Storage Manager

Spring, 2024

Course overview

Relational databases

- Relational data model ✓
- Relational algebra ✓
- Structured query language ✓
- Relational database design theory ✓

DBMS internals

- Database storage
- Indexing
- Query processing and optimization
- Concurrency control
- Crash recovery

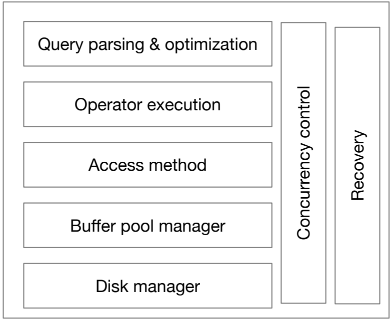


Figure: Classical DBMS architecture

Other topics (TBD): (i) graph database, (ii) parallel query processing

DBMS: Parsing & optimization

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

- Parse, check and verify the SQL query.
- Translate a SQL query into a logical plan.
- Optimization: generate an optimal physical plan.

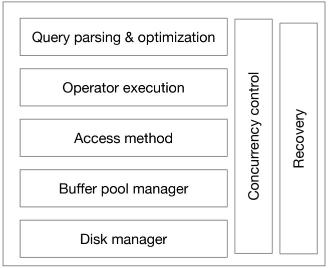
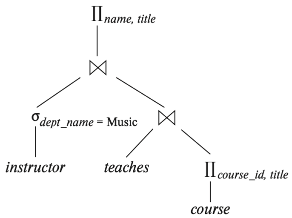
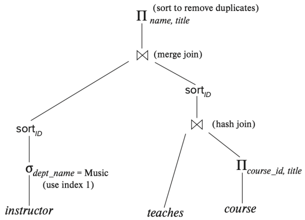


Figure: DBMS architecture

DBMS: Parsing & optimization (cont'd)

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

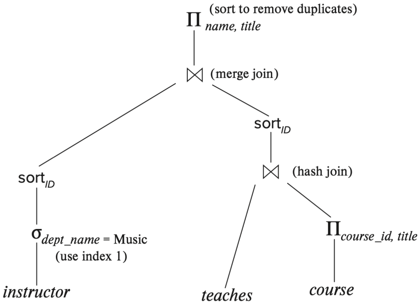




- Each node of a logical plan is a relational operator.
- Each node of a physical plan represents an operator algorithm.

DBMS: operator execution

Execute a dataflow by operating on tuples and files.



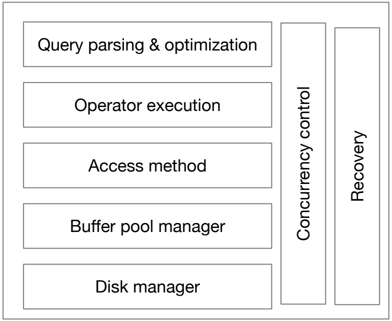


Figure: DBMS architecture

DBMS: Access method

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

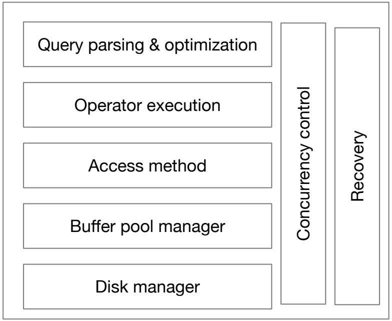
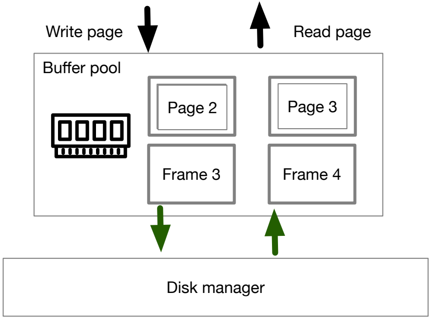


Figure: DBMS architecture

DBMS: Buffer pool manager

Provide the illusion of DBMS operating directly in RAM.



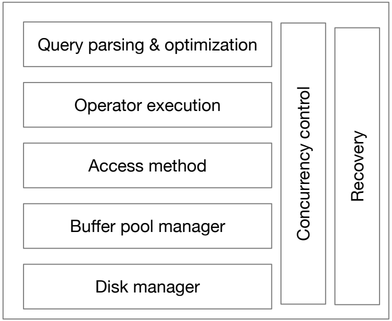
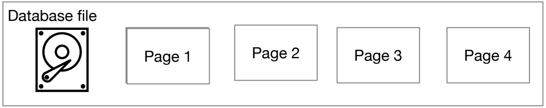


Figure: DBMS architecture

DBMS: Disk manager

Manage the database in files on disk.



