ICS 321 Fall 2009
Storage & Indexing

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa
Application View of DBMS

- DBMS holds data in the form of relations or tables
- A table is a bag of tuples or records
- SQL is used to manage and query the data
- Data stored in a DBMS is **persistent**
Data Storage

- **Main Memory**
  - Random access
  - Volatile
- **Flash Memory**
  - Random access
  - Random writes are expensive
- **Disk**
  - Random access
  - Sequential access cheaper
- **Tapes**
  - Only sequential access
  - Archiving
Relational Tables on Disk

- **Record** -- a tuple or row of a relational table
- **RIDs** – record identifiers that uniquely identify a record across memory and disk
- **Page** – a collection of records that is the unit of transfer between memory and disk
- **Bufferpool** – a piece of memory used to cache data and index pages.
- **Buffer Manager** – a component of a DBMS that manages the pages in memory
- **Disk Space Manager** – a component of a DBMS that manages pages on disk
Magnetic Disks

- A disk or platter contains multiple concentric rings called **tracks**.
- Tracks of a fixed diameter of a spindle of disks form a **cylinder**.
- Each track is divided into fixed sized **sectors** (i.e., “arcs”).
- Data stored in units of disk **blocks** (in multiples of sectors)
- An array of **disk heads** moves as a single unit.
- **Seek time**: time to move disk heads over the required track
- **Rotational delay**: time for desired sector to rotate under the disk head.
- **Transfer time**: time to actually read/write the data
Accessing Data on Disk

- **Seek time**: time to move disk heads over the required track
- **Rotational delay**: time for desired sector to rotate under the disk head.
  - Assume uniform distribution, on average time for half a rotation
- **Transfer time**: time to actually read/write the data
Example: Barracuda 1TB HDD (ST31000528AS)

• What is the average time to read 2048 bytes of data?

= Seek time + rotational latency + transfer time

= 8.5 msec + 4.16 msec + (2048 / 512) / 63 * (60000 msec / 7200 rpm)

= 8.5 + 4.16 + 0.265

<table>
<thead>
<tr>
<th>cylinders</th>
<th>121601</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes/cylinder</td>
<td>16065*512</td>
</tr>
<tr>
<td>Blocks/cylinder</td>
<td>8029</td>
</tr>
<tr>
<td>Sectors/track</td>
<td>63</td>
</tr>
<tr>
<td>Heads</td>
<td>255</td>
</tr>
<tr>
<td>Sprindle Speed</td>
<td>7200 rpm</td>
</tr>
<tr>
<td>Average Latency</td>
<td>4.16 msec</td>
</tr>
<tr>
<td>Random read seek time</td>
<td>&lt; 8.5 msec</td>
</tr>
<tr>
<td>Random read Write time</td>
<td>&lt; 9.5 msec</td>
</tr>
</tbody>
</table>
File Organizations

How do we organize records in a file?

• **Heap files**: records not in any particular order
  – Good for scans

• **Sorted files**: records sorted by particular fields
  – Scans in the sorted order or range scans in the sorted order

• **Indexes**: Data structures to organize records via trees or hashing.
  – Like sorted files, they speed up searches for a subset of records, based on values in certain (“search key”) fields
  – Updates are much faster than in sorted files
Comparing File Organizations

Consider an employee table with search key <age,sal>

- Scans: fetch all records in the file
- Point queries: find all employees who are 30 years old
- Range queries: find all employees aged above 65.
- Insert a record.
- Delete a record given its RID.
Simple Evaluation Model

- B : number of data pages
- R : number of records per page
- D : average time to read/write a disk page
  - From previous calculations, if a page is 2K bytes, D is about 13 milliseconds
- C : average time to process a record
  - For the 1 Ghz processors we have today, assuming it takes 100 cycles, C is about 100 nanoseconds