J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

MySQL과 Redis의 데이터 처리 성능 비교 평가☆

Comparative Evaluation of Data Processing Performance between MySQL and Redis

방 혁[1†] 김 서 현[2†] 전 상 훈[1*]
Hyeok Bang Seo-Hyeon Kim Sanghoon Jeon

요 약

최근 디지털 변화와 코로나19의 영향으로 온라인 활동이 급증함에 따라 대규모 데이터 처리와 유지보수의 중요성이 점점 커지고 있다. 이 연구는 데이터 저장 및 관리에 널리 사용되는 두 주요 데이터베이스 유형인 관계형 데이터베이스(RDBMS)와 비관계형 데이터베이스(NoSQL)의 성능을 비교 분석한다. 구체적으로, RDBMS의 대표 예인 MySQL과 NoSQL의 대표 예인 Redis를 사용하여 데이터 삽입, 조회, 삭제 기능의 수행 시간을 측정하고 평가했다. 실험 결과, Redis는 MySQL에 비해 데이터 삽입에서 약 5.84배, 조회에서는 약 6.61배, 삭제에서는 약 12.33배 빠른 성능을 보였다. 이 결과는 Redis가 특히 대규모 데이터 처리와 유지보수가 필요한 환경에서 뛰어난 성능을 제공함을 보여준다. 이에 따라 기업 및 온라인 서비스 제공자들은 Redis와 같은 NoSQL 데이터베이스를 선택함으로써 보다 효율적인 데이터 관리 솔루션을 확보할 수 있을 것이다. 본 연구가 데이터베이스 선택 시 데이터 처리 성능을 고려하는 데 중요한 참고 자료로 활용되기를 기대한다.

☞ 주제어 : 데이터, 관계형 데이터베이스, 비 관계형 데이터베이스

ABSTRACT

As online activities have rapidly increased due to recent digital changes and the impact of COVID-19, the importance of large-scale data processing and maintenance is increasing. This study compares the performance of the two main types of databases widely used for data storage and management: Relational Database Management Systems (RDBMS) and Non-Relational Databases (NoSQL). Specifically, we measured and evaluated the execution time of data insertion, query, and deletion functions using MySQL, a representative example of RDBMS, and Redis, a representative example of NoSQL. The experimental results showed that Redis showed performance about 5.84 times faster in data insertion, 6.61 times faster in query, and 12.33 times faster in deletion than MySQL. These results demonstrate that Redis provides superior performance, especially in environments requiring large-scale data processing and maintenance. Therefore, companies and online service providers can choose NoSQL databases such as Redis to ensure more efficient data management solutions. We hope this study will be an essential reference when selecting a database based on data processing performance.

☞ keyword : Data, Relational Database Management System, Not only SQL

1. 서 론

최근 급속한 디지털 변화, 특히 코로나19의 급속한 확산으로 인터넷 산업이 이전보다 더 중요한 역할을 하게 되었다[1]. 글로벌 비상사태로 인한 온라인 활동 증가로 인하여 온라인 서비스에 대한 수요가 급증하면서, 기업 및 온라인 서비스 제공자들은 대규모 데이터 처리와 유지보수에 대한 과제에 직면하고 있다[2].

데이터 저장 및 관리를 위해 주로 사용되는 두 가지 주요 데이터베이스 유형은 관계형 데이터베이스(RDBMS, Relational Database Management System)와 비관계형 데이터베이스(NoSQL, Not only SQL)이다[3]. RDBMS는 데이터를 테이블 형태로 구조화하여 저장함으로써, 데이터의 일관성과 무결성을 보장한다[4]. 이러한 구조는 안정적이고 신뢰성 있게 데이터를 관리하는 데에 효과적이다. 그러나 복잡한 쿼리 및 대량의 데이터 처리

1 Department of Information Security, The University of Suwon, 17, Wauan-gil, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 18323, Republic of Korea
2 Department of Information Communication, The University of Suwon, 17, Wauan-gil, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 18323, Republic of Korea
† Co-first authors (bh1848@suwon.ac.kr and valent9@suwon.ac.kr)
* Corresponding author (shjeon@suwon.ac.kr)
[Received 13 April 2024, Accepted 13 May 2024]
☆ 본 연구는 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2021R1I1A1A0104094313)

에서는 성능 저하가 발생할 수 있으며, 수평적 확장이 어려워 대규모 데이터 처리에 제한이 있다.

