УДК 004.622:004.912

ORM-РЕШЕНИЕ НА БАЗЕ POLAR

[1]Платонов Ю.Г., [2]Бычков Д.А.

[1]ФГБУН «Институт систем информатики им. А.П. Ершова» Сибирского отделения Российской академии наук, Новосибирск, e-mail: y.platonov@mail.ru;

[2]Новосибирский государственный университет, Новосибирск

В статье описывается и обосновывается оригинальный подход, позволяющий строить ORM системы для работы с так называемыми нереляционными (NoSQL) базами данных. В качестве тестовой системы для построения ORM-решения используется специализированная система Polar – оригинальная разработка ИСИ СО РАН, создававшаяся как некоторая система хранения и обработки информации с моделью RDF в качестве состоявшейся области применения. Таким образом, одновременно расширяются возможности использования Polar – в рамках описываемой технологии Polar может быть использована в качестве СУБД. При этом сравнительный анализ скорости поиска, загрузки данных и обновления данных для ORM Polar и наиболее популярных на данный момент систем – конкурентов системы ORM Polar – позволяет сделать вывод о целесообразности и перспективности использования ORM Polar.

Ключевые слова: ORM, Polar, NoSQL базы данных, Entity Framework, модель «Code First»

ORM-SOLUTION FOR POLAR

[1]Platonov Yu.G., [2]Bychkov D.A.

[1]A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, e-mail: y.platonov@mail.ru;

[2]Novosibirsk State University, Novosibirsk

The original approach which allows building ORM systems to work with not-relational (NoSQL) databases is described and locates in article. As a test system for creation of the ORM solution the specialized Polar system – original development of ISI of the Siberian Branch of the Russian Academy of Science, is used. Polar was implemented as a system for storing and processing an information and allows applying RDF as a standard area of usage. Thus, possibilities of use of Polar at the same time extend – within the described Polar technology can be used as DBMS. The comparative analysis of speed of search, loading of data and updating of data for ORM Polar and systems most popular at the moment – competitors of ORM Polar system – allows to draw a conclusion on expediency and prospects of use of ORM Polar.

Keywords: ORM, Polar, NoSQL database, Entity Framework, «Code First» model

В настоящей статье предложен способ решения задачи построения ORM системы для работы с так называемыми нереляционными (NoSQL) базами данных (БД).

Под ORM (Object Relational Mapping) [6] обычно понимается технология программирования, дающая возможность связывать базу данных с её объектным представлением в памяти, то есть, иначе говоря, трансляция атрибутов баз данных в объектные типы данных и обратно.

Отказавшись от традиционного подхода, где ORM-решения применяются для работы с универсальными БД, авторы использовали в качестве системы хранения информации специализированную систему Polar – оригинальную разработку лаборатории САПР и А СБИС ИСИ СО РАН (см. раздел 1).

Система Polar позволяет быстро сохранять различные данные за счет использования системного кэша (cache) компьютера. Заметим, что здесь и далее под системным кэшем мы будем понимать некое динамическое количество оперативной памяти и дискового пространства, выделяемое системой для виртуализации памяти в части доступа к файловым страницам. Система Polar хорошо зарекомендовала себя как система хранения данных в RDF-базах данных. Поэтому авторы сочли перспективным расширить систему хранения данных и сделать к ней надстройку, позволяющую реализовать ORM-технологию с использованием Polar (см. разделы 3, 4).

В свою очередь, реализация ORM-технологии для Polar существенно расширяет практические возможности использования этой системы (см. раздел 2).

1. Polar как система хранения данных

Система хранения данных Polar [1] представляет собой некую коллекцию данных, сохраненных на жестком носителе. При этом чтение и запись данных осуществляется при помощи прямого обращения к носителю данных.

Само существование и конкурентоспособность данной системы стали возможными, благодаря созданию современных

поколений жестких носителей, а также возможностям операционной системы – а именно, системному кэшу.

При этом важно, что структура данных (ячейка в терминах Polar) может иметь достаточно произвольный характер и не является, в общем случае, таблицей.

Например, структура данных может представлять собой коллекцию однотипных записей, каждую из которых можно проассоциировать со строкой таблицы. Тогда запись будет представлять собой аналог строки таблицы, а вся коллекция – таблицу. Также структура может быть представлена и как совокупность записей разного типа – например, при построении RDF-графа [2].

