ИНСТРУМЕНТАЛЬНОЕ СРЕДСТВО ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ ДЛЯ ПРОЕКТИРОВАНИЯ КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Э. С. Воронов

Институт кибернетики Национального исследовательского Томского политехнического университета, 634034, Томск, Россия

УДК 004.4.22:004.4.242

Рассмотрены традиционные методы построения архитектуры корпоративных информационных систем. Выявлены основные недостатки существующих методов. Предложена архитектура на основе шаблонов проектирования "единица работы" и "хранилище". Рассмотрен результат создания инструментального средства, автоматизирующего использование предложенной технологии, для интегрированной среды разработки.

Ключевые слова: объектно-реляционное отображение, шаблон "единица работы", шаблон "хранилище", инструментальные средства, информационные системы.

Key words: object-relational mapping, unit of work pattern, repository pattern, work bench, information system.

Современный рынок накладывает жесткие финансовые и временные ограничения на создание корпоративных информационных систем, связанных в первую очередь с высокой конкуренцией в области разработки программного обеспечения. Создание качественного, масштабируемого и гибкого продукта с учетом ограничений невозможно без использования современных технологий и инструментальных средств, облегчающих, систематизирующих и упорядочивающих процесс разработки программного обеспечения.

Неотъемлемой частью корпоративных информационных систем является база данных (БД). Традиционный способ работы с БД – использование в приложении поставщиков данных для извлечения и модификации информации [1]. На рис. 1 представлена модель приложения, реализующего традиционный доступ к БД. Модель приложения состоит из слоев *View* (представление), *Model* (модель) и *Data Model* (модель данных). Слой модели данных содержит методы обращения к БД, состоящие из SQL-запросов, и предоставляет результирующие наборы данных (*Data Set*). Слой модели определяет бизнес-логику приложения. Слой представления является отображением данных пользователю. Для каждой отдельной подзадачи выделяется отдельная цепочка модель данных – модель – представление. На рис. 1 видно, что традиционный подход имеет несколько недостатков: 1) создание всех SQL-запросов к БД вручную и как следствие

Работа выполнялась по тематике госбюджетной НИР в рамках государственного задания "Наука".

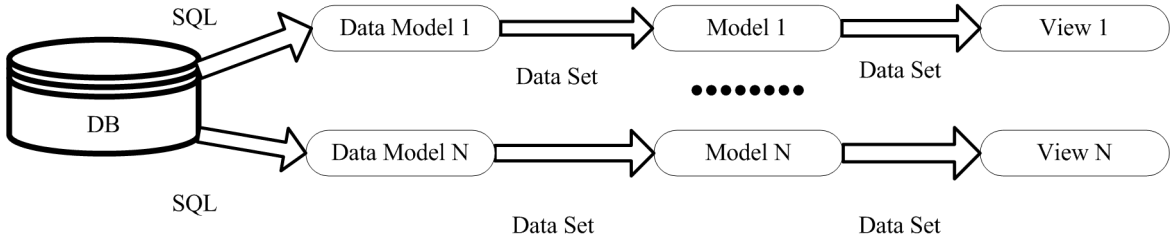

Рис. 1. Модель приложения, реализующего традиционный доступ к БД

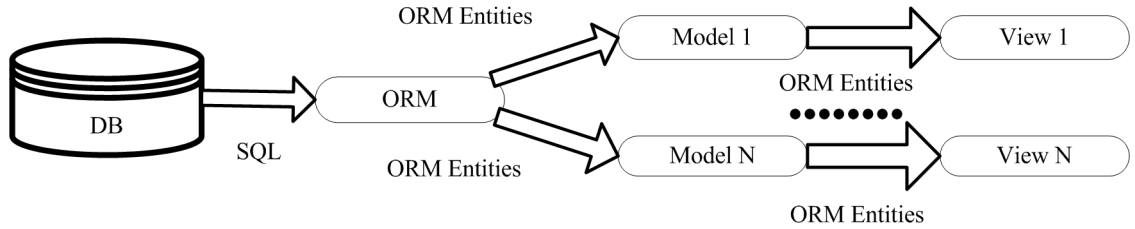

Рис. 2. Модель приложения, реализующего традиционный доступ к БД на основе ORM

высокие временные затраты и вероятность ошибки; 2) зависимость слоев представления и модели от наборов данных, предоставляемых моделью данных, что обусловливает невозможность применения модульного тестирования.

