

## ТЕМА 13

План лекции:

1. Моделирование данных в среде Rational Rose.
2. Создание структуры данных в среде Rational Rose.
3. Моделирование данных при помощи Data Modeler в Rational XDE.

### 13.1 Моделирование данных в Rational Rose

С появлением подключаемого модуля *Data Modeler* у разработчиков появилась возможность использовать среду автоматизированного синтеза Rational Rose не только для создания логического представления системы, но и для моделирования физического представления данных.

Поскольку язык UML не имеет в своем составе средств, позволяющих адекватно отображать физическую модель данных, для этого предусмотрены дополнительные стереотипы классов, которые не имеют отражения в стандартном языке UML. Поэтому *Data Modeler* не работает с новым подмножеством языка, а является только инструментом для моделирования физической структуры данных. *Data Modeler* позволяет создавать все необходимые объекты базы данных: таблицы, триггеры, хранимые процедуры и представления данных. Модуль поддерживает работу с основными системами обработки баз данных: *IBM DB2 MVS, UDB, Oracle, Microsoft SQL Server, Sybase Adaptive Server*.

После установки в среду модуля *Data Modeler* в меню Tools появляется дополнительный раздел *Data Modeler*, состоящий из трех пунктов: *Add Schema, Add Domain Package, Reverse Engineering*. Рассмотрим их подробнее.

1. *Add Schema* – добавить схему. Схема является основным контейнером для построения модели данных, поэтому она создается в первую очередь.

2. *Add Domain Package* – добавить доменный пакет. Домены позволяют определять типы данных, длины, точность, значения по умолчанию для колонок таблицы данных и могут использоваться для различных схем данных.

3. *Reverse Engineering* – обратное моделирование. Оно позволяет создавать модель по готовой базе данных. При запуске *Reverse Engineering* активируется мастер, который помогает провести обратное моделирование как из базы данных, так и из скрипта на языке определения данных. Для базы данных достаточно задать тип и параметры соединения.




Эти же пункты доступны из контекстных меню объектов и диаграмм.

### 13.2 Создание структуры данных

Для разработки новой структуры данных необходимо выполнить следующие действия:

1. Создать схему данных.
2. Создать таблицы данных в схеме.
3. Создать связи данных.
4. Перенести созданные объекты в базу данных.

Схема данных позволяет трансформировать физическую модель в логическую и обратно, а также создавать диаграмму модели данных – *Data Model Diagram*. Для построения схемы данных необходимо перейти в окно Browse и из контекстного меню *Logical View* выбрать следующее: *Data Modeler => New => Schema*. Для построения физической модели данных имеется специальная диаграмма *Data Model Diagram*. Она создается из контекстного меню схемы базы данных при помощи пункта *Data Modeler => New => Data Model*. При активизации диаграммы становятся доступными ее инструменты, позволяющие создавать объекты баз данных и связи между ними. На рис. 131 приведен пример диаграммы данных.

Инструмент таблица –  предназначен для создания таблицы в схеме данных. Инструмент  используется для создания не идентифицирующей реляционной связи. Инструмент  позволяет создать идентифицирующую реляционную связь. Начиная с версии Rational Rose 2002, имеется еще один инструмент, позволяющий создать представление данных.

