Ядро объектно-реляционной системы ODESTOR

В.В. Рубанов, М.А. Миткевич, Д.А. Марковцев, А.И. Гриневич

Аннотация. В статье рассматриваются детали реализации ядра системы объектно-реляционного отображения ODESTOR. Описание основывается на схеме работы с системой, рассматриваются аспекты работы с каждым модулем для случаев прямого и обратного проектирования (forward и reverse engineering).

1. Введение

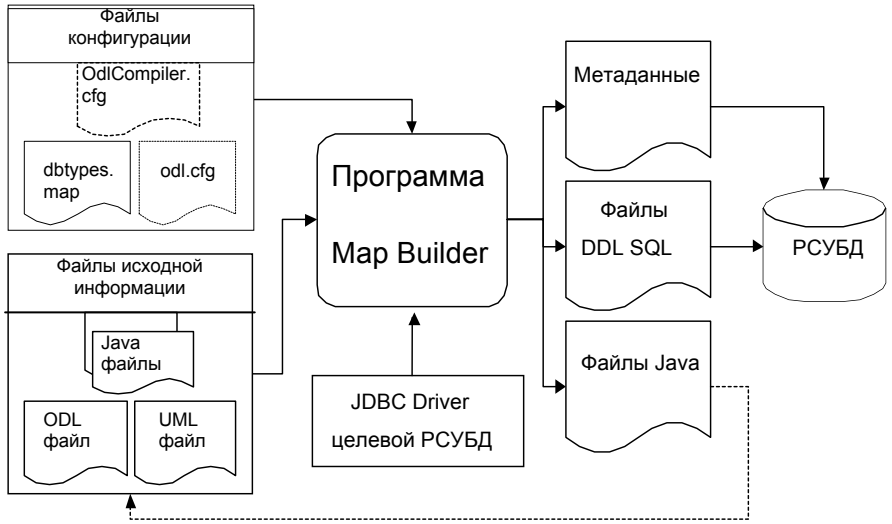
В данной статье рассматривается технология работы с системами объектно-реляционных трансформаций на примере ядра программной среды ODESTOR (Object Database Engine System on the Top Of Relations). Актуальность использования таких систем обусловлена, с одной стороны, наличием на рынке развитых объектно-ориентированных технологий, причем в области долговременного хранения информации все большее признание находят объектно-ориентированные базы данных (ООБД), а с другой стороны, существованием в мире массы унаследованных систем на основе реляционных систем (РСУБД). В результате возникает необходимость в построении системы объектно-реляционной трансляции, которая представляет собой комплекс утилит и библиотек, обеспечивающего полный цикл разработки прикладного программного обеспечения и эмулирующего для прикладного программиста работу ООБД, но использующего в качестве реального хранилища данных реляционную базу данных. Центральным моментом статьи является описание объектно-реляционного ядра такой системы. Рассматриваются задачи как прямого (Forward engineering), так и обратного (Reverse Engineering) проектирования. В первом случае проект пишется с чистого листа, и реляционные таблицы могут быть сформированы системой автоматически. Во втором случае уже имеется заданная структура реляционных таблиц, и нужно с помощью специальных средств описать, как объектные данные будут отражаться в данных таблицах. В настоящее время в рамках проекта ODESTOR реализованы основные модули объектно-реляционной трансляции исходных метаданных и модули базового интерфейса ООБД. Планируются работы по расширению возможностей этого интерфейса и повышению удобства использования системы со стороны прикладного программиста

(графические утилиты). Работа поддерживалась грантами РФФИ и выполняется в рамках внутреннего проекта ИСП РАН. Начальные сведения о проекте можно найти в [27] – [28].

В первом разделе статьи описываются методы создания объектной модели целевой задачи. Следующий этап, обсуждаемый во втором разделе, -- это описание реляционной схемы прикладной системы. Наконец, в третьем разделе рассматривается процесс собственно определения способов объектно-реляционного отображения, заданных на первых этапах. Описываются различные аспекты работы системы, как с точки зрения пользователя, так и с точки зрения программиста. В заключение приводится общая схема работы с системой, отражающая полный цикл создания приложений, в которых для хранения объектных данных используются возможности рассматриваемой системы.

