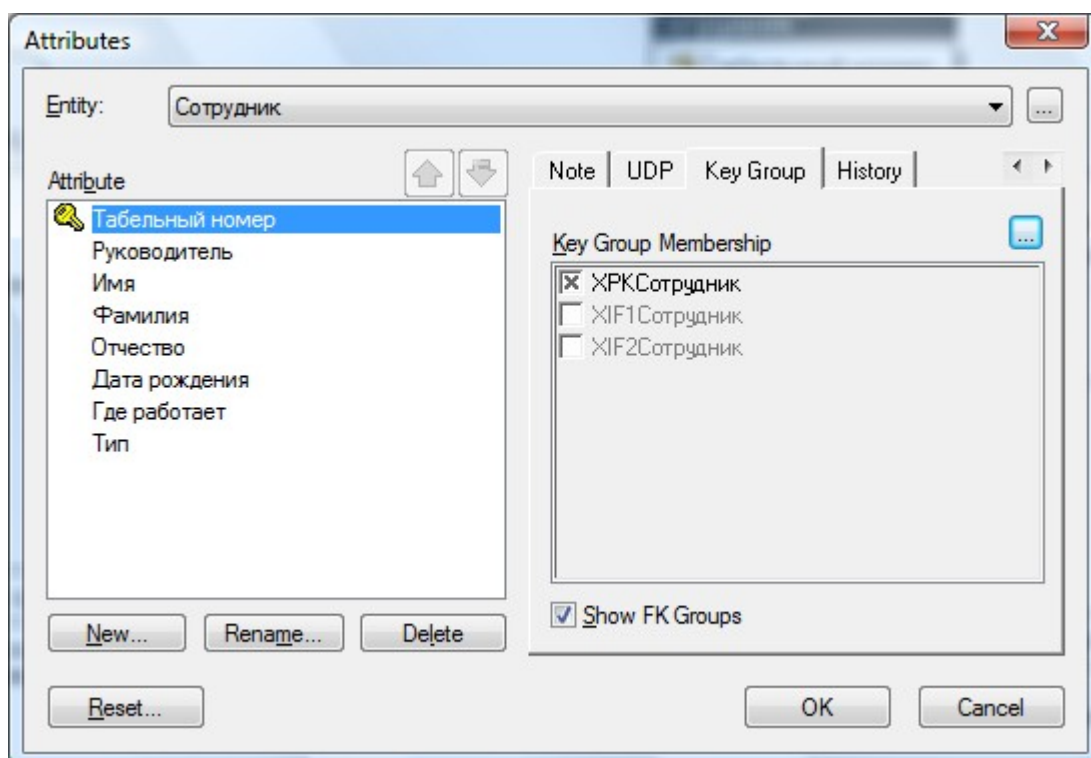


Рис. 46. Закладка Key Group диалога Attributes

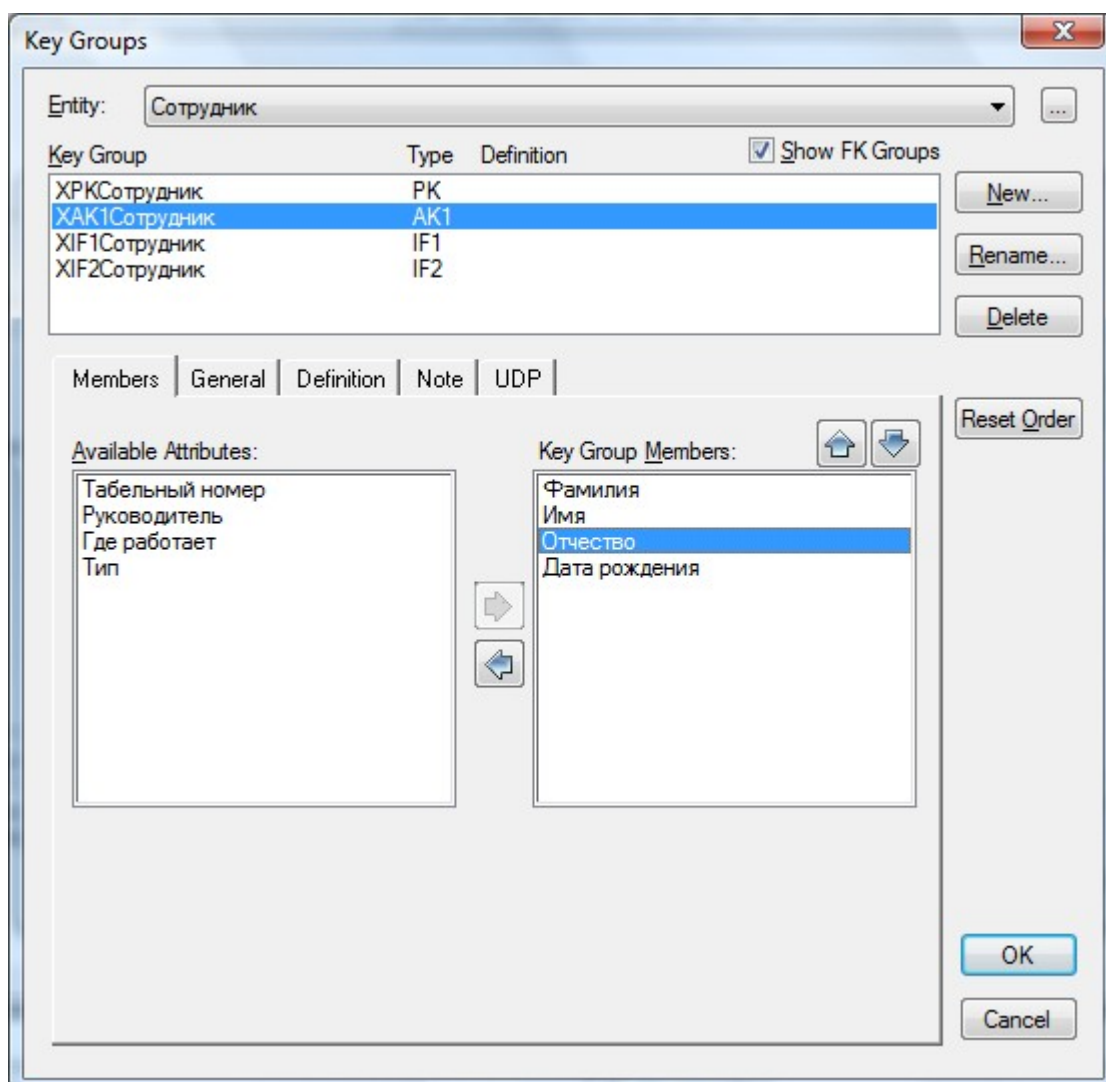


Рис. 47. Диалог Key Groups

Для создания нового ключа следует щелкнуть по кнопке New. Появляется диалог New Key Group (Рис. 48). Имя нового ключа присваивается автоматически ("XAKNENTITY" для альтернативного ключа и "XIENENTITY" для инверсионного входа, где N - порядковый номер ключа, ENTITY - имя сущности).

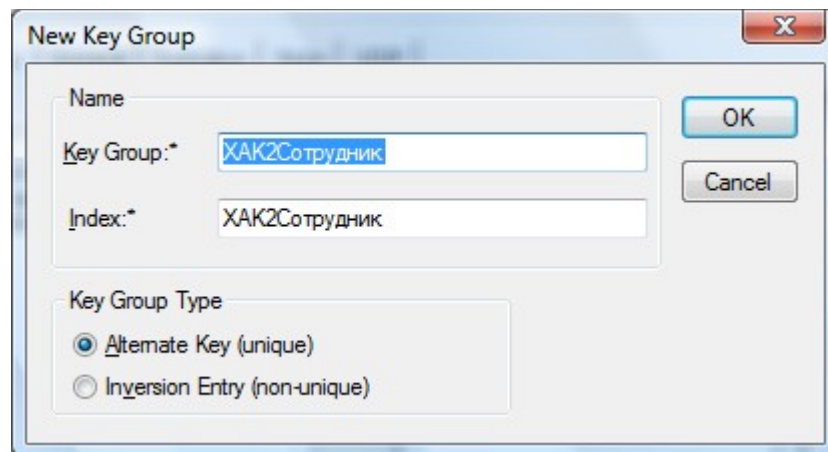


Рис. 48. Диалог New Key Group

Каждому ключу соответствует индекс, имя которого также присваивается автоматически ("XAKNENTITY" для альтернативного ключа и "XIENENTITY" для инверсионного входа, где N - порядковый номер ключа, ENTITY - имя сущности). Имена ключа и индекса при желании можно изменить вручную.

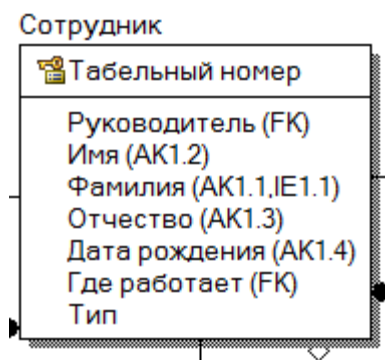


Рис. 49. Сущность "Сотрудник" с отображением ключей

На диаграмме атрибуты альтернативных ключей обозначаются как (AKn.m), где n - порядковый номер ключа, m - порядковый номер атрибута в ключе. Когда альтернативный ключ содержит несколько атрибутов, (AKn.m) ставится после каждого. На Рис. 49 атрибуты *Фамилия*, *Имя*, *Отчество* и *Да-*

та рождения входят в альтернативный ключ № 1 (AK1). Инверсионные входы обозначаются как (IE_{n.m}), где *n* - порядковый номер входа, *m* - порядковый номер атрибута. Инверсионный вход IE1 (атрибут *Фамилия*) позволяет выбрать всех сотрудников с одинаковой фамилией. Если один атрибут входит в состав нескольких ключей, ключи перечисляются в скобках через запятую. По умолчанию номера альтернативных ключей и инверсионных входов рядом с именем атрибута на диаграмме не показываются. Для отображения номера следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели, выбрать пункт Entity Display и затем включить опцию Alternate Key Designator (AK).

Внешние ключи (Foreign Key) создаются автоматически, когда связь соединяет сущности: связь образует ссылку на атрибуты первичного ключа в дочерней сущности и эти атрибуты образуют внешний ключ в дочерней сущности (миграция ключа). Атрибуты внешнего ключа обозначаются символом (FK) после своего имени (см. Рис. 49). Атрибут внешнего ключа *Где работает* ("Где работает" - имя роли) является атрибутом первичного ключа (PK) в сущности *Отдел*.

Нормализация данных

Нормализация - процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам - формализованным требованиям к организации данных. Известны шесть нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- нормальная форма Бойса - Кодда (усиленная 3NF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма (5NF).

На практике обычно ограничиваются приведением данных к третьей нормальной форме (полная атрибутивная модель, 3NF).

Нормальные формы основаны на понятии функциональной зависимости (в дальнейшем будет использоваться термин "зависимость"). Приведем формальное определение для функциональной зависимости.

Функциональная зависимость (FD). Атрибут В сущности Е функционально зависит от атрибута А сущности Е тогда и только тогда, когда каждое значение А в Е связало с ним точно одно значение В в Е, т. е. А однозначно определяет В.

Полная функциональная зависимость. Атрибут В сущности Е полностью функционально зависит от ряда атрибутов А сущности Е тогда и только тогда, когда В функционально зависит от А и не зависит ни от какого подряда А.

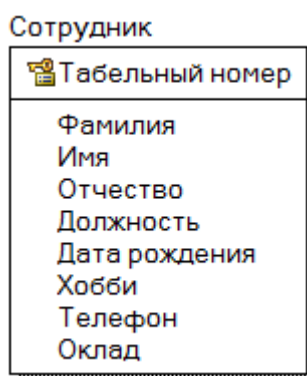


Рис. 50. Ненормализованная сущность "Сотрудник"

На Рис. 50 в сущности *Сотрудник* значение атрибутов *Фамилия*, *Имя* и *Отчество* однозначно определяются значением атрибута *Табельный номер*, т. е. атрибуты *Фамилия*, *Имя* и *Отчество* зависят от атрибута *Табельный но-*

мер. Функциональные зависимости определяются бизнес-правилами предметной области. Так, если оклад сотрудника определяется только должностью, то атрибут *Оклад* зависит от атрибута *Должность*; если оклад зависит еще, например, от стажа, то такой зависимости нет. В нижеследующих примерах будем считать для определенности, что такая зависимость есть.

