УДК 51:681.3

И.А. Лихацкий

ОБ ОДНОЙ МЕТОДИКЕ ФОРМИРОВАНИЯ ОБЪЕКТНОГО ПРЕДСТАВЛЕНИЯ РЕЛЯЦИОННЫХ ДАННЫХ

Предложена методика формирования объектного представления реляционных данных, направленная на достижение наибольшей производительности генерируемого кода и труда программиста. Методика использует концепцию "трех проекций" представления реляционных данных в объектном коде, на основе которой описывается модель поведения реляционных объектов в объектном коде.

Введение

В настоящее время в мире хранения данных доминируют реляционные системы управления базами данных (РСУБД) [1], тогда как для обработки данных все чаще используются объектные языки программирования (ОЯП) [2]. Доля рынка РСУБД составляет 80% [3] от всего сегмента рынка СУБД, а доля рынка ОЯП составляет 50% от всего сегмента рынка языков программирования [4, 5].

При разработке информационных систем, использующих РСУБД, часто возникают проблемы в преобразовании данных из реляционного вида в объектный и наоборот. В качестве решения данной проблемы предлагается создать объектную форму представления реляционной БД, объекты которой отражают сущности предметной области и однозначно связаны с таблицами, представлениями и хранимыми процедурами БД.

В данной статье предложена методика формирования объектного представления реляционных данных, целью которой является достижение высокого уровня автоматизации процесса.

Актуальность использования таких систем обусловлена, с одной стороны наличием на рынке развитых объектно-ориентированных технологий, а с другой – доминированием реляционных баз данных в мире хранения данных. В результате возникает необходимость в построении системы, позволяющей осуществить взаимодействие между объектными языками программирования и реляционными базами данных.

Рассмотрим методику формирования объектного представления реляционной базы данных на примере системы кодогенерации C-Gen [6]. Данная система на входе получает структуру базы данных. На выходе, на стороне сервера БД система генерирует набор CRUD (Create Retrieve Update Delete) хранимых процедур, и программный код на выходном языке программирования.

1. Проекция реляционной модели данных в объектно-ориентированную

Предлагаемая объектная форма представления реляционной БД позволяет представить имеющиеся таблицы, представления и хранимые процедуры в виде иерархии объектов, при этом несущественные для пользователя данные скрыты, а объекты и атрибуты имеют названия, характерные для рассматриваемой предметной области.

Формирование объектного представления реляционной БД выполняется в автоматизированном режиме, на основе структуры БД.

Для построения объектной формы будут анализироваться такие объекты БД.

1.1. База данных (DataBase) – совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними. Определим, что на объектном уровне представления данных, БД это совокупность объектов (таблицы, представления) и методов (хранимых процедур) осуществляющих некоторые действия над объектами.

1.2. Таблица (Table) – таблица в реляционной базе данных (рис. 1).

