важно оптимизировать доступ к данным. Это необходимо сделать в процессе создания приложения в данной работе.

В результате работы создано приложение с полной интеграцией и возможностью управления существующей базы данных.

Е.В. Беззубов (УО «ГГУ им. Ф. Скорины», Гомель)
Науч. рук. **В.Н. Леванцов**, старший преподаватель

ДОСТОИНСТВА И НЕДОСТАТКИ OBJECT-RELATIONAL MAPPING

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.

Объектная и реляционная модели ортогональны. Они моделируют одну и ту же сущность, но с разных сторон. Реляционная модель акцентирует свое внимание на структуре и связях сущностей, объектная – на их свойствах и поведении. Цель использования реляционной модели – информационное моделирование, выделение существенных для нас атрибутов, сохранение их значений и последующего поиска, обработки и анализа. Цель использования объектной – моделирование поведения, выделение существенных для нас функций и последующего их использования. Между моделями есть пересечение – структурные сущности, которые по-разному в этих моделях отражаются. Для того, чтобы отобразить артефакты реляционной модели в артефакты же объектной в наших программах и требуется средство объектно-реляционной проекции – ОРП или ORM (Object Relational Mapping).

Объектно-реляционный проектор – ОРП – теоретически позволяет программисту работать с таблицами, полями и связями реляционной БД, как с объектами, свойствами и коллекциями (массивами), не отвлекаясь на подробности более низкого уровня, такими, например, 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.

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.

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.

Использование промежуточного слоя ORM имеет большое применение в таких реализациях, как и QxORM, EntityFramework, Dapper, Hibernate и пр. Но все эти технологии эффективно используются лишь для данных, хранящихся и управляющихся только одной СУБД. Использование технологии ORM позволяет автоматизировать контроль расположения данных. При классическом проектировании разработчик должен обязательно указывать в каждом запросе расположение данных в гибридном облаке, программно подключаться и отключаться от БД. Все это приводит к увеличению трудоемкости разработки программных средств и появлению ошибок в коде.

Другой проблемой при работе с распределенными хранилищами данных является контроль их целостности. При работе с БД, управляемой одной СУБД, за контроль целостности отвечает сама СУБД. При распределенных БД данная функция перекладывается на разработчика программного обеспечения (ПО), так как для повышения надежности ПО данный функционал должен быть специально предусмотрен. Поэтому для автоматизации разработки и увеличения надежности функцию контроля целостности данных в распределенных хранилищах должна выполнять прослойка ORM.

К достоинствам данной технологии можно отнести следующее:

– Существует явное описание схемы БД в терминах языка программирования; описание это существует и изменяется в одном месте.

– Программист манипулирует привычными элементами языка программирования – классами, объектами, атрибутами и методами.

– Автоматическая генерация SQL-запросов. не надо писать DDL самому – ORM сгенерирует описание схемы. Не надо менять зашитые в программу DML-запросы при изменении схемы БД. Не надо менять запросы при переносе на другую СУБД – низкоуровневый драйвер ORM будет создавать новые запросы сам.

– Безопасность. ORM позволяет защитить приложение от SQL-инъекций.

– Ленивое исполнение запросов. Позволяет кэшировать результат запросов на уровне ORM.

К недостаткам можно отнести:

– Объектно-реляционное отображение создаёт дополнительный слой между программой и базой данных. Этот слой имеет свой собственный API, который необходимо изучить.

– Этот слой создаёт дополнительный уровень абстракции.

– Эта абстракция отображает друг на друга не вполне совместимые (по своим операциям) парадигмы – объектно-ориентированную и реляционную.

– Дополнительный слой – это дополнительный код, который надо распространять вместе с программой; он вызывает увеличение объёма и падение скорости программы.

– В случае ошибок в реализации ORM в программе возникают трудноотлаживаемые ошибки. Особенно тяжёлый случай – ошибки в реализации кэширования, когда ORM кэширует слишком мало, или наоборот, слишком агрессивно.

– Неоптимальный SQL.

М.С. Березовский (УО «ГГУ им. Ф. Скорины», Гомель)
Науч. рук. **М.И. Жадан,** канд. физ.-мат. наук, доцент

СОЗДАНИЕ БАЗЫ ДАННЫХ «КЛУБ» В СУБД DB2

Предметная область – часть реального мира, подлежащая изучению с целью организации управления и автоматизации. Предметная область представляется множеством фрагментов. Каждый фрагмент области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, которые хранятся в некоторой базе данных.