Storage Manager

April 14, 2023

DBMS architecture

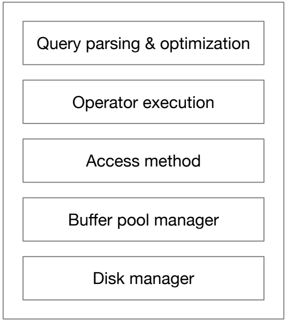


Figure: DBMS architecture

DBMS: Parsing & optimization

Purpose: Parse, check and verify the SQL

```
SELECT name, title
FROM instructor natural join teaches
     natural join course
WHERE dept_name ='Music';
```

And translate into an efficient RA query plan.

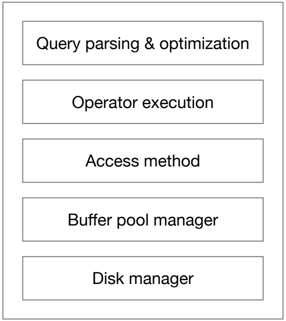
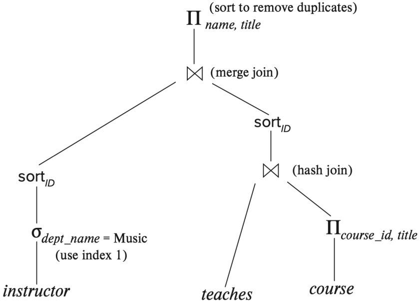


Figure: DBMS architecture

DBMS: Operator execution

Purpose:
Execute a dataflow by operation on tuples and files.



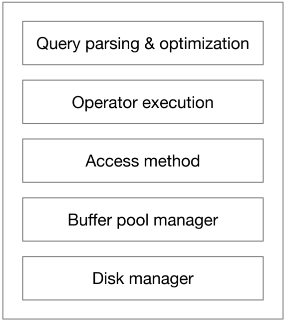


Figure: DBMS architecture

DBMS: Access method

Purpose: Support DBMS's execution engine to read/write data from pages more efficiently.

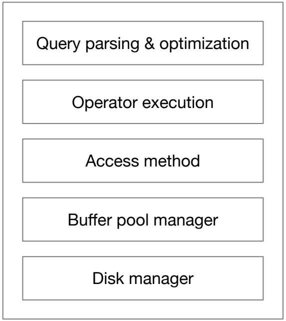
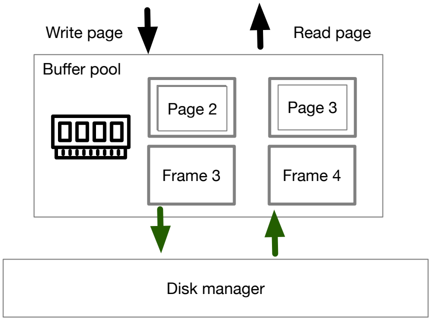


Figure: DBMS architecture

DBMS: Buffer pool manager

Purpose: Provide the illusion of operation in memory.



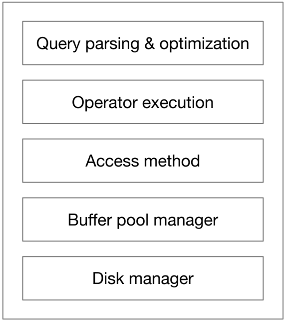
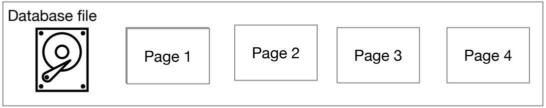


Figure: DBMS architecture

DBMS: Disk manager

Purpose: Manage the database in files on disk.



