연세대학교 YONSEI UNIVERSITY

LSM Tree & Bloom Filter

데이터베이스 시스템 12조

Sunghyun Ahn, Kijung Lee

{skd, rlwjd4177}@yonsei.ac.kr

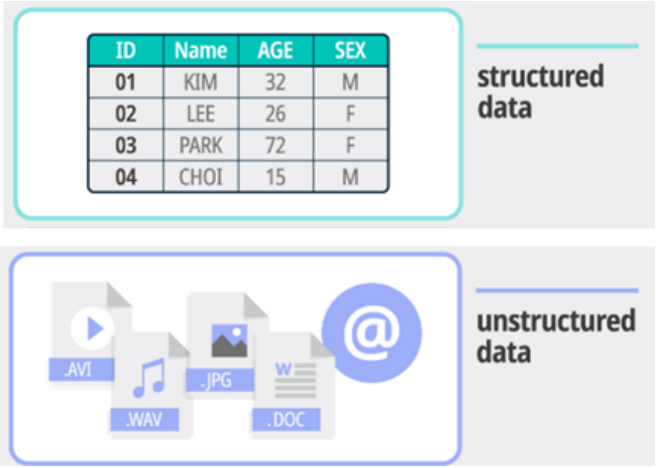
<2023/11/21>

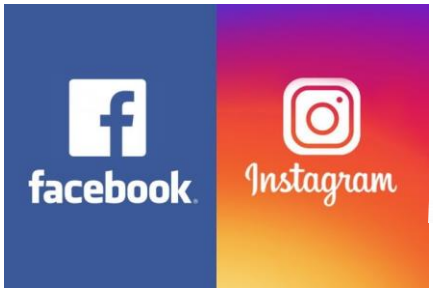
1 NoSQL

연세대학교 YONSEI UNIVERSITY

NoSQL과 비정형 데이터

- "Not only SQL"로, SQL만을 사용하지 않는 데이터베이스 관리 시스템(DBMS) → 비정형 데이터를 관리할 수 있음
- 비정형 데이터는 텍스트, 이미지, 음원 등과 같이 틀이 잡혀 있지 않은 데이터를 의미함
- Web 기술이 발전하면서 사용자가 데이터를 생성 및 공유 → 무수히 많은 비정형 데이터가 생성됨





YouTube

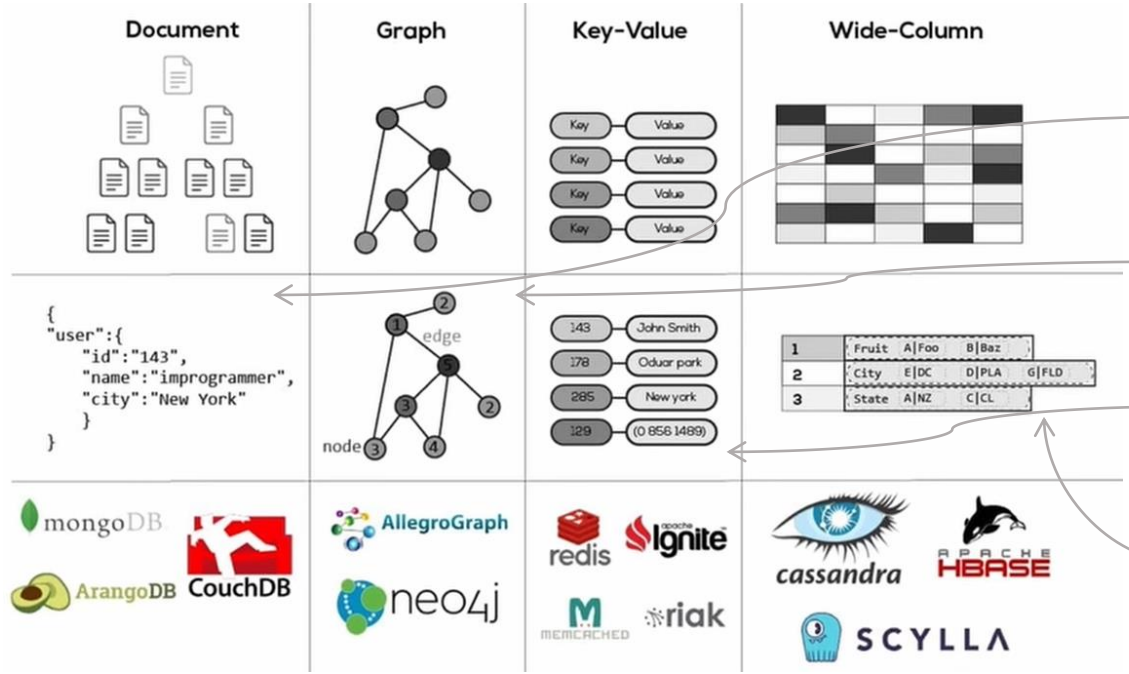
NAVER Google

1 NoSQL

연세대학교 YONSEI UNIVERSITY

Not Only SQL

- NoSQL DB의 특성을 크게 네 가지로 나누면 다음과 같음 (Document DB, Graph DB, Key-Value DB, Wide-Column DB)
- 어떤 상황에서 어떤 데이터베이스가 적합할지 알아보고 이용하면 됨