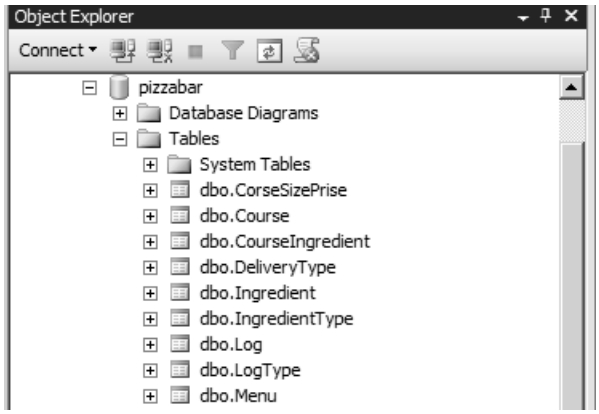

Рис. 1. Список таблиц БД

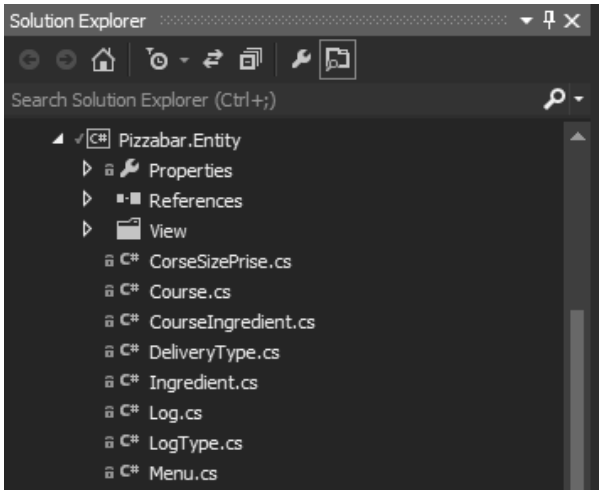

Рис. 2. Список объектов

Определим, что на объектном уровне представления данных, таблица это некоторый объект (рис. 2) определенного типа (строго типизированный), а результатом выборки данных из таблицы является либо один объект данного типа, либо список объектов данного типа.

1.3. Представление (View) – виртуальная таблица, представляющая собой поименованный запрос, который будет представлен как подзапрос при использовании представления. Определим, что на объектном уровне представления данных, представление также является некоторым объектом определенного типа (строго типизированный), а результатом выборки из представления является либо один объект данного типа, либо список объектов данного типа.

1.4. Хранимая процедура (Stored procedure) – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

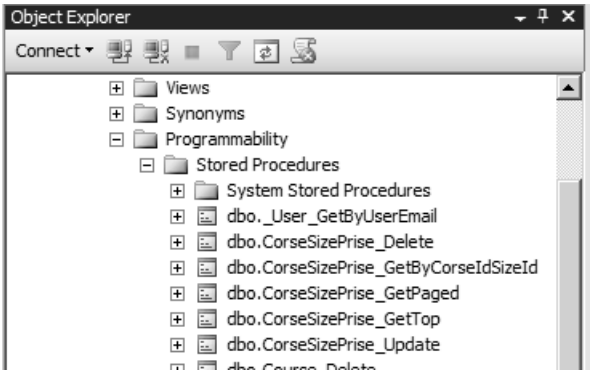

Рис. 3. Список хранимых процедур

Определим, что на объектном уровне представления данных, хранима процедура это метод, который выполняет некоторые действия над объектами (таблицы, представления). В зависимости от характера операции (вставка, выборка, удаление обновление), метод (хранимая процедура) может либо возвращать объект, либо возвращать список объектов, либо ничего не возвращать (рис. 4) . Также определим, что хранимая процедура обязательно относится к одному из объектов БД (таблица, представление). Под отношением понимается абстрактная связь с учетом которой определяется тип возвращаемых данных.

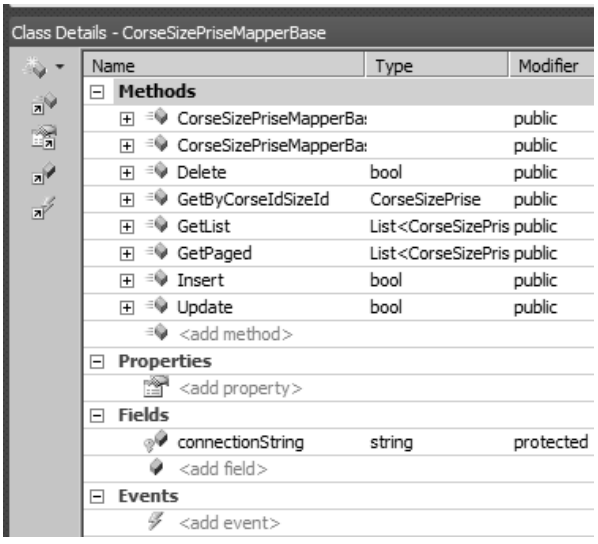

Рис. 4. Объектное представление хранимых процедур относящихся к таблице CorseSizePrice

Например, таблица CourseSizePrise (рис. 1) в объектном виде представлена в виде класс CourseSizePrise.cs (рис. 2), а хранимые процедуры (рис. 3), которые относятся к CourseSizePrise, в объектном виде, представлены в виде соответствующих методов (рис. 4.)