Рассмотрим нормальные формы.

Первая нормальная форма (1NF). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, т. е. несколько значений для каждого экземпляра. На Рис. 50 атрибуты *Телефон* и *Хобби* являются нарушением первой нормальной формы. Что будет, если у сотрудника несколько рабочих телефонов? Запись значения колонки через разделитель, например "124-56-78, 124-56-79, 124-56-90" или "Аквалангист, мотоциклист, шахматист", приводит к ряду проблем. Размера поля может не хватить для хранения данных (нельзя увеличивать список телефонов до бесконечности), по такой колонке невозможно построить индекс и т. д. и т. п. Сущность, приведенная на Рис. 51, не является решением проблемы. Что будет, если у сотрудника появится четвертый телефон или третье хобби? Эту информацию будет негде хранить.

Сотрудник	
Табельный номер	
Фамилия	
Имя	
Отчество	
Должность	
Дата рождения	
Хобби1	
Хобби2	
Телефон1	
Телефон2	
Телефон3	
Оклад	

Рис. 51. Еще один пример ненормализованной сущности

Для приведения сущности к первой нормальной форме следует:

- разделить сложные атрибуты на атомарные,

- создать новую сущность,
- перенести в нее все "повторяющиеся" атрибуты,
- выбрать возможный ключ для нового РК (или создать новый РК).
- установить идентифицирующую связь от прежней сущности к новой, РК прежней сущности станет внешним ключом (FK) для новой сущности.

На Рис. 52 показана сущность *Сотрудник*, приведенная к первой нормальной форме.

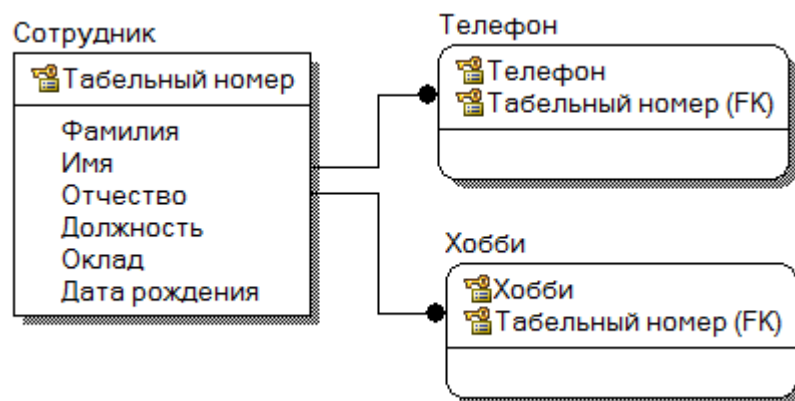


Рис. 52. Сущность "Сотрудник", приведенная к первой нормальной форме

Вторая нормальная форма (2NF). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа). Вторая нормальная форма имеет смысл только для сущностей, имеющих сложный первичный ключ.

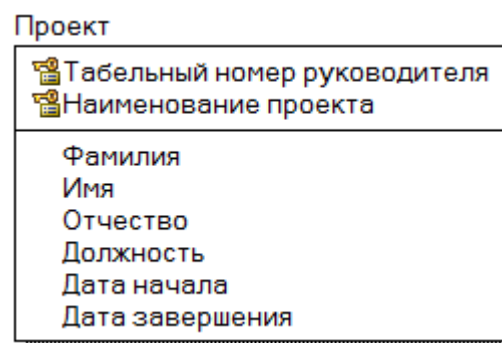


Рис. 53. Сущность "Проект"

Предположим, сущность *Проект* содержит информацию о проекте, которым руководит сотрудник, причем информация содержится как непосредственно о проекте, так и о руководителе проекта (Рис. 53). Атрибуты *Фамилия*, *Имя*, *Отчество* и *Должность* зависят только от атрибута *Табельный номер* руководителя, но вовсе не от *Наименования проекта*. Другими словами, имеется зависимость только от части ключа.

Для приведения сущности ко второй нормальной форме следует:

- выделить атрибуты, которые зависят только от части первичного ключа, создать новую сущность;
- поместить атрибуты, зависящие от части ключа, в их собственную (новую) сущность;
- установить идентифицирующую связь от прежней сущности к новой (Рис. 54).



Рис. 54. Сущность "Проект", приведенная ко второй нормальной форме

Вторая нормальная форма позволяет избежать следующих аномалий при выполнении операций:

Обновление (UPDATE). Имеет место дублирование данных о сотруднике, если он руководит несколькими проектами. Если данные о сотруднике изменяются, необходимо менять несколько записей (по числу ведомых проектов).

Вставка (INSERT). Невозможно ввести данные о сотруднике, если он в данный момент не руководит проектами.

Удаление (DELETE). Если сотрудник временно прекращает руководство проектами, данные о нем теряются.

На Рис. 54 показана сущность *Проект*, приведенная ко второй нормальной форме.

Третья нормальная форма (3NF). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

На Рис. 52 сущность *Сотрудник* находится во второй нормальной форме (имеется только один атрибут первичного ключа, поэтому не может быть зависимости неключевых атрибутов от части ключа), но неключевой атрибут *Оклад* зависит от другого неключевого атрибута - *Должности*.

Для приведения сущности ко второй нормальной форме следует:

- создать новую сущность и перенести в нее атрибуты с одной и той же зависимостью от неключевого атрибута;
- использовать атрибут(ы), определяющий эту зависимость, в качестве первичного ключа новой сущности;
- установить неидентифицирующую связь от новой сущности к старой (Рис. 55).



Рис. 55. Сущность "Сотрудник", приведенная к третьей нормальной форме

В третьей нормальной форме каждый атрибут сущности зависит от ключа, от всего ключа целиком и ни от чего другого, кроме как от ключа. Третья нормальная форма также позволяет избежать ряда аномалий:

- Обновление (UPDATE). Имеет место дублирование данных об окладе, если должность занимают несколько сотрудников. Если оклад соответствующих должности меняется, необходимо менять несколько записей (по числу сотрудников на одной должности).
- Вставка (INSERT). Невозможно ввести данные об окладе, соответствующем должности, если в данный момент нет сотрудника, занимающего эту должность.
- Удаление (DELETE). В случае удаления из таблицы сотрудника, занимающего уникальную должность, данные об окладе теряются.

Домены

Домен можно определить как совокупность значений, из которых берутся значения атрибутов. Каждый атрибут может быть определен только на одном домене, но на каждом домене может быть определено множество атрибутов. В понятие домена входит не только тип данных, но и область значений данных. Например, можно определить домен "Возраст" как положительное целое число и определить атрибут *Возраст сотрудника* как принадлежащий этому домену.

Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам БД на этапе эксплуатации системы. На логическом уровне домены можно описать без конкретных физических свойств. На физическом уровне они автоматически получают специфические свойства, которые можно изменить вручную. Так, домен "Возраст" может иметь на логическом уровне тип Number, на физическом уровне колонкам домена будет присвоен тип INTEGER.

Для создания домена в логической модели служит диалог Domain Dictionary Editor, (Рис. 56). Его можно вызвать из меню Model/Domain

Dictionary или по кнопке, расположенной в верхней левой части закладки General диалога Attributes. Для создания нового домена в диалоге Domain Dictionary следует:

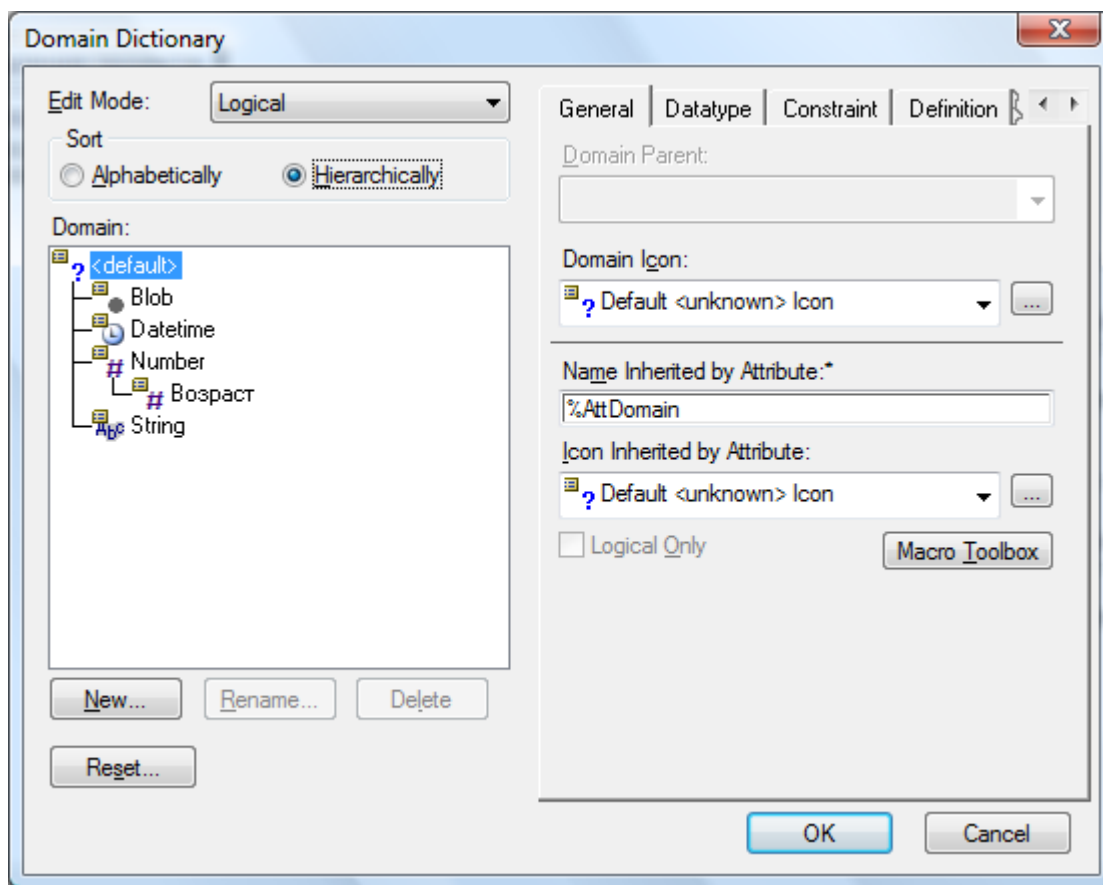


Рис. 56. Диалог Domain Dictionary

- щелкнуть по кнопке New. Появляется диалог New Domain (Рис. 57);
- выбрать родительский домен из списка Domain Parent. Новый домен можно создать на основе уже созданного пользователем домена либо на основе изначально существующего. По умолчанию ERwin имеет четыре предопределенных домена (String, Number, Blob, Datetime). Новый домен наследует все свойства родительского домена. Эти свойства в дальнейшем можно переопределить;
- набрать имя домена в поле Logical Name. Можно также указать имя домена на физическом уровне в поле Physical Name. Если физическое имя не указано, по умолчанию оно принимает значение логического имени;
- щелкнуть по кнопке ОК.

В диалоге Domain Dictionary Editor можно связать домен и иконкой, с которой он будет отображаться в списке доменов (Domain Icon), и иконкой, с которой атрибут, определенный на домене, будет отображаться в модели (Icon Inherited by Attribute).

