

P R E S E N T A T I O N

# 15주차 주제 NoSQL

## NoSQL & Redis

by mun

# NoSQL 이란 ?

대량의 비정형 및 반정형 데이터를 처리하고 저장하도록 설계된  
데이터베이스 관리 시스템(DBMS)의 일종

**not only SQL**

원래 "non-SQL" 또는 "non-relational" 데이터베이스를 지칭했지만,  
이후 NoSQL 데이터베이스가 다양한 데이터베이스 아키텍처 및 데이터 모델을  
포함하도록 확장됨에 따라 "not only SQL"라는 의미로 발전했다.

# NoSQL 이란 ?

비관계형 데이터베이스 유형을 가리키며 관계형 테이블과는 다른 형식으로 데이터를 저장

## 기존의 관계형 데이터베이스

데이터를 저장하기 위해  
미리 정의된 스키마가 있는 테이블을 사용

# NoSQL 이란 ?

비관계형 데이터베이스 유형을 가리키며 관계형 테이블과는 다른 형식으로 데이터를 저장

## 기존의 관계형 데이터베이스

데이터를 저장하기 위해  
미리 정의된 스키마가 있는 테이블을 사용

데이터 구조의 변화에 적응할 수 있고  
증가하는 양의 데이터를 처리하기 위해  
수평으로 확장할 수 있는 유연한 데이터 모델 사용

# 주요 유형

데이터 모델에 따라 다양한 유형으로 제공된다

## Key-Value

가장 유연한 NoSQL 데이터베이스 유형  
해시 테이블을 사용하여 키와 값의 쌍을 저장

## Document

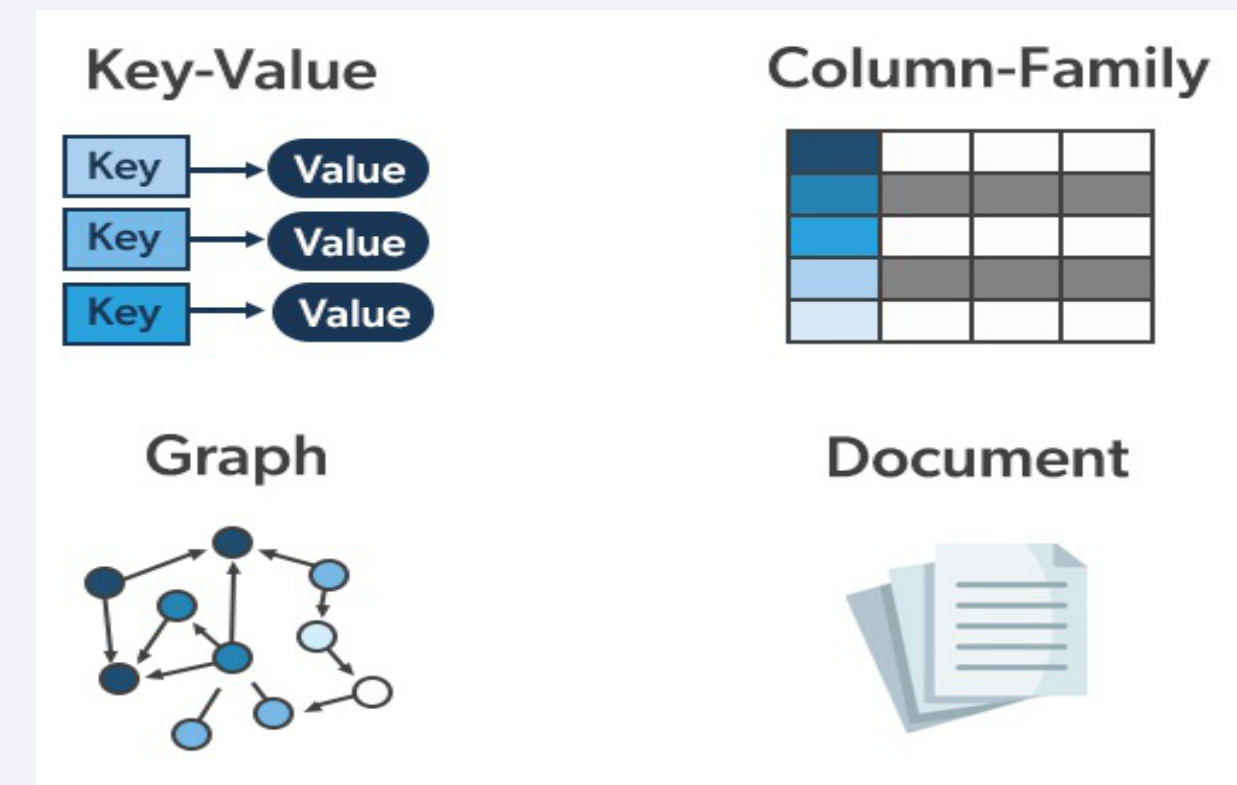
데이터를 문서로 저장  
반구조적 데이터의 저장, 검색, 관리에 사용

## Graph

데이터를 노드 및 노드 간 연결을 보여주는 관계로 구성  
상호 연결된 데이터를 표현하고 복잡한 관계를 간단하게 살펴볼 수 있도록 지원  
소셜 네트워크, 예약 시스템, 사기 감지 등에 적용