반면, NoSQL은 유연한 데이터 저장 구조를 통해 높은 확장성과 병렬 처리를 가능하게 하여, 대규모 데이터베이스 환경에서 높은 성능을 제공한다. 예를 들어, 데이터를 수평으로 확장할 수 있으며, 쓰기 및 읽기 작업을 병렬로 처리할 수 있어 고성능을 제공한다. 특히, NoSQL 데이터베이스 중에서는 메모리 기반의 데이터 처리를 제공하는 Redis가 뛰어난 성능을 제공한다[5]. 하지만 NoSQL은 일관성 모델의 한계를 가질 수 있다. 일부 NoSQL 시스템은 느슨한 일관성을 제공하며, 데이터의 일관성이 시간에 따라 보장되지 않을 수 있다.

RDBMS는 보안과 데이터 무결성이 필수인 정부 및 규제 기관에서 주로 사용되며, 은행 시스템, 예약 시스템, 주문 처리 시스템 등 트랜잭션 처리가 중요한 금융 및 예약 시스템 등에서도 활용된다. 그러나 RDBMS를 사용하는 대부분의 웹 서비스에서는 대량의 접속 로그를 처리하는 과정에서 성능 저하 문제가 발생할 수 있다. 이 문제를 해결하기 위해, 테이블 간의 연결이 필요가 없는 데이터에 대해서는 NoSQL과 같은 메모리 기반의 데이터 저장 방식을 적용함으로써 처리 속도를 효과적으로 개선할 수 있다. 특히, 불필요한 연결이 없는 경우에는 Redis를 활용하여 데이터 처리 속도를 높일 수 있다[6]. Redis는 NoSQL 데이터베이스 중에서 가장 빠른 속도를 제공한다[7]. 그리고, 최근 연구에 따르면 대용량 데이터 처리에서 NoSQL이 기존 RDBMS인 MySQL, PostgreSQL, Oracle 등 보다 우수한 성능을 보여준다[8][9][10].

본 논문은 RDBMS와 NoSQL의 성능을 비교 분석한다. 이를 위해 RDBMS 대표 예인 MySQL과 NoSQL의 대표적인 예인 Redis를 활용하여 데이터 처리 함수(삽입, 조회, 삭제)의 시간을 측정하고 평가한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 MySQL과 Redis의 주요 차이점을 설명한다. Ⅲ장에서는 MySQL과 Redis의 성능을 비교하기 위한 평가지표와 실험 방법에 대해 자세히 기술한다. Ⅳ장에서는 MySQL과 Redis의 데이터 처리 성능을 검증한다. Ⅴ장에서는 결론 및 앞으로의 연구 계획을 제시하며 마무리한다.

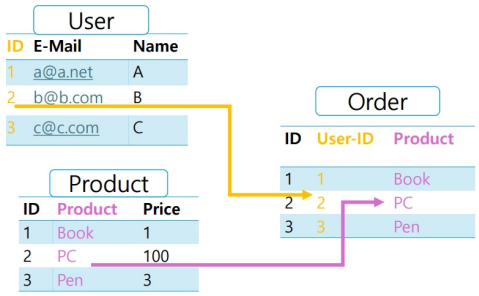
2. MySQL과 Redis

2.1 RDBMS 구조 소개

RDBMS는 데이터베이스에 저장된 구조, 즉 데이터 스키마를 명확하게 정의하는 것이 중요하다. 각 테이블은 여러 열(column)을 가지며, 각 열은 특정 데이터 유형(예: 문자열, 숫자, 날짜 등)을 가진다. 테이블의 각 행(row)은 하나의 데이터 레코드를 나타내며, 각 열의 데이터 유형에 맞는 값이 저장된다. 즉, 데이터의 모든 내용은 정의된 열을 준수해야 하며, 해당 형식에서만 데이터 조작을 수행할 수 있다[11].