웹에서 서버와 클라이언트가 통신할 때 이용되는 파일: Json

소셜 네트워킹, 추천 엔진, 이상 탐지 등에 활용됨

IoT 기기의 센서들 값은 Key-Value 형태로 클라우드 저장됨

구조적으로 데이터를 저장하는 장점 유지, column은 flexible

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

LSM Tree

- Log-Structured Merge-Tree, Log 파일을 기반으로 작동하는 방식 (Log Structured Storage Engine)
- 연속적인 쓰기 연산을 수행하는 워크로드를 위해 설계됨
- NoSQL 중 Key-Value DB (ex. LevelDB, RocksDB 등)에서 이용하는 데이터 저장 방식

Log File

- Append-only 파일 (한 번 쓰여진 정보는 바뀌지 않고 새로운 내용은 항상 파일의 끝에 추가됨)

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

Log Structured Storage Engine 예제

Log 파일을 생성 및 조회하는 Bash Function

```
#!/bin/bash

db_set() {
  echo "$1,$2" >> database
}

db_get() {
  grep "$1," database | sed -e "s/^$1,//" | tail -n 1
}
```

새로운 key-value pair를 추가하는 함수

key에 들어가 있는 value를 리턴해주는 함수

Disk

1234, {"name":"sam"}
5678, {"name":"tom"}
1234, {"name":"sally"}

→ db_set의 성능은 좋지만, db_get의 성능은 좋지 않다.
(요청이 올 때마다 파일 전체를 스캔해야 하기 때문)

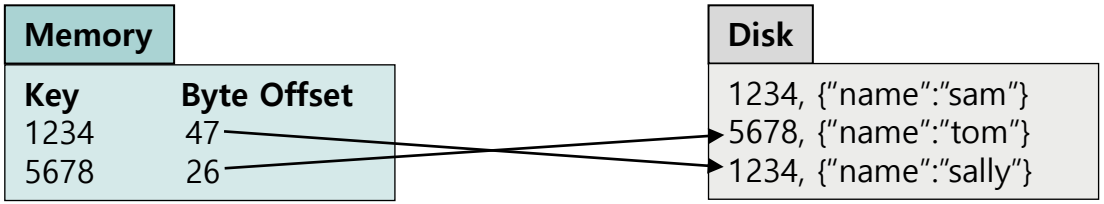
```
db_set 1234 '{"name":"sam"}'
db_set 5678 '{"name":"tom"}'
db_get 1234 ──────────→ {"name":"sam"}
db_set 1234 '{"name":"sally"}'
db_get 1234 ──────────→ {"name":"sally"}
```

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

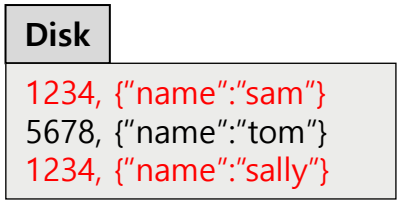
Index

- 데이터 Read 성능을 더 좋게 만들기 위해서 메모리에 추가적으로 가지고 있는 데이터 구조
- Log파일의 Key와 Disk의 Byte Offset을 저장하면, offset에 대응되는 위치에서 value를 가져올 수 있음
  - → Log 파일을 full scan하지 않고도 Read를 할 수 있음



문제점

- Database에 update가 많을수록 쓸데없는 데이터가 차지하게 됨



2 LSM Tree

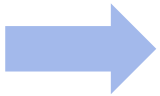
연세대학교 YONSEI UNIVERSITY

Segment File

Log file의 일정 사이즈가 되면 새로운 파일을 만들어서 Database 파일을 여러 개로 가지고 있게 하는 개념

Disk

1234, {"name":"sam"}
5678, {"name":"tom"}
1234, {"name":"sally"}
2111, {"name":"lee"}
2116, {"name":"hong"}
2112, {"name":"ahn"}
2114, {"name":"kim"}



Segment File1

1234, {"name":"sam"}
5678, {"name":"tom"}
1234, {"name":"sally"}
2111, {"name":"lee"}