В рамках рассматриваемой методики формирования объектного представления, все хранимые процедуры базы данных делятся на два типа.

- **"Простые хранимые процедуры"**. Данный тип процедур создается для каждого объекта БД (таблицы, представления). Он включает в себя набор стандартных CRUD хранимых процедур, а также процедуру для постраничного отображения контента. Отметим, что на объектном уровне представления данных, метод, соответствующий данному типу хранимых процедур, всегда будет возвращать значения того типа объекта, к которому относится хранимая процедура. А так как мы определили в пункте 1.4, что хранимая процедура обязательно относится к какому-либо объекту БД, то и тип данных будет определен однозначно.

- **"Пользовательские хранимые процедуры"**. Данный тип процедур используется для решения нетривиальных задач по выборке данных, а также реализации выборки, которая не поддерживается стандартными хранимыми процедурами. "Пользовательские хранимые процедуры" не генерируются автоматически, а пишутся разработчиком вручную. На объектном уровне представления данных данный тип хранимых процедур, может возвращать значения типа отличного от типа объекта, к которому он относится (более детально данная особенность будет описана далее).

Так как основная цель данной методики формирования объектного представления реляционных данных, – достижение высокого уровня производительности информационной системы в целом и значительного уровня автоматизации, подход предполагает создание (генерацию) "простых хранимых" процедур автоматически, перед построением объектной модели реляционной структуры данных.

К "стандартным хранимым процедурам" относятся следующие хранимые процедуры:

- **Insert** – добавляет новую запись в таблицу (генерируется для таблиц);
- **Update** – обновляет существующую запись в таблице (генерируется для таблиц);
- **Delete** – удаляет запись из таблицы (генерируется для таблиц);
- **GetTop** – возвращает n-записей из таблицы (генерируется для таблиц и представлений);
- **GetByPrimaryId** – в качестве параметра данная хранимая процедура получает уникальный первичный ключ (либо составной первичный ключ, в зависимости от структуры таблицы) по которому осуществляется выборка. (генерируется для таблиц);
- **GetPaged** – данная хранимая процедура позволяет делать постраничную выборку контента длинных списков. (генерируется для таблиц и представлений). Хранимая процедура возвращает список элементов, которые должны быть отображены на соответствующей странице, а также общее количество элементов в списке.

Таким образом, "простые хранимые процедуры" обеспечивают разработчиков информационной системы средствами базового управления данными "из коробки".

2. Первая проекция. Таблицы реляционных БД в объектном виде

Исходя из определения, что на объектном уровне представления данных таблица – объект определенного типа, определим, что проекция таблицы в объектном коде представляется в виде класса сущности (Entity), тип которой соответствует типу таблицы.

Мы знаем, что работа с таблицами

реляционной БД и классами в объектно-ориентированном программировании (ООП) отличается существенно [7]. Например, в ООП уникальность проверяется по адресу объекта в памяти, на которую он ссылается, а в РСУБД уникальность записи можно определить по первичному ключу (либо совокупности ключей). Структура класса (является проекцией таблицы БД) объекта в объектном языке, спроектированна таким образом, что позволяет программисту работать с проекцией таблицы, как с обычным классом в ООП. Вся логика, которая осуществляет сопоставление класса объекта в ООП на его реляционное представление инкапсулирована в базовом классе. Таким образом, работа с классом объекта ни чем не отличается от работы с обычным классом.

Для того, чтобы экземпляр класса объекта мог быть однозначно сопоставлен с его реляционным представлением, необходимо точно идентифицировать, является ли данный экземпляр класса новой записью по отношению к его реляционному представлению или нет. Нам известно, уникальность экземпляра класса и записи в таблице идентифицируются по разному.

