УДК: 519.688

Романов С. С.
Магистрант 2 курса,
Хакасский Государственный Университет им. Н. Ф. Катанова

ДОСТОИНСТВА, НЕДОСТАТКИ И АЛЬТЕРНАТИВЫ ОБЪЕКТНО-РЕЛЯЦИОННОГО ОТОБРАЖЕНИЯ (ORM)

В работе рассматривается понятие объектно-реляционного отображения, его достоинства и недостатки, а также возможные альтернативы.

Ключевые слова: *ORM, объектная модель, реляционная модель, объектно-реляционное отображение.*

Применение объектно-реляционных отображений (англ. Object — Relational Mapping, ORM) в настоящее время является общим средством в процессе разработки сложных систем, позволяющим объединить объектно-ориентированную модель представления данных с реляционной [4].

В конкретных реализациях ORM, как правило, используется шаблон, который заключается в отображении класса в таблицу и атрибутов объекта класса в поля этой таблицы.

Рассмотрим понятие ORM. Согласно [5], ORM — это мощное средство для проектирования и осуществления запросов к моделям базы данных на концептуальном уровне, где приложение описано в терминах, понятных для технически неподкованных пользователей.

В другом источнике [3] отмечается, что ORM — это библиотека языка программирования, выполняющая отображение объектов реляционной модели на объекты языка программирования.

Также ORM называют «технологией программирования, позволяющей преобразовывать несовместимые виды моделей в ООП, для того, чтобы упростить процедуру сохранения объектов в реляционную БД и их извлечения» [2].

Как правило, отображение двунаправлено: операции с атрибутами объекта ведут к чтению/записи данных из/в соответствующие таблицы базы данных [3]. Для использования и управления ORM в программном коде задается отображение схемы базы данных на схему языка программирования. Пример на языке Python [там же] представлен в листинге ниже.

Листинг 1 — Пример определения отображения схемы базы данных на схему языка программирования Python

```
class Elephant(DBTable): # Класс, определяющий таблицу elephants
  table_name = «elephants»
  id = IntegerColumn(auto_increment=True, primary_key=True) # Первичный ключ
  color = StringColumn(default=None, length=30)
elephant = Elephant.new() # INSERT, т.е. создание новой строки в таблице
elephant = Elephant.new(color=«pink») # INSERT, т.е. новая строка с данными
elephant = Elephant(id=2112) # извлечение новой строки по id
elephants = Elephant.select(color=«pink») #извлечение нескольких строк по цвету
color = elephant.color # Выполняет SELECT и, возможно, кэширует результат
elephant.color = "pink" # Выполняет UPDATE, т.е. запись данных в таблицу
```

Для того, чтобы «привязать» объект к его соответствующим данным в БД применяется отображение. Создается некая виртуальная схема БД в памяти, благодаря

которой OMR дает возможность оперировать данными на уровне объектов и которая определяет связь между свойствами объекта с одной и более таблицами и полями в БД. ORM, опираясь на эти данные, управляет преобразованием данных между БД и объектами, создает SQL-запросы при изменении объектов.

Среди **достоинств** ORM выделяют [3, 2]:

- наличие явного описания схемы БД, представленное в терминах какого-либо языка программирования, которое находится и редактируется в одном месте;
- возможность программисту оперировать элементами языка программирования, т.е. классами, объектами, атрибутами, методами, а не элементами реляционной модели данных;
- возможность автоматического создания SQL-запросов, которая избавляет от необходимости использования языка для описания структуры БД (Data Definition Language) и языка манипулирования данными (Data Manipulation Language) при проектировании БД и изменении её схемы соответственно;
- не нужно создавать новые SQL-запросы при переносе на другую систему управления базами данных, поскольку за это отвечает низкоуровневый драйвер ORM.
- ORM избавляет от необходимости работы с SQL и проработки значительного количества программного кода, который зачастую однообразен и подвержен ошибкам.
- код, генерируемый ORM гипотетически проверен и оптимизирован, следовательно не нужно беспокоиться о его тестировании;
- развитые реализации ORM поддерживают отображение наследования и композиции на таблицы;
- ORM дает возможность изолировать код программы от подробностей хранения данных.

