INDEX

인덱스

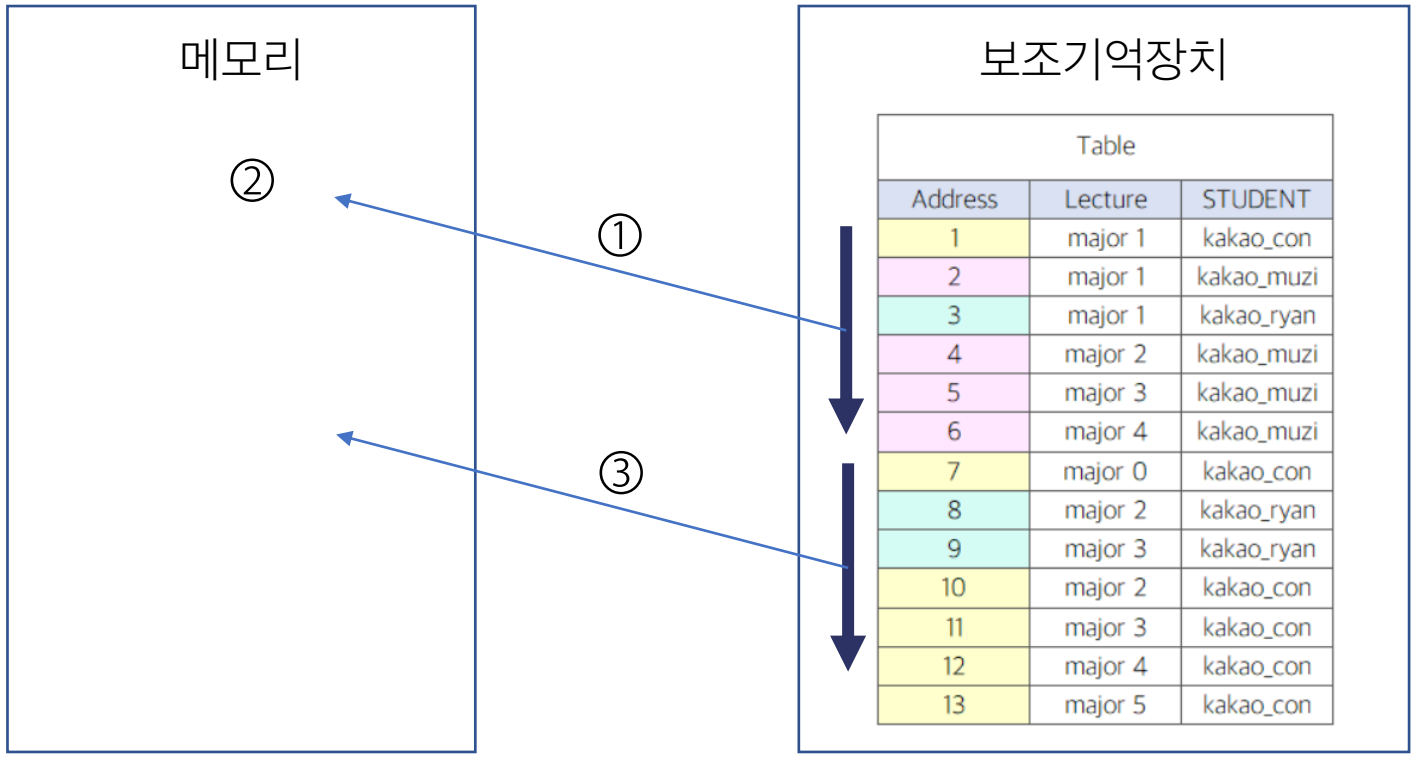
가장 느린 보조기억장치에
저장된 데이터를
빠르게 찾기 위한 자료구조

데이터를 찾는 방법

FULL SCAN
VS
INDEX SCAN

FULL SCAN 테이블 전체 스캔

STUDENT 가 `kakao_con` 인 학생이 수강하고 있는 과목 리스트를 뽑아주세요



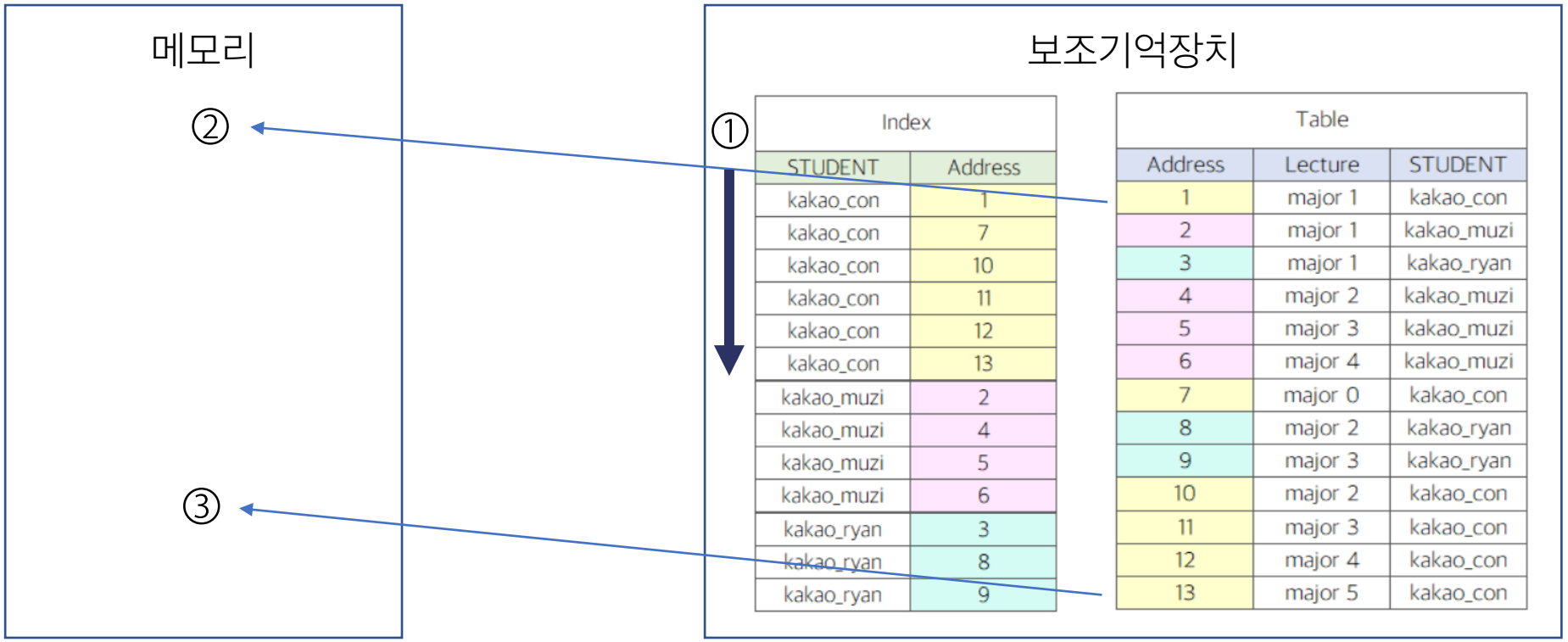
Table

Address	Lecture	STUDENT
1	major 1	kakao_con
2	major 1	kakao_muzi
3	major 1	kakao_ryan
4	major 2	kakao_muzi
5	major 3	kakao_muzi
6	major 4	kakao_muzi
7	major 0	kakao_con
8	major 2	kakao_ryan
9	major 3	kakao_ryan
10	major 2	kakao_con
11	major 3	kakao_con
12	major 4	kakao_con
13	major 5	kakao_con

① STUDENT 테이블의 블록을 메모리에 올린다
② 메모리에 올라온 블록에서 student가 kakao_con 인 데이터를 찾는다
③ kakao_con은 다른 블록에 존재할 수 있으므로 다음 블록을 다시 메모리에 올려서 검사한다

INDEX SCAN 테이블이 아닌 인덱스 스캔

STUDENT 가 `kakao_con` 인 학생이 수강하고 있는 과목 리스트를 뽑아주세요



① 테이블이 아닌 인덱스를 먼저 조회한다. 인덱스는 STUDENT를 기준으로 인덱싱이 되어있다.
② 인덱스에는 인덱스에 사용되는 키와 row의 주소가 저장 되어있기 때문에 바로 해당 row를 메모리에 올린다.
③ 인덱스는 이미 정렬 되어있기 때문에 kakao_con이 마지막으로 조회된 이후의 row는 검사할 필요가 없다.