← immutable

Segment File2

2116, {"name":"hong"}
2112, {"name":"ahn"}
2114, {"name":"kim"}

한 Segment 파일 당 4개씩 저장

→ 이전에 만들어진 segment file들은 더 이상 바뀌지 않으므로 최적화하기에 좋음

(->) Compaction: 쓸데 없는 데이터 지우기

Segment File1

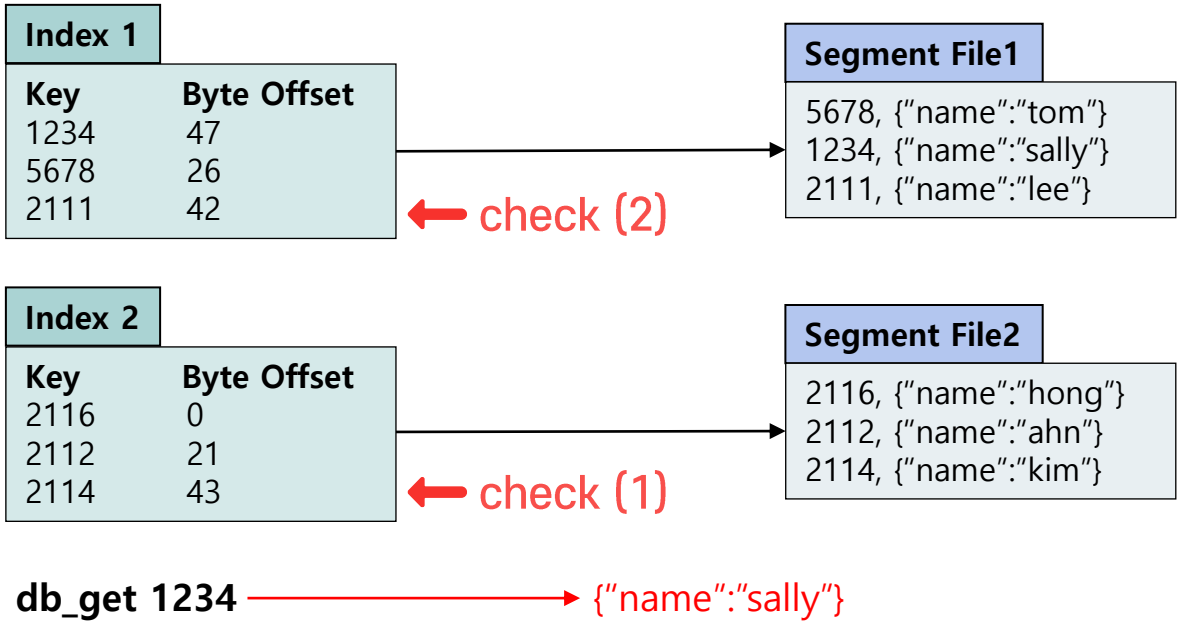
~~1234, {"name":"sam"}~~
5678, {"name":"tom"}
1234, {"name":"sally"}
2111, {"name":"lee"}

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

Segment File Read

- Key가 가장 최근에 있는 segment file의 인덱스에 있는지 확인함 (->) 없으면 그 다음 오래된 인덱스를 확인함 (->) 끝까지 없으면 key가 존재하지 않음



2 LSM Tree

연세대학교 YONSEI UNIVERSITY

내용 정리

- 초기 Storage Engine은 Write의 성능은 좋지만 Read의 성능은 안 좋다 (전체 파일을 scan해야 되기 때문)
  - → Index를 이용해서 전체 파일을 읽는 수고를 덜음 (Read 성능 개선)
- Log file의 특성 상 write가 많을수록 쓸데없는 데이터가 차지하게 됨 (한 번 쓰여진 정보는 바뀌지 않는 특성 때문)
  - → Log file을 segment file들로 분할 후 최적화하여 데이터 공간 낭비를 줄임 (Disk 공간 낭비 문제 해결)

문제점

- 존재하는 모든 Key가 Memory(Index) 안에 들어갈 수 있어야 함 (->) 메모리는 많은 공간 요구
- Key들이 정렬되어 있지 않기 때문에 Range 쿼리하기가 어려움

Index 1

Key	Byte Offset
1234	47
5678	26
2111	42

Index 2

Key	Byte Offset
2116	0
2112	21
2114	43

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

LSM Tree

- SSTable (Sorted String Table)을 이용해서 두 가지 문제점(메모리 공간 낭비, Range 쿼리 어려움)을 해결한 데이터 구조
- SSTable은 Segment File과 비슷한데 파일 안에 Key로 정렬이 되어 있음

Segment File1

5678, {"name":"tom"}
1234, {"name":"sally"}
2111, {"name":"lee"}

SSTable

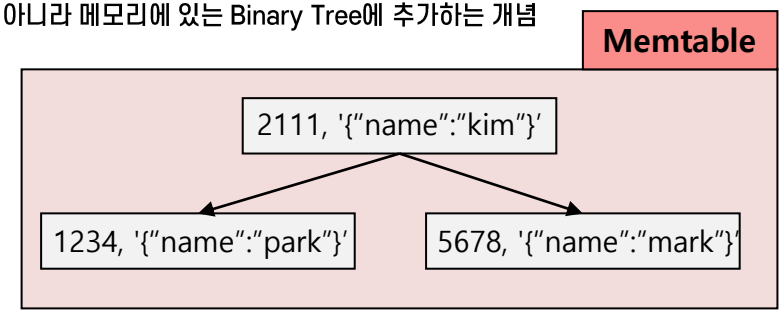
1234, {"name":"sally"}
2111, {"name":"lee"}
5678, {"name":"tom"}

→ 데이터의 끝에만 추가하는 것이 불가능함 (append-only 불가능)

MemTable

- 메모리에 저장하고 있는 Binary Tree *[Balancing Binary Tree]*
- 새로운 Key-Value를 추가하고자 할 때 SSTable에 넣는 것이 아니라 메모리에 있는 Binary Tree에 추가하는 개념

1234, '{"name":"sam"}'
5678, '{"name":"mark"}'
1234, '{"name":"park"}'
2111, '{"name":"kim"}'

