Московский государственный колледж электромеханики и информационных технологий

ОТЧЕТ

по практическому занятию №1

**«Сравнительный анализ СУБД»**

Выполнила:

Студентка группы 3ИП-11-19

Шевчук Варвара Игоревна

Преподаватель:

Басыров Сергей Амирович

СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 3](#_Toc88043685)

[**ОСНОВНАЯ ЧАСТЬ** 4](#_Toc88043686)

[**1** **Сравнение баз данных в виде таблицы** 4](#_Toc88043687)

[**2** **Разница между MongoDB и Redis** 10](#_Toc88043688)

[**3** **Разница Разница между MySQL и Redis** 11](#_Toc88043689)

[**ЗАКЛЮЧЕНИЕ** 12](#_Toc88043690)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 13](#_Toc88043691)

# **ВВЕДЕНИЕ**

Цель – сравнить три базы данных (Redis, MySQL, MongoDB)

Актуальность – благодаря анализу и сравнению данных БД, мы сможем выявить основные преимущества и недостатки каждой из них и понять какую БД удобнее было бы использовать именно нам и в каких сферах могут быть полезны эти СУБД.

Задачи:

1) Составить общую характеристику каждой БД в виде таблицы

2) Разница между MongoDB и Redis

3) Разница между MySQL и Redis

Предмет исследования – СУБД

Объект исследования –исследование СУБД

# **ОСНОВНАЯ ЧАСТЬ**

# **1 Сравнение баз данных в виде таблицы**

Для удобного анализировано и выявления основных преимуществ и недостатков БД я решила составить таблицу, где представлено сравнение трех основных баз данных Redis, MySQL, MongoDB по пяти критериям.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | назначение субд | основные возможностеи субд | запросы в субд | Тип данных | Синтаксис SQL |
| Redis | \Redis - резидентная система управления базами данных класса NoSQL с открытым исходным кодом, работающая со структурами данных типа «ключ — значение». \Используется как для баз данных, так и для реализации кэшей, брокеров сообщений. | \Redis позволяет хранить не только строки, но и массивы (которые могут использоваться в качестве очередей или стеков), словари, множества без повторов, большие массивы бит (bitmaps), а также множества, отсортированные по некой величине. | Redis-это в основном хранилище значений ключей (немного более сложное, чем простое, но все же - база данных значений ключей).  \Кэширование данных (да, банально и скучно, но это классный инструмент для кэширования и обойти стороной этот кейс, кажется будет не правильно)  \Работа с очередями на базе redis  \Организация блокировок (mutex)  \Делаем систему rate-limit  \Pubsub — делаем рассылки сообщений на клиенты | **Строки** (strings). Базовый тип данных Redis. Строки в Redis бинарно-безопасны, могут использоваться так же как числа, ограничены размером 512 Мб.  **Списки** (lists). Классические списки строк, упорядоченные в порядке вставки, которая возможна как со стороны головы, так и со стороны хвоста списка. Максимальное количество элементов — 232 — 1.  **Множества** (sets). Множества строк в математическом понимании: не упорядочены, поддерживают операции вставки, проверки вхождения элемента, пересечения и разницы множеств. Максимальное количество элементов — 232 — 1.  **Хеш-таблицы** (hashes). Классические хеш-таблицы или ассоциативные массивы. Максимальное количество пар «ключ-значение» — 232 — 1.  Упорядоченные множества (sorted sets).  **Упорядоченное множество** отличается от обычного тем, что его элементы упорядочены по особому параметру «score». | **Назначение-SET** значение ключа  *127.0.0.1:6379> set test 123*  *OK*  **Значение**-ключ **GET**  *127.0.0.1:6379> get test*  *"123“*  **Значения задания** -Значение ключа **GetSet**  *127.0.0.1:6379> getset s2 222*  *"111"*  *127.0.0.1:6379> get s2*  *"222*  **Установка / получать множество ключей**  ключевое значение MSET [ключевое значение ...]  ключ MGET [ключ ...]  *127.0.0.1:6379> mset k1 v1 k2 v2 k3 v3*  *OK*  *127.0.0.1:6379> get k1*  *"v1"*  *127.0.0.1:6379> mget k1 k3*  *1) "v1"*  *2) "v3"*  **Удалить-DEL**  127.0.0.1:6379> del test  (integer) 1 |
