ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
«Нижегородский государственный университет им. Н.И.Лобачевского»

Факультет ВМК

Кафедра ИАНИ

Методология IDEF1X и программный продукт ERWin

(Учебно-методическое пособие)

г. Нижний Новгород, 2007

Введение

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для построения *концептуальной схемы* структуры данных предприятия, независимой от конечной реализации базы данных и аппаратной платформы.

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER и ENALIM является строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая является значительным недостатком ER.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира.

Программное средство **ERWin** позволяет проектировать, документировать и сопровождать базы данных, хранилища данных. Визуальное моделирование повышает качество создаваемой базы данных, продуктивность и скорость её разработки.

Основные особенности IDEF1X/ERWin:

1. Поддерживается прямое (создание БД на основе модели) и обратное (генерация модели по имеющейся базе данных) проектирование для 20 типов СУБД.
2. Увеличивает производительность труда благодаря удобному интерфейсу и автоматизации рутинных процедур.
3. Поддерживает методологию структурного моделирования SADT и следующие нотации: IDEF1X.
4. Поддерживает 20 различных СУБД: настольные, реляционные и специализированные СУБД, предназначенные для создания хранилищ данных.
5. Позволяет повторно использовать компоненты созданных ранее моделей, а также использовать наработки других разработчиков.
6. Возможна совместная работа группы проектировщиков с одними и теми же моделями (с помощью AllFusion Model Manager 4.1).
7. Позволяет переносить структуру БД из одной СУБД в другую.
8. Позволяет документировать структуру БД.
9. Продукт можно использовать на всех стадиях жизненного цикла БД: проектировании, разработке, тестировании и поддержке.

ERWin - это не просто средство проектирования, но и инструмент разработки, способный автоматически создавать таблицы и генерировать текст хранимых процедур для всех популярных СУБД. Революционная технология Complete - Compare (Завершить-Сравнить) позволяет организовать итеративную разработку, поддерживая постоянную согласованность модели и базы данных. Благодаря интеграции с популярными средами разработки программ, ERWin позволяет ускорить создание приложений для обработки данных.

1. CASE-средство ERWin

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.

Для переключения между логической и физической моделью данных служит список выбора в левой части панели инструментов ERWin (рис. 1).

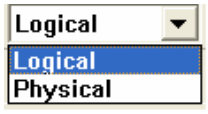


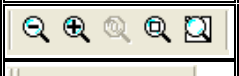
Рис. 1. Переключение между логической и физической моделью

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

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 табл 1.

Таблица 1.

Основная панель инструментов

Кнопки	Назначение кнопок
	Создание, открытие, сохранение и печать модели
	Вызов диалога Report Browser для генерации отчетов
	Изменение уровня просмотра модели: уровень сущностей, уровень атрибутов и уровень определений
	Изменение масштаба просмотра модели
	Генерация схемы БД, выравнивание схемы с моделью и выбор сервера (доступны только на уровне физической модели)

Для создания моделей данных в ERWin можно использовать две нотации: IDEF1X и IE (Information Engineering). Методология IDEF1X была разработана для армии США и широко используется в государственных учреждениях США, финансовых и промышленных корпорациях. Методология IE, разработанная Мартином (Martin), Финкельштейном (Finkelstein) и другими авторами, используется преимущественно в промышленности. Переключение между нотациями можно сделать в закладке Methodology диалога Preferences (меню Option/Preferences). В дальнейшем будет использоваться нотация IDEF1X.

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 кон-

текстном 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.

2. 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 детализирована, 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 м 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 (но не Заказчики!) с атрибутами Номер заказчика, Фамилия заказчика и Адрес заказчика. На уровне физической модели ей может соответствовать таблица Customer с колонками Customer_number, Customer_name и Customer_address.

Для внесения сущности в модель необходимо (убедившись предварительно, что вы находитесь на уровне логической модели - переключателем между логической и физической моделью служит раскрывающийся список в правой части панели инструментов) "кликнуть" по кнопке сущности на панели инструментов (ERWin Toolbox) , затем "кликнуть" по тому месту на диаграмме, где необходимо расположить новую сущность. Щелкнув правой кнопкой мыши по сущности и выбрав из всплывающего меню пункт Entity Editor, можно вызвать диалог Entity Editor, в котором определяются имя, описание и комментарии сущности (рис. 2).

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. В прежних версиях ERWin закладкам Note2 и Note3 соответствовали окна Query и Sample.

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.

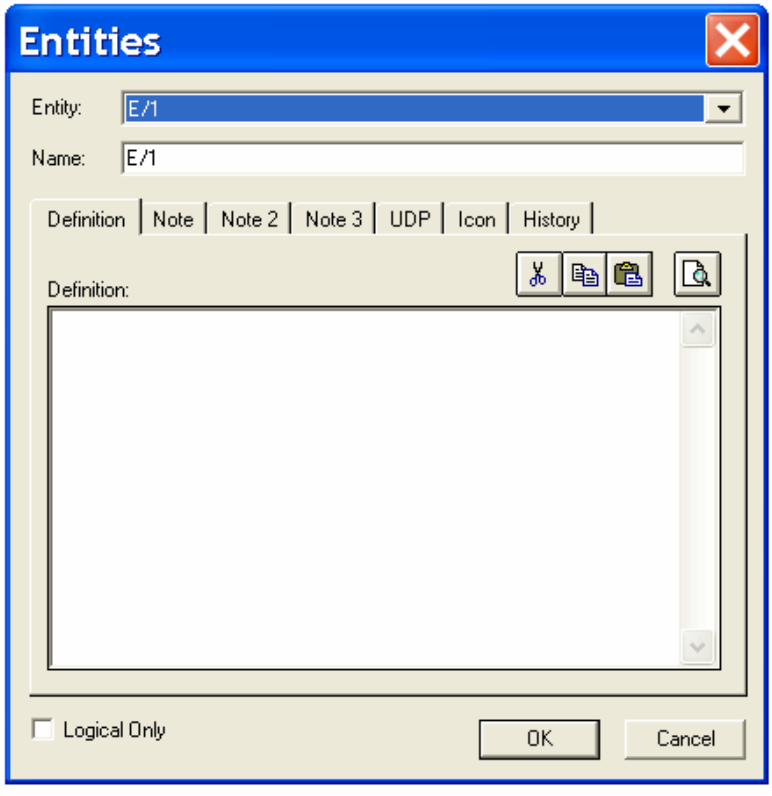


Рис. 2. Диалог Entity Editor

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. 3).

