МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(национальный исследовательский университет)»

Проектирование баз данных

Электронные методические указания
к лабораторным работам

САМАРА
2010

Составители: **Додонов Михаил Витальевич,**
Сопченко Елена Вильевна

Рецензент: Авсиевич А.В.

В пособии приведены методические указания к лабораторным работам по разделу "Проектирование баз данных".
Учебное пособие предназначено для студентов, обучающихся по магистерской программе 010300.68 "Фундаментальная информатика и информационные технологии" и изучающих дисциплину "Распределенная обработка данных в современных СУБД".

Разработано на кафедре программных систем.

Введение

Анализ современных тенденций развития автоматизированных систем обработки информации и управления (АСОИУ) однозначно свидетельствует об усложнении используемых в них структур данных при одновременном увеличении количества хранимой и обрабатываемой информации. Опыт работы многих разработчиков показывает, что, как правило, редко удается создать идеальную АСОИУ.

После ввода в эксплуатацию АСОИУ выявляются недочеты (нередко связанные не с ошибками программистов, а с неверно сформулированным техническим заданием). У пользователей возникают новые потребности (создать тот или иной дополнительный сервис, модернизировать структуру таблиц, добавить новые таблицы, запросы или отчеты, изменить выходные формы или предусмотреть возможность их редактирования). Начинается переработка исходных текстов, структур таблиц, добавление «заплат». В конечном итоге получается программный продукт, в исходном тексте которого никто, кроме автора, разобраться не в состоянии. Примерно так же выглядит и используемая структура данных. При этом связи между таблицами могут быть просто неочевидными.

Следовательно, необходимо изначально предусматривать возможность безболезненной модернизации АСОИУ. Поэтому нужны иные подходы к проектированию АСОИУ, по сравнению с теми, что применялись до недавнего времени. Одним из таких подходов и является применение специализированных средств проектирования структур данных – так называемых CASE-средств (Computer Aided Software Engineering или Computer Aided System Engineering).

CASE-технология представляет собой методологию проектирования АСОИУ, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех этапах разработки и сопровождения АСОИУ, разрабатывать приложения в соответствии с информационными потребностями пользователей. Большинство существующих CASE-средств основано на методологиях структурного (в основном) или объектно-ориентированного анализа и проектирования. В них используются спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

CASE-средства позволяют автоматизировать процесс создания структур данных, создавать серверные части приложений, вносить в них логику приложения, правила контроля целостности данных, а также приписывать полям в таблицах расширенные атрибуты (тип интерфейсного элемента, максимальные и минимальные значения, значения по умолчанию и т.д.). Немаловажно, что в процессе проектирования и перепроектирования структур данных можно документировать создаваемые таблицы, их поля и связи между ними и отображать графически полученную структуру.

Одним из CASE-средств, наиболее удачных с точки зрения соотношения цены, простоты использования и возможностей, является ERwin фирмы Computer Associates. Данное средство обладает всеми перечисленными возможностями и при этом имеет удобный интерфейс, интуитивно понятные инструменты, разнообразные возможности графического представления структуры данных. Это средство поддерживает много разных форматов плоских таблиц и серверных БД и умеет создавать триггеры и хранимые процедуры на соответствующих процедурных расширениях языка запросов SQL, поддерживаемых обслуживаемыми серверами (в случае Oracle это PL/SQL). При этом, про-

ектируя структуру данных, проектировщики не обязаны знать ни SQL, ни его расширения.

ERwin - это графический инструментарий для моделирования данных, основной целью которого является помощь аналитику в использовании правил и требований к информации при создании логических и физических моделей данных. Как и многие другие инструментальные средства, ERwin следует использовать тому, кто понимает, для чего именно этот инструмент предназначен, и кто способен использовать его наиболее продуктивно.

Моделирование данных представляет собой деятельность по формированию и документированию требований к информации. Требования к информации описывают данные и правила, необходимые для поддержки профессиональной деятельности. Модель данных может выражать как сложные информационные потребности целой корпорации, так и конкретные информационные потребности одной единственной программы.

Модель данных является визуальным представлением структур данных, данных и правил для СУБД. Обычно она разрабатывается как часть более крупного проекта по разработке программного обеспечения (ПО). Модель данных состоит из двух компонент - логической и физической моделей. В большинстве случаев первой создается логическая модель, а уже потом модель физическая. Иногда модель данных получается путем реконструкции из существующей базы данных. Процесс реконструкции иногда называют обратным проектированием.

В большинстве случаев для построения модели данных потребуется выполнить следующие операции:

1. Определение проблемы и функциональных границ предметной области.
2. Формирование требований.
3. Анализ предметной области.
4. Формирование логической модели.
5. Формирование физической модели.
6. Генерация базы данных.

Логическое моделирование должно производиться в процессе разработки как можно раньше, часто уже на этапе формирования проблемы. Кроме того, логическое моделирование данных является мощным средством как для определения и документирования требований к данным, так и для выявления правил, описывающих способы использования данных. Фактически, многие профессионалы считают, что модель данных должна служить фундаментом проекта разработки ПО.

1. Создание новой модели данных

Существует два разных уровня моделирования - логический уровень (Logical) и физический уровень (Physical). Логическое моделирование данных начинается с определения проблемы. Этот шаг иногда определяют как формулирование цели или функциональных границ проекта. Определение проблемы может состоять из одного параграфа или представлять собой сложный документ, отражающий систему целей. Эта операция задает функциональные границы модели данных.

Понятие *логический уровень* подразумевает, что мы мыслим в понятиях реального мира и непосредственно из него берем объекты для моделирования. Например, подразделения, сотрудники, студенты, аудитории, документы - это реальные объекты. Объекты модели, представляемые на логическом уровне, называются сущностями, представлен-

ными атрибутами, и должны получать имена из естественного языка, с использованием таких разделителей (пробелов, черточек и т.п.), которые имеют смысл для заказчика. На логическом уровне не имеет значения, например, какая СУБД будет использоваться, является ли некоторое число целым или действительным, как лучше индексировать таблицу. Таким образом, логическая модель данных является универсальной и никак не зависит от конкретной СУБД.

Результатом моделирования данных должна быть структура БД, т.е. отображение системного каталога конкретной СУБД. Таким образом, при построении полной модели данных необходимо логическую модель выразить в терминах физической базы данных, включая выбор СУБД, типов данных по умолчанию, эффективных схем хранения и индексирования. Это принято называть *физическим уровнем* модели данных. В физической модели содержится информация обо всех объектах СУБД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Таким образом, одной и той же логической модели могут соответствовать несколько разных физических моделей. Разделение модели данных на логические и физические позволяет решить несколько важных задач:

1. Документирование модели (возможность обсуждать структуру данных с конечными пользователями или экспертами предметной области с использованием понятной им терминологии и независимо от ограничений выбранной СУБД, например, требований именований объектов).
2. Масштабирование (возможность построения независимой от СУБД модели данных, что позволяет разрабатывать гетерогенные АСОИУ и переносить структуру данных из одной СУБД в другую СУБД, например, из Microsoft Access в Oracle).

Уровень представления (Model Type) указывается при создании новой модели данных. Новую модель данных можно создать следующими двумя способами:

- прямым проектированием (Forward Engineering, см. рис. 1),
- обратным проектированием (Reverse Engineering, см. рис. 2).

При прямом проектировании системный каталог СУБД или соответствующие команды создания БД генерируется на основе физического уровня представления модели данных. При обратном проектировании, модель данных создается на основе системного каталога или соответствующих команд создания БД, т.е. данный способ используется, когда необходимо изменить или перенести в другую СУБД существующую структуру БД.

