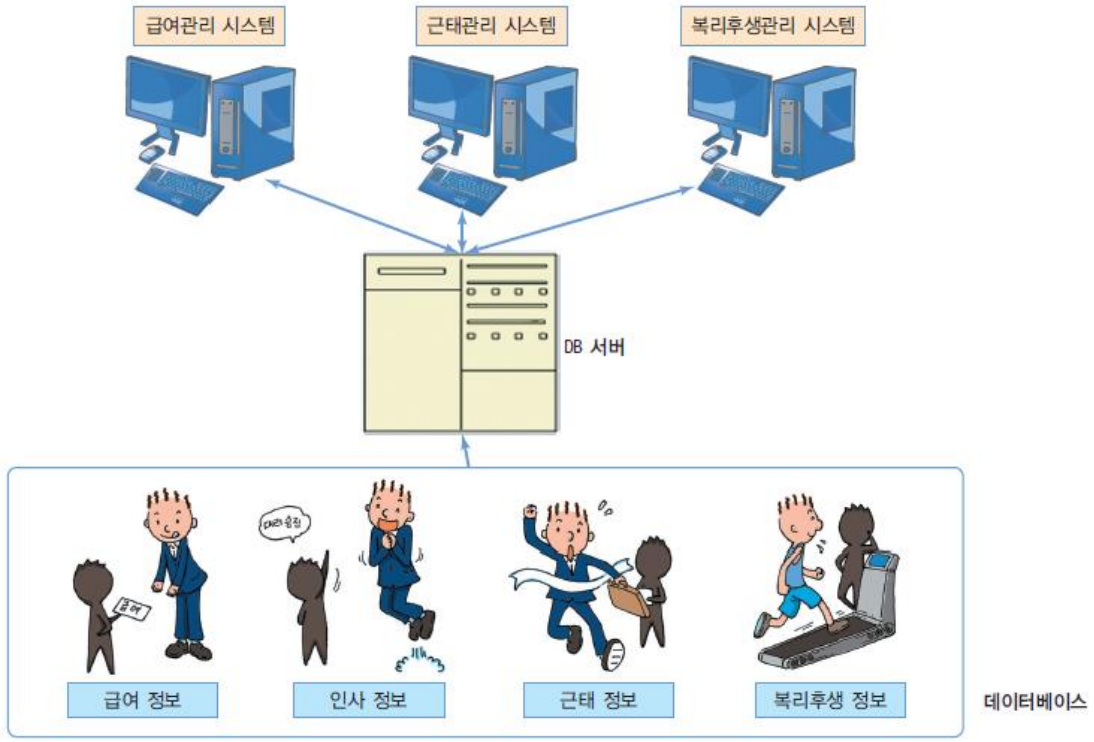
Redis

Quick Guide

데이터베이스와 데이터베이스 관리 시스템

- Database
  - 데이터의 집합 (a Set of Data)
  - 여러 응용 시스템(프로그램)들의 통합된 정보들을 저장하여 운영할 수 있는 공용(Shared) 데이터의 집합
  - 효율적으로 저장, 검색, 갱신할 수 있도록 데이터 집합들끼리 연관시키고 조직화되어야 한다



데이터베이스의 특성

- 실시간 접근성 (Real-time Accessibility)
  사용자의 요구를 즉시 처리할 수 있다
- 계속적인 변화 (Continuous Evolution)
  정확한 값을 유지하기 위해 삽입, 삭제, 수정 작업 등을 이용하여 데이터를 지속적으로 갱신할 수 있다
- 동시 공유성 (Concurrent Sharing)
  사용자마다 서로 다른 목적으로 사용하므로 동시에 여러 사람이 동일한 데이터에 접근하고 이용할 수 있다
- 내용 참조 (Contents Reference)
  저장한 데이터 레코드의 위치나 주소가 아닌 사용자가 요구하는 데이터의 내용, 즉 데이터 값에 따라 참조할 수 있어야 한다

