УДК 004.652
DOI: 10.15827/0236-235X.125.063-067
Дата подачи статьи: 21.03.18
2019. Т. 32. № 1. С. 063–067

Разработка концепции миграции данных между реляционными и нереляционными системами БД

Ю.А. Королева [1], *к.т.н., ассистент, jakoroleva@corp.ifmo.ru*
В.О. Маслова [1], *студент, victoria_95m@mail.ru*
В.К. Козлов [1], *студент, 172652@niuitmo.ru*

[1] *Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (Университет ИТМО), г. Санкт-Петербург, 197101, Россия*

В данной работе исследуются реляционные и нереляционные подходы к построению, хранению и извлечению данных.

В настоящее время все информационные и информационно-аналитические системы не обходятся без использования БД. Данные системы должны обрабатывать, читать, записывать определенные наборы данных, которые нужно упорядочивать, структурировать и хранить.

Для многих компаний актуальной проблемой является выбор подходящих БД и системы управления, от которых в дальнейшем будут зависеть производительность, надежность, безопасность, особенности поддержки, разработки и другие характеристики работы. В одной информационной системе компании обычно применяются несколько моделей данных, это обосновано разноплановостью характера манипуляции используемых в непосредственной работе данных. Например, для задач, где необходимы полная консистентность данных и транзакционный контроль, используют реляционную БД, в то время как аналитические, агрегированные или метаданные могут храниться в БД NoSQL. Данное разделение зачастую необходимо для наиболее эффективного функционирования конечного продукта.

В процессе работы были выявлены самые востребованные системы управления БД для обоих подходов к построению БД, проанализированы их особенности, достоинства и недостатки. На основании внутреннего устройства реляционных и нереляционных БД предложена схема преобразования данных из одной модели в другую как первый этап подготовки данных к прозрачной миграции между системами.

Теоретической основой исследований являются отечественные и зарубежные публикации на тему моделей данных, а также компьютерных технологий.

Ключевые слова: *разнородные информационные системы, перенос данных, синхронизация данных, NoSQL, системы управления БД, миграция данных.*

Большую роль в информационных системах с 80-х годов и по сей день играют *реляционные БД* (РБД). Они хранят хорошо структурированные данные в виде таблиц с заданными столбцами определенного типа данных. В силу строгой организации требований от разработчика приложений необходимо четкое структурирование используемых данных [1]. Каждая таблица или отношение представляет собой один объект, строки – экземпляры этого объекта, а столбцы – значение, приписанное к этому экземпляру. При данной организации существуют ссылки на другие строки и таблицы ключевых атрибутов – внешние ключи, на которые действуют ограничения; уникальный искусственно введенный ключ для идентификации строк объектов, являющийся предметом реализации, а не описанием объекта и не имеющий семантического значения, – первичный или суррогатный ключ [2]. Первичный ключ БД позволяет уникально идентифицировать строку внутри таблицы. При перемещении ключей в другую таблицу они становятся внешними по отношению к той таблице, из которой мигрировали. Данная конструкция позволяет определять логическую связь между различными таблицами на основе взаимодействия между ними – отношения один-к-одному или один-ко-многим. РБД, как известно, подчиняются требованиям ACID (atomicity, consistency, isolation, durability – атомарность, согласованность, изолированность, устойчивость). В качестве ключевой характеристики ACID определяет надежность работы в транзакционной системе [3].

Количество байтов в мире стремительно растет. Начиная с 2005 года аналитики исследовательской компании IDC публикуют годовую оценку всех байтов, добавленных к цифровой вселенной. По их подсчетам, с 2012 года произошел рост с 130 млрд Гб до 2,8 трлн Гб. По прогнозу IDC, к 2020 году это число достигнет 50 трлн Гб (рис. 1). Каждый год потребите-

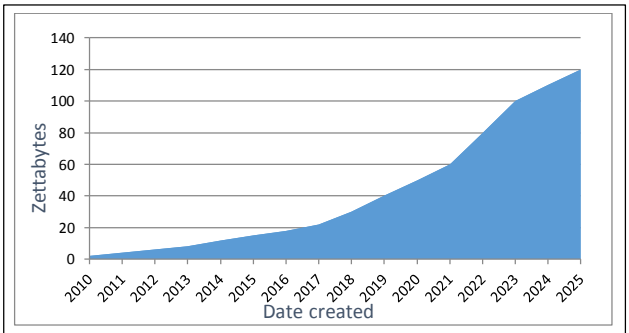

Рис. 1. Количество использованных данных и прогноз на будущее

Fig. 1. The amount of data used and the forecast for the future

лями используется около 70–75 % от общей суммы данных, из которых 80 % приходится на цифровое телевидение, 10 % на фото и видео. В связи с быстрым ростом количества данных и их усложнением возникла необходимость в новых подходах к хранению и обработке, отличной от реляционной, таким решением стала NoSQL-технология.

