ПРЕИМУЩЕСТВА И НЕДОСТАТКИ НЕРЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Шахрай А.В.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Хлудеев И.И. – канд. биол. наук, доцент

Произведен сравнительный анализ нереляционных баз данных в сравнении с реляционными.

В течение десятилетий центральное место в разработке приложений занимала реляционная модель данных, которая использовалась в реляционных базах данных, таких как Oracle, DB2, SQL Server, MySQL и PostgreSQL. Но в середине – конце 2000-х годов заметное распространение стали получать и другие модели данных. Для обозначения появившихся классов БД и моделей данных был введен термин «NoSQL». Часто «NoSQL» используется в качестве синонима к термину «нереляционный».

NoSQL-базы оптимизированы для приложений, которые должны быстро, с низкой временной задержкой (low latency) обрабатывать большой объем данных с разной структурой [2]. Таким образом, нереляционные хранилища непосредственно ориентированы на Big Data. Однако, идея баз данных такого типа зародилась гораздо раньше термина «большие данные», еще в 80-е годы прошлого века, во времена первых компьютеров (мэйнфреймов) и использовалась для иерархических служб каталогов. Современное понимание NoSQL-СУБД возникло в начале 2000-х годов, в рамках создания параллельных распределённых систем для высокомасштабируемых интернет-приложений, таких как онлайн-поисковики [1].

Вообще термин NoSQL обозначает «не только SQL» (Not Only SQL), характеризуя ответвление от традиционного подхода к проектированию баз данных. Изначально так называлась опенсорсная база данных, созданная Карло Строззи, которая хранила все данные как ASCII-файлы, а вместо SQL-запросов доступа к данным использовала шелловские скрипты [3]. В начале 2000-х годов Google построил свою поисковую систему и приложения (GMail, Maps, Earth и прочие сервисы), решив проблемы масштабируемости и параллельной обработки больших объёмов данных. Так была создана распределённые файловая и координирующая системы, а также колоночное хранилище (column family store), основанное на вычислительной модели MapReduce. После того, как корпорация Google опубликовала описание этих технологий, они стали очень популярны у разработчиков открытого программного обеспечения. В результате этого был создан Apache Hadoop и запущены основные связанные с ним проекты. Например, в 2007 году другой ИТ-гигант, Amazon.com, опубликовав статьи о своей высокодоступной базе данных Amazon DynamoDB. Далее в эту гонку NoSQL- технологий для управления большими данными включилось множество корпораций: IBM, Facebook, Netflix, eBay, Hulu, Yahoo! и другие ИТ-компаний со своими проприетарными и открытыми решениями (рисунок 1) [1].

Рисунок 1 – Области применения NoSQL

По сравнению с классическими SQL-базами, нереляционные СУБД обладают следующими преимуществами:

– линейная масштабируемость – добавление новых узлов в кластер увеличивает общую производительность системы [1];

– гибкость, позволяющая оперировать полуструктурированные данные, реализуя, в. т.ч. полнотекстовый поиск по базе [2];

– возможность работать с разными представлениями информации, в т.ч. без задания схемы данных [1];

– высокая доступность за счет репликации данных и других механизмов отказоустойчивости, в частности, шаринга – автоматического разделения данных по разным узлам сети, когда каждый сервер кластера отвечает только за определенный набор информации, обрабатывая запросы на его чтение и запись. Это увеличивает скорость обработки данных и пропускную способность приложения [4].

– производительность за счет оптимизации для конкретных видов моделей данных (документной, графовой, колоночной или «ключ-значение») и шаблонов доступа [2];

– широкие функциональные возможности – собственные SQL-подобные языки запросов, RESTful-интерфейсы, API и сложные типы данных, например, map, list и struct, позволяющие обрабатывать сразу множество значений [2].

Обратной стороной вышеуказанных достоинств являются следующие недостатки:

– ограниченная емкость встроенного языка запросов [4]. Например, HBase предоставляет всего 4 функции работы с данными (Put, Get, Scan, Delete), в Cassandra отсутствуют операции Insert и Join, несмотря на наличие SQL-подобного языка запросов. Для решения этой проблемы используются сторонние средства трансляции классических SQL-выражений в исполнительный код для конкретной нереляционной базы. Например, Apache Phoenix для HBase или универсальный Drill.

– сложности в поддержке всех ACID-требований к транзакциям (атомарность, консистентность, изоляция, долговечность) из-за того, что NoSQL-СУБД вместо CAP-модели (согласованность, доступность, устойчивость к разделению) скорее соответствуют модели BASE (базовая доступность, гибкое состояние и итоговая согласованность) [1]. Впрочем, некоторые нереляционные СУБД пытаются обойти это ограничение с помощью настраиваемых уровней согласованности, о чем мы рассказывали на примере Cassandra. Аналогичным образом Riak позволяет настраивать требуемые характеристики доступности-согласованности даже для отдельных запросов за счет задания количества узлов, необходимых для подтверждения успешного завершения транзакции [1]. Подробнее о CAP-и BASE-моделях мы расскажем в отдельной статье.

– сильная привязка приложения к конкретной СУБД из-за специфики внутреннего языка запросов и гибкой модели данных, ориентированной на конкретный случай [4];

– недостаток специалистов по NoSQL-базам по сравнению с реляционными аналогами [4].

Подводя итог описанию основных аспектов нереляционных СУБД, стоит отметить некоторую некорректность запроса «NoSQL vs SQL» в связи с разными архитектурными подходами и прикладными задачами, на которые ориентированы эти ИТ-средства. Традиционные SQL-базы отлично справляются с обработкой строго типизированной информации не слишком большого объема. Например, локальная ERP-система или облачная CRM. Однако, в случае обработки большого объема полуструктурированных и неструктурированных данных, т.е. Big Data, в распределенной системе следует выбирать из множества NoSQL-хранилищ, учитывая специфику самой задачи. В частности, для самостоятельных решений интернета вещей (Internet of Things), в т.ч. промышленного, отлично подходит Cassandra, о чем мы рассказывали здесь. А в случае многоуровневой ИТ-инфраструктуры на базе Apache Hadoop стоит обратить внимание на HBase, которая позволяет оперативно, практически в режиме реального времени, работать с данными, хранящимися в HDFS.

Список использованных источников:

1. *Что такое NoSQL? [Электронный ресурс], - Режим доступа: https://aws.amazon.com/ru/nosql/*
2. *NoSQL – Материал из Национальной библиотеки им. Н. Э. Баумана [Электронный ресурс], - Режим доступа: https://ru.bmstu.wiki/NoSQL*
3. *Exploring the different types of nosql databases part II [Электронный ресурс], - Режим доступа: https://tproger.ru/translations/types-of-nosql-db/*
4. *Сильные и слабые стороны NoSQL – [Электронный ресурс], - Режим доступа: https://habr.com/ru/sandbox/113232/*