технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.
Суть задачи состоит в преобразовании объектов в форму, в которой они могут быть сохранены в файлах или базах данных, и которые легко могут быть извлечены в последующем, с сохранением свойств объектов и отношений между ними. Эти объекты называют «хранимыми» (англ. persistent).
Решение проблемы хранения данных существует — это реляционные системы управления базами данных. Использование реляционной базы данных для хранения объектно-ориентированных данных приводит к семантическому провалу, заставляя программистов писать программное обеспечение, которое должно уметь 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.
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 ее обработки. Например, в рассмотренном варианте с записной книгой, для хранения данных, скорее всего, будут использоваться как минимум две таблицы: люди и адреса, и, возможно, даже таблица с телефонными номерами.
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 затратно. 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.
Некоторые реализации ORM автоматически синхронизируют загруженные в память объекты с базой данных. Для того чтобы это было возможным, после создания объект-в-SQL-преобразующего SQL-запроса 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, которые затрагивают большой участок базы данных, но объектно-ориентированный доступ более эффективен при работе с малыми объёмами данных, так как это позволяет сократить семантический провал между объектной и реляционной формами данных.
При одновременном существовании этих двух разных миров увеличивается сложность объектного кода для работы с реляционными базами данных, и он становится более подвержен ошибкам. Разработчики программного обеспечения, основывающегося на базах данных, искали более легкий способ достижения постоянства их объектов.
Некоторые пакеты решают эту проблему, предоставляя библиотеки классов, способных выполнять такие преобразования автоматически. Имея список таблиц в базе данных и объектов в программе, они автоматически преобразуют запросы из одного вида в другой. В результате запроса объекта «человек» (из примера с адресной книгой) необходимый SQL-запрос будет сформирован и выполнен, а результаты «волшебным» образом преобразованы в объекты «номер телефона» внутри программы.
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.