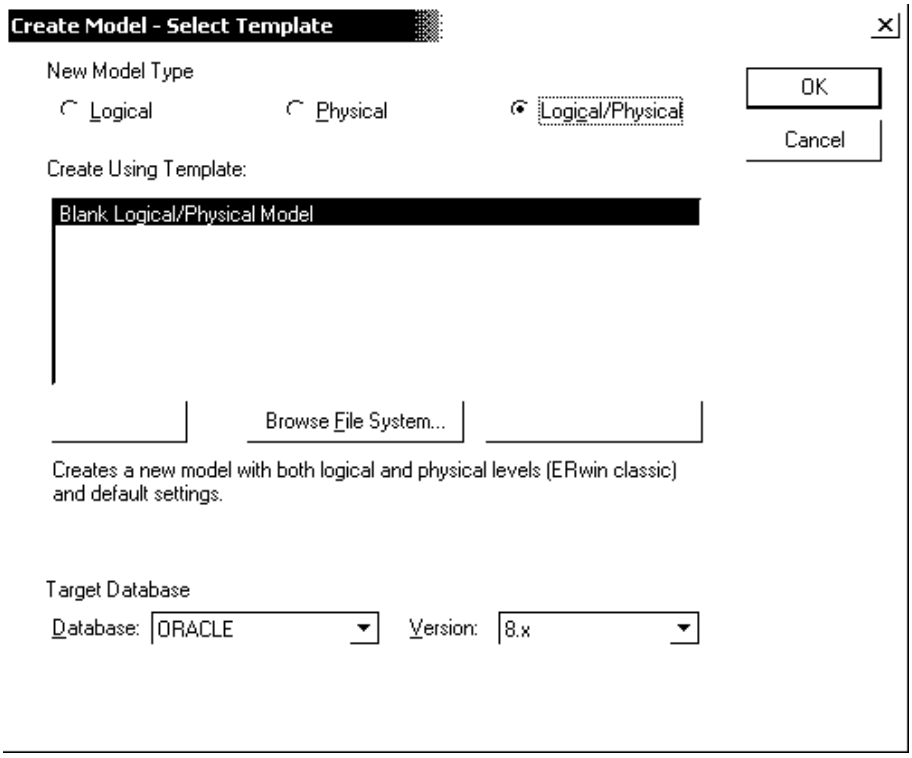

Рис. 1. Диалоговое окно Create Model

Обычно выбирают оба уровня представление модели данных: Logical/Physical (см. рис. 1, рис. 2). Если будет использоваться модель данных на физическом уровне, то необходимо указать СУБД (Target Database) и, если требуется, версию (Version).

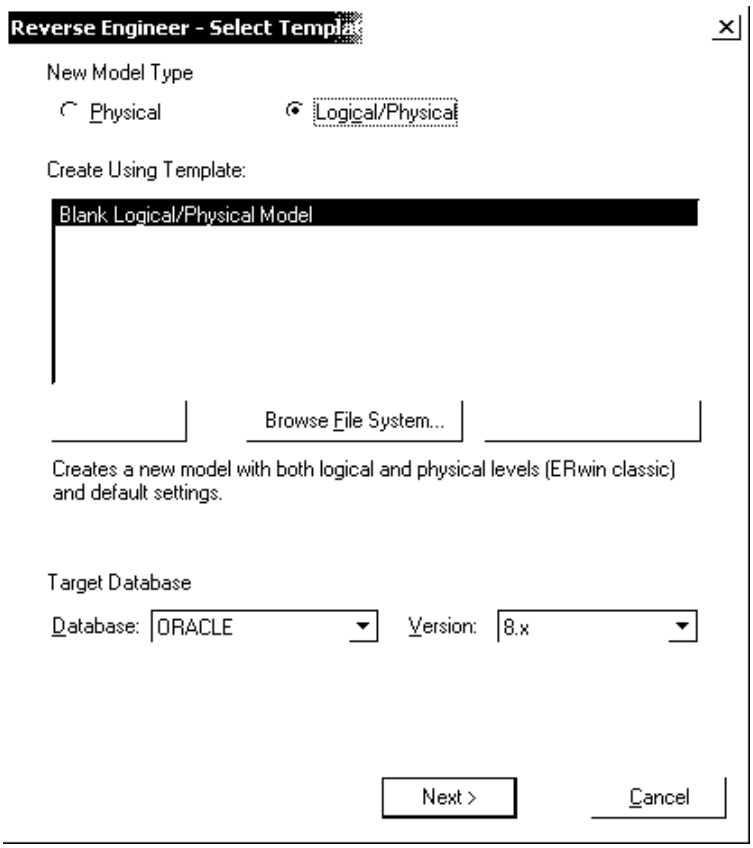

Рис. 2. Диалоговое окно Select Template (Reverse Engineer)

При обратном проектировании необходимо, также, указать источник (БД или исходный файл SQL-скрипта), необходимые объекты (таблицы, представления, хранимые

процедуры и т.п.), имя пользователя и другие параметры, зависящие от СУБД (см. рис. 3).

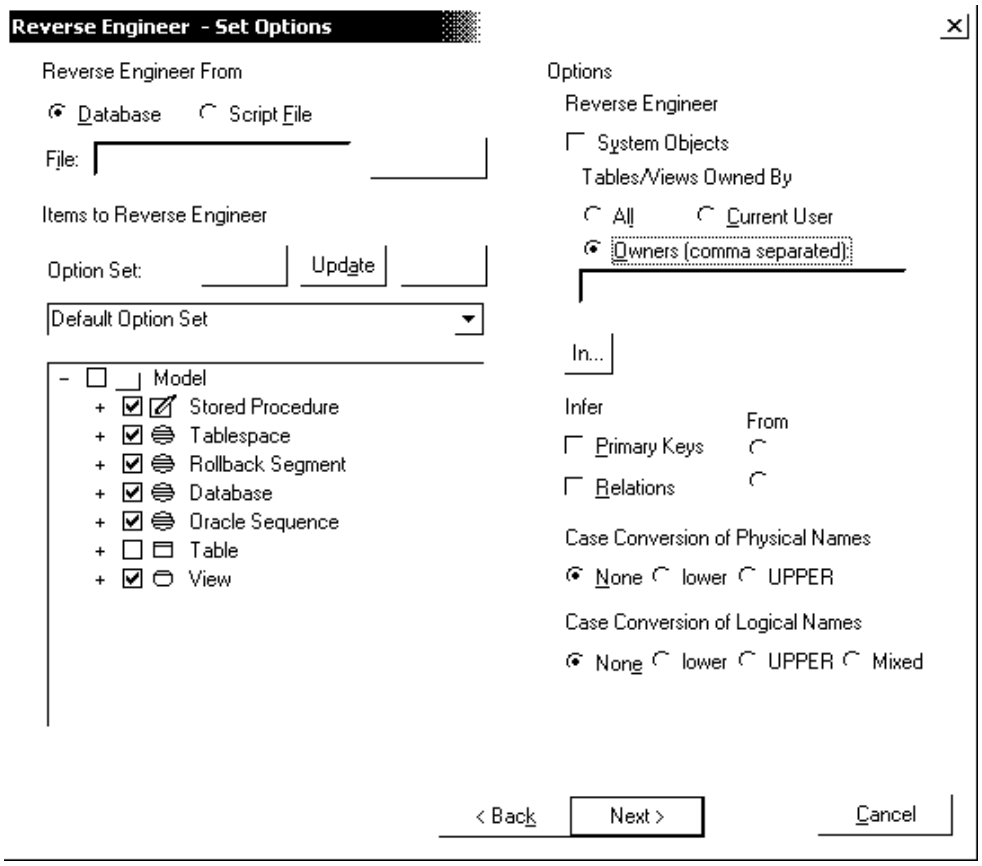

Рис. 3. Диалоговое окно Set Options

Для новой модели данных нужно сначала заполнить ее свойства в соответствующем диалоговом окне (Model Properties) (см. рис. 4). Для этого выберите пункт меню Model/Model Properties … и заполните поля следующих вкладок:

- General (название модели данных и ее автора),
- Definition (описание модели данных),
- Notation (нотация для логической и/или физической модели данных).

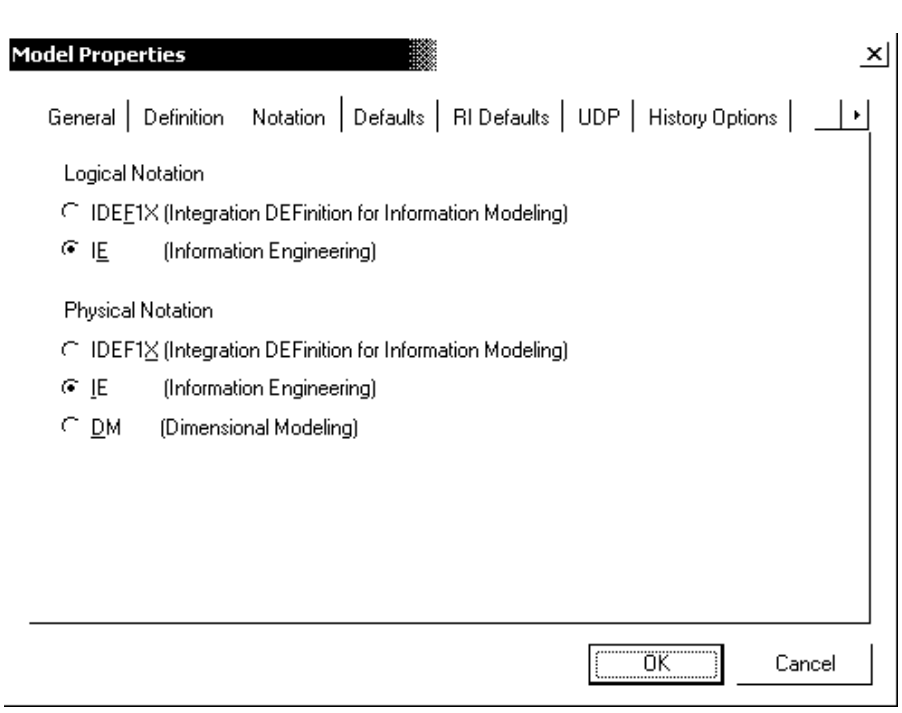

Рис. 4. Диалоговое окно Model Properties

Описывать модель данных можно в трех международно-признанных системах обозначений (нотациях):

1. Inegration DEFinition for Information Modeling (IDEF1X).
2. Information Engenering (IE).
3. Dimensional Modeling (DM).

