CS522 Advanced Database Systems
Data Storage

Chengyu Sun
California State University, Los Angeles

The Megatron 3000 Approach

- Text file
- One tuple per line
- Fields are separated by #

  e.g. /usr/db/students

  Smith#123#CS
  Jones#522#EE


The Real DBMS Approach

- File Manager
- Buffer Manager
- Disk Manager


Record Request

- Which page is record r in?
- Where is page n on disk?
- Is page n in memory?

  Execution Engine
  File Manager
  Disk Manager
  Buffer Manager


Why So Many "Managers"?

- Isn't the OS supposed to handle file I/O??
- Don't we have both OS and disk buffering already??

Disk Manager

- Abstracts storage as a sequence of pages
  - allocate, de-allocate
  - read, write
- Keeping track of free disk blocks
  - Linked list
  - Bitmap directory
Buffer Manager
- Buffer pool
  - pages in the buffer pool are called frames
- Replacement policy
- Additional information per frame
  - pin-count
  - dirty

How Buffer Manager Works
- If page in buffer pool, increment pin-count of the frame; otherwise
  - Choose a frame for replacement
  - Write out the frame if it’s “dirty”
  - Read in frame
  - Increment pin-count of the frame
- Return frame

A Few Finer Points
- When do we increment/decrement pin-count??
- What if all frames are pinned??
- Concurrency and recovery??

Replacement Policies
- Optimal??
- FIFO
- Random
- Least Recent Used (LRU)
- Most Recent Used (MRU)
  - Sequential flooding
- Clock

LRU Replacement Implementation
- A queue of pointers to frames with 0 pin count
  - Add a pointer to a frame to the queue when the pin count of the queue reaches 0

Clock Replacement Implementation
- Current frame
- Per frame
  - pin-count
  - referenced
More Replacement Strategies
- Buffer pool partition
  - bind a partition to a database, or relation, or index
  - apply different replacement strategies
- Considering more parameters
  - page type
- Higher level controls
  - Love/hate
  - Buffer reservation

Project 1 and Accounts
- CS Server accounts
- Turnin Server accounts
- Minibase discussion

Data Organization
```
<table>
<thead>
<tr>
<th>field</th>
<th>record</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>&quot;File&quot;</td>
</tr>
</tbody>
</table>
```

Heap File
- Unsorted
  - why??
- Operations
  - create and delete files
  - iterate through all records (scan)
  - insert a record
  - get, update, and delete a record with a given id

Fields for Common SQL Types
- char(n), varchar(n), bit(n)
- int, float, decimal(n,m)
- date, time
- text, BLOB

Fixed-Length Records
- Fixed number of fields
- Fixed length of each field
System Catalog

- For each relation
  - name, file name, file structure
  - attribute name and type
  - index name
  - constraints
- Index and view information
- Accounts and privileges
- Statistics

System Catalog as a Collection of Relations

<table>
<thead>
<tr>
<th>attr_name</th>
<th>rel_name</th>
<th>type</th>
<th>position</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr_name</td>
<td>SC_Attributes</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>rel_name</td>
<td>SC_Attributes</td>
<td>String</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>SC_Attributes</td>
<td>String</td>
<td>3</td>
</tr>
<tr>
<td>position</td>
<td>SC_Attributes</td>
<td>Integer</td>
<td>4</td>
</tr>
<tr>
<td>sid</td>
<td>Students</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>sname</td>
<td>Students</td>
<td>String</td>
<td>2</td>
</tr>
<tr>
<td>gpa</td>
<td>Students</td>
<td>Real</td>
<td>3</td>
</tr>
<tr>
<td>cid</td>
<td>Courses</td>
<td>String</td>
<td>1</td>
</tr>
<tr>
<td>cname</td>
<td>Courses</td>
<td>String</td>
<td>2</td>
</tr>
</tbody>
</table>

Variable-Length Records

| F1 | F2 | F3 | F4 |

Organize Records into Pages

- Record ID (rid)
  - <page id, slot number>
- Concerns
  - insert, delete, update, and search records in a page

Fixed-length Records – Packed Page

```
slot 1
slot 2
        ...
slot n
header
```

Fixed-length Records – Unpacked Page

```
slot 1
slot 2
        ...
header
```
Variable-Length Records – Slotted Page

From Pages to Files

Other File Organizations – Sorted File

Other File Organizations – Hashed File

A Simple Cost Model

About Slotted Page

Delete, insert, update, search??

Does the <page id, slot number> scheme still work?

Sorted file (Sequential file)
  - Key
    - multi-field key??
    - Usually packed
  - Maintenance – almost sorted
    - Sliding
    - Overflow page
    - Sequential page

Hashed file
  - H(key) → primary page
  - overflow page

The model
  - B data pages
  - R records per page
  - Avg time to read or write a disk page: D
  - Avg time to process a record: C
  - Time to compute the hash value of a record: H

The method – the complexity of a DB algorithm can usually be estimated by its number of disk page accesses

The problem(s)??
R&D Methodology
- Simplified model
- Extensive experimentation

Three File Organizations
- Heap file
- Sorted file
- Hashed file

Five Operations
- Scan
- Equality selection
- Range selection
- Insert
- Delete

Cost Estimations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Heap</th>
<th>Sorted</th>
<th>Hashed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan</td>
<td>BD</td>
<td>BD</td>
<td>BD</td>
</tr>
<tr>
<td>E. Selection</td>
<td>0.5BD</td>
<td>DlogB</td>
<td>D</td>
</tr>
<tr>
<td>R. Selection</td>
<td>BD</td>
<td>DlogB</td>
<td>BD</td>
</tr>
<tr>
<td>Insert</td>
<td>2D</td>
<td>Search + BD</td>
<td>2D</td>
</tr>
<tr>
<td>Delete</td>
<td>Search+D</td>
<td>Search + BD</td>
<td>Search + D</td>
</tr>
</tbody>
</table>

Readings
- Stanford book: Chapter 12
- [Wisconsin book: Chapter 7, 8]
- [PAX paper]