Polar разрабатывалась как некоторая система хранения и обработки информации. При этом модель RDF является для Polar состоявшейся областью применения.

Для эффективного поиска триплетов [7] был разработан авторский механизм индексирования на основе бинарного поиска. Этот механизм прошел проверку эффективности следующим образом.

RDF-модель описания данных была реализована на системах Polar, MS SQL и MySQL. Реализация представляла собой коллекцию RDF-триплетов, хранящихся в одной таблице. Был создан индекс по столбцу таблицы и сгенерированы тестовые RDF данные.

Для заранее выбранного произвольным образом объекта выполнялся поиск всех соответствующих ему триплетов, и время этого поиска для каждой из систем служило характеристикой ее эффективности.

Измерения, выполненные для разного числа триплетов в таблице, позволили оценить производительность систем в зависимости от количества записей. Результаты сравнительных тестов по поиску случайно выбранного объекта приведены в табл. 1. В ней отражено время в миллисекундах, затраченное на поиск случайного объекта в таблице с различным числом триплетов. Данные демонстрируют, что Polar реализация RDF-модели эффективно работает вплоть до 10 миллионов триплетов в коллекции.

2. Актуальность построения ORM-решения на базе системы Polar

Традиционно ORM-решения разрабатываются для реляционных баз данных и используются для устранения семантического разрыва между объектно-ориентированным способом хранения данных и необходимостью сохранять такие данные на жестком носителе.

Иными словами, ORM-технологии используются в объектно-ориентированном программировании для удобства – объекты преобразуются в некий набор данных, который может быть сохранен в файлах или базе данных. В дальнейшем этот набор может быть легко извлечен, с сохранением свойств объектов и отношений между ними.

Существует несколько подходов к решению этой задачи [8]:

- **Model First** – Подход, при котором сначала реализуется модель БД (например, в виде UML-диаграммы).
- **Database First** – Реализация основывается на использовании уже существующей БД для формирования объектной модели.
- **Code First** – Подход, использованный при реализации для Polar. При нем сначала пишется код, задающий схему, необходимую для конструирования БД.

Система Polar обладает рядом объективных достоинств – она не требовательна к системным ресурсам, не требует специализированной установки, неприхотлива к объемам памяти и дискового пространства, а также к скорости работы процессора. В связи с этим авторы сочли интересным расширить круг задач, для решения которых система хранения Polar потенциально могла бы быть использована в качестве СУБД.

Независимо от типа используемой СУБД, при решении таких задач разработчикам неизбежно приходится решать ряд конструктивных проблем. Например, разработка приложений с использованием практически любой системы хранения данных, будь то MS SQL, MySQL или Polar в качестве СУБД, требует длительного времени для изучения ее спецификации [4, 9, 10]. Далее, программные модули, использующие API, имеют достаточно громоздкий

Таблица 1

Сравнительные характеристики реляционных и Polar хранилищ триплетов

Число триплетов в данных	MS SQL	MySQL	Polar
100 000	2,0 мс	0,3 мс	0,2 мс
1 000 000	2,6 мс	0,5 мс	0,3 мс
10 000 000	2,8 мс	2,0 мс	0,3 мс

и однообразный код, в котором легко допустить ошибку.

В связи с этим представляется целесообразной реализация механизмов, обеспечивающих не только простое взаимодействие разработчика с данной системой, но и позволяющих эффективно производить манипуляции с БД. Технология ORM является одним из таких механизмов.

Заметим, что все ORM-решения, разработанные для универсальных СУБД, имеют ряд коммуникационных сложностей. Как правило, приложения, обеспечивающие реализацию ORM-технологии, и универсальные базы данных, которые используются как системы хранения данных при реализации ORM-технологии, представляют собой независимые решения, созданные различными группами программистов. Сообщение между приложениями в этих случаях осуществляется за счет универсальных протоколов и технологий предоставления данных, таких как ODBC [5] и ADO.NET [11]. Соответственно, обобщенная система затрачивает дополнительное время и дополнительную память на обеспечение взаимодействия.

