УДК 004.62 DOI: 10.30748/soi.2017.148.10

Ю.Э. Парфенов

Харьковский национальный экономический университет, Харьков

ОСОБЕННОСТИ РАЗРАБОТКИ JAVA-ПРИЛОЖЕНИЙ ДЛЯ MONGODB

В работе рассмотрены вопросы применения языка программирования Java для разработки приложений для документно-ориентированной СУБД MongoDB. Приведен пример использования Java-драйвера для доступа к данным, проанализированы его достоинства и недостатки. Описаны основы разработки Java-приложений, выполняющих CRUD-операции с данными на базе фреймворка объектно-документного отображения Morphia.

Ключевые слова: *СУБД, NoSQL, документ, MongoDB, CRUD-операции, Java-драйвер, Morphia.*

Введение

Большинство приложений в процессе своей работы сохраняет данные в том или ином виде.

В простейшем случае это может быть один или несколько текстовых файлов, в которых каждая запись находится в отдельной строке, а в качестве разделителей ее полей могут использоваться пробелы, запятые и т.д. Такая «плоская» организация данных не предусматривает структурных взаимосвязей между ними, что существенно затрудняет работу с данными.

Более сложные программы для хранения данных, как правило, используют базу данных, взаимодействие с которой осуществляется с помощью интерфейса соответствующей СУБД.

Любая СУБД основана на некоторой модели данных. До недавнего времени наиболее распространенной была реляционная модель данных.

Реляционная модель данных включает следующие аспекты [1]:

структурный аспект – данные представлены в виде набора таблиц;

аспект целостности – таблицы удовлетворяют определенным ограничениям целостности;

аспект манипулирования – пользователю доступны операторы манипулирования таблицами.

Однако, реляционные СУБД имеют ряд недостатков:

не обеспечивают естественного представления данных, имеющих иерархическую структуру;

поля кортежа могут содержать лишь атомарные значения;

недостаточная производительность, так как для выполнения большинства запросов требуется несколько операций соединения.

В начале 2000-х годов с развитием web-технологий и социальных сервисов разработчики соответствующих приложений столкнулись с задачами, для которых традиционные реляционные СУБД оказались либо слишком дороги, либо недостаточно производительны. Например, одной из проблем является тот факт, что функционирование подобных приложений часто связано с обработкой больших объемов быстроменяющихся слабоструктурированных или неструктурированных данных.

Это привело к появлению новых подходов к хранению и обработке данных, одним из которых является NoSQL.

Таким образом, представляется целесообразным рассмотреть аспекты разработки приложений для СУБД, базирующихся на NoSQL-подходе.

Основная часть

NoSQL-подход, в частности, обеспечивает:

гибкую модель данных;

малое время отклика и высокую производительность;

высокую масштабируемость.

В настоящее время существует множество типов NoSQL-решений. Основными из них являются [2; 3]:

хранилища «ключ-значение»;

хранилища семейств колонок;

документо-ориентированные СУБД;

базы данных на основе графов.

Вследствие их гибкости, производительности и простоты использования наиболее популярными являются документо-ориентированные СУБД.

Они хорошо подходят для целого ряда вариантов использования [4], таких как:

поддержка бек-энда веб-сайтов с большим объемом операций чтения и записи;

управление типами данных с переменными атрибутами;

отслеживание изменяющихся метаданных;

обработка данных, представленных в формате JSON.

Согласно мировому рейтингу DB-Engines Ranking [5] на февраль 2017 года первое место среди NoSQL-решений и пятое место среди всех извест-

ных СУБД занимает MongoDB – свободно-распространяемая документо-ориентированная СУБД разработки компании MongoDB Inc. Текущей версией MongoDB является 3.4. Дистрибутив доступен для загрузки по адресу https://www.mongodb.com/download-center#community.

Приложения, взаимодействующие с MongoDB, можно разрабатывать на различных языках программирования (в настоящее время реализована поддержка около двенадцати языков). В данной работе рассматриваются особенности разработки приложений для MongoDB на языке программирования Java.

Использование Java-драйвера для MongoDB

Доступ к СУБД MongoDB из Java-приложения осуществляется с использованием соответствующего драйвера.

Библиотеки драйвера можно добавить в проект одним из двух способов. Первый способ заключается в загрузке с Web-сайта MongoDB и непосредственном импорте файла mongodb-driver-3.4.2.jar. При этом также придется аналогичным образом подключить его зависимости: bson-3.4.2.jar и mongodb-driver-core-3.4.2.jar

Рекомендуемым же способом подключения библиотеки драйвера к проекту является использование фреймворка для автоматизации сборки проектов, например, Apache Maven. Для этого в секцию описания зависимостей Maven-проекта в файле pom.xml необходимо добавить код:

```
<dependency>
   <groupId>org.mongodb</groupId>
   <artifactId>mongodb-driver</artifactId>
    <version>3.4.2</version>
 </dependency>
```

Java-драйвер версии 3.4.2 предоставляет API для доступа к СУБД MongoDB, включающий несколько сотен классов, интерфейсов и других элементов, которые распределены по более чем тридцати пакетам.