FULL SCAN 단점

속도

데이터는 기억장치 중 속도가 가장 느린 보조기억장치에 저장
특정 데이터를 찾기 위해 테이블의 모든 행을 메모리에 가져와서 검사하는 것은 시간이 많이 걸림
인덱스를 이용하여 일치하는 행만 메모리에 가져오는 것이 효율적

INDEX 자료구조

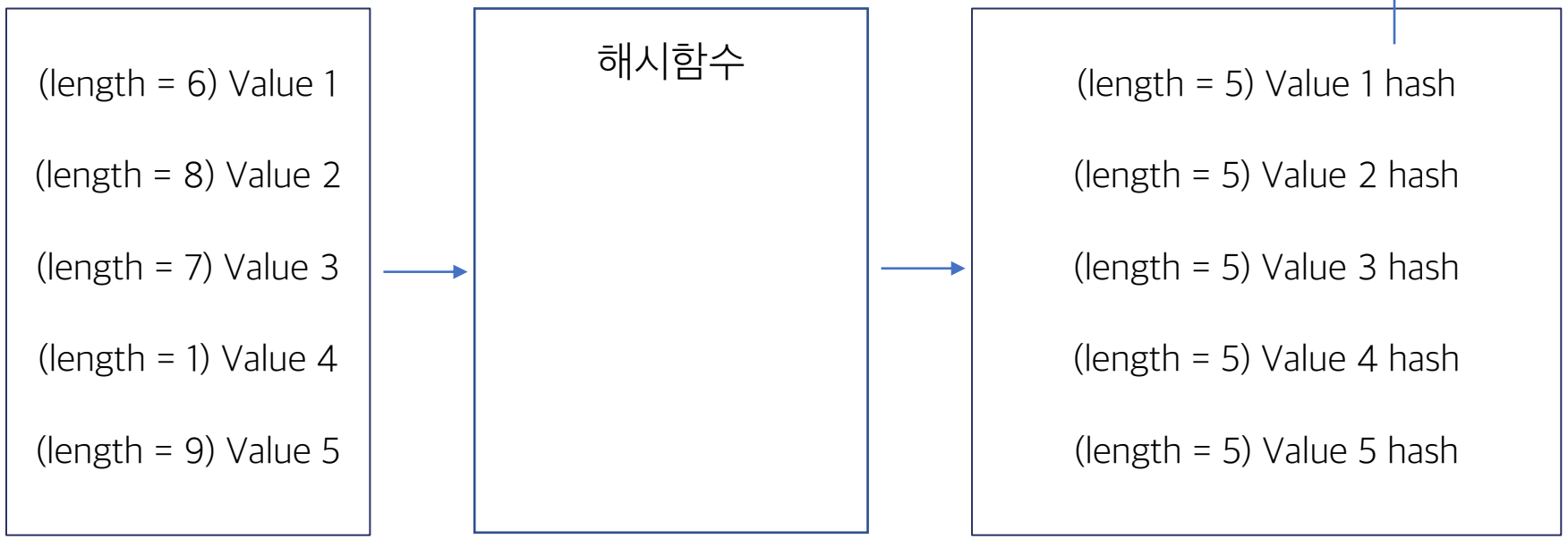
Hash INDEX
B-TREE INDEX
B+TREE INDEX

Hash INDEX

해시의 충돌
- 길이가 다른 value들이 해시함수를 통해 고정길이의 값으로 변했을 때 일치한다면 이를 해시의 충돌이 발생했다고 한다.

해시

다양한 길이를 가진 데이터를 해시 함수를 통해
고정 길이의 값으로 변환시키는데 이를 해시라고 한다.

