



# NoSQL

|      |     |
|------|-----|
| ≡ 태그 | DB  |
| ≡ 주차 | 3주차 |



## 목차

### 1. NoSQL

#### 1-1. 특징

- 1) 데이터간의 관계를 정의하지 않음
- 2) RDBMS에 비해 대용량 데이터 저장 가능
- 3) 분산형 구조로 설계
- 4) 고정되어 있지 않은 테이블 스키마를 가짐

#### 1-2. 장단점

- 1) 장점
- 2) 단점

#### 1-3. NoSQL의 종류

- 1) **Key-Value** DB
- 2) **Document** DB
- 3) **Wide-Column** DB
- 4) **Graph** DB
- 5) **In-memory** DB

#### 1-4. 다양한 상황에 따른 DB 솔루션 선택

## 1. NoSQL

- 비관계형 데이터베이스
- Not Only SQL 혹은 Non-Relational Operational SQL
- 기존의 RDBMS와 같은 관계형 데이터 모델을 지양
- 대량의 분산된 비정형 데이터를 저장하고 조회하는데 특화
- 스키마 없이 사용하거나 느슨한 스키마 제공

- 주로 빅데이터, 분산 시스템 환경에서의 대용량 데이터 처리에 적합

## 1-1. 특징

### 1) 데이터간의 관계를 정의하지 않음

- 데이터간 관계를 Foreign Key로 저장하는 RDBMS와 달리 Key-Value 형태로 저장되므로 Join 불가

### 2) RDBMS에 비해 대용량 데이터 저장 가능

### 3) 분산형 구조로 설계

#### 분산형 구조

- 물리적으로 떨어진 DB에 네트워크로 연결하여 단일 DB의 이미지를 보여주고 분산된 작업 처리를 수행하는 데이터 베이스
- 사용자 관점에서는 시스템이 분산되어있는지 인식하지 못하면서 자신만의 DB를 사용하는 것 처럼 사용 가능
  - 투명성을 제공해야 함

| 투명성       | 설명   |
|-----------|--|
| 분할 투명성    | - 고객은 하나의 논리적 릴레이션이 여러 단편으로 분할되어 각 단편의 사본이 여러 시스템에 저장되어있음을 인식할 필요가 없음                      |
| 위치 투명성    | - 고객이 사용하려는 데이터의 저장 장소를 명시할 필요가 없음<br>- 고객은 데이터가 어느 위치에 있더라도 동일한 명령을 사용하여 데이터에 접근할 수 있어야 함 |
| 지역 사상 투명성 | - 지역 DBMS와 물적 데이터베이스 사이의 사상이 보장됨에 따라 각 지역 시스템 이름과 무관한 이름이 사용가능함                            |
| 중복 투명성    | - 데이터베이스 객체가 여러 시스템에 중복되어 존재함에도 고객과는 무관하게 데이터의 일관성이 유지                                     |
| 장애 투명성    | - 데이터베이스가 분산되어 있는 각 지역의 시스템이나 통신망에 이상이 발생해도 데이터의 무결성은 보장                                   |

| 투명성    | 설명   |
|--------|--|
| 병행 투명성 | - 여러 고객의 응용 프로그램이 동시에 분산 데이터베이스에 대한 트랜잭션을 수행하는 경우에도 결과에는 이상 없음 |

- 설계 방식
  - 상향식 설계 방식
    - 지역 스키마 작성 후 전역 스키마를 작성해 분산 DB 구축
    - 지역별로 DB 구축 후 전역 스키마로 통합
  - 하향식 설계 방식
    - 전역 스키마 작성 후 해당 지역 사상 스키마를 작성해 분산 DB 구축
    - 기업 전체의 전사 데이터 모델을 수렴해 전역 스키마를 생성하고, 그 다음 각 지역별로 지역 스키마를 생성해 분산 DB 구축
- 장점
  - 데이터베이스 신뢰성과 가용성이 높음
  - 분산 데이터베이스가 병렬처리를 수행하기 때문에 빠른 응답이 가능
  - 분산 데이터베이스를 추가하여 시스템 용량 확장이 쉬움
- 단점
  - 데이터베이스가 여러 네트워크를 통해서 분리되어 있기 때문에 관리와 통제가 어려움
  - 보안관리가 어려움
  - 데이터 무결성 관리가 어려움
  - 데이터베이스 설계가 복잡