## Wide-Column

테이블, 행, 열 형식으로 데이터를 저장 및 관리  
소셜 네트워킹 웹사이트 및 실시간 데이터 분석 등 다양한 사용 사례에 사용



<https://www.geeksforgeeks.org/types-of-nosql-databases/>

# 주요 유형

데이터 모델에 따라 다양한 유형으로 제공된다

## Key-Value

가장 유연한 NoSQL 데이터베이스 유형  
해시 테이블을 사용하여 키와 값의 쌍을 저장

## Document

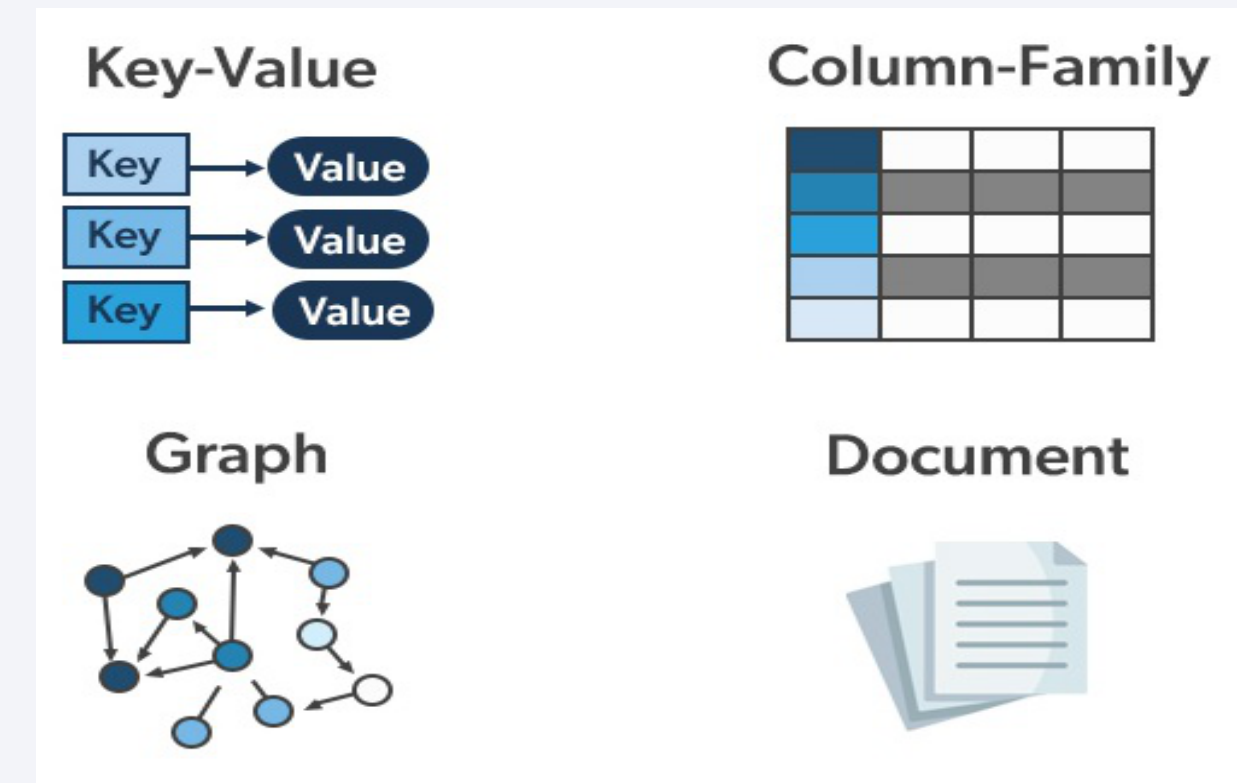
데이터를 문서로 저장  
반구조적 데이터의 저장, 검색, 관리에 사용

## Graph

데이터를 노드 및 노드 간 연결을 보여주는 관계로 구성  
상호 연결된 데이터를 표현하고 복잡한 관계를 간단하게 살펴볼 수 있도록 지원  
소셜 네트워크, 예약 시스템, 사기 감지 등에 적용

## Wide-Column

테이블, 행, 열 형식으로 데이터를 저장 및 관리  
소셜 네트워킹 웹사이트 및 실시간 데이터 분석 등 다양한 사용 사례에 사용



<https://www.geeksforgeeks.org/types-of-nosql-databases/>

# 주요 유형 **Key-Value**

데이터를 키-값 쌍으로 저장하며, 간단하고 빠른 읽기/쓰기 작업에 최적화

데이터베이스의 각 요소에 할당된 고유한 키를 사용하여 데이터를 검색할 수 있다.

키의 값을 사용하거나 키 범위별로 간단한 조회를 수행에는 적절하지만  
다른 키/값 테이블 사이에서 데이터를 쿼리해야 하는 시스템에는 덜 적절하다.