Предлагаемый алгоритм сопоставления класса в ООП к его реляционному представлению, основанный на инкапсуляции логики по обработке дополнительного поля первичного ключа в структуру класса. Явно программист не может получить доступ к данному полю или изменить его. Логика работы заключается в том, что если экземпляр класса является новой записью по отношению к своему реляционному представлению, то это поле будет установлено в null, и наоборот, если экземпляр класса отображает существующую запись в таблице, поле будет содержать уникальный идентификатор данной записи. Таким образом, неявно осуществляется сопоставление экземпляра класса к его реляционному представлению.

3. Вторая проекция. Представления в объектном виде

Как и таблицы, представления представляются в объектном коде в виде классов сущностей (Entity). Так как представление это виртуальная таблица, представляющая собой поименованный запрос, то над ним мы можем совершать только операции выборки, поэтому и на объектном уровне, классы сущностей представлений по своей структуре отличаются от классов сущностей таблиц.

Теперь мы можем отображать (представлять) объекты реляционной БД с помощью первой и второй проекции и создавать/редактировать записи с помощью первой проекции. Для обеспечения взаимодействия (выборка, вставка, удаление, обновление) экземпляров касса с реляционным представлением необходима третья проекция – проекция на хранимые процедуры.

4. Третья проекция. Хранимые процедуры как методы в объектном виде

4.1. "Простые хранимые процедуры". Исходя из определения, что данный тип хранимых процедур генерируется автоматически и сопоставляется каждому объекту БД, с помощью "простых хранимых процедур" осуществляются базовые операции с данными хранящимися в таблицах. На объектном уровне представления данных, каждой "простой хранимой процедуре" соответствует метод в объектном коде.

Существует ограничение при использовании методов, которые являются проекциями на хранимые процедуры *Insert* и *Update*. Программист не может вызвать данные методы напрямую, а должен делать это через метод *Save*. Поскольку экземпляр класс, являющийся проекцией на таблицу инкапсулирует логику по установлению, является ли данный экземпляр класса новой записью по отношению к своему реляционному представлению, то и решение о том, какой метод должен быть вызван в том или ином случае скрывается от разработчика. С одной стороны это предотвращает возможность возникновения ошибки в связи с вызовом неправиль-

ного метода сохранения, с другой стороны, делает работу с объектами БД интуитивно понятной.

Каждый метод принимает тот же набор параметров, что и соответствующая ему хранимая процедура, и возвращает объект, либо список объектов соответствующего типа сущности к которой относится хранима процедура. Таким образом взаимодействие объектного кода с БД происходит через вызовы методов, оперирующими строго типизированными объектами.

4.2. "Пользовательские хранимые процедуры". Пользовательские хранимые процедуры отличаются от простых тем, что тип результата возвращаемого хранимой процедурой может отличного от типа объекта к которому относится хранимая процедура.

Например, на рис. 5 показан листинг пользовательской хранимой процедуры GetTopCourses. Хранимая процедура относится к объекту Course, однако результат, возвращаемый процедурой не является объектом типа Course (рис. 6). Как видно из листинга, хранимая процедура состоит из двух запросов, результат выполнения которых, должен быть представлен в виде класса в объектном виде.

Следовательно был предложен алгоритм, позволяющий построить проекцию объекта на возвращаемый тип хранимой процедурой. Поскольку хранимая процедура может возвращать несколько результатов одновременно был предложен алгоритм ***"рекурсивного выведения типа объекта".***

1. Выполнить запрос.

2. Если ответ содержит массив результирующих наборов данных выполнить шаг 3 иначе метод возвращает void.

3. Пока есть результирующий набор, на основе метаданных полей описываем класс.