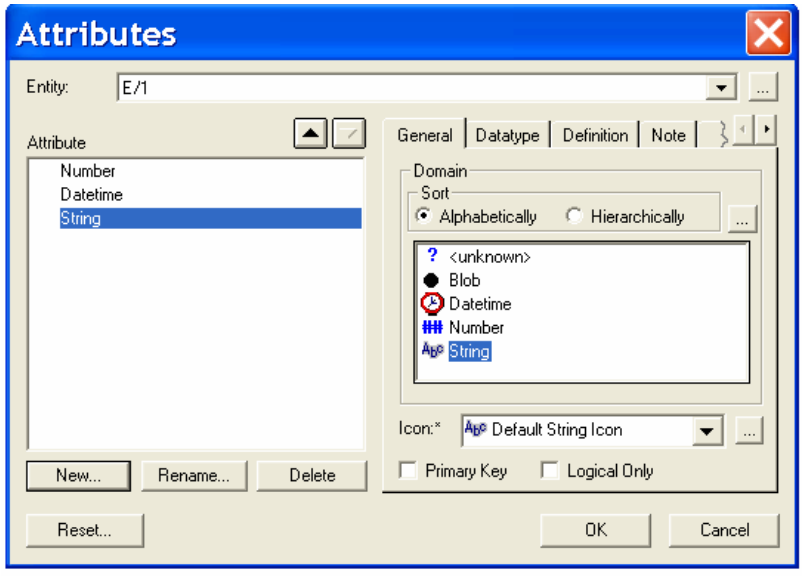


Рис. 3. Диалог Attribute Editor

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.

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 ***Комментарии/3*** **и т. д.**

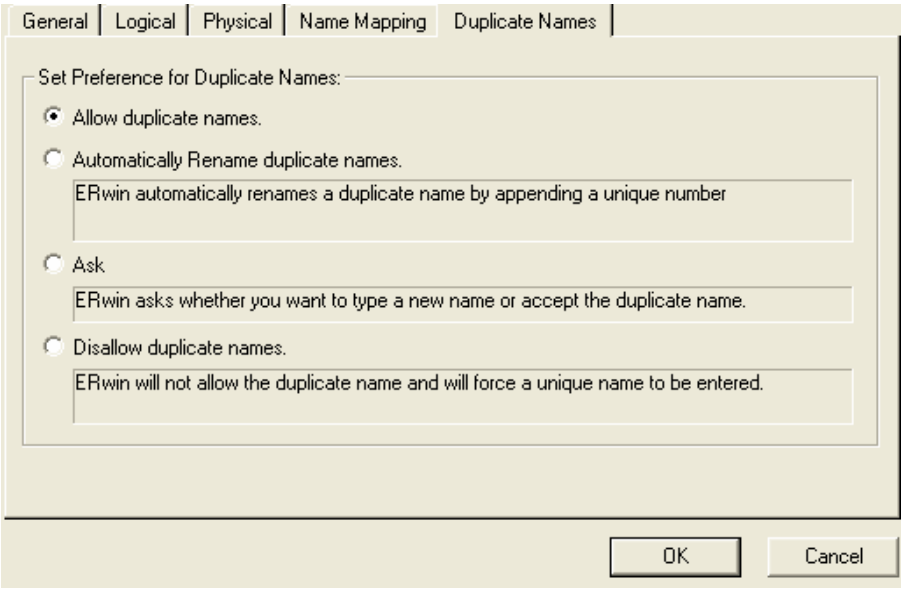


Рис. 4. Диалог Unique Name Option

На практике такое переименование не всегда удобно, поэтому существует возможность отменить эту опцию. В диалоге Unique Name Option (меню Option/Unique Name) (рис. 4) можно задать следующие режимы именования атрибутов:

- Allow - позволить внесение одинаковых имен;

- Rename - переименовывать атрибуты (по умолчанию);

- Ask - запрашивать возможные действия каждый раз при внесении одноименных атрибутов. ERWin будет показывать на экране окно-диалог Edit Unique Name каждый раз, когда вводится неуникальное имя сущности или атрибута. В диалоге Edit Unique Name можно ввести другое имя или разрешить дублирование. При этом новое имя не проверяется на уникальность;

- Disallow - запретить внесение одинаковых имен. Если двойное имя обнаружено, то ERWin выдает на экран окно с сообщением, что ввод неуникальных имен запрещается.

Каждый атрибут должен быть определен (закладка Definition), при этом следует избегать циклических определений, например? когда термин 1 определяется через термин 2, термин 2 - через термин 3, а термин 3 в свою очередь - через термин 1

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

- Каждый КЛИЕНТ <*размещает*> ЗАКАЗы;
- Каждый ЗАКАЗ <*выполняется*> СОТРУДНИКом.

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 рису-

ется идентифицирующая связь, ERWin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами (сущность ***Заказ*** на рис. 5). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т. е. в структуре на рис. 5 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 новые атрибуты помечаются как внешний ключ - (FK).