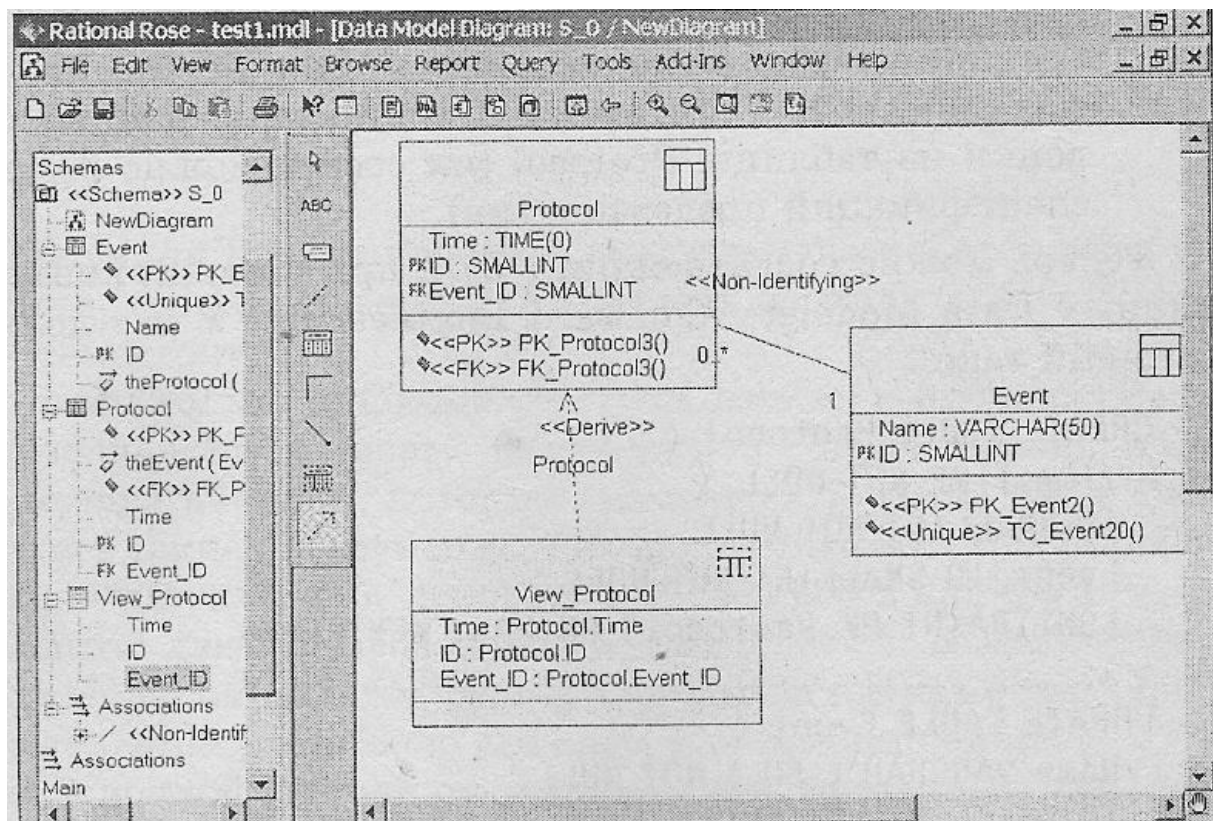


Рисунок 13.1 – Модель данных

Графически в области диаграммы данных таблица представляется прямоугольником, разделенным на три части (рис. 13.2). Вверху отражено название таблицы, затем имена и типы столбцов. Ключевое поле отмечено красным маркером «PK» (Primary Key), далее могут быть показаны ограничения целостности, индексы и триггеры, каждый из которых предлагается со своим стереотипом. Для задания колонок, ключей, индексов

и триггеров необходимо активизировать спецификации для таблицы. Окно спецификации

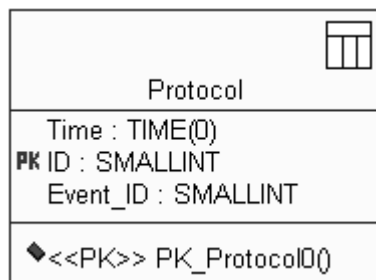


Рисунок 13.2 – Изображение таблицы на диаграмме данных

имеет шесть вкладок. Рассмотрим их и некоторые из их полей. Зкладка General позволяет задать следующие поля:

- Name – имя таблицы.
- Schema – схема данных, к которой принадлежит таблица.
- Mapped – имя класса, с которым связана таблица.

Зкладка Columns позволяет создать колонки в таблице данных, для чего существуют следующие поля:

- Name – имя колонки.
- PK – устанавливает, что колонка является первичным ключом.
- Type – тип данных.
- Not Null – определяет, что значения в колонке не могут быть Null.
- Unique – определяет, что значения должны быть уникальными.
- Default – задает значения по умолчанию.

Зкладка Key Constraints позволяет задать ограничения для колонок. Здесь можно ввести ограничения на уникальность первичного ключа и альтернативных ключей, а также определить индекс. Зкладка Check Constraints задает содержание ограничения. Зкладка Triggers содержит определения триггеров.

После создания таблиц необходимо установить связи между ними. Для модели данных используются два вида связей. Идентифицирующая – сильная связь между родительской и дочерней таблицами базы данных. Не идентифицирующая – независимая связь между таблицами, когда каждый первичный или уникальный ключ помещается автоматически в дочернюю таблицу как foreign key и помечается красным маркером FK. Для идентифицирующей связи в ограничениях первичного ключа учитывается foreign key. После создания связи необходимо настроить ее спецификации, включающие в себя три вкладки:

- General – основные свойства связи, здесь задаются ее имя, тип, наименования ролей (Parent, Child).
- Migrated Key – содержит список внешних ключей, образующихся в результате создания связи.

- RI – задание условий ссылочной целостности или на основе триггеров, или на основе декларативной ссылочной целостности, используя ограничения внешних ключей.

