7. *Литтл Р. Дж. А, Рубин Д. Б.* Статистический анализ данных с пропусками // М.: Финансы и статистика. 1990.
8. *Харин Ю. С.* Оптимальность и робастность в статистическом прогнозировании // Мн.: БГУ. 2008.
9. *Хьюбер Дж. П.* Робастность в статистике. М.: Мир. 1984.

СИСТЕМА ОБЪЕКТНО-РЕЛЯЦИОННОГО ОТОБРАЖЕНИЯ

В. К. Агекян

ВВЕДЕНИЕ

Большинство современных языков программирования используют объектно-ориентированный подход. В программах, использующих доступ к базам данных (БД), у программистов чаще всего возникает необходимость создавать классы, объекты которых соответствуют записям в соответствующей классу таблице БД. При этом все изменения данных, введенных конечным пользователем в прикладной программе, записываются сначала в эти объекты, а затем эти изменения синхронизируются с БД [1]. То же самое удобно делать при чтении информации из БД: сначала инициализировать объекты, наполняя их содержимым из таблиц БД, а затем манипулировать свойствами этих объектов. Набор действий, необходимый для синхронизации объектов в памяти и записей инвариантен относительно самих классов, поэтому целесообразно создать набор стандартных процедур, позволяющих записывать данные из таблиц БД в объекты программы и синхронизировать изменения объектов с БД. Эту задачу решают системы объектно-реляционного отображения.

Задачей данной работы является реализация собственной системы объектно-реляционного отображения на языке программирования Java, предоставляющий следующий функционал для работы с БД:

- возможность отображения Java классов на соответствующие сущности БД;
- возможность выполнения основных операций по добавлению и извлечению данных из БД;
- реализация объектно-ориентированного метода генерации динамических запросов;
- поддержка механизма транзакций.

Областью применения разработанной системы является разработка приложений, написанных на языке программирования Java, выполняющие работу с реляционными БД.

ОПИСАНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ

Целесообразно обособить код SQL от бизнес-логики, разместив его в специальных классах. Необходимо изолировать модель предметной области от БД, возложив на промежуточный слой всю полноту ответственности за отображение объектов домена в сущности БД. Подобный преобразователь данных обслуживает все операции загрузки и сохранения информации, инициируемые бизнес-логикой, и позволяет независимо варьировать как модель предметной области, так и схему БД.

Разработанная система удовлетворяет базовым требованиям, предъявляемым к системам объектно-реляционного отображения [3]. Архитектура системы представлена на рисунке 1.

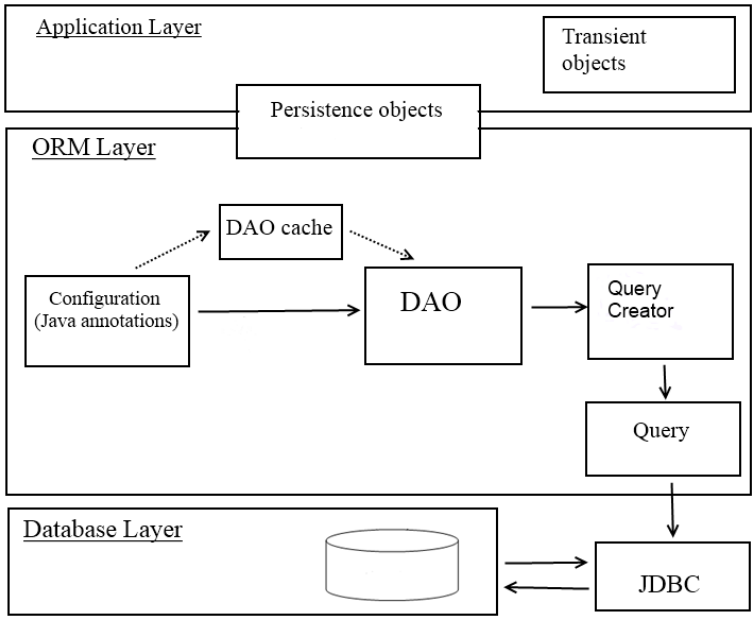

Рис. 1. Архитектура разработанной системы объектно-реляционного отображения

Для реализации системы использовался объектно-ориентированный язык высокого уровня Java. Она, как показано на рисунке, является промежуточным слоем между уровнем приложения (Application layer) и уровнем реляционной БД (Database layer).

