БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра информатики

Факультет КСИС

Специальность ИиТП

Курсовой проект

по дисциплине

«Модели данных и системы управления базами данных. Часть 2»

Сравнительный анализ быстродействия современных серверных СУБД

Выполнил студент: Драгун О.В.

группа 893551

Зачетная книжка № 2520050

Минск 2022

Оглавление

[Сравнение SQL/NoSQL СУБД 3](#_Toc98072782)

[Классификация СУБД 5](#_Toc98072783)

[Сравнительный анализ быстродействия 7](#_Toc98072784)

[Обзор современных бенчмарков 9](#_Toc98072785)

[Выводы 10](#_Toc98072786)

[Источники 11](#_Toc98072787)

# Сравнение SQL/NoSQL СУБД

SQL (Structured Query Language) базы данных были основным механизмами хранения данных почти 4 десятка лет. Их популярность резко возросла в 90ых вместе с ростом сетевых приложений и решений с отрытым кодом, вроде MySQL, PostgreSQL и SQLite.

NoSQL базы данных существовали с 60ых и получили свою популярность

не так давно, с появлением MongoDB, CouchDB, Redis и Cassandra.

Основное отличие между SQL и NoSQL базам – формат представления данных. В SQL базах данные представляются в виде таблиц, где каждая строка – это запись. Структура таких бас довольно жесткая. NoSQL - базы представляют данные в виде документов, в виде пар поле-значение; часто это JSON подобный формат. Схожие документы хранятся в коллекциях, которые аналогичны SQL таблицам. NoSQL базы более гибкие, но это имеет также отрицательные стороны, часто приводящие к несогласованности данных.  
  
 В SQL базах невозможно добавлять данные, пока не будет определена схема таблицы. В дополнение к структуре информации, схема таблицы может также содержать дополнительные сведения, например **первичные ключи**, **индексы**, и, главное, отношения между различными полями. Так же предоставляется дополнительная функциональность вроде триггеров и хранимых процедур; Из-за жесткости структуры, схема должна быть тщательно проработана заранее, ибо изменения в будущем могут быть довольно сложными.

NoSQL базы не требуют наличия схемы: данные могут быть добавлены когда угодно и куда угодно. Нет необходимости устанавливать спецификацию документа или коллекции заранее. Удобно использовать это преимущество в проектах, где не определены четкие требования к данным;

В NoSQL базах, возможно создание ссылок на данные из других коллекций или же они могут быть встроены в документ, «денормализовав» его. Это ускорит выборку информации из документов, но обновления данных может оказаться значительно медленнее.

В SQL есть мощный инструмент JOIN, который позволяет осуществить выборку информации из нескольких таблиц одним запросом. В NoSQL базах, такой операции не предусмотрено; Именно в таких случаях бывает полезна «денормализация» документов, описанная выше.

SQL базы позволяют контролировать соблюдения целостности данных, с помощью ограничений(constraints): запрещать пустые поля, запрещать удалять данные, которые имеют связи с другими таблицами; Схема бд заставляет следовать правилам, невозможно изменять добавлять или удалять записи, если что-то не так. Такой инструмент не доступен для NoSQL баз.

В SQL базах несколько изменений могут быть выполнены в виде **транзакции** «все или ничего». Который гарантирует успех выполнения всех операций. В NoSQL базах, изменения документа – атомарно, то есть если мы изменяем 3 записи в одном документе, то только в случае успешного применения всех 3 изменений документ перезапишется, иначе все изменения откатятся.

NoSQL базы используют javaScript – подобные запросы с JSON подобными аргументами. Базовые операции просты, но вложенные JSON аргументы могут быть очень запутанными. SQL - очень мощный и выразительный язык запросов, однако в различных системах присутствует вариативность синтаксиса.

В вопросе **производительности**, кратко. SQL и NoSQL могут быть эквивалентны. И тут сильное влияние имеет конкретный дизайн. Часто NoSQL базы называют более быстрыми, конкретно в случае денормализованных данных NoSQL базы выполнят в один запрос выборку данных быстрее, чем SQL с JOINами по большому кол-ву таблиц.

Немаловажным является и **масштабируемость** системы. Для SQL систем это может быть затруднительным, кластеризация может быть решением, много серверов имеют доступ к центральному хранилищу, однако здесь все равно возникают сложности. В случае с NoSQL базами, более простая модель данных упрощает процесс масштабируемости, и многие из современных NoSQL баз были построены вокруг идеи о быстрой масштабируемости.