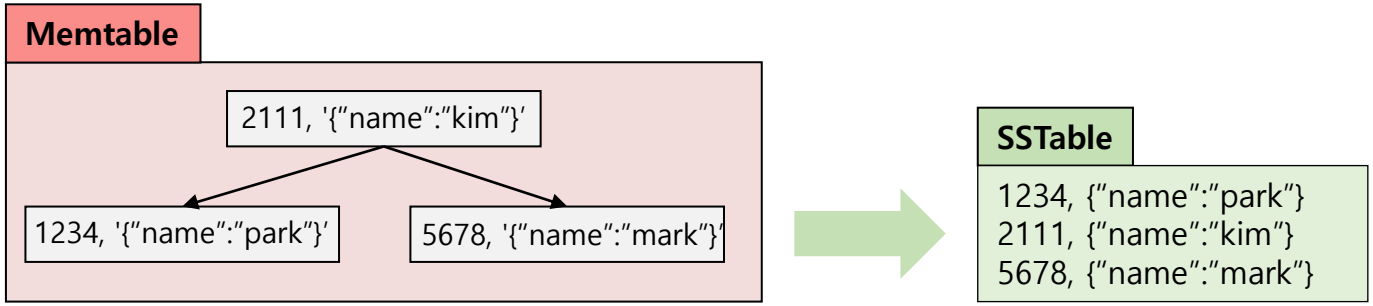


2 LSM Tree

연세대학교 YONSEI UNIVERSITY

Memtable & SSTable

- Memtable이 일정 사이즈를 넘어가면 트리에 있는 내용을 SSTable에 옮김 (->) SSTable도 여러 개 제작 가능함
  - → 데이터의 끝부분에 추가하는 형태, Key가 Sorted된 순서대로 write를 할 수 있음



→ 내용을 옮기는 중 DB가 Crash되면 복구가 불가능함

- Disk에 Log 파일을 제작하여 Memtable과 같이 기록함 (Memtable 복구 용도)

1234, '{"name":"sam"}'
5678, '{"name":"mark"}'
1234, '{"name":"park"}'
2111, '{"name":"kim"}'

Disk

1234, '{"name":"sam"}'
5678, '{"name":"mark"}'
1234, '{"name":"park"}'
2111, '{"name":"kim"}'

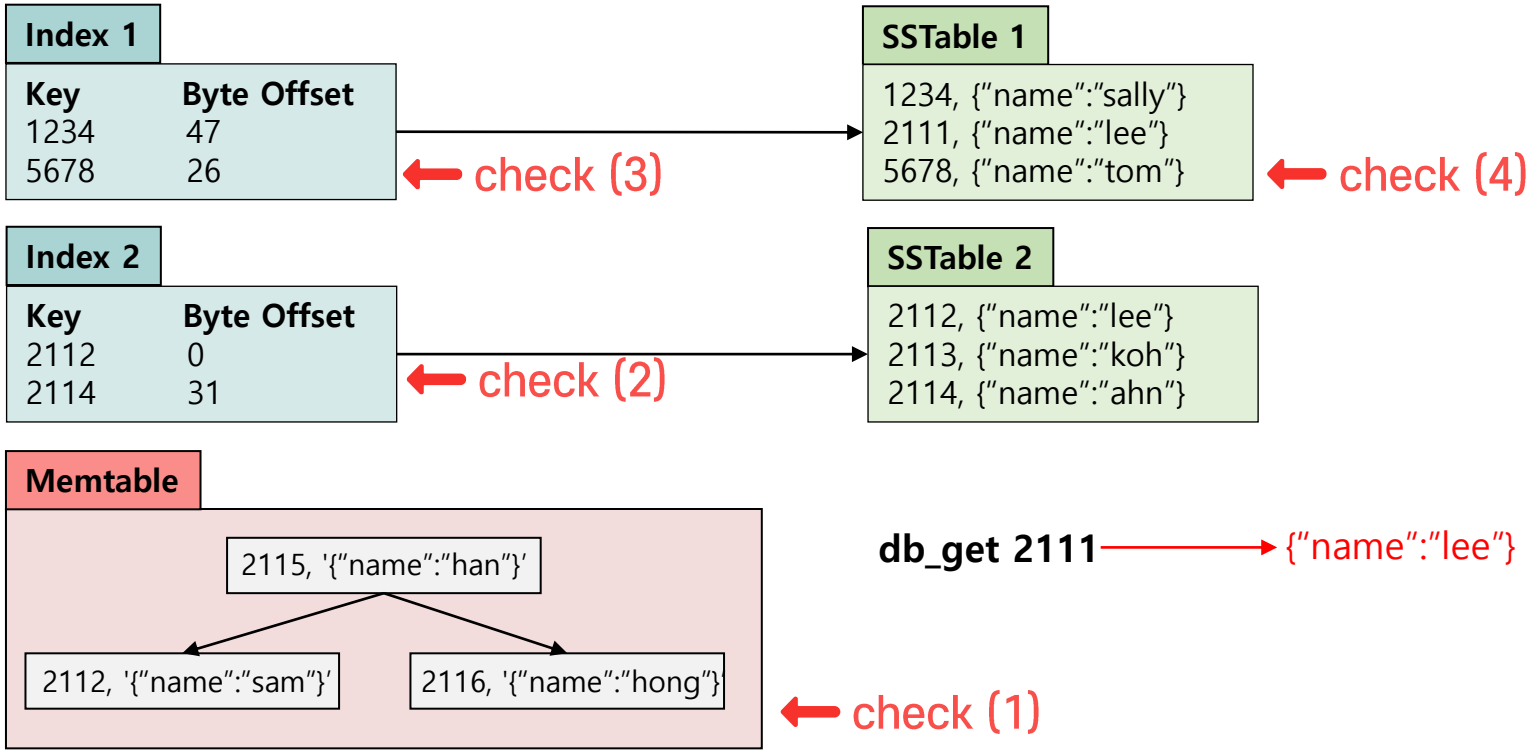
SSTable에 기록이 완료되면
Log 파일의 내용도 지움

2 LSM Tree

연세대학교 YONSEI UNIVERSITY

LSM Tree Read

- Read 성능을 좀 더 빠르게 하기 위해 Index를 사용함 (SSTable이 정렬되어 있으므로 모든 Key가 있을 필요가 없음)