2. Создание объектной модели

В процессе разработки прикладной части, использующей возможности рассматриваемой системы, в первую очередь возникает задача описания схемы классов целевого приложения. На основании данной схемы формируются описания *метаданных* в реляционной базе данных, которые будут затем использоваться при работе приложения (run-time). Кроме того, данные об объектной схеме используются системой для формирования шаблонов классов на целевом языке разработки (в данном случае Java). Также при решении задачи прямого проектирования метаданные используются для формирования SQL-скриптов для создания таблиц, в которых будут храниться данные сохраняемых объектов. Общая схема работы по обработке объектной схемы классов приложения выглядит следующим образом:



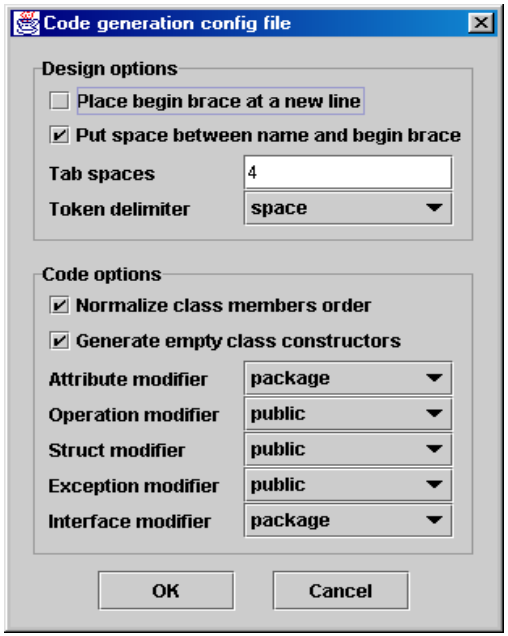
3.1 Файлы конфигурации

Программа Map Builder использует в своей работе ряд конфигурационных файлов, которые играют центральную роль в работе системы. В файле odl.cfg указываются опции по формированию скелетонов Java классов на основе описания объектной схемы на языке ODL или UML. Пример содержания файла имеет следующий вид:

```
[Edit]
parenthnewline = false
parenthspace = true
identspaces = 4
tokendelimeter = space

[Class]
classmemberorder = standard
constructors = true
attrmodifier = package
opermodifier = public
structmodifier = public
exceptmodifier = public
interfmodifier = package
```

Параметры в разделе Edit задают форматирующие опции, указывающие, в частности, следует ли переносить открывающую фигурную скобку на новую строку, использовать пробел для отделения фигурных скобок от текста, количество пробелов в табуляции и тип разделителя лексем. В разделе Class указываются опции, связанные с семантикой генерации кода. Эти опции указывают, нужно ли генерировать скелеты конструкторов, менять порядок следования членов класса в соответствии со стандартной организацией, задают модификаторы для различных элементов языка Java. Графический интерфейс по настройке этого файла выглядит следующим образом:

