УДК 004

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ ИСПОЛЬЗОВАНИЯ ORM В РАЗРАБОТКЕ

Гаврилов К.А.
Научный руководитель: Чердынцев Е.С., к.т.н, доцент

Национальный Исследовательский Томский политехнический университет,
634050, Россия, г. Томск, пр. Ленина, 30
E-mail: kgavrilov_kz@mail.ru

Usage object-relational mapping for solving the problem of interaction between object-oriented programming and relational database management system. Advantages and disadvantages.

Ключевые слова: *объектно-реляционная модель, разработка, работа с БД.*

Key words: *Object-relational mapping, ORM, developing, working with database.*

В методологии объектно-ориентированного программирования (ООП) классы, описывающие свойства и поведение сущностей, являются абстрактным описанием объектов реального мира. Например, если рассматривать упрощённый вариант журнала успеваемости по дисциплине, то все объекты класса «Студент» будут содержать поля ФИО и массив оценок (баллов). При этом время жизни объектов данного класса ограничивается временем работы приложения. Поэтому возникает вопрос хранения таких объектов: объекты должны храниться в файлах или базах данных, быть легко извлекаемыми, с сохранением своих свойств и отношений с другими объектами.

Решением проблемы хранения данных таким образом является использованием реляционных систем управления базами данных (СУБД). Однако использование реляционной СУБД для хранения данных объектно-ориентированной среды приводит к нарушению семантики (так называемому семантическому разрыву), в результате чего проходиться разрабатывать программное обеспечение (ПО), которое должно как поддерживать работу с программными объектами, так и уметь сохранять данные этих объектов в реляционной форме. Постоянная необходимость преобразования формы данных снижает скорость работы решения и оказывает дополнительную нагрузку на разработчиков.

В основе работы реляционных СУБД лежит набор таблиц, содержащих простые данные. Связанная информация хранится в других таблицах, а связь осуществляется по внешнему ключу. Если объект достаточно сложен, то для его хранения может использоваться более десятка таблиц, а для получения всей информации об объекте требуется применять множество операций JOIN. Выполнение всех запросов последовательно, особенно при получении нескольких объектов разного типа, может быть достаточно затратным по времени.

СУБД обеспечивают прекрасную производительность на запросах, затрагивающих большой участок БД, однако при работе с небольшим объёмом данных эффективнее использовать объектно-ориентированный доступ, так как такой подход сокращает семантический разрыв между реляционной и объектной формами данных.

Как уже было сказано выше – такой подход оказывает дополнительную нагрузку на разработчиков, так как повышается сложность кода для работы с СУБД, увеличивается риск появления ошибок. Простым способом устранения данных проблем является использование ORM Фреймворков.

ORM (англ. Object-Relational Mapping) – это технология программирования, связывающая СУБД с объектно-ориентированным языком программирования, создавая «виртуальную объектную базу данных» [1].

Разработано множество решений, которые устраняют необходимость выполнять преобразование данных объектно-ориентированной среды перед их сохранением в СУБД. При этом существуют решения (например, Entity Framework (EF)), которые автоматически выполняют данные преобразования. Разработчик может задать мэппинг таблиц СУБД на классы решения, после чего ORM будет автоматически преобразовывать запросы из одного вида в другой.

Рассмотрим основные преимущества использования ORM. Для этого сравним объём и сложность кода при запросе сущности из БД напрямую из кода (рис. 1, слева), и с использование ORM (рис. 1, справа).

```
var query = @"SELECT * FROM Students WHERE ID = @ID";
using (var connection = new SqlConnection())
{
    connection.ConnectionString =
    @"Data Source=.\SQLEXPRESS;Initial Catalog=Test_DB;" +
    "Integrated Security=True";
    connection.Open();
    using (var command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("ID", id);
        using (var reader = command.ExecuteReader())
        {
            var student = new Student
            {
                Name = reader.
                GetString((reader.GetOrdinal("Surname")))
            };
            return student;
}}}
```

```
public static Student LoadStudent(Guid id)
{
    using (var context = new StudentContext())
    {
        return context
            .Students
            .FirstOrDefault(x => x.Id == id);
    }
}
```

Рис. 9. Запрос сущности из БД напрямую (слева) и с помощью ORM (справа)

Основные преимущества использования ORM [2, 3]:

- для реализации одинакового функционала требуется написать меньше кода;
- не нужно писать SQL запросы;
- не нужно самому создавать объекты;
- легко сопровождать.

