УДК 004.6

МОДЕЛИРОВАНИЕ ДАННЫХ В NOSQL СУБД НА ПРИМЕРЕ РАЗРАБОТАННОГО ВЕБ-ПРИЛОЖЕНИЯ «ИНФОРМАЦИОННЫЙ АГРЕГАТОР БАНКОВ РЕСПУБЛИКИ БЕЛАРУСЬ»

Р.П. ГУСЕВСКИЙ
(Представлено: Д.В. ПЯТКИН)

Проанализированы отличия моделирования в NoSQL и SQLСУБД на примере разработанного веб-приложения «Информационный агрегатор банков Республики Беларусь», созданного с использованием одной из популярных СУБД MongoDB.

К настоящему времени человечеством накоплено поистине гигантское количество информации об объектах и явлениях. Но эта информация не лежит мертвым грузом, она хранится в электронном виде и используется в базах данных. Базы данных – это часть информационных систем – программно-аппаратных комплексов, осуществляющих хранение и обработку огромных информационных массивов.

База данных представляет собой определенным образом структурированную совокупность данных, совместно хранящихся и обрабатывающихся в соответствии с некоторыми правилами. Как правило, база данных моделирует некоторую предметную область или ее фрагмент. Программа, производящая манипуляции с информацией в базе данных, называется СУБД (система управления базами данных). Основными составляющими информационных систем, построенных на основе баз данных, являются файлы БД, СУБД и программное обеспечение (клиентские приложения), позволяющие пользователю манипулировать информацией и совершать необходимые для решения его задач действия. Базы данных как способ хранения больших объемов информации и эффективного манипулирования ею используются практически во всех областях человеческой деятельности.

Выбор NoSQLСУБДMongoDB

Для того чтобы хранить нужные данные, следует определиться с выбором БД, для реализации информационного агрегатора была выбрана база данных MongoDB, которая хорошо интегрируется с веб-приложениями. Это обусловлено тем, что MongoDB хранит данные в формате JSON. Формат JSON имеет хорошую поддержку в Javascript, поэтому работа с JSON хорошо отразится на производительности данного приложения.

MongoDB является NoSQL базой данных. NoSQL базы данных не работают с реляционными моделями. Существует много различных решений, каждое из которых работает по-своему и служит специфической цели. Эти безсхемные решения снимают ограничения с формирования сущностей и допускают хранения данных в виде ключ-значение.

NoSQL базы данных не используют общий формат запроса, такой как SQL в реляционных базах данных. Каждое NoSQL решение использует собственную систему запросов.

В MongoDB данные имеют гибкую схему хранения документов в одной коллекции. Это означает, что документы не должны иметь одинаковый набор полей или структуру. Общие поля в документах коллекции могут хранить различные типы данных [1].

Для того чтобы моделировать данные в MongoDB, нужно:

1. Комбинировать объекты в одном документе, если есть необходимость использовать их вместе.
2. Дублировать данные, так как время вычислений «дороже», чем место на диске.
3. Объединять данные во время записи, а не во время чтения.
4. Использовать сложные агрегации в схеме.
5. Оптимизировать схему под наиболее частые случаи.

Наглядная разница моделирования данных в NoSQL от реляционной базы данных

Чтобы более детально понять, в чем разница между моделированием в NoSQL базы данных и реляционной БД, рассмотрим следующий **пример** [2].

Предположим, нужно создать базу данных для хранения проектов и разработчиков, которые работают над этими проектами. Следует учесть, что каждый проект имеет уникальное имя и описание, и имеет много разработчиков.

Если использовать стандартную реляционную БД, то наша структура таблиц, примерно, имела вид, который представлен на рисунке 1.

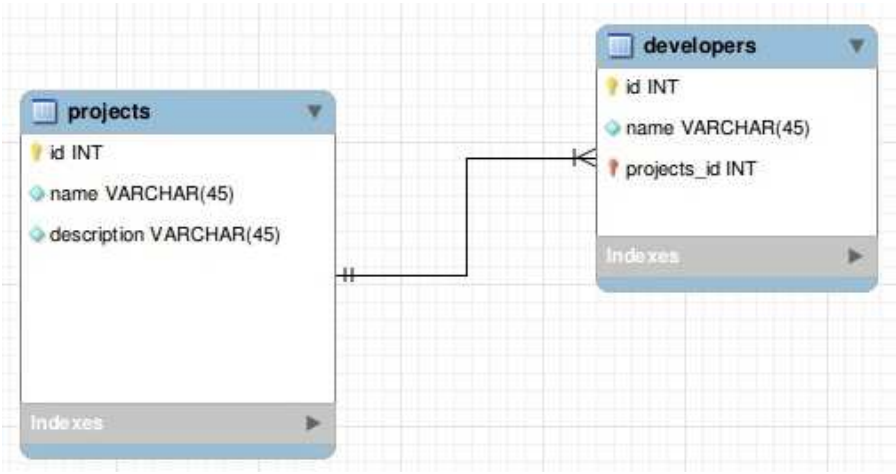

Рисунок 1. – Структура таблиц, рассматриваемая в примере

В MongoDB данная схема выглядела бы в виде структуры в формате JSON (листинг 1).