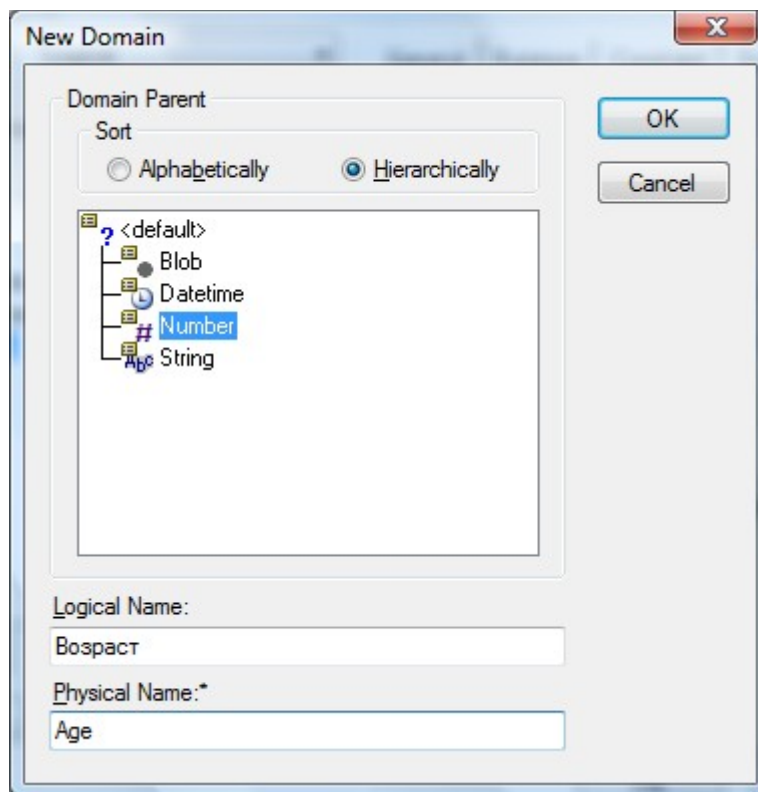


Рис. 57. Диалог *New Domain*

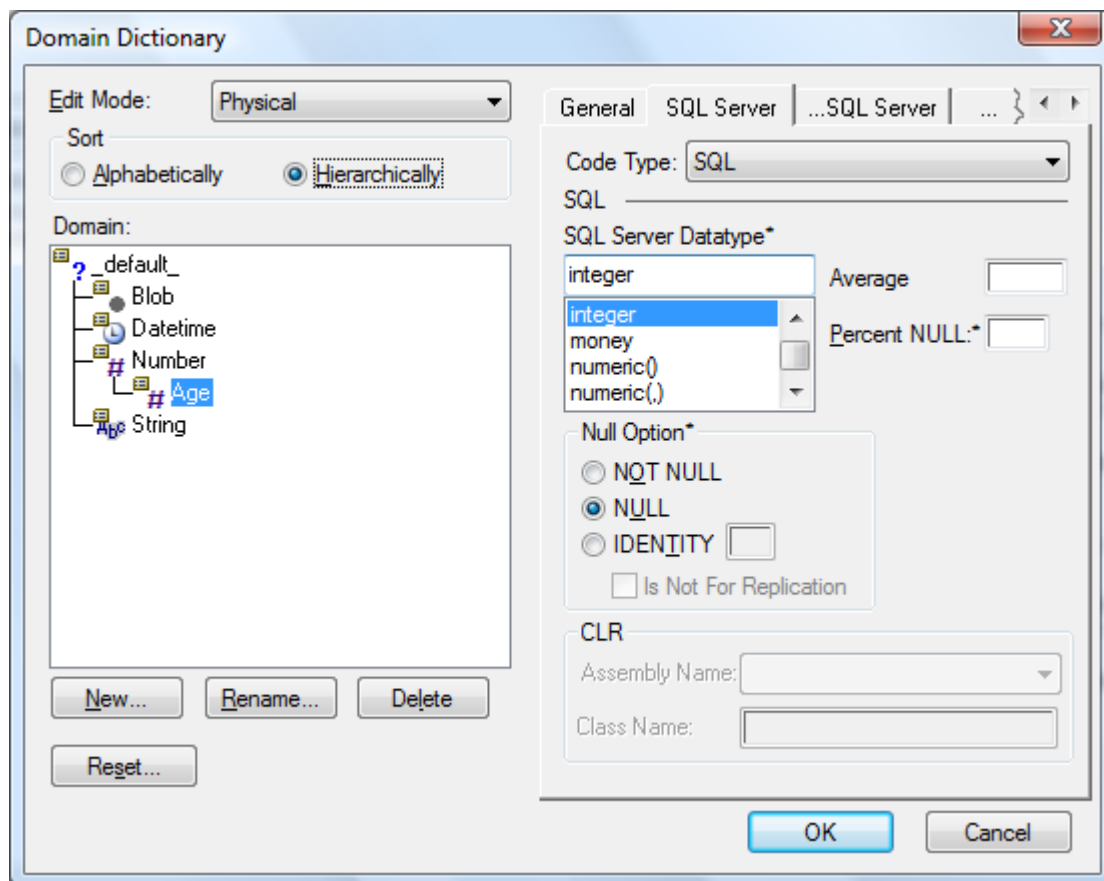


Рис. 58. Диалог Domain Dictionary на физическом уровне

На физическом уровне диалог Domain Dictionary позволяет редактировать физические свойства домена. На Рис. 58 показана закладка SQL Server. Имя этой закладки зависит от выбранного сервера БД. На ней можно задать конкретный тип данных, соответствующих домену, правила присвоения NULL-значений, правила валидации (правила проверки допустимых значений) и задания значения по умолчанию.

Рассмотрим функции других закладок диалога Domain Dictionary:

- General (Рис. 59). Задание родительского домена (Domain Parent) и имени, присваиваемого колонке при ее создании с помощью Independent Column Browser. С помощью опции Physical Only домен можно определить только на уровне физической модели.
- Comment. Внесение комментария к атрибуту.
- UDP. Свойства, определяемые пользователем.
- Visual Basic - PowerBuilder. Задание специальных свойств домена для кодогенерации клиентского приложения.

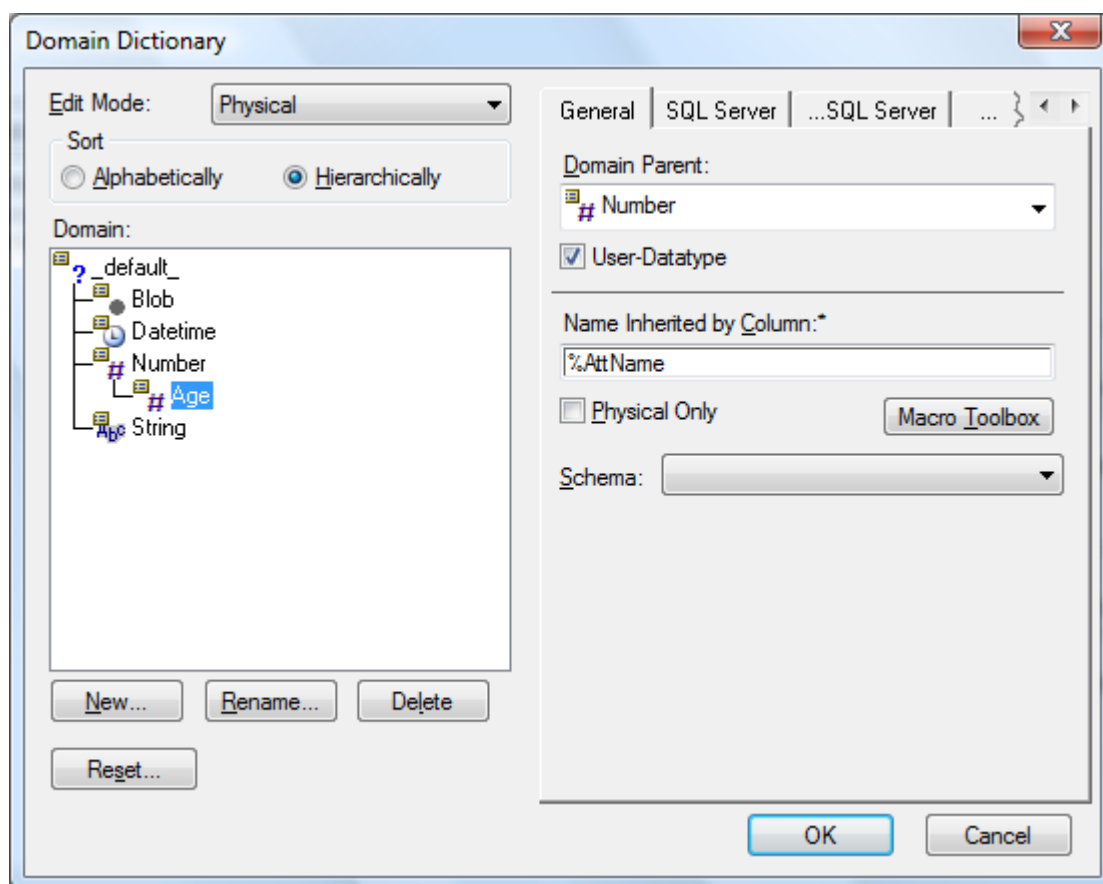


Рис. 59. Закладка *General* диалога *Domain Dictionary Editor*

Домены могут быть использованы при генерации схемы БД для создания типов, определяемых пользователем для тех СУБД, которые поддерживают такие конструкции (DB2, Rdb, Interbase, SQL Anywhere, SQL Server и SYBASE). Типы, определяемые пользователем, представляют собой синонимы существующих в БД типов, создаваемых для удобства работы с данными.

При выборе соответствующего сервера на закладке General появляется флажок:

- Distinct Types - для DB2/CS и DB2/UDB;
- Domains - для Rdb и Interbase;
- User Datatypes - для SQL Anywhere, SQL Server и SYBASE.

Для создания типов, определяемых пользователем, необходимо включить соответствующую опцию. При генерации схемы БД колонки, определенные на соответствующем домене, будут иметь тип, определяемый пользователем.

Задание на лабораторную работу

Доопределить недостающие сущности, атрибуты и связи в модели, построенной на предыдущем занятии. Привести модель к третьей нормальной форме.

Лабораторная работа №6

Создание физической модели данных

Цель работы

Создание физической модели.

Уровни физической модели

Различают два уровня физической модели:

- трансформационная модель (Transformation Model);
- модель СУБД (DBMS Model).

Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. ERwin поддерживает ведение отдельных проектов, позволяя проектировщику выделять подмножество модели в виде предметных областей (Subject Area). Трансформационная модель позволяет проектировщикам и администраторам БД лучше представлять, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель данных удовлетворяет требованиям к ИС.

Модель СУБД автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД. ERwin непосредственно поддерживает эту модель путем генерации системного каталога.

Выбор сервера

Физический уровень представления модели зависит от выбранного сервера. Для выбора СУБД служит редактор Target Server (меню Server/Target Server... доступно только на физическом уровне) (Рис. 60).

ERwin поддерживает практически все распространенные СУБД, всего более 20 реляционных и нереляционных БД. Для выбора СУБД нужно щелкнуть по соответствующей кнопке рядом с именем СУБД.

Диалог Target Server позволяет задать тип данных и опцию NULL для новых колонок. Тип данных по умолчанию можно выбрать в раскрывающемся списке Default Datatype, который автоматически заполняется типами данных, поддерживаемых выбранным сервером.

Группа кнопок Default Non-Key Null Option позволяет разрешить или запретить значения NULL для неключевых колонок.

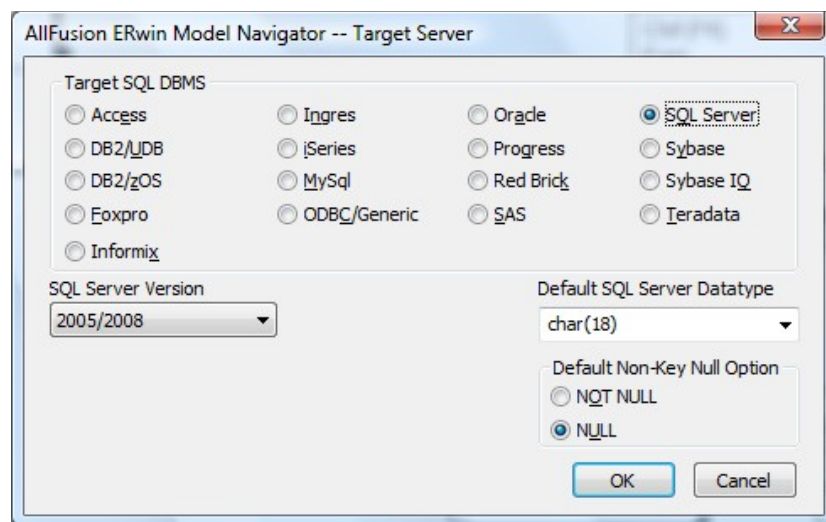



Рис. 60. Диалог Target Server

Таблицы, колонки и представления (view)

Для внесения новой таблицы в модель на физическом уровне служит кнопка  на палитре инструментов. Связи между таблицами создаются также, как на логическом уровне. Щелкнув правой клавишей мыши по таблице и выбрав во всплывающем меню пункты Table Editor или Column Editor, можно вызвать редакторы для задания свойств таблиц и колонок.

