Redis(레디스)

지현정

정의 및 특징

시스템 메모리를 사용하는 키-값 데이터 스토어

인메모리 상태에서 데이터를 처리함으로써 흔히 사용하는 관계형 데이터베이스(RDB), 몽고 DB와 같은 문서형(Document) 데이터베이스보다도 빠르고 가볍게 동작함

레디스를 사용하는 방법 (HOW)

1. 로컬 환경에서 레디스를 호출

예) AWS EC2

인스턴스에 레디스를 설치해 인스턴스 메모리를 사용해 레디스를 사용하는 방법

인스턴스의 메모리에 여유가 있다면 비용, 사용성 측면에서 뛰어남

2. 클라우드 서비스를 사용해 외부 자원을 사용

레디스랩(https://redislabs.com/) 등의 서드파티 서비스를 이용하여 통신하는 웹 서버가 많아도 하나의 프레임워크 바인딩을 사용하는 방법

여러 웹 서버들의 공유 메모리 역할을 감당할 수 있음

레디스를 어디에 사용하는지 (WHERE)

대표적인 예시: 유튜브

- 조회수(카운트 형태) 데이터

운영 중인 웹 서버에서 키-값 형태의 데이터 타입을 처리하고

IO가 빈번히 발생하여 다른 저장 방식을 사용하면 효율이 떨어지는 경우에 사용

1. 캐싱 처리
2. 사용자 세션관리: 사용자의 세션을 유지하고 불러오고 여러 활동을 추적하는게 매우 효과적으로 사용
3. 메세지 큐잉: 매우 빠르게 동작함을 이용
4. API 캐싱: 라우터로 들어온 요청에 대해 요청 값을 캐싱하면 동일 요청에 대해 캐싱된 데이터를 리턴하는 방식 (https://fors.tistory.com/582)

레디스의 데이터 구조(1)

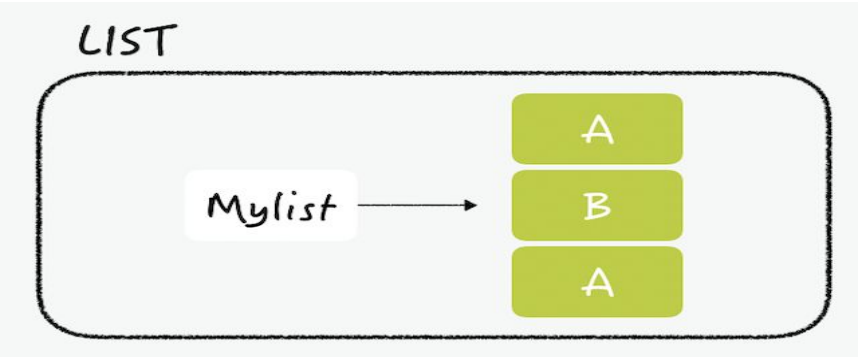
RDBMS가 아니므로 VARCHAR, INT, DATETIME 등을 지원하지 않음

1. 문자열(string): 모든 종류의 문자열 저장 가능

레디스의 키가 문자열이므로 문자열을 다른 문자열에 매핑하는 구조

- string을 정수로 파싱, 이를 atomic하게 증감하는 커맨드
- 키를 새 값으로 변경하고 이전 값을 반환하는 커맨드
- 키가 이미 존재하거나 존재하지 않을때만 데이터를 저장하게 하는 옵션

레디스의 데이터 구조(2)