С момента своего появления сокращение NoSQL означало еще одну реляционную систему управления БД, но не использовавшую стандарты SQL, в дальнейшем термин стали использовать по отношению к *нереляционным БД* (НРБД).

В целом БД NoSQL стали первой альтернативой РБД. Масштабируемость, доступность, отказоустойчивость являлись ее ключевыми решающими факторами. Эти БД имеют преимущества по сравнению с РБД и удовлетворяют потребностям современных бизнес-приложений. Обычно эту технологию характеризуют гибкость схемы данных, свободная модель данных, горизонтальная масштабируемость, распределенные архитектуры, а также использование языков и интерфейсов, которые являются «не только» SQL.

В основе NoSQL-базы заложена идея, не подразумевающая внутренних связей или строго определенных моделей, она позволяет задавать собственную модель данных в зависимости от поставленных задач. NoSQL ставит перед собой цель обеспечить гибкость и масштабируемость системы. Эти качества могут быть описаны аббревиатурой BASE. В соответствии с CAP-теоремой в распределенных вычислительных системах одномоментно может быть гарантирована работа двух свойств из трех: быстрый доступ, согласованность данных, устойчивость к разделению [4].

В процессе работы было произведено сравнение NoSQL и SQL на основе практических характеристик:

– РБД используют определенные типы и структуры данных, описываемые ER-моделью, НРБД допускают более свободное использование типов и не используют строгие модели данных;

– запросы к РБД должны быть написаны в соответствии с требованиями SQL-стандартов, НРБД используют особые подходы к построению запросов [5];

– масштабируемость в вертикальном смысле присуща обоим типам БД, но горизонтальное масштабирование может быть проще реализовано в НРБД;

– благодаря системе транзакций РБД считаются надежнее в плане консистентности данных;

– в НРБД партиционирование, настройка репликации в большинстве случаев более легкая задача, чем в реляционной СУБД.

Приведем примеры ситуаций, в которых целесообразно применять NoSQL-подход.

- BigData. NoSQL становится ключевой частью новой модели данных, поддерживающей большой объем данных, множество пользователей и т.д.
- Большая производительность записи. Для таких проектов, как Facebook или Twitter, одной из главных проблем является запись большого количества данных за минимальное время. Для обеспечения высокой скорости записи используются так называемые in-memory-системы (в основном они располагаются в оперативной памяти) [6].
- Быстрый доступ по ключу. Когда время отклика играет определяющую роль, тяжело представить что-то более быстрое, чем кэширование по ключу и чтение напрямую из памяти.
- Доступность записи. Случай, когда всегда необходимо иметь возможность чтения записи независимо от того, что записывается.
- Возможность параллельных вычислений. Например, использование технологии MapReduce [7].

Выбирая то или иное NoSQL-решение для реализации конкретной системы, необходимо точно представлять решаемые задачи, а также учитывать базовые идеи работы NoSQL-хранилищ:

– отсутствие определенных требований к данным, которые могут изменяться в процессе развития проекта [8];

– необходимость старта проекта, когда известно, что его цель позднее может быть изменена;

– масштабируемость системы и скорость обработки данных, являющиеся ключевыми требованиями к системе.

В связи с тем, что теорема ACID, лежащая в основе РБД, снимает с разработчиков необходимость отслеживать блокировки, непротиворечивость данных, устаревшие данные и коллизии, реляционный подход может быть рекомендован к построению систем. Они имеют заранее определенные логические требования к данным, обязательным требованием является целостность используемых данных (если они, например, стоят дорого).

Был проанализирован официальный всемирный рейтинг DB-Engines (см. таблицу).

Рейтинг DB-Engines на февраль 2018 г.