ERwin автоматически создает имена таблиц и колонок на основе имен соответствующих сущностей и атрибутов, учитывая максимальную длину имени и другие синтаксические ограничения, накладываемые СУБД. При генерации имени таблицы или колонки по умолчанию все пробелы автоматически преобразуются в символы подчеркивания, а длина имени обрезается до максимальной длины, допустимой для выбранной СУБД. Все изменения, сделанные в Table Editor или Column Editor, не отражаются на именах сущно-

стей и атрибутов, поскольку информация на логическом и физическом уровнях в ERwin хранится отдельно.

Редактор Table Editor позволяет задать свойства любой таблицы модели, отличные от значения по умолчанию, в том числе имя таблицы, синонимы, правила валидации, процедуры и т. д. Переключиться на другую таблицу можно при помощи раскрывающегося списка выбора в верхней части диалога (Рис. 61).

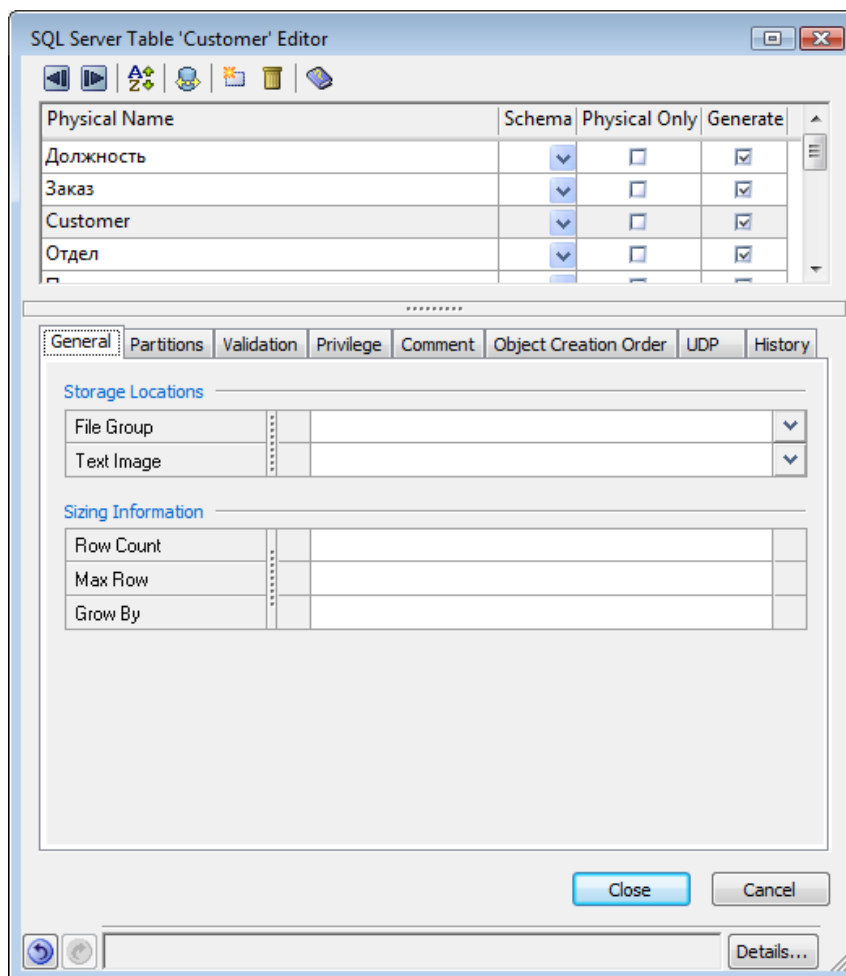



Рис. 61. Диалог Table Editor для SQL Server

Вид этого диалога (и других) может существенно зависеть от выбранной базы данных. В верхней части окна для SQL Server показан список таблиц. В колонке Name можно задать имена таблицы в соответствии с ограничениями выбранной СУБД. В колонке Physical Only служит для указания, что таблица существует только на физическом уровне. Если выбрана опция Generate, при генерации схемы БД будет выполняться команда CREATE TABLE. Кнопка

 (DB Sync) служит для немедленной синхронизации модели с системным каталогом БД. Ниже показан диалог для Oracle.

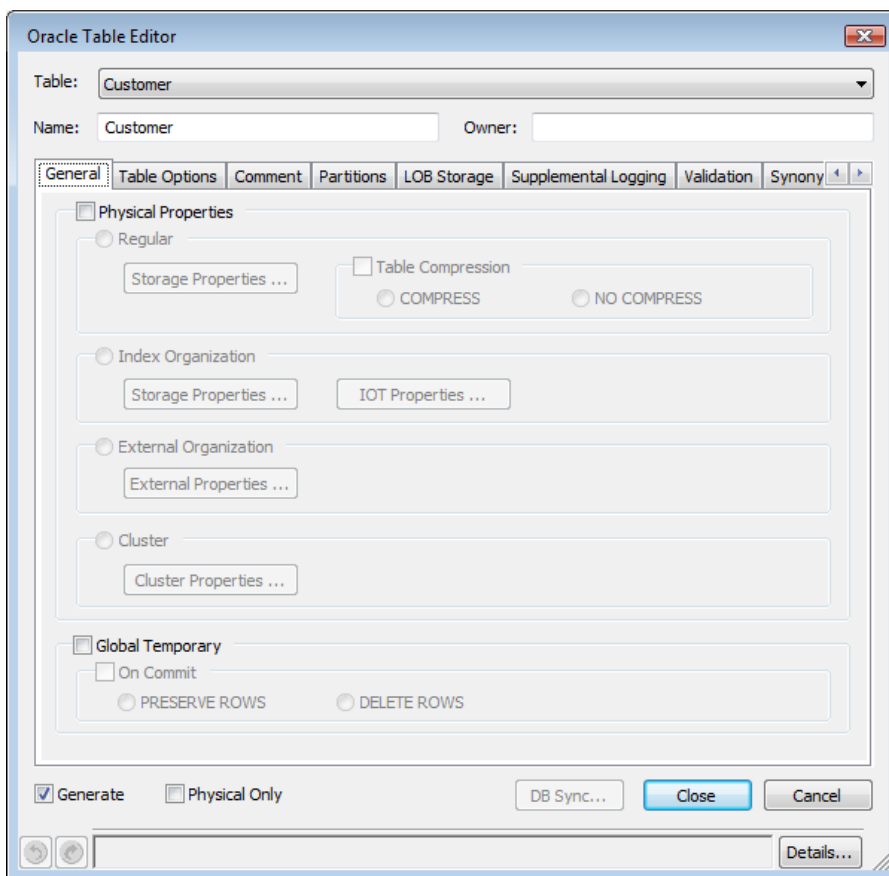


Рис. 62. Диалог Table Editor для Oracle

Диалог Table Editor может содержать следующие закладки:

General. Служит для оценки размера БД или задания физических свойств таблицы..

Partitions. Служит для задания значений деления.

Validation. Задание правил валидации.

Comment. Внесение комментария к таблице.

Synonym. Задание синонимов таблицы (если сервер таковые поддерживает).

Object Creation Order. Служит для создания функций, скриптов которые будут выполняться до и после создания таблицы при генерации схемы БД, хранимых процедур.

UDP. Задание свойств, определяемых пользователем.

History. История работы с таблицей.

Для задания свойств колонок, отличных от значения по умолчанию, служит редактор Column Editor (Рис. 63). Чтобы вызвать его, нужно щелкнуть правой клавишей мыши по таблице и выбрать во всплывающем меню пункт Column Editor.

По умолчанию ERwin присваивает режимы нулевых значений всем не-ключевым колонкам, исходя из значений по умолчанию, устанавливаемых в редакторе Target Server. Для колонок первичного ключа и альтернативных ключей устанавливается режим NOT NULL. Режим NOT NULL не присваивается автоматически инверсионным входам (Inversion Entry).

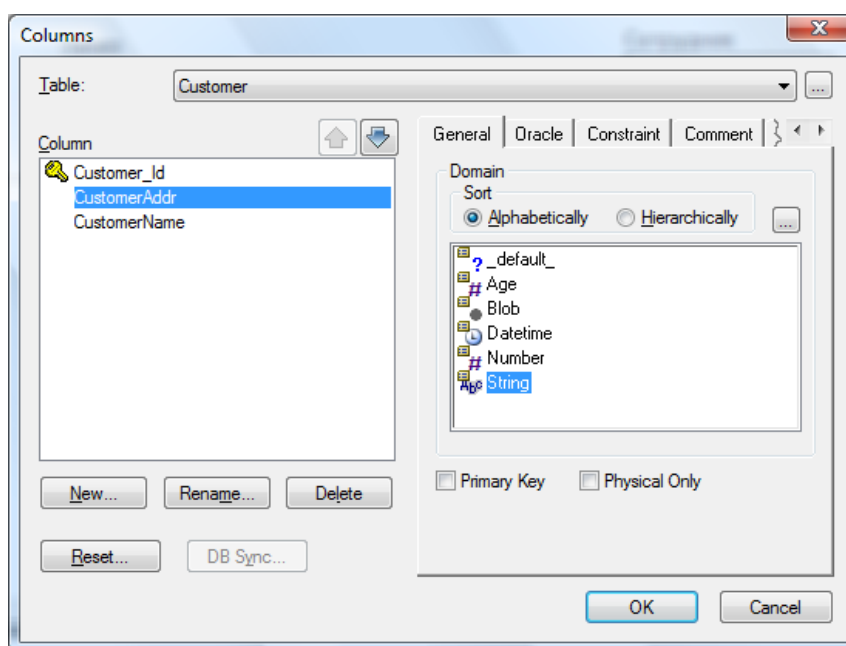


Рис. 63. Диалог Column Editor

Внешне диалог Column Editor напоминает диалог Attribute Editor. В правой части диалога находятся закладки:

General. Позволяет задать колонке определенный домен, создать колонку только на физическом уровне и включить ее в состав первичного ключа.

Закладка, соответствующая выбранной СУБД (на Рис. 63 и Рис. 64 -ORACLE). Имя закладки устанавливается автоматически соответствующей выбранной СУБД. Позволяет задать тип данных, опцию NULL.