Нотация IDEF1X была разработана для армии США и является федеральным стандартом США. Кроме того, она является стандартом в ряде международных организаций (НАТО, Международный валютный фонд и др.).

Нотация IE, разработанная Мартином, Финкельштейном и другими авторами, используется преимущественно в промышленности.

Специализированная нотация DM предназначена для разработки хранилищ данных и используется только на уровне физической модели данных.

При разработке модели данных удобно использовать панели инструментов, которые можно скрывать и показывать при помощи пункта меню View/Toolbars. ERwin имеет 7 перемещаемых панелей инструментов:

- стандартная панель (Standart),
- основная панель (Toolbox),
- панель шрифтов и цвета (Font&Color),
- панель работы с хранилищем моделей (ModelMart),
- панель мастеров изменения сущностей (таблиц) и отношений (связей) между ними (Transform),
- панель рисования (Drawing),
- панель выравнивания (Alignment).

В зависимости от уровня модели и выбранной нотации палитра панелей инструментов будет выглядеть по-разному. Для создания логической и физической модели данных в дальнейшем будет рассматриваться нотация IE.

2. Создание логической модели данных

Логическая модель данных является визуальным представлением структур данных, их атрибутов и правил. Логическая модель представляет данные таким образом, чтобы они легко воспринимались конечными пользователями или экспертами предметной области. Проектирование логической модели должно быть свободно от требований платформы и языка реализации или способа дальнейшего использования данных.

Для переключения между логической и физической моделью данных служит список выбора на стандартной панели инструментов (см. рис. 5).

Разработчик модели использует требования к данным и результаты анализа для формирования логической модели данных. Разработчик приводит логическую модель к третьей нормальной форме и проверяет ее на соответствие корпоративной модели данных, если она существует. Логическая модель использует сущности, атрибуты и связи между ними для представления данных и правил.

Сущности это объекты, о которых необходимо хранить данные. Сущностями могут быть вещественные объекты, такие как подразделение, сотрудник, но они могут представлять и абстрактные концепции, такие как центр затрат или производственная единица. Сущности для ясности и обеспечения целостности обозначаются существительными в единственном числе, например, Потребитель (CUSTOMER), а не Потребители (CUSTOMERS).

Сущность должна быть описана, используя фактографические подробности, которые уникально ее идентифицируют. Каждый экземпляр сущности должен быть отдельным и отличным от всех других экземпляров этой сущности. Например, модель данных для хранения информации о клиентах должна обеспечивать способ, позволяющий отличить одного клиента от другого.

Для создания новой сущности необходимо на основной панели инструментов (см. рис. 5) нажать кнопку Entity (прямоугольник) и указать ее месторасположение на рабочем листе модели.

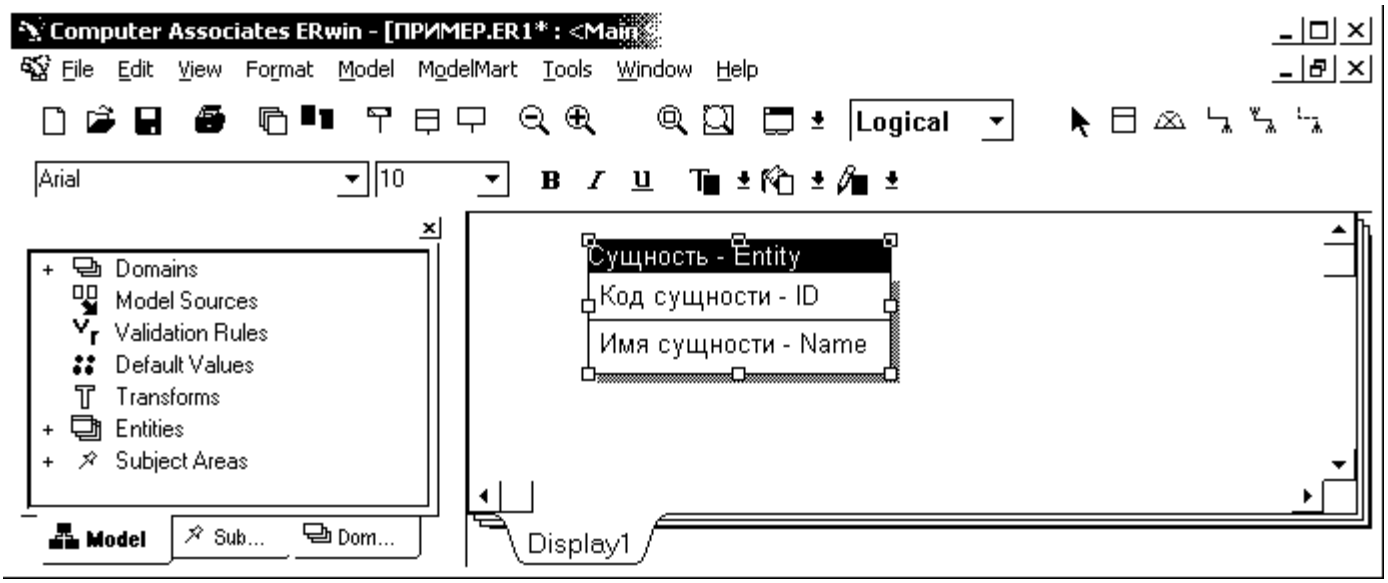

Рис. 5. Создание новых сущностей и атрибутов

У каждой сущности нужно заполнить свойства в соответствующем диалоговом окне (Entities) (см. рис. 6).

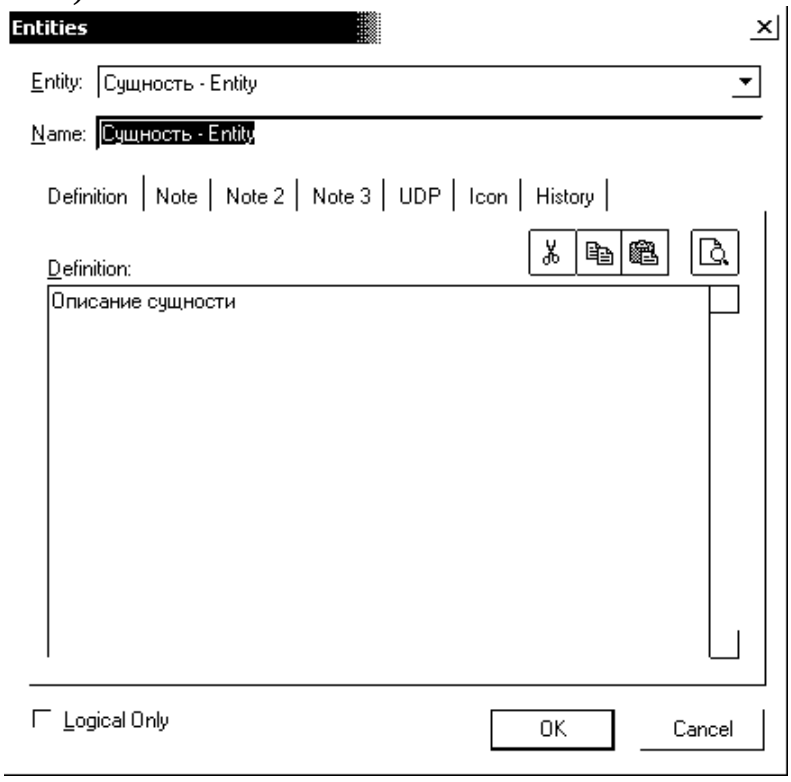

Рис. 6. Диалоговое окно Entities

Для этого выберите пункт меню Model/Entities …, укажите имя сущности (Name) и заполните поля следующих вкладок:

- Definition (описание сущности),
- Note (дополнительные замечания о сущности, например, описание правила),
- Note 2 (возможные запросы по отношению к сущности),
- Note 3 (примеры данных для сущности в произвольной форме),
- UDP (свойства сущности, определяемые пользователем),
- Icon (маленькое и большое графическое изображение сущности).

У каждой сущности должны быть заданы атрибуты. Атрибуты задают, какие именно данные необходимо хранить об объектах. Атрибуты представляются именами существительными, которые описывают характеристики сущностей. Для добавления атрибутов нужно выбрать нужную сущность и в меню выбрать пункт Model/Attributes … (рис. 7). В этом диалоговом окне (Attributes) можно создать новый атрибут (New …), изменить имя существующего атрибута (Rename …), удалить ненужный атрибут (Delete). Так же, у каждого атрибута должны быть заполнены нужные поля следующих вкладок:

- General (домен атрибута, признак первичного ключа и т.д.),
- Datatype (тип данных, ограничения на значения атрибута, значение по умолчанию),
- Definition (описание атрибута),
- Note (дополнительные замечания об атрибуте, например, описание правила),
- UDP (свойства атрибута, определяемые пользователем),
- Key Group (включение атрибута в состав первичного, альтернативного или инвертированного ключа).

