CS3223 Database Sys Implementation

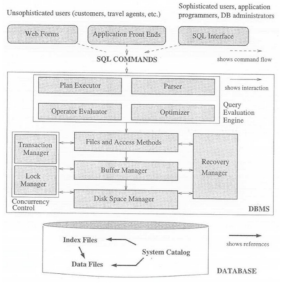
AY23/24 Sem 2, github.com/gerteck

Introduction

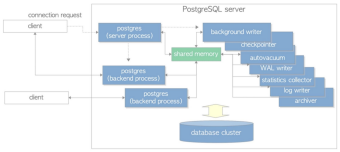
Course Details

- Prerequisite knowledges: CS2040S, CS2102, CS2106 background (helpful).
- Reference Textbook: Raghu & Johannes Database M. Systems, 2002. Encouraged to read ahead based on schedule before the lecture.
- Course covers data structures, algorithms, different components making up database systems.

Architecture of DBMS



Process Architecture in PostgreSQL



Source: H. Suzuki's The Internals of PostgreSQL

Source: Ramakrishnan & Gehrke's Figure 1.3

- OLTP: Online Transaction Processing is a type of data processing that consists of executing a number of transactions occurring concurrently—online banking, shopping, order entry, or sending text messages, for example.
- OLAP: Online Analytical Processing.
- Focusing on centralized database running on a single server.

1. Data Storage

References: R&G Chapt 8. (Storage & Indexing Overview), Chapt 9. (Storing Data: Disks and Files).

A DBMS stores

- Relations (Actual tables)
- System catalog (aka data dictionary) storing metadata about relations. (Relation schemas, structure of relations, constraints, triggers. View definitions, Indexes - derived info to speed up access to relations, Statistical information about relations for use by query optimizer.)
- Log files: Information maintained for data recovery.

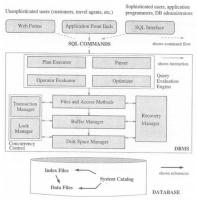
DBMS Storage

Memory Hierarchy: Primary (registers, RAM), secondary (HDD, SSD), tertiary memory with capacity / cost / access speed / volatility tradeoffs.

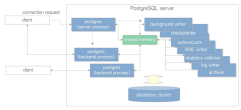
- DBMS stores data on non-volatile disk for persistence.
- DBMS processes data in main memory (RAM).
- Disk access operations (I/O). Read: transfer data from disk to RAM. Write: transfer data from RAM to disk.
- Make use of index to speed up access, so that don't have to retrieve all the data when you run a query. Retrieve index and read only the block that contains specified data. Minimize I/O cost.

Magnetic Hard-Disk Drive HDD

Architecture of DBMS



Process Architecture in PostgreSQL

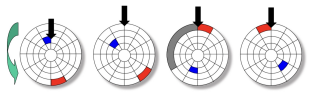


Source: H. Suzuki's The Internals of PostgreSQL

Source: Ramakrishnan & Gehrke's Figure 1.3

- Cylinder, Track, Sector: Units of the HDD storage system. To read from different tracks, need to move the mechanical HDD arm.
- **Disk Access Time**:
  - command processing time: interpreting access command by disk controller.
  - seek time: moving arms to position disk head on track.
  - rotational delay: waiting for block to rotate under head.
  - transfer time: actually moving data to/from disk surface.
  - **access time**: = seek time + rotational delay + transfer time. (CPT considered negligible).

Disk Access Time Components



- Seek time
  - avg. seek time: 5-6 ms
- Rotational delay (or rotational latency)
  - Depends on rotation speed - measured in rotations per minute (RPM)
  - Average rotational delay = time for $\frac{1}{2}$ revolution
  - Example: For 10000 RPM, avg. rotational delay = 0.5 (60 / 10000) = 3 ms
- Transfer time
  - n = number of requested sectors on same track
  - transfer time = $n \times \frac{\text{time for one revolution}}{\text{number of sectors per track}}$
  - avg. sector transfer time: 100-200 µs
- Sequential vs random I/O

Source: R. Burns' slides on storage systems

- **Concept of Sequential vs random I/O.** Sequential: Both sector on same track. Random: Sectors on different track, require seeking (moving arm).
- Given a set of data, we hope to store the data contiguously, on the same track. (Minimize incurring random I/O). If data is too large, store on same track, but different surface (aka same cylinder).
- Complexity hidden to OS by disk controller. Shown as a sequence of memory locations.

