Overview

- Disk-based DBMS
  - Assumptions and goals
  - Improving performance
- Memory-based DBMS
  - Assumptions and goals
  - Applications
- Performance Comparisons
- Architecture
  - Simplified
  - Data Structures
**Disk-based DBMS**

- **Assumptions**
  - Data is disc resident
    - Algorithm design
    - Buffering
    - Indexing methods
- **Goal**
  - Minimise
    - Disc I/O (slow)
      - Add programming logic instead of I/O access
### One Solution - RAM Disc

- **RAM**
  - Looks like a disc, acts like a disc
  - Only exists in memory
    - Very popular in older DOS systems
- **Traditional DBMS can store data in RAM Disc**
  - Improves disc access
  - But
    - Caching overhead cannot be switch off
    - Algorithms assume disc based operation

### Main Memory DBMS

- **Assumptions**
  - Data is memory resident
  - Direct access to data
  - Simpler coding, algorithms, structure
- **Goal**
  - Reduce space requirements
    - Data Compression
    - Smaller programs
  - Simplify algorithms
    - Eliminate all disc I/O
Applications

- Embedded Systems
  - Database is part of the application
  - Not a separate server
- Examples
  - Network Routers
  - Set-Top Boxes (Digital Boxes)
  - Mobile technologies
    - PDAs
    - Phones
    - iPod
      - Not memory, hard disk

Simplified Architecture

- Caching
  - Major performance overhead
    - Managing cached data
  - Main memory database has no cache
    - All data is available in memory
- Data Transfer
  - Traditional databases copy data to memory
    - One copy on disc
    - One copy in cache
    - One copy in program memory
  - Main memory database points to data
    - Only one copy of the data
### Simplified Architecture

#### Query Optimisation

- **Disk-based**
  - Reducing disc input/output
  - Assume worst case
    - All data is disc resident
  - What if all the data is in memory buffers?
    - Algorithms are still disc oriented
    - Assume data could be disc based

- **Memory-based**
  - All data in memory
    - No memory buffers
  - Execution plans simpler
    - No disc access
    - Sorts in main memory
  - Less need for indexes

#### Transaction Processing

- **Disk-based**
  - Logs transactions
  - Write to disc on commit
- **Memory-based**
  - “Before Image” lists
    - Store old objects until commit
    - On commit, throw away “before image”
    - On rollback, copy “before image” into database
Performance Comparisons

Ref: http://www.mcobject.com

Performance Comparison

Ref: http://www.timesten.com/products/perfbrf.html
What if the machine fails?

- Data lost when machine fails
  - Acceptable for some applications, eg
    - Set-top box
      - Reloads programme information when switched on
    - PDA
      - Synchronises with PC when rebooted
    - Routers
      - Rebuilds router tables
    - Test systems
      - Hsqldb
        - main memory JDBC database
  - Data can be saved to disc at set intervals
Index Structures Still Important

Trash the Traditional DBMS!

- Largest Relational DBMS
  - Terabytes
- Moving towards Petabytes

<table>
<thead>
<tr>
<th>Database Size - BBF - TB</th>
<th>Database Size - Terabytes</th>
<th>Database Size - Petabytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. UnitedHealthcare</td>
<td>$20,000,000</td>
<td>$5,000,000</td>
</tr>
<tr>
<td>2. Everest</td>
<td>$5,000,000</td>
<td>$1,250,000</td>
</tr>
<tr>
<td>3. Aetna</td>
<td>$15,000,000</td>
<td>$3,750,000</td>
</tr>
<tr>
<td>4. Aetna</td>
<td>$15,000,000</td>
<td>$3,750,000</td>
</tr>
<tr>
<td>5. Kaiser</td>
<td>$10,000,000</td>
<td>$2,500,000</td>
</tr>
<tr>
<td>6. UnitedHealthcare</td>
<td>$5,000,000</td>
<td>$1,250,000</td>
</tr>
<tr>
<td>7. HealthCare Corporation</td>
<td>$10,000,000</td>
<td>$2,500,000</td>
</tr>
<tr>
<td>8. BlueCrossBlueShield</td>
<td>$15,000,000</td>
<td>$3,750,000</td>
</tr>
<tr>
<td>9. Cigna</td>
<td>$20,000,000</td>
<td>$5,000,000</td>
</tr>
</tbody>
</table>

- Largest Memory-resident DBMS
  - 64GB