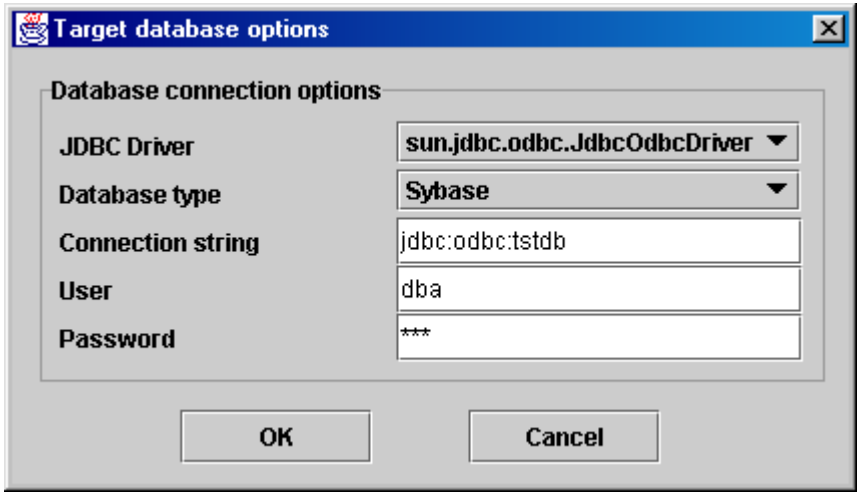


Файл dbtypes.map определяет соответствие элементарных типов данных, используемых в объектной модели, «родным» реляционным типам данных конкретных РСУБД (в данное время поддерживаются Oracle, Sybase, MS SQL Server и MS Access (для быстрого тестирования)). Эта информация используется при генерации SQL файлов описания реляционной схемы при решении задачи прямого проектирования. Пример раздела файла для Oracle выглядит следующим образом:

```
[Oracle]
short = NUMBER(5)
ushort = NUMBER(10)
long = NUMBER(10)
ulong = NUMBER(19)
float = NUMBER
double = NUMBER
char = CHAR(1)
boolean = NUMBER(1)
octet = NUMBER(3)
string = VARCHAR2(255)
date = DATE
time = DATE
timestamp = DATE
```

Наконец, в файле ODLCompiler.cfg хранятся настройки параметров самой программы Map Builder, в частности, тип целевой РСУБД, строка соединения JDBC для нее, имя пользователя и пароль.

Содержимое файлов odl.cfg и ODLCompiler.cfg можно редактировать как с помощью текстовых редакторов, так и с помощью визуальных настроек внутри программы Map Builder (см. рисунки):

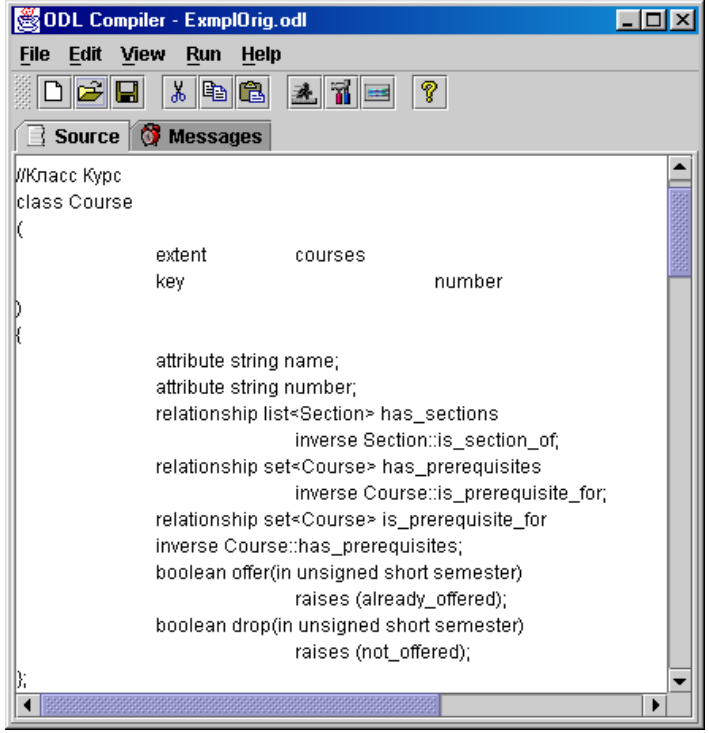


3.2 Файлы исходной информации

Исходная информация о структуре объектной схемы приложения может быть предоставлена системе одним из трех способов:

- Описание на языке Object Definition Language (ODL)
- Описание в виде файлов Java
- Описание в виде диаграмм UML

Первый способ рекомендован ODMG и описан в соответствующем стандарте. Пример кода на этом языке в окне программы имеет следующий вид:



То есть на данном языке описываются все элементы объектной модели с использованием текстовой нотации, аналогичной языку Java. Более подробно об ODL см. [9] - [10].

Второй и третий способы продиктованы практическими потребностями. В частности, регенерация файлов Java необходима для обеспечения цикличности разработки, то есть схема разработки может выглядеть следующим образом. Начальная объектная схема описывается на ODL, затем с использованием Map Builder формируются файлы Java. После этого они редактируются программистом и, чтобы отразить внесенные изменения в объектной схеме БД, опять подаются на вход Map Builder. Так обеспечивается цикличность использования Java кода. Наряду с файлами Java, в соответствии с последней версией стандарта [10], используются специальные дополнительные конфигурационные файлы для отражения информации объектной модели ODMG, которая отсутствует в Java. В частности, конфигурационный файл содержит такую информацию о свойствах хранимых классов, как собственно свойство хранимости (persistency), описания локальных (transient) атрибутов и расширенную информацию о связях.