4. На основе описанного класса (классов) выводим тип объекта.

```
ALTER PROCEDURE
    [dbo].[_Course_GetTopCourses]
    @CourseId smallint
as
    set nocount on;

    select top (5)
         c.CourseId,
         c.CourseName
    from [Course] c with(nolock)
    where c.CourseId <> @CourseId
        and StatusRecordId =
            dbo.fnStatusRecordRecord()
    order by c.SortOrder, c.MenuId;

    select
        CourseId,
        CourseName,
        MenuId,
        StatusRecordId,
        SortOrder,
        MultimediaId
    from dbo.Course c with(nolock)
    where CourseId = @CourseId
```

Рис. 5. Пользовательская хранимая процедура GetTopCourses

Results | Messages

	CourseId	CourseName
1	12	Крылышки
2	18	Тортильони в томатном соусе
3	23	Салат "Фруктовый"
4	2	Пицца Грибная с ветчиной
5	13	Крылышки острые

	CourseId	CourseName	Me...	Sta...	Sor...	Mu
1	1	Пицца Маргарита	1	1	1	5

Рис. 6. Результат выполнения процедуры GetTopCourses

В результате, метод, который является проекцией на пользовательские хранимые процедуры, в качестве результирующего набора возвращает строго типизированный объект (рис 7) состоящий из двух результирующих наборов, представленных в виде списков классов Course и ResultSet1.

Class Details - GetTopCoursesItem

Name	Type	Modifier
Methods		
<add method>		
Properties		
ResultSet2	List<Course>	public
ReusltSet1	List<ResultSet1>	public
<add property>		
Fields		
<add field>		
Events		
<add event>		

Рис. 7. Класс сгенерированный для хранимой процедуры GetTopCourse

5. Проекция связей между таблицами

Одна из самых непростых проекций, это проекция связей между таблицами в ООП.

Данный подход предполагает наличие следующих двух способов отображения связей. Все связи, между таблицами которые необходимо отобразить в объектном виде можно представить одним из следующий способов.

- ***отображение связи через представления.*** Представление будет отображать связь между таблицами и/или представлениями. На объектном уровне представления данных будет сгенерирован соответствующий класс, который является проекцией на данное представление, а также два стандартных метода (GetTop, GetPaged) для взаимодействия с ним. Таким образом, пользователь сможет отобразить конкретную связь в виде представления, к которому можно получить доступ в объектном коде через сгенерированный класс;

- ***отображение связей через "пользовательские хранимые процедуры"***. "Пользовательская хранимая процедура", которая также как и представление отображает связь между таблицами и/или представлениями, но в отличии от представления, для хранимой процедуры, на объектом уровне будет сгенерирован объект и соответствующий метод доступа.

Рассмотрим пример проекции связей между в объектный код (рис. 8), на примере связи многие-ко-многим между таблицами Cource и Ingredient. Допустим, нам необходимо вывести все продукты из таблицы Ingredient, которые входят в блюда из таблицы Course (рис. 9).

Запрос, который необходимо выполнить, чтобы получить результат (рис. 9), как уже было сказано выше, может быть отображен в виде хранимой процедуры либо же представления. Разница, заключается в том, что на объектном уровне представления, для представления будет построен объект, который будет в дальнейшем являться типом возвращаемых данных для хранимых процедур связанным с данным представлением и два

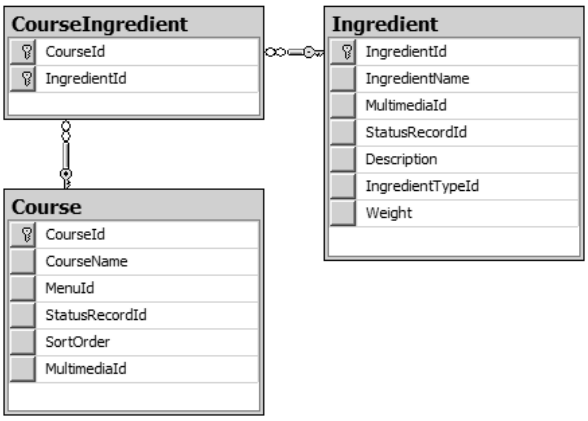

Рис. 8. Связь многие-ко-многим

Results | Messages

	CourseName	IngredientName
1	Пицца Маргарита	моцарелла
2	Пицца Маргарита	шампиньоны
3	Пицца Маргарита	курица
4	Пицца Грибная с ветчиной	петрушка
5	Пицца Грибная с ветчиной	шампиньоны
6	Пицца Грибная с ветчиной	бекон
7	Пицца Грибная с ветчиной	ветчина