DB-Engines Raking as of February 2018

Rank (Feb 2018)	DBMS	Database Model
1	Oracle	Relation DBMS
2	MySQL	Relation DBMS
3	Microsoft SQL Server	Relation DBMS
4	PostgreSQL	Relation DBMS
5	MongoDB	Document store
6	DB2	Relation DBMS
7	Microsoft Access	Relation DBMS
8	Redis	Key-value store
9	Elasticsearch	Search engine
10	Casandra	Wide column store

Список СУБД ранжирован по их текущей популярности. DB-Engines собирает статистику ежемесячно, что позволяет получить наиболее точную информацию на текущий момент. В целом, анализируя динамику изменения позиций в данном рейтинге, можно заметить, что изменения происходят крайне медленно.

Также был проанализирован подобный рейтинг на российском рынке, где лидирующие позиции занимали те же СУБД. На основе данного рейтинга были выбраны две наиболее популярные разнотипные СУБД – Oracle и MongoDB. Эти системы хорошо документированы, имеют полноценный язык запросов и большое сообщество разработчиков.

СУБД Oracle в общем случае представляет собой сервер, надежно управляющий большим количеством данных в многопользовательской среде, поэтому многие пользователи одновременно получают доступ к одним и тем же данным. Сервер БД также предотвращает несанкционированный доступ и предоставляет инструменты для восстановления после сбоя [9]. Данная СУБД разработана для корпоративных распределенных вычислений, наиболее гибкого и экономически эффективного способа управления информацией и приложениями. БД содержит логические и физические структуры. Так как эти структуры разделены, физическое хранение данных может осуществляться без ущерба для доступа к логическим структурам хранения. При проектировании БД разработчики зачастую придерживаются третьей, нормальной, формы, что практически гарантирует устранение избыточности данных, однако возможны случаи, когда данные не нормализованы.

Рассмотрим нереляционную документ-ориентированную СУБД MongoDB. В основе таких СУБД лежат документные хранилища, имеющие древовидную структуру. Все древовидные структуры начинаются с корневого узла, содержащего внутренние поддеревья и листовые узлы. В «листьях» хранятся данные. При вставке данных в коллекцию для них создается индекс, который облегчает поиск запрашиваемых данных даже в случае, когда древовидная структура становится сложной. Результатом выполнения запроса будут документы или части документов. БД может иметь несколько коллекций. Коллекция в MongoDB может быть в некотором смысле сопоставлена с таблицей РБД, однако стоит заметить, что также допустимо агрегирование всей схемы РБД внутри одной коллекции [10]. Все документы-коллекции могут быть рассмотрены как строки РБД и включают в себя определенное количество полей, подобных колонкам. Основное различие данных определений в том, что РБД определяют колонки на уровне таблицы, а документ-ориентированные БД определяют поля на уровне документа, что позволяет документам внутри коллекции иметь свой собственный набор полей, отвечающий поставленным целям. В данном случае документ коллекции MongoDB несет в себе несколько больше информации, чем строка.

В качестве примера рассмотрим представленную в таблицах БД Oracle часть БД «Сотрудник» (рис. 2), которая включает следующие атрибуты: название отдела, номер телефона, местонахождение отдела (здание и аудитория), люди, работающие в отделе.

В MongoDB атрибуты всех таблиц части БД «Сотрудник» могут быть описаны в одной коллекции или каждая сущность может храниться в отдельной коллекции. Представим первый способ описания части БД «Сотрудник» в НРБД MongoDB:

```
Documentotdel:
{
   "id":"id_документа",
   "название":"название_отдела",
   "телефон":"телефон_отдела",
   "здание":"адрес_отдела",
   "аудитория":"номер_аудитории",
   "люди":["человек_1","человек_2",…]
}
```

Например, в результате выполнения SQL-запроса будет получена полная информация об отделах:

```
Select * from department where depart-
ment_name ='name1';
```

Опишем запрос, возвращающий такую же информацию из MongoDB:

```
Db.department.find("название",  "назва-
ние_отдела").
```

Зачастую необходимо перемещать данные, опираясь на условия использования и преследуемые цели, внутри информационной системы, построенной на нескольких СУБД различной организации.