데이터베이스와 데이터베이스 관리 시스템

- 데이터베이스 관리 시스템 (Database Management System = DBMS)
  - 데이터베이스를 관리하는 소프트웨어
  - 여러 응용 소프트웨어(프로그램) 또는 시스템이 동시에 데이터베이스에 접근하여 사용할 수 있게 한다
  - Oracle, Microsoft SQL Server, MySQL, DB2 등의 상용 또는 공개 DBMS가 있다

- NoSQL (Not Only SQL)
  - 기존의 DBMS는 데이터베이스 스키마를 작성하고 그 구조에 적합한 SQL이라는 구문을 이용하여 데이터를 관리
  - 빅데이터가 주요 흐름이 되면서 기존의 구조화된 DBMS로는 처리가 어려운 데이터들이 발생하게 됨
    - Volume : 데이터의 양 자체가 많아졌다
    - Velocity : 데이터가 생성되는 속도 자체가 빨라졌다
    - Variety : 형태 및 종류가 다양하여 구조화된 기존 DBMS의 스키마 틀 안에서 관리하기 어렵다
  - 이러한 새로운 데이터들을 다루기 위해 DBMS에도 변화가 필요 -> NoSQL의 등장

빅데이터 시대와 NoSQL

- 관계형 데이터베이스 모델은 1970년대에 처음 소개되었기 때문에
  인터넷과 클라우드 환경이 발달한 현대 애플리케이션들의 일부 수요를 만족시키지 못함
  - NoSQL은 기존 관계형 데이터베이스의 규칙 일부를 포기하는 대신,
    뛰어난 **확장성**이나 **성능**을 발전시키는 방향으로 개발

- NoSQL DBMS의 네 가지 분류

종류	예시
키-값 스토어	Redis, AWS Dynamo
컬럼 지향 스토어	HBase, Cassandra
도큐먼트 지향 스토어	MongoDB
그래프 데이터베이스	Neo4J

Redis

- Remote Dictionary Server
  - 메모리 기반 키-값 구조 관리 데이터 시스템
  - 모든 데이터를 메모리제 저장하고(in-memory) 조회하기 때문에 빠른 READ, WRITE 속도를 보장
  - 비 관계형 데이터베이스

- Memcached vs Redis

차이	Memcached	Redis
처리속도	In-Memory 방식으로 빠름	디스크와 메모리에 저장됨에도 Memcached와 비슷
데이터 저장	데이터는 메모리에만 저장 장애 발생시 데이터 복구 불가	데이터는 메모리와 디스크에 저장 장애 발생시에도 데이터 복구 가능
만료시점 지정	Expire 시점 지정, 만료시 데이터는 사라짐(캐시)	좌동
메모리 재사용	저장소 메모리를 재사용. 만료 이전에도 메모리 부족시 LRU 알고리즘에 따라 데이터를 삭제	저장소 메모리 재사용하지 않음 명시적으로만 데이터 제거 가능
데이터타입	문자열만 지원	문자열, Hash, List, Set, Sorted Set 등 다양

Redis
: Installation (on Ubuntu)

- https://redis.io/
- apt 저장소 업데이트

```
sudo apt update
sudo apt upgrade
```

- redis-server 패키지 설치

```
sudo apt install redis-server
sudo service redis-server status # 서비스 상태 확인
ps -ef | grep redis # 레디스 프로세스 확인
```

Redis
: Configuration (on Ubuntu)

- redis-server 서비스 정보를 확인해 봅니다.

```
more /etc/init.d/redis-server
```

```
# in /etc/init.d/redis-server
# 실행 정보
DAEMON=/usr/bin/redis-server
DAEMON_ARGS=/etc/redis/redis.conf
# ...
```

Redis
: Configuration (on Ubuntu)

- redis 서버 설정을 변경

```
sudo vi /etc/redis/redis.conf
```

```
# in /etc/redis/redis.conf
# ip binding
bind 0.0.0.0 ::1
# protected mode config
protected-mode no
# port config
port 6379
# log location
logfile /var/log/redis/redis-server.log
```