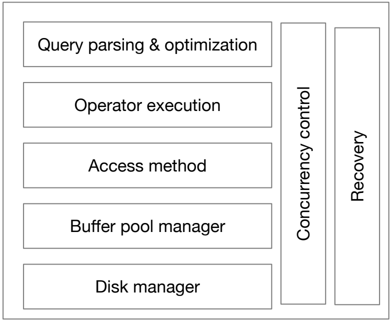


Figure: DBMS architecture

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

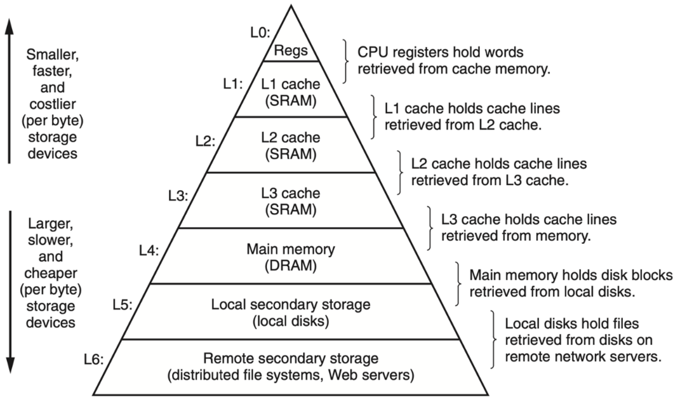


Figure: Storage Hierarchy

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

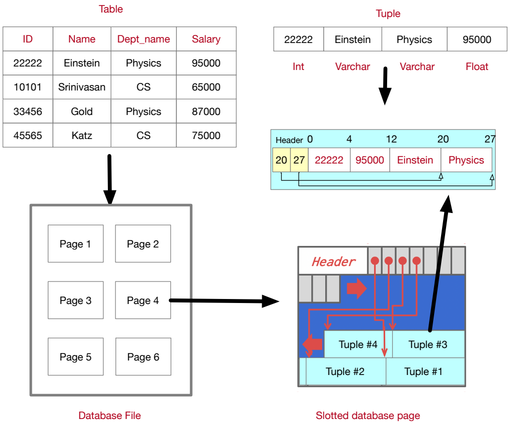
- It's all bout reducing I/O's.
- Cache blocks from non-volatile storage into memory.
- Sequential I/O are generally cheaper than random I/O.

Agenda

- Q1: How does DBMS represent the database in disk files?
- Q2: How does DBMS manage its memory and transfer data to and from the disk?

Storage structures

Data storage structures: overview



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

Database files

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

- Heap files: Tuples are placed arbitrarily across pages.
- Sorted files: Pages and tuples are are stored in a specific 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.
  - Operations: Create/Get/Write/Delete pages.
  - Should support iterating over all pages.
- Require meta-data to track existing pages and identify those with free space.
- Two ways to organize a heap file: linked list and page directory.

Heap file via linked list

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

- Each page tracks the number of free slots in itself.

Question. What happens if we insert a record?

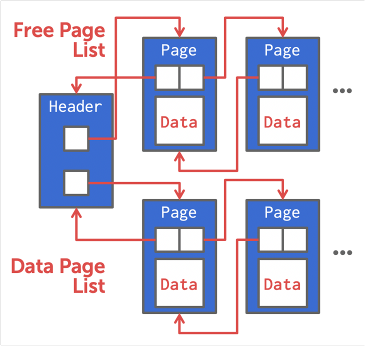


Figure: Heap file via linked list

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

Heap file via page directory

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

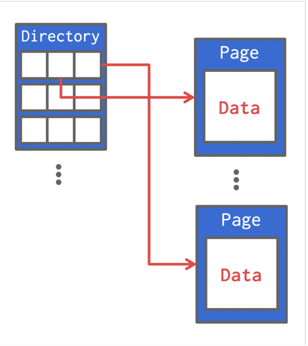


Figure: Heap file via page directory

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

Database page

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

Each page is given a unique page id as its identifier.

A page header that contains

- Number of slots/tuples
- Free space
- Data checksum
- Transaction visibility

Figure: A database page

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

Database page structure

Question. How are tuples organized within a database page?

1. Tuple length: fixed vs. variable.
2. Locating records by tuple_id:
   – tuple_id = (page_id, location_in_page)
3. Insertion and deletion 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.

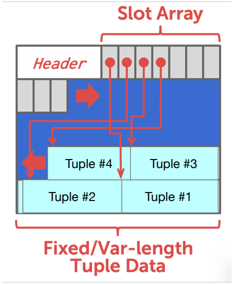
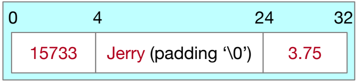


Figure: Slotted page

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

Tuple layout: fixed length

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



- All field lengths and offsets are constant.
  - These are computed from schema and stored in the system catalog.
- The system catalog is just another table that stores the metadata for other tables.
- Handling NULL values:
  - Incorporate a bitmap at the beginning of the tuple for efficient tracking.

Tuple layout: variable length

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

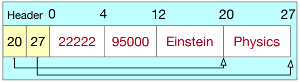


Figure: A tuple with variable length Fields

- Move all variable length fields to end to facilitate fast access.
- Utilize an offset array within the tuple header for efficient navigation.