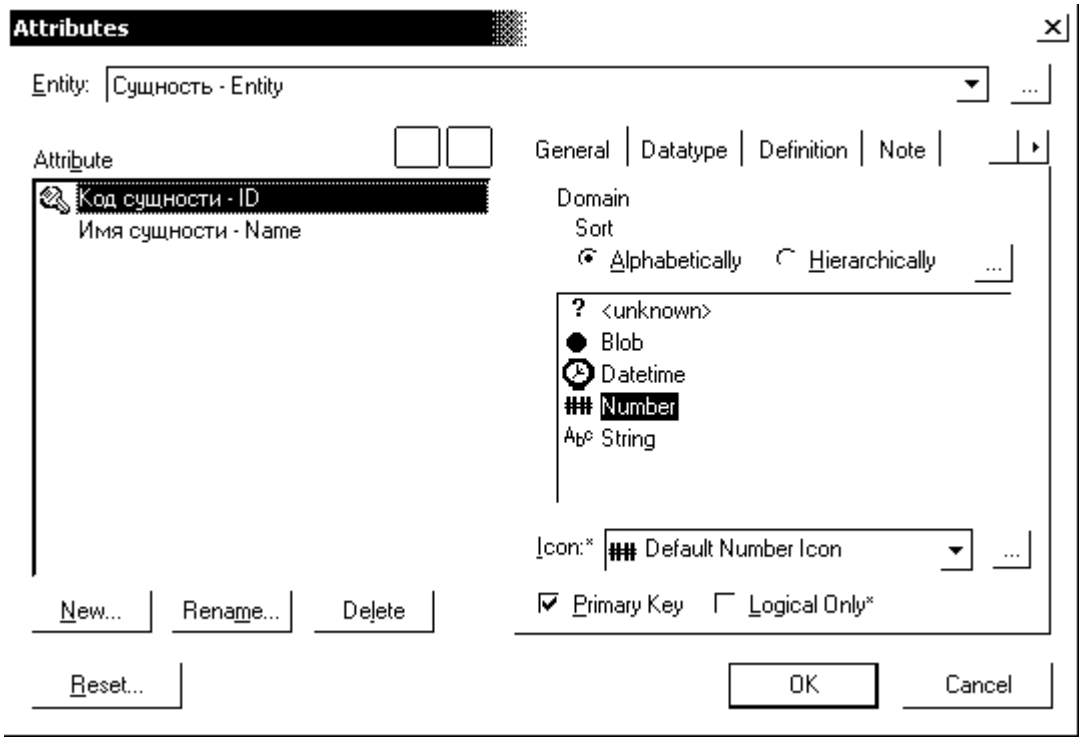

Рис. 7. Диалоговое окно Attributes

Связи описывают отношения между сущностями в терминах правил. Связи выражаются глаголами или глагольными фразами, которые описывают эти отношения. После того, как в модели данных созданы необходимые сущности и заданы их атрибуты, можно создать связи между сущностями. Между сущностями могут существовать следующие типы связей:

- категориальная связь (Exclusive sub-category),
- идентифицирующая связь один ко многим (Identifying relationship),
- связь многие ко многим (Many-to-Many relationship),
- неидентифицирующая связь один ко многим (Non-identifying relationship).

Для создания новой связи необходимо на основной панели инструментов (см. рис. 5) выбрать тип связи (Exclusive sub-category, Identifying relationship, Many-to-Many relationship, Non-identifying relationship). Затем указать (выбрать) сначала главную сущность (или подкатегорию) а потом зависимую сущность.

У каждой связи нужно заполнить свойства в соответствующем диалоговом окне (Relationships) (см. рис. 8). Для этого выберите нужную связь и в меню пункт Model/ Relationships …, после чего можно заполнить поля следующих вкладок:

- General (имя связи главный-подчиненный и подчиненный-главный, мощность и тип),
- Definition (описание связи),
- Rolename (имя атрибута внешнего ключа, мигрирующего из главной сущности),
- RI Actions (правила ссылочной целостности данных - referentail integrity),
- UDP (свойства связи, определяемые пользователем).

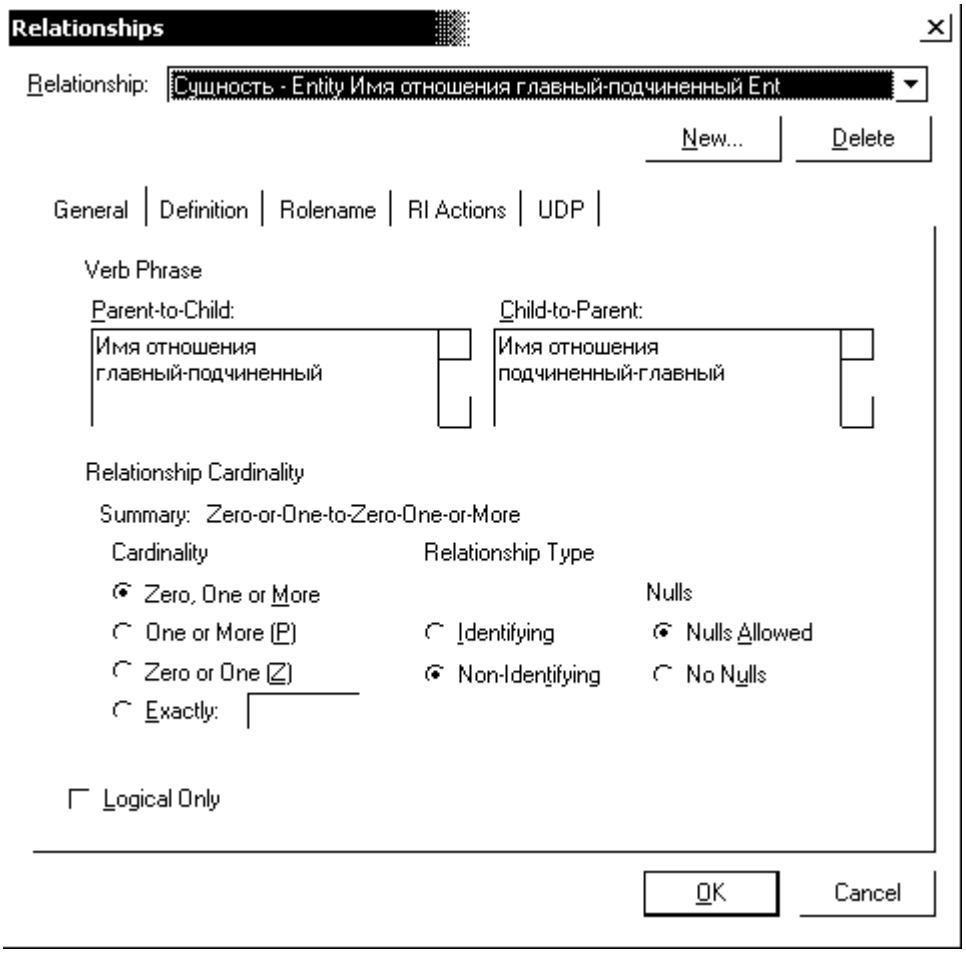

Рис. 8. Диалоговое окно Relationships

3. Пример проектирования БД

В качестве примера рассмотрим создание логической модели для деканата в СУБД Access. Описание предметной области: в БД необходимо хранить информацию о студенческих группах, студентах, обучающихся в этих группах, и оценках по дисциплинам, полученными студентами в результате сдачи сессии. О студенческих группах должна храниться информация: номер группы и курс обучения. О студентах должна храниться следующая информация: фамилия, имя, отчество студента и его дата рождения. Информация о результатах сдачи сессии студентами должна содержать данные об оценке каждого студента по различным дисциплинам.

В повседневной практике, логическая модель нормализуется до третьей нормальной формы. Нормализация является операцией перемещения атрибутов в подходящие сущности в соответствие с требованиями нормальных форм. Нормализацию обычно представляют как набор сложных правил, из-за чего эта концепция кажется трудной для понимания. В действительности, нормализация достаточно очевидна: «Один факт в одном месте» [1].

Нормализация данных означает проектирование структур данных таким образом, чтобы удалить избыточность и ограничить несвязанные структуры. Широко используются пять нормальных форм. Эти формы называются просто: первая нормальная, вторая нормальная, третья нормальная, четвертая нормальная и пятая нормальная формы. На практике многие логические модели приведены только к третьей нормальной форме. Ниже приведены простые правила нормализации:

1. Размещайте повторяющиеся атрибуты в зависимых сущностях.