- 서비스 재시작

```
sudo service redis-server restart # 서비스 상태 확인
```

Redis
: Connect to Redis

- redis-cli를 이용한 redis 접속

```
$ redis-cli --raw
127.0.0.1:6379> PING
PONG
127.0.0.1:6379> INFO
# Server
...
127.0.0.1:6379> INFO Server
# Server
...
127.0.0.1:6379> QUIT
```

Data Types in Redis

Redis의 데이터 처리

- 용어의 정리
  - Key : 하나의 KEY는 하나 이상의 조합된 값으로 표현
    - 예: 주문번호, 주문번호+순번, 연월일+순번
    - KEY는 기본적으로 ASCII 데이터로 구성
  - Values : 해당 KEY에 의해 참조되는 구체적인 데이터 값
    - Value는 하나 이상의 Field 또는 Element로 구성
  - Table : 하나의 Database에서 데이터를 저장하는 논리적 구조
  - Data Sets : 테이블을 구성하는 논리적 단위
    - 하나의 데이터 셋은 하나의 KEY와 한 개 이상의 Field/Element로 구성

Database Commands in Redis

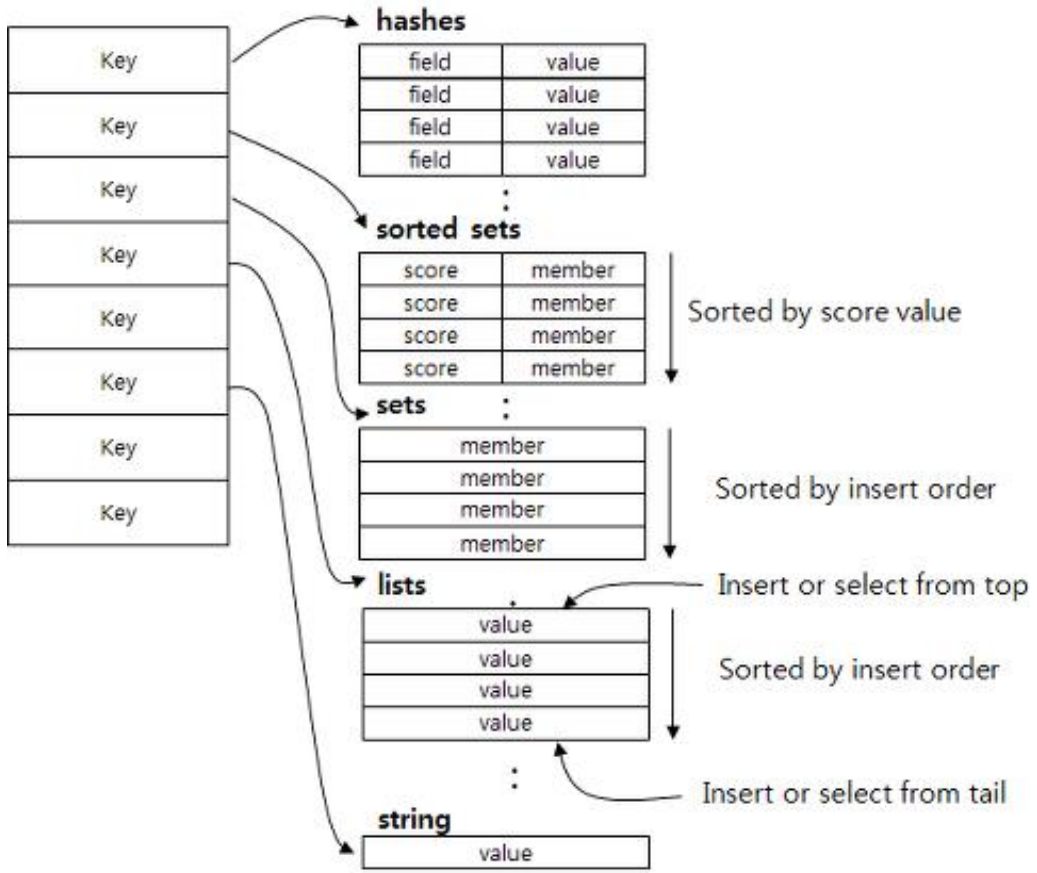
- Database Commands

Command	Description
keys {pattern}	{pattern}에 매칭되는 key 목록을 반환
exists {key}	{key}의 존재 여부 확인
rename {key} {newKey}	{key}를 {newKey}로 변경
expire {key} {seconds}	{key}를 {seconds}초 이후 만료
expireAt {key} {timestamp}	{key}의 만료 시점을 {timestamp}로 설정
pexpire {key} {milliseconds}	{key}를 {milliseconds}초 이후 만료