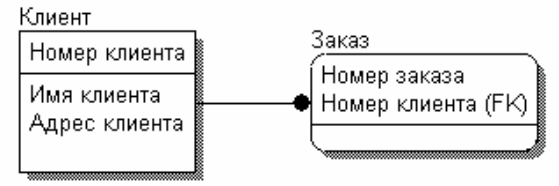


Рис. 5. Идентифицирующая связь между независимой и зависимой таблицей

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

При установлении неидентифицирующей связи (рис. 6) дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

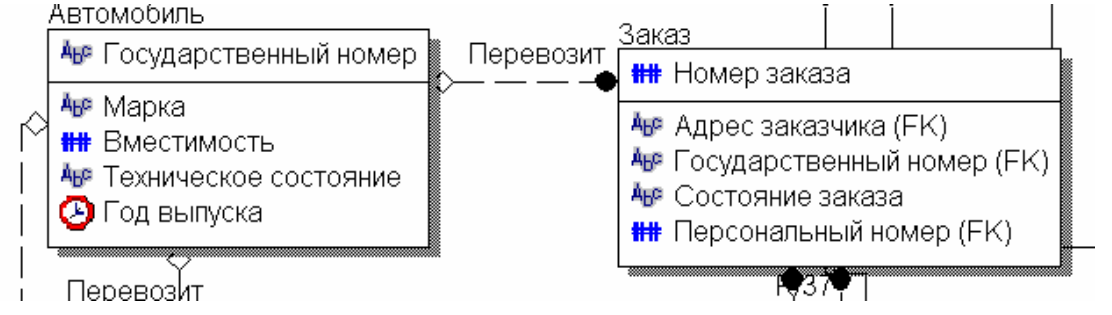


Рис. 6. Неидентифицирующая связь

Экземпляр сущности ***Заказ*** может существовать безотносительно к какому-либо экземпляру сущности ***Автомобиль,*** т. е. заказ может не перевозиться ни одним автомобилем.

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи (см. рис. 5), неидентифицирующая - пунктирной (рис. 6).

Для создания новой связи следует:

- установить курсор на нужной кнопке в палитре инструментов (идентифицирующая или неидентифицирующая связь) и нажать левую кнопку мыши ;
- щелкнуть сначала по родительской, а затем по дочерней сущности.

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. 7).

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 P 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.

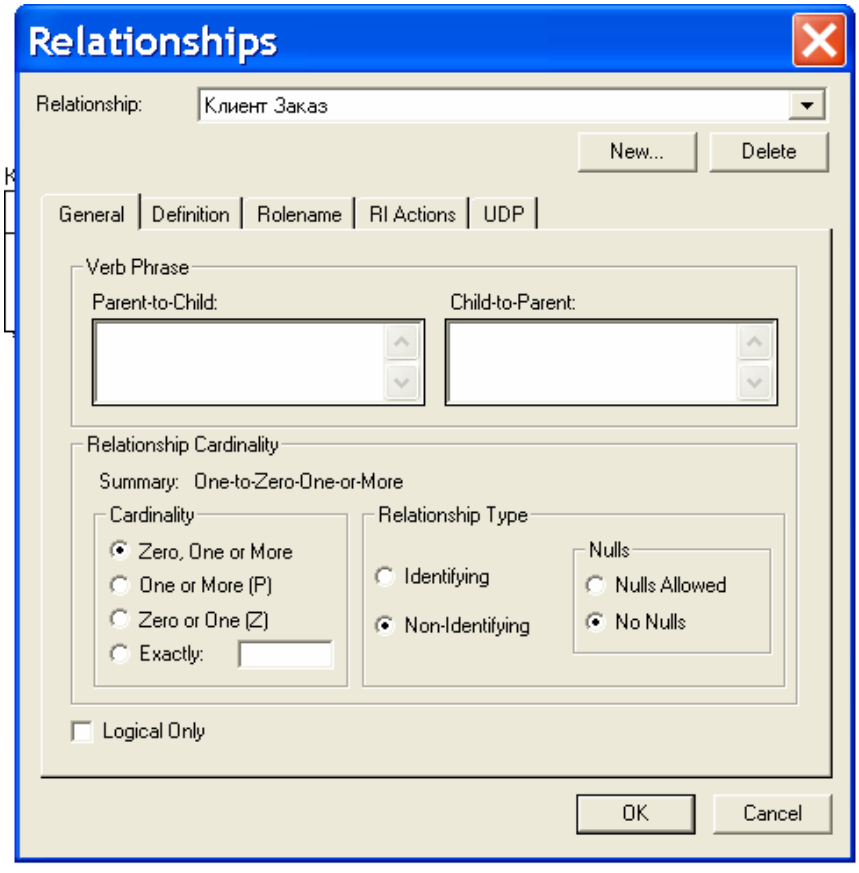


Рис. 7. Диалог Relationship Editor

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 необязатель-

ной связи (Nulls Allowed) внешний ключ может принимать значение NULL. Необязательная неидентифицирующая связь помечается прозрачным ромбом со стороны родительской сущности (см. рис. 2.22).

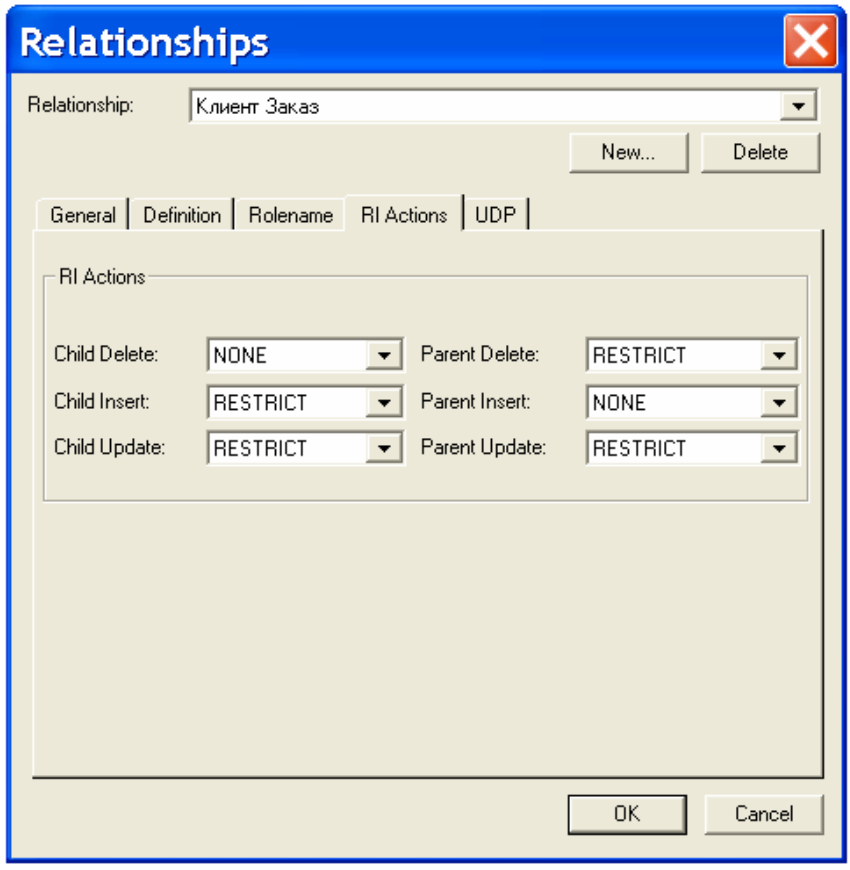


Рис. 8. Закладка Rolename/RI Actions диалога Relationship Editor

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

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

Связь многие-ко-многим возможна только на уровне логической модели данных. На рис. 9 вверху показан пример связи многие-ко-многим. Врач может принимать много пациентов, пациент может лечиться у нескольких врачей. Такая связь обозначается сплошной линией с двумя точками на концах.

