Data Structure

Tutorail

- **Keys**
  - 키의 길이는 너무 길지 않는 것이 좋다. → 메모리 측면에서 손해
  - 키의 길이는 너무 짧은 않는 것이 좋다. → 읽기 쉬운 값으로 하자
  - ex) `object-type:id`, `user:1000`
- **Altering and querying the key space**
  - 키가 존재 하지 않아도 오류를 발생하지 않고 결과를 리턴해준다.
  - Exists: 키의 존재 확인 → 1 OR 0 리턴
  - Del: 키 제거 → 삭제 시 1, 해당 키 없을 경우 0 리턴
- **Key expiration**
  - 키 단위로 만료 시간(TTL) 설정 가능
  - 정확도는 1 millisecond
  - redis 서버가 중지되어 있더라도 시간은 계속 카운팅 된다.

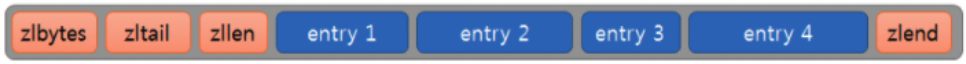
String

- `SET` : 값을 저장한다.
- `SETNX` : 이미 존재하는 키라면 저장하지 않는다.
  - `setnx mykey newValue` → 존재 시 0 리턴
  - `set myket newVlaue nx` → (nil)
- `INCR`, `INCRBY` 를 사용하여 정수를 증가 시킬 수 있다.
  - 소수점 더할 경우 → `(error) ERR value is not an integer or out of range`
  - `INCRBYFLOAT` 사용 가능

Lists

- Linked List로 구현되어 있다.

- 버전 3.2 이전에는 데이터량이 적을 때 Zip List 많을 때 Linked List로 구현
  - Zip List
    - 리스트 구조



    - 엔트리 구조

  - 버전 3.2 이후에는 Quick List 구현으로 통일
    - Quick List
      - 각 노드를 ziplist로 구성하고 적정 크기로 유지하면서 포인터로 노드를 연결하는 구조
      - Zip List의 리스트 중간 노드 추가/삽입시 발생하는 메모리 재할당, 복사/이동 문제를 해결
        Linked List의 오버헤드 문제 해결

- Blocking commands: `BLPOP`, `BRPOP`, `BLMOVE`
  - list가 비어있을 경우, 새로운 항목이 추가되거나, timeout 까지 기다린다.

Sets

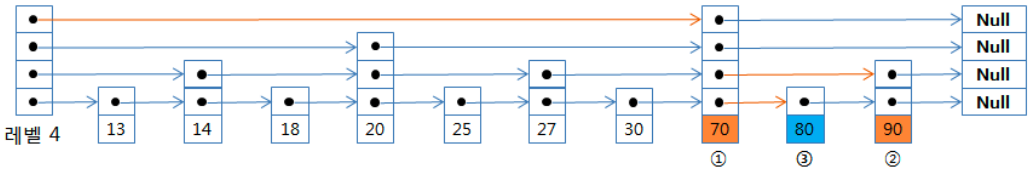
- SADD, SREM, SISMEMBERS, SINTER, SCARD 등 다양한 명령어 제공
  - 교집합, 합집합 등 다양한 기능을 제공
- 내부 구조
  - Hash Table SETS : 데이터 형태가 문자 OR 데이터 개수가 512개 이상
  - Intset SETS (정수 배열) : 데이터 형태가 정수 AND 데이터 개수가 512개 이하

Hashes

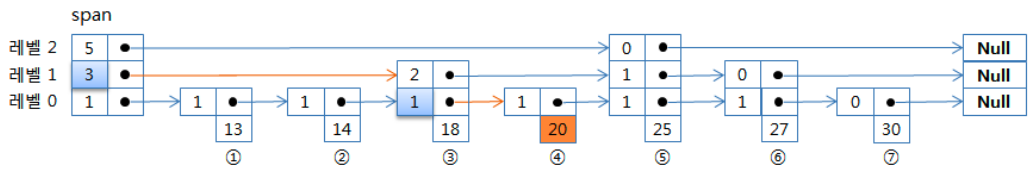
- key - (hashKey - value) 구조
- `HSET`, `HGET`, `HSETNX` 등 String과 명령어 비슷하게 사용 가능