Касаемо **безопасности**, NoSQL бд – это довольно свежие решения, и они могут иметь куда больше «дыр», чем зрелые SQL решения. Также опытного персонала, умеющего работать с NoSQL базами, меньше чем таковых со знаниями SQL.

Краткое **резюме** по поводу этой темы:

SQL и NoSQL делают тоже самое(хранят данные), но немного по-разному. В этих отличиях и есть их преимущества друг перед другом.

Где хорош SQL:

Проекты с четкой логикой и требованиями к данным, которые могут быть определены заранее.

Проекты где целостность данных – ключевая ценность

Проекты где важны стандарты, проверенная технология, поддержка.

Где хорош NoSQL:

Неопределенный, изменяющиеся требования к данным

Когда необходимо начать делать проект прямо сейчас

Скорость и масштабируемость это императив

# Классификация СУБД

Теперь более подробно остановимся на классификации баз данных. Системы управления базами данных – по сути это приложения, позволяющие редактировать, хранить, создавать администрировать базы данных; Наиболее известные СУБД: MySQL, PostgreSQL, MongoDB, MariaDB, Microsoft SQL Server, Oracle, Sybase, SAP HANA, MemSQL и IBM DB2. Базы данных часто не совместимы между другими базами, но существуют стандарты вроде **SQL**, **ODBC** и **JDBC**, которые позволяют работать разным СУБД с одной базой.

Чаще всего базы данных классифицируются по модели данных, которые они поддерживают. Самая популярная – **реляционная модель**, представленная языком SQL. Как отмечалось в предыдущем разделе NoSQL базы поддерживают **документную модель.** Так же существуют Иерархические, Сетевые, Объектно-ориентрованные, Объектно-реляционные.

Так же СУБД можно классифицировать по **типу/ам компьютеров** на которых они работают (от кластера до мобильных телефонов), по **языкам запросов** SQL, XQuery, их **внутреннему дизайну**, который влияет на производительность, масштабируемость, эластичность, безопасность.

По степени распределенности: локальные, распределенные.

По способу доступа к бд: **файл-серверные**(централизованно на файлсервере, доступ через локальную сеть(Access,FoxPro) **Клиент-серверные** (Oracle, DB2, MySQL) **Встраиваемые** (SQLite, firebird);

По типу работы с внешней памятью( **непосредственная** запись, **отложенная**, через буфер)

Существуют СУБД **общего** назначения, а так же **специализированые.** Последние, естественно, имеет смысл сравнивать лишь в их узкой области применения)

Развитие СУБД можно разделить на 3 этапа, по модели данных и структуре: **навигационные**, **SQL/реляционные**, и **пост-реляционные**;

Наибольший интерес представляют современные пост-реляционные системы. Особо востребованными являются массивно-распределенные субд с высокой толерантностью к разбиению данных. Однако по теории, для распределенных система невозможно одновременное представление консистентности, доступности и разделенности.

**NewSQL** это класс современных реляционных бас данных, которые нацелены на предоставление аналогичных NoSQL возможностей масштабирования, при сохранении ACID ­гарантий традиционных субд. ScaleBase, Clustrix, EnterpriseDB, MemSQL, NuoDB, VoltDB.

# Сравнительный анализ быстродействия

Гладкая и быстрая работа СУБД является критически важным для современных решений. Поэтому базы данных имеют сложные внутренние механизмы для обеспечения быстродействия, доступности, безопасности. Если какое-то программное решение сильно опирается на хранение и обработку данных, то стоит делать это максимально эффективно, или иметь много ресурсов)

Начнём с того что почти все современные СУБД включают **Query Optimizer,** который автоматически оптимизирует каждый запрос, чтобы выбрать наиболее эффективный план запроса, определенный порядок операций; Будет специфичен для конкретного типа хранилища.

**Индексация** один из основных способов улучшения производительности, применяющийся активно в СУБД.

Многие СУБД самостоятельно умеют управлять системными ресурсами, но все равно существует огромное пространство для увеличения эффективности, с помощью тонкой настройки и конфигурирования баз данных и СУБД.

