DASC / CSE 5300

Module III

Sharma Chakravarthy
Information **T**echnology **Lab**oratory (IT Lab)
Computer Science and Engineering Department
The University of Texas at Arlington, Arlington, TX 76019
Email: sharmac@cse.uta.edu
URL: http://itlab.uta.edu/sharma

Analyzing Disk-Resident Data

In Module III

- I will cover the following with examples so you can practice further on your own
  - Disk-based large amounts of data
  - Difference between analyzing data that can fit in main memory and data that cannot
  - Implications of the above
  - Disk-base data can be in
    - A DBMS (most likely in a Relational DBMS)
    - Data warehouse
    - NoSQL systems
    - Plain old file
    - Spread sheet, etc.
- Analysis of data stored in a DBMS

Main memory Algorithms

- Most algorithms (including those used for analysis) are developed as main memory algorithms to begin with
  - All Python packages are Main memory algorithms
  - Load the entire data and let loose the algorithm
  - This was fine when we had smaller data (of course, it is relative)
    - Small meaning whatever fits in main memory
- Data – whether small or large – need to be stored on persistent devices (disks, USB, DVD etc.)
- How do you manage to query and analyze data when it does not fit in main memory

Dealing with Disk-based Data

- It is not easy if you have to deal with it yourself
- To handle disk-resident data
  - Need to build a buffer manager
  - Need to bring in data as needed for computation
  - Remove data from buffer when no longer needed
- The good news is that we are not modifying any data during analysis
- But still, this staging of data in and out is quite complicated
- Your focus should be on analysis, not implementation of a buffer manager!

Example

- Suppose you are computing an aggregate value
  - Sum, average, standard deviation, etc.
- Suppose data size is more than memory available
- What is the solution?
  - Bring (or stage) data from disk from beginning
  - Compute partial result
  - Bring more data; continue computing partial results
  - Until there is no more data
- Output the result when all data has been used
- Not all computations can be done this way
- This is a simple scenario
- Multiple users may be accessing and computing on the same shared data

Database Management Systems (DBMSs)

- This is where DBMSs come into picture (think of Mymav)
- Also, most corporate/enterprise data is stored in a DBMS
  - Oracle, DB2, SQLServer, Sybase (Relational)
  - Mongo DB, Neo4J, etc. (NoSQL systems)
  - Logs, web crawls, etc. (stored in files)
- You need to be able to do the analysis irrespective of where the data is stored or comes from
- Some of the above systems (e.g., Relational) support extensive query and analysis functions
- DBMiner supports mining on data stored in a relational DBMS (DB2)

Data Warehouses

- Limitations of RDBMS for complex analysis was realized more than 20 years ago
- SQL (or structured Query Language) was enhanced to included multi-dimensional analysis
- These were typically used on Data warehouses
- What is a data warehouse?
  - A database which collects (or accumulates) required data from multiple operational databases for the purpose of analysis and report generation
  - It has its own schema and contents are kept fresh (or current) using various schemes
    - Star and snowflake schemas

DBMS Alternatives

- DBMSs were available since 1970's
- Before that we were using only file systems for storing data
- Hierarchical and Network databases were available from the 70's before the advent of Relational DBMSs
- RDBMSs came about during late 70's and early 80's
- Were used in mission-critical applications
  - Banks, financial institutions
  - Airlines
  - Employee data
  - Corporate data, …

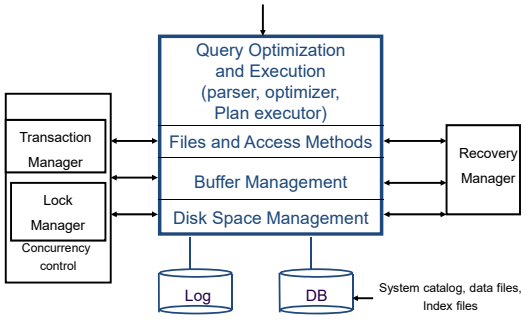
Alternatives

- Today, data can be stored in newer NoSQL systems (MongoDB, Neo4J, …)
- Map/Reduce paradigm is also available to perform distributed data processing (or analysis)
  - These can scale very well
- But, algorithms have to be developed or mapped into this paradigm
- Map/Reduce is a useful alternative if you do not want to create a schema and load data into a DBMS to be managed
  - DBMSs and Map/Reduce serve different purposes

