ТЕХНОЛОГИЯ OBJECT-RELATIONAL MAPPING

АРХИРИЙ Ольга

Технический Университет Молдовы

***Аннотация**: Статья посвящена анализу технологии Object-Relational Mapping. Показано назначение данной технологии. Дано описание основных преимуществ и недостатков, а также признаки рациональности её использования.*

***Ключевые слова:** ORM, сравнение технологий, объектно-ориентированное программирование, классы, объекты.*

1. Введение

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) — технология программирования. Она связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Для разработчиков программного обеспечения не является секретом, что в мире хранения данных доминируют реляционные СУБД, тогда как объектный подход к проектированию и программированию - в мире обработки данных.

2. Зачем нужна технология ORM?

ORM должен избавить нас от написания SQL запросов и, в идеале, вообще абстрагировать от базы данных (от способа хранения данных), чтобы мы могли работать с классами, в той или иной степени выражающими объекты бизнес логики, не задаваясь вопросом, в каких таблицах всё это по факту лежит.

Технология ORM позволяет программисту работать с таблицами, полями и связями реляционной БД, как с объектами, свойствами и коллекциями (массивами). Пользователь не отвлекается на подробности более низкого уровня, такие, как порядок выборки и сохранения модифицированных данных, вопросы переносимости и особенностей диалекта SQL конкретной СУБД, генерации уникальных первичных ключей, заполнения полей ссылок для моделирования связей.

3. Сравнение систем управления данными

Так как системы управления реляционными базами данных обычно не реализуют реляционного представления физического уровня связей, выполнение нескольких последовательных запросов (относящихся к одной «объектно-ориентированной» структуре данных) может быть слишком затратным. В частности, один запрос вида «найти такого-то пользователя и все его телефоны и все его адреса и вернуть их в таком формате», скорее всего, будет выполнен быстрее серии запросов вида «Найти пользователя. Найти его адреса. Найти его телефоны». Это происходит благодаря работе оптимизатора и затратам на синтаксический анализ запроса.

Необходимо обеспечить работу с данными в терминах классов, а не таблиц данных и напротив, преобразовать термины и данные классов в данные, пригодные для хранения в СУБД. Необходимо также обеспечить интерфейс для CRUD-операций над данными. В общем, необходимо избавиться от необходимости писать SQL-код для взаимодействия в СУБД.

Решение проблемы хранения данных существует — это реляционные системы управления базами данных. Использование реляционной базы данных для хранения объектно-ориентированных данных приводит к семантическому разрыву, заставляя программистов писать программное обеспечение, которое должно как обрабатывать данные в объектно-ориентированном виде, так и уметь сохранить эти данные в реляционной форме. Эта постоянная необходимость в преобразовании между двумя разными формами данных не только сильно снижает производительность, но и создает трудности для программистов, так как обе формы данных накладывают ограничения друг на друга. Схема сравнения данных подходов представлена на рисунке 1.

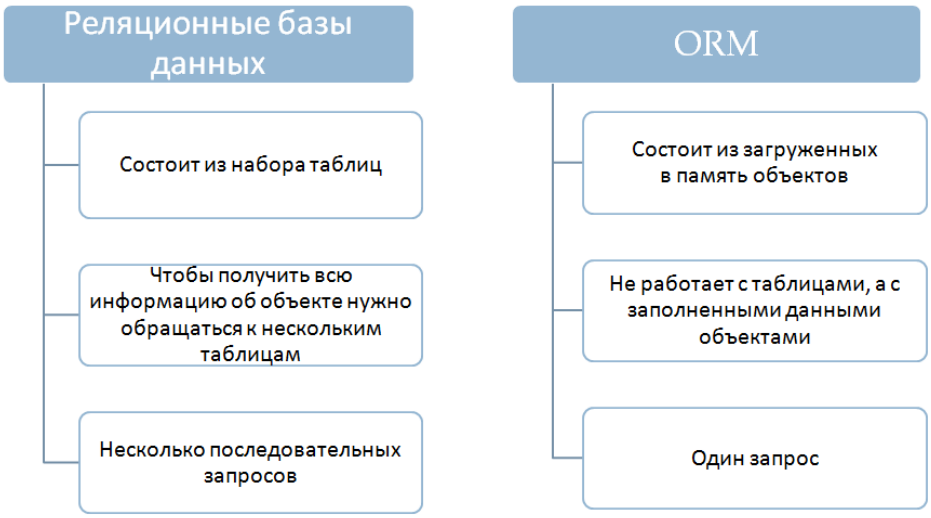

Рис. 1. Схема сравнения реляционных баз данных и ORM

Реляционные базы данных используют набор таблиц, представляющих простые данные. Дополнительная или связанная информация хранится в других таблицах. Часто для хранения одного объекта в реляционной базе данных используется несколько таблиц; это, в свою очередь, требует применения операции JOIN для получения всей информации, относящейся к объекту, для её обработки. Например, для хранения данных записной книжки, скорее всего, будут использоваться как минимум две таблицы: люди и адреса, и, возможно, даже таблица с телефонными номерами.