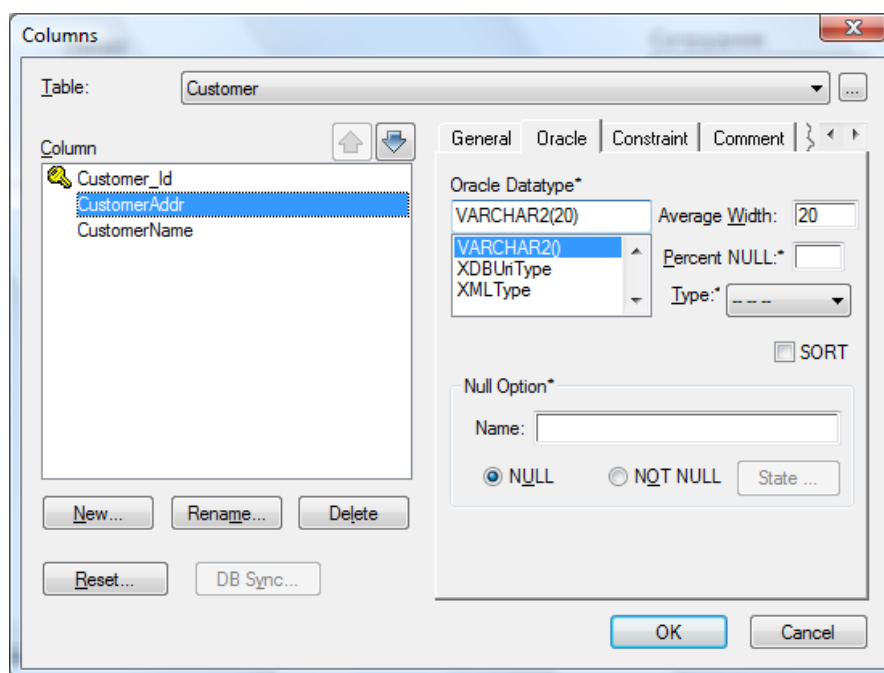


Рис. 64. Закладка СУБД диалога *Column Editor*

Constraint. Позволяет задать правила валидации и значения по умолчанию. Правила валидации и значение по умолчанию должны быть описаны и именованы предварительно соответственно в диалогах *Validation Rule Editor* и *Default/Initial Editor*. Для вызова этих диалогов служат кнопки !!! справа от соответствующих раскрывающихся списков. Для СУБД Access, AS/400, PROGRESS и Teradata создаются дополнительные закладки для задания свойств.

Comment. Служит для внесения комментария к каждой колонке.

UDP. Задание свойств, определяемых пользователем.

Index. Служит для включения колонки в состав индексов.

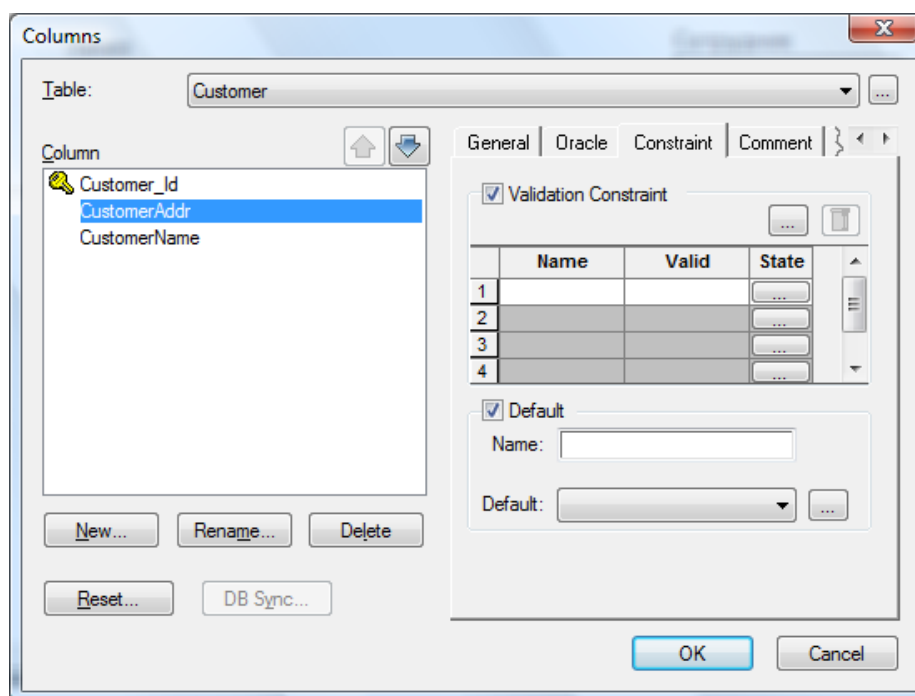





Рис. 65. Закладка Constraint диалога Column Editor

В левой части диалога содержится упорядоченный список колонок таблицы. Кнопки  предназначены для перемещения колонки в списке на позицию вверх и вниз. Кнопки New, Rename и Delete служат соответственно для создания, переименования и удаления колонки. При помощи кнопки Reset можно переустановить свойства колонки, заданные вручную, на значения по умолчанию. Кнопка DB Sync позволяет запустить процесс синхронизации модели с системным каталогом БД.

Представления (view), или, как их иногда называют, временные или производные таблицы, представляют собой объекты БД, данные в которых не хранятся постоянно, как в таблице, а формируются динамически при обращении к представлению. Представление не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц. Применение представлений позволяет разработчику БД обеспечить каждому пользователю или группе пользователей свой взгляд на данные, что решает проблемы простоты использования и безопасности данных. ERwin имеет специальные инструменты для создания и редактирования представлений. Палитра инструментов на физическом уровне содержит кнопки внесения представле-

ний и установления связей между таблицами и представлениями. Для внесения представления нужно щелкнуть по кнопке  в палитре инструментов, затем по свободному месту диаграммы. По умолчанию представление получает номер V_n, где n - уникальный порядковый номер представления. Для установления связи нужно щелкнуть по кнопке  затем по родительской таблице и, наконец, по представлению (Рис. 66). Связи с представлениями и прямоугольники представлений показываются на диаграмме пунктирными линиями.

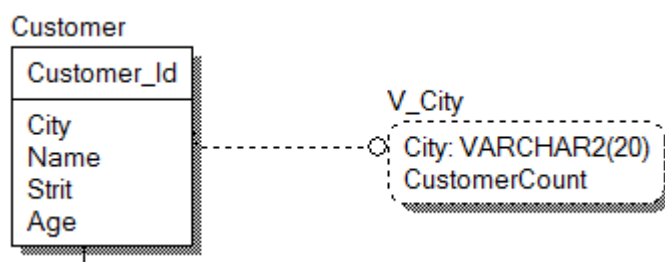


Рис. 66. Создание представления

Для редактирования представления служит диалог View Editor (Рис. 67). Для его вызова следует щелкнуть правой кнопкой мыши по представлению и выбрать в меню пункт View Editor.

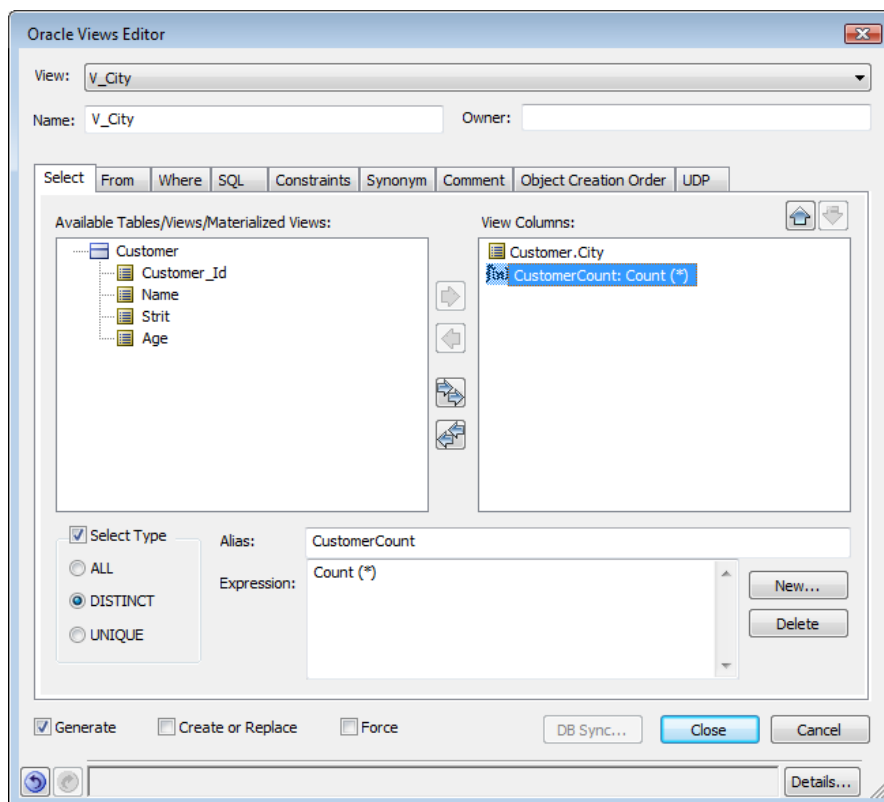


Рис. 67. Диалог View Editor

Раскрывающийся список View позволяет выбрать для редактирования любое представление модели. Окно Name служит для редактирования имени, а Owner-владельца представления.

Диалог View Editor имеет следующие закладки:

Select (Рис. 67). Имеет два списка: в правом отображаются колонки представления, в левом - колонки доступные для включения в представление. Кнопка New позволяет задать выражение в качестве выходного столбца. Например, для представления V_City на Рис. 66 в качестве колонок созданы City и выражение с именем CustomerCount, которое представляет собой агрегативную функцию, подсчитывающую количество строк, Count(*).

From. Позволяет выбрать родительские таблицы представления. По умолчанию включаются таблицы, с которыми связано представление. Каждой таблице можно задать синоним (поле Alias), который будет использоваться при создании SQL-команды создания представления.

Where. Закладка содержит три поля - Where, Group By и Having. На основе этой информации Erwin генерирует SQL-команду создания представления, причем на основе содержания этих полей генерируются предложения SQL-запроса. Для представления V_43 в поле Where содержатся значения "Country='Россия'", Group By - "City", Having - "Count(*)>2". В результате представление будет содержать информацию о количестве клиентов в российских городах, при условии, что количество клиентов в этих городах больше двух.

SQL. Закладка содержит поле, в котором отображается SQL-запрос создания представления и окно выбора User-Defined SQL. По умолчанию опция User-Defined SQL выключена, и SQL-запрос генерируется автоматически на основе информации, занесенной в закладках Select, From и Where. Запрос можно задать вручную, включив эту опцию, но в этом случае список полей и связи представления на диаграмме отображаться не будут. Для представления City на Рис. 66 SQL-запрос будет выглядеть так:

```
"CREATE VIEW V_City (City, CustomerCount) AS
```

```

SELECT CUSTOMER.City, Count(*)
FROM CUSTOMER
WHERE Country= 'Россия'
GROUP BY City
HAVING Count(*)>2"

```

В закладке Comment можно внести комментарий для представления.

Object Creation Order позволяет связать с представлением хранимые процедуры, команды, выполняемые до и после генерации представления.

UDP позволяет связать с представлением свойства, определяемые пользователем.

Для редактирования свойств колонок представления служит редактор View Column Editor (Рис. 68). Для его вызова следует щелкнуть правой кнопкой мыши по представлению и выбрать в меню пункт View Column Editor.

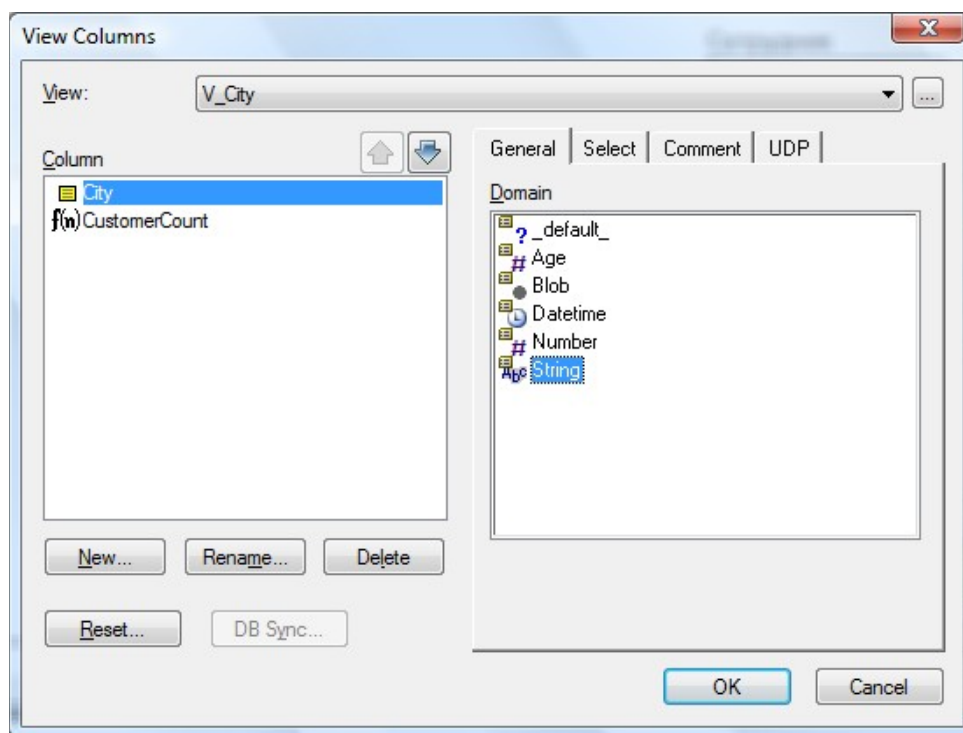


Рис. 68. Диалог View Column Editor

Редактор содержит следующие закладки:

General. Позволяет ассоциировать колонку с доменом. По умолчанию колонка представления принадлежит тому же домену, что и колонка родительской таблицы.