2. Убедитесь, что каждый факт в модели представлен только один раз.
3. Размещайте атрибуты, независящие от первичного ключа, в зависимых сущностях.
4. Устраняйте связи многие ко многим.

Сначала создаем модель, для этого выбираем пункт главного меню File/New … и указываем, что необходимо создать логическую и физическую модель (Logical/Physical), а в качестве целевой СУБД (Target Database) выбираем Access 2000.

У созданной модели данных заполняем соответствующие свойства (пункт меню Model/Model Properties …):

- General (Name: «БД деканата», Author: «Сидоров Иван Иванович»),
- Definition (Definition: «БД хранит информацию о результатах сдачи сессии студентами»),
- Notation (Logical Notation: «IE», Physical Notation: «IE»).

Далее создаем сущность «Студент», для этого нажимаем левой кнопкой мыши кнопку Entity на панели инструментов Toolbox, указываем месторасположение новой сущности и ее название. Затем заполняем соответствующие свойства созданной сущности «Студент» (выбираем сущность и пункт меню Model/Entities …):

- Definition (Definition: «Информация о студентах факультета»),
- Note (Note: «Поля: фамилия, имя и дата рождения студента являются обязательными»),
- Note 2 (Note 2: «Выбрать всех студентов по указанной студенческой группе, не сдавших сессию, сдавших сессию на 4 и 5 и т.д.»),
- Note 3 (Note 3: «Сидоров Иван Иванович - 19/11/1970 и т.д.»).

Затем указываем атрибуты созданной сущности «Студент». Для этого необходимо открыть диалоговое окно Attributes (выбираем сущность и пункт меню Model/Attributes …). В диалоговом окне Attributes создаем необходимые атрибуты сущности «Студент» и заполняем необходимые свойства каждого атрибута. В качестве суррогатного первичного ключа создаем атрибут «Код студента», т.к. другие атрибуты могут либо изменятся в процессе эксплуатации БД, либо однозначно не идентифицировать каждый экземпляр сущности.

Аналогично создаем остальные сущности «Группа» и «Оценка». Для каждой созданной сущности создаем атрибуты, соответствующие описанию предметной области (рис. 9). Для сущности «Группа» создаем атрибут «Код группы» в качестве первичного ключа. Предполагаем, что атрибуты «ФИО» и «Дисциплина» являются первичным ключом сущности «Оценка».

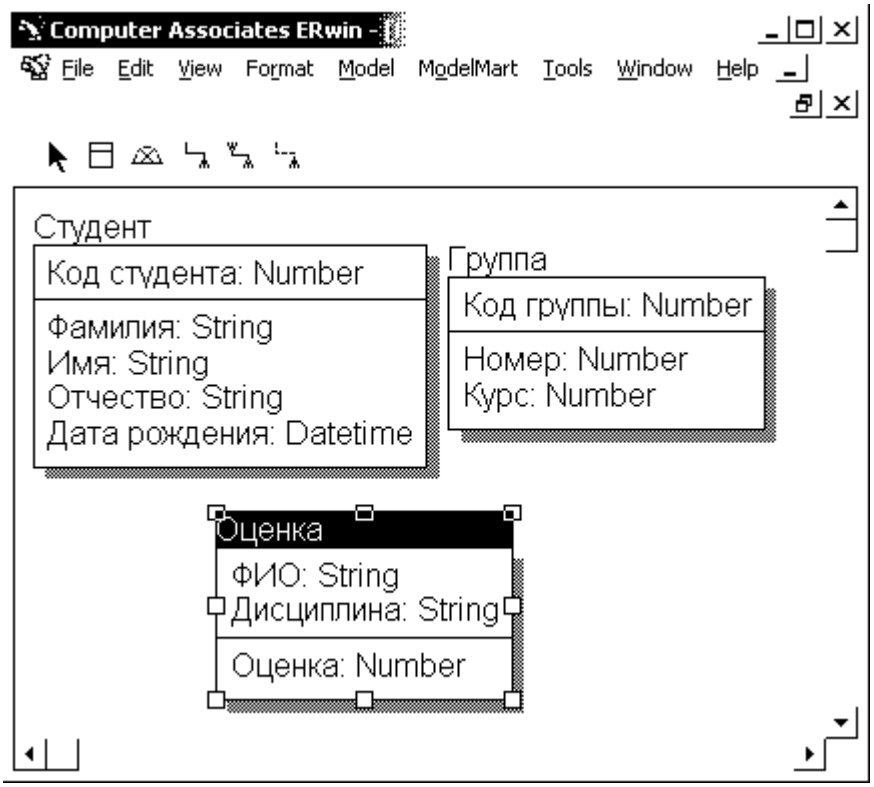

Рис. 9. Пример создания сущностей и атрибутов

После того, как созданы необходимые сущности и заданы их атрибуты, можно создать связи между этими сущностями (рис. 10).

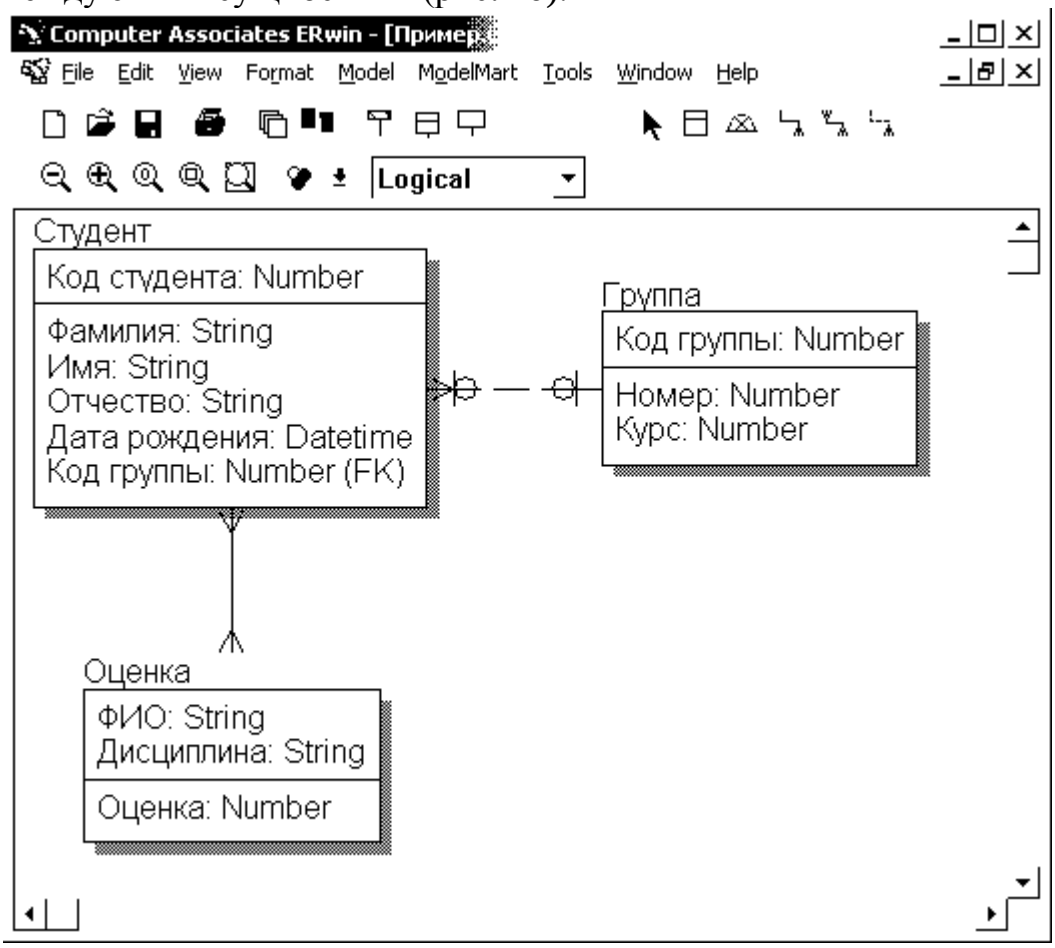

Рис. 10. Пример создания связей между сущностями

Так между сущностями «Группа» и «Студент» необходимо создать неидентифицирующую связь один ко многим, поскольку в каждой группе обучаются несколько студен-

тов. При этом из главной сущности «Группа» в зависимую сущность «Студент» в качестве внешнего ключа мигрирует атрибут первичного ключа «Код группы». Между сущностями «Студент» и «Оценка» нужно создать связь многие ко многим, т.к. студент за сессию получает несколько оценок по разным дисциплинам.