| MySQL | \Свободная реляционная система управления базами данных  \MySQL взаимодействует с базой данных на языке, называемом SQL  \*SQL предназначен для манипуляции данными, которые хранятся в субд | \Полностью многопоточное использование ядерных нитей. Это означает, что пакет может легко использовать много CPUs, если они есть.  \Интерфейсы для языков C, C++, Eiffel, Java, Perl, PHP, Python и Tcl.  \Работает на многих различных платформах. | **Простые запросы:**  *SELECT count(\*) FROM table\_name;*  Выведет количество всех записей в таблице  SELECT \* FROM table\_name;  Выбирает все записи из таблицы БД  *SELECT \* FROM table\_name LIMIT 2,3;*  Выбирает 3 записи из таблицы, начиная с 2 записи. Этот запрос полезен при создании блока страниц навигации.  **Сложные запросы:**  *SELECT DISTINCT last\_name FROM person p, address adr WHERE p.adress\_no = adr.address\_no AND city LIKE 'L%';*  Выводит все уникальные фамилии людей (last\_name), которые живут в городе с названием на букву L. (предполагаем, что в таблице address есть поля address\_no, city). | **Символьные типы**  *CHAR: представляет строку фиксированной длины.*  *VARCHAR: представляет строку переменной длины.*  *TINYTEXT:*  *TEXT:*  *MEDIUMTEXT:*  *LARGETEXT:*  **Числовые типы**  *TINYINT: представляет целые числа от -128 до 127, занимает 1 байт*  *TINYINT UNSIGNED:*  *SMALLINT:*  *SMALLINT UNSIGNED:*  *INT:*  **Типы для работы с датой и временем**  *DATE:*  *TIME:*  *DATETIME:*  *TIMESTAMP:*  *YEAR:*  **Составные типы**  *ENUM:*  *SET:*  **Бинарные типы**  *TINYBLOB:*  *BLOB:*  *MEDIUMBLOB:*  *LARGEBLOB:* | \SHOW DATABASES  просмотр доступных баз данных.  \CREATE DATABASE  создание новой бд  \ USE  выбирается бд, для дальнейшей работы с ней.  *USE <database\_name>*  \ SOURCE  *SOURCE <file.sql>* позволит выполнить сразу несколько SQL-команд, содержащихся в файле с расширением .sql.  \ DROP DATABASE  удаление  \ CREATE TABLE  Создание табл  Например  *CREATE TABLE instructor (*  *ID CHAR(5),*  *name VARCHAR(20) NOT NULL,*  *dept\_name VARCHAR(20),*  *salary NUMERIC(8,2),*  *PRIMARY KEY (ID),*  *FOREIGN KEY (dept\_name) REFERENCES department(dept\_name)*  *);*  \ DELETE  *DELETE FROM <table\_name>*  *\* *DROP TABLE*  Полностью удаляет |
| MongoDB | \документоориентированная система управления базами данных, не требующая описания схемы таблиц.  \считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. | \Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)  \Достаточно гибкий язык для формирования запросов  \Динамические запросы  \Полная поддержка индексов  \Профилирование запросов  \Быстрые обновления "на месте" | **Find** — аналог SELECT в MySQL. Используется для выборки документов из MongoDB. Возвращает массив документов в виде коллекции, если документов нет — пустую коллекцию  Например  *> db.users.find();*  Вернёт всех пользователей из коллекции.  *> db.users.find( { age: 27 } );*  Вернёт всех пользователей, у которых возраст равен 27.  **Запросы с условием**  Операторы: $lt — меньше, $lte — меньше или равно, $gt — больше, $gte — больше или равно, $ne — не равно.  Пример  Получаем всех пользователей, возраст которых больше 18 и меньше 30  *> db.users.find( { age: { $gte: 18, $lte: 30 } } );*  **Запросы в массивах**  Допустим есть у нас коллекция food и мы туда вставляем документ с массивом фруктов  *> db.food.insert( { "fruit" : [ "apple", "banana", "peach" ] } );* | **String** — это наиболее часто используемый тип данных для хранения данных. Строка в MongoDB должна быть действительной в формате UTF-8.  **Integer** — этот тип используется для хранения числового значения. Целое число может быть 32-разрядным или 64-разрядным в зависимости от вашего сервера.  Boolean,  Double, | Основной синтаксис использования оператора DATABASE следующий:  *use DATABASE\_NAME*  бд с именем  *>use mydb*  *switched to db mydb*  Проверка бд  *>db*  *mydb*  Проверка списков  *>show dbs*  *local 0.78125GB*  *test 0.23012GB*  Для отображения базы данных вам необходимо вставить в нее хотя бы один документ.  *>db.movie.insert({"name":"tutorials point"})*  *>show dbs*  *local 0.78125GB*  *mydb 0.23012GB*  *test 0.23012GB*  Удаление бд  *db.dropDatabase()* |