**주요 기능**

- 단순함
- 확장성
- 속도

# Redis 레디스



## Remote Dictionary Server

"key-value" 구조의 비정형 데이터를 저장하고 관리하기 위한  
오픈 소스 기반의 비관계형 데이터베이스 관리 시스템(DBMS)

캐시, 벡터 데이터베이스, 문서 데이터베이스,  
스트리밍 엔진 및 메시지 브로커로 사용하는  
오픈 소스 인 메모리 데이터 저장소



# 주요 특징

## In-memory DB

데이터를 디스크에 저장하지 않고 메모리 영역에 저장해 속도가 훨씬 빠르다.  
영속성을 위한 DB는 아니지만 원하는 수준의 영속성 구성은 가능하다.

## Key-value store

NoSQL의 데이터 모델의 하나인 Key-value 를 사용한다.  
분산환경에서의 확장성과 사용자 편의성, 빠른 속도를 가진다.

## 다양한 자료구조

String, Hash, List 등 다양한 자료구조를 지원한다.  
이러한 자료구조를 활용해 Middleware로서의 역할을 수행할 수도 있다.  
예) Sorted Sets을 이용한 리더보드 구현, Bitmaps를 이용한 방문자 수 count

# 자료구조 활용 예 Sorted Set

## 레디스 템플릿

```
@Autowired
public RankingService(StringRedisTemplate stringRedisTemplate){
    this.redisTemplate = stringRedisTemplate;
}
```

## 유저 점수 저장

```
// 정보 저장
2 usages  👤 시원 *
public boolean setUserScore(String userId, int score){
    ZSetOperations<String, String> zSetOps = redisTemplate.opsForZSet();
    //leaderBoard 라는 key 에 userId, score 세팅한다.
    zSetOps.add(LEADERBOARD_KEY, userId, score);

    return true;
}
```

# 자료구조 활용 예 Sorted Set

## 특정 유저 랭킹 조회

```
public Long getUserRanking(String userId){
    ZSetOperations<String, String> zSetOps = redisTemplate.opsForZSet();
    //LEADERBOARD_KEY 로 들어가 있는 set 에서 userId 라는 value 의 순위를 받는다.
    //reverseRank: 내림차순
    Long rank = zSetOps.reverseRank(LEADERBOARD_KEY, userId);

    return rank;
}
```

## 범위 기반 랭킹 조회

```
public List<String> getTopRank(int limit){ Complexity is 4 Everything is cool!
    ZSetOperations<String, String> zSetOps = redisTemplate.opsForZSet();
    //범위기반 조회의 결과는 Set 으로, reverseRange: 내림차순
    Set<String> rangeSet = zSetOps.reverseRange(LEADERBOARD_KEY, start: 0, end: limit-1);

    assert rangeSet != null;
    return new ArrayList<>(rangeSet);
}
```

# 자료구조 활용 예 Sorted Set

특정 유저  
랭킹 조회

```
public Long getUserRanking(String userId){
    ZSetOperations<String, String> zSetOps = redisTemplate.opsForZSet();
    //LEADERBOARD_KEY 로 들어가 있는 set 에서 userId 라는 value 의 순위를 받는다.
    //reverseRank: 내림차순
    Long rank = zSetOps.reverseRank(LEADERBOARD_KEY, userId);

    return rank;
}
```

범위 기반  
랭킹 조회

```
public List<String> getTopRank(int limit){
    ZSetOperations<String, String> zSetOps = redisTemplate.opsForZSet();
    //범위기반 조회의 결과는 Set 으로, reverseRange: 내림차순
    Set<String> rangeSet = zSetOps.reverseRange(LEADERBOARD_KEY, start: 0, end: limit-1);

    assert rangeSet != null;
    return new ArrayList<>(rangeSet);
}
```

다양한 자료구조를 활용해  
적절한 기능 구현 가능

# SQL vs NoSQL

적합한 환경은?

## SQL

사전에 파악 가능한 논리적이고  
뚜렷한 요구 사항이 있는 관계형 데이터 처리

앱과 데이터베이스 간에 동기화된 상태로  
유지해야 하는 스키마

관계형 구조를 위해 빌드된 레거시 시스템

복잡한 쿼리 또는 다중 행 트랜잭션이 필요한 앱

## NoSQL

대규모 데이터, 확정되지 않은 데이터  
또는 빠르게 변화하는 데이터 처리

스키마 종립적 데이터 또는 앱에 의해 결정되는 스키마

강력한 일관성보다 성능과 가용성이 더 중요한 앱

세계 각지의 사용자에게 서비스를 제공하는  
상시 가동형 앱

**관계형과 비관계형 DB를 이해하고  
환경에 따라 적합한 DB를 선택해보아요**

# 감사합니다



참고 :

<https://azure.microsoft.com/ko-kr/resources/cloud-computing-dictionary/what-is-nosql-database>,  
<https://www.geeksforgeeks.org/types-of-nosql-databases/> , <https://www.mongodb.com/ko-kr/nosql-explained>, <https://www.ibm.com/kr-ko/topics/nosql-databases>