ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), ISSN 2222-0402
www.ores.su
original-research.ru

УДК 004.6

ОБЗОР ОСНОВНЫХ ПОНЯТИЙ ORM И JPA

Якунькин Владислав Романович
Тверской государственный технический университет, кафедра информационных систем
студент 2-го курса магистратуры,
г. Тверь,
Россия
gidranoit@gmail.com

Панин Егор Алексеевич
Тверской государственный технический университет, кафедра информационных систем
студент 2-го курса магистратуры,
г. Тверь,
Россия
rudi.pach@ya.ru

Брагин Андрей Алексеевич
Тверской государственный технический университет, кафедра информационных систем
студент 2-го курса магистратуры,
г. Тверь,
Россия
Почта: snaki2012@yandex.ru

Аннотация

ORM технология заняла одно из ключевых мест в разработке информационных систем различных масштабов с использованием объектных языков программирования, что привело к быстрой популяризации технологии. В статье рассматривается структура основных элементов, на которых строится технология ORM, особенности каждого элемента, методы сохранения, связь технологии с реляционными СУБД. В роли примера реализации технологии рассматривается JPA.

Ключевые слова: ORM, Сущность, Ассоциации, Запросы, Проекции, СУБД, JPA, XML, аннотации;

OVERVIEW OF THE BASIC CONCEPTS OF ORM AND JPA

Vladislav R. Yakunkin
Tver' State Technical University, Department of Information Systems, second-year master's student, Tver', Russia
gidranoit@gmail.com

Egor A. Panin
Tver' State Technical University, Department of Information Systems, second-year master's student, Tver', Russia
rudi.pach@ya.ru

Andrey A. Bragin
Tver' State Technical University, Department of Information Systems, second-year master's student, Tver', Russia
Mail: snaki2012@yandex.ru

ABSTRACT

ORM technology has taken one of the key places in the development of information systems of various scales using object programming languages, which led to the rapid popularization of the technology. The article discusses the structure of the main elements on which the ORM technology is built, the features of each element, methods of preservation, the relationship of technology with relational databases. JPA is considered as an example of technology implementation.

Keywords: ORM, Entity, Associations, Queries, Projections, DBMS, JPA, XML, annotations;

Введение.

В области объектно-реляционного отображения существует ряд терминов, с которыми мы постоянно сталкиваемся. Многие термины придуманы разработчиками ORM решений, другие же заимствованы из области баз данных. Object Relational Mapping (ORM) [1] — это метод преобразования данных из объектно-ориентированной модели в модель реляционной базы данных. Объектно-ориентированное программирование (ООП) основано на сущностях, в то время как система управления реляционными базами данных (СУБД) [2] стоит на отношениях и полях для хранения данных. Чтобы избежать проблем с отображением, ORM устраняет разрыв между платформами и управляет несоответствием между графами объектов и языком структурированных запросов – SQL [3].

Сущность - Entity

Сущность – это некий набор данных, хранящихся в объекте в соответствии с требованиями разработчика для дальнейшей обработки конкретным приложением. Этот

набор данных, можно сказать, повторяет данные, которые находятся в базе данных. Это может быть клиент, сотрудник или пост в соц. сети – все они являются типичными представителями сущности. [4] В коде такие объекты-сущности могут содержать множество данных, при этом демонстрировать дополнительную функциональность, чем требуется на уровне базы данных, но общая идея состоит в том, что, обращаясь к сущности в коде, то вы получаете доступ ко всем данным в базе, с которыми эта сущность связана.

Для администратора баз данных сущности обычно соответствуют таблицам, а объект в коде соответствует кортежу в этой таблице. Кроме того, сущности в коде используются не только для простого представления записей в таблице базы данных. Аналогично сущности могут использоваться для комбинации данных в моментах, когда разработчику требуются комбинация множества данных, которые изначально находятся в разных таблицах.

