THE ANALYSYS STORING SOCIAL DATA FROM THE INTERNET METHODS

Sukhanov A.[1], Maratkanov A.[2] (Russian Federation)

АНАЛИЗ СПОСОБОВ ХРАНЕНИЯ СОЦИАЛЬНЫХ ДАННЫХ ИЗ СЕТИ ИНТЕРНЕТ

Суханов А. А.[1], Маратканов А. С.[2] (Российская Федерация)

[1]Суханов Александр Александрович / Sukhanov Aleksandr – магистрант;
[2]Маратканов Александр Сергеевич / Maratkanov Aleksandr – магистрант,
кафедра компьютерных систем и сетей, факультет информатики и систем управления,
Московский государственный технический университет им. Н. Э. Баумана, г. Москва

Abstract: *the problem of collecting social data becomes more urgent. Storage of that data becomes a problem. The specificity of that data structures and architecture problems in common database management systems are the weaknesses of many databases. Basic SQL and NoSQL solutions that can be used for these purposes are discussed in the article. Also comparison between most popular solutions like PostgreSQL and MySQL was made. Also detail analysis of most popular NoSQL databases was made. Advantages and disadvantages of storing types in NoSQL databases were also considered.*

Аннотация: *на данный момент задача сбора социальных данных приобретает всю большую актуальность. Однако это ставит такую проблему, как хранение этих данных. Из-за специфичности подобных данных, а также из-за того, что многие СУБД не были проектированы для хранения больших массивов часто обновляющихся и слабо структурированных данных, особое внимание требуется уделить выбору СУБД. В статье рассмотрены основные SQL и NoSQL решения, которые могут быть использованы для подобных целей. Кроме того, проведено сравнение этих решений.*

Keywords: *social data, SQL, NoSQL, databases.*
Ключевые слова: *социальные данные, SQL, NoSQL, базы данных.*

1. Анализ SQL- решений для хранения данных

SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов [1].

Наиболее распространенными бесплатными решениями являются MySQL и PostgreSQL, поэтому далее мы постараемся их сравнить по следующим параметрам:

- Репликация.
- Документация.
- Стандарты.

Более подробное сравнение MySQL и PostgreSQL приведено в таблице 1.

Таблица 1. Сравнение MySQL с PostgreSQL [2]

Тип сравнения	**PostgreSQL**	**MySQL**
Репликация	Основное преимущество – физические репликации. PostgreSQL сохраняет запросы, которые потом попадают в журнал и после этого журнал используется для репликации. Этот же журнал используется и для восстановления после сбоев, то есть данный метод уже прошел проверку временем. Кроме того существует триггерная репликация.	Основыне проблемы: корректность работы, производительность репликации и стабильность репликации. К сожалению, в MySQL репликация недостаточно продумана и не может быть полностью использована из-за поддержки так называемых storage engine, то есть подключаемых движков. Из-за этого возникают проблемы с транзакциями между таблицами с разными storage engine.
Сложность администрирования	Сложности администрирования сглаживаются засчет полноты официальной документации и большого размера PostgreSQL-сообщества в сети.	Более простая в администрировании СУБД, однако это во многом связано с достаточно ограниченным функционалом.
Стандарты	SQL-92 SQL-98 SQL-2003 Идет работа над поддержкой стандарта SQL-2011	SQL-92
Документация	Документация PostgreSQL является наиболее объемной среди всех СУБД. Отчасти это объясняется	Документация MySQL оставляет желать лучшего. Несмотря на наличие официальной документации, для

	ее повсеместным распространением. Кроме наличия подробной официальной документации также существует достаточное количество информации по правилам проектирования и оптимизированию PostgreSQL баз данных.	понимания работы СУБД часто приходится не только обращаться к дополнительным источникам, но и разбираться в арихтекутуре СУБД. Из-за этого проектирования хранилищ для большого числа данных упирается в то, что можно допустить ошибки просто из-за того, что какие-либо моменты не были освещены в документации.

