Isn’t Implementing a Database System Simple?

Relations ➞ Statements ➞ Results
Introducing the MEGATRON 3000
Database Management System

- The latest from Megatron Labs
- Incorporates latest relational technology
- UNIX compatible

Megatron 3000 Implementation Details

! First sign non-disclosure agreement !
Megatron 3000 Implementation Details

• Relations stored in files (ASCII)
  e.g., relation R is in /usr/db/R

Smith # 123 # CS
Jones # 522 # EE

• Directory file (ASCII) in /usr/db/directory

R1 # A # INT # B # STR ...
R2 # C # STR # A # INT ...
  ...
Megatron 3000 Sample Sessions

% MEGATRON3000
   Welcome to MEGATRON 3000!
&
...
& quit
%

Megatron 3000 Sample Sessions

& select *
   from R #

   Relation R
   A   B   C
   SMITH 123  CS

&
& select A,B
from R,S
where R.A = S.A and S.C > 100 #

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>CAR</td>
</tr>
<tr>
<td>522</td>
<td>CAT</td>
</tr>
</tbody>
</table>

Result sent to LPR (printer).
& select *
from R
where R.A < 100 | T #

New relation T created.

• To execute “select * from R where condition”:
  1. Read dictionary to get R attributes
  2. Read R file, for each line:
     (a) Check condition
     (b) If OK, display
To execute “select * from R where condition | T”:

1. Process select as before
2. Write results to new file T
3. Append new line to dictionary

To execute “select A,B from R,S where condition”:

1. Read dictionary to get R,S attributes
2. Read R file, for each line:
   (a) Read S file, for each line:
      (i) Create join tuple
      (ii) Check condition
      (iii) Display if OK
What's wrong with the Megatron 3000 DBMS?

- Tuple layout on disk
  - Change string from 'Cat' to 'Cats' and we have to rewrite file
  - ASCII storage is expensive
  - Deletions are expensive
- Search expensive; no indexes
  - Cannot find tuple with given key quickly
  - Always have to read full relation

What's wrong with the Megatron 3000 DBMS?

- No buffer manager
  - Need caching
- Brute force query processing
  - select *
    from R,S
    where R.A = S.A and S.B > 1000
  - Do select first?
  - More efficient join?
What’s wrong with the Megatron 3000 DBMS?

• No concurrency control
• No reliability
  – Can lose data
  – Can leave operations half done
• No security
  – File system insecure
  – File system security is coarse

What’s wrong with the Megatron 3000 DBMS?

• No application program interface (API)
  – How can a payroll program get at the data?
• No GUI
• Cannot interact with other DBMSs.
• Poor dictionary facilities
  – How do we know what is in the database?
• Lousy salesman!!
What do we need to know?

- **File & System Structure**
  Records in blocks, dictionary, buffer management,…

- **Indexing & Hashing**
  B-Trees, hashing,…

- **Query Processing**
  Query costs, join strategies,…

- **Crash Recovery**
  Failures, stable storage,…

---

What do we need to know?

- **Concurrency Control**
  Correctness, locks,…

- **Transaction Processing**
  Logs, deadlocks,…

- **Security & Integrity**
  Authorization, encryption,…

- **Distributed Databases**
  Interoperation, distributed recovery,…
System Structure

- Strategy Selector
- Query Parser
- User
- User Transaction
- Transaction Manager
- Concurrency Control
- Buffer Manager
- Recovery Manager
- Lock Table
- File Manager
- M.M. Buffer
- Log

Statistical Data
- Indexes
- User Data
- System Data

Hardware Constraints

- Hardware
- DBMS
- Data Storage
Typical Computer

Processor
Fast, slow, reduced instruction set, with cache, pipelined...
Speed: 100 → 500 → 1000 MIPS

Memory
Fast, slow, non-volatile, read-only,...
Access time: 10^{-6} → 10^{-9} sec.
1 \mu s → 1 ns
Secondary storage
Many flavors:
- Disk: Floppy (hard, soft)
  Removable Packs
  Winchester
  Ram disks
  Optical, CD-ROM...
  Arrays
- Tape: Reel, cartridge
  Robots

Focus on: “Typical Disk”

Terms: Platter, Head, Actuator
  Cylinder, Track
  Sector (physical),
  Block (logical), Gap
"Typical" Numbers

Diameter: 1 inch → 15 inches
Cylinders: 100 → 2000
Surfaces: 1 (CDs) →
(Tracks/cyl) 2 (floppies) → 30
Sector Size: 512B → 50K
Capacity: 360 KB (old floppy) → TB
Disk Access Time

I want block X in memory

Time = Seek Time + Rotational Delay + Transfer Time + Other
Seek Time

Average Random Seek Time

\[ S = \frac{\sum_{i=1}^{N} \sum_{j=1\atop j\neq i}^{N} \text{SEEKTIME}(i \rightarrow j)}{N(N-1)} \]

“Typical” S: 5 ms → 10 ms
Rotational Delay

Head Here

Block I Want

Average Rotational Delay

\[ R = \frac{1}{2} \text{ revolution} \]

“typical” \[ R = 2 \text{ ms (15000 RPM)} \]
Complication

• May have to wait for start of track before we can read desired block

Transfer Rate: \( t \)

• “typical” \( t: 1 \rightarrow 3 \text{ MB/second} \)
• transfer time: \[ \text{block size} \quad t \]
Other Delays

- CPU time to issue I/O
- Contention for controller
- Contention for bus, memory

“Typical” Value: 0

So far: Random Block Access
What about: Reading “Next” block?
If we do things right (e.g., Double Buffer, Stagger Blocks...)

Time to get = Block Size + Negligible
t

- skip gap
- switch track
- once in a while, next cylinder

Rule of Thumb
Random I/O: Expensive
Sequential I/O: Much less

• Ex: 1 KB Block
  » Random I/O: ~ 10 ms.
  » Sequential I/O: ~ 1 ms.
Cost for **Writing** similar to **Reading**

.... unless we want to verify!

need to add (full) rotation + **Block size**

\[ t \]

---

**To Modify a Block?**

**To Modify Block:**

(a) Read Block
(b) Modify in Memory
(c) Write Block
[(d) Verify?]
Block Address:

- Physical Device
- Cylinder #
- Surface #
- Sector

Complication: Bad Blocks

- Messy to handle
- May map via software to integer sequence

1
2
. . m

Map → Actual Block Addresses