При разработке ORM-решения для Polar (далее ORM Polar) вышеописанные потери были изначально минимизированы за счет использования внутренних методов языка Polar и отказа от универсальности. Это позволило упростить обобщенную систему, что служит дополнительным доводом в пользу целесообразности разработки ORM Polar.

Безусловно, технология ORM обладает рядом недостатков, существенных для ее реализации на практике. Например, дополнительный слой абстракции может сказаться на производительности, решение простых задач может оказаться слишком сложным, либо само ORM решение может быть недостаточно гибким. Кроме того, дизайн системы может оказаться зависимым от конкретной ORM-библиотеки. Ставя перед собой задачу разработки ORM для нереляционной базы (в нашем случае – ORM Polar), авторам приходится принимать их во внимание.

3. Принципы реализации ORM Polar

При создании программного модуля, реализующего ORM-технологию для системы хранения данных Polar, было предложено использовать подход Code First как основной. Ему отдано предпочтение за гибкость, простоту и удобство управления автоматически создаваемыми структурами данных путем редактирования соответствующих классов. В PolarDB предложено было создавать ячейку со структурой типа «запись», то есть ячейка представляет собой коллекцию однотипных массивов данных. В качестве методики разработки приложения ORM Polar авторы выбрали набор специальных классов, описывающих механизм размещения объектов модели в БД, а не xml-файл с сохраненным представлением полей базы данных и связей между таблицами [3]. Этот вариант авторам показался более удобным, так как с его помощью можно непосредственно в коде задавать различные изменения в БД.

Тогда программный модуль транслирует схему в структуры данных системы PolarDB с применением технологии «рефлексия» [6] по следующим правилам:

- каждый класс транслируется в определенную ячейку PolarDB с тем же названием;
- каждое поле класса сопоставляется с полем ячейки;
- стандартные типы C# транслируются во внутренние типы PolarDB;
- для определения полей, к которым должны быть созданы индексные структуры, используется маркерный атрибут [Index].

Среди набора классов, обеспечивающих функциональность технологии ORM на базе PolarDB, можно выделить два основных: DbContext и DbSet.

Кроме того, существует вспомогательный класс UserContext, он дает возможность пользователю взаимодействовать с наборами данных из БД, как с коллекциями.

Задачу взаимодействия между таблицами БД, а также анализ и создание структуры данных решает класс DbContext.

DbSet класс, который представляет собой набор сущностей, хранящихся в БД.

Задачу взаимодействия таблиц можно решать несколькими способами:

- созданием дополнительной ячейки, которая бы сопоставляла ключи реляционных таблиц между собой;
- определением атрибутов полей, содержащих информацию о связи с другой ячейкой;
- заданием внешних ссылок в схеме структуры данных.

Рассматривая задачу эффективного нахождения поисковых ключей в таблицах БД на диске, стоит подчеркнуть важность использования индексных структур, которые в некоторых случаях позволяют достичь достаточно высокого уровня производительности системы PolarDB.

4. Сравнительный анализ производительности ORM Polar и популярных ORM-решений

Для проведения сравнительного анализа производительности была разработана авторская методика тестирования.

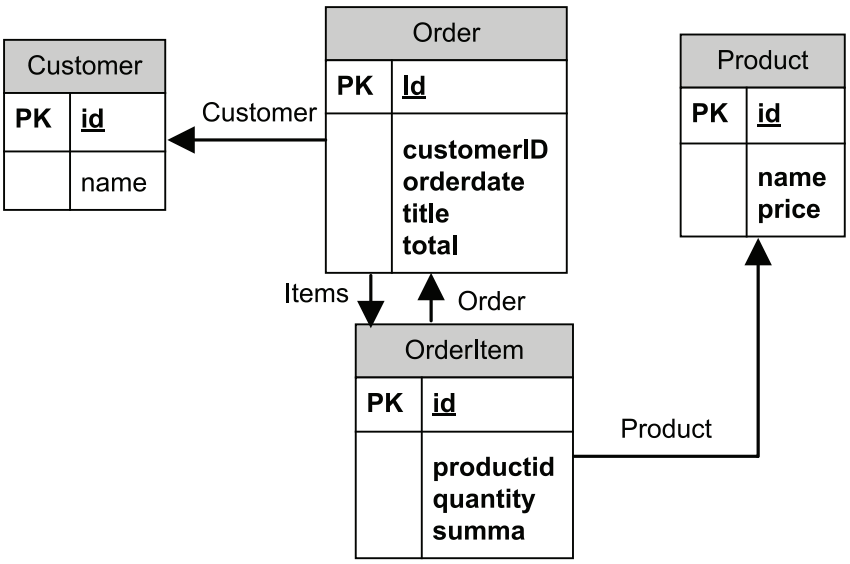
Для этого, взяв за основу популярную модель NordWind [3, 8], описывающую условный магазин, было разработано специальное приложение, позволяющее через специализированный модуль-контейнер использовать различные популярные ORM-системы-связки в качестве СУБД данного приложения.