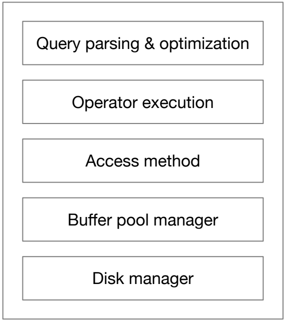


Figure: DBMS architecture

Volatile storage and non-volatile storage

Volatile storage: loses contents when power is switched off

- Example: DRAM, CPU caches
- Random access, byte-addressable

Non-volatile storage: contents persist even when power is switched off

- Example: SSD, HDD, network storage, tap archives
- Sequential access, block-addressable

Storage hierarchy

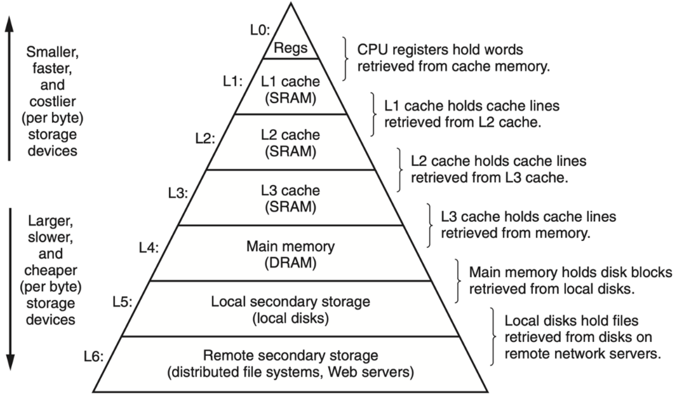


Figure: Storage Hierarchy

Ref. Computer systems: a programmer's perspective

Access time

Access time	Hardware	Scaled time
0.5 ns	L1 Cache	0.5 sec
7 ns	L2 Cache	7 sec
100 ns	DRAM	100 sec
350 ns	NVM	6 min
150 us	SSD	1.7 days
10 ms	HDD	16.5 weeks
1s	Network Storage	11.4 months

Table: Latency comparison numbers

Ref. Latency Numbers Every Programmer Should Know

Disk-oriented DBMS

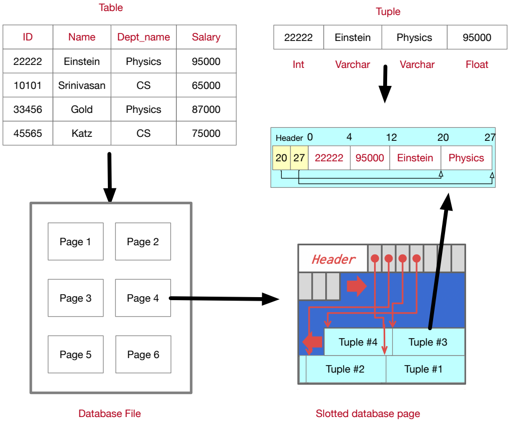
- It's all bout reducing I/O's.
- Cache blocks from non-volatile storage into memory.
- Sequential I/O generally cheaper than random I/O.

Agenda

- Q1: How DBMS represents the database in files on disk?
- Q2: How DBMS manager its memory and move data back-and-forth from disk?

Storage structures

Data storage structures: overview



- Tables are stored as database files.
- Each database file consists of a collection of pages.
- Each page contains a collection of tuples.

Database files

A database file is a collection of pages, each containing a collection of tuples.

- Heap files: tuples placed arbitrarily across pages.
- Sorted files: pages and tuples are in stored order
- Index files: B+ trees, hashing tables and others.

Database heap file

- A heap file is an unordered collection of pages where tuples are stored in random order.
  - Create/Get/Write/Delete pages
  - Must also support iterating over all pages
- Need meta-data to keep track of what pages exist and which ones have free space.
- Two ways to represent a heap file: linked list and page directory.

Heap file via linked list

- Maintain a header page at the beginning of the file that stores two pointers:
  - HEAD of the data page list
  - HEAD of the free page list

- Each page keeps track of the number of free slots in itself.

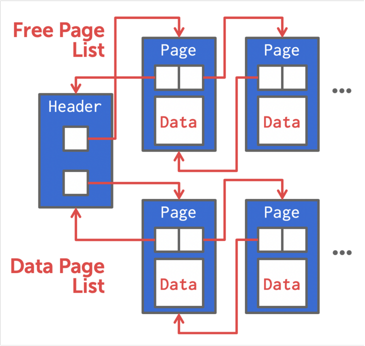


Figure: Linked list

Ref. https://15445.courses.cs.cmu.edu/fall2019

Heap file via page directory

- Maintain special pages called directory pages that tracks the location of data pages in the database files.
- The directory also records the number of free slots per page.
- DBMS has to ensure that the directory pages are in sync with the data pages.