Solid-State Drive: SSD

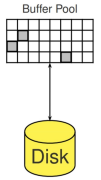
- Build with NAND flash memory without any mechanical moving parts. Lower power consumption.
- **Random I/O**: 100x faster than HDD. (no moving parts)
- **Sequential I/O**: slightly faster than HDD (2x)
- **Disadvantages**: update to a page requires erasure of multiple pages (5ms) before ovewriting page. Limited number of times a page can be erased ($10^5 - 10^6$)

Storage Manager Components

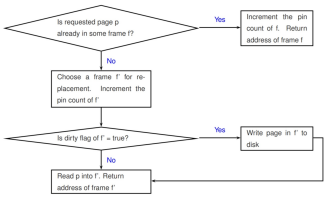
- Data is stored, retrieved in units called **disk blocks (or pages**. (Each block = sequence of one or more contiguous sectors.
- **FIles & access methods layer (aka file layer)** - deals with organization and retrieval of data.
- **Buffer Manager** - controls reading/writing of disk pages.
- **Disk Space Manager** - keeps track of pages used by file layer.

Storage Manager Components

Buffer Manager



BM: Handling request for page p

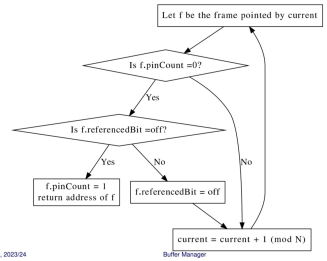


Buffer Manager

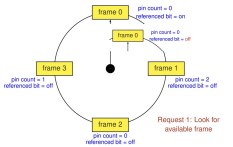
- **Buffer pool**: Main memory allocated for DBMS.
- Buffer pool is partitioned into block-sized pages called **frames**.
- Clients of buffer pool can request for disk page to be fetched into buffer pool, release a disk page in buffer pool.
- A page in the buffer is **dirty** if it has been modified & not updated on disk.
- **Two variables** maintained for each frame in buffer pool:
  - **pin count**: number of clients using page (initialized 0)
  - **dirty flag**: whether page is dirty (initialized false)
- Free list: Keeps track of frames that are free / empty.
- **Pin count**: Incrementing pin count is **pinning** the requested page in its frame. Decrementing is **unpinning** the page.
  - Unpinning a page, dirty flag should be updated to true if page is dirty.
  - A page in buffer can be replaced only when pin count is 0.
  - Before replacing buffer page, needs to be written back to disk if its dirty flag is true.
- Buffer manager coordinates with transaction manager to ensure data correctness and recoverability.
- **Replacement Policies**
  - Replacement policy: Deciding which unpinned page to replace. (some examples:)
  - Random, FIFO, Most Recently Used (MRU), Least Recently Used (LRU): (Use queue of pointers to frames with pin count = 0), most common, makes use of temporal locality.
  - **Clock**: cheaper popular variant of LRU
    - **current** variable: points to some buffer frame.
    - Each frame has a **referenced bit**, turns on when its pin count turns 0.
    - Replace a page that has referenced bit off & pin count = 0.

Clock Replacement Policy

N = number of frames in buffer pool



Clock Replacement Policy: Example



Files

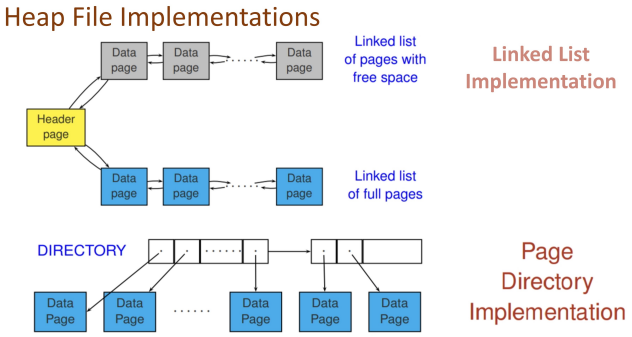
File Abstraction

- Each relation is a file of records.
- Each record has a unique record identifier called RID / TID.
- Common file operations: create/delete file, insert record, delete/get record with given RID, scan all records.

File Organization: Method of arranging data records in a file that is stored on disk.

- **Heap file**: Unordered file
- **Sorted file**: Records order on some search key.
- **Hashed file**: Records located in blocks via a hash function.

Heap File Implementations



- **Linked list implementation**: Two linked lists, one with pages with free space, other of completely full pages.
- **Page Directory Implementation**: Two leveled implementation. Each big block is a disk block with some metadata. Each disk block has a number of data pages.