

Lecture 06: Storage

15-445/645 Database Systems (Fall 2017)

Carnegie Mellon University

Prof. Andy Pavlo

"How the DBMS represents the database in files on disk"

Storage

This course is focused on a "disk oriented" DBMS architecture. This means the DBMS assumes the database is on non-volatile disk → you have to copy data from/to disk/mem

Storage follows a hierarchy

- At the very top you have the storage that is closest to the CPU. This is the fastest storage but it is also the smallest and most expensive.
- As you go down the stack you get larger capacities but the storage device is much slower and farther away from the CPU. These devices also get cheaper per GB

• Volatile devices

- Volatile means that if you pull the power from the machine, then the data is lost
- Volatile storage supports fast random access with byte-addressable locations

• Non-Volatile devices

- Non-volatile means that the storage device does not need to be provided continuous power in order for the device to retain the bits that it is storing.
- Non-volatile storage are traditionally better at sequential access (reading multiple chunks of data at the same time) and block addressable

• There is also a new class of storage devices that are coming out soon called "non-volatile memory". These devices are designed to be the best of both worlds: almost as fast as DRAM but with the persistence of disk. We won't cover that in this course.

The DBMS and the OS

• Goals of the DBMS

- Allow the DBMS to manage databases that exceed the amount of memory available
- Reading/writing to disk is expensive, so it must be managed carefully

→ cf. virtual memory. pages & frames.

• You can use mmap to map the contents of a file in a process address space, but if mmap hits a page fault, this will block the process, which is bad if the process held locks to other tuples → range of control ↓

• You never want to use mmap in your DBMS if you need to write

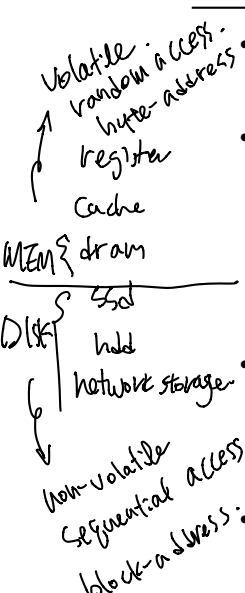
• The DBMS (almost) always wants to control things itself

• The operating system is not your friend

↳ generalized, standardized.

1

↳ (. prefetching .
 . process, thread scheduling .
 . buffer replacement policy, flush data to disk .



cf. non-volatile memory: Nonvolatile + random access. A whole new concept.

File Storage *(Nowadays nobody implements their own filesystem. not portable, hard to implement)*

- In its most basic form, a DBMS is going to store a database as files on disk. Some may use a file hierarchy, others may use a single file
- The OS doesn't know anything about the files, just their existence, and its protections. Only the DBMS knows how to decipher their contents
- The DBMS's storage manager is responsible for managing a database's files. It represents the files as a collection of pages. Also keeps track of reads/writes

Database Pages

- The DBMS organizes the database across one or more files in **fixed-size blocks of data called "pages"**
- Pages can contain different kinds of data (tuples, indexes, etc). Most systems will not mix these types within pages *→ maintenance issues.*
- Some DBMS require pages to be self-contained : *all the metadata included. ex oracle: includes ways to restore data.*
- Each page is given a unique identifier. If your database is a single file, then the page id can just be the offset. Most DBMSs have an indirection layer that keeps maps a page id to a file path and offset.
- There are three concepts of pages in DBMS
 1. Hardware page (usually 4KB)
 2. OS page (4KB)
 3. Database page (1-16KB)
- Each tuple in the database is assigned a unique identifier
 - Most common: page_id + offset/slot
 - An application **cannot** rely on these ids to mean anything *→ it can be changed.*

Database Heap

relational model does not ensure anything about it.

- A heap file is an unordered collection of pages where tuples that are stored in random order
- The DBMS needs a way to find a page on disk given a page_id
- Two Approaches
 1. Linked List: Header page holds pointers to list for free and data pages
 2. Page Directory: DBMS maintains special pages that track locations of data pages

Log Structured File Organization

- Instead of storing tuples, the DBMS only stores log records
- Stores records to file of how the database was modified (insert, update, deletes)
- To read a record, the DBMS scans the log file backwards and recreates the tuple *↷*
- Fast writes, potentially slow reads *periodically compact logs.*

Page Storage Architecture.

- Heap file organization
- Sequential / sorted file organization
- Hashing file organization
- Log-structured file organization.

Page Layout

- Page header: Header records meta-data about the page's contents *↳ "self contained" means header is included in page.*
 - Page size
 - Checksum
 - DBMS version
 - Transaction visibility
 - Some systems require pages to be self-contained (e.g oracle)
- Slotted Pages: Page maps slots to offsets
 - Most common used layout scheme
 - Header keeps track of used slots and offset of starting location of last used slot

Tuple Layout

- A tuple is essentially a sequence of bytes
- It's the job of the DBMS to interpret those bytes into attribute types and values
- Tuple Header: Contains meta data about tuple
 1. Visibility (concurrency control)
 2. Bit Map for NULL values
 3. *Note:* we do not need to store meta-data about the schema of the database here
- Tuple data: Actual data for attributes *↳ Storing all attrs of tuples may not be the best layout for some cases.*
 - Attributes are typically stored in the order that you specify them when you create the table
 - Most DBMS don't allow a tuple to exceed the size of a page
 - ↳ cf. "overflow pages" handle those situations.*

Remark. Relational model does not specify physical storage.