2. Анализ NoSQL решений для хранения данных

Традиционные СУБД ориентируются на требования ACID к транзакционной системе: атомарность (англ. *atomicity*), согласованность (англ. *consistency*), изолированность (англ. *isolation*), надёжность (англ. *durability*), тогда как в NoSQL вместо ACID может рассматриваться набор свойств BASE [3]:

• базовая доступность (англ. *basic availability*) — каждый запрос гарантированно завершается (успешно или безуспешно).

• гибкое состояние (англ. *soft state*) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.

• согласованность в конечном счёте (англ. *eventual consistency*) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Далее представлен анализ по таким критериям как масштабируемость, модель данных и запросов и системе хранения данных.

2.1. Масштабируемость

Под масштабируемостью часто подразумевается репликация, поэтому в данном контексте имеется ввиду автоматическое распределение данных между несколькими серверами.

В распределенной базе данных следует обращать внимание на следующие параметры: поддержка нескольких датацентров и возможность добавления новых машин в работающий кластер прозрачно для ваших приложений.

Таблица 2. Сравнение NOSQL баз данных

	Прозрачное добавление в кластер	Поддержка нескольких датацентров
Cassandra	Есть	Есть
HBase	Есть	Нет
RIAK	Есть	Есть
Scalaris	Есть	Нет
Voldemort	Нет	Требуется ручная оптимизация

2.2. Модель данных и запросов

Существует огромное многообразие моделей данных и API запросов в NoSQL базах данных.

Таблица 3. Модели данных и API запросов NOSQL баз данных

	Модель данных	API запросов
Cassandra	Семейства столбцов	Thrift
CouchDB	Документы	Map/Reduse
HBase	Семейства столбцов	Thrif,REST
MongoDB	Документы	Cursor
Neo4J	Графы	Graph
Redis	Коллекции	Collection
RIAK	Документы	Nested cashes, REST
Scalaris	Ключ/значение	Get/Put
Tokyo Cabinet	Ключ/значение	Get/Put
Voldemort	Ключ/значение	Get/Put

Документо-ориентированные базы данных - это по существу следующий уровень систем ключ/значение, позволяющий связывать вложенные данные с каждым ключом. Поддержка таких запросов более эффективна, чем просто возвращение всего BLOB каждый раз.

Neo4J обладает поистине уникальной моделью данных, храня объекты и связи в качестве узлов и ребер

графа. Для запросов, которые соответствуют этой модели (например, иерархических данных) они могут быть в тысячу раз быстрее, чем альтернативные варианты.

Scalaris уникальна в использовании распределенных транзакций между несколькими ключами. Обсуждение компромиссов между последовательностью и наличием свободных мест также необходимо учитывать при оценке в распределенных системах.

2.3. Система хранения данных

Под системой хранения данных подразумевается то, как данные хранятся внутри системы.

Таблица 4. Модели данных NOSQL баз данных

Название	Модель данных
Cassandra	Memtable/SStable
CouchDB	Append-only-B-Tree
HBase	Memtable/SStable on HDFS
MongoDB	B-Tree
Neo4J	On-disk linked lists
Redis	In-memory with backgorund snapshots
RIAK	Hash
Scalaris	In-memory-only
Tokyo Cabinet	Hash/B-Tree
Voldemort	Pluggable

Система хранения данных может сказать нам многое о том, какие нагрузки база может нормально выдерживать.

Таким образом, наиболее оптимальным решением для рассматриваемой системы является связка SQL и NOSQL решения. В качестве СУБД было принято решение выбрать Cassandra и PostgreSQL.

Литература

1. Введение в SQL. SQL: Обзор. [Электронный ресурс]. Режим доступа: http://www.mysql.ru/docs/gruber/mg02.html/ (дата обращения: 08.12.2016).
2. Хабрахабр. PostgreSQL vs MySQL. [Электронный ресурс]. Режим доступа: https://habrahabr.ru/company/mailru/blog/248845/ (дата обращения: 07.12.2016).
3. Википедия. NoSQL. [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/NoSQL/ (дата обращения: 08.12.2016).
4. Хабрахабр. Обзор NoSQL систем. [Электронный ресурс]. Режим доступа: https://habrahabr.ru/post/77909/ (дата обращения: 06.12.2016).