**I/O tuning** –аппаратные и программные конфигурации дисковых систем различны, RAID уровни, размеры аллокации блоков, конфигурации дисков, типы контроллеров, внешних систем хранения. Все это влияет на скорость работы баз данных. Часто объединяемые таблицы и индексы могут быть размещены так, чтобы они могли запрашиваться параллельно из хранилища.

Настройки СУБД могут включать частоту восстановления, назначения параллелизма, сетевых протоколов, памяти доступной для данных, кэширования. Естественно куда быстрее извлекать данные из памяти, нежели с диска, поэтому поддержание кэша является очень важным шагом к ускорению работы СУБД. Кэширование процедур, так же ускоряет работу, нет необходимости заново перекомпилировать их для исполнения. Многие СУБД умеют производить **автоматический тюнинг** под различными нагрузками.

На cтороне приложения, например, может сильно влиять на время отклика, подключение к бд.

Поддержка базы данных может включать дефрагментацию данных, обновление статистики, создание резервных копий. Чистка логов транзакций, анализ использования бд позволяет также оптимизировать запросы. Дефрагментация данных также ускоряет работу БД.

Современные сложные enterprise базы данных могут использовать **ИИ** и **ML** для оптимизации работы на лету.

Вернемся к нашему разговору о SQL/NoSQL базах. Тут вопрос о производительности стоит в плоскости ACID, который действительно ресурсозатратный, и будучи “универсальным” средством, в некоторых случаях может быть лишней нагрузкой.

Вот конкретный искусственный пример SQL Server/MongoDB Изображение выглядит как стол

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Эти измерения были выполнены на локальной машине и показывают только скорость работы конкретных запросов, не учитывая сложность, масштаб и требования реальной системы.

# Обзор современных бенчмарков

**TPC**Включает **TPC-C** симуляция окружения, в котором какое-то число пользователей выполняет запросы к бд; Метод основывается на выполнении различных транзакций, ввода и доставки заказов, записи платежей, проверки статусов заказа и проверки наличия товаров на складах. Проводится 5 типов одновременных транзакций разной сложности. Быстродействие измеряется в кол-ве транзакций в минуту. Основные показатели это рейт транзакций, стоимость за транзакцию;

**TPC-DI** – проверяет скорость трансформации данных из разных источников и форматов в унифицированное предствавление.

**TPC-DS** – стандартный бенчмарк для измерения производительности «decision support solutions»

**TPC-E –** моделируетброкерскую фирму и измеряет аналогичные показатели

**TPC\_VMS –** включает 3метода тестирования и используется для тестирования в виртуализированной среде.

**OSDB – Open source database benchmark**

**OO1,OO7, HyperModel – бенчмарки для объектно-ориентированных СУБД**

**Bucky, BORD – объектно-реляционные бенчмарки**

**XML – Xbench – для субд на основе xml документов**

**SAP benchmark**

# Выводы

Оценка быстродействия различных СУБД в искусственных условиях малоэффективна, ибо в конкретных условиях, на конкретной инфраструктуре, поведения СУБД может сильно различаться. Оптимизации могут сработать в конкретном случае, в случае с ML оптимизаторами, длительное использование БД так же значительно влияет на скорость получения результата. Большое количество «сравнений» баз данных, носят часто чисто рекламный характер и зачастую не несут объективной оценки быстродействия. Также очень важным остается порядок и правильность использование БД и инфраструктуры. Множество оптимизаций как на уровне приложения, платформа и инфраструктуры, могут значительно ускорить работу конкретного решения. При выборе какую базу данных использовать, стоит исходить из требований, к целостности данных, безопасности, быстродействия конкретных механизмов и простоты использования.

# Источники

1. <https://web.archive.org/web/20080905185433/http://members.value.com.au/christie/auug93.htm>
2. <https://www.cs.ubc.ca/~bestchai/papers/dbms_tuning.pdf>
3. <https://en.wikipedia.org/wiki/Database_tuning>
4. <http://web.archive.org/web/20210507050503/https://blog.michaelckennedy.net/2010/04/29/mongodb-vs-sql-server-2008-performance-showdown/>
5. <https://www.arangodb.com/2015/10/benchmark-postgresql-mongodb-arangodb/>
6. <https://arxiv.org/pdf/1701.08052.pdf>
7. <http://www.oracle.com/technetwork/database/performance/news-1537765.html>