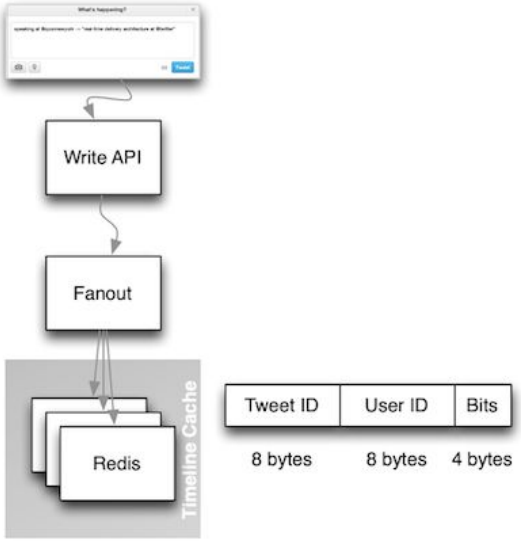


2. List: 일반적인 linked list의 특징

list 내 수백만개의 아이템이 있더라도 head, tail에 값을 추가할 때 동일한 시간 소요

- Pub-Sub(생산자-소비자) 패턴
  - 프로세스간의 통신 방법에서 생산자가 아이템을 만들어 list에 넣으면 소비자가 꺼내와서 액션을 수행하는 식
  - 트위터에서 각 유저의 타임라인에 트윗을 보여주기 위해 레디스 list 사용

레디스의 데이터 구조(2)



using redis

- native list structure
- RPUSHX to only add to cached timelines

RPUSHX는 키가 이미 존재할 경우에만 데이터를 저장,

이를 이용하여 이미 캐시된 타임라인에만 데이터를 추가할 수 있음

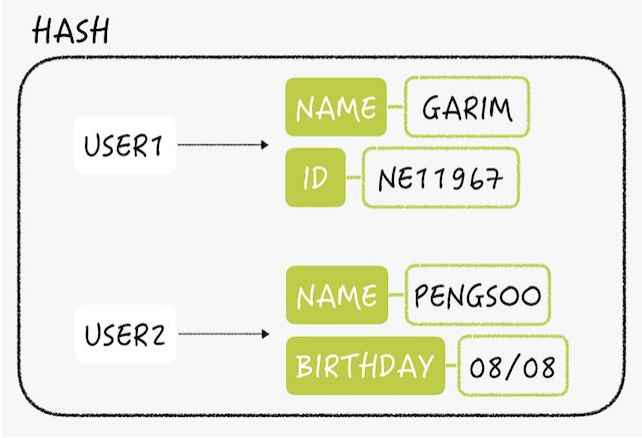
일시적으로 list를 blocking하는 기능도 유용하게 사용될 수 있음

Pub-Sub 상황에서 list가 비어있을 때 pop을 시도하면 대개 NULL을 반환

=> 소비자는 일정시간을 기다린 후 다시 pop을 시도(= polling)

BRPOP을 사용하면 새로운 아이템이 리스트에 추가될 때에만 응답하므로 불필요한 polling 프로세스를 줄일 수 있음

레디스의 데이터구조(3)



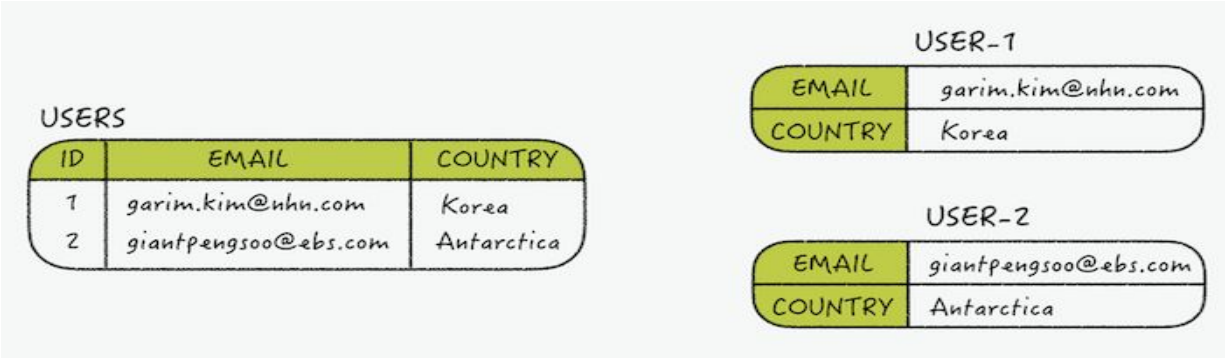
3. HASH" 필드-값 쌍을 이용한 일반적인 해쉬

키에 대한 필드의 갯수는 제한이 없어 여러방법으로 사용

필드와 값으로 구성된다는 면에서 RDB 테이블과 비슷

hash key = table PK, hash field = column, hash value = value

key가 PK와 같은 역할을 하기 때문에 key 하나는 table의 한 row와 같음



레디스의 데이터구조(4)

