УДК 004.852

Журавльов Павло Володимирович
студент
Національний технічний університет України «Київський політехнічний інститут»
Журавлёв Павел Владимирович
студент
Национальный технический университет Украины «Киевский политехнический институт»
Zhuravlyov P.
student
National Technical University of Ukraine «Kyiv Polytechnic Institute»

ПОРІВНЯННЯ МОДЕЛЕЙ ДАНИХ NOSQL СХОВИЩ

СРАВНЕНИЕ МОДЕЛЕЙ ДАННЫХ NOSQL ХРАНИЛИЩ

DATA MODELS OF NOSQL STORAGES COMPARISON

Анотація. *Досліджено характеристики NoSQL СКБД та їх відмінності від реляційних СУБД. Також були описані переваги та недоліки кожного типу моделей даних і приклади їх використання.*

Ключові слова: *нереляційні бази даних, комп'ютерні науки, комп'ютерні технології.*

Аннотация. *Исследованы характеристики NoSQL СУБД и их отличия от реляционных СУБД. Также были описаны преимущества и недостатки каждого типа моделей данных и примеры их использования.*

Ключевые слова: *нереляционные базы данных, компьютерные технологии, компьютерные науки.*

Summary. *The characteristics of NoSQL DBMS and their differences from the relational DBMS, advantages and disadvantages of each type of data models, and examples of their usage described.*

Key Words: *NoSQL, non-relational databases, computer science.*

Введение

Настоящее время характеризуется бурным развитием web-сервисов, социальных сетей и других интернет-услуг. Хранение и обработка больших объемов информации становится актуальной задачей. В последние несколько десятков лет эту задача решалась использованием реляционных СУБД, поддерживающие язык структурированных запросов SQL.

Модель облачных вычислений набирает большую популярность в последние годы и возможность масштабирования приложения становиться ключевой. Реляционные СУБД с трудом масштабируются и не обеспечивают гибкость модели данных. Обработка большего числа пользователей означает добавление более мощного сервера что увеличивает сложность и дороговизну непропорционально по отношению к облачным архитектурам.

Это привело к разработке новой модели данных NoSQL. Большинство таких хранилищ отказались от многих функций реляционных СУБД, в пользу масштабируемости и производительности.

В большинстве своем нереляционные модели совместимы с принципами BASE. Характеристики различных типов NoSQL хранилищ соответствуют теореме CAP.

Как только предприятия начинают сталкиваться с проблемами производительности с их реляционными СУБД, возникает потребность в более быстром и гибком слое данных. В таком случае хорошим вариантом будет оценить существующие NoSQL решения и внедрить выбранное в приложение предприятия.

Основные типами нереляционных моделей данных являются хранилища пар ключ-значение и семейств колонок, графовые и документно-ориентированные СУБД.

Основные характеристики баз данных

Реляционная модель структурирует данные в виде кортежей (строк). Кортеж — это ограниченная структура, которая содержит множество значений так, что невозможно вставить кортеж либо массив кортежей в другой для того, чтобы получить дерево. Это

ограничивает реляционную модель тем, что операции являют собой операции над кортежами. Агрегатный подход являет собой оперирование более сложными структурами.

Агрегат — это набор данных, с которым производятся операции как с атомарной частицей. Агрегаты формируют границы для ACID операций с БД. Один из компонентов агрегата является его корнем. Ссылки извне агрегата должны указывать на его корень для того, чтобы обеспечивать целостность данных, заключенных в нем.

Агрегатно-ориентированные СУБД предназначены для совершения операций в пределах агрегата и усложняют транзакции, затрагивающие связи между агрегатами.

Одной из популярных формулировок требований к СУБД является ACID. Ключевой особенностью модели ACID является то, что она обеспечивает безопасную среду для операций над данными. Акроним ACID расшифровывается как:

- Атомарность (англ. *Atomicity*) — все операции в транзакции выполняются успешно, либо данные возвращаются в исходное состояние
- Целостность (англ. *consistency*) — по завершению транзакции, данные находятся в корректном состоянии
- Изолированность (англ. *isolation*) — параллельно выполняющиеся транзакции не должны оказывать влияние на результат друг друга
- Длительность (англ. *durability*) — изменения, совершенные транзакцией не могут быть отменены вследствие какого либо сбоя

Выполнение принципов ACID означает, что как только транзакция завершена, измененные данные являются целостными и стабильно хранятся на диске, даже если они хранятся в разных местах памяти. Большинство графовых СУБД используют модель ACID.

Для многих предметных областей и случаев, ACID принципы выполнения транзакций предполагают более строгие ограничения, чем того требует предметная область.

