DBMS 비교 분석

DBMS : DataBase Managemant System 의 약자로 다수의 사용자가 DB에 접근할 수 있게 해주는 시스템이다. 사용자 혹은 프로그램의 요구에 적절한 응답을 통해 데이터의 사용을 가능케 해준다.

다음은 위키백과에서 설명한 DBMS의 기능이다.

* 정의 : 데이터에 대한 형식, 구조, 제약조건들을 명세하는 기능이다. 이때 데이터베이스에 대한 정의 및 설명은 카탈로그나 사전의 형태로 저장된다.
* 구축 : DBMS가 관리하는 기억 장치에 데이터를 저장하는 기능이다.
* 조작 : 특정한 데이터를 검색하기 위한 질의, 데이터베이스의 갱신, 보고서 생성 기능 등을 포함한다.
* 공유 : 여러 사용자와 프로그램이 데이터베이스에 동시에 접근하도록 하는 기능이다.
* 보호 : 하드웨어나 소프트웨어의 오동작 또는 권한이 없는 악의적인 접근으로부터 시스템을 보호한다.
* 유지보수 : 시간이 지남에 따라 변화하는 요구사항을 반영할 수 있도록 하는 기능이다.

DBMS는 2가지로 나눌 수 있다. 관계형 데이터베이스를 기반으로 하는 시스템과 그러한 전통적인 관계형 데이터베이스를 채용하지 않은 이른바 NoSQL이다.

관계형 데이터베이스는 행과 열로 이루어진 테이블로 데이터를 저장하며 SQL은 IBM이 만들어낸 질의응답형 특수 언어로 DB로부터 data를 송수신하기 위해 고안되었다. 각 행은 데이터로 고유 키 값을 가지고 있고 각 열은 각 데이터가 지닐 수 있는 속성을 의미한다.

이러한 유형의 DB를 관리하는 언어가 SQL이며 이를 기반으로 다양한 RDBMS가 발달했다.

반대로 이러한 전통적인 테이블 형태를 지양하고 좀 더 개방적이고 유연한 형태로 DB를 구성하기 위해 고안된 유형의 DBMS를 NoSQL이라고 부른다.

NoSQL에는 다양한 유형이 있으며 벤 스코필드가 이러한 유형들 중 가장 정형화된 유형들에 대하여 성능 분석을 해놓은 표가 있다.

| **데이터 모델** | **성능** | **확장성** | **유연성** | **복잡성** | **기능** |
| --- | --- | --- | --- | --- | --- |
| 키-값 스토어 | 높음 | 높음 | 높음 | 없음 | 가변적 (없음) |
| 컬럼 지향 스토어 | 높음 | 높음 | 준수 | 낮음 | 최소 |
| 도큐먼트 지향 스토어 | 높음 | 가변적 (높음) | 높음 | 낮음 | 가변적 (낮음) |
| 그래프 데이터베이스 | 가변적 | 가변적 | 높음 | 높음 | [그래프 이론](https://ko.wikipedia.org/wiki/%EA%B7%B8%EB%9E%98%ED%94%84_%EC%9D%B4%EB%A1%A0) |
| 관계형 데이터베이스 | 가변적 | 가변적 | 낮음 | 준수 | [관계대수](https://ko.wikipedia.org/wiki/%EA%B4%80%EA%B3%84%EB%8C%80%EC%88%98) |

(2014.06.26)

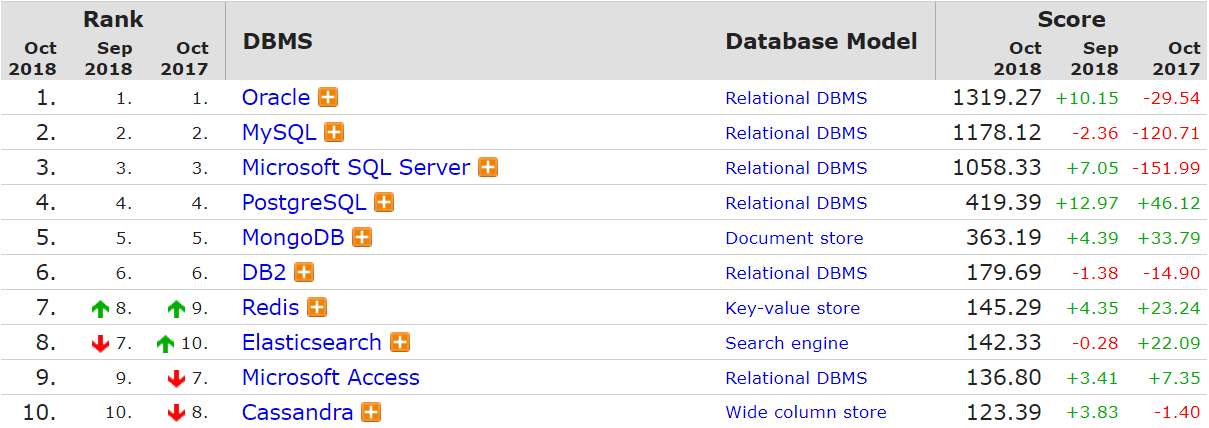
이런 관계형 데이터베이스 관리 시스템, RDBMS와 NoSQL의 차이는 다음과 같다.