2.2 MySQL의 구조 및 동작 원리

MySQL은 오픈 소스 RDBMS로 가장 널리 사용되는 데이터베이스 중 하나이다. MySQL의 구조는 모든 데이터를 2차원 테이블 형태로 표현한다. 테이블은 하나 이상의 열과 행으로 이루어져 있다. 그림 1은 MySQL 구조를 나타낸다. User 테이블의 ID는 Order 테이블의 User-ID와 연결되어 있고, Product의 Product는 Order의 Product와 연결되어 있어 테이블과 테이블 간 관계를 맺고 있는 형태이다[12]. 따라서 정해진 스키마에 따라 데이터를 저장해야 하고, 명확한 데이터 구조를 보장하는 장점이 있음과 동시에 정해진 스키마로 인해 데이터가 유연하지 못하다는 단점이 있다[13].

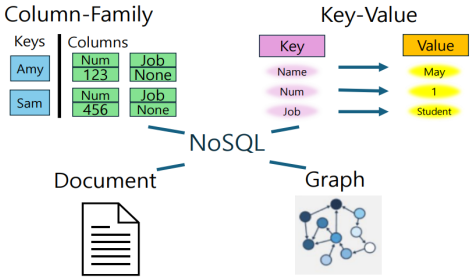


(그림 1) MySQL 구조 예시
(Figure 1) MySQL Structure Example

2.3 NoSQL 구조 소개

NoSQL 구조는 크게 네 가지로 분류할 수 있다[14]. 그림 2와 같이, Column-Family, Key-Value, Document, Graph 구조로 나뉜다. 첫째, Column-Family는 전통적인 RDBMS와 같이 테이블 형태로 보이지만, RDBMS와는 다르게 행보다는 열에 데이터를 저장하는 구조다[15]. 이는 주로 대규모 빅데이터 분야에서 활용되며, 하나의 열에 방대한 양의 데이터를 추가할 수 있다. 둘째, Key-Value는 여러 종류의 NoSQL 중 가장 간단한 구조이고, Key 값과

Value 값이 쌍을 이루어 데이터를 저장한다. 저장 형태의 단순함으로 인해 뛰어난 확장성을 제공한다. 셋째, Document 구조는 데이터를 테이블이 아닌 문서 형태로 저장하며[16], 주로 JSON과 유사한 형식으로 데이터를 저장한다. 이런 특정 문서 형태를 가지는 구조 때문에 Key-Value 구조보다 한 단계 높은 복잡성을 보인다. 마지막으로, Graph 구조는 NoSQL 구조 중 가장 높은 복잡도를 가진 종류이다. 주로 객체들 사이의 관계를 저장하는 것이 중요한 경우 사용한다. 객체와 간선을 사용하여 관계와 방향을 표시한다.

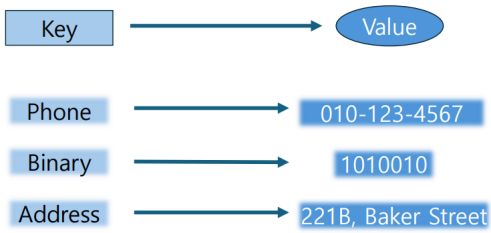


(그림 2) NoSQL 구조 예시
(Figure 2) NoSQL Structure Example

2.4 Redis의 구조 및 동작 원리

Redis는 Key-Value의 구조를 가진다. 이 구조는 데이터베이스를 키-값 쌍의 형태로 저장하는 NoSQL 유형으로, 각 키는 고유한 식별자 역할을 하여 다양한 데이터를 저장한다. 그림 3은 키-값 기반 데이터베이스를 나타내는 Redis 구조를 나타낸다. 이 방식은 단순한 객체부터 복잡한 집합체에 이르기까지 모든 것을 키와 값으로 표현할 수 있으며, 다른 데이터베이스 유형에서는 어려운 수평적 확장성을 제공한다[17].