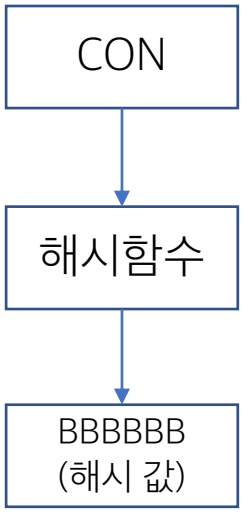


해시 값은 복호화가 불가능하다

Hash 장점

1. 데이터를 가져오는데 걸리는 시간은 O(1)
2. 인덱스 값을 고정 크기의 값으로 저장하기 때문에 용량을 덜 차지함

ID가 CON인 학생의 전체 정보가 궁금합니다
Select * from student where id = 'con'



보조기억장치

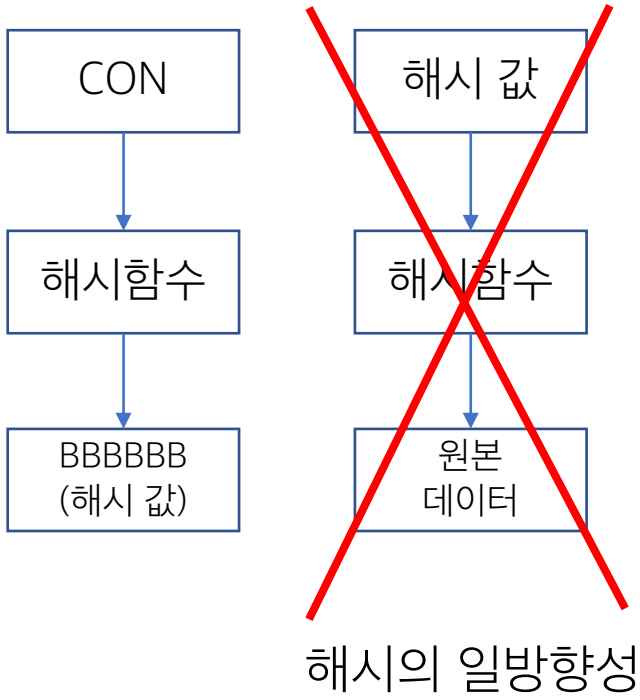
INDEX	
학생 ID	ADDRESS
AAAAAA	1
BBBBBB	2
CCCCCC	3
DDDDDD	4
EEEEEE	5
FFFFFF	6
GGGGGG	7

STUDENT		
ADDRESS	학생 ID	핸드폰 번호
2	CON	
3	RYAN	
7	MUZI&CON	
6	NEO	
5	SALLY	
1	MUZI	
4	BROWN	

Hash 단점

1. 길이가 다른 데이터를 고정 크기의 값으로 만들기 때문에 해시 값이 충돌이 날 수 있다. 그렇기 때문에 적절한 해시알고리즘을 사용해야 함
2. 등호 연산 불가능 (> , < , LIKE와 같이 문자열 추출 등..)

ID의 앞자리가 C로 시작하는 학생의 전화번호가 궁금합니다
Select * from student where id LIKE='C%'



해시의 일방향성

보조기억장치

INDEX

학생 ID	ADDRESS
AAAAAA	1
BBBBBB	2
CCCCCC	3
DDDDDD	4
EEEEEE	5
FFFFFF	6
GGGGGG	7

STUDENT

ADDRESS	학생 ID	핸드폰 번호
2	CON	
3	RYAN	
7	MUZI&CON	
6	NEO	
5	SALLY	
1	MUZI	
4	BROWN	

Linked List를 사용한다면?

ID가 gggggg인 학생의 전체 정보가 궁금합니다

Linked List 다음 노드의 주소를 가지고 있는 자료구조

만약 10000000개의 데이터를 선형탐색한다면?

