개발자를 위한 Redis

3장. 레디스 기본 개념

 # 레디스의 자료 구조

1. String
최대 512MB의 문자열 데이터를 저장할 수 있으며 이진 데이터를 포함하는 모든 종류의 문자열이 binary-safe하게 처리되기 때문에 JPEG 이미지와 같은 바이트 값, HTTP 응답값 등의 다양한 데이터를 저장하는 것도 가능하다.

NX – 지정한 키가 없을 때에만 새로운 키를 저장
XX – 지정한 키가 있을 때에만 새로운 값으로 저장
INCR – 저장된 데이터를 1씩 증가할 수 있으며 증가된 값이 반환
INCRBY – 입력한 값만큼 데이터를 증가
DECR - 저장된 데이터를 1씩 감소할 수 있으며 감소된 값이 반환
DECRBY – 입력한 값만큼 데이터를 감소

증가,감소는 원자적으로 처리하는데 같은 키에 접근하는 여러 클라이언트가 경쟁 상태를 발생시킬 일이 없음을 의미한다.
커맨드를 수행하는 타이밍이나 순서에 따라 이미 실행한 커맨드가 무시되거나 같은 커맨드가 중복 처리돼 수행 결과가 달라지는 일이 발생하지 않는다.

MSET, MGET 커맨드를 이용하면 한번에 여러 키를 조작할 수 있다. 성능이 중요한 대규모 시스템에서는 밀리세컨드 단위의 속도 향상도 서비스 전체 속도 향상으로 이어진다.

2. List
최대 42억여 개의 아이템을 저장할 수 있는 순서를 가지는 문자열 목록이다.

LPUSH - head에 데이터를 추가
RPUSH - tail에 데이터를 추가
LRANGE - 시작(0)과 끝(-1) 아이템의 인덱스를 각각 인수로 받아 출력한다
LPOP - 첫번째 아이템을 반환 하는 동시에 list에서 삭제
LTRIM - 시작과 끝 아이템의 인덱스를 인자로 전달 받아 지정한 범위에 속하지 않는 아이템 모두 삭제
LINSERT - 원하는 데이터의 앞이나 뒤에 데이터를 추가할 수 있다. BEFORE , AFTER ex) LINSERT mylist before B E
LSET - 지정한 인덱스의 데이터를 신규 입력, 없으면 error
LINDEX – 원하는 인덱스의 데이터 확인

LPUSH 와 LTRIM 으로 고정된 길이의 큐를 쉽게 유지 가능하다.

Ex)1,000개의 로그 데이터를 보관하고 할때
LPUSH logdata <data>
LTRIM logdata 0 999

3. Hash
필드-값 쌍을 가진 아이템의 집합이다. 하나의 hash 자료 구조 내에서 아이템은 필드-값 쌍으로 저장된다.

HSET - 데이터 저장 ex) hset product:123 Name “Happy Hacking”
HGET - 데이터 가져오기 키와 아이템 필드를 함께 입력해야함 ex) hget product:123 TypeID
HMGET - 하나의 hash내에서 다양한 필드의 값을 가져옴 ex) hmget product:234 Name TypeID
HGETALL - 모든필드 가져옴 ex) hgetall product:234

4. Set
정렬되지 않은 문자열의 모음

SADD - 아이템 저장, 여러 개의 아이템을 저장하는 것도 가능, 실제 아이템 수를 반환
SMEMBERS - 전체 아이템 출력, 저장한 순서와 관계없이 랜덤 출력
SREM - 원하는 데이터 삭제
SPOP - 내부의 아이템중 랜덤으로 반환하는 동시에 그 아이템 삭제
SUNION - 합집합
SINTER - 교집합
SDIFF - 차집합

5. Sorted Set
스코어 값에 따라 정렬되는 고유한 문자열의 집합이다. 모든 아이템은 스코어-값 쌍을 가지며, 저장될 때부터 스코어 값으로 정렬돼 저장된다.
데이터는 중복 없이 유일하게 저장되므로 set과 유사하고, 각 아이템은 스코어라는 데이터에 연결돼 있어 이 점에서 hash와 유사하다고 생각할 수 있다.
또한 모든 아이템은 스코어 순으로 정렬돼 있어 list처럼 인덱스를 이용해 각 아이템에 접근할 수 있다.