3.3 Структура метаданных объектной модели

В результате обработки объектной модели соответствующим компилятором (ODL, Java или UML) в системе формируется описание объектной схемы приложения. Для внутреннего представления используются специальные классы. Эта система классов с префиксом *Sch* предназначена для представления структуры классов приложения, описанной в исходных файлах (метаданные приложения). Корневым классом является SchemaDcl. Объект данного класса можно создать на основе объекта спецификации или загрузить из указанной базы данных. Первый способ задействуется в модуле компиляции программы Map Builder и для модуля ODL выглядит следующим образом:

```
import RU.ISPRAS.ODLComp.*;

// инициализация входного потока
ODLParser     parser;
try {
       parser = new ODLParser(new
FileReader("inputfile.odl"));
} catch (java.io.FileNotFoundException e) {
       // ERROR
}

// генерация ODL спецификации входного файла
ODLSpec       odlSpecification;
try {
       odlSpecification = parser.getSpecification();
} catch (ParseException e) {
       // ERROR
}

// инициализация схемы классов
SchemaDcl     schema = new SchemaDcl(odlSpecification);
```

Другой способ используется во время выполнения прикладных программ. При этом данные загружаются из специальных таблиц РСУБД. Для данного способа существуют два конструктора – задание явной строки соединения JDBC, пользователя и пароля, либо использование уже созданного соединения JDBC (до вызова конструктора нужный JDBC драйвер должен быть загружен):

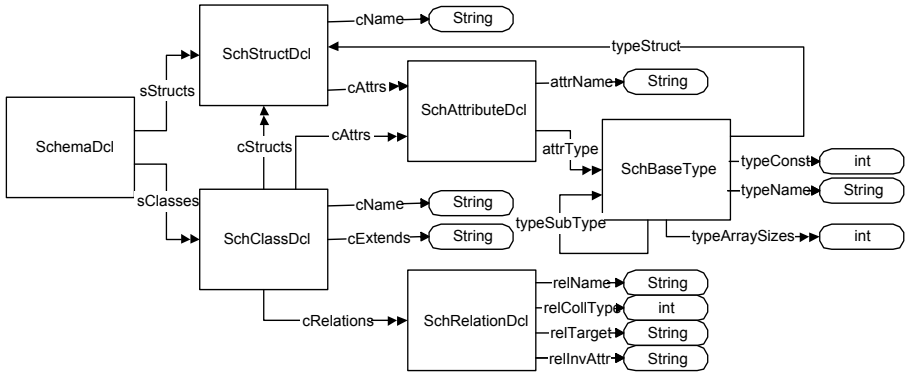
```
try {
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       SchemaDcl     schema =
        new SchemaDcl("jdbc:odbc:ODESTOR", "userID",
       "passwd", schID);
} catch(ClassNotFoundException e){
       // ERROR
} catch(SQLException e){
       // ERROR
}
```


Или:

```
try {
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       Connection conn = DriverManager.getConnection(
                     "jdbc:odbc:ODESTOR", "userID",
              "passwd");
       SchemaDcl     schema = new SchemaDcl(conn, schID);
} catch(ClassNotFoundException e){
       // ERROR
} catch(SQLException eSql){
       // ERROR
}
```

Созданный объект используется как классами самой системы ODESTOR, так и прикладными классами для доступа к метаданным приложения на основе навигации по атрибутам. Структура и связи данных классов представлены на рисунке:

Эти метаданные сохраняются в РСУБД в виде системных таблиц с префиксом Sch. Такие таблицы формируются программой Map Builder при первичной обработке информации об объектной схеме с помощью прямого соединения с базой данных.