Объектно-ориентированный интерфейс для синхронизации объектов Java с базой данных предоставляют классы, реализующие шаблон проектирования Data Access Object (DAO) [2]. Если DAO-класс уже был сгенерирован в ходе работы программы, то вместо повторной генерации объект будет браться из кэша.

Класс QueryCreator непосредственно генерирует SQL-запросы к БД на основании вызываемых методов и передаваемых параметров. Сгенерированные запросы передаются базе данных, используя классы под-

ключенного JDBC-драйвера (PreparedStatement, CallableStatement) [4]. Для журналирования используется библиотека log4j.

Пример. По методу

public T queryForId(int id) throws SQLException

формируется SQL-запрос:

SELECT * FROM tableName WHERE idColumn = id;

СРАВНЕНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ С СУЩЕСТВУЮЩИМИ АНАЛОГАМИ

Сравним реализованную систему с такими системами объектно-реляционного отображения, как Hibernate, EclipseLink, OpenJPA по объему потребляемой памяти. Тестовая программа содержит 10 потоков для вставки данных и 10 потоков для выборки данных. Анализируется объем потребляемой памяти за фиксированный промежуток времени (3 минуты). Результаты проведенных экспериментов представлены в таблице 1.

Таблица 1

Результаты проведенных экспериментов

	Среднее количество памяти, используемое JVM, Mb	Максимальное количество памяти, используемое JVM, Мб	Количество загруженных в память Java-классов
EclipseLink	43	68	3079
Hibernate	85	130	3863
OpenJPA	62	96	3177
Реализованная система	29	61	1725

Исходя из результатов эксперимента, можно сделать вывод о том, что реализованная система потребляет меньший объем памяти в сравнении с рассмотренными аналогами. Это обусловлено, прежде всего, меньшим количеством Java-классов, загружаемых JVM при работе системы.

Также выполнена оценка производительности системы в сравнении с использованием непосредственно JDBC для доступа к данным. Результаты эксперимента представлены в таблице 2.

Таблица 2

Результаты проведенных экспериментов

	Общее количество запросов	Количество запросов вставки	Количество запросов выборки
JDBC	402547	195418	207129
Реализованная система	327952	158378	169574

Исходя из результатов эксперимента, можно сделать вывод о том, что при использовании непосредственно JDBC при работе с СУБД мы полу-

чаем некоторый незначительный выигрыш в производительности в сравнении с использованием разработанной системой. Это обусловлено тем, что разработанная система является надстройкой над JDBC и использует JDBC API на более низком уровне для передачи сгенерированных SQL-запросов в СУБД. В то же время разработанная система позволяет выполнять отображение объектов Java-классов на таблицы СУБД, а также обеспечивает механизм формирования запросов в СУБД при помощи объектно-ориентированного подхода, вызывая методы Java-классов. Это избавляет от необходимости написания запросов на языке SQL и вызова методов JDBC. Данные преимущества значительно сокращают время разработки, объем кода и поддержку программного продукта, использующего СУБД в качестве хранилища данных.

ЗАКЛЮЧЕНИЕ

В данной работе рассмотрена проблематика совместного использования реляционной БД и объектно-ориентированного языка программирования. Предложена собственная реализация системы объектно-реляционного отображения, соответствующая основным требованиям, предъявляемым к подобным системам.

Актуальность данной работы обусловлена широким расспространением объектно-ориентированной методологии разработки прикладных систем и одновременно доминирующим положением на рынке реляционных БД. Разработанная система является промежуточным слоем между объектно-ориентированным языком программирования Java и реляционными БД, предоставляя необходимые объектно-ориентированные интерфейсы к данным, хранимым под управлением реляционных систем управления БД (СУБД). В этом случае удачно сочетаются преимущества реляционных СУБД с преимуществами объектно-ориентированного подхода.

Литература

1. *Atkinson M.* The Object-Oriented Database System Manifesto. Kyoto, Japan. 1992.
2. *Фаулер М.* Архитектура корпоративных программных приложений. М., 2007.
3. *Hambrick G.* Persistence in the Enterprise: A Guide to Persistence Technologies. IBM Press. 2008.
4. *Блинов И. Н. Промышленное программирование.* Минск, 2007.