ОБЪЕКТНО-РЕЛЯЦИОННЫЕ ОТОБРАЖЕНИЯ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA И АВТОМАТИЗАЦИЯ ИХ ПОСТРОЕНИЯ

Юрченко В.А.,
выпускник ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь
Научный руководитель – Ермоченко С.А., канд. физ.-мат. наук

В настоящее время язык программирования Java используется чаще всего создания web-приложений, активно использующих базы данных (БД). При этом большая часть таких БД функционирует под управлением реляционных систем управления базами данных. В тоже время приложение манипулирует данными в объектном представлении. Для получения данных из реляционной БД и преобразования их в объектное представление в Java-приложении вручную приходится писать большое количество шаблонного кода. Для решения этой проблемы разрабатываются специальные библиотеки, которые берут на себя большую часть работы с БД, при этом связывание (Mapping) происходит посредством создания конфигурационных файлов, в которых описывается соответствие объектов и таблиц БД. Такие библиотеки (Frameworks) обычно называют Object-Relational Mapping Frameworks (ORM-Frameworks) [1]. Недостатком большинства существующих ORM-Frameworks является необходимость конфигурировать библиотеку посредством сложных конфигурационных файлов или аннотаций, что снижает эффективность использования библиотеки в случае работы со сложной предметной областью [2].

Целью данного исследования является разработка собственной простейшей ORM-библиотеки и графического редактора схемы данных.

Актуальность исследования заключается в том, что предлагаемое решение избавляет программиста от рутинных операций с конфигурированием, делает моделирование предметной области более наглядным за счёт графического представления схемы БД и в целом повышает эффективность использования БД.

Материал и методы. ORM-библиотека должна, в соответствии с общепринятыми требованиями, реализовывать базовые операции с данными в БД: создание, чтение, обновление и удаление [3]. Библиотека проектируется в соответствии с принципами объектно-ориентированного проектирования [4]. Графическое приложение разрабатывается в соответствии с общепринятыми принципами организации пользовательского интерфейса. Для реализации основного функционала ORM-библиотеки используется возможность ретроспекции в языке программирования Java [5].

Результаты и их обсуждение. Для возможности управления сущностями через проектируемую ORM-библиотеку было принято решение использовать наследование от базового класса BaseEntity, описанного в этой библиотеке. В классе BaseEntity создано только одно поле для хранения первичного ключа, тип которого определяется в подклассах через механизм обобщений. Что позволяет использовать первичные ключи разных типов, в том числе составных.

Далее разработаем структуру конфигурационного файла для ORM-библиотеки. Данный файл было решено построить на базе формата XML.

Листинг 1. Пример конфигурационного файла для хранения в БД авторов и книг

```
<database name="myorm" user="root" pass="root" url="jdbc:mysql://localhost/myorm"
 driver="com.mysql.jdbc.Driver">
 <table name="books" primary-key="id" entity_name="Book">
 <field name="id"/>
 <field name="name"/>
 <field name="year"/>
 <field name="author_id"
 entity_field="authorId"
 foreign-key-entity="authors"/>
 </table>
 <table name="authors" primary-key="id" entity_name="Author">
 <field name="id"/>
 <field name="name"/>
 <field name="surname" entity_field="surName"/>
 <field name="year"/>
```

```
 </table>
</database>
```

Как видно из приведённого листинга, ORM-библиотека поддерживает лишь базовые возможности: конфигурирование соответствия классов и таблиц БД, моделирование внешних связей один-ко-многим (связи один-к-одному и многие-ко-многим, как не столь популярные, не рассматривались) и общее конфигурирование доступа к БД. Библиотекой не поддерживается «ленивая» загрузка данных, использование пула соединений и кэширование данных в соответствии с шаблоном проектирования «Identity Map» («Карта идентификаторов»). Эти возможности планируется добавить в дальнейшем.

Далее в библиотеке через ретроспекцию (Reflection API) реализуется доступ к БД, формирование SQL-запросов на сервер в соответствии с конфигурационным файлом, получение информации и представление её в виде объектов соответствующих Java-классов. После этого классы можно использовать на уровне программирования бизнес-логики приложения, который уже будет абстрагирован он особенностей работы постоянного хранилища.

Следующим шагом в данной работе стала разработка графического редактора, позволяющего создавать схему БД более удобно и наглядно без необходимости самостоятельного написания описанного выше конфигурационного файла. Общий интерфейс разработанного приложения с описанной выше структурой БД приведён на рисунке 1:

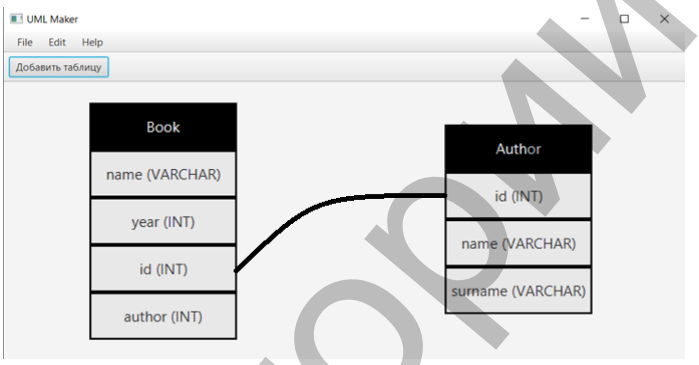

Рисунок 1. Общий вид графического редактора

Для построения интерфейса приложения использовалась кроссплатформенная библиотека JavaFX, что позволяет использовать приложения в самых разных операционных системах.

Описанное приложение помимо создания конфигурационного файла позволяет автоматически сгенерировать исходный код на языке программирования Java для классов-сущностей (что избавляет от необходимости вручную создавать многочисленные поля классов и методы доступа к этим полям). Также редактор генерирует SQL-скрипты для создания БД с необходимой структурой. В будущем в редактор можно будет добавить возможность сразу выполнять эти сгенерированные SQL-скрипты, ещё более автоматизируя рутинные действия разработчиков.

Заключение. Разработанная ORM-библиотека и графический редактор демонстрируют концепцию быстрой разработки (Rapid Application Development – RAD) при использовании ORM-Frameworks. Что позволяет повысить эффективность работы, избавившись от рутинных операций и возможных ошибок. Разработанное решение является демонстрационным, но может быть доработано и использоваться при решении практических задач. Также планируется добавить возможность работы с конфигурационными файлами существующих ORM-Frameworks.

1. Википедия: ORM [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ORM. – Дата доступа: 15.05.2018.
2. Документация Oracle: JPA [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html. – Дата доступа: 15.05.2018.
3. Википедия: CRUD [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/CRUD. – Дата доступа: 15.05.2018.
4. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс. Быстрая разработка программ. Принципы, примеры, практика – Вильямс, 2004
5. Введение в Java Reflection API [Электронный ресурс]. – Режим доступа: http://www.quizful.net/post/java-reflection-api. – Дата доступа: 15.05.2018.