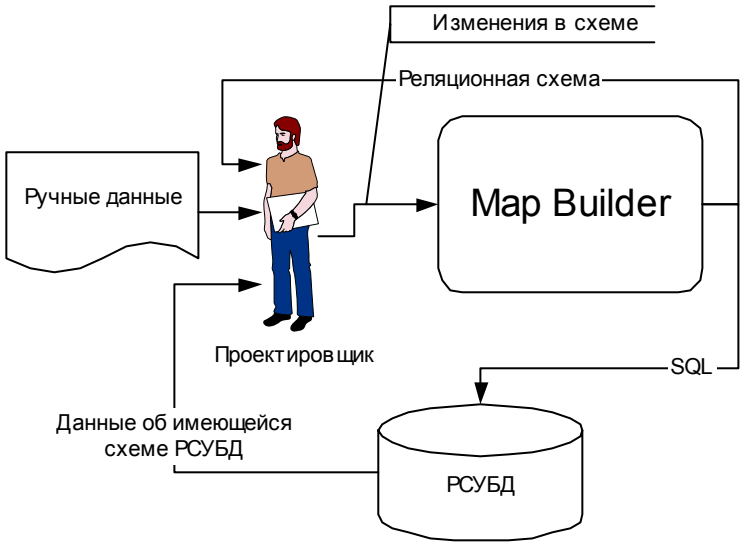


3.4 Таблицы для хранения данных объектов

При решении задачи прямого проектирования программа Map Builder генерирует объектно-реляционное отображение на основании собственных правил, обеспечивающих максимальную производительность прикладных программ. Результатом такого отображения является набор SQL файлов, в которых содержатся описательные операторы DDL. Используя эти файлы, можно с помощью средств РСУБД (например SQL*Plus для Oracle) создать все необходимые таблицы для хранения данных прикладных объектов.

3. Создание реляционной модели

Следующим этапом в работе по созданию приложения пользователя при решении задачи обратного проектирования является создание реляционной схемы, в которой должны храниться данные объектов. Пользователь должен описать таблицы в существующей реляционной базе данных, чтобы система могла построить соответствующий программный код для трансляции объектных данных в реляционные поля. Схема работы с данным модулем выглядит следующим образом:



Существует два основных способа задать данные о реляционной схеме – вручную или автоматически из РСУБД. Также можно применять комбинации этих способов.

3.1 Описание вручную

При отсутствии непосредственного доступа к работающей РСУБД реляционную схему можно описать полностью вручную. Для этого в системе

предоставляется интерфейс, аналогичный визуальным средствам описания таблиц РСУБД (Enterprise Manager для Oracle, Sybase Central для Sybase SQL Anywhere). В нем пользователь задает имя для каждой таблицы и все ее столбцы с указанием типов данных. Также задается информация о первичных ключах. На основании этих данных система может сформировать файлы на языке SQL, в которых содержатся все необходимые операторы для создания описанных таблиц в конкретной базе данных.

Данный способ можно использовать также для собственного описания реляционной схемы при решении задачи прямого проектирования, когда программист сам хочет описать структуру реляционных таблиц для хранения объектных данных, то есть когда его не устраивает автоматическое отображение, генерируемое программой Map Builder.

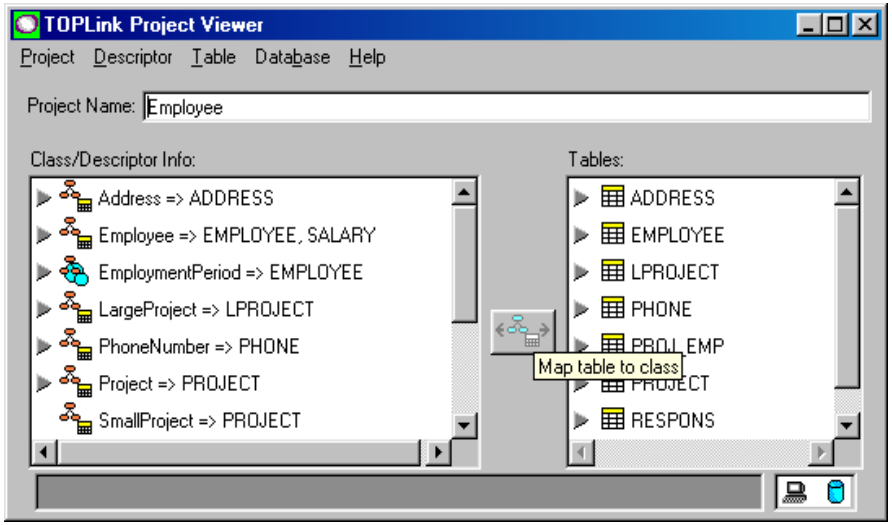
3.2 Автоматическое создание из существующей базы

Для первоначального построения реляционной схемы можно использовать автоматическую загрузку данных непосредственно из работающей РСУБД. Для этого программист задает источник данных, то есть строку соединения и, возможно, некоторые параметры фильтра. После этого система связывается с указанной РСУБД и извлекает существующую схему таблиц. Пользователь имеет возможность отредактировать эту схему с помощью той же программы, что и при ручном вводе.

