DS 1300 - Introduction to Database Systems

Based on slides by Dan Suciu
Adapted by Michael Hahsler

MY NEW PRODUCT IS A DATABASE OF FAMOUS SERIAL KILLERS.

YOU CAN SEARCH THE DATABASE BY NAME, WEAPON OR TATTOO.

LET ME GUESS, WALLY: SIX MONTHS AGO OUR YOUNG INTERN ASKED YOU WHAT THE TERM “KILLER APPLICATION” MEANT.
Database

What is a database?

• Physical storage: A collection of files storing related data.
• Logical: A collection of tables (or objects).

Examples of databases

• Accounts database; payroll database; SMU’s students database; Amazon’s products database; airline reservation database.
Database Management System

What is a DBMS?
• A complicated (and often expensive) piece of software typically running on a large (remote) server written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time.

Examples of DBMS
• Commercial: DB2 (IBM), SQL Server (MS), Oracle, Sybase
• Open Source: MySQL, Postgres, SQLite, ...
• Big Data: often NoSQL like MongoDB, Apache Cassandra, etc.
Architecture: Using a DMBS

“Client-server Architecture”

- Data files
- Database server running the DBMS
- Applications running a client

connection (ODBC, JDBC)
Operations: Query/Update

Assume we have a database for movies and actors.

- Simple query:
  In what year was ‘Star Wars’ produced?

- Multi-table query:
  Find all movies with ‘Harrison Ford’
  (combine actor and movie tables)

- Complex query:
  For each actor, count her/his movies

- Updating
  Insert a new movie;
  add an actor to a movie; etc
Operations: Query/Update

- **Files** (e.g., CSV)
  - Simple queries

- **Spreadsheets**
  - Multi-table queries (maybe)

- **DBMS**
  - All

Updates: generally OK
Change the Structure of a DB

Add *Address* to each Actor

- Files (e.g., CSV) → *Very hard*
- Spreadsheets → *Yes*
- DBMS → *Yes*
Issue: Concurrent Access

Multiple users access/update the data concurrently

• What can go wrong?
  Lost update; resulting in inconsistent data

• How do we protect against that in OS?
  Locks

• Databases need a similar concept to deal with concurrent updates.
Issue: Recover from crashes

- Transfer $100 from account #4662 to #7199:

  \[
  X = \text{Read}(\text{Accounts}, 4662);
  X.\text{amount} = X.\text{amount} - 100;
  \text{Write}(\text{Accounts}, 4662, X);
  \]

  \[
  Y = \text{Read}(\text{Accounts}, 7199);
  Y.\text{amount} = Y.\text{amount} + 100;
  \text{Write}(\text{Accounts}, 7199, Y);
  \]

CRASH!

What is the problem?
Concurrency & Recovery: Transactions

- A transaction = sequence of statements that either all succeed, or all fail together.
- E.g., Transfer $100

```
BEGIN TRANSACTION;
UPDATE Accounts
SET amount = amount - 100
WHERE number = 4662

UPDATE Accounts
SET amount = amount + 100
WHERE number = 7199

COMMIT
```
Transactions

Transactions have the **ACID** properties:

- **A** = atomicity
- **C** = consistency
- **I** = isolation
- **D** = durability

All or nothing

Valid state to valid state

Transactions are independent

No data loss after commit

Transactions also allow rollbacks (undo).
Relational Data Base  
= Collection of Tables

<table>
<thead>
<tr>
<th>id</th>
<th>fName</th>
<th>lName</th>
<th>mid</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>15901</td>
<td>Harrison</td>
<td>Ford</td>
<td>15901</td>
<td>Star Wars</td>
<td>1977</td>
</tr>
<tr>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
<td>. . .</td>
</tr>
</tbody>
</table>
Create/Store Large Datasets

Use SQL to create and populate tables:

\[
\text{CREATE TABLE Actors ( fName CHAR(30), lName CHAR(30), . . . )}
\]

\[
\text{INSERT INTO Actors VALUES('Harrison', 'Ford', . . . )}
\]

Physical organization of the data is handled by DBMS
We focus on modeling the database!
Querying

• Find all movies with ‘Harrison Ford’

```sql
SELECT title
FROM Movies, Actors, Movie_Actors
WHERE Actors.lname = 'Ford' and Actors.fname = 'Harrison' and Movies.mid = Movie_Actors.mid and Movie_Actors.id = Actors.id
```

• What happens behind the scene?
  The DBMS uses indices and optimizes automatically the query...
Change the Structure of a Table

Add *Address* to each Actor

```sql
ALTER TABLE Actor
ADD address CHAR(50)
DEFAULT 'unknown'
```
What comes next?

1) Using a DBMS
2) Using SQL to Query Databases
3) Designing a Database