* 고전적 형태, 즉 테이블로 이루어져 이에 SQL로 접근하는가?
* 여러 대의 서버에 분산 저장이 가능한가? (수평적 확장성의 용이)

사실 NoSQL의 종류에 따라 RDBMS와 차이가 있는 정도는 달라진다. NoSQL은 no sql이 아닌 not only sql이기 때문에 기존의 형태를 어느 정도 채용하는 시스템도 있고 이러한 형태를 고전적인 것으로 보고 지양하는 시스템도 있다.

**RDBMS**

RDBMS는 데이터베이스에 대한 최초의 접근법이기도 하며 오랜 세월이 지난 만큼 다양한 전문가들의 노력으로 정형화된 데이터에 대해서 빠른 처리 속도를 보여준다. 또한 오랜기간 사용되어 많은 사람들이 익숙해하며 인기도투표에서도 상위권을 차지한다. 다음은 DBMS ranking chart이다.



1위부터 4위까지 RDBMS가 차지하고 있으며 score 점수를 봐도 그 차이가 어마어마하다.

랭킹과 별개로 성능적으로 봤을 때도 정형화된 데이터의 처리, 관계형 데이터베이스의 구축과 관리에 RDBMS는 가장 알맞고 옳은 선택이다. ACID 규정 준수, 정형화된 데이터는 일관성, 안정성에 큰 이점을 준다.

다만, NoSQL과 비교하였을 때, 확장성에 있어서는 확실하게 떨어진다. 고도로 정형화되어 있다는 사실이 DB의 확장, 특히 수평적 확장성을 저해하며 이는 곧 매우 거대한 용량의 데이터, 트래픽을 처리하는 데 불리한 위치를 갖게 한다.

**NoSQL**

우선 NoSQL은 RDBMS를 제외한 나머지 DBMS의 종류들을 모두 일컫는 단어이며 2010년, NoREL이라는 단어로도 등장하였다. 비관계형 데이터베이스를 기반으로 하며 이로 인해 대부분 유연성, 확장성이 큰 장점으로 부각되며 특히 페타바이트급의 큰 용량도 처리가능하다는 점이 RDBMS에서 차별화된 특성 중 하나이다.

이러한 특징은 몇 년전부터 부각되던 빅데이터, SNS의 성장으로 인해 급격하게 늘어나는 데이터들의 처리를 용이하게 만들고 그러한 작업에서 NoSQL을 선택하게 되는 이유가 되었다. RDBMS의 경우 수직적 확장, 즉 cpu 등의 하드웨어의 상위 교체를 통해 성능이나 용량의 증가를 달성해야 하는데 이 경우 비용이 굉장히 많이 든다. 그에 비해 비교적 저성능의 컴퓨터를 여러 대 연결하여 분산 저장이 가능한 NoSQL의 경우 비교적 저렴하게 컴퓨터의 추가를 통해서 (수평적 확장) 성능 및 용량 증가가 가능하기 때문에 특히 빅데이터 분야에서 많이 쓰이는 것이다.

다음은 총 4가지 유형의 NoSQL에 대한 간략한 정리이다.

* 문서 유형 스토어

JSON, XML, BSON 문서에 데이터를 저장하는 비관계형 데이터베이스이다. 문서 구조는 전혀 강요되는 구조가 없으며 문서에는 어떤 원하는 데이터도 집어넣을 수 있다. 키-값 짝이 있지만 속성 메타데이터를 통해 더 쉽게 질의할 수 있다.

요약하자면 문서에 데이터를 저장하지만 그 문서는 정해진 구조가 없이 그저 키-값 짝을 가지는 속성들의 집합이 된다는 것이다. 대표적으로 MongoDB가 있다.

강점 : 데이터 처리에 있어서 유연성이 정말 높다. 반정형 및 비정형 데이터의 처리에 유용하다. 미리 데이터의 유형을 설정해 놓을 필요도 없다. 작성속도 역시 전반적으로 빠르며 NoSQL에서 중요한 강점으로 부각되는 수평적 확장이 특히 용이하다. 수평 확장에 필요한 샤딩(sharding)이 RDBMS에 비해 직관적이기 때문이다.

단점 : 유연성을 위해 ACID 준수를 포기했다. 이로 인해 데이터 처리의 안정성이 보장되지 않는다. 또한 한 문서 내에서만 질의가 가능하고 여러 문서에 걸친 질의는 불가능하다.