Альтернативным подходом по отношению к традиционному является использование технологии object-relational mapping (ORM) − объектно-реляционного отображения – технологии связывания базы данных с концепциями объектно-ориентированного программирования. С точки зрения программного обеспечения ORM – механизм преобразования объектов и методов приложения в данные и запросы к БД [2]. В то же время с точки зрения разработчика ORM – инструмент для отображения модели БД в объектно-ориентированную модель языка программирования. Использование ORM избавляет от необходимости написания вручную запросов к БД на извлечение и модификацию данных, что существенно сокращает временные затраты разработки программного обеспечения, а также уменьшает вероятность ошибки. На рис. 2 приведена модель приложения, реализующая доступ к БД на основе ORM.

Слой ORM содержит определение сущностей отображения модели БД и определения data context − аккумулирующей сущности, через которую происходит обращение к БД. Следует отметить, что современные ORM выполняют генерацию этих сущностей автоматически, позволяя разработчикам настраивать их поведение непосредственно в интегрированной среде разработки.

В отличие от традиционного подхода бизнес-логика приложения не зависит от специфичных наборов данных модели данных, а использует вместо них сущности ORM (ORM Entities) – сущности отображения БД. Таким образом, архитектура приложения становится более разделенной и с более слабой зависимостью между слоями, что позволяет применять модульное тестирование.

Однако применение одного ORM совместно с различными базами данных как по типу, так и по структуре невозможно. Таким образом, масштабируемость такой архитектуры остается крайне низкой, так как модель и представление также транзитивно зависимы от типа и структуры БД.

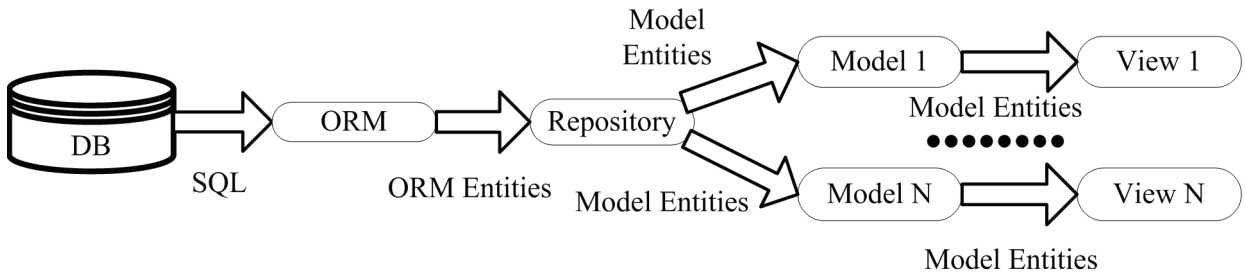

Рис. 3. Модель приложения, реализующего технологию репозитория

Для повышения масштабируемости и редукции связности слоев поставлена задача разработки типовой архитектуры корпоративных информационных систем, которая удовлетворяет следующим требованиям:

− наличие слоя доступа к данным, реализованного на основе ORM*;*

− модульная независимость слоев модели и представления от ORM-сущностей;

− наличие инструментальных средств, автоматизирующих применение технологии.

Разработанная типовая архитектура основана на применении шаблона проектирования "единица работы" (unit of work) [3], в рамках которого модель приложения обращается к данным или модифицирует их в рамках отдельных транзакций. Для достижения модульной независимости разработана технология репозитория (Repository), позволяющая проводить преобразование между сущностями ORM в сущности модели (Model Entities). Модель приложения с предложенной технологией репозитория представлена на рис. 3.

Слой репозитория реализует преобразование между сущностями ORM и сущностями модели. Таким образом, слой репозитория полностью обеспечивает независимость слоев модели и представления от конкретного ORM и от типа и структуры БД. В то же время сущности модели обладают набором функциональности, обеспечивающим их удобное использование при создании бизнес-логики приложения:

− наличие ссылочных значений на родительские сущности, соответствующих связям по внешним ключам модели БД;

− возможность определения ограничений целостности сущностей модели (обязательность полей, ограничение значений и т. д.) декларативным методом.

Апробация технологии репозитория проводилась на платформе .Net Framework с применением интегрированной среды разработки Microsoft Visual Studio 2010 и ORM Microsoft Linq To Sql. На рис. 4 представлена диаграмма классов разработанной технологии.

Класс Entity является базовым классом для всех сущностей репозитория и реализует интерфейсы IDataErrorInfo и INotifyPropertyChanged для поддержки валидации данных и оповещения об изменениях. Класс EntityWithPK, наследующий класс Entity, обеспечивает функциональность для сущностей с суррогатным первичным ключом. Класс RepositoryDMBase, реализующий интерфейс IRepositoryDM, является точкой доступа к данным и обеспечивает подключение к источнику данных (ORM). Абстрактный класс RepositoryBase, реализующий интерфейс IRepository, обеспечивает функциональность преобразования между сущностями ORM и сущностями репозитория.