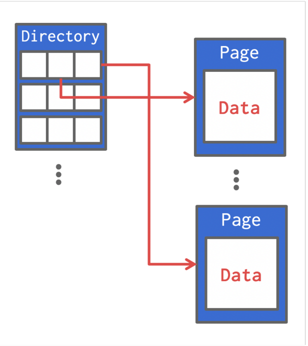


Figure: Heap file via page directory

Ref. https://15445.courses.cs.cmu.edu/fall2019

Database page

A database page is a fixed-sized block of data.

Each page is given a unique identifier.

DBMS uses an indirection layer to map page ids to physical locations.

A page header that contains

- Number of slots/tuples
- Free space
- Checksum
- Transaction visibility

Figure: A database page is a fixed-sized block of data.

Database page issues

1. Record length? Fixed or variable.
2. How to find records by tuple_id?
   - tuple_id = (page_id, location_in_page)
3. How to insert/delete tuples?

Slotted pages

- The most common page layout scheme is called slotted pages.
- The slot array maps slots to the tuples' starting position offsets.
- The header keeps track of
  - The number of used slots
  - The offset of the starting location of the last slot used.

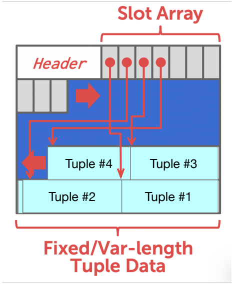
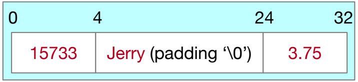


Figure: Slotted page

Ref. https://15445.courses.cs.cmu.edu/fall2019

Tuple layout: fixed length

```
CREATE TABLE foo (
  uid int NOT NULL,
  name char(20),
  gpa float);
```



- All field length and offsets are constant.
  - Computed from schema, sorted in the system catalog.
- System catalog is just another table that stores the metadata for tables.
- What about NULL?
  - Add a bitmap at the beginning of the tuple.

Tuple layout: variable length

```
CREATE TABLE instructor (
    ID int NOT NULL,
    name varchar(20),
    dept_name varchar(20),
    salary float);
```

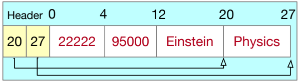


Figure: A tuple with variable length Fields

- Move all variable length fields to end to enable fast access.
- Use an offset array in the tuple header.

Recap

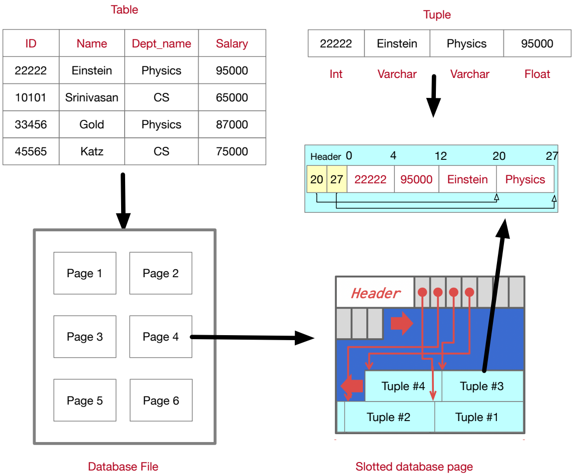


Figure: Data storage structures

Buffer pool manager

Buffer pool

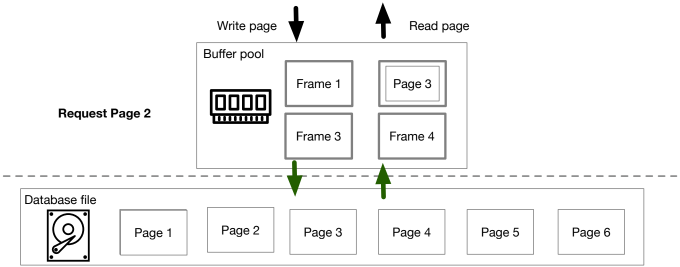


Figure: Buffer pool

Design goal: provide the illusion of operation in memory

- A buffer pool is a memory region organized as an array of fixed-sized pages.
- Each array entry is called a frame.
- When DBMS request a page, an exact copy is placed into one of these frames.