Select. Так же как в диалоге View Editor (закладка Select, кнопка New Expression), здесь можно создать выражение (в том числе включающее агрегативные функции) для колонки.

Comment содержит комментарий для каждой колонки.

UDP позволяет связать с колонкой свойства, определяемые пользователем.

Правила валидации и значения по умолчанию

ERwin поддерживает правила валидации для колонок, а также значение, присваиваемое колонкам по умолчанию. Правило валидации задает список допустимых значений для конкретной колонки 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. С каждой колонкой или доменом можно связать значение по умолчанию (если выбранная СУБД поддерживает домены).

Если щелкнуть по кнопке , расположенной справа от раскрывающегося списка Valid (см. Рис. 65), появляется диалог Validation Rules (Рис. 69), который служит для задания правил валидации. В нем можно задать максимальное и минимальное значение и тип валидации (где проверять - на сервере или в клиентском приложении).

Например, в таблице *CUSTOMER* значение, вводимое в колонку. *Age*, должно быть больше 18, но меньше 180. Для описания этого правила можно создать правило валидации с именем "Проверка_возраста", которое содержит выражение: *Age BETWEEN 18 AND 180*. Использование этого правила валидации гарантирует, что диапазон вводимых значений будет от 18 до 180. СУБД выдаст сообщение об ошибке, если вводимый возраст находится вне границ заданного диапазона.

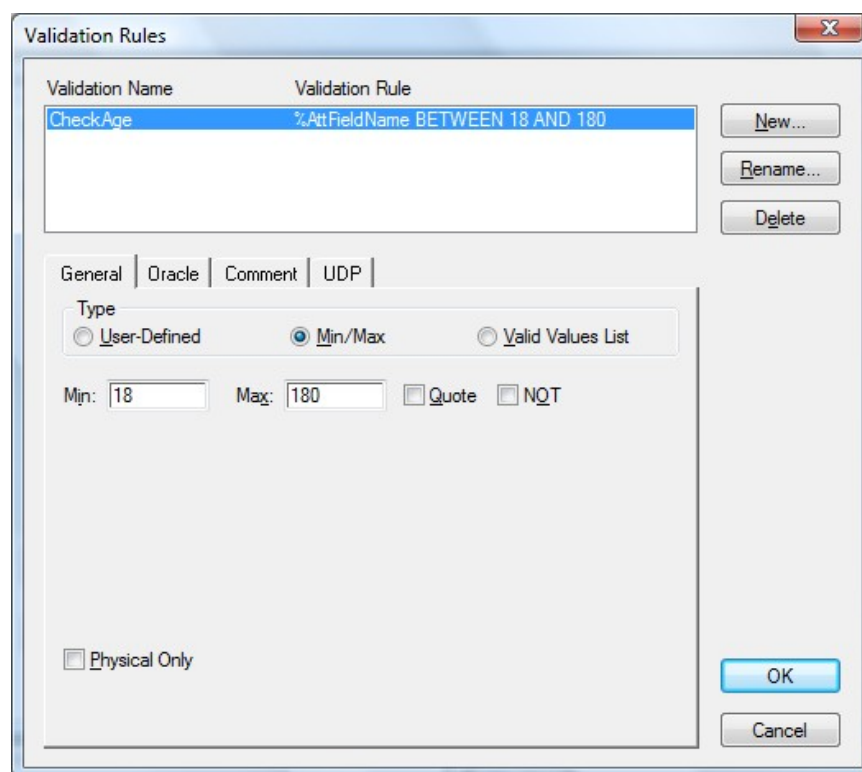


Рис. 69. Диалог *Validation Rules*

В верхней части окна *Validation Rules* содержится список всех существующих правил валидации. Для создания нового правила валидации следует щелкнуть по кнопке *New*, ввести имя правила в поле *Name* диалога *New Validation Rule* (Рис. 70) и щелкнуть по кнопке *OK*. После этого можно ввести выражение для правила валидации. Минимальное и максимальное значение можно ввести включив на закладке *General* переключатель *Min/Max* (Рис. 69).

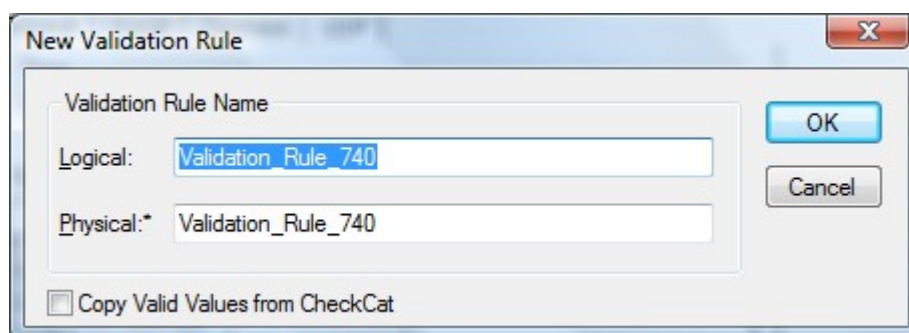


Рис. 70

Установив переключатель в положение *Valid Value List* можно перечислить все допустимые значения (Рис. 71). Например, если в таблице *CUSTOMER* имеется колонка *Category*, то можно задать список допустимых

значений для соответствующей колонки, который может содержать значения "Местный", "Иногородный" и "Иностранный".

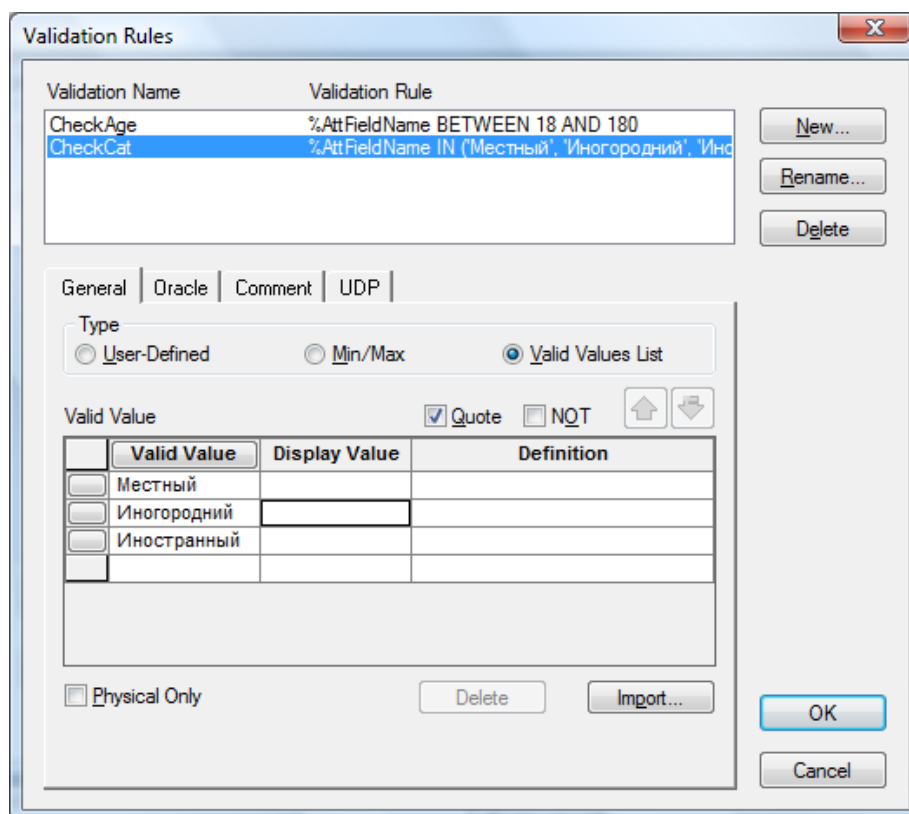



Рис. 71. Закладка *Valid Values List*

По умолчанию ERwin создает выражение - команду языка СУБД, используя значения, связанные с правилом валидации, и разделяя значения запятыми (например, C, D, M). В некоторых случаях правила синтаксиса СУБД требуют, чтобы каждое значение в команде заключалось в одинарные кавычки ('C', 'D', 'M'). Чтобы автоматически заключить каждое значение в одинарные кавычки, нужно включить опцию Quote.

Редактор Default/Initial Editor (Рис. 72) позволяет создать значение, которое автоматически, по умолчанию, присваивается колонке. Для вызова редактора служит кнопка  справа от раскрывающегося списка Default диалога Column Editor (см. Рис. 65). Например, дате приема сотрудника может быть присвоено значение по умолчанию "сегодняшнее число", т. е. автоматически задается, что все новые сотрудники зачисляются в день ввода информации о них в БД.

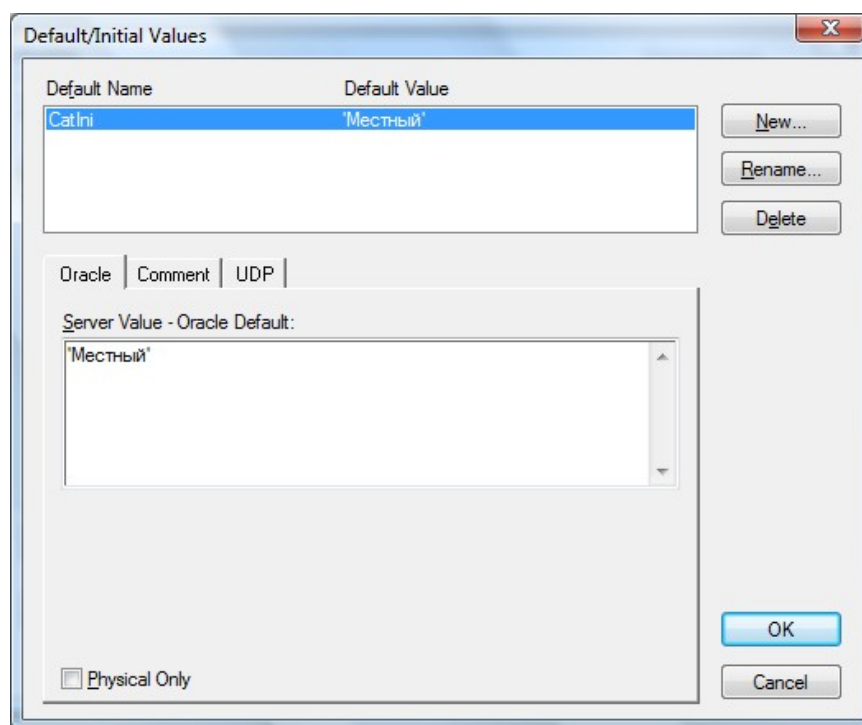


Рис. 72. Диалог *Default/Initial Editor*

Для создания нового значения по умолчанию следует щелкнуть по кнопке New, ввести имя правила в поле Name диалога New Default и щелкнуть по кнопке OK. В окне Default Name показывается список всех имен значений по умолчанию.

После создания правила валидации и значения по умолчанию их можно присвоить одной или нескольким колонкам или доменам.

Индексы

В таблице БД данные обычно хранятся в том же порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой физически таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно. Хотя такой способ хранения и позволяет быстро вводить новые данные, но для того, чтобы найти нужную строку, придется просмотреть всю таблицу. В промышленных системах каждая таблица может содержать миллионы строк, поэтому простой перебор ведет к катастрофическому падению производительности ИС.