Sorted Sets

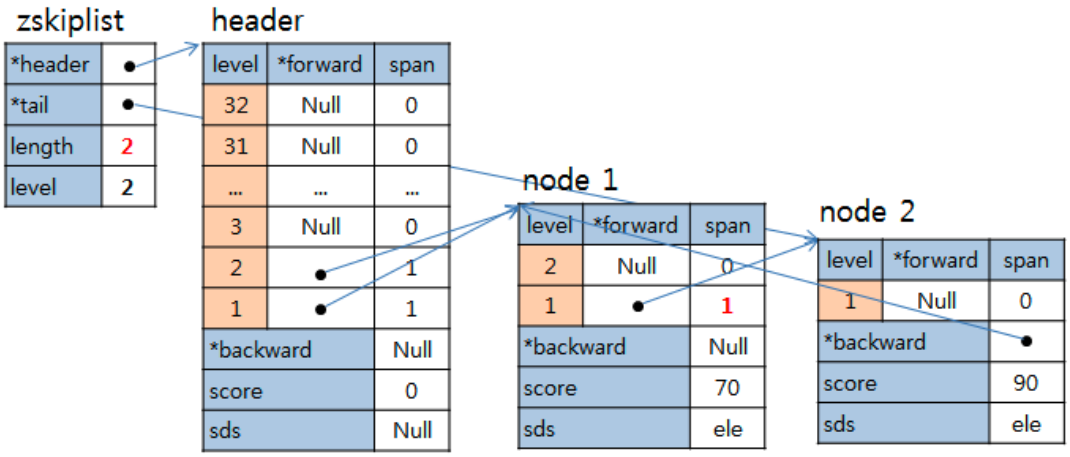
- Score(double-precision)를 통해 정렬, Score가 같다면 Key를 사전순으로 정렬
- 저장 할 때 정렬하여 저장 → O(log(n))
- 구조
  - SKIP LIST : 129개 이상 데이터 저장 OR 멤버의 길이가 하나라도 65바이트 이상
  - ZIP LIST : 128개 이하 데이터 저장 AND 멤버의 길이가 모두 64바이트 이하
- SKIP LIST의 탄생 과정
  - Level 을 갖는 Linkded List
    - Linked List에서 n번째 노드를 찾기 위해서는 n번 비교해야 한다는 단점을 극복하는 방법



  - 동전 던지기 (Redis에서는 1/4 확률로 고정)
    - 같은 것이 나오면 레벨을 올리고 다른 것이 나오면 해당 노드의 레벨 지정
    - 중간에 삽입시 이미 정해진 레벨을 모두 수정해야 하는 문제를 노드의 레벨을 독립적으로 지정하므로써 해결
  - 노드의 레벨에 SPAN 필드 추가
    - 동전 던지기 방법으로 인한 노드의 순서를 알수 없는 문제



  - 최종 결과물



Streams

- 이벤트 소싱, 모니터링, 알림에 사용하기 좋은 자료 구조
- 각 항목에 ID를 부여하고, ID를 이용해 조회 가능하다.
  - ID: 직접 입력 가능, 미 설정 시 `<millisecondsTime>-<sequenceNumber>`
- List 와 차이점
  - Consumer를 지정하여 데이터 읽기
  - Consumer가 처리에 성공, 실패했는지 확인 가능 → 실패 시 다른 소비자 할당

Geospatial

- 지리 공간 데이터 → 위도 경도 데이터 필요
- 가까운 공간 찾을 때 사용하기 유용함

HyperLogLog

- 집합의 원소 개수 추정
- 다른 자료구조에 비해 매우 적은 메모리를 사용하고 매우 낮은 (0.81%) 오차를 갖는다.
  - Sets
    - 1,000,000 개 숫자 → 4,858kb

        - 10,000,000 개 숫자 → 46,387kb
    - HyperLogLog
        - 12kb 사용