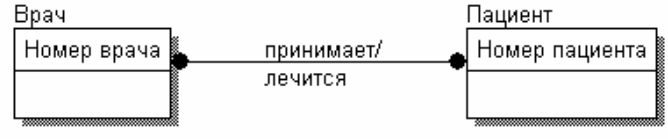


Рис. 9. Связь многие-ко-многим

Для внесения связи следует установить курсор на кнопке в палитре инструментов, щелкнуть сначала по одной, а затем по другой сущности.

Связь многие-ко-многим должна именоваться двумя фразами - в обе стороны (в примере "принимает/лечится"). Это облегчает чтение диаграммы. Связь на рис. 9 следует читать ***Врач***<принимает> ***Пациент"а, Пациент*** <лечится> у ***Врач"а.***

При переходе к физическому уровню ERWin автоматически преобразует связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старых к новой таблице (рис. 10). При 'этом имя новой таблице присваивается автоматически как ***“Имя1 Имя2".***

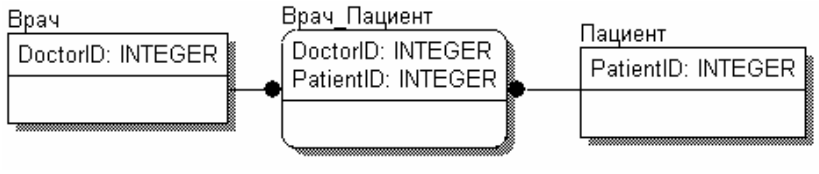


Рис. 10. Иллюстрация автоматического разрешения связи многие-ко-многим на уровне физической модели

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.

Ассоциативная - сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

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. 11), 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. 11).

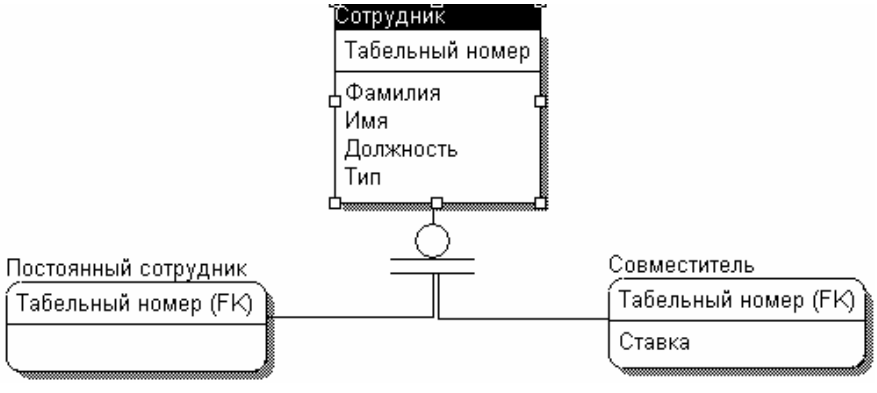


Рис. 11. Иерархия наследования

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. 12) можно указать атрибут - дискриминатор категории (список Discriminator Attribute Choice) и тип категории - полная/неполная (радиокнопки Complete/Incomplete).

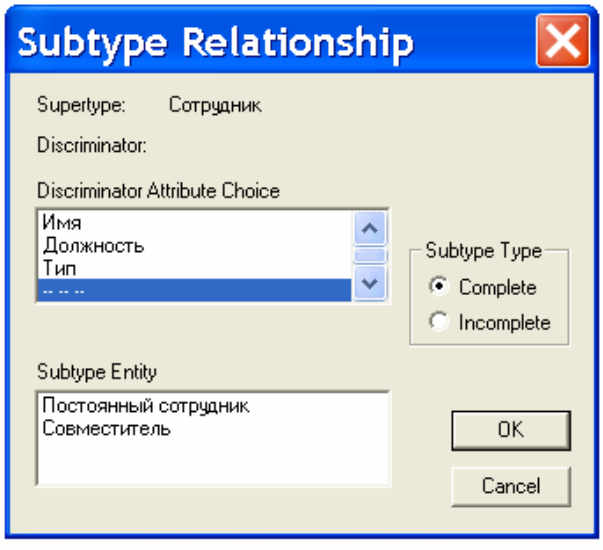


Рис. 12. Диалог Subtype Relationship

Ключи. Каждый экземпляр сущности должен быть уникален и отличаться от других атрибутов.

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 диалоге Attribute Editor для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок Primary Key в нижней части закладки General. На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

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:"

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.

Зависимая сущность может иметь один и тот же внешний ключ из нескольких родительских сущностей. Сущность может также получить один и тот же внешний ключ несколько раз от одного и того же родителя через несколько разных связей. Когда ERWin обнаруживает одно из этих событий, он распознает, что два атрибута одинаковы, и помещает атрибут внешнего ключа в зависимой сущности только один раз. Хотя в закладке Key Group диалога Attribute Editor этот атрибут будет входить в два внешних ключа, на диаграмме он показывается только один раз. Это комбинирование или объединение идентичных атрибутов называется **унификацией.**

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).

Поддержка нормализации в ERWin. ERWin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание нормализованной модели данных. Запрет на присвоение неуникальных имен атрибутов в рамках модели (при соответствующей установке опции Unique Name) облегчает соблюдение правила "один факт - в одном месте". Имена ролей атрибутов внешних ключей и унификация атрибутов также облегчают построение нормализованной модели.