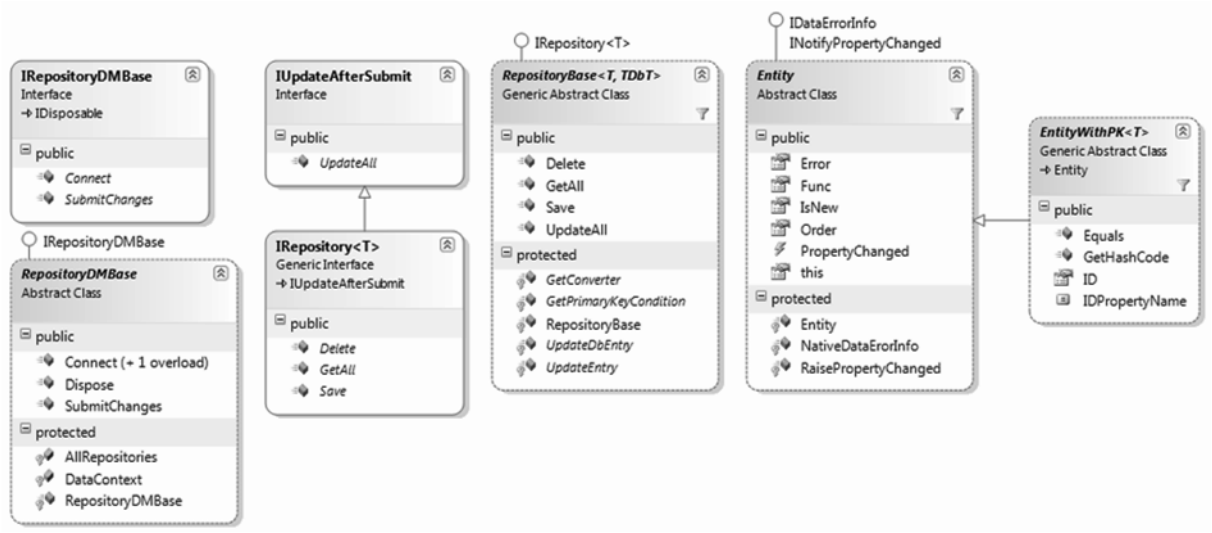

Рис. 4. Диаграмма классов

Рис. 5. Схема инструментального средства, интегрированного в среду разработки

Для автоматизации процесса применения технологии разработано инструментальное средство на языке программирования C#, интегрируемое в среду разработки. Схема инструментального средства представлена на рис. 5.

Инструментальное средство состоит из интегрированного в среду разработки редактора (Editor), позволяющего изменять настройки отображения сущностей ORM в сущности модели. Для упрощения создания настроек использована объектная модель автоматизации Visual Studio (DTE) [4], позволяющая получить доступ к элементам среды разработки (проекты, файлы, типы) как к объектам. Настройка отображения сохраняется в файл в регулярном формате (XML).

На основе созданного файла настроек генерируется код слоя репозитория с сущностями модели. Генерацию автоматически осуществляет зарегистрированная в системе COM-сборка генератора (Code Generator). В основе генератора лежит объектная модель Code DOM [5], которая позволяет описывать модель исходного кода предоставляемыми ею типами (класс, пространство имен, свойство, метод и т. д.). Полученную модель можно преобразовывать как в исходный код, так и в скомпилированную библиотеку.

Описанные технология и инструментальное средство использованы при разработке корпоративной информационной системы "АИС склада метанола", автоматизирующей процесс отгрузки товарного метанола ООО "Сибметахим".

Таким образом, предложенная архитектура репозитория является развитием архитектуры на основе ORM и полностью соответствует парадигме объектно-ориентированного программирования. Инструментальное средство, встраиваемое в интегрированную среду разработки в сочетании с библиотеками базовых классов, образует фреймворк, который значительно упрощает проектирование корпоративных информационных систем.

Список литературы

1. Фаулер М. Шаблоны корпоративных приложений / Пер. с англ. М.: Издат. дом "Вильямс", 2011.

2. Mapping objects to relational databases: O/R mapping in detail. 2012. [Electron. resource]. http://www.agiledata.org/essays/mappingObjects.html.

3. Фримен Э. Паттерны проектирования / Э. Фримен, К. Сьерра, Б. Бейтс. СПб.: Питер, 2012.

4. Модели объектов автоматизации. MSDN. 2012. [Electron. resource]. http://msdn.microsoft.com/ru-ru/library/za2b25t3.aspx.

5. Использование CodeDOM. MSDN. 2012. [Electron. resource]. http://msdn.microsoft.com/ru-ru/library/y2k85ax6.aspx.

Воронов Эдуард Сергеевич – асп. Института кибернетики Томского политехнического университета; тел.: (382-2) 70-17-77; e-mail: voronoves@tpu.ru

Дата поступления − 15.09.12 г.