* **키-값 저장소**

각 값이 고유의 키를 가지는 형태이다. 보통 해시 테이블로 표현된다. 각 키는 DBMS에서 허용하는 어떤 것이라도 가능하다. 가령, Redis에서는 최대 512mb의 어떤 2진 시퀀스라도 키가 될 수 있다. 값은 블롭으로 저장되지만 미리 정의한 스키마는 필요없다. 값에는 거의 모든 형태가 들어갈 수 있다. 숫자, 문자열, 계수기, JSON, XML, HTML, PHP, 2진수, 이미지, 짧은 동영상, 목록, 심지에 객체에 압축된 또 다른 키-값 짝도 가능하다.

비정형 데이터 작업이 많거나 자주 접근하지만 자주 업데이트되지는 않는 데이터베이스에 적합하다.

블롭 (Blob) : Binary Large Object의 약자로 데이터를 저장하는 형태의 일종이다. 대부분 대용량의 데이터를 저장하기 위해 쓰이며 일반적인 데이터와는 쓰는 방법이 조금 다르다. 이미지, 영상 등 다양한 데이터가 담길 수 있다.

강점 : 역시 유연성이 뛰어나며 다양한 유형의 데이터를 처리할 때 유용하다. 처리 성능이 빠르며 이동성이 매우 뛰어나다. 다른 시스템으로 옮길 때 다시 코딩을 할 필요가 없다. 전체적인 운영비용 역시 낮다.

단점 : 값 질의가 불가능하다. 값이 블롭으로 저장되어 있기 때문에 블롭으로만 반환 가능하기 때문이다. 값의 일부만 수정하는 등의 작업이 어려우며 모든 객체를 키-값짝으로 모델화하는 것도 쉬운 일이 아니다.

* **와이드 칼럼 저장소 (wide-column storage)**

칼럼저장소, 확장 가능 기록 저장소라고도 부르며 키-값 저장소라고 보는 경우도 있지만 전통적 데이터베이스의 형태도 가지고 있다. 스키마 대신 키스페이스 개념을 사용하며 각 칼럼 군을 포괄한다. 테이블과 비슷하지만 더 유연하다. 각 칼럼 군은 열이 구별된 행을 가지고 있으며 각 행은 같은 수의 열을 가질 필요가 없다. 타임스탬프를 통해 가장 최근 버전의 데이터를 판별한다.

대표적으로 카산드라가 있으며 일반적인 데이터 트랜잭션보다는 빅데이터처럼 데이터를 보관하는 데에 특화되어 있다고 볼 수 있다.

강점 : RDBMS와 NoSQL의 강점을 골고루 가지고 있다. 일반적인 RDBMS에 비해 수평확장성이 높고 NoSQL에 비해 정형 데이터 및 반정형 데이터의 처리에 또한 용이하다.

단점 : 소규모 데이터베이스에 대해선 상대적으로 비용이 많이 든다. 한꺼번에 업데이트하기는 쉽지만 개별적으로 처리하는 것은 어렵기 때문이다. 또한 데이터 트랜잭션을 관리할 때 RDBMS에 비해 느리다.

* **검색 엔진**

말 그대로 검색엔진이다. 문서 기반으로 저장되며 데이터의 저장과 검색에 맞게 배열화 및 최적화 되어 있다.

대표적으로 엘라스틱서치가 있다.

강점 : 확장성이 역시 뛰어나고 유연한 스키마와 기록의 빠른 검색이 특징이다. 특히 검색에 특화되어 있는데 다양한 고급 검색 기능을 통해 사용자의 편의 향상이 가능하다.

단점 : 기본 데이터베이스로 쓰이기 보다는 중간 또는 보완 저장소로 더 많이 사용된다. 지속성이 낮고 보안이 부실하기 때문이다. 또한, 트랜잭션을 지원하지 않는다.

그런데, 최근..은 아니고 2011년도에 이러한 RDBMS의 장점과 NoSQL의 장점을 합쳐보겠다는 의지로 만들어진 이른 바 NewSQL이라는 용어가 생겨났다. 기본적으로 RDBMS이며 ACID를 준수하고 또한 NoSQL과 같은 확장성을 갖는다.

기존에 RDBMS와 NoSQL로만 나뉘어 데이터베이스를 구성하던 시기엔 안정성과 큰 용량을 모두 갖추기 위해선 비싼 가격을 주고 좋은 장비를 사거나 (single-node) 직접 미들웨어를 작성해서 하는 방법밖에 없었다. 그러나 NewSQL은 내부 구조적으로 다르지만 sql을 기본 인터페이스로 사용하고 관계형 데이터베이스를 지원한다는 공통점이 있다.