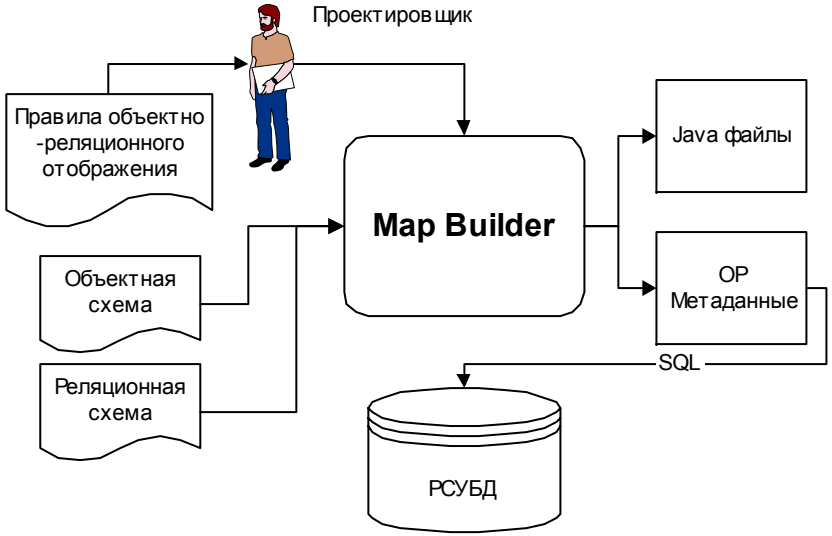
На основании созданной схемы можно сформировать файлы SQL, с помощью которых можно внести все необходимые изменения в РСУБД, в частности, добавить в таблицы столбцы для хранения объектных идентификаторов. При выполнении этой процедуры возникает отдельная задача заполнения столбцов значениями, уже существующими в базе данных. Эта проблема решается с помощью специальной полуавтоматической процедуры.

4. Описание объектно-реляционного отображения

Итак, после завершения предыдущих этапов при решении задачи обратного проектирования в системе формируются две схемы данных: объектная и реляционная. С одной стороны, формально описана внутренняя структура и взаимосвязи прикладных объектов, и, с другой стороны, описаны реляционные таблицы с указанием столбцов, в которых должны храниться эти объектные данные. Теперь нужно описать *объектно-реляционное отображение* между этими схемами. Загруженные данные отображаются в главном окне программы отображения модуля Map Builder подобно окну, используемому в продукте TopLink for Java:



Далее программист последовательно выбирает класс в левом списке и таблицу (таблицы) в правом. Для каждой такой комбинации открывается окно, где уже для каждого из атрибутов класса описываются правила, по которым записываются и считываются данные из РСУБД. Возможные способы отображения рассматриваются в [27]-[28]. Структура работы выглядит следующим образом:



Результатом работы по созданию объектно-реляционного отображения является набор специальных данных – *объектно-реляционные метаданные*. Эти данные должны в конечном итоге сохраняться в служебных таблицах РСУБД. Определения этих таблиц в базе данных могут быть сформированы непосредственно из программы Map Builder или записаны в отдельные файлы SQL для последующей обработки средствами РСУБД. После создания полного объектно-реляционного отображения система формирует SQL файлы для заполнения этих служебных таблиц конкретными данными. В дальнейшем информация об этих OP метаданных будет загружаться во время работы приложения, и использоваться как средствами сервиса долговременного хранения объектов, так и прикладными программистами для обеспечения возможности создавать оптимизированные программы.

Кроме OP метаданных, Map Builder формирует Java код, который собственно и будет обеспечивать объектно-реляционную трансляцию данных во время работы прикладных программ. После создания этого кода программист имеет возможность изменить некоторые функции для обеспечения оптимизации работы приложений.