Фундаментальное различие между MongoDB и RDBMS кроется в моделях данных. В RDBMS данные хранятся в виде таблиц, а в MongoDB – внутри коллекций в JSON-формате. Коллекции создаются без предварительного определения их структуры, а количество полей и их формат в документе одной коллекции могут отличаться. JSON является удобным для чтения и написания человеком форматом данных. Первоначально разработанный для обмена данными между браузером и сервером, он стал широко использоваться многими типами приложений [3]. Документ в формате JSON состоит из набора полей, которые сами по себе

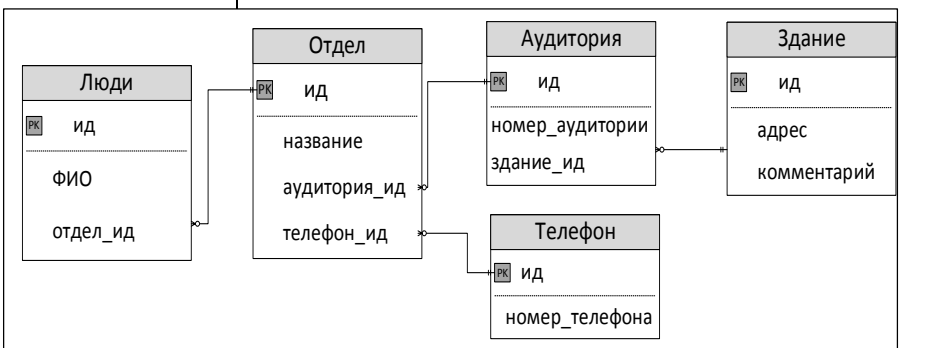

Рис. 2. БД «Сотрудник»

Fig. 2. Database "Sotrudnik"

являются парами ключ–значение, что позволяет документу легко перемещаться между приложениями и БД клиентов. JSON также является естественным форматом данных для использования в приложениях. JSON поддерживает более богатую и более гибкую структуру данных, чем таблицы, составленные из столбцов и строк. Поля JSON-документа могут быть массивами, вложенными объектами. Это позволяет представить множество сложных отношений, приближенных к настоящему представлению объектов в приложениях. Использование JSON-документов в БД позволяет исключить реляционный тип хранения данных, представляя их в формате, используемом конечным приложением.

В ходе исследования рассматривался перенос данных из таблиц Oracle в коллекции MongoDB как наиболее рейтинговых СУБД по версии DB-Engines (см. http://www.swsys.ru/uploaded/image/2019-1/2019-1-dop/9.jpg). В качестве примера показана работа с частью БД «Сотрудник». Как уже сказано, MongoDB (как и любая другая разновидность НРБД) не предназначена для построения БД на основе таблиц, и, как следствие, невозможна работа со структурированным языком запросов. Для миграции данных между рассматриваемыми БД необходимо выполнить следующие действия:

– спроектировать объекты или документы, которые требуется перенести в Mongo; объекты должны быть близки к существующим объектам, используемым в приложениях;

– определить параметры объекта, например, где присутствуют таблицы, в которых хранится информация о данном объекте; в MongoDB все данные объекта будут объединены в одну коллекцию;

– написать/запустить скрипт преобразования данных из Oracle в «объект», сериализовать этот объект в JSON и затем записать в MongoDB.

В результате проделанной работы были исследованы реляционные и нереляционные модели данных на примере СУБД Oracle и MongoDB, выявлены основные достоинства и недостатки. В ходе исследования разработана схема преобразования данных из таблиц РБД (Oracle) в коллекции НРБД (MongoDB). Исследование показало, что нереляционные системы, в основе которых заложен BASE-принцип, не должны использоваться, например, в приложениях, связанных с банковской деятельностью, которая невозможна без механизма транзакций.

Литература

1. Fowler M., Sadalage P.J. NoSQL: new methodology of no relation DB developing. Moscow, Williams Publ. House, 2013, pp. 5–15.
2. Deinum M. et al. NoSQL and BigData. Spring Recipes. Apress Publ., 2014, pp. 549–590.
3. He J.S.K.T.G., Naughton C.Z.D.D.W.J. Relational databases for querying XML documents: Limitations and opportunities. Very Large Data Bases: Proc., 1999, pp. 302–314.
4. Chodorow K. MongoDB: the definitive guide. Proc. O'Reilly Media, Inc., 2013, pp. 11–22.
5. Vohra D. Pro MongoDB Development. Apress Publ., 2015, pp. 297–330.
6. Russell M., Russell M. Mining the social web: Analyzing data from Facebook, Twitter, LinkedIn, and other social media sites. Head First, O'Reilly Media, 2011, pp. 52–94.
7. Jia T. et al. Model Transformation and Data Migration from Relational Database to MongoDB. BigData Congress, Proc. IEEE Intern. Congress on, 2016, pp. 60–67.
8. Edward S.G., Sabharwal N. Using MongoDB Shell. Practical MongoDB. Apress Publ., 2015, pp. 53–93.
9. Annual size of the global datasphere. URL: https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf (дата обращения: 20.03.2018).
10. Ma K., Yang B. Column Access-aware In-stream Data Cache with Stream Processing Framework. J. Signal Processing Syst., 2016, pp. 1–15.