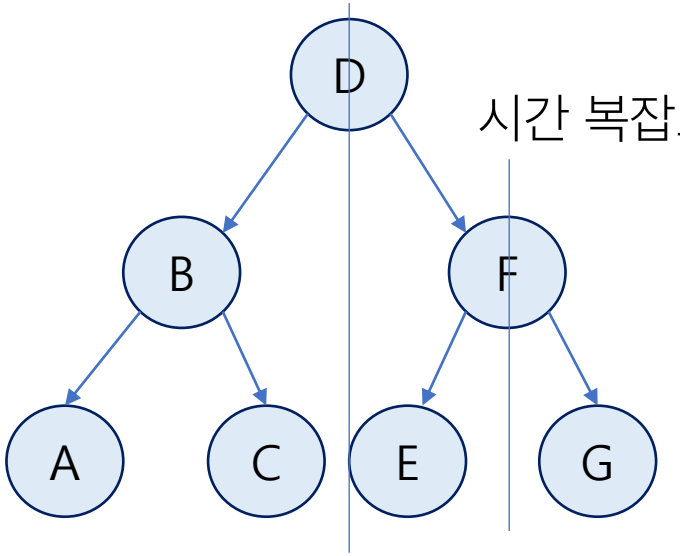
시간 복잡도 O(N)

보조기억장치

INDEX	
학생 ID	ADDRESS
AAAAAA	1
BBBBBB	2
CCCCCC	3
DDDDDD	4
EEEEEE	5
FFFFFF	6
GGGGGG	7

이진트리(Binary Search Tree)

ID가 ggggggg인 학생의 전체 정보가 궁금합니다



시간 복잡도 o(logn)