У каждой связи нужно заполнить соответствующие свойства в диалоговом окне Relationships (см. рис. 8). Из схемы данных (рис. 10) видно, что атрибуты «ФИО» и «Дисциплина» будут повторяться для различных экземпляров сущности «Оценка», т.к. один и тот же студент может иметь несколько оценок по различным дисциплинам, и по одной дисциплине могут иметь оценки различные студенты. Таким образом, повторяющиеся атрибуты необходимо разместить в зависимых сущностях (при этом атрибут «ФИО» является лишним, т.к. между сущностью «Студент» и сущностью «Оценка» уже имеется связь). Для этого создаем новую сущность «Дисциплина», перетаскиваем в нее атрибут «Дисциплина» и переименовываем его в «Название». Дополнительно создаем в сущности «Дисциплина» атрибут первичного ключа «Код дисциплины». Между сущностью «Дисциплина» и сущностью «Оценка» создаем идентифицирующую связь один к одному. Заменяем связь многие ко многим между сущностями «Студент» и «Оценка» на идентифицирующую связь один к одному, т.к. теперь каждая оценка однозначно идентифицируется кодом студента и кодом дисциплины (рис. 11).

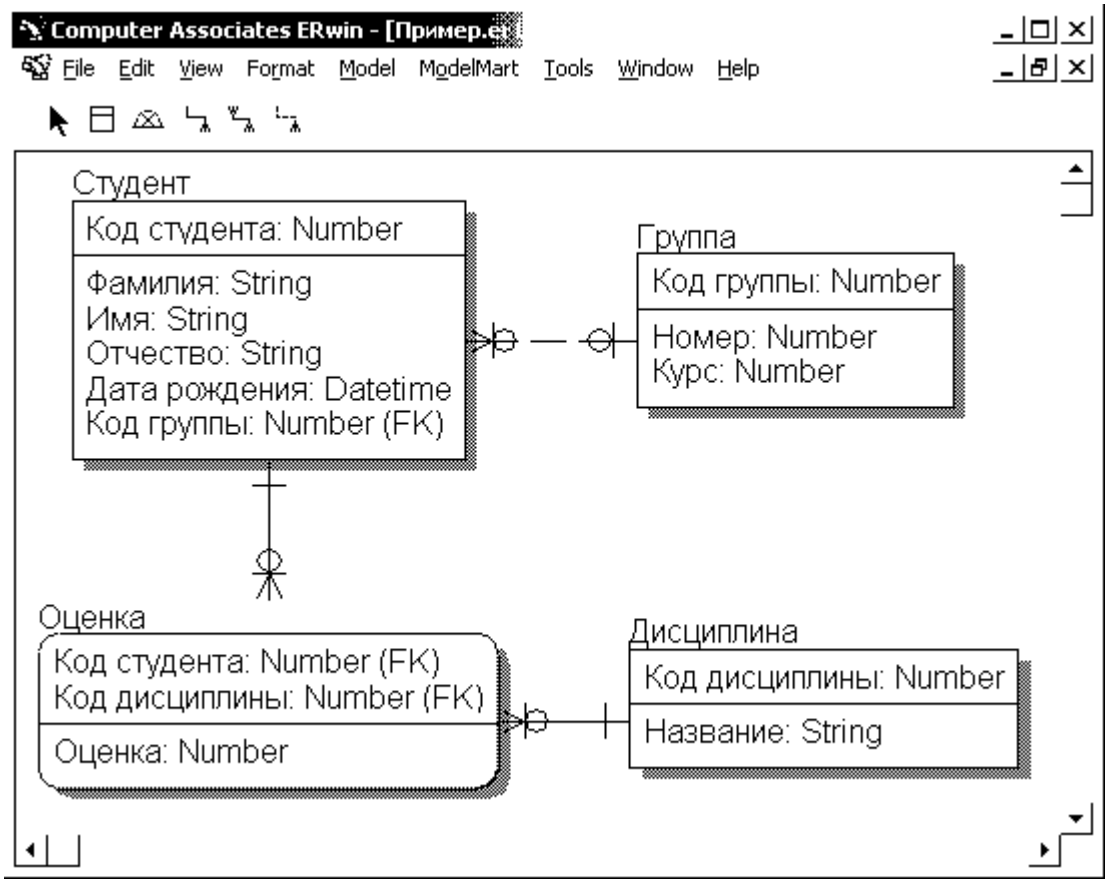

Рис. 11. Пример нормализации данных

У созданных связей нужно заполнить соответствующие свойства. В данном случае рекомендуется заполнить свойство Definition, чтобы впоследствии значение каждой связи и ее особенности были понятны как разработчику приложения, так и другим специалистам, работающим с созданной логической моделью данных.

4. Области и хранимые отображения модели

Для работы с моделью данных удобно использовать окно Model Explorer, которое можно скрыть или показать, выбрав пункт главного меню View/Model Explorer …. Это окно имеет три вкладки:

1. Model - показывает все объекты модели,
2. Subject Areas - показывает области модели,
3. Domains - показывает домены атрибутов модели.

Область модели (Subject Aria) - именованная версия модели данных, которая может включать в себя все сущности, связи, подтипы и текстовые блоки или любое подмножество объектов из полной модели данных. Например, можно создать область модели, в которой будут только те сущности и связи между ними, которые используются конкретной подсистемой.

По умолчанию исходная область модель данных получает название Main Subject Aria. Для создания новой области модели или внесения изменений в существующую область можно выбрать пункт главного меню Model/Subject Arias …. В этом диалоговом окне (см. рис. 12) можно создать новую область модели (New …), переименовать (Rename …) или удалит (Delete) существующую область. В выбранную область модели можно добавить (или убрать) объекты (Included Objects) из доступных объектов (Available Objects). В каждой области модели можно заполнить нужные поля следующих вкладок:

- General (автор области модели),
- Definition (описание области модели),
- UDP(свойства области модели, определяемые пользователем).

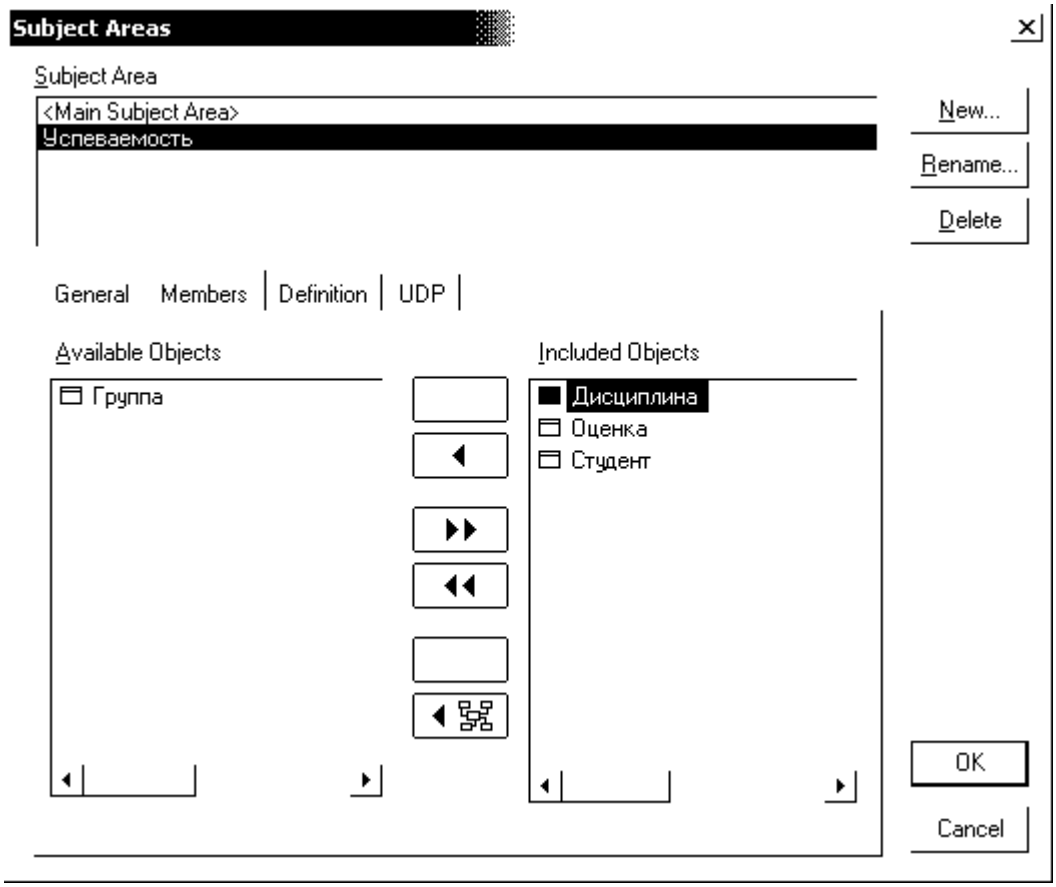

Рис. 12. Диалоговое окно Subject Areas

Области модели могут оказаться особенно полезными при проектировании и управлении большой и сложной моделью данных. Разбиение главной области на несколько меньших областей позволяет различным группам, входящим в организацию, сконцентрироваться на процессах и задачах, относящихся к их предметной области.