Redis는 키-값 기반의 인메모리(In-memory) 데이터베이스로서, 데이터를 메모리에 직접 저장하고 관리한다. 이는 디스크 기반의 MySQL과 달리 더 빠른 속도로 데이터를 처리할 수 있다. 또한, Redis는 MySQL과 달리 고정되어 있지 않은 테이블 스키마를 갖는다. 그렇기에 데이터 간의 관계를 정의하지 않고, 사전에 스키마를 정의하지 않아도 데이터를 저장할 수 있는 유연성을 제공한다[18]. 데이터를 저장하는 열이, 각각 다른 이름을 가지거나 다른 데이터 형태로 존재하는 것이 가능하다. 이러한 특징은 Redis를 효율적이고 확장 가능한 데이터 저장 솔루션으로 만든다.



(그림 3) 키-값 기반 Redis 구조
(Figure 3) Key-Value Based Redis Structure

2.5 RDBMS와 NoSQL의 비교

RDBMS는 데이터를 디스크에 저장하며 쿼리 성능은 인덱스와 디스크 I/O(Disk Input/Output)에 크게 의존한다[19]. 따라서 복잡한 쿼리와 조인 연산을 지원하지만, 대규모 데이터 처리에서는 성능이 저하될 수 있다. 반대로 NoSQL은 모든 데이터를 메모리에 저장하여 매우 빠른 읽기와 쓰기 성능을 제공한다[20].

RDBMS 구조인 MySQL은 정형화된 데이터와 복잡한 쿼리, 트랜잭션 처리에 적합하지만, NoSQL구조인 Redis는 고속 데이터 접근, 대용량 데이터 처리, 실시간 데이터 처리가 필요한 경우에 더 적합하다[21].

3. 성능 평가 실험 방법

3.1 데이터 처리 성능 평가 방법

본 성능 평가는 스프링 부트(Spring Boot)와 자바(Java) 언어를 사용하여 진행한다. MySQL과 Redis의 데이터 처리 성능을 비교 평가하기 위해, 데이터 삽입, 데이터 조회, 데이터 삭제 함수를 스프링 부트 환경에서 개발한다. 보다 정확한 측정을 위하여 본 논문에서 다루는 데이터베이스 성능 측정을 위한 프로세스 이외에는 아무것도 실행하지 않는 상태로 진행한다. 그리고, 각 함수의 성능 평가 실험에 소요된 시간을 측정한 후 로그 형태로 기록한다. 기록된 데이터를 통해 두 데이터베이스 시스템의 성능을 비교 분석한다.

3.1.1 MySQL 성능 평가

MySQL에 대한 데이터 처리 성능을 평가하기 위해 특정 실험을 수행한다. 이 실험에서는 데이터 삽입, 조회, 삭제 작업을 각각 10,000회 실행하며, 이 과정에서 무작위로 생성된 정수형 데이터를 사용한다. 각 함수의 성능

은 1,000회 실행마다 평균 수행시간을 측정하여 기록한다. 이러한 측정은 총 10번 반복하여 전체적인 평균 시간을 계산한다. 각 함수의 측정 결과는 로그 파일로 저장하고 성능 평가에 사용한다.

데이터 삽입 함수는 10,000개의 무작위 정수형 데이터를 객체 형태로 만들어 MySQL 데이터베이스에 저장한다. 데이터 조회 함수는 무작위 정숫값으로 생성된 특정 ID에 대한 데이터를 조회하는 작업을 수행한다. 데이터 삭제 함수는 무작위로 생성된 특정 ID에 해당하는 데이터의 존재 여부를 확인한 후, 데이터가 존재할 경우 해당 데이터를 삭제한다.