Денормализация. В результате нормализации все взаимосвязи данных становятся правильно определены, исключаются аномалии при оперировании с данными, модель данных становится легче поддерживать. Однако часто нормализация данных не ведет к повышению производительности ИС в целом.

В отличие от процесса нормализации денормализация не может быть представлена в виде четко сформулированных правил. К сожалению, в каждом конкретном случае приходится искать конкретные решения, которые используют специфику ИС и предметной области и не могут быть тиражированы.

Поддержка денормализации в ERWin. Денормализация, как правило, проводится на уровне физической модели. ERWin позволяет сохранить на уровне логической модели нормализованную структуру, при этом построить на уровне, физической модели структуру (возможно, денормализован-ную), которая обеспечивает лучшую производительность, используя особенности конкретной СУБД и бизнес-правил предметной области.

ERWin имеет следующую функциональность для поддержки денормализации:

Сущности, атрибуты, ключи и домены можно создавать только на уровне логической модели, включив в соответствующих редакторах опцию Logical Only. Такие объекты не будут отображаться на уровне физической модели и не будут создаваться при генерации БД.

Таблицы, колонки, домены и индексы можно создавать только на уровне физической модели. При автоматическом разрешении связи многие-ко-многим в физической модели создается новая таблица и структура данных может быть дополнена только на уровне физической модели.

2. Создание физической модели данных

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. 13).

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, а также правила ссылочной целостности, принимаемые по умолчанию. Тип данных 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.

Окно выбора Allow special chars in names позволяет разрешить или запретить использование специальных символов и пробелов в именах таблиц. Эта опция действует только для тех СУБД, которые поддерживают использование специальных символов.

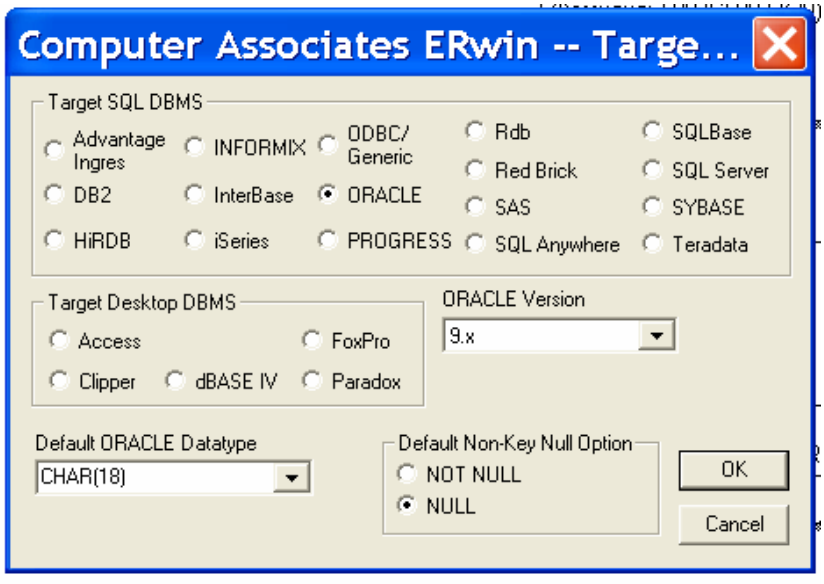


Рис. 13. Диалог Target Server

По умолчанию ERWin генерирует имена таблиц и индексов по шаблону на основе имен соответствующих сущностей и ключей логической модели. Окна Table Name Macro и Index

Name Macro позволяют изменить шаблон генерации имен, заданный по умолчанию. В дальнейшем имена таблиц и индексов можно изменить вручную.

Таблицы, колонки и представления (view). Для внесения новой таблицы в модель на физическом уровне служит кнопка на палитре инструментов. Связи между таблицами создаются так же, 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 сущностей 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. 13).

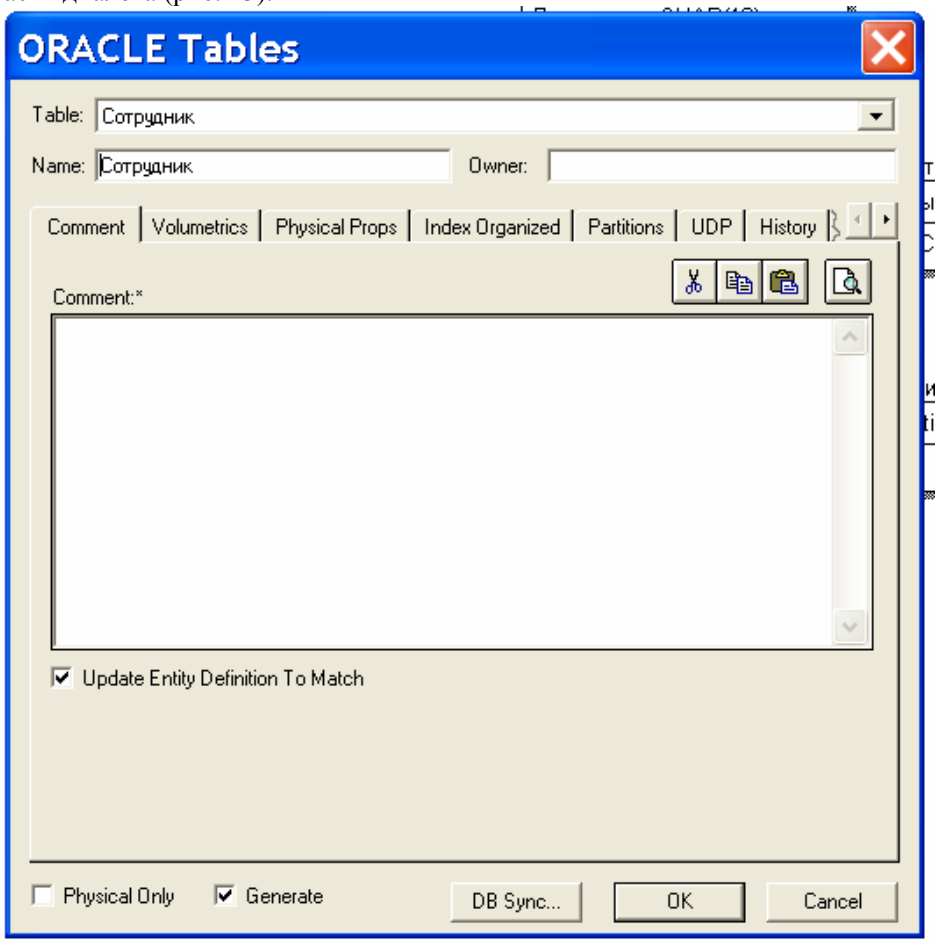


Рис. 14. Диалог Table Editor

Окно Name служит для задания имени текущей таблицы. Окно Owner позволяет внести имя владельца таблицы, отличное от имени пользователя, производящего генерацию схемы БД.