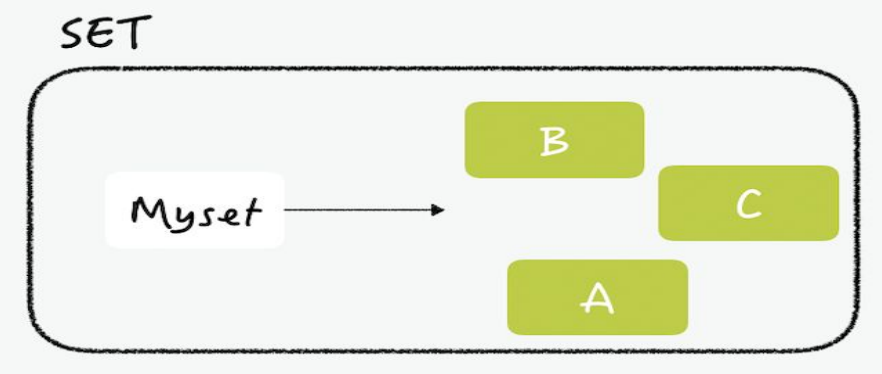
4. SET: 정렬되지 않은 문자열의 모음

교,합,차집합 연산은 레디스에서 수행 가능하여 객체간의 관계를 표현할 때 좋음

- 예시: 태그 기능

ID가 1000인 프로젝트에 1,2,5,77번 태그 ID 연결

key값을 project:1000:tags로 지정하고 여기에 태그 1,2,5,77번 태그를 add



레디스 스키마

데이터를 정규화, 데이터의 로우에 대해 일관된 레퍼런스를 가지게 해 줄 수 있게 해주는 용도로 존재함

-> 데이터 입출력에 도움을 줌

ex) 사용자 이메일, 닉네임, 최근 로그인 시간 저장

user:userId:email:문자열

user:userId:nickname:문자열

user:userId:lastLogin:문자열

데이터를 콜론으로 구분하여 접근 가능함

해당 스키마를 활용하여 사용자 아이디만 알아도 연결된 데이터 이메일, 닉네임. 최근로그인을 알 수 있음

레디스의 키

레디스의 스키마는 RDS의 로우의 숫자와 비슷하게 동작하지 않음

즉, 성능에 영향을 주지 않음

RDS의 경우 탐색해야 하는 row가 많아짐에 따라 검색 속도가 느려지다 보니 INDEX, PK, FK 등 설정하여 사용

반면 레디스는 O(1)의 수행시간을 가지며 키가 많건 적건, 스키마가 많건 적건 동일한 시간이 사용됨

레디스의 유의사항

레디스는 한 번 생성한 키를 선택적으로 삭제하기 어려움

레디스의 키는 삭제가 아닌 보관되며 서버를 종료했다가 다시 켜도 상태가 유지됨

삭제 방법

1. 일괄 삭제: FLUSHDB 명령어를 통해 모든 키를 파괴

복구 불가능, 개발 수준에서 사용

2. 일정 시간 이후 삭제

각각의 키를 저장할 때 SET에 저장하여 특정 시간이나 조건에 따라 삭제

3. 기간 만료 후 삭제: 가장 많이 사용

키를 추적할 필요 없이 쉽게 관리 가능

- 키-값을 SET 커맨드로 저장하여 EXPIRE 커맨드로 기간만료 시간을 정하는 방법 사용
- 또는 SETEX (SET+EXPIRE) 커맨드 사용