По завершении выполнения всех необходимых действий для создания структуры базы данных можно перейти к пункту Forward Engineering и построить скрипт DDL, который затем использовать для генерации структуры базы данных.

### 13.3 Моделирование данных при помощи Data Modeler в Rational XDE

При разработке программных систем процесс создания модели данных является одним из важнейших этапов. Для его реализации в Rational XDE включен специальный модуль для моделирования данных – *Data Modeler*. Он позволяет моделировать физическую структуру данных. В *Rational XDE* включены дополнительные стереотипы классов, не имеющие отражения в стандартном языке UML, не позволяющие на основании стандартных элементов создавать модели структур данных.

*Data Modeler* предназначен для работы со всеми необходимыми объектами базы данных: таблицами, триггерами, хранимыми процедурами и представлениями данных. Модуль может моделировать данные в стандарте ANSI SQL, а также поддерживает форматы для основных систем управления базами данных: Oracle, Microsoft SQL Server, Sybase.

Для того чтобы воспользоваться возможностями моделирования данных и выполнением обратного проектирования, необходимо включить в модели поддержку профиля *Data Modeler*. В случае если модель уже создана, в ее свойства необходимо добавить поддержку работы с базами данных. Для этого необходимо выполнить следующие действия:

1. Перейти в окно *Model Explorer*.
2. Перейти в созданную модель, например виртуальный магазин.
3. Из контекстного меню модели выбрать пункт *Properties Window*.
4. В окне *Properties* установить значение свойства *Applied Profiles* = *Date Modeler*.

После этого в контекстном меню модели появятся новые пункты, относящиеся к проектированию данных и обратному проектированию (рис. 13. 3).

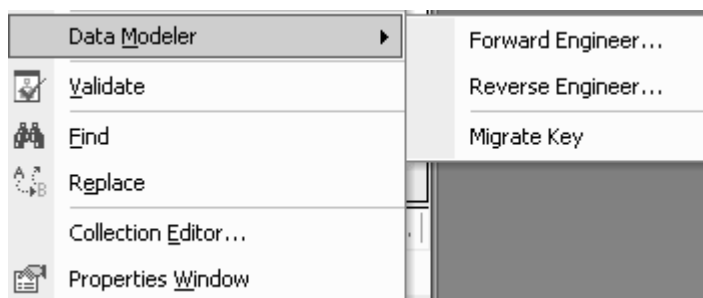


Рисунок 13.3 – Пункт Data Modeler в контекстном меню модели

Для запуска режима *Reverse Engineering* необходимо выбрать соответствующий пункт в контекстном меню модели, после чего активизируется мастер обратного проектирования. Он позволяет по шагам пройти весь процесс обратного проектирования, предоставляя инструкции по дальнейшим действиям. Поскольку для различных систем управления БД синтаксис процедур и таблиц данных различен, то мастер позволяет указать конкретную БД для проектирования, начав с указания источника, в котором будет выполняться поиск БД. Таких источников может быть два: непосредственно БД и описание на языке определения данных DDL. В последнем случае возможно указание и SQL скрипта, по которому будет создана структура.

Следующий шаг – это определение типа БД и параметров соединения. Затем следует указать необходимую схему БД. Позже схема преобразуется в пакет, который содержит все полученные из БД элементы. Далее необходимо указать, какие элементы нужно получить из БД. После окончания работы будет создан пакет *dbo*, в котором присутствуют все полученные из БД элементы. Однако пока модель еще трудна для обзора, так как все элементы представлены в окне Model Explorer и связи между ними не очевидны. Для того чтобы получить наглядную диаграмму, необходимо перенести все элементы в область рисования диаграммы и расставить связи между элементами.

Таким образом, для работы с моделями данных необходимо выполнить несколько общих шагов. Сначала создать файл модели, затем в ней создать базу данных, которую наполнить различными элементами: таблицами, триггерами, хранимыми процедурами и представлениями.

На рис. 13.4 показано окно инструментов, используемых для создания модели данных.



Рисунок 13.4 – Окно инструментов Data Modeler

Значок *Database* позволяет создать на диаграмме отображение базы данных. Он представляется аналогией компонента для класса, поскольку БД

является контейнером, объединяющим все другие элементы. С помощью окна спецификаций можно настроить значительное количество дополнительных параметров.

Элемент *Table Space* не имеет значка в окне инструментов, однако он может быть создан посредством пункта *Add Data Modeler => Table Space* контекстного меню *Database*. *Table Space* – логическое хранилище, в которое помещаются таблицы данных. Возможно создание нескольких таких элементов. В момент создания *Table Space* выполняется автоматическое связывание его с элементом *Database* при помощи связи *Dependency*, а связи с таблицами данных определяются при помощи *Realization*, которую необходимо с каждой таблицей создавать вручную.