Акроним BASE рашифровывается как

1) базовая доступность (англ. *basic availability*) — при сбоях в распределенной системе доступность сохраняется в большинстве случаев;

2) неустойчивое состояние (англ. *soft-state*) — реплики не являются согласованными в каждый момент времени;

3) согласованность в конечном счёте (англ. *eventual consistency*) — модель согласованности, гарантирующая возврат последнего обновленного значения объекта.

BASE обеспечивает доступность, поскольку это важно для масштабируемости, но не гарантирует целостность копий данных во время записи. Они будут целостными в будущем, или во время чтения, или всегда, но для определенного количества обработанных снимков. BASE принципы совместимы с теоремой CAP и используются агрегатно-ориентированными хранилищами, такими как хранилища ключ-значение, семейств колонок и документов.

Теорема CAP — утверждение о том, что в любой реализации системы распределённого хранения информации возможно обеспечить не более двух из трёх следующих свойств:

1) согласованность (англ. *consistency*) — в отдельно взятый момент времени данные во всех узлах не противоречат друг другу;

2) доступность (англ. *availability*) — любой запрос должен получить ответ об успехе или ошибке;

3) устойчивость к разделению (англ. *partition tolerance*) — система продолжает корректно функционировать при сбое работы сети.

Важной характеристикой, от которой зависит способность системы справляться с нагрузками является масштабируемость хранилища. Существует два способа масштабирования:

1. Вертикальное масштабирование — увеличение производительности устройства, обрабатывающего запросы.

2. Горизонтальное масштабирование — распределение системы по большему количеству устройств, обрабатывающих запросы.

Способы горизонтального масштабирования хранилища:

1. Репликация — содержание нескольких копий хранилища на разных серверах, что увеличивает доступность и ухудшает согласованность из за возрастающей сложности управления такой структурой и необходимости затратной по времени синхронизации. Широко используется master-slave модель, в которой один master узел кластера обрабатывает запросы на запись и обновляет данные на slave узлах, задача которых — обработывать запросы на чтение.

2. Шардинг — разбитие множества агрегатов в агрегатно-ориентированной БД либо строк таблицы в реляционной БД на части и распределение частей по разным серверам. При этом возрастает сложность модифицирования схемы данных, ухудшается целостность и длительность.

Сравнение моделей данных

Хранилище «ключ-значение»

Хранилища ключ-значение одни из базовых моделей данных семейства NoSQL. Они являются довольно простыми и высокопроизводительными моделями. Эти хранилища похожи на хэш-таблицы где каждому

ключу поставлено в соответствие значение, которое может быть представлено различными форматами, например бинарными данными, строкой, форматами JSON, XML.

Современные хранилища ключ-значение предпочитают высокую масштабируемость над целостностью. Поэтому аналоги таких операций, присутствующих в реляционных СУБД, как объединение таблиц и агрегатные функции, отсутствуют.

Целостность обеспечивается только при операциях над агрегатами, которые представлены парами ключ-значение. В распределенных реализация хранилищ этого типа используется модель согласованности в конечном счете.

Присутствует возможность масштабирования с использованием шардинга. При этом значение ключа определяет на каком узле хранится информация.

Хранилища ключ-значение могут быть использованы для хранения несвязанных друг с другом данных, таких как сессии, корзины с покупками и конфигурации, либо при большом количестве запросов на чтение.

Не рекомендуется использовать эти хранилища при необходимости совершения транзакций, поиска по атрибутам и наличии отношений между сущностями.

Популярные хранилища этого типа: Redis, Riak, Memcached, BerkleyDb, HamsterDB, Amazon Dynamo.

Хранилище семейств колонок

В хранилищах семейств столбцов каждая строка состоит из коллекций пар имя-значение, которые называются столбцами. Столбцы хранятся с временной меткой, которая используется для того, чтобы разрешать конфликты и управлять устаревшими данными. Коллекция похожих строк формирует семейство колонок, эквивалентно таблицам в реляционных СУБД. Но строки в семействах столбцов могут иметь разные столбцы.

Подобно хранилищами ключ-значение и хранилищам документов, хранилища семейств колонок не поддержвают ACID транзакции, но взамен реализуют модель BASE. Агрегатные функции в запросах не поддерживаются. Хранилища колоночных семейств имеют высокую доступность. Они используют p2p репликацию, что означает отсутствие master узла и любой узел доступен для чтения и записи. Если в семействе столбцов какая либо колонка читается чаще других, она может быть использована как ключ для ускорения чтения. Тем не менее такой уровень доступности вносит ухудшения в согласованность.

Масштабирование кластера означает добавление новых узлов, что позволяет кластеру обрабатывать больше запросов на чтение и запись, в виду отсутствия master узла. В случае отказа какого либо узла не влияет кластера продолжает обрабаывать запросы, а выполнение функций этого узла переходит на другой узел до восстановления первого.

