CS3223 Database System Implementation

Yadunand Prem

1. Reference

2. Lecture 1

- intro°
- storage°

2.1. Disk Access Timings

- Seek time: move arm to position disk head on track
- rotation delay: wait to rotate under head
- transfer time: time to read/write data

2.2. Storage Manager

- You are reading / writing data in Blocks (aka pages)

2.2.1. Buffer Manager

- Buffer Pool - Main Memory allocated for DBMS, partitioned into pages called frames
- Clients are the queryplanner, etc.
  - Can request for disk page to be fetched to the buffer
  - Can release a disk page to the buffer
- Page is dirty if it has been modified & not written back to disk
- Variables maintained for each frame
  - Pin Count: Number of clients using the frame (initial: 0)
  - dirty: Is the frame dirty? (initial: false)
- Initial: All frames are free
- When a client requests a page p:
  - Is p in frame f?
    - If yes, increment pin count (aka **pinning**)
    - If no, is free list empty?
      - If no, move some frame f' from free list to buffer pool, $\text{pin count} = 1$, read p into f', return address of frame f
      - If yes, Pick frame f' with pin count = 0 for replacement, set $\text{pin count} = 1$. Is dirty flag of f' = true?
        - If yes, write page in f' to disk
        - read p into f', return address of frame f'
- Buffer Manager replacement policy
  - Random
  - FIFO: First In First Out
  - MRU: Most Recently Used
  - **LRU: Least Recently Used** - Most commonly used
    - Temporal Locality - If a page is accessed, it is likely to be accessed again soon
    - Spatial Locality - If a page is accessed, its neighbours are also likely to be accessed
    - Built with a queue of pointers to frames with $\text{pin count} = 0$
  - Clock: LRU Variant (aka 2nd Chance)
    - Keep track of a referenced bit for each frame
    - When frame's pin count is 0, check referenced bit

    - If 0, replace frame
    - If 1, set referenced bit to 0 and move to end of queue

2.2.2. File

- Abstraction
  - Each relation is a file of records
  - Each record is a tuple, with a RID (Record ID) or TID (Tuple ID) as a unique identifier
  - CRUD operations on file
- Organization: Method of arranging records in a file
  - Heap: Unordered
  - Sorted: Ordered on some search key
  - Hashed: Records are located in blocks via hash fn

2.2.2.1. Heap Impl

- Linked list Impl
  - Problem: FInding a data page requires a linear search, accessing n pages takes $O(n)$ time
- Page Directory Impl
  - Directory of pages, where the directory stores whether the page is free or occupied (1 bit + pointer)
  - 1 page can contain many directory entries. Searching for a page is much faster.
  - Problem: If directory is in main memory, it can be accessed in $O(1)$ time, but if not, it takes $O(n)$ time to access

2.2.2.2. Fix length Page Format

- RID = (Page ID, Slot Number)
- Packed organization: Store records in contiguous slots
  - Invariant: All recors have to be contiguous
  - Problem: If slot is deleted,
    - the records needs to be shifted, which is expensive
    - Not only that, the slot number changes, which requires updating all RIDs
- Unpacked organization: Use bit array to maintain free slots