## 중앙 집중형 DB

한 대의 물리적 시스템에 DBMS를 설치하고, 여러 명의 사용자가 DBMS에 접속하여 DB를 사용하는 구조

## 4) 고정되어 있지 않은 테이블 스키마를 가짐

## 1-2. 장단점

### 1) 장점

- 비용 효율성
  - 신속하게 수평으로 확장하여 리소스의 효과적인 배정 및 비용 최소화 가능
- 유연성
  - 수평적 확장이 가능하고 유연한 데이터 모델을 따름
  - 따라서 가변적인 대량의 데이터 처리 가능
- 복제
  - 여러 서버에 걸쳐 데이터를 복제하고 저장해 데이터 신뢰성 제공
  - 다운타임 동안에도 액세스를 지원하고 서버가 오프라인 상태가 될 경우 데이터 손실로부터 보호 가능
- 속도
  - 빠르고 민첩한 저장과 처리 지원
  - 복잡한 최신 웹앱, 전자상거래 사이트 또는 모바일 앱에 적합

### 2) 단점

- 데이터 업데이트 중 장애 발생 시 데이터 손실 발생 가능
- 많은 인덱스를 사용하려면 충분한 메모리 필요. 인덱스 구조가 메모리에 저장
- 중복된 데이터 발생 가능
- 데이터 일관성이 항상 보장되지 않음
  - 최종적 일관성 지향



### 최종적 일관성 (Eventually Consistent)

한번 실행했던 기능은 반드시 처리되어야 함

## 1-3. NoSQL의 종류

### 1) Key-Value DB

- 가장 단순한 형태의 NoSQL DB
- 사용자 세션 정보를 캐시하고 저장하는 데 흔히 사용
- *한번에 여러 레코드를 가져와야 할 경우엔 적합치 않음*
- ex) Redis, Oracle NoSQL, VoldeMorte

### 2) Document DB

- 데이터를 문서로 저장
- 준정형 데이터 관리에 유용
- JSON, XML, BSON 형식으로 저장
- 문서간 스키마가 일치하지 않아도 되므로 유연성이 큼
- *복잡한 트랜잭션의 경우 문제가 될 수 있음*
- *데이터 손상 유발*
- ex) MongoDB, CouchDB, Riak, Azure Cosmos DB

### 3) Wild-Column DB

- 정보를 칼럼에 저장
- 관련 없는 데이터에 추가 메모리를 할당하지 않고 특정 칼럼에만 액세스 가능
- Key-Value와 Document DB의 단점을 해소할 수 있지만, 관리가 어려워 비권장
- ex) Apache Hbase, Apache Cassandra, GoogleBigTable, Vertica

### 4) Graph DB

- 지식 그래프 데이터 보관
- 노드, 간선, 속성으로 저장

- 노드: Object/Client
- 간선: 노드간의 관계
- 그래프 내 요소들 간의 연결 네트워크를 저장하고 관리되는데 사용
- ex) Sones, AllegroGraph, neo4j, BlazeGraph, OrientDB

## 5) In-memory DB

- 데이터가 디스크가 아닌 메인 메모리에 보관
- 통상적인 디스크 기반 DB보다 빠르게 데이터에 액세스 가능
- ex) IBM solidDB


## 1-4. 다양한 상황에 따른 DB 솔루션 선택

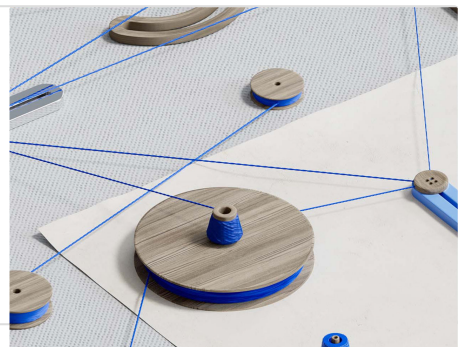
- SQL
  - 데이터가 자주 수정되는 경우
  - 명확한 스키마가 중요한 경우
- NoSQL
  - 스키마를 정의할 수 없거나 수정될 수 있을 경우
  - 읽기(Read)가 잦고, 데이터가 자주 수정되지 않을 경우
  - 데이터의 양이 방대할 경우

### 출처

#### NoSQL 데이터베이스란? | IBM


"not only SQL", "non-SQL"으로도 불리는 NoSQL은 관계형 데이터베이스의 전통적인 구조 밖에서 데이터 저장 및 쿼리를 가능하게 하는 데이터베이스

 <https://www.ibm.com/kr-ko/topics/nosql-databases>



## SQL vs NoSQL

정의 SQL(Structured Query Language) 관계형  
데이터베이스 관리 시스템(relational database  
management system, RDBMS)에서 데이터를 저

 <https://minnnji23.tistory.com/31>

# SQL vs NoSQL