트리 높이

노드 개수 = $2^{h+1} - 1$

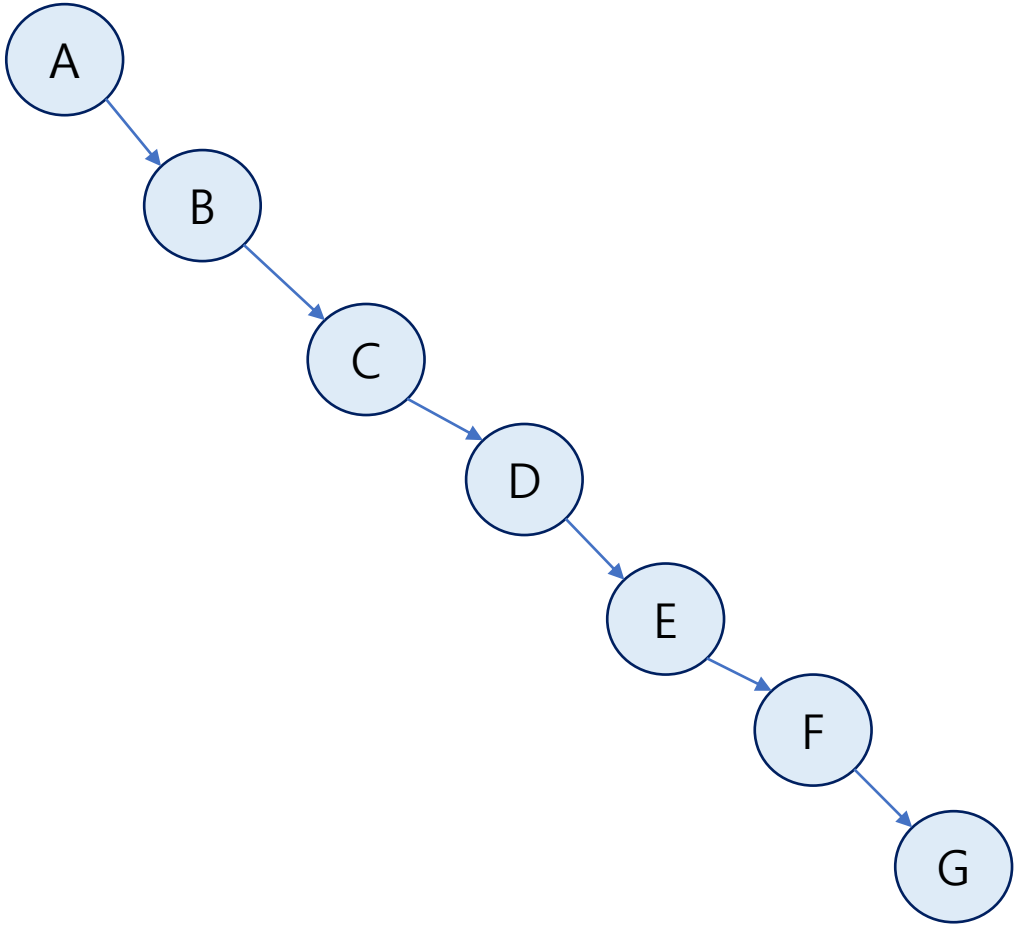
$2^{h+1} = N+1$

$h+1 = \log_2(N+1)$

$h = \log_2(N+1) - 1$

이진트리(Binary Search Tree)

ID가 gggggg인 학생의 전체 정보가 궁금합니다

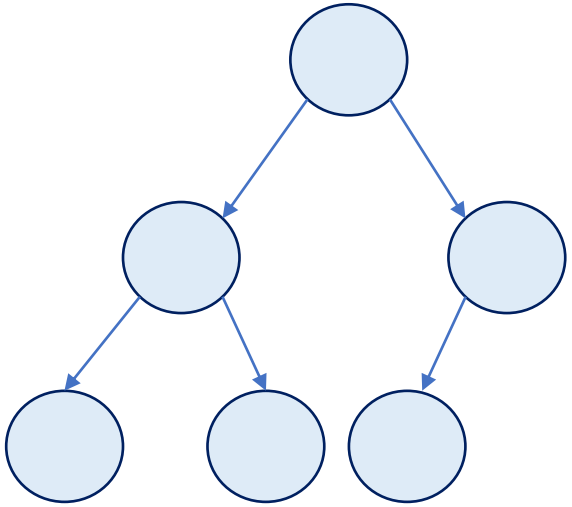


최악의 경우로 저장이 되어있을 때
선형 탐색과 다를게 없음 O(n)