ZADD - 아이템 저장할 수 있으며 스코어-값 쌍으로 입력해야 한다. 한번에 여러 아이템 입력 가능하며 저장되는 동시에 스코어 값으로 정렬된다.
Ex) ZADD score:220817 100 user:B 150 user:C
만약 저장하고자 하는 데이터가 이미 속해 있으면 스코어만 업데이트 되며, 업데이트된 스코어에 의해 아이템 재정렬된다.
옵션
XX - 존재할 때에만 스코어 업데이트
NX - 존재하지 않을 때에만 신규 삽입, 기존 아이템의 스코어를 업데이트 하지 않음.
LT - 업데이트하고자 하는 스코어가 기존 아이템의 스코어보다 작을 때에만 업데이트한다. 존재하지 않을 때에는 신규 삽입
GT - 업데이트하고자 하는 스코어가 기존 아이템의 스코어보다 클 때에만 업데이트한다. 존재하지 않을 때에는 신규 삽입

ZRANGE - start와 stop이라는 범위를 입력하여 데이터 조회 ex) ZRANGE key start stop [BYSCORE | BYLEX] [REV] [LIMIT offset count]
Withscore - 데이터와 함께 스코어 값이 출력
Rev - 데이터 역순

Byscore - 스코어를 이용해 데이터 조회 ex) ZRANGE score:220817 (100 150 byscore withscores
(문자 추가하면 not
-inf +inf - 최소값과 최대값 표현 ex) ZRANGE score:220817 150 +inf byscore withscores
Bylex – 사전순 입력한 문자열을 포함하려면 (을 포함하지 않으면 [문자 사용, -는 가장 처음 +는 가장 마지막으로 대체하므로 - + 하면 모든 데이터 조회

6. 비트맵
독자적인 자료 구조는 아니며 string 자료 구조에 bit 연산을 수행할 수 있도록 확장한 형태
가장 큰 장점은 저장 공간을 획기적으로 줄일 수 있다는 것이다. 예를 들어 각각의 유저가 정수 형태의 ID로 구분되고, 전체 유저가 40억이 넘는다고 해도 각 유저에 대한 y/n 에 데이터는 512MB 안에 충분히 저장 가능

SETBIT - 비트 저장
GETBIT - 비트 조회
BITFIELD - 한번에 여러 비트를 SET
BITCOUNT - 1로 설정된 비트의 개수를 카운팅

7. Hyperloglog
집합의 원소 개수인 카디널리티를 추정할 수 있는 자료 구조다. 대량 데이터에서 중복되지 않는 고유한 값을 집계할 때 유용하게 사용할 수 있다.
일반적으로 set과 같은 데이터 구조에서는 중복을 피하기 위해 저장된 데이터를 모두 기억하므로 데이터가 많아지면 많은 메모리를 사용한다.
Hyperloglog는 입력되는 데이터 그 자체를 저장하지 않고 자체적인 방법으로 데이터를 변경해 처리한다. 따라서 저장되는 데이터 개수에 구애받지 않고 계속 일정한 메모리를 유지할 수 있으며, 중복되지 않는 유일한 원소의 개수를 계산할 수 있다.

PFADD - 아이템 저장
PFCOUNT - 저장된 아이템 개수, 즉 카디널리티를 추정할 수 있다.

8. Geospatial
경도, 위도 데이터 쌍의 집합으로 간편하게 지리 데이터를 저장할 수 있다.
내부적으로 데이터는 sorted set으로 저장되며, 키는 중복돼 저장되지 않는다.

GEOADD <key> - 경도 위도 member 순서로 저장되며 sorted set과 마찬가지로 XX 옵션 사용하면 이미 있는 경우에만, NX 옵션 사용하면 없는 경우에만 데이터 저장
GEOPOS – 저장된 위치 데이터를 조회
GEODIST – 두 아이템 사이의 거리를 반환
GEOSEARCH – 특정 위치를 기준으로 원하는 거리 내에 있는 아이템을 검색 할 수 있다. Byradius 옵션을 사용하면 반경 거리를 기준으로, bybox 옵션을 사용하면 직사각형 거리를 기준으로 데이터 조회

9. Stream
레디스를 메시지 브로커로서 사용할 수 있게 하는 자료 구조이다. 전체적인 구조는 카프카에서 영향을 받아 만들어졌으며, 카프카에서처럼 소비자 그룹 개념을 도입해 데이터를 분산 처리할 수 있는 시스템이다.

