Lecture #03: Database Storage (Part I)

15-445/645 Database Systems (Fall 2018)
https://15445.courses.cs.cmu.edu/fall2018/
Carnegie Mellon University
Prof. Andy Pavlo

1 Storage

This course is focused on a "disk-oriented" DBMS architecture that assumes that primary storage location of the database is on non-volatile disk.

At the top of the storarge hierarchy you have the devices that are closest to the CPU. This is the fastest storage but it is also the smallest and most expensive. The further you get away from the CPU, the storage devices have larger the capacities but are 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.
- For our purposes, we will always refer to this storage class as "memory".

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.
- We will refer to this as "disk" throughout the course. We will not make a (major) distinction between solid-state storage (SSD) or spinning hard drives (HDD).

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 will not cover these devices in this course.

2 DBMS vs. OS

A high-level design goal of the DBMS is to support databases that exceed the amount of memory available. Since reading/writing to disk is expensive, so it must be managed carefully.

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.

- 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.

3 File Storage

In its most basic form, a DBMS stores a database as files on disk. Some may use a file hierarchy, others may use a single file (e.g., SQLite).

The OS does not know anything about the contents of these files. 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.

4 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.

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.

5 Database Heap

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`.

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.

6 Page Layout

Every page includes a header that records meta-data about the page's contents:

There are two approaches to laying out data in pages: (1) tuple-oriented and (2) log-structured.

Slotted Pages: Page maps slots to offsets.

- Most common approach used in DBMSs today.
- Header keeps track of used slots and offset of starting location of last used slot.

Log-Structured: 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.

7 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 information for concurrency control.
2. Bit Map for `NULL` values.
3. Note that we do not need to store meta-data about the schema of the database here.

Tuple Data: Actual data for attributes.

- 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.