Рис. 9. Результат

метода для доступа к нему (см. пункт 3), а для пользовательской хранимой процедуры – класс, который будет относиться только к данной хранимой процедуре (см. пункт 4.2). Следовательно, если в результате, необходимо построить несколько запросов, возвращающих один и тот же объект, но удовлетворяющих разным условиям, целесообразней использовать для этих целей представления. Иначе, для разовой выборки – "пользовательские хранимые процедуры".

Таким образом, осуществляется неявная проекция связей таблиц в ООП. Одним из преимуществ данного подхода является то, что в отличии от использования технологии динамических запросов (таких как LinQ), данный подход может быть реализован на любом объектно ориентированном языке программирования, в не зависимости от того, поддерживает ли он динамические запросы или нет.

Выводы

В данной работе предложена методика формирования объектного представления реляционных данных. Описаны ре-

ляционные объекты (таблицы, представления, хранимые процедуры), а также определены свойства которыми они обладают на объектном уровне представления данных. Введены понятия: "отношения" хранимых процедур с таблицами и представлениями, "пользовательские хранимые процедуры", "простые хранимые процедуры". Предложена концепция "трех проекций", на основе которой строится объектная модель реляционных данных:

- таблицы реляционной БД (первая проекция);
- представления реляционной БД (вторая проекция);
- хранимые процедуры БД (третья проекция).

Также предложен алгоритм сопоставления экземпляров класса к записям в таблице, позволяющий сопоставлять реляционное и объектное представление данных.

В рассмотренной методике, предлагается механизм отображения реляционных связей между объектами БД, а также два подхода по отображению таких связей в объектном виде:

- отображение связи через представления;
- отображение связей через "пользовательские хранимые процедуры".

Данный подход позволяет работать с данными в форме специфических для домена объектов и свойств, без необходимости обращаться к базовым таблицам и столбцам базы данных, где хранятся эти данные. Данный подход дает разработчикам возможность работать с данными на более высоком уровне абстракции.

Также следует отметить, что данная методика предполагает формирование строго типизированных объектов, через которые осуществляется взаимодействие с реляционной БД.

Практически, рассмотренная методика, была реализована в системе кодогенерации C-Gen [6].

1. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2008– 987с.
2. *Роганов Е.А.* Основы информатики и программирования. – М.: 2001. – 587 с.
3. http://www.tadviser.ru/index.ph /Статья:СУБД_(мировой_рынок)СУБД (мировой рынок)
4. *http*://dou.ua/lenta/articles/programming-languages-rating-2011-07 Рейтинг языков программирования.
5. *http*://habrahabr.ru/post/137833/Индекс популярности языков программирования за февраль 2012.
6. *Лихацкий И.А.* Средства кодогенерации для взаимодействия с базой данных через объекты // Проблеми програмування. – 2012. – № 2–3. – 380–387.
7. *Bauer C., King G.* Java Persistence with Hibernate. – New York: Manning, 2007. – 876 p.
8. *Кунгурцев А.Б., Завалин А.А.* Методика формирования объектного представления реляционной базы данных // Труды Одесского политехнического института, 2004.

Получено 04.01.2013

Об авторе:

Лихацкий Игорь Александрович,
младший научный сотрудник.

***Место работы автора*:**

Институт программных систем
НАН Украины,
03680, Киев-187,
Проспект Академика Глушкова, 40.
Тел.:+38(095)361 0330.
E-mail: igor_md@ukr.net