 # 레디스에서 키를 관리하는 법

키의 생성과 삭제 세 가지 공통적인 규칙
1. 키가 존재하지 않을 때 아이템을 넣으면 아이템을 삽입하기 전에 빈 자료구조를 생성한다.
2. 모든 아이템을 삭제하면 키도 자동으로 삭제된다.(stream은 예외)
3. 키가 없는 상태에서 키 삭제, 아이템 삭제, 자료 구조 크기 조회 같은 읽기 전용 커맨드를 수행하면 에러를 반환하는 대신 키가 있으나 아이템이 없는 것처럼 동작한다.

키와 관련된 커맨드
EXISTS - 키가 존재하는지 확인

KEYS - 레디스에 저장된 모든 키를 조회하는 커맨드, 패턴은 글롭 패턴 스타일로 동작한다. (위험한 커맨드)

SCAN - keys를 대체해 키를 조회할때 사용. Keys는 한번에 모든 키를 반환하므로 잘못 사용시 문제가 발생할 수 있지만, SCAN은 커서를 기반으로 특정 범위의 키만 조회할 수 있기 때문에 비교적 안전하게 사용할 수 있다. Match 옵션을 사용하면 keys에서처럼 입력한 패턴에 맞는 키 값을 조회한다.
SSCAN, HSCAN, ZSCAN - 각각 set, hash, sorted set에서 아이템을 조회하기 위해 사용되는 SMEMBERS, HGETALL, ZRANGE WITHSCORE를 대체해서 서버에 최대한 영향을 끼치지 않고 반복해서 호출할 수 있음

SORT - list, set, sorted set에서만 사용할 수 있는 커맨드로, 키 내부의 아이템을 정렬해 반환한다. Limit 옵션으로 일부 데이터만 조회할 수 있으며 ASC/DESC 옵션으로 정렬 ALPHA로 문자열일 경우 사전순으로 정렬해 조회할 수 있다. By와 get 옵션을 이용하면 정렬한 결과를 이용해 다른 키에 접근해서 데이터를 조회할 수 있다.

RENAME/RENAMENX - 키의 이름을 변경하는 커맨드, 하지만 RENAMENX 커맨드는 오직 변경할 키가 존재하지 않을 때에만 동작한다.
COPY - 지정된 키를 destination 키에 복사한다. 지정한 키가 이미 있으면 에러를 반환하는데 replace 옵션을 사용하면 삭제한 뒤 값을 복사하기 때문에 에러나지 않는다.
OBJECT - 키에 대한 상세 정보를 반환한다.
FLUSHALL - 레디스에 저장된 모든키를 삭제한다.
DEL - 키와 키에 저장된 모든 아이템을 삭제한다.
UNLINK - DEL과 비슷하게 키와 데이터를 삭제하지만 백그라운드에서 다른 스레드에 의해 처리 되므로 저장된 아이템이 많은 경우 UNLINK를 사용해 데이터 삭제하는 것이 좋다.
EXPIRE - 키가 만료될 시간을 초단위로 정의할 수 있다.
EXPIREAT - 키가 특정 유닉스 타임스탬프에 만료될 수 있도록 직접 지정한다.
EXPIRETIME - 키가 삭제되는 유닉스 타임스탬프를 초 단위로 반환한다. 키가 존재하지만 만료 시간이 설정돼 있지 않은 경우 -1, 키가 없을때에는 -2를 반환한다.
TTL - 키가 몇초뒤에 만료되는지 반환한다. 키가 존재하지만 만료 시간이 설정돼 있지 않으면 -1, 키가 없으면 -2를 반환한다.
앞에 P를 붙이면 밀리초 단위로 계산된다.