Среди **недостатков** ORM выделяются [3, 4, 2]:

- Дополнительная нагрузка на программиста, которому, в случае использования ORM необходимо изучать этот некий «дополнительный слой» между программной и базой данных, который к тому же создает дополнительный уровень абстракции — объекты ORM. В связи с этим могут возникнуть вопросы соответствия особенностям ООП и соответствующим реляционным операциям. Эту проблему называют impedance mismatch, а сама реализация ORM ведет к увеличению объема программного кода и снижению скорости работы программы. Однако, с другой стороны, ORM наглядно и в одном месте концентрирует различие между реляционной и объектно-ориентированной парадигмами, что нельзя назвать недостатком;
- Появление трудно поддающихся отладке ошибок в программе, если присутствуют ошибки в реализации ORM, например, ошибки в реализации кэширования ORM, такие как согласование изменений в разных сессиях.
- Недостатки реализаций, которые могут иметь определенные ограничения и выдвигать определенные требования, например, требование собственной схемы базы данных и ограничение на средства создания базы данных. Также может отсутствовать возможность написать в явном виде SQL-запрос.
- Требуются отдельные таблицы в случае прямого отображения классов в таблицы и необходимости отображения атрибутов множественного характера.

Если говорить о главном минусе ORM, снижении производительности, то причина этого состоит в том, что большинство из ORM нацелены на обработку значительного большего количества различных сценариев использования данных, чем в случае отдельного приложения.

В случае небольших проектов, которые не сталкиваются с высокой нагрузкой, применение ORM очевидно, особенно, если учесть такой важный критерий разработки, как время. «То, что с легкостью пишется с использованием ORM за неделю, можно реализовывать ни один месяц собственными усилиями», отмечается в [2].

Касательно **альтернатив** технологии ORM, то среди них выделяются [1, 4]:

- сознательное нарушение нормализации таблиц в реляционной модели, которое, хотя и приводит к избыточности данных наряду с появлением необходимости их синхронизации, обеспечивает преимущество ускорения доступа к данным;
- подход CoRM (Collection-Relational Mapping — реляционное отображение коллекций), при котором осуществляется объединение разрозненных коллекций объектов с помощью хорошо определенных реляционных взаимоотношений между коллекциями и прототипом которого могут служить документ-ориентированные СУБД (например, MongoDB);

В случае CoRM отсутствует ограничение на возможность хранить состояние разных объектов одного класса в разных коллекциях и ограничение на одновременное хранение в коллекции объектов, которые принадлежат разным классам, как отмечается в [4].

От обычного ORM-подхода реляционное отображение коллекций отличает то, что коллекция напрямую не привязана к классу и, в теории, может включать в себя любой объект, в случае соблюдения некоторых минимальных требований к данному объекту, например, требование наличия способности к сериализации). Однако к этим требованиям не относятся ограничения на структуру объекта либо использование особых типов данных.

Подводя итог, можно сказать, что ORM — это инструмент решения проблемы семантического провала между реляционной и объектной моделями данных. Имеющий, однако, определенные проблемы, которых должны быть лишены его альтернативы, позволяющие вывести объектную сущность приложения из ограничений, накладываемых реляционным хранилищем.

Литература

1. Буч Гради, Максимчук Роберт А., Энгл Майкл У., Янг Бобби Дж., Коналлен Джим, Хьюстон Келли А. Объектно-ориентированный анализ и проектирование с примерами приложений [Текст] / Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. — М.: ООО «И.Д. Вильямс», 2008. — 720 с.

2. Введение в ORM (Object Relational Mapping) [Электронный ресурс] // internetka.in.ua. — URL: http://internetka.in.ua/orm-intro/. — (дата обращения: 12.11.2016).

3. Объектно-реляционные отображения. Их достоинства и недостатки [Электронный ресурс] // phdru.name. — URL: http://phdru.name/Russian/Software/orm.html. — (дата обращения: 10.11.2016).

4. Реляционное отображение коллекций — альтернатива объектно-реляционному отображению? [Электронный ресурс] // habrahabr.ru. — URL: https://habrahabr.ru/post/181213/. — (дата обращения: 09.11.2016).

5. Object Role Modeling [Электронный ресурс] // www.orm.net. — URL: http://www.orm.net. — (дата обращения: 09.11.2016).