단점으로는 다른 기존의 NoSQL과 RDBMS에 비해 설계가 복잡하며 이는 소프트웨어 개발 비용 증가 등의 소요로 이어질 수 있다.

NewSQL은 3가지로 나뉘어질 수 있다.

**New architecture**

완전히 새로운 데이터베이스 플랫폼이다. 각 노드가 데이터의 subset을 가지는 이른바 shared-nothing nodes형태의 cluster의 operation을 위해 디자인되었다. 대표적으로는 구글 스패너, 알티베이스, VoltDB가 있다.

**SQL engines**

SQL언어를 위한 고도로 최적화된 storage engine이다. 기본적으로 SQL형식과 동일한 interface이지만 내장된 엔진보다 확장 (scaling)성능이 훨씬 좋다. 대표적으로는 MySQL Cluster, MariaDB cloumnstore 등이 있다.

**Transparent Sharding**

확장성에서 중요한 요소인 sharding을 자동으로 실행해주는 미들웨어레이어를 제공해준다. 대표적으로 ScaleBase가 있다.

**프로젝트의 데이터베이스 구조 분석**

현재 진행중인 프로젝트의 데이터베이스는 image 폴더 내 이미지 파일로 확인 가능하다. 사이트 자체적으로 내장될 데이터, 회원정보, articles & side\_info 이렇게 총 3가지로 나뉘어지며 이번 프로젝트의 주 목적 중 하나는 바로 스터디 자료들의 archiving이다.

앞으로 끊임없이 자료가 추가될 것이기에 확장성이 중요한 요소라고 생각된다. 그러나 현재 사용되는 방식은 학교에서 제공하는 **1개의** 서버를 이용하고 최근엔 이 서버의 이동 중에 kweb 자료가 날라가는 불상사가 생기기도 하였다. 아카이빙된 자료가 날라가지 않도록, 또한 데이터베이스의 용량이 커지는 것을 대비해 sharding하는 방안도 생각해야 할 것이다.

지금으로써는 1개의 서버를 이용해 홈페이지를 운영하지만 후에 자체적인 서버 운영이나 아마존, MS, 구글 등의 클라우드 서비스를 이용하게 되는 것을 상정하기로 한다.

**케이웹 커뮤니티의 활성화와 출석체크 기능의 구현, 축적형 자료들의 아카이빙 등 프로젝트의 특성을 종합적으로 보았을 때 NewSQL이 적합하다.**

특히 19년 3월부터 5G가 완전상용화되는 만큼 이에 대한 대비를 미리미리 해두어야 한다고 본다. 이에 따라 NewSQL이 가장 적합하다고 보았다.

이제는 모바일 페이지에서도 아카이빙된 자료를 다운받는 시기가 올 것이고 어쩌면 더 나아가 케이웹 페이지가 한국 웹 프로그래머들을 위한 소통의 장이 될 수도 있다는 큰 그림이 담겨져 있다.

이 중 우리가 이용할 수 있는 오픈 소스는 VoltDB community edition, Postgre-xc, NuoDB, Drizzle, H-store, MySQL cluster등이 있다.

아직까지 NewSQL은 상용화도 덜 되었고 다른 DBMS에 비해 역사도 짧다보니 검증되고 확실한 DBMS가 없는 것이 사실이다. 하지만 도전은 안정감이 아니라 쾌락과 진취를 바라는 사람들의 전유물이다. 새로운 것에 뛰어드는 것이 결국 이 프로젝트의 본연의 목적이기에 이를 잊지 않았다면 NewSQL은 좋은 선택이다.

위 기술한 NewSQL 중 가장 유명한 것은 MySQL cluster 인데 사실 해당 DBMS 자체가 어느 정도 쓰이고 있는 상태에서 MySQL cluster를 통해 sharding이 가능해져 NewSQL과 같은 기능이 가능해졌다라고 보는 것이 마땅할 것이다.

Open source, node.js 지원, 클러스터, ACID 등을 고려했을 때도 MySQL은 좋은 선택이다. 많은 사람들이 아직까지 MySQL을 쓰는 이유가 다른 데에 있진 않을 것이다. 다음은 다른 NewSQL 과의 비교페이지의 URL이다.

<https://db-engines.com/en/system/MariaDB%3BMySQL%3BNuoDB%3BVoltDB>

MariaDB, MySQL, NuoDB, VoltDB 순으로 비교가 되어 있다. 오픈소스가 공개되어 있는 기준으로 뽑은 것이다. 이외에도 Teradata, SAP Hana 등이 있지만 상용 라이선스이거나 현재 진행하는 프로젝트와 맞지 않다고 판단하였다.

**따라서, 이번 프로젝트에선 MySQL을 사용하되 추가적인 서버 확보와 MySQL cluster를 통해 sharding과 같은 확장성에 대해 공부하면서 진행하기로 한다.**