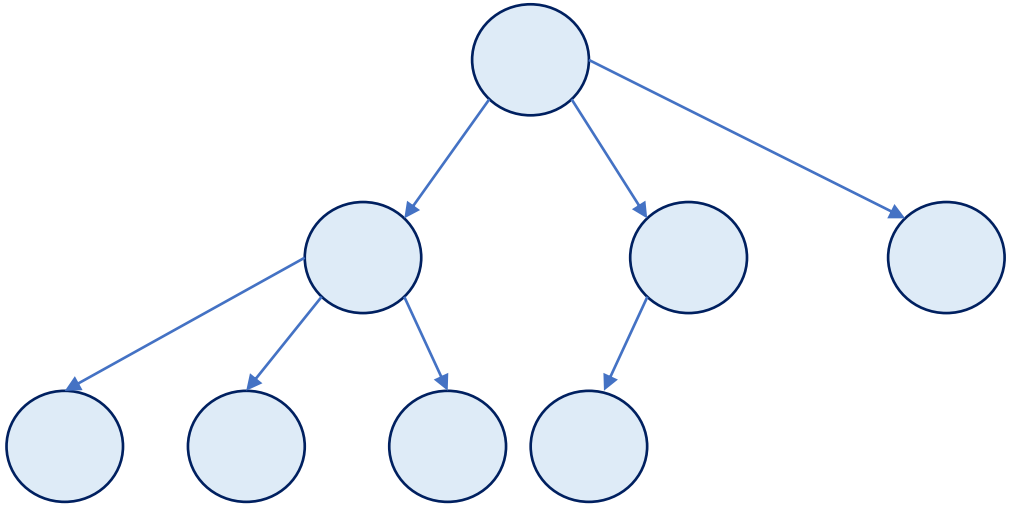
Balanced-TREE 등장

B- TREE

이진트리 VS B-TREE



최대 2개의 자식 노드를 가질 수 있는 자료구조
하나의 노드에 하나의 값만 저장할 수 있는 자료구조
삽입은 루트 노드 에서부터 시작



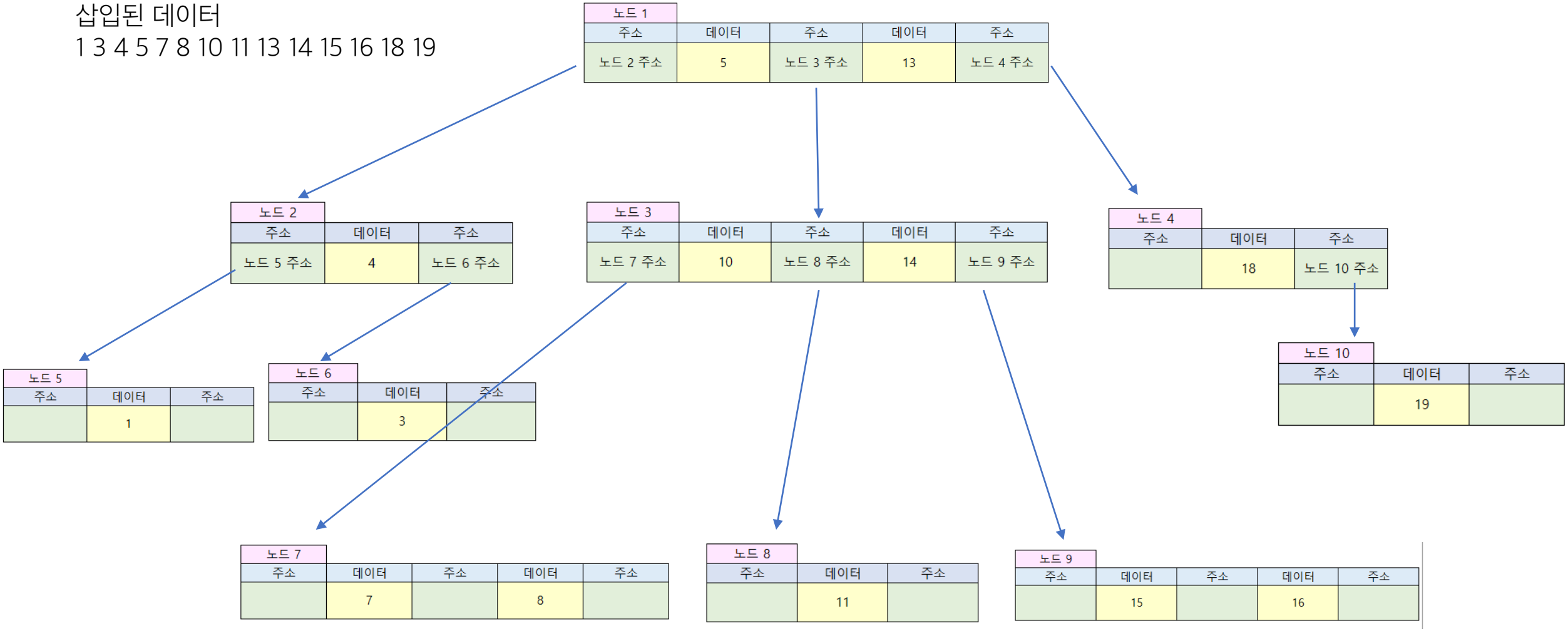
2개 이상의 자식 노드를 가질 수 있는 자료구조
하나의 노드에 여러 개의 값을 저장할 수 있는 자료구조
(3차 B-TREE 구조는 하나의 노드에 자식 노드 3개를 가질 수 있음)
삽입은 리프노드에서 시작

B- TREE

삽입된 데이터
1 3 4 5 7 8 10 11 13 14 15 16 18 19

3차 B-Tree 구조

자식 노드를 3개 가질 수 있고 한 노드에 2개의 데이터를 저장할 수 있다



B- TREE 장점과 단점

시간 복잡도는 최악의 경우에도 O(logN)

균등 트리를 만들기 위해 연산 필요

INDEX 선정 기준?

1. 중복도가 낮은 것
2. 조회가 자주 되는 것
3. 삽입삭제가 빈번하지 않은 것