5. Общая схема работы

Итак, сценарий и основные принципы работы с описываемой системой обеспечения объектно-реляционной трансляции ODESTOR выглядят следующим образом:

1. Нужно описать структуру классов приложения, в котором будет использоваться долговременное хранение объектов. Вообще это можно сделать тремя способами:
   - описать объектную схему в соответствии со стандартом ODMG на специальном декларативном языке ODL
   - описать схему, используя технологии объектного моделирования (UML)
   - описать собственно Java файлы приложения

Важное значение имеет здесь возможность подачи на вход системы набора файлов целевого языка программирования Java в качестве описания объектной схемы. Это обеспечивает возможность циклической работы с системой. Например, можно описать объектную схему на ODL, затем система сгенерирует на основе этой схемы файлы Java, их можно поправить и подать обратно на вход системы для учета внесенных изменений. В любом случае, результатом работы этого этапа является представленная во внутреннем формате информация об объектной схеме прикладной задачи. На основе этих данных в дальнейшем в РСУБД формируются специальные служебные

таблицы, куда записываются метаданные для последующего извлечения на этапе работы приложения.

2. Вторым этапом является создание реляционной модели. Эта модель формируется или вручную, или на основе имеющейся РСУБД, или на основе комбинации этих способов. Результатом этапа является информация о структуре реляционных таблиц, в которых в дальнейшем должны сохраняться данные объектов. Существует возможность сформировать SQL файлы для отражения этой информации в РСУБД (создание и модификация таблиц).
3. Наконец, третьим этапом является описание правил, по которым будут транслироваться объектные данные в реляционные таблицы и обратно. Это так называемая информация об объектно-реляционном отображении. Третий этап является ключевым в процессе описания данных приложения. Ему следует уделить особое внимание для обеспечения оптимальной производительности системы. Фактически, на основе описанной на данном этапе информации будет сформирован программный код, обеспечивающий непосредственную загрузку и запись данных в РСУБД.
4. После того, как на предыдущих трех этапах описана вся необходимая информация, система готова сформировать все необходимые компоненты для обеспечения дальнейшей работы приложения. Сценарий в этом случае такой:

   - Система генерирует файлы Java, в которых отражена структурная информация о классах (описаны все атрибуты, связи и методы) и реализован код объектно-реляционной трансляции.
   - Затем программист кодирует собственно семантику работы программы. При этом для хранения своих объектных данных он пользуется предоставленной библиотекой. То есть он имеет возможность писать код таким образом, как если бы при разработке использовалась чисто объектная база данных. При этом, как уже отмечалось, существует возможность циклического возврата к первым этапам и последовательной отладки кода.
   - После завершения кодирования Java файлы окончательно компилируются стандартным компилятором Java и полученные class файлы являются конечным программным продуктом, который можно использовать по назначению.

6. Заключение

Данная статья описывает существующую на данный момент схему работы с системой ODESTOR. Обсуждаются некоторые основные модули системы, отвечающие за начальную компиляцию, представление метаданных приложения и описание объектно-реляционного отображения. Первый модуль отвечает за компиляцию информации об объектной схеме задачи. В настоящее

время реализованы компиляторы ODL и Java. В дальнейшем планируется добавить входные форматы UML и XML. Второй модуль обеспечивает обработку данных о реляционной схеме. И, наконец, третий модуль предоставляет возможности по описанию правил объектно-реляционного отображения. Все эти модули объединены в единую интегрированную среду разработки Map Builder. Также разработаны отдельные модули проекта, выходящие за рамки данной статьи и относящиеся ко всему проекту в целом: реализация ODMG коллекций и OQL выборок над ними, диспетчер объектных транзакций и другие модули, обеспечивающие ODMG интерфейс ООБД. Детально проработана полная архитектура и алгоритмы взаимодействия модулей и действия всей системы в целом.

В дальнейшем планируется доработать систему для работы в распределенной среде с наличием большого количества пользователей с применением технологий CORBA и XML. Также планируется разработка улучшенных графических средств администрирования и работы с системой. Система ODESTOR является свободно распространяемой (freeware), и она обеспечивает полный интерфейс ООБД, специфицированный ODMG.