UML-диаграмма классов модели приведена на схеме.

Это позволило сравнить скорость поиска, скорость загрузки данных и скорость обновления данных для тестируемых систем, наиболее популярных на данный момент среди подобных решений. Они могут являться, по мнению авторов, конкурентами системы ORM Polar: Entity Framework [8], Hibernate [3] for MS SQL, Hibernate for MySQL. Итоговые результаты анализа можно увидеть в табл. 2.

Из табл. 2 видно, что система ORM Polar может быть эффективно использована для большого круга коммерческих приложений.

Важно отметить, что при работе с большим количеством данных (например, при реализации приложений по автоматизации управления магазином с количеством продуктов, превышающим 1 млн наименований) предложенная система начинает проигрывать по скорости обработки и поиска популярным решениям. Это объясняется тем, что на таких объемах данных алгоритмы, используемые при реализации описанного в данной статье ORM-решения, делают системный кэш неэффективным. Но в реальности задачи такого рода следует относить к специфическим и требующим специального решения, что позволяет считать предложенное ORM-решение приемлемым для решения задач такого класса.



UML-диаграмма классов модели NordWind

Таблица 2

Сравнительные характеристики времени поиска
1000 случайных запросов популярных ORM-решений и ORM Polar (в мс)

Система	Общее количество записей в таблице, по которой осуществляется поиск				
	1 000	3 000	10 000	100 000	1 000 000
SQLite Container	4029	4570	4210	7260	5808 с
SQLite EF	4990	4500	5001	6921	5190 с
MSSQL Nhibernate (HQL)	1803	2023	3601	9024	6230 с
Polar Entity	123	158	531	635	762 с

Заключение

Способ построения ORM-решения, описанный в настоящей статье, существенно расширяет возможности использования технологии ORM применительно к нереляционным NoSQL базам данных.

Кроме того, практическая реализация предложенного авторами способа для специализированной NoSQL базы данных Polar, а также сравнительный анализ полученного решения ORM Polar с другими популярными решениями, подтверждает гипотезу о перспективности использования ORM-технологии для баз данных данного типа.

Данная статья поддержана грантом РФФИ № 14-07-00386 А.

Список литературы

1. Марчук А.Г. На пути к большим RDF данным // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: труды XV Всероссийской научной конференции RCDL›2013, Ярославль, Россия, 14–17 октября 2013 года. – Ярославль: ЯрГУ, 2013. – С. 51–56.

2. Марчук А.Г., Лельчук Т.И. Язык программирования Поляр: описание. использование, реализация / под ред. В.Е. Котова; АН СССР, Сиб. отделение, ВЦ. – Новосибирск: ВЦ СО АН СССР, 1986. – 94 с.

3. Сахил Малик. Microsoft ADO.NET 2.0 для профессионалов = Pro ADO.NET 2.0. – М.: Вильямс, 2006. –С. 560. – ISBN 1-59059-512-2.

4. Фленов М.Г. Transact-SQL // БХВ-Петербург. – 2006. – 897 с.

5. Эрик С. Реймонд. MySQL. Руководство администратора = MySQL. Administrator›s Guide. – М.: Вильямс, 2005. – С. 624. – ISBN 5-8459-0805-1.

6. Bauer, Christian, King, Gavin. Hibernate In Action. – Greenwich: Manning Publications, 2004. – 408 p. – ISBN 1-932394-15-X, ISBN 978-1-932394-15.