Пункт *Table* позволяет создать на диаграмме отображение таблицы данных. Хотя в окне инструментов для таблицы используется свой значок, в самой диаграмме она изображается как обычный класс со стереотипом «Table». В дальнейшем таблицу также можно преобразовать в класс.

Значок *View* позволяет отразить в модели представление данных. Элемент является виртуальной таблицей, которая физически не присутствует в БД, и создается в момент запроса. Доступ к *View* такой же, как и к обычным таблицам. Поскольку создание *View* оптимизируется на уровне СУБД, то использование представлений позволяет увеличить производительность приложений. Представления также могут использоваться для предоставления пользователям доступа к данным без возможности внесения изменений.

Создание модели реляционной БД невозможно без установления связей между таблицами. Rational XDE поддерживает все необходимые типы связей, которые используются при создании структуры данных.

Значок *Identifying Relationship* позволяет создать на диаграмме отображение такой связи между таблицами. Она отображает связь между родительской и дочерней таблицами, где дочерняя таблица не может существовать без родительской. При создании такого типа связи создается элемент внешнего ключа (*foreign key*), для которого автоматически создается ограничение (*constraint*) (рис. 13.5).

Значок *Non-Identifying Relationship* позволяет создать на диаграмме отображение отношений между таблицами, при котором нет необходимости в создании внешнего ключа, т.е. записи дочерней таблицы могут существовать и без ссылки на родительскую таблицу. Обычно это показывают мощностью связи (*cardinality*), например, 0..1 (рис. 13.6).

Значок *Many To Many Relationship* не создает нового типа связей «многие ко многим», поскольку такая связь не может быть создана без промежуточной таблицы. При выборе этого типа связи система сама создает еще одну таблицу и связывает с ней родительские таблицы идентифицирующими связями.

Значок *Dependency* позволяет указать зависимость одного элемента диаграммы данных от другого, причем делается это системой автоматически.

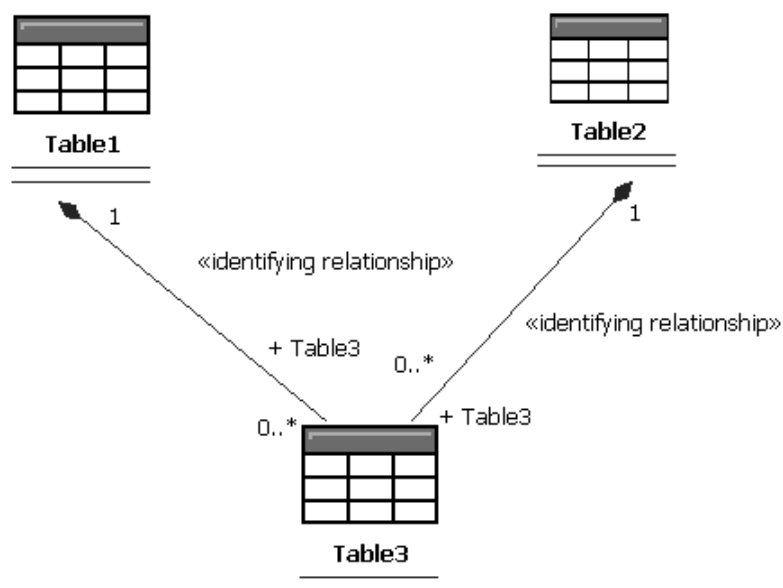


Рисунок 13.5 – Таблицы с идентифицирующими связями

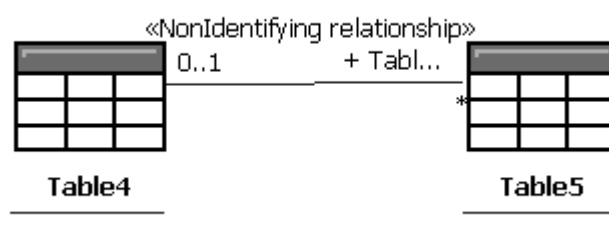


Рисунок 13.6 – Таблицы с не идентифицирующими связями

Элемент *Realization* позволяет показать реализацию элемента.

Пункт *Stored Procedure* позволяет создать на диаграмме контейнер для хранимой на сервере процедуры. Она представляет собой скрипт, который может быть в любой момент запущен любым клиентом.

Пункт *Domain* позволяет создать шаблон для типов данных пользователя, а также для определения правил обработки колонок в таблицах.