Что такое ORM?
ORM или **Object-relational mapping** *(рус. Объектно-реляционное отображение)* — это технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, в частности, между хранилищем данных и объектами программирования. ORM используется для упрощения процесса сохранения объектов в реляционную базу данных и их извлечения, при этом ORM сама заботится о преобразовании данных между двумя несовместимыми состояниями. Большинство ORM-инструментов в значительной мере полагаются на метаданные базы данных и объектов, так что объектам ничего не нужно знать о структуре базы данных, а базе данных — ничего о том, как данные организованы в приложении. ORM обеспечивает полное разделение задач в хорошо спроектированных приложениях, при котором и база данных, и приложение могут работать с данными каждый в своей исходной форме.

Парадигма «несоответствия»
Говоря конкретнее, использование ORM решает проблему так называемой парадигмы «несоответствия», которая гласит о том, что объектные и реляционные модели не очень хорошо работают вместе. Реляционные базы представляют данные в табличном формате, в то время как объектно-ориентированные языки представляют их как связанный граф объектов. Основные проблемы и несоответствия возникают во время сохранения этого графа объектов в реляционную базу или его загрузки:

- реляционная модель может быть намного детальнее, чем объектная, т.е. для хранения одного объекта в реляционной базе данных используется несколько таблиц;
- реляционные СУБД не имеют ничего похожего на наследование — естественную парадигму объектно-ориентированных языков программирования;
- в СУБД определен только один параметр для сравнения записей — первичный ключ. В то время как ООП предоставляет как проверку идентичности объектов (a==b), так и их равенства (a.equals(b));
- для связи объектов СУБД использует понятие внешних ключей, в объектно-ориентированных языках связь между объектами может быть только однонаправленной. Если же нужно организовать двунаправленные отношения, то придется определить две однонаправленные ассоциации. Кроме того, нет возможности определить кратность отношения, глядя на модель предметной области;
- принцип доступа к данным в ООП кардинально отличается от доступа к данным в БД. Для доступа к данным в ООП используются последовательные переходы от родительского объекта к свойствам дочерних элементов и инициализации объектов по необходимости. Такой подход считается не эффективным способом извлечения данных из реляционных баз данных. Как правило, количество запросов к БД должно быть сведено к минимуму, необходимые сущности должны по возможности загружаться сразу с использованием JOIN-ов.

Принцип работы ORM

Ключевой особенностью ORM является отображение, которое используется для привязки объекта к его данным в БД. ORM как бы создает «виртуальную» схему базы данных в памяти и позволяет манипулировать данными уже на уровне объектов. Отображение показывает как объект и его свойства связанны с одной или несколькими таблицами и их полями в базе данных. ORM использует информацию этого отображения для управления процессом преобразования данных между базой и формами объектов, а также для создания SQL-запросов для вставки, обновления и удаления данных в ответ на изменения, которые приложение вносит в эти объекты.

Преимущества и недостатки использования

Использование ORM в проекте избавляет разработчика от необходимости работы с SQL и написания большого количества кода, часто однообразного и подверженного ошибкам. Весь генерируемый ORM код предположительно хорошо проверен и оптимизирован, поэтому не нужно в целом задумывается о его тестировании. Это несомненно является плюсом, но в тоже время не стоит забывать и о минусах. Основной из них — это потеря производительности. Это происходит потому, что большинство ORM предназначены для обработки широкого спектра сценариев использования данных, гораздо большего, чем любое отдельное приложение когда-либо сможет использовать. Вопрос о целесообразности использования ORM по большому счету затрагивается только в больших проектах, которые сталкиваются с высокой нагрузкой, здесь приходится выбирать что более приоритетно — удобство или производительность? Конечно, работа с БД посредством грамотно написанного SQL-кода будет намного эффективнее, но не стоит забывать и о таком параметре, как время — то, что с легкостью пишется с использованием ORM за неделю, можно реализовывать ни один месяц собственными усилиями. Кроме того, большинство современных ORM позволяют программисту при необходимости самому задавать код SQL-запросов. Без сомнений, для небольших проектов использование ORM будет куда более оправдано, чем разработка собственных библиотек для работы с БД.