pexpireAt {key} {ms timestamp}	{key}의 만료 시점을 {ms timestamp}로 설정
ttl {key}	{key}의 남은 시간을 초 단위로 반환
pttl {key}	{key}의 남은 시간을 밀리초 단위로 반환
persist {key}	{key} 데이터의 만료 시점 제거
randomKey	임의의 키를 반환
type {key}	{key}의 데이터 타입 확인

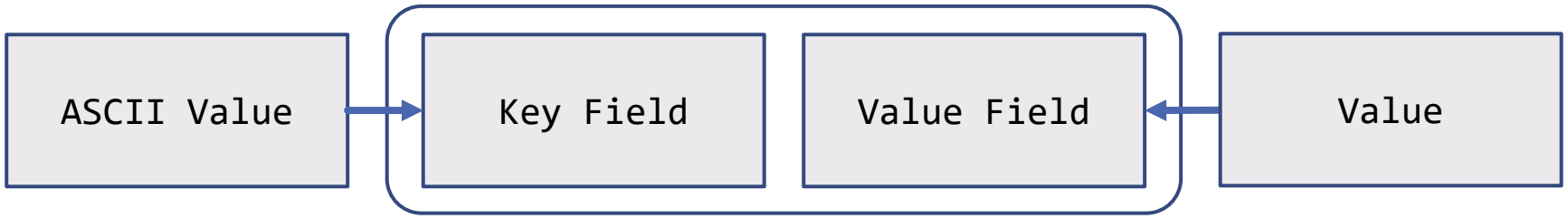
Data Types in Redis

- Data Types

종류	설명
strings	문자(text), Binary 유형
List	하나의 Key에 여러 배열 값을 저장
Hash	하나의 Key에 여러 Field-Value로 구성된 테이블을 저장
Set/Sorted Set	순서가 없는 데이터의 집합
Bitmaps	0 / 1로 표현되는 데이터
HyperLogLogs	Element 중 유일한 개수의 Element만 계산
Geospartial	좌표 데이터 저장/관리



String in Redis



- String
  - 가장 기본적인 **KEY-VALUE** 저장 방식. 문자, 숫자 등을 저장함
  - 문자, 숫자를 구분하는 형은 없다

Strings in Redis

- Strings - Basics

Command	Description
set {key} {value}	데이터를 저장
get {key}	데이터를 검색
getset {key} {newValue}	문자열 반환 후 {newValue}로 변경
strlen {key}	{key}에 매칭되는 문자열의 길이 반환
mget/mset	여러 개의 {key}와 {value}를 저장 혹은 검색
getrange {key} {start} {end}	{key}에 매칭되는 문자열의 범위 값 반환
setrange {key} {offset} {newValue}	{key}에서 범위 내용을 {newValue}로 변경