Некоторые реализации ORM автоматически синхронизируют загруженные в память объекты с базой данных. Для того чтобы это было возможным, после создания преобразующего в объект SQL-запроса (класса, реализующего связь с DB) полученные данные копируются в поля объекта, как во всех других реализациях ORM. После этого объект должен следить за изменениями этих значений и записывать их в базу данных.

Системы управления реляционными базами данных показывают хорошую производительность на глобальных запросах, когда затрагивают большой участок базы данных. Но объектно-ориентированный доступ более эффективен при работе с малыми объёмами данных. Это позволяет сократить семантический провал между объектной и реляционной формами данных.

При одновременном существовании этих двух концепций при работе с БД, увеличивается сложность объектного кода для работы с реляционными базами данных. Код становится более подверженным к ошибкам. Разработчики программного обеспечения, основывающегося на базах данных, искали более легкий способ достижения постоянства их объектов.

4. Когда использование ORM оправдано

С точки зрения программиста система должна выглядеть как постоянное хранилище объектов. Он может просто создавать объекты и работать с ними как обычно, а они автоматически будут сохраняться в реляционной базе данных.

На практике всё не так просто и очевидно. Все системы ORM обычно проявляют себя в том или ином виде, уменьшая в некотором роде возможность игнорирования базы данных. Более того, слой транзакций может быть медленным и неэффективным (особенно в терминах сгенерированного SQL). Все это может привести к тому, что программы будут работать медленнее и использовать больше памяти, чем программы, написанные «вручную».

Но ORM избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому жёстко задать код SQL-запросов, который будет использоваться при тех или иных действиях (сохранение в базу данных, загрузка, поиск и т. д.) с постоянным объектом.

ORM является дополнительным слоем абстракций и создает накладные расходы по использованию процессора и памяти, а работа с реляционной СУБД становится в некоторых случаях неоптимальной или даже неудобной по сравнению с SQL-командами. Не следует также забывать, что

SQL - это промышленный стандарт, тогда как внутренние языки запросов, используемые в ОРП, таковым не являются. Поэтому кроме объектно-ориентированной работы с данными большинство ОРП поддерживают возможность прямого использования SQL и вызовов хранимых процедур, а некоторые могут даже отображать объекты на хранимые процедуры. Данные возможности следует отнести к разряду оптимизации, сейчас мы не будем их рассматривать подробно, но если ваша БД содержит миллионы записей при одновременном доступе множества пользователей, то с большой вероятностью вам они понадобятся. Процесс принятия решения программистом представлен на рисунке 2.

Рис. 2. Критерии выбора ORM

Из уже названных различий между целями использования реляционной и объектной моделей следует в частности, что если в вашей системе основной упор делается на многокритериальный поиск и массированное извлечение информации (класс информационно-поисковых систем, OLAP, генерация отчетности), то использование объектов для доступа к данным не является оправданным, а попросту излишне. Никакого различия между табличным представлением информации в базе данных, внутри вашей программы и на экране пользователя или в отчете нет, промежуточная обработка сводится к соединениям все тех таблиц и простым пересчетам значений их полей. Другое дело, если ваша система осуществляет транзакционную обработку (OLTP), сложные расчеты, оповещения о событиях, диспетчеризацию, моделирует поведение - здесь преимущества использования ОРП наибольшие.

5. Заключение

ORM является полезным инструментом, избавляющим программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки.

Но данную технологию необходимо использовать с умом и только в тех местах, где её использование оправдано, использует шаблоны кода, которые имеют отличный дизайн и следуют шаблонам проектирования. Это очень сокращает время на разработку и тестирование, проще сопровождать, меньше кода, нет необходимости самостоятельно писать запросы, создавать объекты, потому как можно работать с имеющимися бизнес объектами. Не нужно преобразовывать вручную объекты из или в БД и тестировать то, чего нет – такой код уже написан и протестирован. Главная задача ORM заключается в том, чтобы являться связующим звеном, которое прозрачно для нас будет делать всю работу по сохранению данных бизнес объектов и заполнению их данными из базы при необходимости.

Литература

1. Обзор средств объектно-реляционной проекции (ORM) для платформы .NET. [Электронный ресурс]. - Режим доступа: https://arbinada.com/en/node/33
2. Введение в ORM (Object Relational Mapping). [Электронный ресурс]. - Режим доступа: http://internetka.in.ua/orm-intro/
3. ORM. [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/ORM
4. What’s New in MDS of SQL Server 2016. [Электронный ресурс]. - Режим доступа: http://www.radacad.com/whats-new-in-mds-of-sql-server-2016