3.1.2 MySQL 쿼리

데이터 삽입, 조회, 삭제문의 쿼리는 각각 INSERT, SELECT, DELETE를 이용하여 구성된다. 삽입문은 'INSERT INTO TEST_TABLE (ID, DATA) VALUES (?, ?)'를 이용하여 성능 실험용 테이블인 TEST_TABLE에 새로운 행을 삽입한다. ID는 기본 키, DATA는 데이터를 뜻한다. 조회문은 'SELECT * FROM TEST_TABLE WHERE ID = ?'를 사용하여 해당 행의 데이터를 조회하도록 하였다. 특히 다른 데이터보다 기본 키인 ID를 사용하여 불필요한 디스크 I/O를 줄여 효율적인 탐색이 가능하도록 하였다. 삭제문은 'DELETE FROM TEST_TABLE WHERE ID = ?'를 사용하여 해당 행의 데이터를 지우도록 하였다. 삭제문 또한 조회문과 마찬가지로 기본 키인 ID를 사용하였다. 각 쿼리문은 ID를 기본 키로 사용하여 데이터베이스의 성능을 향상시켰다. INSERT, UPDATE, DELETE를 사용하여 데이터 무결성 또한 지켜지도록 하였다.

3.1.3 Redis 성능 평가

Redis는 데이터를 Redis의 문자열 형식으로 키-값 쌍을 저장하는데, 여기서 키는 'key' 문자열에 순번을 붙인 문자열이고 값은 0부터 999까지의 무작위 정수이다. 이렇게 만들어진 무작위 데이터 10,000개를 이용하여 수행시간을 측정하여 Redis의 성능을 평가한다. 각 함수의 측정 결과는 로그 파일로 저장하고 성능 평가에 사용한다.

데이터 삽입 함수는 키-값 쌍으로 만들어진 데이터를 삽입하고, 데이터 조회 함수는 무작위 정숫값으로 생성된 특정 키에 대한 데이터를 조회한다. 데이터 삭제 함수는 무작위 정숫값으로 생성된 특정 키에 대한 데이터를 삭제할 때, 삭제할 키를 이용하여 Redis에서 데이터를 삭제한다.

3.2 실험 환경 구축 및 성능 평가

본 실험은 Windows 11의 노트북 환경에서 진행되며, 16.0 GB의 RAM을 사용한다. 64비트 운영 체제로, x64 기반 프로세서의 시스템 종류를 사용하며 프로세서는 13th Gen Intel(R) Core(TM) i5-1340P, 1.90 GHz이다.

소프트웨어 개발 환경은 Java와 스프링 부트를 이용하였고, MySQL 및 Redis의 클라이언트 라이브러리를 프로젝트에 추가하여 이 두 데이터베이스 시스템과 통신할 수 있도록 한다.

MySQL과 Redis를 설치한 후 스프링 부트를 사용하여 데이터 삽입, 조회, 삭제 기능을 각각 수행하는 애플리케이션을 개발하여 MySQL 및 Redis와 상호 작용할 수 있는 데이터베이스 연결 및 쿼리를 구현한다. 이 애플리케이션을 실행하여 MySQL 및 Redis에서 각각의 기능을 10,000회씩 수행하여, 1,000회마다 평균 수행시간을 측정하여 총 10회 반복한 결과를 기록한다. 이 결과를 바탕으로 두 데이터베이스의 성능을 비교 및 평가한다.

3.2.1 Redis 쿼리

RedisTemplate을 사용하여 키-값 쌍을 Redis에 저장할 수 있도록 쿼리를 구성하였다. 삽입문은 'redisTemplate.opsForValue().set(key, keyIndex))'를 사용하여 MySQL에 저장되는 것과 비슷하게 두 가지의 데이터가 저장되도록 하였다. key와 keyIndex는 각각 키-값 쌍이다. 조회문은 'redisTemplate.opsForValue()..get(key)'를 사용하였다. 키를 사용하여 값을 조회하는 방식이다. 삭제문은 'redisTemplate.opsForValue().delete(key)'를 사용하였다. 조회문과 같은 방식으로 키를 사용하지만 조회한 값을 삭제하는 쿼리문이다. 각 쿼리문은 키를 이용하여 값을 조회하는 키-값 데이터베이스의 특징을 살려 작성되었다. 데이터베이스에 부하가 가지 않도록 적절히 분산되어 실행되도록 하였다.