Литература

1. “Object-Oriented Design”. G. Booch. 1992, Behjamin/Cummings Publishing Company.
2. “Основы современных баз данных”. Кузнецов С.Д. Информационно-аналитические материалы Центра Информационных Технологий. www.citforum.ru
3. “Тенденции в мире систем управления базами данных”. Кузнецов С.Д. Информационно-аналитические материалы Центра Информационных Технологий. www.citforum.ru
4. “Object Database vs. Object-Relational Databases”. Steve McClure. IDC Bulletin. www.cai.com/products/jasmine/analyst/idc/14821E.htm
5. “Extending Relational Databases”. Martin Rennhackkamp. DBMS Magazine, December 1997
6. “Why Use an ODBMS?”. Poet Software White Paper. www.poet.com/products/oss/white_papers/rel_vs_obj/rel_vs_obj.html
7. “Объектно-ориентированные базы данных: среда разработки программ плюс хранилище объектов”. Андреев А.М., Березкин Д.В., Кантонистов Ю.А. www.inteltec.ru/publish/russian/articles/objtech/oodbms_o.htm
8. “Манифест систем объектно-ориентированных баз данных”. М. Аткинсон, Ф. Бансилон, Д. ДеВитт, К. Диттрих, Д. Майер, С. Здоник. Системы Управления Базами Данных, # 4/95, стр. 142-155.
9. .“The Object Database Standard: ODMG 2.0”. R.G.G. Cattell, Douglas K. Barry, 1997. www.odmg.org
10. .“The Object Data Standard: ODMG 3.0”. R.G.G. Cattell, Douglas K. Barry, 2000. www.odmg.org
11. “Mapping Objects To Relational Databases”. Scott W. Ambler. AmbySoft White Paper. http://www.ambysoft.com/mappingObjects.pdf
12. “Object Identity”. Khoshafian S. , Copeland G. OOPSLA Conference, 1986
13. “A rigorous model of object reference, identity and existence”. Kent W. JOOP June, 1991
14. “Storage management for persistent complex objects”. Khoshafian, Franklin, Carey. Information Systems, vol.15, no.3, p. 303-320, 1990
15. “TopLink for Java Documentation”. Object People. www.objectpeople.com/toplink/java/java.htm
16. “Java 2 Platform, Standard Edition Documentation”. java.sun.com/docs/index.html
17. “Object Relational Mapping Strategies”. Visual Business Sight Framework documentation. Mapping tool guide. www.objectmatter.com
18. “Foundations of Object Relational Mapping”. Mark L. Fussell. www.chimu.com

19. “Object-relational Access Layers – a Roadmap, Missing Links and More Patterns”. Wolfgang Keller. EPLoP’98.
20. “Patterns for Object/Relational Access Layers”. Wolfgang Keller. www.objectarchitects.de/ObjectArchitects/orpatterns/index.htm
21. “Integrating Objects with RDBMs”. GemStone White Paper. www.gemstone.com/products/s/papers_integrate.html
22. “Crossing Chasms: A Pattern Language for Object-RDBMS Integration (The Static Patterns)”. Kyle Brown and Bruce G. Whitenack. www.ksccary.com/Articles/ObjectRDBMSPattern/ObjectRDBMSPattern.htm
23. “A Pattern Language for Relational Databases and Smalltalk”. Kyle Brown and Bruce Whitenack. www.ksccary.com/Articles/PatternLangForRelationalDB/PatternLangForRelationalDB.htm
24. “Baker’s dozen”. Tim Ottinger. www.oma.com/ottinger/BakersDozen.html
25. “Cetus Links on Objects and Components: Object Relational”. WWW links collection. http://www.cetus-links.org/oo_db_systems_3.html
26. “Architecting Object Applications for High Performance with Relational Databases”. S. Agarwal, C. Keene, A.M. Keller. Persistence Software White Paper.
27. “Проблемы организации объектно-ориентированного доступа к реляционным базам данных”. К.В. Антипин, В.В. Рубанов. Труды Института Системного Программирования, вып. 2. 2000 г.
28. “Объектно-ориентированное окружение, обеспечивающее доступ к реляционным СУБД”. В.П. Иванников, С.С. Гайсарян, К.В. Антипин, В.В. Рубанов. Труды Института Системного Программирования, том. 2. 2000 г.