Основными абстракциями Java-драйвера, используемыми в любой программе, являются классы MongoClient и Document, а также интерфейсы MongoDatabase и MongoCollection.

Объект класса MongoClient представляет собой пул подключений к базе данных, а объект класса Document – документ базы данных.

Ссылки типов MongoDatabase и MongoCollection используются при полиморфных вызовах методов классов, реализующих данные интерфейсы. Эти методы возвращают ссылки на экземпляры, представляющие базу данных и коллекцию документов соответственно.

Для выполнения CRUD-операций с данными в интерфейсе MongoCollection имеется ряд методов.

Для извлечения документов коллекции используется метод find, поведение которого сравнимо с SQL-запросом SELECT в реляционных СУБД. Перегруженные версии метода find позволяют находить как все документы коллекции, так и документы, соответствующие заданным критериям фильтрации.

Методы для сохранения данных в целом аналогичны SQL-командам INSERT, UPDATE и DELETE.

Для вставки документов в коллекцию базы данных предназначены методы insertOne (вставка одного документа) и insertMany (вставка нескольких документов). Для обновления существующих документов коллекции базы данных используются методы updateOne и updateMany, а для удаления – deleteOne и deleteMany. Для замены документа коллекции на новый документ существует метод replaceOne. Методы обновления и замены документов коллекции имеют интересную особенность: они могут выполнять не только свои «прямые обязанности», но и осуществлять вставку документов в коллекцию. Это происходит, если в коллекции не существует документов, удовлетворяющих критериям обновления или замены.

Java-драйвер также поддерживает все операции агрегирования данных MongoDB (см. https://docs.mongodb.com/manual/meta/aggregation-quick-reference/). Это позволяет получать сводную информацию из множества документов, тем самым способствуя повышению эффективности обработки информации.

Фрагмент простейшей консольной программы, взаимодействующей с СУБД MongoDB и выполняющей CRUD-операции, приведен ниже:

```
public static void main(String[] args) {
MongoClient c = new MongoClient();
//test – имя базы данных
MongoDatabase db = c.getDatabase("test");
//animals – имя коллекции документов
MongoCollection<Document> animals =
db.getCollection("animals");
//создание двух новых документов
Document animal1 = new Document("animal", "rat");
Document animal2 = new Document("animal", "cat");
ArrayList<Document> documents = new
ArrayList<Document>();
documents.add(animal1);
documents.add(animal2);
//вставка документов в базу данных
animals.insertMany(documents);
//обновление документа в базе данных
animals.updateOne(animal1, set("animal", "dog"));
```

```
//вывод на консоль первого документа коллекции
animals в формате JSON
System.out.println(animals.find().first().toJson());
//удаление документа в базе данных
animals.deleteOne(animal2);
//закрытие подключения к базе данных
 c.close(); }
```

Таким образом, в отличие от реляционных СУБД, для NoSQL-решений в целом и MongoDB, в частности, не существует стандартного языка запросов, подобного SQL. Вместо этого компании-разработчики предоставляют некоторый специализированный API для своих продуктов, позволяющий выполнять операций с данными. Для MongoDB этот API является составной частью соответствующего драйвера.

Такой подход имеет как достоинства, так и недостатки. С одной стороны, драйвер предоставляет достаточно низкоуровневый API, что обеспечивает широкие возможности по разработке приложений для работы с данными.

С другой стороны, такие программные системы неизбежно будут содержать большое количество кода, непосредственно не связанного с их бизнес-логикой. Это с большой вероятностью потребует увеличения затрат, как минимум временных, на разработку подобных систем.

Использование фреймворка Morphia

Одним из вариантов повышения эффективности разработки приложений для MongoDB является использование одного из фреймфорков для объектно-документного отображения данных. Такие фреймворки обеспечивают прозрачное отображение данных из объектной модели языка программирования в документную модель СУБД и обратно.

Для взаимодействия с MongoDB из Java-приложения существует около двадцати подобных фреймворков. Одним из наиболее популярных является Morphia [6]. Morphia – открытое программное обеспечение, разработка которого поддерживается компанией MongoDB Inc. Текущей версией фреймворка является 1.3.2.

Morphia – это программная оболочка для Java-драйвера, рассмотренного выше. Таким образом, данный фреймворк обеспечивает абстракции высокого уровня для преобразования объектов предметной области в документы MongoDB, в то же время позволяя в случае необходимости использовать низкоуровневые возможности базового драйвера.

Подключить фреймворк Morphia к проекту можно разными способами, в частности, с помощью Maven. При этом файл pom.xml должен содержать следующую зависимость:

```
<dependency>
<groupId>org.mongodb.morphia</groupId>
    <artifactId>morphia</artifactId>
    <version>1.3.2</version>
</dependency>
```

Для отображения объектов предметной области на соответствующие документы MongoDB прежде всего нужно описать один или несколько классов сущностей. Это обычные POJO-классы с аннотациями для самих классов и их элементов.