Свойства - Properties

Данные сущности хранятся в свойствах точно так же, как и в классических объектах, используя свойства для хранения данных. Как упоминалось ранее, сущности соответствуют таблицам и кортежам в них; отсюда следует, что эти объекты должны быть однозначно идентифицированы платформой ORM на уровне сущности. Такая задача решается путем присвоения определенному свойству функции первичного ключа. Эта концепция во многом совпадает с идеей первичных ключей в базах данных, но одно из основных различий заключается в том, что сущности могут содержать простые или сложные типы данных.

Ассоциации - Associations

Сущности, как и таблицы, создаются для поддержки конкретной модели данных, поэтому сущности не имеют особого смысла, если они не формируют отношения, представляющие логические и концептуальные понятия. Эти отношения называются ассоциациями в рамках ORM, и они формируются между одним или несколькими свойствами. Данные свойства в пространстве ORM известны как endpoint'ы ассоциаций, и в зависимости от типов данных они могут определять различные типы ассоциаций (мощность), которые представлены общими отношениями "один-к-одному", "один-ко-многим" и "многие-ко-многим" [5].

Хотя отношения один-к-одному и многие-ко-многим могут быть описаны в теоретических терминах и реализованы на уровне базы данных, они вряд ли имеют особый бизнес-смысл. Поэтому в большинстве ORM решений все ассоциации представляют собой двунаправленные отношения "один-ко-многим". Ассоциации, как правило, являются двунаправленными, поэтому сущности имеют полный доступ друг к другу, во многих отношениях на уровне базы данных свойства endpoint'ов работают так же, как и внешние ключи, и ведут себя аналогично операциям с join'ом таблиц.

Запросы - Criteria

Когда разработчик хочет получить данные из базы данных, он создаёт запрос, используя соответствующий язык (SQL), при этом фильтруя результаты запроса при необходимости. На уровне ORM запросы строятся на основе условий, которые передаются ядру базы данных. Эти условия, которые могут быть общими или специфическими,

формируются они путем объединения критериев поиска. Большинство ORM решений предоставляют удобный интерфейс для управления запросом – то есть построение запроса отображается пользователю как естественный язык. [5] Так же можно ожидать наличие критериев для всех полезных реляционных сравнений, например: больше, равно, логическое И, логическое ИЛИ и т. д.) и функций для сортировки.

Проекции - Projection

Хотя и можно получить все столбцы из таблицы базы данных с помощью запроса, общая практика программирования показывает, что необходимо извлекать только те свойства, которые требуются в конкретной ситуации. Такой тип запросов управляется ORM решениями иначе, чем типичные запросы, и они называются проекциями. Проекции также позволяют разработчикам детализировать сложные структуры данных и даже выполнять некоторые (базовые) математические вычисления, например: среднее, суммирование и т.д.

Контейнеры

ORM решения, как правило, создают буфер между кодом и базой данных [5]. Когда пользователь передает операцию в ORM, фреймворк должен иметь доступ к состоянию всех данных в базе данных. Он загружает соответствующие данные в память в виде сущностей или других соответствующих структур данных, выполняет указанные разработчиком операции, а затем многократно возвращает изменения в базу данных. Прежде чем разработчик сможет взаимодействовать с сущностью, ORM решение загружает все необходимые данные в контейнер.

Контейнеры обычно недолговечны, так как их суть состоит в обслуживании определенного набора операций, и легки, поскольку их необходимо создавать и уничтожать множество раз в течение жизни приложения. Во многих аспектах контейнер несколько аналогичен транзакции в базе данных.

Для выполнения запроса и эффективного извлечения реляционных данных в объектно-ориентированном программировании был введен язык под названием DQL [6]. Помимо очевидного соглашения о программировании, ORM ускоряет процесс оптимизации за счет блокировки транзакций и поддерживает запись данных через определенные границы транзакций [7]. Более того, ORM настраивает данные, к которым осуществляется доступ, по шаблонам на основе записей. Так же ORM настраивает данные, к которым осуществляется доступ, по шаблонам на основе записей. ORM стандартизировал процесс сохранения объектов в базу данных через интерфейс Java Persistence API (JPA).