Strings in Redis

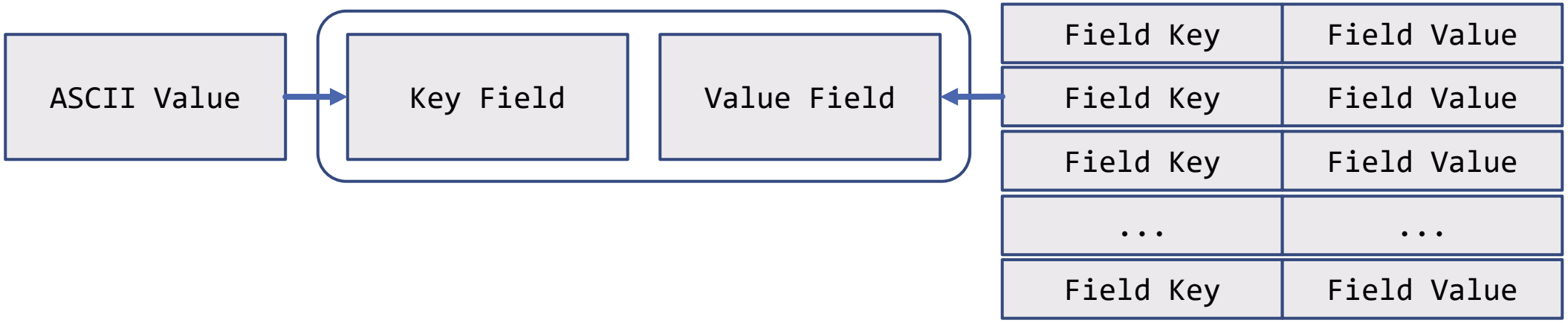
- Strings - Advanced

Command	Description
`setnx {key} {value}`	데이터를 저장(키가 없을 때)
`setex {key} {seconds} {value}`	데이터를 저장(만료 시점 명시)
`psetex {key} {milliseconds} {value}`	데이터를 저장(만료 시점 ms 명시)

- Strings - Numeric (atomic 연산)

Command	Description
`incr {key}`	정수로 변환하여 1 증가
`incrby {key} {increment}`	정수로 변환하여 {increment} 만큼 증가
`decr {key}`	점수로 변환하여 1 감소
`decrby {key} {decrement}`	정수로 변환하여 {decrement} 만큼 감소

Hash in Redis



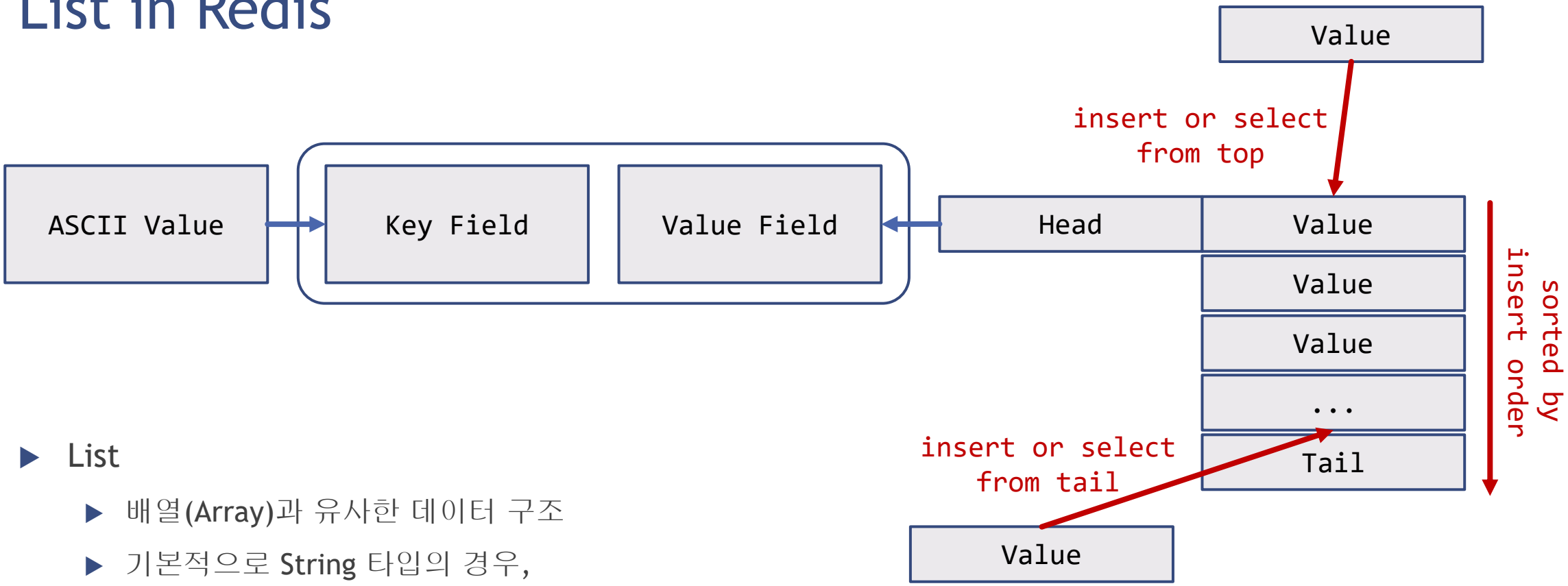
- Hash
  - RDB에서 Primary Key와 하나 이상의 컬럼으로 구성된 테이블과 유사한 데이터 구조
  - 기본적으로 필드 개수는 제한 없음
  - 주로 H 접두어를 가진 명령어를 사용