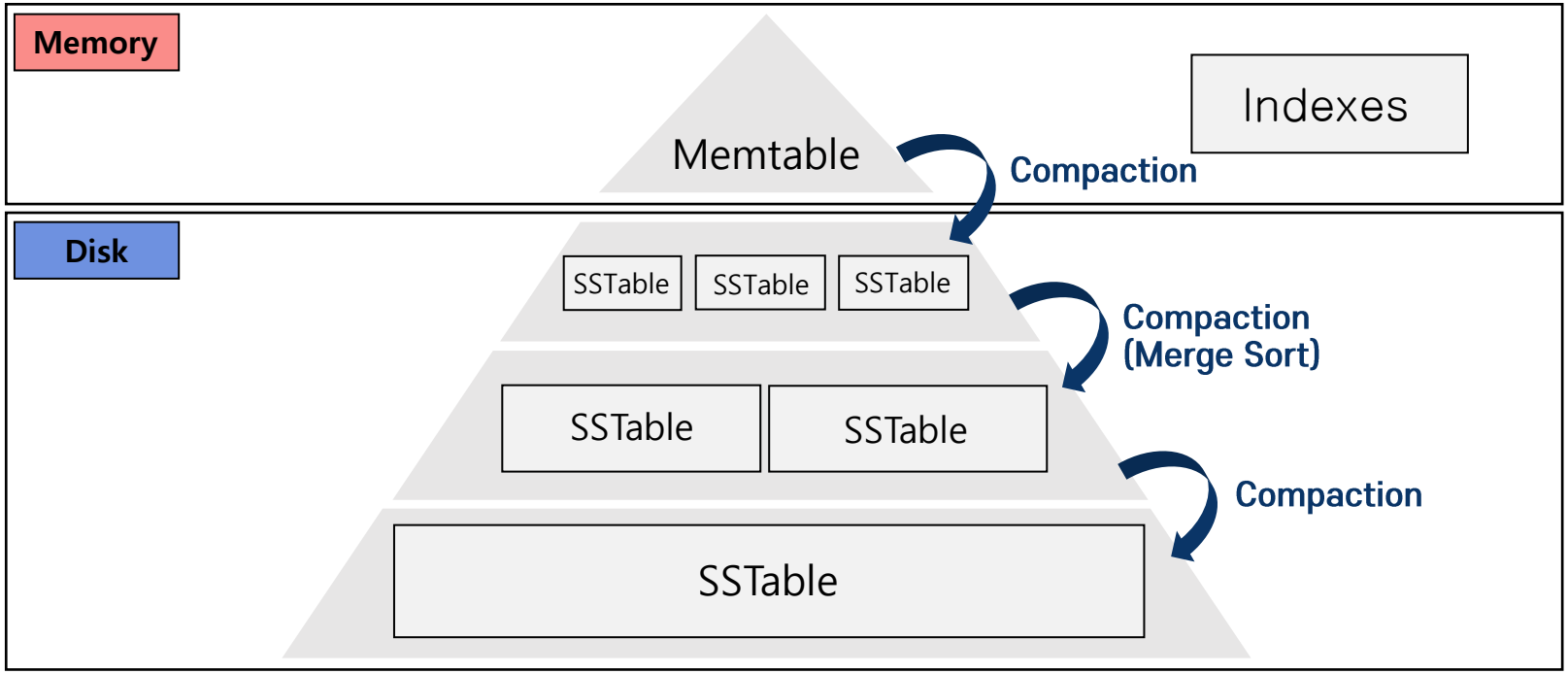


2 LSM Tree

연세대학교 YONSEI UNIVERSITY

LSM Tree Architecture

SSTable의 개수가 많아지면 Read 성능이 저하될 수 있기 때문에 상위 레벨의 사이즈가 넘어가면 Compaction하여 더 큰 SSTable을 제작하는 구조