4. 성능 평가

4.1 MySQL 성능 평가

먼저, MySQL에서 성능을 평가한다. 표 1은 삽입, 조회, 삭제 기능을 각각 10,000번씩 수행하고, 1,000회당 1번 평균 소요 시간을 산출한 결과를 나타낸다. 데이터 삽입은 평균 소요시간은 평균 1.3763ms, 데이터 조회는 평균 1.055ms, 데이터 삭제는 평균 1.7473ms가 걸렸다. 이

결과는 MySQL 데이터베이스의 각 기능별 처리 속도를 구체적으로 보여준다.

(표 1) MySQL에서 데이터 처리 성능 평가 결과
(Table 1) Data processing performance evaluation results in MySQL

회차	삽입	조회	삭제
1-1,000회	1.622	1.163	1.977
1,001-2,000회	1.306	1.012	1.692
2,001-3,000회	1.173	0.863	1.465
3,001-4,000회	1.177	0.915	1.554
4,001-5,000회	1.971	1.54	2.58
5,001-6,000회	1.454	1.107	1.846
6,001-7,000회	1.372	1.075	1.812
7,001-8,000회	1.242	0.973	1.473
8,001-9,000회	1.414	1.056	1.734
9,001-10,000회	1.032	0.846	1.340
평균	**1.3763(ms)**	**1.055(ms)**	**1.7473(ms)**

4.2 Redis 성능 평가

다음은 Redis의 성능 평가 결과이다. Redis는 키와 값의 쌍을 이루는 키-값 기반 데이터베이스이므로, 데이터 삽입 시에 각 데이터값에 키를 부여했다.

표 2는 Redis의 삽입, 조회, 삭제 함수에 대한 데이터 처리 성능을 보여준다. 데이터 삽입은 평균 결과가 0.2357ms, 데이터 조회는 평균 0.1595ms, 데이터 삭제는 평균 0.1417ms가 소요되었다. 이 결과는 Redis 데이터베이스의 각 기능별 처리 속도를 구체적으로 보여준다.

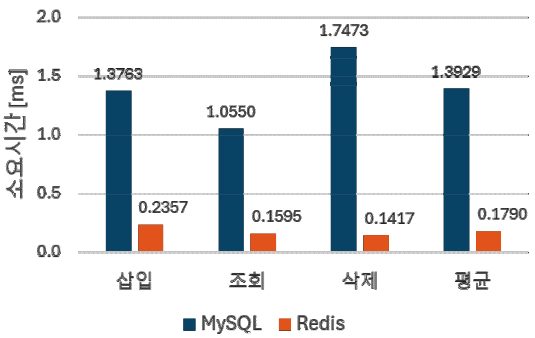
(표 2) Redis에서 데이터 처리 성능 평가 결과
(Table 2) Data processing performance evaluation results in Redis

회차	삽입	조회	삭제
1-1,000회	1.074	0.213	0.185
1,001-2,000회	0.137	0.137	0.114
2,001-3,000회	0.122	0.143	0.146
3,001-4,000회	0.13	0.14	0.121
4,001-5,000회	0.126	0.132	0.105
5,001-6,000회	0.158	0.182	0.165
6,001-7,000회	0.15	0.148	0.102
7,001-8,000회	0.142	0.173	0.17
8,001-9,000회	0.131	0.122	0.129
9,001-10,000회	0.187	0.205	0.18
평균	**0.2357(ms)**	**0.1595(ms)**	**0.1417(ms)**

4.3 결과 비교 및 정리

본 섹션에서는 MySQL과 Redis의 기능별 성능을 비교하고 평가한다. 표 1과 표 2의 실험 결과를 바탕으로 두 데이터베이스의 처리 성능을 분석한다.

