Fall 2004 CS 186 Discussion Section

Notes - Week 2 ending 9/10

Topics:

- Administrivia
- Relational Model
- Storing Data - Disks & Files
- Q&A on Homework 1

I. ADMINISTRIVIA

- Re-announce DIS section hrs, office hrs, contact info.
- Use newsgroup ucb.class.cs186
- About Homework 1
    - Deadlines, etc.
    - Q & A - Issues, clarifications, etc.

II. RELATIONAL MODEL

- Relational database is a collection of relations.
- Relations are collection of tuples (records).
- All tuples of a relation have same set of domains (columns).
- Cardinality (# of rows), Airity/Degree (# of columns).
- SQL: Language for operating a relational database
    - Data Definition Language (DDL)
    - CREATE TABLE, DROP TABLE, etc.
    - Data Manipulation Language (DML)
    - INSERT, DELETE, UPDATE, SELECT, etc.
    - Integrity Constraints
    - PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, etc.

II. STORING DATA - DISKS & FILES

- Main memory - Random access, volatile.
- Tapes - Sequential access, persistent.
- Disks - Random + sequential, persistent.

A) Disks

- Simple picture of disk.
    - Tracks, cylinders, sectors, blocks.
- Seek time
    - Time to move the disk heads to a specific track.
- Rotational delay
    - Time required for the platter to rotate until the block is under the disk head, on average half a rotation.
- Transfer time
    - Actual time it takes to transfer the data in the block.
- Access time = Seek time + Rotational time + Transfer time.
- Typically Seek time > Rotational time > Transfer time.
- How to reduce access time for two records needed frequently?

- Arrange them "closely" for sequential access.
- Decreasing order of closeness:
    - Same block, adj block, same track, same cylinder, adj cylinder.

B) Disk Space Management

- Lowest layer of DBMS
- Manages space on disk.
- Can use OS functionality or manage on its own.
- Many DBMSs manage disk space on their own.
    - DBMS can predict page reference patterns and optimize page access.
    - Circumvents file size limit imposed by OS.
    - Can make files span several devices.
    - Crash recovery, an OS can be lazy in writing to disk.
    - Portability across several OS.

C) Buffer Management

- One layer above Disk Space Management.
- Manages reading/writing pages between disk and main memory.
- Buffer pool
    - Available main memory divided into pages called frames.
    - Load disk pages into available frames.
    - Bookkeeping – for each frame, store page id, dirty bit, pin count.
    - Dirty bit: ON if page is modified, OFF otherwise.
    - Pin count: Number of current users of the page.
- Pinning – Incrementing the pin_count for a requested page.
- Unpinning – Decrementing the pin_count for a released page.
- Typically, Number of memory frames is far < number of pages on disk.
- Page replacement
    - If all frames are occupied, choose a frame with pin_count==0 and replace.
    - If (dirty_bit(frame_to_replace) == true) write it to disk
    - Read new page from the disk into the chosen frame.
- Page replacement policy – which frame with pin_count==0 to choose?

LRU - Least Recently Used.

- Keep track of the order in which pages reach pin_count = 0, and choose the one that was unpinned first for replacement.
- We can implement this using a queue, by adding pages to the end of the queue when the page is a candidate for replacement, and choosing the head of the queue for replacement.
- Sequential flooding – repeated sequential accesses can cause all reads to result in misses, if the data cannot fit in the buffer pool.

MRU - Most Recently Used.

- Implement this by using a stack, and adding pages to the top of the stack when they become available for replacement.

CLOCK – An approximation of LRU.

- For each frame, we keep track of a second chance bit, which is set to 1 when the pin_count reaches 0. For SIMPLECLOCK, do not keep track of a second chance bit (ref_bit).
- Below is pseudo code for CLOCK – this is only the pseudo code for the replacement part of the strategy.

```
frames = # of frames in buffer pool
counter = 0
pool = array of frames
function chooseframe () { // returns the frame number to replace
      while true {
            frame = pool[count]
            if (frame.pin_count == 0) {
                  if (frame.ref_bit == 0) {
                        replace = count
                        count = count++ % frames
                        return replace
                  }
                  else { // give this page a second chance
                        frame.ref_bit = 0
                        count = count++ % frames
                  }
            }
            else { // the page is pinned
                  count = count++ % frames
            }
      }
}
```