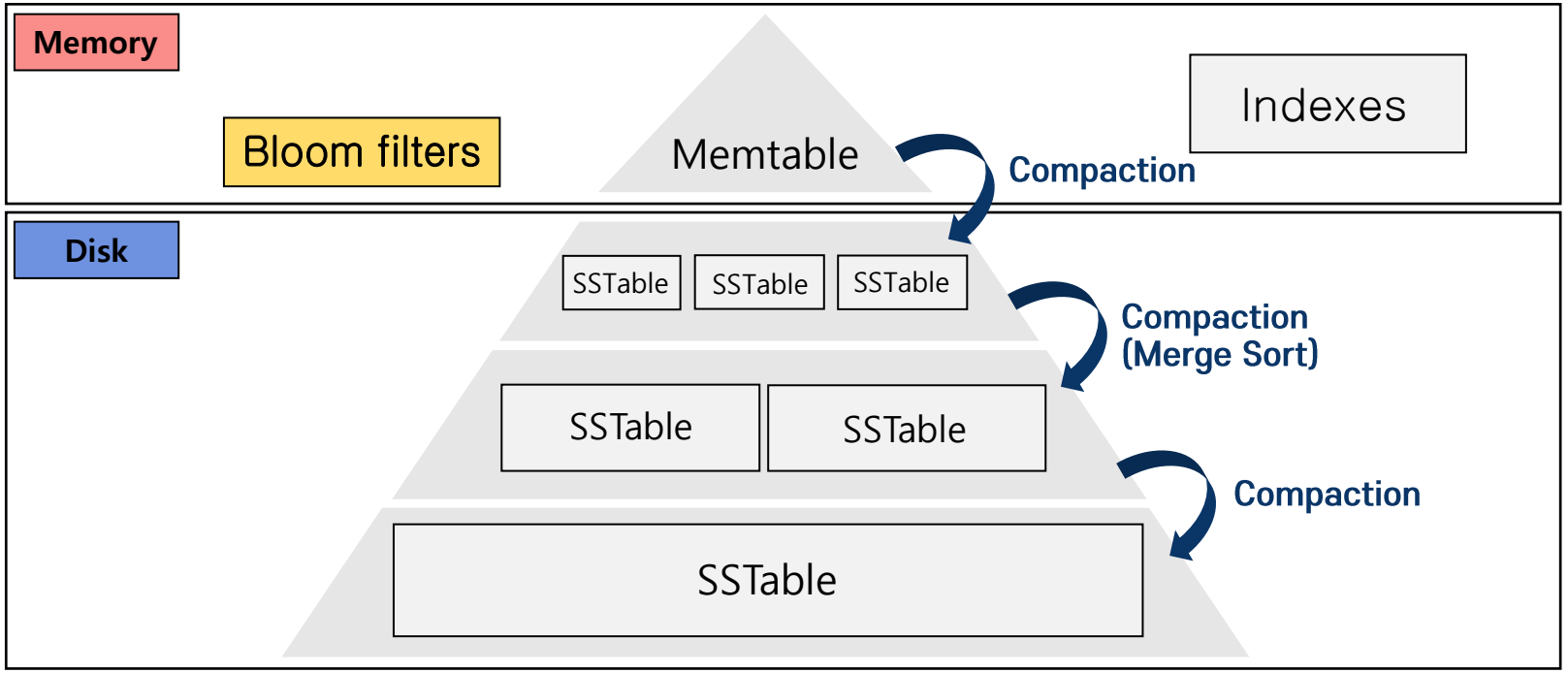


2 LSM Tree

연세대학교 YONSEI UNIVERSITY

LSM Tree Read with Bloom filters

- Bloom filter를 이용하면 각 레벨에서 key가 존재하지 않으면 해당 레벨을 건너뛸 수 있으므로 연산량이 줄어듦

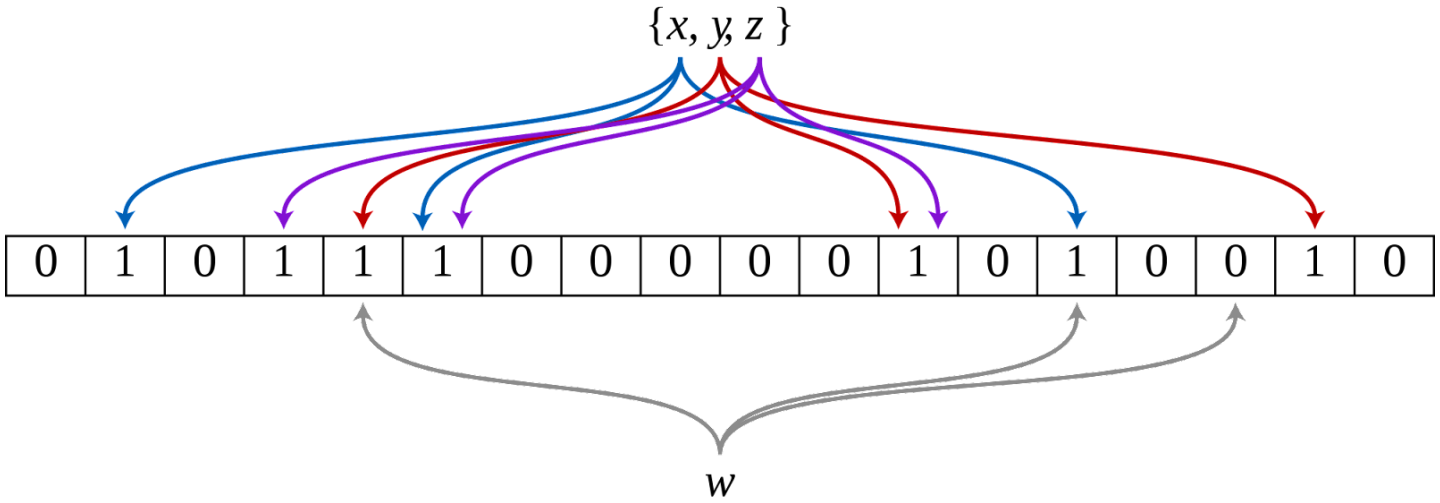


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters?