Redis는 데이터 처리 성능에서 MySQL에 비해 뛰어난 결과를 보여주었다. 데이터 삽입에서 Redis는 MySQL보다 약 5.839배 빠르며, 데이터 조회에서는 약 6.614배 더 빠른 성능을 제공하였다. 데이터 삭제 작업에서는 Redis가 MySQL보다 약 12.331배 더 빠른 결과를 나타내었다. 그림 4는 MySQL과 Redis의 기능별 성능 평가 결과를 시각적으로 나타낸다. 데이터 삽입, 조회, 삭제 모두에서 Redis의 함수 수행 시간이 MySQL보다 짧은 것을 보여준다. 구체적으로, 삽입은 MySQL 대비 Redis가 5.84배, 조회는 6.61배, 삭제에서는 12.33배 빠르고, 전체 평균으로는 7.78배 빠른 성능을 보였다. 이러한 결과는 Redis가 MySQL에 비해 데이터 처리 속도 면에서 상당한 우위를 가지고 있음을 명확히 보여준다.



(Figure 4) Comparison of data processing performance between MySQL and Redis

5. 결론 및 향후 연구 계획

본 논문은 대규모 데이터 처리 및 유지보수를 위한 두 가지 주요 데이터베이스, MySQL과 Redis의 데이터 처리 성능을 비교 평가한다. 데이터 처리 성능 실험을 통해, Redis가 데이터 삽입, 조회, 삭제 기능에서 MySQL에 비해 우수한 성능을 보였다. Redis는 인메모리 데이터베이스 특징으로 인해, MySQL 대비 빠른 데이터 처리 속도를 제공한다. 이러한 결과는 대규모 데이터 처리와 유지보수가 필요한 기업 및 온라인 서비스 제공자들에게 중

요한 참고 자료를 제공한다.

향후 연구에서는 Redis 성능을 더욱 향상시킬 수 있는 방안을 탐구할 것이다. 또한 다양한 데이터베이스 유형과의 성능 비교를 통해 보다 효과적인 데이터 관리 전략을 제안할 것이다. 데이터 처리 성능을 고려한 최적의 데이터베이스 선택은 필수적이며, 이러한 연구를 통해 현대 디지털 환경에서 데이터 관리의 효율성을 향상시키는 데 기여할 것으로 기대한다.

참고문헌(Reference)

[1] Yoon Ju Jeong, “Internet usage time has increased due to the covid-19… Messenger usage among people in their 70s is also ‘worsening’,” Yeonhap News, 2021.
https://yna.co.kr/view/AKR20210303084400017

[2] Lisa Richwine, “Netflix says it has fixed outage that hit some in U.S., UK,” Reuters, March 26, 2020.
https://www.reuters.com/article/us-netflix-outages/netflix-says-it-has-fixed-outage-that-hit-some-in-u-s-uk-idUSKBN21C337/

[3] Humberto Farias, “From RDB to search engines… Choosing the DB that’s right for you,” CIOKorea, April 25, 2018.
https://www.ciokorea.com/news/38041

[4] Jin Ok Jung, “NoSQL market growing 31.4% annually, reaching ‘$22.087 billion’ by 2026,” GTT KOREA, October 12, 2023.
https://www.gttkorea.com/news/articleView.html?idxno=7092

[5] Bob Violino, “‘From MongoDB to Couchbase’… Choosing NoSQL that’s right for your company,” CIOKorea, March 13, 2018.
https://www.ciokorea.com/tags/2428/NoSQL/37551

[6] Chul-Ho Kim, Kyeong-Won Park, and Yong-Lak Choi, “Web Service Performance Improvement with the Redis,” Journal of the Korea Institute of Information and Communication Engineering, Vol. 19, No. 9, pp. 2064-2072, 2006.
http://dx.doi.org/10.4093/jkda.2006.30.2.87

[7] Cristian Andrei BARON, “NoSQL Key-Value DBs Riak and Redis,” Database Systems Journal, Vol. 6, No. 4, pp. 3-10, March 3, 2015.
https://www.dbjournal.ro/archive/22/22.pdf#page=4

[8] Hong-Jin Park, “A Study about Performance Evaluation of Various NoSQL Databases,” Journal of the Korea Institute of Information and Communication Engineering 16-06, Vol. 9, No. 3, pp. 298-305, 2016.
https://koreascience.kr/article/JAKO201621650894999.page

