Notes 1: Introduction to DBMS Implementation

Yousef M. Elmehdwi
Department of Computer Science
Illinois Institute of Technology
yelmehdwi@iit.edu

August 20th, 2019

Slides: adapted from a course taught by Hector Garcia-Molina, Stanford
Core Terminology Review

- **Data**
  - any information worth preserving, most likely in electronic form

- **Database**
  - a collection of data, organized for access and modification, preserved over a long period.

- **Query**
  - an operation that extracts specified data from the database.

- **Relation**
  - an organization of data into a two-dimensional table, where rows (tuples) represent basic entities or facts of some sort, and columns (attributes) represent properties of those entities.

- **Schema**
  - a description of the structure of the data in a database, often called “metadata”

- **Database Management System (DBMS)**
  - system software for creating and managing databases. It provides users and programmers with a systematic way to create, retrieve, update and manage data.
Advanced Database Organization?

- Database Implementation
- How to implement a database system
- and have fun doing it ;-)

3 / 31
What do you want from a DBMS?

- Keep data around (persistent)
- Answer questions (queries) about data
- Update data
Isn’t Implementing a Database System Simple?

- Relation $\Rightarrow$ Statements $\Rightarrow$ Results
Introduction the Megatron 3000 Database Management System

- “Imaginary” database System
- The latest from Megatron Labs
- Incorporates latest relational technology
- UNIX compatible
- Lightweight & cheap!
Megatron 3000 Implementation Details

- Megatron 3000 uses the file system to store its relations.
- Relations stored in files (ASCII)
  - e.g., relation Students(name, id, dept) is in /usr/db/Students
  - Values of components of a tuple are stored as a character string, separated by special marker character #

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>#</td>
<td>123</td>
<td>#</td>
</tr>
<tr>
<td>Jonson</td>
<td>#</td>
<td>522</td>
<td>#</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
The database schema is stored in a special file

Schema file (ASCII) in /usr/db/schema

<table>
<thead>
<tr>
<th>Students</th>
<th>name</th>
<th>STR</th>
<th>id</th>
<th>INT</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depts</td>
<td>C</td>
<td>STR</td>
<td>A</td>
<td>INT</td>
<td></td>
</tr>
</tbody>
</table>
Megatron 3000 Implementation Details

Students

Depts

relations

domains/types

attributes/columns
% MEGATRON3000

Welcome to MEGATRON 3000!

&

:

:

& quit

%
Megatron 3000 Sample Sessions

```sql
& select * from Students #

Relation Students

name    id    dept
Smith   123    CS
Johnson 522    EE
...
&
```

- A # ends a query
Megatron 3000 Sample Sessions

- Execute a query and send the result to printer

```sql
& select *
    from Students | LPR #
&
```

- Result sent to LPR (printer).
Execute a query and store the result in a new file

```
& select *
    from Students
    where id < 100 | LowId #
&
```

New relation LowId created.
To execute

\textbf{SELECT \textastern FROM R WHERE <condition>}

1. Read schema to get attributes of R
2. Check validity of condition
3. Display attributes of R as the header
4. Read file R; for each line:
   a. Check condition
   b. If TRUE, 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 usr/db/schema
Consider a more complicated query, one involving a join of two example relations \( R, S \)

To execute

\[
\text{SELECT } A,B \text{ FROM } R,S \text{ WHERE } <\text{condition}> 
\]

1. Read schema to get \( R,S \) attributes
2. Read \( R \) file, for each line \( r \):
   - Read \( S \) file, for each line \( s \):
     1. Create join tuple \( r & s \)
     2. Check condition
     3. If TRUE, Display \( r,s[A,B] \)
What’s wrong with MEGATRON 3000 DBMS?

- DBMS is not implemented like our “imaginary” MEGATRON 3000
- Described implementation is inadequate for applications involving significant amount of data or multiple users of data
- Partial list of problems follows
What’s wrong with MEGATRON 3000 DBMS?

- Tuple layout on disk is inadequate with no flexibility when the database is modified.
- e.g., change String from *CS* to *CSDept* in one *Students* tuple, we have to rewrite the entire file.
  - ASCII storage is expensive
  - Deletions are expensive
What’s wrong with MEGATRON 3000 DBMS?

- Search expensive; no indexes
  - e.g., cannot find tuple with given key quickly
  - Always have to read full relation
What’s wrong with MEGATRON 3000 DBMS?

- Brute force query processing
- e.g.,

```
SELECT * FROM R, S WHERE R.A = S.A and S.B > 1000
```

- Much better if use index to select tuples that satisfy condition (Do select using S.B >1000 first)
- More efficient join (sort both relations on A and merge)
What’s wrong with MEGATRON 3000 DBMS?

- No buffer manager
  - There is no way for useful data to be buffered in main memory; all data comes off the disk, all the time
  - e.g., need caching.
What’s wrong with MEGATRON 3000 DBMS?

- No concurrency control
  - Several users can modify a file at the same time with unpredictable results.
What’s wrong with MEGATRON 3000 DBMS?

- No reliability
- e.g., in case of error/crash, say, power failure or leave operations half done
  - Can lose data
What’s wrong with MEGATRON 3000 DBMS?

- No security
- e.g., file system security is coarse
  - Unable to restrict access, say, to some fields of a relation and not others
What’s wrong with MEGATRON 3000 DBMS?

- No application program interface (API)
  - e.g., how can a payroll program get at the data?
What’s wrong with MEGATRON 3000 DBMS?

- Cannot interact with other DBMSs.
What’s wrong with MEGATRON 3000 DBMS?

- No GUI
Introduce students to better way of building a database management systems.
Reading assignment

- Refresh your memory about basics of the relational model and SQL
  - from your earlier course notes
  - from some textbook
  - [http://cs.iit.edu/~cs425/schedule.html](http://cs.iit.edu/~cs425/schedule.html)
Course Blackboard: Assignments & Projects\Reading subfolder
  Chapter 1: “Introduction to DBMS Implementation”
Notes 2: Hardware