7. Forman Nate. Java Reflection in Action. – Manning Publications Co., 2004. – ISBN 1932394184.

8. John Hebeler, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez. Semantic Web Programming // John Wiley & Sons, 2009. – 648 с. – ISBN 9780470418017.

9. Julie Lerman, Rowan Miller. Programming Entity Framework: Code First. – O`Relly, 2011. – 194 с. – ISBN 978-1-4493-1294-7.

10. Microsoft Ltd. Электронная документация по SQL Server 2014: [Электронный документ] – (https://msdn.microsoft.com/ru-ru/library/ms130214. .aspx?tduid=(be6dfd6dd38ed827a1a8a2c5a71abe81) (256380) (2459594) (XdSn0e3h3.k-ee1xh8LWNRKyjfy20i8ZmQ)). Проверено 10.02.2016 г.

11. Roger Sippl. SQL Access Group›s Call-Level Interface:[Электронный документ] – (http://www.drdobbs.com/sql-access-groups-call-level-interface/184410032). Проверено 10.02.2016 г.

12. Scott W. Amber. Mapping Objects to Relational Databases: O/R Mapping In Detail: [Электронный документ] – (http://www.agiledata.org/essays/ mappingObjects.html). Проверено 10.02.2016 г.

13. W3C. Resource Description Framework (RDF) Model and Syntax Specification: [Электронный документ] – (https://www.w3.org/TR/PR-rdf-syntax/) Проверено 17.02.2016 г.

References

1. Marchuk A.G. Na puti k bolshim RDF dannym // Jelektronnye biblioteki: perspektivnye metody i tehnologii, jelektronnye kollekcii: trudy XV Vserossijskoj nauchnoj konferencii RCDL2013, Jaroslavl, Rossija, 14–17 oktjabrja 2013 goda. Jaroslavl: JarGU, 2013. pp. 51–56.

2. Marchuk A.G., Lelchuk T.I. Jazyk programmirovanija Poljar: opisanie. ispolzovanie, realizacija / pod red. V.E. Kotova; AN SSSR, Sib. otdelenie, VC. Novosibirsk: VC SO AN SSSR, 1986. 94 p.

3. Sahil Malik. Microsoft ADO.NET 2.0 dlja professionalov = Pro ADO.NET 2.0. M.: Viljams, 2006. –S. 560. ISBN 1-59059-512-2.

4. Flenov M.G. Transact-SQL // BHV-Peterburg. 2006. 897 p.

5. Jerik C. Rejmond. MySQL. Rukovodstvo administratora = MySQL. Administrators Guide. M.: Viljams, 2005. pp. 624. ISBN 5-8459-0805-1.

6. Bauer, Christian, King, Gavin. Hibernate In Action. Greenwich: Manning Publications, 2004. 408 p. ISBN 1-932394-15-X, ISBN 978-1-932394-15.

7. Forman Nate. Java Reflection in Action. Manning Publications Co., 2004. ISBN 1932394184.

8. John Hebeler, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez. Semantic Web Programming // John Wiley & Sons, 2009. 648 p. ISBN 9780470418017.

9. Julie Lerman, Rowan Miller. Programming Entity Framework: Code First. O`Relly, 2011. 194 p. ISBN 978-1-4493-1294-7.

10. Microsoft Ltd. Jelektronnaja dokumentacija po SQL Server 2014: [Jelektronnyj dokument] (https://msdn.microsoft.com/ru-ru/library/ms130214..aspx?tduid=(be6dfd6dd38ed827a1a8a2c5a71abe81)(256380)(2459594)(XdSn0e3h3.k-ee1x-h8LWNRKyjfy20i8ZmQ)). Provereno 10.02.2016 g.

11. Roger Sippl. SQL Access Groups Call-Level Interface:[Jelektronnyj dokument] (http://www.drdobbs.com/sql-access-groups-call-level-interface/184410032). Provereno 10.02.2016 g.

12. Scott W. Amber. Mapping Objects to Relational Databases: O/R Mapping In Detail: [Jelektronnyj dokument] (http://www.agiledata.org/essays/ mappingObjects.html). Provereno 10.02.2016 g.

13. W3C. Resource Description Framework (RDF) Model and Syntax Specification: [Jelektronnyj dokument] (https://www.w3.org/TR/PR-rdf-syntax/) Provereno 17.02.2016 g.