Чтобы решить проблему поиска данных, СУБД использует особый объект, называемый индексом. Он подобен содержанию книги, которое указывает на все номера страниц, посвященных конкретной теме. Индекс содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки.

Например, если необходимо найти клиента по имени, можно создать индекс по колонке *CustomerName* таблицы *CUSTOMER*. В индексе имена клиентов будут отсортированы в алфавитном порядке. Для имени индекс будет содержать ссылку, указывающую, в каком месте таблицы хранится эта строка.

Для поиска клиента серверу направляется запрос с критерием поиска (*CusfomerName* = "Иванов"). При выполнении запроса СУБД просматривает индекс, вместо того чтобы просматривать по порядку все строки таблицы *CUSTOMER*. Поскольку значения в индексе хранятся в определенном порядке, просматривать нужно гораздо меньший объем данных, что значительно уменьшает время выполнения запроса. Индекс можно создать для всех колонок таблицы, по которым часто производится поиск.

При генерации схемы физической БД ERwin автоматически создает отдельный индекс на основе первичного ключа каждой таблицы, а также на основе всех альтернативных ключей, внешних ключей и инверсионных входов, поскольку эти колонки наиболее часто используются для поиска данных. Можно отказаться от генерации индексов по умолчанию и для повышения производительности создать собственные индексы. Администратор СУБД должен анализировать наиболее часто выполняемые запросы и создавать индексы с различными колонками и порядком сортировки для увеличения эффективности поиска при работе конкретных приложений.

ERwin автоматически генерирует имя индекса, созданного на основе ключа по принципу "X" + имя ключа + имя таблицы (физическое имя таблицы, а не логическое имя сущности!), где имя ключа "PK" для первичного

ключа, "IFn" - для внешнего, "AKn" - для альтернативного, "IEн" - для инверсионного входа.

Изменить характеристики существующего индекса или создать новый можно в редакторе Index Editor (Рис. 73). Для его вызова следует щелкнуть правой кнопкой мыши по таблице и выбрать во всплывающем меню пункт Indexes.

В редакторе Index Editor можно изменить имя индекса, изменить его определение так, чтобы он принимал уникальные или дублирующиеся значения, или изменить порядок сортировки данных.

ERwin создает индексы, которые могут содержать либо повторяющиеся, либо только уникальные значения. При создании нового уникального индекса (кнопка New, диалог New Index, Рис. 74) следует включить опцию Unique, для создания индекса с неповторяющимися значениями опцию следует выключить. Если на основе колонки создается уникальный индекс, то при попытке вставить запись с неуникальным (повторяющимся) значением сервер выдаст ошибку и значение не будет вставлено. Например, уникальный индекс в таблице *CUSTOMER*, построенный на колонке *QustomerName*, предотвратит от внесения двух строк с информацией об одном и том же клиенте.

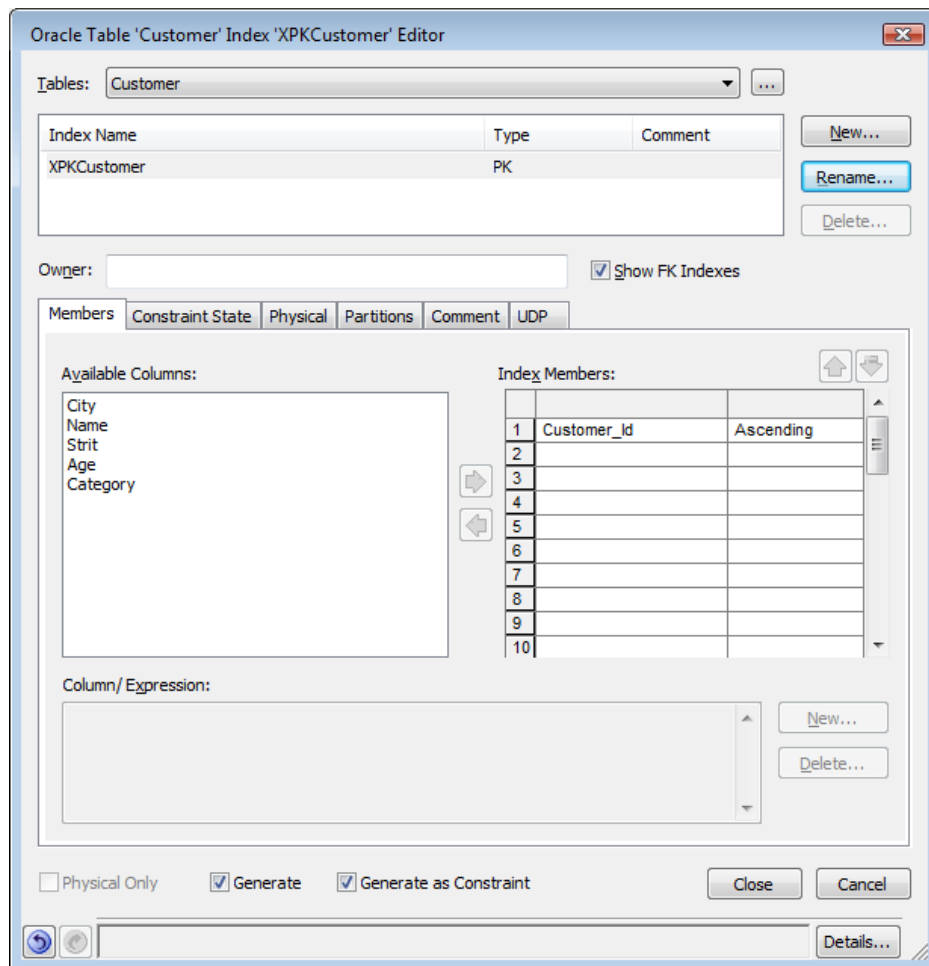


Рис. 73. Диалог Index Editor

И наоборот, при создании нового индекса автоматически создается альтернативный ключ для уникального и инверсионный вход для неуникального индекса, а также соответствующее ключу имя индекса. Имя сгенерированного индекса в дальнейшем при необходимости можно изменить вручную.

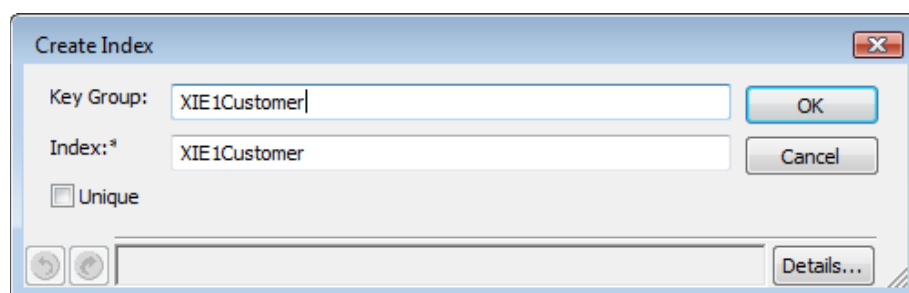


Рис. 74. Диалог New Index

Триггеры и хранимые процедуры

Триггеры и хранимые процедуры - это именованные блоки кода SQL, которые заранее откомпилированы и хранятся на сервере для того, чтобы бы-

стро производить выполнение запросов, валидацию данных и выполнять другие часто вызываемые функции.

Хранение и выполнение кода на сервере позволяет создавать код только один раз, а не в каждом приложении, работающем с БД, что экономит время при написании и сопровождении программ. При этом гарантируется, что целостность данных и бизнес-правила поддерживаются независимо от того, какое именно клиентское приложение обращается к данным. Триггеры и хранимые процедуры не требуется пересылать по сети из клиентского приложения, что значительно снижает сетевой трафик.

Хранимой процедурой называется именованный набор предварительно откомпилированных команд SQL, который может вызываться из клиентского приложения или из другой хранимой процедуры.

Триггером называется процедура, которая выполняется автоматически как реакция на событие. Таким событием может быть вставка, изменение или удаление строки в существующей таблице. Триггер сообщает СУБД, какие действия нужно выполнить при выполнении команд SQL INSERT, UPDATE или DELETE для обеспечения дополнительной функциональности, выполняемой на сервере.

Триггер ссылочной целостности - особый вид триггера, используемый для поддержания целостности между двумя таблицами, которые связаны между собой. Если строка в одной таблице вставляется, изменяется или удаляется, то триггер ссылочной целостности (RI-триггер) сообщает СУБД, что нужно делать с теми строками в других таблицах, у которых значение внешнего ключа совпадает со значением первичного ключа вставленной (измененной, удаленной) строки. По умолчанию ERwin генерирует триггеры, дублирующие декларативную ссылочную целостность. Например, если удаляется клиент из таблицы CUSTOMER, то в зависимости от установленных правил ссылочной целостности могут быть сгенерированы RI-триггеры, которые будут воздействовать на соответствующие удаляемому клиенту заказы из таблицы ORDER.

В отличие от триггера хранимая процедура не выполняется в ответ на какое-то событие, а вызывается из другой программы, которая передает на сервер имя процедуры. Хранимая процедура более гибка, чем триггер, поскольку может вызывать другие хранимые процедуры. Ей можно передавать параметры, и она может возвращать параметры, значения и сообщения.

ERwin не имеет встроенных шаблонов хранимых процедур, которые можно было бы использовать как основу при создании новой хранимой процедуры. Для создания или редактирования хранимой процедуры следует щелкнуть правой кнопкой мыши по таблице и выбрать в контекстном меню пункт Stored Procedures. Появляется окно Procedures Editor (Рис. 75).

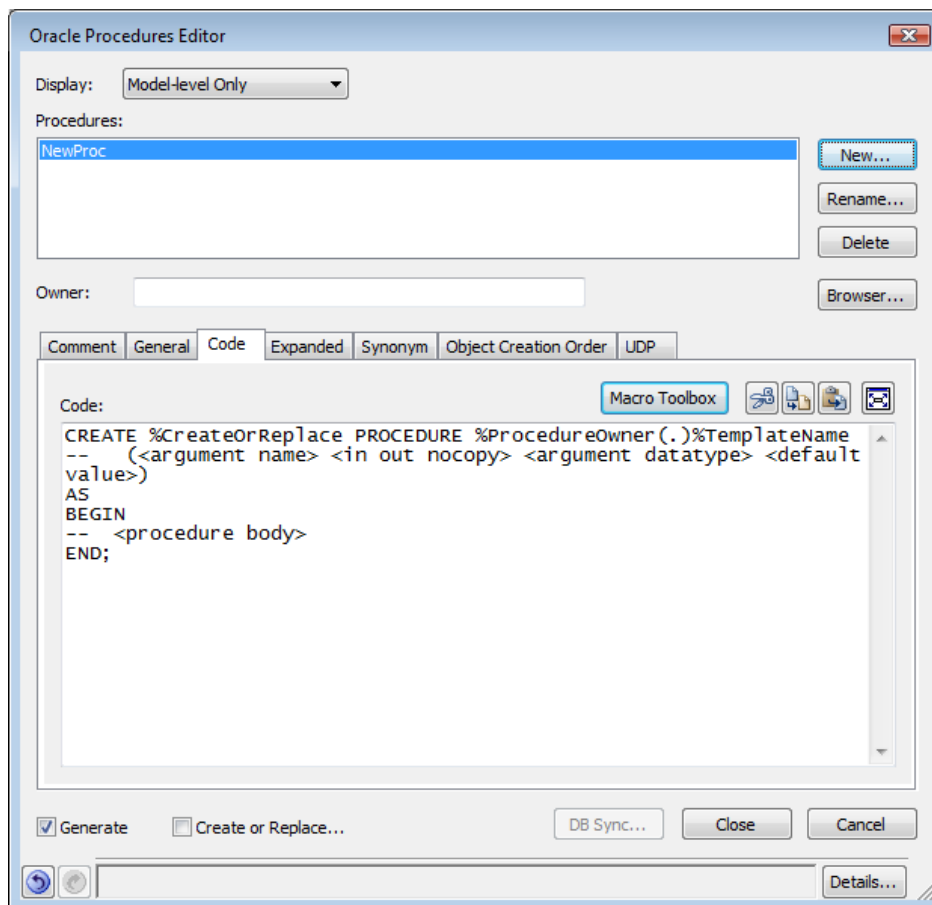


Рис. 75. Окно Procedures Editor

ERwin позволяет связывать хранимые процедуры не только с отдельными таблицами, но и со всей моделью. Например, это может быть хранимая процедура, которая выполняет административную функцию.