Software & Systems
DOI: 10.15827/0236-235X.125.063-067

Received 21.03.18
2019, vol. 32, no. 1, pp. 063–067

Development of the concept of data migration between relational and non-relational database systems

Yu.A. Koroleva [1], *Ph.D. (Engineering), Assistant, jakoroleva@corp.ifmo.ru*
V.O. Maslova [1], *Student, victoria_95m@mail.ru*
V.K. Kozlov [1], *Student, 172652@niuitmo.ru*

[1] *The National Research University of Information Technologies, Mechanics and Optics, St. Petersburg, 197101, Russian Federation*

Abstract. The article investigates relational and non-relational approaches to constructing, storing, and extracting data.

Nowadays all information and analytical systems use databases. These systems require the ability to process, read, write specific data sets that need to be organized, structured and stored.

Finding a suitable database and database management system is one of the most common problem for many companies, as this choice will determine performance, reliability, security, design features and other work features. Usually several data models can be used in one information system of a company. For example, companies use a relational database for the tasks that require using full data consistency and transaction control, whereas analytical, aggregated or meta-data can be kept in a NoSQL database. This separation is often necessary for the most effective functioning of the final product. Combining these systems is the main problem.

The research discovered the most popular database management systems for both approaches of developing databases. Their advantages and disadvantages are analyzed. As the first phase of data preparation for transparent migration between two systems, the authors propose the transformation scheme from relational data to non-relational data. This scheme is based on the databases internal organization and their peculiar properties.

Keywords: heterogeneous information systems, data transfer, data synchronization, nosql, database management systems, data migration.

References

1. Fowler M., Sadalage P.J. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence.* Addison-Wesley, 2013 (Russ. ed.: Moscow, Williams Publ., 2013, 192 p.).

2. Deinum M. et al. *NoSQL and BigData. Spring Recipes*. Apress, 2014, pp. 549–590.

3. He J.S.K.T.G., Naughton C.Z.D.D.W.J. Relational databases for querying XML documents: Limitations and opportunities. *Very Large Data Bases: Proc*. 1999, pp. 302–314.

4. Chodorow K. *MongoDB: the Definitive Guide*. O'Reilly Media Publ., 2013, pp. 11–22.

5. Vohra D. *Pro MongoDB Development*. Apress, 2015, pp. 297–330.

6. Russell M., Russell M. *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. Head First Series. 2011, pp. 52–94.

7. Jia T. et al. Model Transformation and Data Migration from Relational Database to MongoDB. *Big Data (BigData Congress), 2016 IEEE Intern. Congress on. IEEE*. 2016, pp. 60–67.

8. Edward S.G., Sabharwal N. Using MongoDB Shell. *Practical MongoDB*. Apress, 2015, pp. 53–93.

9. *Annual Size of the Global Datasphere*. Available at: https://www.seagate.com/files/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf (accessed March 20, 2018). 2017, pp. 4–24.

10. Ma K., Yang B. Column Access-aware In-stream Data Cache with Stream Processing Framework. *J. of Signal Processing Systems*. 2016, pp. 1–15.

Примеры библиографического описания статьи

1. Королева Ю.А., Маслова В.О., Козлов В.К. Разработка концепции миграции данных между реляционными и нереляционными системами БД // Программные продукты и системы. 2019. Т. 32. № 1. С. 63–67. DOI: 10.15827/0236-235X.125.063-067.

2. Koroleva Yu.A., Maslova V.O., Kozlov V.K. Development of the concept of data migration between relational and non-relational database systems. *Software & Systems*. 2019, vol. 32, no. 1, pp. 63–67 (in Russ.). DOI: 10.15827/0236-235X.125.063-067.