В итоге использование ORM влияет на разработку следующим образом:

- сокращает время разработки – для работы с базой, содержащей 25–30 таблиц, необходимо завести 50–80 объектов, а это примерно от 7000 до 10000 строк кода, которые надо написать и протестировать. С ORM то же можно сделать за 1–2 дня;
- позволяет создать универсальный код – разные люди в команде могут создать свой уникальный код для преобразования объектов из/в БД, опираясь исключительно на свой опыт. ORM использует шаблоны кода, которые имеют отличный дизайн;
- сокращает время тестирования – программисту нет необходимости вручную преобразовывать объекты из/в базу данных, а код, который делает такое преобразование, написан и уже протестирован;
- упрощает сопровождение – жизнь программы не заканчивается выпуском релиза. Она живёт и развивается, и изменение кода в случае с ORM менее затратное занятие, так как написано меньше кода.

На данный момент разработано достаточно большое количество ORM для .Net, но есть как платные, так и бесплатные решения.

Как было сказано выше – недостаток ORM заключается в снижении скорости работы с базой. Слой транзакций (выполняющий преобразования) может быть недостаточно эффективным, поэтому при обработке большого количества данных работа приложения будет медленнее, чем при использовании чистого SQL.

Но большинство запросов, не требующих обработки больших объёмов данных, выполняется через ORM за вполне приемлемое время, а учитывая все плюсы – использование ORM может стать очень выгодным и разумным решением.

Список литературы

1. Object-relational mapping. [Электронный ресурс]. Режим доступа: https://en.wikipedia.org/wiki/Object-relational_mapping, свободный.
2. ORM или как забыть о проектировании БД. [Электронный ресурс]. Режим доступа: https://habrahabr.ru/post/237889/, свободный.
3. ORM технологии в .NET. [Электронный ресурс]. Режим доступа: http://www.slideshare.net/ptsukanov/orm-net-nhibernate-linq-to-sql-entity-framework, свободный.

УДК 004

СРЕДСТВА MICROSOFT ПРИ ПРОЕКТИРОВАНИИ И РЕАЛИЗАЦИИ ПРОГРАММНЫХ ПЛАТФОРМ ПОДДЕРЖКИ СОВРЕМЕННЫХ СТАНДАРТОВ ПЕРЕДАЧИ ДАННЫХ ДЛЯ НЕФТЕДОБЫЧИ

Гончаров А.С.
Научный руководитель: Марчуков А.В.

Национальный Исследовательский Томский политехнический университет,
634050, Россия, г. Томск, пр. Ленина, 30
E-mail: asg19@tpu.ru

***Ключевые слова**: Web-интерфейс, .net Framework, сайт, сервер WITSML, WITSML агент.*

***Key words**: Web-interface, .net Framework, site, server WITSML, WITSML agent.*

С появлением технологии цифрового месторождения одним из основных элементов Российской интерактивной системы управления жизненным циклом нефтегазового месторождения Unofactor является программный продукт Wellook, в основе которого применяются самые актуальные международные стандарты, включая открытый стандарт обмена данными WITSML компании Energistics.

WITSML (Wellsite Information Transfer Standard Markup Language) – язык разметки по передаче скважинных данных, в основе которого заложена технология XML, имеющая ценность для бизнеса за счет эффективных стандартных протоколов обмена данными.

Данная программная платформа включает в себя следующие компоненты:

- WITSML Server – сервер для обработки данных о бурении в формате передачи и хранения данных WITSML;
- WITSML Agent – программный модуль, конвертирующий данные поступающие из контроллеров бурения в единый формат WITSML;
- Web-interface – система управления агрегированными данными на WITSML сервере.