Скрипты "до и после генерации". Скриптами "до и после генерации" (pre&post scripts schema-generation) называются скрипты SQL, которые

ERwin выполняет до или сразу же после генерации таблиц или схемы в целом (pre&post schema-generation). Например, при прямом проектировании БД из модели ERwin может выполнить скрипт "до генерации схемы", который удаляет старую БД и создает новую до того, как начать генерацию таблиц и индексов.

Скрипты уровня таблиц могут быть созданы в окне *Script Template Editor*, которое вызывается щелчком правой кнопкой мыши по таблице (пункт меню Pre&Post Scripts).

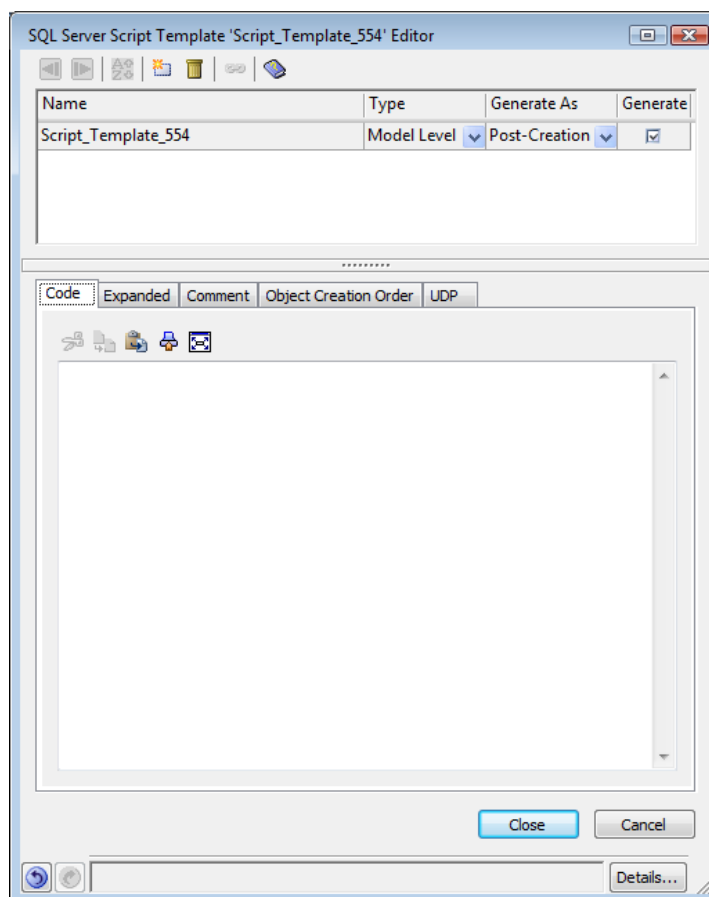



Рис. 76. Окно *Script Template Editor*

Прямое проектирование

Процесс генерации физической схемы БД из логической модели данных называется прямым проектированием (Forward Engineering). При генерации физической схемы ERwin включает триггеры ссылочной целостности, хранимые процедуры, индексы, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД. Процесс генерации логической

модели из физической БД называется обратным проектированием (Reverse Engineering). ERwin позволяет создать модель данных путем обратного проектирования имеющейся БД. После того как модель создана, можно переключиться на другой сервер (модель будет конвертирована) и произвести прямое проектирование структуры БД для другой СУБД. Кроме режима прямого и обратного проектирования ERwin поддерживает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания ИС.

Для генерации системного каталога БД следует выбрать пункт меню Tools/Forward Engineer/Schema Generation или нажать кнопку  на панели инструментов. Появляется диалог Schema Generation (Рис. 77).

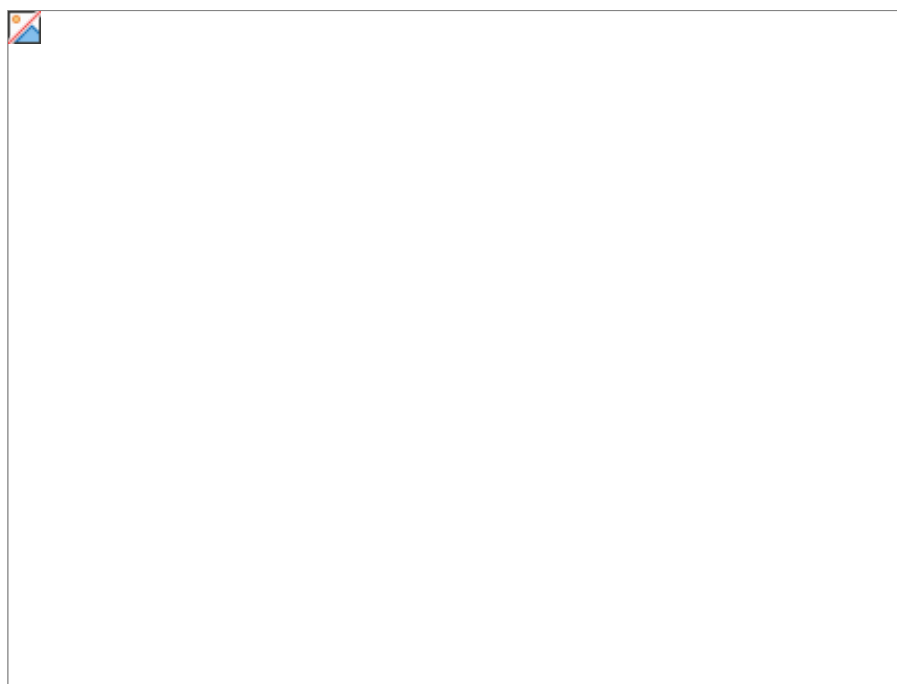


Рис. 77. Диалог *Schema Generation*

Диалог Schema Generation имеет три закладки:

Options. Служит для задания опций генерации объектов БД - триггеров, таблиц, представлений, колонок, индексов и т. д. Для задания опций генерации какого-либо объекта следует выбрать объект в левом списке закладки, после чего включить соответствующую опцию в правом списке (см. Рис. 77).

В закладке **Summary** отображаются все опции, заданные в закладке **Options**. Список опций в **Summary** можно редактировать так же, как и в **Options**.

Comment. Позволяет внести комментарий для каждого набора опций. Каждый набор опций может быть именован (окно **Option Set**, кнопки **New**, **Rename** и **Delete**) и использован многократно.

Кнопка **Preview** вызывает диалог **Schema Generation Preview** (Рис. 78), в котором отображается SQL-скрипт, создаваемый ERwin для генерации системного каталога СУБД. Нажатие на кнопку **Generate** приведет к запуску процесса генерации схемы.

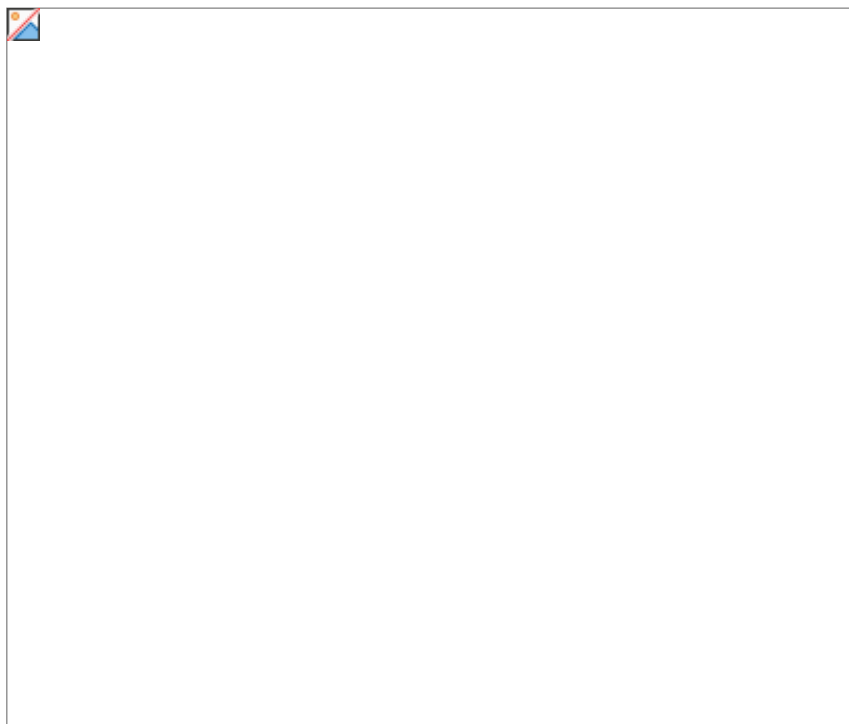


Рис. 78. Диалог *Schema Generation Preview*

Кнопка **Print** предназначена для вывода на печать создаваемого ERwin SQL-скрипта.

Кнопка **Report** сохраняет тот же скрипт в ERS или SQL текстовом файле. Эти команды можно в дальнейшем редактировать любым текстовым редактором и выполнять при помощи соответствующей утилиты сервера.

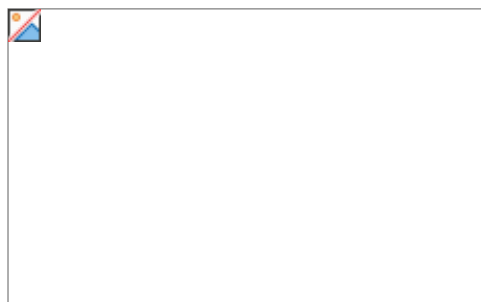


Рис. 79. Диалог связи с БД

Кнопка Generate запускает процесс генерации схемы. Возникает диалог связи с БД (Рис. 79), устанавливается сеанс связи с сервером и начинает выполняться SQL-скрипт. При этом возникает диалог Generate Database Schema (Рис. 80).

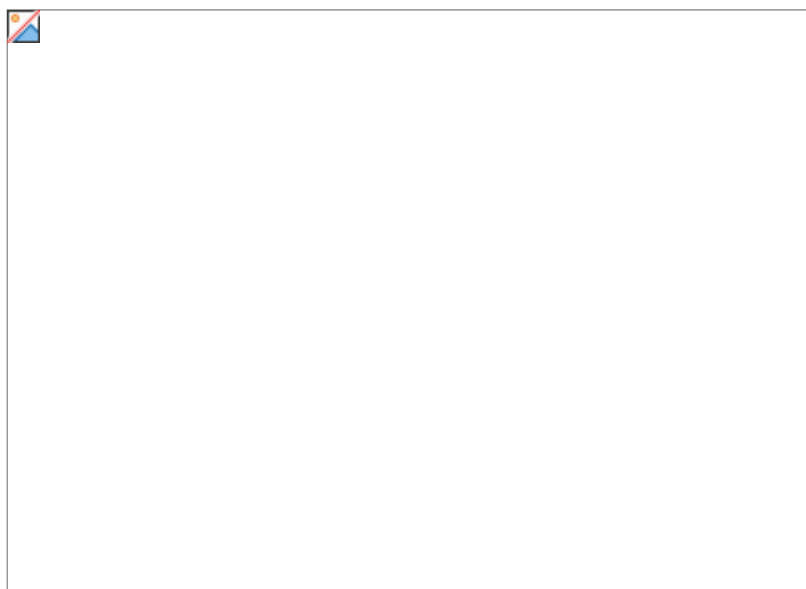


Рис. 80. Диалог Generate Database Schema

По умолчанию в диалоге Generate Database Schema включена опция Stop If Failure. Это означает, что при первой же ошибке выполнение скрипта прекращается. Щелкнув по кнопке Continue можно продолжить выполнение. Кнопка Abort прерывает выполнение. При выключенной опции Stop It Failure скрипт будет выполняться, несмотря на встречающиеся ошибки.