Хранимое изображение (Stored Displays) - альтернативное представление области модели, освещающее какой-то аспект всей структуры данных. Хранимое изображение включает в себя все объекты, содержащиеся в родительской области, но объекты могут располагаться иначе и диаграмма может быть установлена на другой уровень демонстрации изображения.

Для создания хранимого изображения необходимо выбрать пункт меню Format/Stored Display Settings …(рис. 13), выбрать пункт New и ввести новое имя. В этом же диалоговом окне можно внести изменения в существующие хранимые изображения.

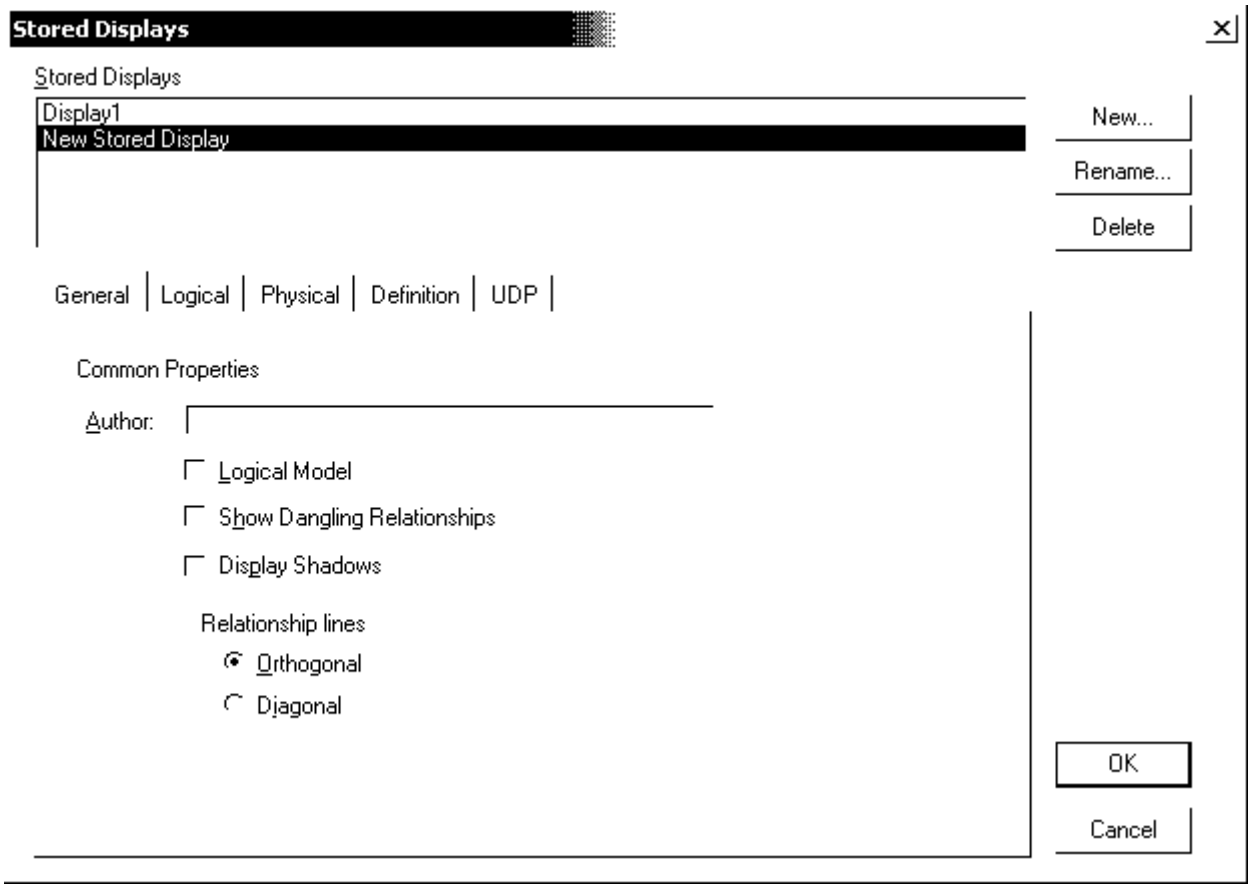

Рис. 13. Диалоговое окно Stored Displays

Используя хранимые изображения, можно создать различные представления области модели. Использование хранимого изображения помогает лучше определить задачи или функции, связанные с конкретным процессом. Например, можно переносить объекты, связанные с процессом, на более заметное место на диаграмме (рис. 14). Изменив расположение объектов, можно лучше сфокусироваться на конкретном участке главной области.

Хранимое изображение можно использовать для презентации. При обращении к конкретной аудитории можно построить диаграмму - хранимое изображение для того, чтобы подчеркнуть те объекты, которые связаны с темой презентации. Можно показать диаграмму на нескольких различных уровнях демонстрации изображения, например, ес-

ли главная область показана на уровне атрибутов, то можно создать хранимое изображение для показа этой же модели на уровне сущностей и т.д.

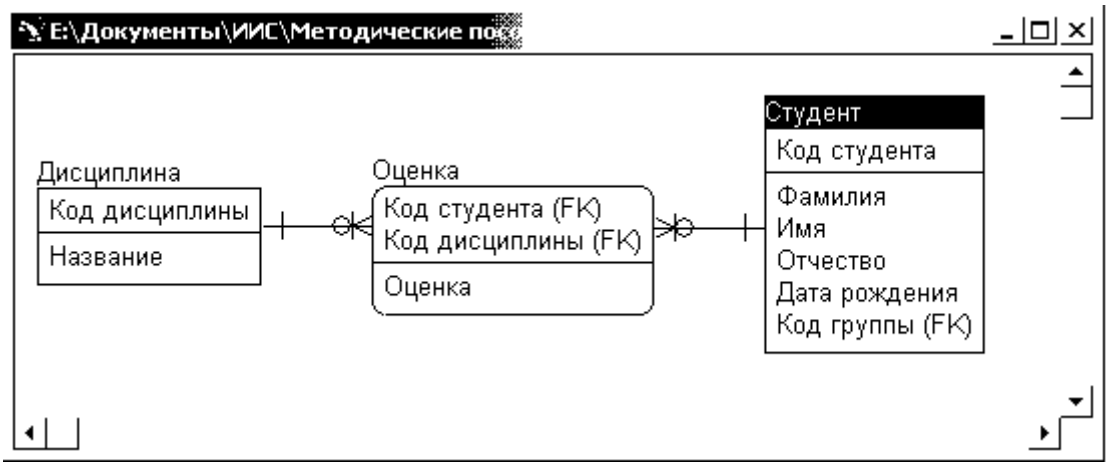

Рис. 14. Хранимое изображение модели

Для перехода от одного хранимого изображения к другому, необходимо в нижней части окна диаграммы выбрать соответствующую закладку хранимого изображения. По умолчанию в ERwin принято, что закладки хранимых изображений демонстрируются на экране. Можно их спрятать с помощью переключателя главного меню View/Stored Display Tabs.

5. Создание физической модели данных

Физическая модель данных зависит от выбранной СУБД. Для выбора СУБД необходимо на уровне физической модели данных выбрать пункт меню Database/Choose Database … (рис. 15). Erwin поддерживает более 20 реляционных и нереляционных СУБД.

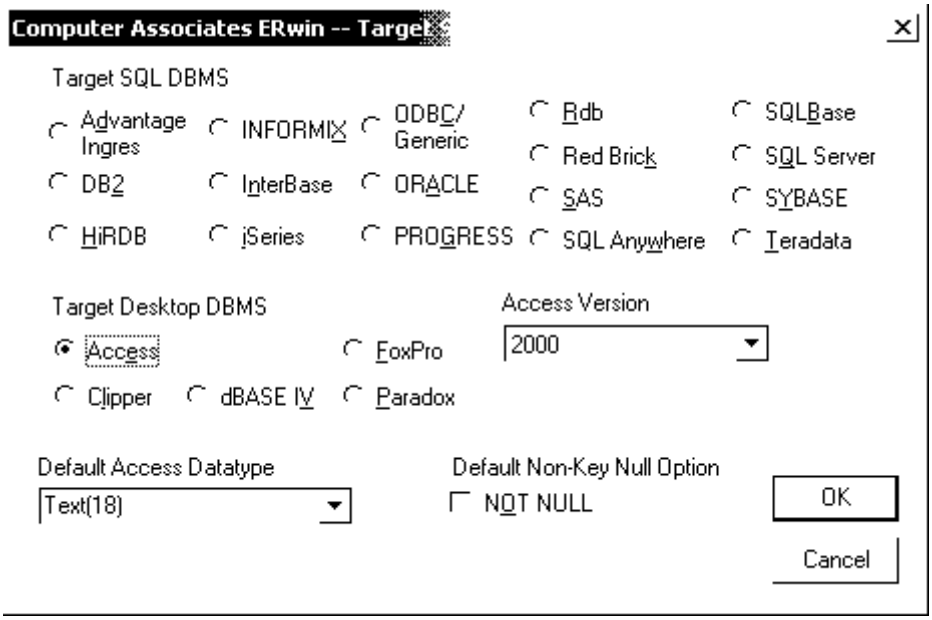

Рис. 15. Выбор СУБД для физической модели данных

При смене СУБД Erwin позволяет автоматически преобразовать тип данных полей таблиц. Добавить таблицы, поля или создать связи можно так же, как и на логическом уровне. Имена таблиц и полей в физической модели создаются на основе имен соответ-

ствующих сущностей и атрибутов, учитывая максимальную длину и другие синтаксические ограничения, выбранной СУБД.