Листинг 1 – Пример схемы в MongoDB

```
{
_id: PROJECT_ID
name: NAME_OF_PROJECT,
developers: [
    {
name:'DEVELOPER_NAME',
    },
    {
name: 'DEVELOPER_NAME'
}
  ]
}
```

Анализ и моделирование данных при разработке веб-приложения в MongoDB

При анализе функциональности веб-приложения выявлены следующие сущности:

1. Сущность *Обменный пункт* содержит данные о конкретном обменном пункте. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.

2. Сущность *Отделение* описывает информацию о конкретном отделении банка. Характеризуется наименованием, описанием, координатами: широты, долготы, городом, адресом.

3. Сущность *Банкомат* содержит конкретную информацию о банкомате, принадлежащем конкретному банку. Характеризуется адресом, точной локацией, наименованием, временем работы, возможной валютой для снятия денег, координатами: широты, долготы.

4. Сущность *График Валют* содержит информацию для построения графиков валют. Характеризуется объектами валюты.

5. Сущность *Валюта* содержит описание валюты. Характеризуется наименованием валюты, ее кодом, массивом продажи и покупки данной валюты.

6. Сущность *Ссылки* содержит полезные ссылки банка. Характеризуется перечнем ссылок для банка.

Опираясь на рекомендации по моделированию данных в MongoDB, а также на результат краткого анализа функциональной части проекта смоделирована общая структура для хранения информации о банке (листинг 2).

Листинг 2 – Общая структура для хранения информации о банке

```
{
nameBank: "ЗАО «БСББанк»",
   description: "Краткоеописаниебанка",
shortName : "BSBBank",
linkNews: "",
linkBankCard: "",
linkCredit: "",
linkInternetBanking: "",
linkInsurance: "",
pathImg: "../img/banksLogo/bsbbank-logo.png",
affilate: [],
exchangeOffice: [],
bankomats: [],
graphicCurrency: [],
mainAddress: "",
mainNumbers: "+375-17-306-20-40\n+375-29-306-20-40\n"+"+375-33-306-20-40",
mainlinkAddress: "https://www.bsb.by/"
}
```

Из описания структуры видно, что каждое поле принимает значение различного типа, так как MongoDB хранит информацию в формате JSON [3], то в данном формате в качестве значений могут быть использованы:

1. Объект – это неупорядоченное множество пар ключ – значение, заключенное в фигурные скобки «{}». Ключ описывается строкой, между ними значением стоит символ «:». Пары ключ – значение отделяются друг от друга запятыми.
2. Массив (одномерный) – это упорядоченное множество значений. Массив заключается в квадратные скобки «[ ]». Значения разделяются запятыми.
3. Число.
4. Литералы – true, false и null.
5. Строка – это упорядоченное множество из нуля и более символов, заключенное в двойные кавычки.

Исходя из того, что некоторые поля структуры (объекта) принимали в качестве значений массив, то данные таких полей представляли собой схожие объекты, которые нацелены на выполнение конкретного функционала.

Заключение

Авторами придумана при помощи длительного анализа общая структура, которая представляет собой объект, который описан выше (см. листинг 2). Данная структура хорошо подойдет для использования ее, как и для работы с графиками, так и для использования отображения местоположений банков, отделений и предоставления общей информации о банке. Выбор MongoDB в качестве СУБД приложения выбрана исходя из следующих плюсов:

- данная система хранения данных основана на принципе хранения документов в BSON (Binary JSON) формате, что делает ее хорошо интегрируемой с большей частью веб-приложений;
- более быстрое извлечение простых структур данных;
- хранение неструктурной информации;
- индекс по любому признаку;
- богатые запросы, репликация и высокая доступность [4].

ЛИТЕРАТУРА

1. METANIT.COM Сайт о программировании [Электронный ресурс]. – Режим доступа: https: //metanit. com/nosql/mongodb/1.1.php. – Дата доступа: 17.09.2017.
2. Руководство по MongoDB. Моделирование данных [Электронный ресурс]. – Режим доступа: http: //proselyte.net/tutorials/mongodb/data_modeling/. – Дата доступа: 19.09.2017.
3. Википедия : свободная энцикл. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/ JSON/. – Дата доступа: 19.09.2017.
4. W3IILatestwebdevelopmenttutorials [Электронный ресурс]. – Режим доступа: http://www.w3ii.com/ru/ mongodb/mongodb_advantages.html. – Дата доступа: 22.09.2017.