Hash in Redis

- Hash

Command	Description
hset {key} {field-key} {field-value}	필드 데이터를 저장
hget {key} {field-key}	필드 데이터를 검색
hgetall {key}	모든 필드 키와 필드 값을 반환
hkeys {key}	모든 필드 키의 목록을 반환
hvals {key}	모든 필드 값을 반환
hlen {key}	내부 아이템의 개수를 반환
hdel {key} {field-key}	필드 데이터 삭제
hincrby {key} {field-key} {increment}	필드 데이터를 {increment} 만큼 증가
hmget / hmset	여러 필드를 동시에 저장/검색

List in Redis



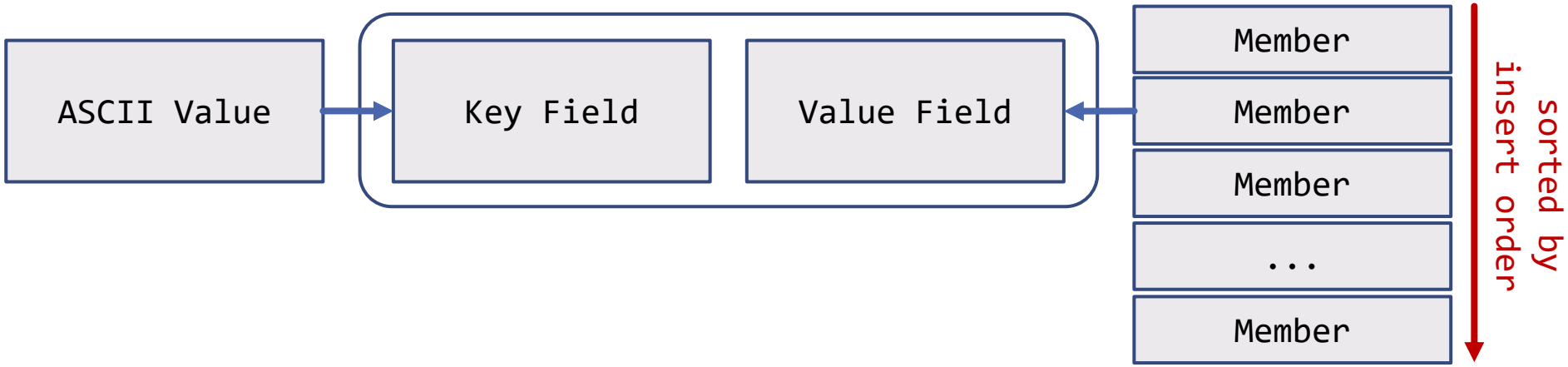
- List
  - 배열(Array)과 유사한 데이터 구조
  - 기본적으로 String 타입의 경우,
    배열에 저장할 수 있는 크기는 512MB