# **Разница между MongoDB и Redis**

MongoDB больше похож на MySQL, индекс поддержки на местах, курсоров операций, его преимуществом является более мощной функцией поиска, хорошие данные JSON - запрос, может хранить огромные объемы данных, но не поддерживает транзакции.

MySQL, существенно уменьшается, когда большое количество данных, MongoDB чаще в качестве альтернативы реляционной базы данных.

**Механизм управления памятью**

Redis данные существует всю память, диск запись на регулярной основе, если память недостаточно, вы можете выбрать конкретный алгоритм LRU для удаления данных. Данные MongoDB в памяти, система Linux ММАПА реализации, когда память не хватает, только горячие данные в память, и другие данные, хранящиеся на диске.

**Поддержка структуры данных**

Богатых поддержки структур данных Redis, включая хэш, набор, список и так далее.Структура данных MongoDB является относительно простой, но выразила поддержку богатых данных, индекс, по аналогии с большинством реляционных языка запросов к базе данных, поддерживаемого очень богатым.

**Кластер**

Технология кластера MongoDB является более зрелым, Redis от 3.0 начал поддерживать кластеры.

# **Разница между MySQL и Redis**

MySQL является постоянной памяти, хранится на диске внутри, извлекается, который будет включать в себя некоторые IO, для того, чтобы решить эту узкое место, так что есть кэш, например, в настоящее время является наиболее часто используемым Memcached (называемый тс). Во- первых, MC доступа пользователей, если не попал, навестил MySQL, после того, как память и жесткий диск, как часть скопированных данных в тс.

Redis и тс кэшируются, и запустить все находятся в памяти, что значительно повышает скорость доступа к вебдоступа большого количества данных. Однако MC только обеспечивает простую структуру данных, такие как хранение строки, а Redis предоставил больше количество структур данных, такие как строки, список, набор, HashSet, отсортированный набор из них, что делает его удобным для многих пользователей, после того, как слой герметизирующего практичным функции, в то время как достижения того же эффекта, конечно, с Redis медленно отказаться тс.

Отношения между памятью и жестким диском, жестким диском, чтобы поместить основные данные для постоянного хранения и памятью является то, что часть данных, работающих в данный момент, доступ к памяти процессора вместо диска, что значительно повышают скорость работы, конечно, это основано на локализованной программе принцип доступа.

Рассуждая на Redis + MySQL, это соотношение дисковой памяти + карты, MySQL на диске, Redis в памяти, в этом случае, веб - приложений, доступ к времени Redis, если данные не найдены, прежде чем посетить MySQL.

Однако использование Redis + MySQL + диск и память предпочтительно является различным.

# **ЗАКЛЮЧЕНИЕ**

Благодаря данной практической работе я поняла основные отличия между тремя базами данных (Redis, MySQL, MongoDB). Провела анализ каждой из них и выявила преимущества и недостатки каждой.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. <https://www.codetd.com/ru/article/6301259>
2. <https://intellect.icu/redis-varianty-ispolzovaniya-sravnenie-s-mysql-storage-engine-memory-8048>
3. <https://habr.com/ru/company/manychat/blog/507136/>