Хранилища семейств колонок хорошо подходят для систем анализа данных; хранения информации о событиях, таких как логи приложения; систем управления контентом, при наличии сущностей с различными наборами атрибутов, например страница или пост с различными изображениями, тегами, категориями, и т.д.

Популярные хранилища семейств колонок: HBase, Apache Cassandra, Amazon SimpleDB, Hypertable.

Документно-ориентированные СУБД

Документно-ориентированные СУБД хранят информацию в виде документов. Хранилища документов предлагают высокую производительность и возможность горизонтального масштабирования. Эти документы имеют стандартные форматы, такие как XML, PDF, JSON и т.д. Каждый документ может иметь разнородные данные.

Документы в БД адресуются с помощью уникального ключа, который представляет этот документ. Эти ключи могут быть простой строкой или строкой, которая ссылается на URI или путь к объекту.

Документно-ориентированные СУБД являются агрегатно-ориентированы подобно хранилищам пар ключ-значение. Транзакции возможны в пределах документа, который является агрегатом.

Одной из главных функций этих хранилищ является обеспечение доступности путем распределения реплик по узлам. Целостность опционально обеспечивается ожиданием синхронизации всех реплик. Обработка большого количество запросов чтения достигается добавлением slave узлов.

В случае большого количества запросов на запись используется шардинг, который реализуется подобно секционированию в реляционных СУБД. Документы автоматически распределяются по узлам для того, чтобы сбалансировать нагрузку. Эта техника позволяет добавить больше узлов производящих запись.

Документно-ориентированные СУБД могут использоваться для логирования, и систем управления контентом, в которых можно выделить объекты-агрегаты, слабо связанные друг с другом. Не рекомендуется использовать при необходимости использования транзакций и операций между агрегатами.

Среди популярных решений — MongoDB, CouchDB, Terrastore, OrientDB and RavenDB.

Графовые СУБД

Графовые СУБД хранят данные в виде графа, который состоит из узлов и ребер, где узлы являются

объектами, а ребра — отношениями между объектами. Также граф содержит параметры, которые относятся к объектам. При этом используется принцип смежности без индексов, который означает, что каждый узел имеет указатели, указывающие на смежные узлы. Также не позволяется наличие неполных отношений: начальный и конечный узлы всегда должны существовать, и узлы могут быть удалены, только если они не имеют каких-либо отношений, прикрепленные к ним.

Поскольку графовые СУБД оперируют связанными узлами, большинство имеющихся продуктов не поддерживает распределение узлов графа по различным серверам. Некоторые решения позволяют использовать master-slave репликацию для раздельной обработки запросов чтения и записи. Возможно разделение графа на уровне приложения с учетом логики предметной области, например хранение сущностей в различных узлах кластера в зависимости от географического местоположения данных. Тем не менее, поскольку такие хранилища не являются агрегатно-ориентированными, при таком подходе возможны проблемы с целостностью.

Преимуществом графовых СУБДперед другими нереляционными хранилищами является поддержка ACID транзакций. Преимуществом перед реляционными базами данных является более быстрое добавление отношений между сущностями и замена операций объединения таблиц и агрегатных функций менее затратными операциями обхода графа.

Основные области использования графовых СУБД — системы содержащие большое количество отношений, такие как социальные сети и рекомендационные системы. Обработка больших объемов информации и поиск закономерностей.

Этот тип хранилищ не следует использовать в тех случаях, когда есть необходимость в частом изменении какого либо атрибута множества узлов.

Среди популярных решений — Neo4J, Infinite Graph, OrientDB, FlockDB.

Вывод

В данной работе были описаны характеристики, преимущества и недостатки NoSQL СУБД. Также были описаны преимущества и недостатки каждого типа моделей данных и примеры их использования.

В настоящее время нереляционные модели используются для повышения продуктивности и обработки больших объемов данных. При выборе типа СУБД необходимо учитывать различные параметры, такие как доступность, целостность, избыточность, возможности масштабирования и выполнения транзакций. Также большое значение имеет предметная область проектируемой системы и такие ее особенности, как типы данныхи отношения между ними.

Література

1. Sadalage, P. J., & Fowler, M. (2013). NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Upper Saddle River, NJ: Addison-Wesley.

2. Redmond, E., Wilson, J. R., & Carter, J. (2012). Seven databases in seven weeks: A guide to modern databases and the NoSQL movement. Dallas, TX: Pragmatic Bookshelf.

3. Elmasri, R., & Navathe, S. (2016). Fundamentals of database systems. Boston: Pearson.

4. Tiwari, S. (2011). Professional NoSQL. Hoboken, NJ: Wiley.