На уровне физической модели данных можно задать дополнительные параметры, зависящие от выбранной СУБД, например, параметры таблиц и табличных пространств, сегменты отката, триггеры и хранимые процедуры, ограничения целостности и значения по умолчанию, индексы и т.д.

6. Генерация схемы данных

После того, как разработана полная и непротиворечивая физическая модель данных, можно синхронизировать с БД или сгенерировать соответствующие команды создания новой БД. Для этого необходимо выбрать пункт меню Tools/Forward Engineer/Schema Generation … (рис. 16).

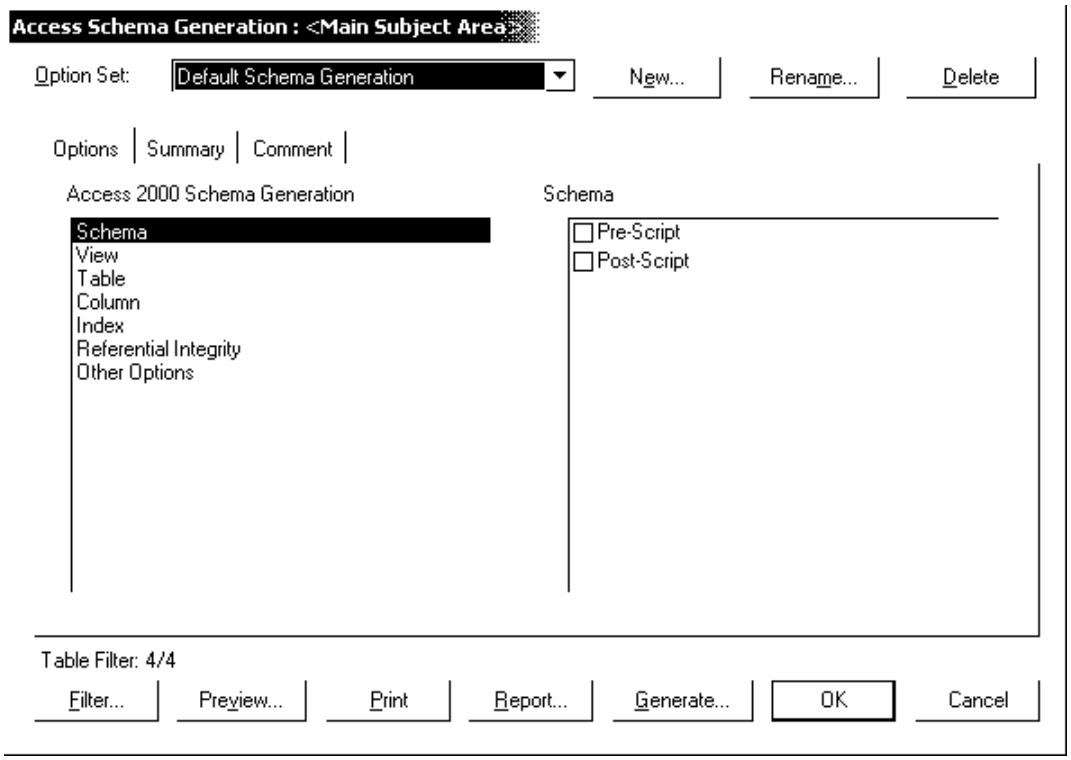

Рис. 16. Генерация БД

Для просмотра соответствующих команд создания БД необходимо нажать кнопку Preview ….

7. Лабораторные работы

Задание: проанализировать предметную область, создать модель и базу данных будущей информационной системы.

Лабораторная работа № 1

Предметная область: Библиотека (учет читателей).

Основные предметно-значимые сущности: Книги, Читатели, Подразделения.

Основные предметно-значимые атрибуты сущностей:

- книги – автор книги, название, год издания, цена, является ли новым изданием, краткая аннотация;

- подразделения – название;
- читатели – номер читательского билета, ФИО, место работы, адрес и телефон читателя.

Основные требования к функциям системы:

- выбрать и упорядочить книги по дате выдачи, которые находятся у читателей или определенного читателя;
- выбрать читателей, которые брали ту или иную книгу с указанием даты выдачи книги и даты сдачи книги читателем;
- выбрать подразделения, читатели которого наиболее часто пользуются услугами библиотеки;
- выбрать книги, пользующиеся наибольшим спросом.

Лабораторная работа № 2

Предметная область: Деканат (успеваемость студентов).

Основные предметно-значимые сущности: Студенты, Группы студентов, Дисциплины.

Основные предметно-значимые атрибуты сущностей:

- студенты – фамилия, имя, отчество, пол, дата рождения, адрес прописки, группа студентов, состояние (учится, отчислен, находится в академическом отпуске);
- группы студентов – название, курс, семестр;
- дисциплины – название.

Основные требования к функциям системы:

- выбрать и упорядочить студентов по успеваемости, которые учатся в студенческих группах или определенной студенческой группе;
- выбрать студентов, не имеющих оценок, по дисциплинам или определенной дисциплине;
- выбрать дисциплины, изучаемые группой студентов на определенном курсе или определенном семестре.

Лабораторная работа № 3

Предметная область: Поликлиника (учет пациентов).

Основные предметно-значимые сущности: Пациенты, Врачи, Кабинеты.

Основные предметно-значимые атрибуты сущностей:

- пациенты – фамилия, имя, отчество, дата рождения;
- врачи – фамилия, имя, отчество, дата рождения, должность, специализация;
- кабинеты – номер.

Основные требования к функциям системы:

- выбрать всех пациентов по диагнозам или определенному диагнозу;
- выбрать всех пациентов записанных к определенному врачу на определенную дату;
- выбрать всех врачей, которых посещал определенный пациент в определенный период времени.

Лабораторная работа № 4

Предметная область: Телефонный узел связи (учет абонентов).

Основные предметно-значимые сущности: Абоненты, Подразделения, Помещения.

Основные предметно-значимые атрибуты сущностей:

- абоненты – фамилия, имя, отчество, дата рождения, подразделение;
- подразделения – название, вид подразделения;

- помещения – название или номер помещения, вид помещения (аудитория, кабинет и т.п.), подразделение.

Основные требования к функциям системы:

- выбрать номера абонентов по подразделениям и помещениям;
- выбрать и упорядочить подразделения по стоимости междугородних разговоров за определенный период;
- подсчитать количество абонентов по подразделениям, помещениям.

Лабораторная работа № 5

Предметная область: Склад (учет материалов).

Основные предметно-значимые сущности: Материал, Местоположения, Материально ответственные лица.

Основные предметно-значимые атрибуты сущностей:

- материал – название, стоимость, остаточная стоимость;
- местоположения – название склада;
- материально-ответственные лица – фамилия, имя, отчество, паспортные данные.

Основные требования к функциям системы:

- выбрать все остатки материалов и их суммарную стоимость по материально-ответственным лицам или определенному лицу;
- выбрать все остатки материалов и их суммарную стоимость по местоположению и материально-ответственным лицам;
- выбрать все остатки материалов и их суммарную стоимость по материалам и местоположению на определенную дату.

Заключение

Моделирование играет большую роль в разработке успешных АСОИУ. Использование CASE-средств поможет правильно оценить стоящие задачи, предложить адекватное решение и разработать центральную часть любой АСОИУ - базы данных - с использованием информации, полученной во время обследования предприятия (моделирование базы данных, ERwin). Эти инструменты сами по себе не являются решением проблемы, но их грамотное и своевременное использование поможет свести рутинный труд разработчика к минимуму, позволит ему сконцентрироваться на собственно разработке системы и снизит потери времени, которые обычно происходят при согласовании моделей со специалистами предметной области. Кроме того, использование этих инструментов дает возможность получить набор полностью документированных и согласованных моделей, что в значительной степени облегчит поддержку созданных систем в будущем, а также может быть повторно использовано при разработке других систем.

Список литературы

1. Дейт К.Д. Введение в системы баз данных, 8-е издание. -М: Вильямс. 2005 г. - 1328 с.
2. Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite. –М: ДИАЛОГ-МИФИ, 2003 г. -432 с.