Окно выбора Physical Only служит для создания объектов 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.

Диалог Table Editor содержит следующие закладки:

Dimensional. Доступна только на уровне моделирования хранилищ данных (Dimensional Modeling) и будет рассмотрена ниже.

Comment. Внесение комментария к таблице.

Volumetrics. Служит для оценки размера БД.

Physical Props. Позволяет задать физические свойства таблицы.

Partitions. Служит для задания значений разделения. Доступна только для Oracle 8.x.

UDP. Задание свойств, определяемых пользователем.

Validation. Задание правил валидации.

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

Stored Procedure. Связывание с таблицей хранимых процедур.

Pre & Post Script. Создание скриптов (наборов команд), которые будут выполняться до и после создания таблицы при генерации схемы БД.

PowerBuilder. Задание расширенных атрибутов для генерации кода клиентского приложения на PowerBuilder.

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 представлений 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. Связи с представлениями и прямоугольники представлений показываются на диаграмме пунктирными линиями. Для редактирования представления служит диалог View Editor (рис. 14). Для его вызова следует щелкнуть правой кнопкой мыши по представлению и выбрать в меню пункт View Editor.

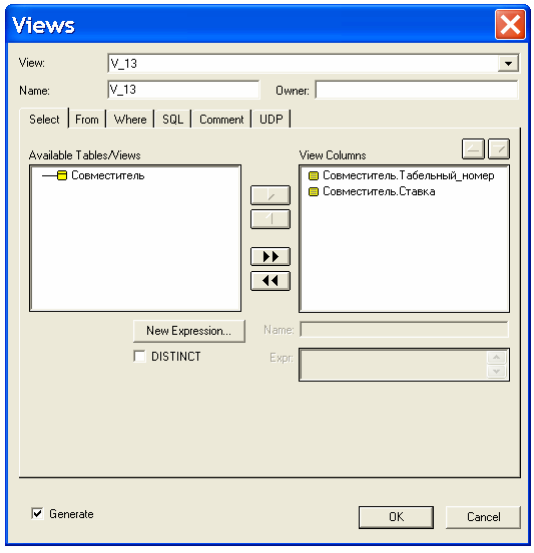


Рис. 15. Диалог View Editor

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. 15). Имеет два списка: в правом отображаются колонки представления, в левом - колонки доступные для включения в представление. Кнопка New Expression позволяет задать выражение в качестве выходного столбца. По умолчанию при создании связи ERWin включает в представление все колонки родительских таблиц.

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.

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.

Правила валидации и значения по умолчанию

ERWin поддерживает правила валидации для колонок, а также значение, присваеваемое 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, появляется диалог Validation Rule Editor (рис. 16), 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).

В верхней части редактора Validation Rule 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 и щелкнуть по кнопке OK. После этого можно ввести выражение для правила валидации. Поля Min и Max служат для задания нижней и верхней границы диапазона значения.

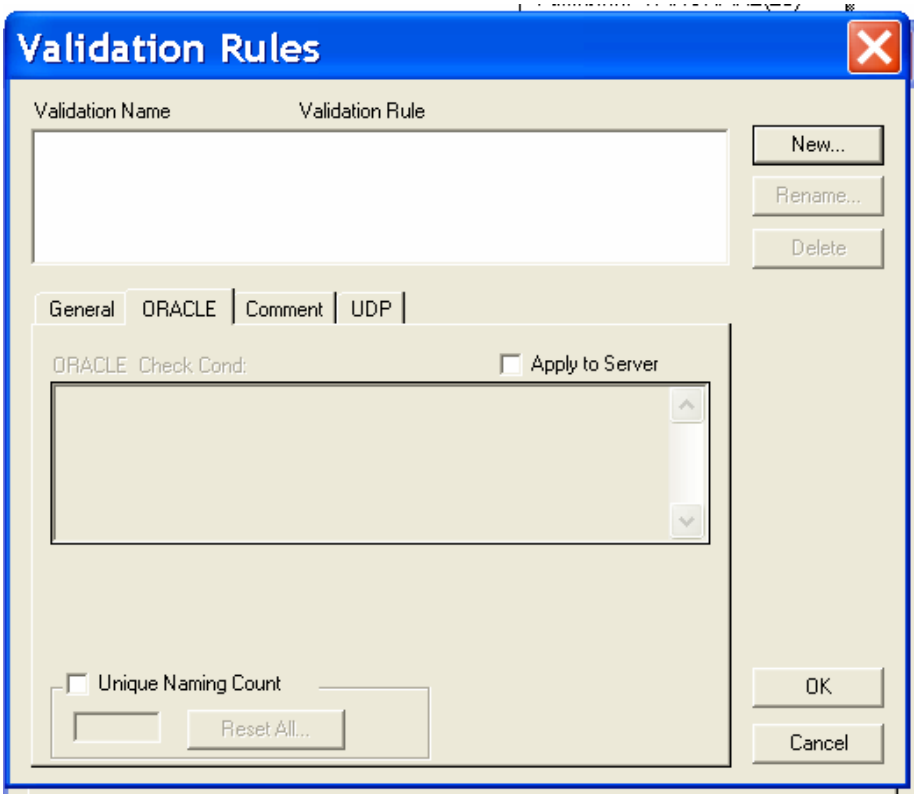


Рис. 16. Диалог Validation Rule Editor

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.