Buffer pool

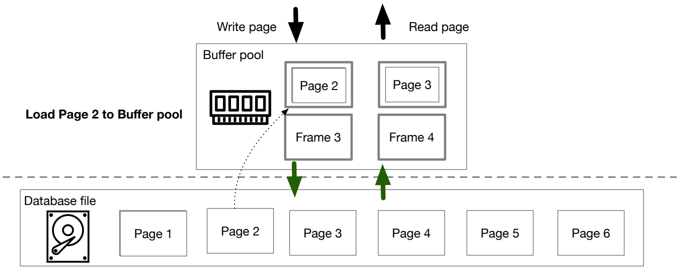


Figure: Buffer pool

Design goal: provide the illusion of operation in memory

- A buffer pool is a memory region organized as an array of fixed-sized pages.
- Each array entry is called a frame.
- When DBMS request a page, an exact copy is placed into one of these frames.

Buffer pool meta-data

Frame ID	Page ID	Dirty Bit	Pin Count
1	2	N	2
2	3	Y	1
3	6	Y	0
4	5	N	0

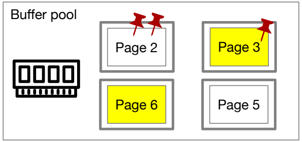


Figure: Buffer pool page table

- The page table keeps track of pages that are currently in memory.
- Also maintains additional meta-data per page.
  - Dirty flag/bit.
  - Pin/reference counter.

Page replacement polices

A page replacement policy decides which page to evict from the buffer pool when the buffer pool is full and a new page is requested.

- Least recently used (LRU)
- CLOCK

LRU

Frame ID	Page ID	Dirty Bit	Pin Count	Last used
1	2	N	2	12
2	3	Y	1	35
3	6	N	0	14
4	5	Y	0	28

Figure: Page 6 will be replaced by LRU

- Track the time of each frame last unpinned (end of use)
- Replace the frame which was least recently used.
- Pined frame: not available to replace.

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

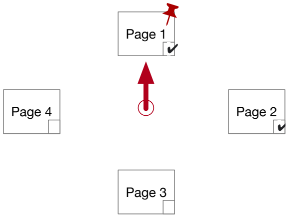


Figure: Skip pinned page

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

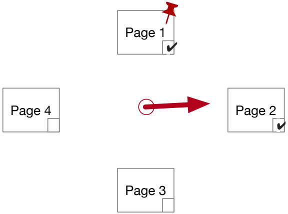


Figure: Clear ref bit

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

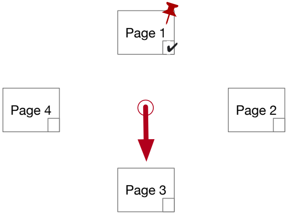


Figure: Replace Page 3 by Page 5

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

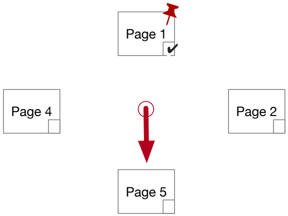


Figure: Set pin count and ref bit

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

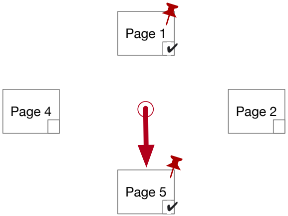


Figure: Advance clock

CLOCK

Approximate LRU without a separate timestamp per page.

- Each page has a reference bit.
- When a page is accessed, set it to 1.

Organized the pages in a circular buffer with a clock hand.

- Upon sweeping, check if a page's bit is 1
- If yes, reset to 0; otherwise evict the page.

Pinned pages are skipped as in LRU.

Example: Request Page 5.

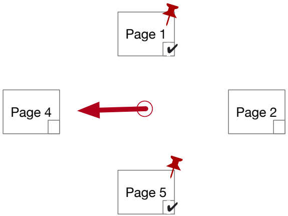


Figure: return

Recap

- Buffer manager provides a level of indirection.
  - Maps disk page IDs to RAM addresses.
  - The illusion of addressing and modifying disk pages in memory.

- Ensures that each requested pages is pinned in RAM.
  - Unpinned by the caller later.

- Page replacement policy aims to minimize caches misses.
  - The access patterns have big impact on I/O cost.