- Bloom filter란 특정 원소가 집합에 속하는지 검사하는데 사용할 수 있는 확률형 자료 구조
- Bloom filter는 원소의 전체 데이터를 저장하지 않고, k개의 Hash 함수를 통해 원소의 특징 값들만 뽑아 비트 배열에 반영시킴
  - → 정확도는 떨어지게 되지만 메모리 사이즈를 매우 절약할 수 있음.
- Bloom filter는 False Positive가 발생하는 것이 가능하지만, False Nagative는 절대로 발생하지 않음

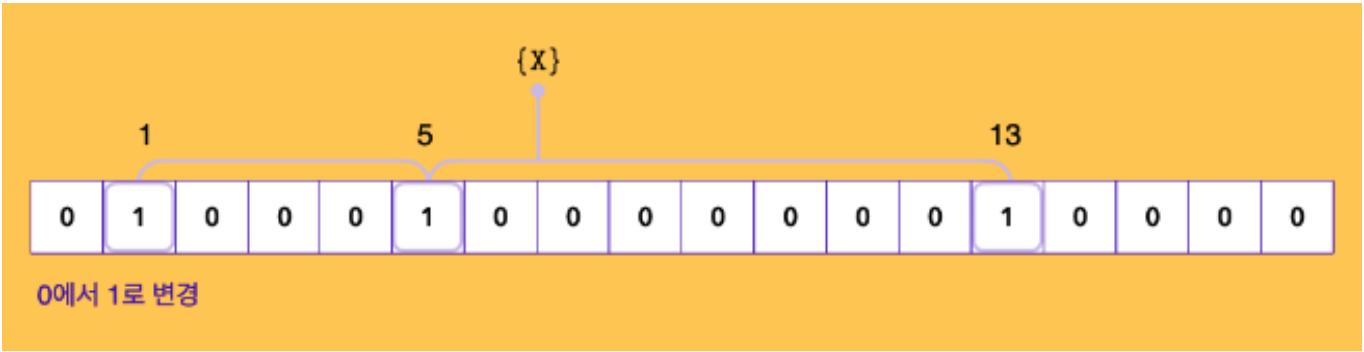


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters 예시

- X라는 원소값에 대해 3개의 해시 함수를 작동시키고 출력값에 해당하는 배열 인덱스를 1로 변경
- Hash1(X) = 1, Hash2(X) = 5, Hash3(X) = 13

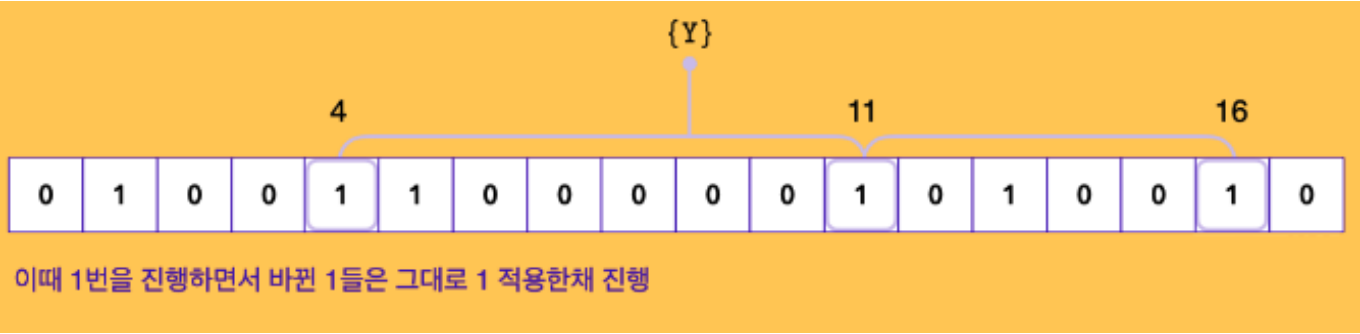


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters 예시

- Y라는 원소값에 대해 3개의 해시 함수를 작동시키고 출력값에 해당하는 배열 인덱스를 1로 변경
- Hash1(Y) = 4, Hash2(Y) = 11, Hash3(Y) = 16

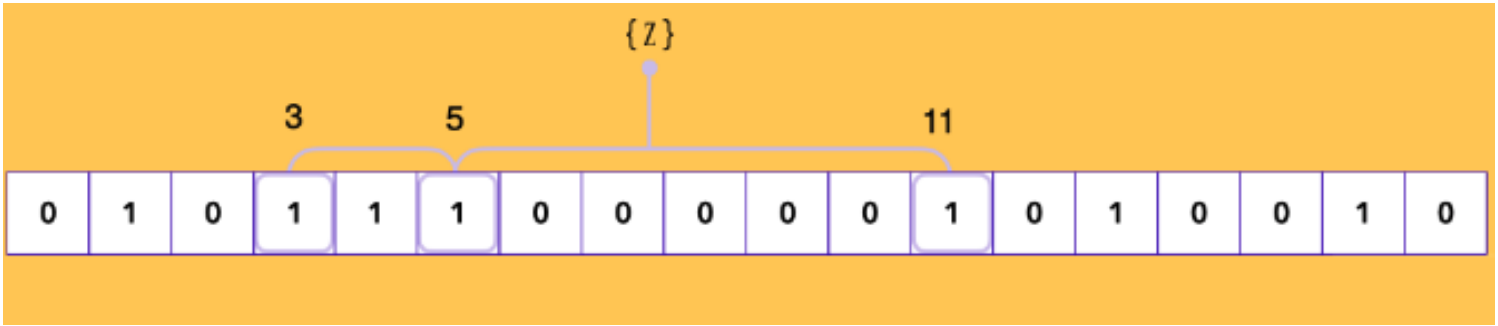


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters 예시

- Z라는 원소값에 대해 3개의 해시 함수를 작동시키고 출력값에 해당하는 배열 인덱스를 1로 변경
- Hash1(Z) = 3, Hash2(Z) = 5, Hash3(Z) = 11