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, "IEn" -для инверсионного входа. Например, по умолчанию при создании таблицы ***CUSTOMER*** будут созданы индексы XPKCUSTOMER (первичный ключ, в состав войдет колонка ***CustomerID),*** XAK1CUSTOMER (альтернативный ключ, колонки ***CusfomerName, Region, City),*** XIE1CUSTOMER (инверсионный вход 1, колонка ***Region)*** и XIE2CUSTOMER (инверсионный вход 2, колонка ***CustomerAddress).***

Изменить характеристики существующего индекса или создать новый можно в редакторе Index Editor (рис. 17). *Для* его вызова следует щелкнуть правой кнопкой мыши по таблице и выбрать во -всплывающем меню пункт Index.

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 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.

Иногда необходимо разрешить повторяющиеся значения, если ожидается, что индексированная колонка будет с большой вероятностью содержать повторяющуюся информацию. Например, на основе инверсионных входов, которые используются для доступа к нескольким строкам и поэтому могут содержать повторяющиеся значения, генерируется неуникальный индекс. Неуникальный индекс генерируется также на основе внешнего ключа. На основе первичного и альтернативных ключей генерируются уникальные индексы.

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.

По умолчанию ERWin автоматически сохраняет значена . в порядке возрастания (значения сортируются по алфавиту от A до Z, а числа от 0 до 9). Если нужно изменить порядок сортировки для колонки и выбранная СУБД поддерживает режим сортировки по убыванию, следует выбрать колонку и включить опцию DESC (см. рис. 17).

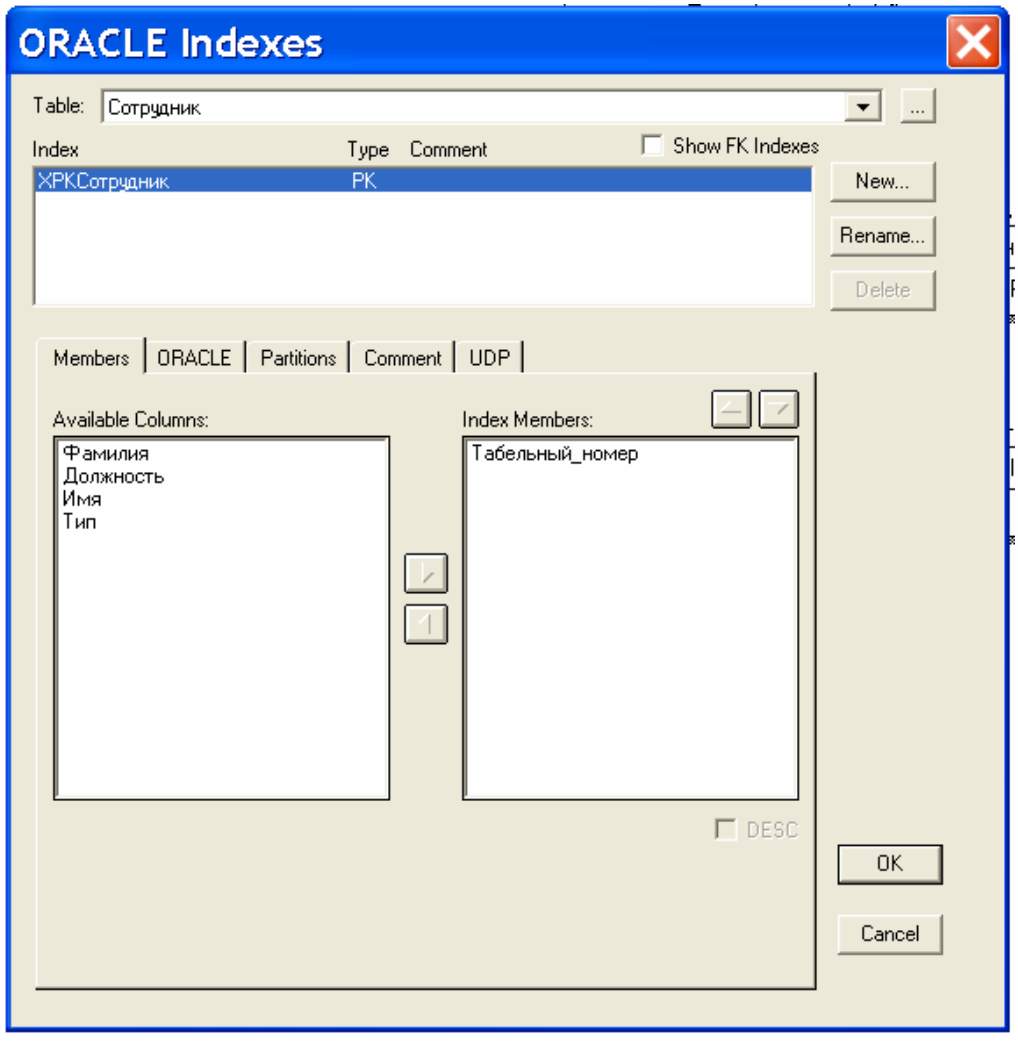


Рис. 17. Диалог Index Editor

Редактор Index Editor содержит следующие закладки:

Members - позволяет включить колонки в состав индекса;

Comment - содержит комментарий для каждого индекса;

UDP - позволяет связать с индексом свойства, определяемые пользователем

Закладка, соответствующая выбранной СУБД задает свойства индекса, специфические для выбранной СУБД.

Некоторые СУБД поддерживают кластеризованные и кластеризованные хешированные индексы. ERWin позволяет создать такие индексы для DB2/MVS, DB2/390, HiRDB, INFORMIX, MS Access, MS SQL Server, SYBASE и SQLBase. Для того чтобы сделать индекс кластеризованным, нужно включить опцию CLUSTER в закладке, соответствующей выбранной СУБД. Кластеризованный индекс - это специальная техника индексирования, при которой данные в таблице физически располагаются в индексированном порядке. Использование кластеризованного индекса значительно ускоряет выполнение запросов по индексированной колонке.