List in Redis

- List

Command	Description
lpush {key} {value} ...	리스트에 값 저장(head 방향)
lpop {key}	리스트로부터 값 인출(head 방향)
rpush {key} {value} ...	리스트에 값 저장(tail 방향)
rpop {key}	리스트로부터 값 인출(tail 방향)
llen {key}	리스트 요소의 개수 반환
lindex {key} {index}	인덱스에 의한 요소에 접근
linsert {key} before|after {pivot} {value}	{pivot} 이전/이후에 {value} 삽입
lset {key} {index} {newValue}	{index} 요소를 {newValue}로 변경
lrem {key} {count} {value}	{value} 값 요소를 {count}개 삭제
lrange {key} {start} {stop}	{start} ~ {stop}까지의 요소를 반환

Set in Redis



- Set
  - 데이터를 집합의 형태로 가지고 있음(중복 없음)

Set in Redis

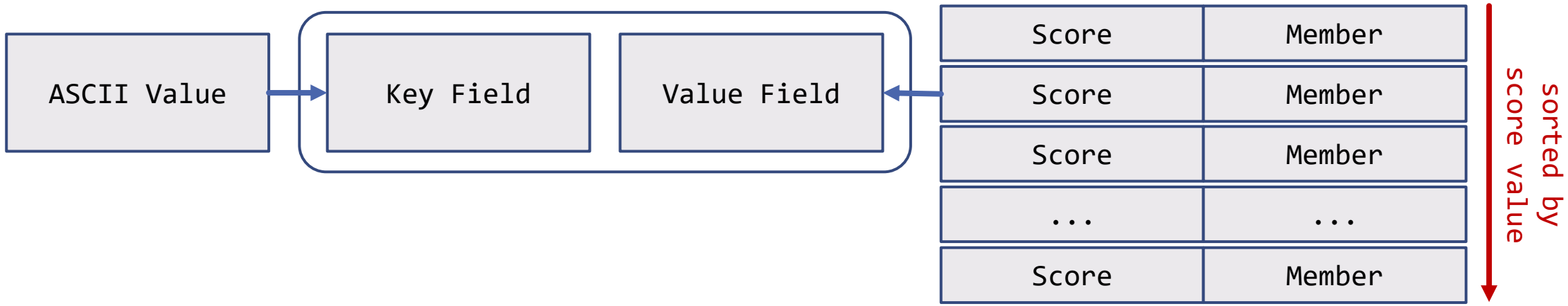
- Set

Command	Description
`sadd {key} {member}`	집합의 원소 등록
`smembers {key}`	집합의 모든 원소 검색
`scard {key}`	집합의 원소의 개수 반환
`sismember {key} {value}`	{value}가 집합의 원소인지 확인
`srem {key} {value}`	{value}와 일치하는 원소를 삭제

- Set - 집합 연산

Command	Description
`sdiff {key} {key2}`	차집합
`sinter {key} {key2}`	교집합
`sunion {key} {key2}`	합집합

SortedSet in Redis



- SortedSet
  - 데이터를 집합의 형태로 가지고 있음
  - Score와 함께 저장되어 Score를 기준으로 정렬

SortedSet in Redis

- SortedSet

Command	Description
zadd {key} {score} {member}	집합의 원소 등록(with Score)
zcard {key}	집합의 원소의 개수 반환
zcount {key} {min} {max}	점수 범위가 {min} ~ {max}인 원소의 개수
zincrby {key} {increment} {member}	{member}의 점수를 {increment}만큼 증가
zrange {key} {start} {stop}	score 기준 순위 범위 지정
zrangebyscore {key} {min} {max}	점수 범위가 {min} ~ {max}인 원소 목록
zrem {key} {member}	원소 삭제
zrank {key} {member}	원소의 순위 반환
zscore {key} {member}	원소의 점수 반환