JPA – это интерфейс прикладного программирования Java [8], который управляет данными между объектами Java и реляционными базами данных. JPA – это спецификация, а не реализация для сохранения данных в СУБД. [9]

Из-за несостоятельности корпоративной модели данных и отсутствия стандарта для сохранения данных, Java разработчики часто представляли реализации JPA как попытку оптимизировать архитектуру отображения [10]. Реализации JPA улучшают масштабируемость и читаемость кода, отделяя спецификации JPA от базовой архитектуры API [11].

По словам Мики Еноки [12] промежуточное ПО – это программное обеспечение, которое объединяет программный компонент или корпоративное приложение; это слой,

который находится между операционной системой и приложениями. Чтобы сопоставить данные с базой данных, реализации JPA выполняют моделирование метаданных и создание схем, которые выполняются либо с помощью стандартных аннотаций путем определения @Annotation, либо с помощью файлов XML с использованием тегов. Аннотации признаны лучшей альтернативой, по сравнению с XML из-за простоты программирования. Оба метода сопоставления имеют схожую функциональность, но из-за высокой сложности XML они не особо предпочтительны при программировании API.

Например, аннотация и тег XML для создания столбца первичного ключа в сущности определяется с помощью «@Id» в аннотациях, а в XML синтаксис следующий: «name=”ID” value=”integer”/».

JPA иллюстрирует шаблонный код во время выполнения путем создания экземпляров предварительно определенных интерфейсов. Чтобы создать соединение между Java объектами и базой данных, JPA определяет интерфейсный объект, называемый EntityManagerFactory [13]. После завершения процесса сопоставления менеджер уничтожается, а ресурсы освобождаются. Объект EntityManagerFactory вызывается объектом интерфейса EntityManager. Затем EntityManager взаимодействует в контексте постоянства для выполнения операций создания, чтения, обновления и удаления (CRUD) над объектами.

Список литературы

1. https://www.researchgate.net/publication/42800142_Understanding_object-relational_mapping_A_framework_based_approach
2. https://www.researchgate.net/publication/321481498_Role_of_Database_Management_Systems_DBMS_in_Supporting_Information_Technology_in_Sector_of_Education
3. Itzik Ben-Gan. “T-SQL Fundamentals”, 2016
4. Chen, Tse-Hsun, ”An Empirical Study on the Practice of Maintaining Object-Relational Mapping Code in Java Systems.”, 2016
5. Ogheneovo,Edward Erhieyovwe, Asagba, Prince Oghenekaro, Ogini,Nicholas Oluwole, Object Relational Mapping Technique for Java Framework, International Journal of Engineering Science Invention, 2013.
6. Benjamin Eberlei, Guilherme Blanco, Jonathan WageRoman Borschel. Doctrine of Objects.
7. Linskey, Patrick Connor, and Marc Prud’hommeaux. ”An in-depth look at the architecture of an object/relational mapper.” Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007.
8. Mike Keith and Merrick Schincariol, Pro JPA 2, A definitive guide to mastering the Java persistence API, Book, 2010.
9. https://www.researchgate.net/publication/313263324_Review_on_ORM_based_JPA_implementations
10. Ireland, Christopher, and David Bowers. ”Exposing the myth: objectrelational impedance mismatch is a wicked problem.” DBKDA 2015, The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications. IARIA XPS Press, 2015
11. https://www.researchgate.net/publication/318733467_Application_Programming_Interface_Documentation_What_Do_Software_Developers_Want

12. Miki Enoki, Yosuke Ozawa, Hiroshi Horii, Tamiya Onodera. MemoryEfficient Index for Cache Invalidation Mechanism with OpenJPA, Web Information Systems Engineering - WISE 206 Volume 7651 of the series Lecture Notes in Computer Science pp 696-703.
13. B.Vasavi, Y.V.Sreevani, G.Sindhu Priya, HIBERNATE TECHNOLOGY FOR AN EFFICIENT BUSINESS APPLICATION EXTENSION , Volume 2, No. 6, Journal of Global Research in Computer Science Journal of Global Research in Computer Science, June 2011.