Поскольку данные физически расположены в индексированном порядке, для каждой таблицы может существовать только один кластеризованный индекс. Если СУБД поддерживает использование кластеризованного индекса, то ERWin автоматически создает индекс первичного

ключа кластеризованным. При создании кластеризованного индекса не по первичному ключу ERWin автоматически снимает кластеризацию с индекса по первичному ключу. Для СУБД SQLBase (CENTURA) ERWin позволяет создать кластеризованный хешированный индекс (clustered hashed index). Хеширование -альтернативный способ хранения данных в заранее заданном порядке с целью ускорения поиска, но физически это более сложно, чем простое сохранение строк в алфавитном порядке или в соответствии с числовыми значениями.

Кластеризованный или хешированный индекс значительно ускоряет операции поиска и сортировки, но добавление и удаление строк замедляется из-за необходимости реорганизации данных для соответствия индексу.

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

Для генерации триггеров ERWin использует механизм шаблонов - специальных скриптов, использующих макрокоманды. При генерации кода триггера вместо макрокоманд подставляются имена таблиц, колонок, переменные и другие фрагменты кода, соответствующие синтаксису выбранной СУБД. Шаблоны триггеров ссылочной целостности, генерируемые Erwin, по умолчанию можно изменять, кроме того, можно переопределить как триггеры для конкретной связи, так и шаблоны во всей модели в целом.

Шаблоны триггеров ссылочной целостности связываются с сущностями в зависимости от типа связи и роли сущности в этой связи. Тип связи и роль сущности определяют, какое правило ссылочной целостности будет по умолчанию дополнено шаблоном триггера. Связи могут быть: идентифицирующими, неидентифицирующими (nulls allowed), неидентифицирующими (no nulls), связями подтипа.

Сущность в связи может быть родительская (Parent) или дочерняя (Child). Если сущность является родительской в данной связи, то ERWin присваивает ей шаблон триггера для родительской сущности. Если сущность является дочерней в данной связи, то ERWin присваивает ей шаблон триггера для дочерней сущности. Код триггера, который генерируется шаблоном триггера для родительской сущности, указывает СУБД, что нужно делать при вставке, изменении или удалении строки в родительской таблице связи. Код триггера, который генерируется шаб-

лоном триггера для дочерней сущности, указывает СУБД, что нужно делать при вставке, изменении или удалении строки в дочерней таблице связи.

Ниже приведен текст шаблона триггера, соответствующего правилу ссылочной целостности ON PARENT DELETE RESTRICT.

```
/* ERWin Builtin %Datetime 7
/* %Parent %VerbPhrase %Child ON PARENT DELETE RESTRICT */ select count(*) into numrows
from %Child
where
/* %%JoinFKPK(%Child,:%%Old; ="," and") */
%JoinFKPK(%Child,:%Old," ="," and");
if (numrows > 0) then
raise_application_error(
-20001,
'Cannot DELETE %Parent because %Child exists.'
);
end if;
```

При генерации схемы СУБД для Oracle 7.2 будет сгенерирован триггер:

```
create trigger tD_CUSTOMER after DELETE on CUSTOMER for each row
- ERWin Builtin Tue Jan 26 21:55:13 1999
-DELETE trigger on CUSTOMER declare numrows INTEGER;
begin
/* ERWin Builtin Tue Jan 26 21:55:131999 7
/* CUSTOMER размещает ORDER ON PARENT DELETE RESTRICT */
select countf*) into numrows
from ORDER
where
/* %JoinFKPK(ORDER,:%Old," ="," and") */
ORDER.CustomerlD = :old.CustomerlD;
if (numrows > 0)
then
raise_application_error(
-20001,
'Cannot DELETE CUSTOMER because ORDER exists.'
):
end if;
```

ERWin по умолчанию использует для генерации кода триггера на языке SQL встроенные шаблоны триггеров ссылочной целостности, которые автоматически присваиваются каждой связи. Если встроенные шаблоны не удовлетворяют бизнес-правилам, можно изменить коды триггера, генерируемые на основе встроенных шаблонов. ERWin позволяет изменить шаблон и указать, что при генерации модифицированная версия должна заменить встроенный шаблон.

ERWin позволяет переопределить триггер, устанавливаемый по умолчанию, тремя способами:

1. Переопределение типа ссылочной целостности. Для каждой комбинации правил ссылочной целостности (например, Parent-Delete RESTRICT) ERWin позволяет создать переопределенный шаблон и использовать этот шаблон вместо шаблона, применяемого по умолчанию, для всех связей диаграммы, которым был присвоен этот тип правила ссылочной целостности. Используя в качестве основы встроенный код шаблона, можно изменить шаблоны ссылочной целостности во всей модели, изменяя коды триггера только в одном месте. Вновь определяемые шаблоны будут использоваться вместо стандартных шаблонов, если при генерации схемы включена опция RI Type Override.

2. Переопределение шаблона триггера для связи. Можно переопределить шаблон, задаваемый по умолчанию, для какой-то конкретной связи. Для этого используется диалог Relationship Template Editor, в котором можно описать шаблоны Relationship Override, применяемые вместо стандартных шаблонов (а также вместо шаблонов RI Type Override, если они есть). Шаблоны Relationship Override будут использоваться вместо стандартных шаблонов, если при генерации схемы включена опция Relationship Override.

3. Переопределение шаблона триггера для сущности. ERWin позволяет создавать собственные триггеры Entity Override для любой сущности в модели. Шаблоны Entity Override используются вместо стандартных шаблонов, а также вместо шаблонов RI Type Override и Relationship Override, если при генерации схемы включена опция Entity Override.

ERWin имеет специальные редакторы, облегчающие создание и редактирование триггеров и процедур.

Для редактирования триггера следует щелкнуть правой кнопкой мыши по таблице и выбрать во всплывающем меню пункт Trigger. Появляется диалог Table Trigger Viewer, в нижней части которого имеется две кнопки -Table Trigger и Trigger Template, которые вызывают диалоги, предназначенные для создания и редактирования пользовательских триггеров и шаблонов триггеров ссылочной целостности

Список литературы

1. *Маклаков С.В.* BPwin и ERwin: CASE-средства для разработки информационных систем. – М.: Диалог-Мифи, 1999. - 295 с.

2. *Федорова Д.Э.*, *Семенов Ю.Д.*, *Чижик К.Н.* CASE-технологии. - М.: Горячая линия Телеком, Радио и связь, 2005. – 160 с.