Recap

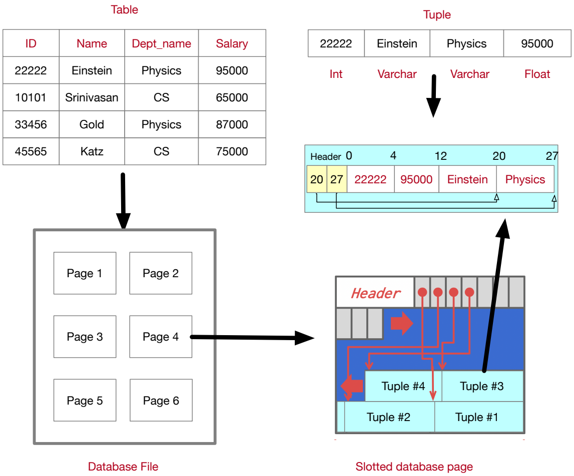


Figure: Data storage structures

Buffer pool management

Buffer pool

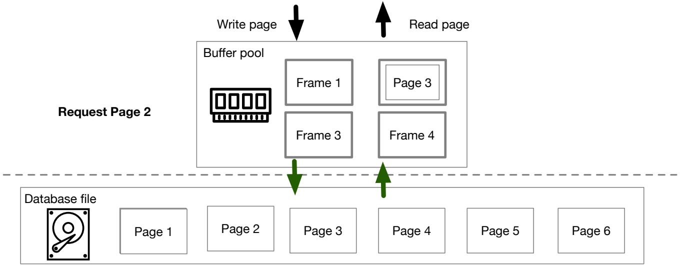


Figure: Buffer pool

Design goal: provide the illusion that the DBMS operates directly 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 the DBMS request a page, an exact copy is retrieved and placed into a frame.

Buffer pool

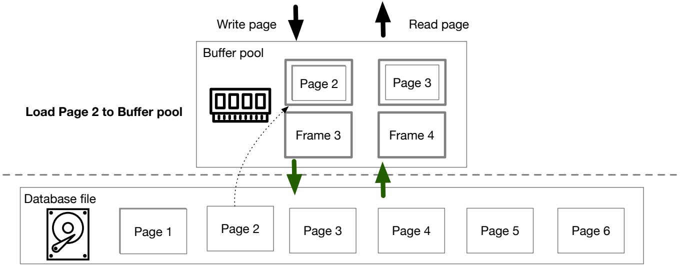


Figure: Buffer pool

Design goal: provide the illusion that the DBMS operates directly 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 the DBMS request a page, an exact copy is retrieved and placed into a frame.

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

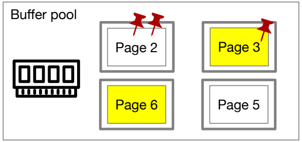


Figure: Buffer pool page table

- The page table tracks pages currently in memory.
- It also maintains additional meta-data per page.
  - Dirty flag/bit.
  - Pin/reference counter.

Page replacement policies

A page replacement policy determines which page to evict when the buffer pool is full, and a new page is needed.

- Least recently used (LRU)
- CLOCK

A page replacement policy aims to minimize caches misses.

LRU policy

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 last unpinned time (end of use) for each frame.
- Replace the least recently used frame.
- Pined frame: not eligible for replacement.
- Good for repeated access to popular pages (temporal locality).

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

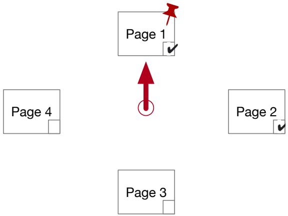


Figure: Skip pinned page

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

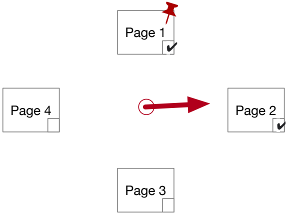


Figure: Clear ref bit

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

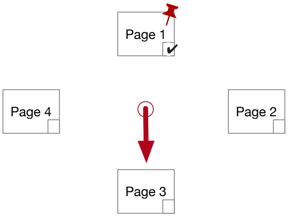


Figure: Replace Page 3 by Page 5

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

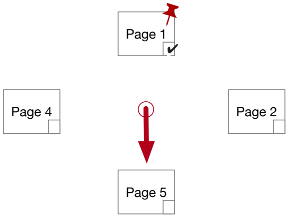


Figure: Set pin count and ref bit

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

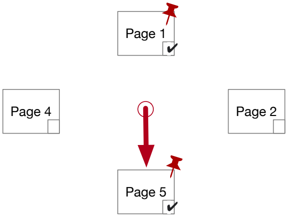


Figure: Advance clock

CLOCK policy

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 set to 1.
- If set, reset to 0; if not, evict the page.

As in LRU, pinned pages are skipped.

Example: Request Page 5.

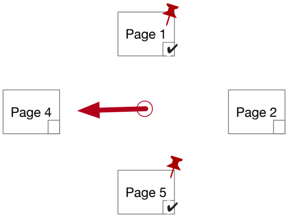


Figure: return

Recap

- Buffer manager provides a level of indirection.
  – Maps disk page id's to RAM addresses.
  – The illusion of addressing and modifying disk pages in memory.
- Page replacement policy aims to minimize caches misses.
  – The access patterns have big impact on I/O cost.