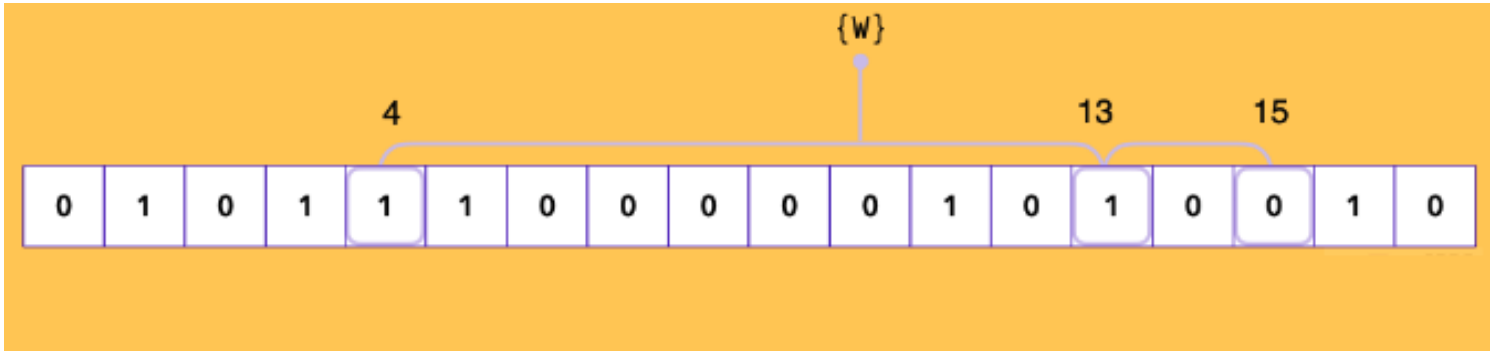


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters 예시

- W라는 원소값에 대해 3개의 해시 함수를 작동시키고 출력값에 해당하는 배열 인덱스를 1로 변경
- Hash1(W) = 4, Hash2(W) = 13, Hash3(W) = 15

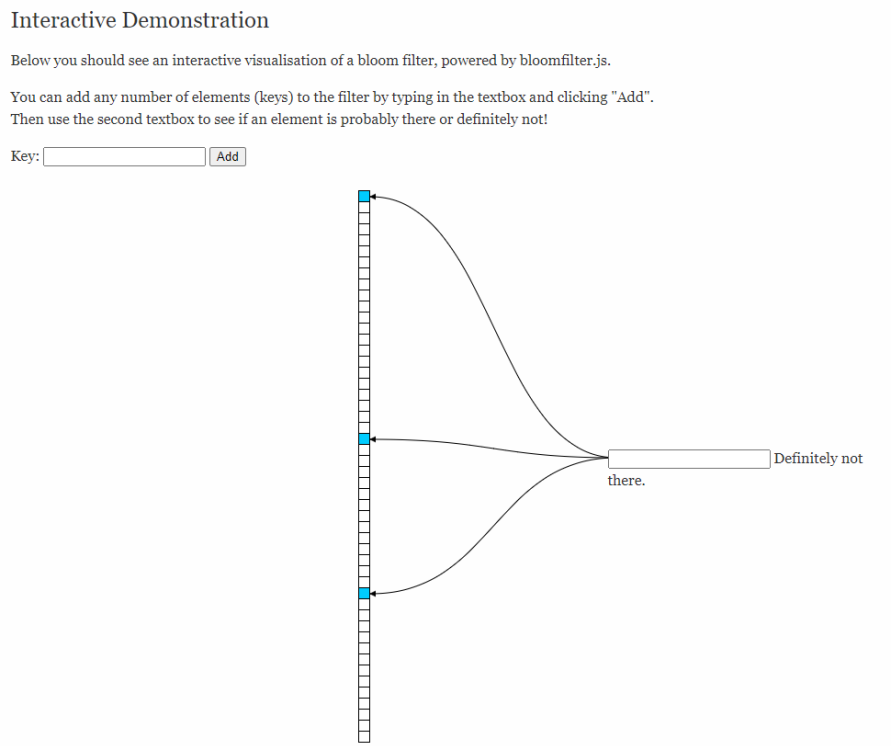


3 Bloom filters

연세대학교 YONSEI UNIVERSITY

Bloom filters 실습

https://www.jasondavies.com/bloomfilter/



4 Reference

연세대학교 YONSEI UNIVERSITY

References

- NoSQL (https://www.youtube.com/watch?v=pk7FP2FDIw8)
- LSM Tree (https://www.youtube.com/watch?v=i_vmkaR1x-I)
- Bloom filter (https://www.youtube.com/watch?v=V3pzxngeLqw)
- Bloom Filter (https://about-tech.tistory.com/175)
- Bloom Filter Simulation (https://www.jasondavies.com/bloomfilter/)

연세대학교 YONSEI UNIVERSITY

Thank You

데이터베이스 시스템 12조

Sunghyun Ahn, Kijung Lee

{skd, rlwjd4177}@yonsei.ac.kr

<2023/11/21>