[9] Seon Pil Ko, “A Study on Comparative Performance Analysis of RDBMS and NoSQL for Large Data Processing,” Domestic Master’s Thesis Konkuk University Graduate School of Information and Communication, 2012.
https://www.riss.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=336823cd83ad1281ffe0bdc3ef48d419&keyword=rdbms%20nosql

[10] Seung Hwan Lee, Jae Min Yun, Donghyun Kwon, Sung Hyun Lee, Dae Gyu Hwang, Ji Weon Kim, and Se Jin Park, “MongoDB, MySQL Performance Comparison for Blind Date App Development,” Proceedings of KIIT Conference, pp.1063-1067, 2023.
https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE11485680

[11] Satya Pasupuleti, “Understanding Schema in SQL,” Hackernoon, June 15, 2023.
https://hackernoon.com/understanding-schema-in-sql

[12] Manuel Silverio, “What Is a Relational Database?,” Builtin, December 21, 2022.
https://builtin.com/data-science/relational-database

[13] Peter Wayner, “8 Disadvantages of MySQL,” CIOKorea, July 17, 2015.
https://ciokorea.com/news/26043

[14] Denis Larionov, “NoSQL: System Design Cheat Sheet,” Hackernoon, October 31, 2023.
https://hackernoon.com/nosql-system-design-cheat-sheet

[15] Alex Williams, “NoSQL Database Types Explained: Column-Oriented Databases,” TechTarget, September 22, 2021.
https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases

[16] Matthew Tyson, “‘Invitation to NoSQL’ Introduction to MongoDB for Developers,” IT World, July 7, 2021, https://itworld.co.kr/news/200094

[17] Alex Williams, “Key-Value Databases, Explained,” KDnuggets, October 4, 2022. https://kdnuggets.com/2021/04/nosql-explained-understanding-key-value-databases.html

[18] Serdar Yegulalp, “Cloud Scale!… NoSQL Compared to SQL,” CIOKorea, June 28, 2022. https://ciokorea.com/tags/1988/SQL/242074

[19] A. Bhushan, P. Rastogi, H. K. Sharma and M. E. Ahmed, “I/O and memory management: Two keys for tuning RDBMS,” 2016 2nd International Conference on Next Generation Computing Technologies(NGCT), Dehradun, pp. 208-214, 2016. https://ieeexplore.ieee.org/abstract/document/7877416

[20] A. Bhushan, P. Rastogi, H. K. Sharma and M. E. Ahmed, “A Study about Performance Evaluation of Various NoSQL Databases,” Journal of the Korean Society of Information, Electronics and Communication Technology Vol. 9, No. 3, pp. 298-305, 2016. https://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE06696379

[21] Eun Ki Kim, “Research on Utilizing NoSQL by Comparison of Processing Large Scale Data in MongoDB and MySQL,” Domestic Master’s Thesis Soongsil University, 2016. https://www.dbpia.co.kr/journal/detail?nodeId=T14144757

◐ 저 자 소 개 ◑

방 혁(Hyeok Bang)
2020년~현재 수원대학교 정보보호학과 학사과정
관심분야 : 데이터베이스, etc.
E-mail : bh1848@suwon.ac.kr

김 서 현(Seo-Hyeon Kim)
2020년~현재 수원대학교 정보통신학과 학사과정
관심분야 : 데이터베이스, 백엔드 개발, API 개발, etc.
E-mail : valent9@suwon.ac.kr

전 상 훈(Sanghoon Jeon)
2012년 경북대학교 IT대학 심화 전자공학 공학사
2014년 대구경북과학기술원 정보통신융합공학전공 공학석사
2020년 대구경북과학기술원 정보통신융합전공 공학박사
2020년 한양대학교 산학협력단 선임연구원
2020년~2022 한양대학교 의과대학 응급의학과 포닥연구원
2022년~2023 한양대학교 의과대학 응급의학과 연구조교수
2023년~현재 수원대학교 지능형SW융합대학 정보보호학과 조교수
관심분야 : 웨어러블컴퓨팅, 의료인공지능, CPS보안
E-mail : shjeon@suwon.ac.kr