DB final

- A atomicity 트랜잭션이 완전히 실행되거나 전혀 실행되지 않아야함.
  C consistency 트랜잭션 전후에 DB가 일관되고 유효한 상태를 유지해야함.
  I isolation 동시에 여러트랜잭션이 실행될때, 서로 영향을 주지 않고 독립적으로 처리.
  D durability 트랜잭션 성공후에는 그 결과가 DB에 영구적으로 저장. (지속)

- 데드락 4가지 조건

Mutual exclusion	자원은 한번에 한 트랜잭션 만
Hold and wait	자원을 가지고있으면서 다른 프로세스의 자원까지 기다림
No preemption	다른 프로세스의 자원을 강제로 뺏을수 없음
circular wait	프로세스들이 cycle 형태로 자원을 기다림

- Buffer

버퍼매니저 read : (replace, dirty bit 1이면 디스크 저장) 버퍼에 가져오기
write : (replace, dirty bit 1이면 디스크 저장) 버퍼에 쓰기
commit : flush dirty pages

Pin 하면 replace 제외, LRU (가장오래전에 사용된것 교체)
↕
MRU

- Indexing (search key, pointer)
  - Ordered : 정렬됨, range query에 유리
  - Hash : 해쉬 함수, 버킷이용

ordered index
- clustered (primary) 이것도 정렬됨
- secondary (non clustered) 정렬 안해도 됨
- ISAM 인덱스 순차 파일

Dense index: search key가 가질 수 있는 모든 값에 대해서 인덱스 레코드를 가짐.

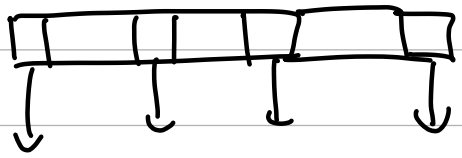
Sparse: 일부키만 인덱스 엔트리 있음

Secondary는 반드시 dense!

multi level index

outer index (Sparse) → inner index → ... (상황에 따라 몇 level 할 수 있어야 함.)

B tree index



N=4

internal: $\lceil N/2 \rceil$ 이상 N 개 의 자식

leaf: $\lceil (N-1)/2 \rceil$ 이상 N-1 개의 값

root: leaf가 아니면 최소 2개 자식

leaf 이면 0이상 N-1 개의 값

트리의 높이 $\lceil \log_{\lceil N/2 \rceil} K \rceil$

delete할때 인터널에서 re distribution 시 상위노드까지 고려!

bottom up : 잎부터 시작 리프 꽉차면 분할 (높이↓, 읽기최적화 쓰기시 split↑)

- hashing

- static hashing (버킷개수 고정)

(closed)

0 □ → □ → □ (over flow chaining)

1 □

2 □ (linear probing)

(open)

- extendable hashing (depth : 해시함수 사용 비트개수)

global > local depth (새버킷 추가)

= (디렉토리 더블링)

- linear hashing

global depth X (디렉토리 더블링 오버헤드 X)

- bit map

~타입스텝프

- 읽기시 W의 타입스탬프가 더 크면 취소
- 쓰기시 R의 타입스탬프나 W의 타입스탬프가 더 크면 취소

Thomas : 쓰기할때 TS < W 이면 쓰기작업 무시.

-OCC