Все аннотации Morphia API логически принадлежат пакету org.mongodb.morphia.annotations.

Рассмотрим основные аннотации.

@Entity – задает имя коллекции документов MongoDB. Применяется к классу сущности, объекты которого будут храниться в коллекции документов. Аннотация опциональна, в случае ее отсутствия имя коллекции будет совпадать с именем класса.

@Property – опциональная аннотация для поля класса сущности. Предназначена для задания имени поля документа, на который отображается поле класса. По умолчанию оно совпадает с именем поля класса.

@Id – аннотация для поля класса сущности, которое будет соответствовать полю _id документа базы данных. Требуется для полей классов, объекты которых отображаются на документы MongoDB, за исключением внедренных в другие документы.

@Reference – аннотация, указывающая, что данное поле класса сущности будет соответствовать ссылке на документ, находящийся в другой коллекции.

@Embedded – может применяться к классу сущности или его полю. Для класса эта аннотация определяет, что его объекты будут отображаться на документ, внедренный в другой документ. Для поля класса – что оно будет соответствовать ссылке на внедренный документ.

@Validation – аннотация для класса сущности. Позволяет определить правила валидации документов коллекции перед выполнением любой операции записи в базу данных. Далее приведен пример использования аннотации @Validation для проверки того, что значение поля не превышает заданной величины:

```
@Validation("{ num : { $le : 75 } }")
public class EntityClass {
   private int num;
}
```

Для использования фреймворка Morphia также необходимо создать объект класса org.mongodb.morphia.Morphia, который отвечает за объектно-документное отображение данных. Кроме того требуется инициализировать ссылку на интер-

фейс org.mongodb.morphia.DataStore, отвечающий за взаимодействие с MongoDB:

```
Morphia morphia = new Morphia();
// использовать пакет, содержащий классы сущностей
morphia.mapPackage("morphia.entity");
//mydb – имя базы данных
Datastore ds = morphia.createDatastore(new MongoClient(), "mydb");
```

CRUD-операции с объектами классов сущностей выполняются путем полиморфных вызовов соответствующих методов интерфейса Datastore:

```
//создание объектов класса сущности User
User user1 = new User("Alex", "Cross");
User user2 = new User("Ivan", "Shapiro");
//сохранение объектов в базу данных
ds.save(user1);
ds.save(user2);
//извлечение всех документов из базы данных
List<User> users = ds.createQuery(User.class).asList();
//обновление документа базы данных
UpdateOperations<User> query =
ds.createUpdateOperations(User.class)
.set("firstname", "John");
ds.update(user2, query);
//удаление документа базы данных
ds.delete(user1);
```

Выводы

Итак, хотя Java-драйвер позволяет разрабатывать приложения для MongoDB, его непосредственное использование имеет ряд недостатков. Для повышения эффективности процесса разработки рекомендуется использовать фреймворк объектно-документного отображения данных, одним из которых является Morphia.

Список литературы

1. Date J. An Introduction to Database Systems / J. Date. – Pearson Education Inc., 2004. – 1024 p.

2. Types Of NoSQL Database Management Systems. [Electronic resource]. – Access mode: https://www.mongodb.com/scale/types-of-nosql-database-management-systems.

3. List of NoSQL Databases. [Electronic resource]. – Access mode: http://nosql-database.org/.

4. Sullivan D. NoSQL for Mere Mortals / D. Sullivan. – Addison-Wesly, 2015. – 552 p.

5. DB-Engines Ranking. [Electronic resource]. – Access mode: http://db-engines.com/en/ranking.

6. The Java Object Document Mapper for MongoDB. [Electronic resource]. – Access mode: https://mongodb.github.io/morphia/.

Поступила в редколлегию 23.02.2017

Рецензент: д-р техн. наук, проф. В.А. Гороховатский, Учебно-научный институт ГВУЗ «Університет банківської справи», Харьков.

ОСОБЛИВСТІ РОЗРОБЛЕННЯ JAVA-ЗАСТОСУНКІВ ДЛЯ MONGODB

Ю.Е. Парфьонов

В роботі розглянуто питання застосування мови програмування Java для розробки застосунків для документно-орієнтованої СКБД MongoDB. Наведено приклад використання Java-драйвера для доступу до даних, проаналізовані його достоїнства і недоліки. Описано основи розроблення Java-застосунків, що виконують CRUD-операції з даними на базі фреймворка об'єктно-документного відображення Morphia.

Ключові слова: *СКБД, NoSQL, документ, MongoDB, CRUD-операції, Java-драйвер, Morphia.*

THE PARTICULARITY OF APPLICATION DEVELOPMENT FOR MONGODB

Y.E. Parfonov

The paper discusses the use of the Java programming language to develop applications for document-oriented DBMS MongoDB. Presented an example of using Java MongoDB driver to access data, analyzed its strengths and weaknesses. Described the basics of development Java application that perform CRUD operations with data based on the document-object mapping framework Morphia.

Keywords: *DBMS, NoSQL, document, MongoDB, CRUD operations, Java driver, Morphia.*