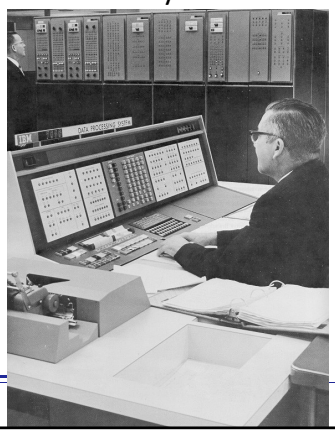
Relational DBMSs

DBMS Architecture

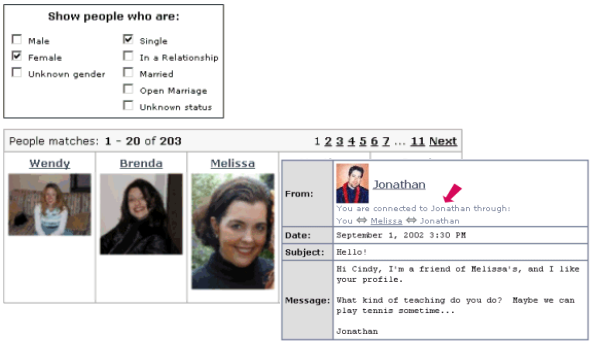
Our focus: SQL commands from applications



Database Systems: Then



Database Systems: Today



From Friendster.com on-line tour

Other Ways Databases Make Life Better?

- "Players could finally sign up for the Star Wars Galaxies game week as Sony opened up registration to the public."
- "Once players got in to game they found the game servers offline because of database problems."
- "Some players spent hours tuning their in-game characters only to find that crashes deleted all their hard work."
- Source: BBC News Online, July 1, 2003.

Other databases you may use

Is the WWW a DBMS?

- Fairly sophisticated search available
  - crawler *indexes* pages on the web
  - Keyword-based search for pages
- But, currently
  - data is mostly unstructured and untyped
  - search only:
    - can't modify the data
    - can't get summaries, complex combinations of data
  - few guarantees provided for freshness of data, consistency across data items, fault tolerance, …
  - Web sites typically have a DBMS in the **background** to provide these functions.
- The picture is changing
  - New standards e.g., XML, Semantic Web can help data modeling
  - Research groups (e.g., at Berkeley) are working on providing some of this functionality *across multiple web sites*.

Is a File System a DBMS?

- Thought Experiment 1:
  - You and your project partner are editing the same file.
  - You both save it at the same time.
  - Whose changes survive?

A) Yours B) Partner's C) Both D) Neither E) ???

- **Thought Experiment 2:**
  - –You're updating a file.
  - –The power goes out.
  - –Which of your changes survive?

Q: How do you write programs over a subsystem when it promises you only "???" ?
A: Very, very carefully!!

A) All B) None C) All Since Last Save D) ???

Why Use a DBMS?

- Data independence and efficient access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.
- Persistence, scalability, portability

Why Study Databases??

- Shift from *computation* to
  information management
  Data analytics/science
  - at the "low end": ad hoc data, spread sheet
  - at the "high end": scientific data management
- Datasets increasing in diversity and volume (4V's)
  - Digital libraries, interactive video, Human Genome project, EOS project, multi-media dbms
  - … need for DBMS is exploding
- DBMS study encompasses all aspects of CS
  - OS, languages, theory, AI, multimedia, logic

Current Commercial Outlook

- A major part of the software industry:
  - Oracle, IBM, Microsoft, Sybase
  - also Informix (now IBM), Teradata
  - smaller players: java-based dbms, devices, OO, …
- Well-known benchmarks (esp. TPC)
- Lots of related industries
  - data warehouse, document management, storage, backup, reporting, business intelligence, app integration
- Relational products dominant and evolving
  - adapting for extensibility (user-defined types), adding native XML support.
- Open Source coming on strong
  - MySQL, PostgreSQL, BerkeleyDB, MongoDB

What's the intellectual content?

- representing information
  - data modeling
- languages and systems for querying data
  - complex queries with real *semantics**
  - over massive data sets
- *concurrency control* for data manipulation
  - controlling concurrent access
  - ensuring *transactional semantics*
- reliable data storage
  - maintain data semantics even if you pull the plug

* semantics: the meaning or relationship of meanings of a sign or set of signs

Impedance mismatch

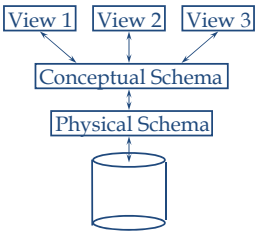
- If you are studying DBMS, you have to understand this clearly
- CPU is way faster than disk access. In fact, it is a Million or more times faster
  - CPU accesses are in nano seconds and disk accesses are still in milli seconds!
- Because of this speed difference, it takes much longer to bring data from a disk into memory
- So, CPU is waiting/idling most of the times for data (if not architected properly)!
- This is called impedance mismatch!

Data Models

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using the a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

Levels of Abstraction

- Many *views*,
- single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



Schemas are defined using DDL; data is modified/queried using DML.

Example: University Database

- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Data Independence

- Applications insulated from how data is structured and stored.
- *Logical data independence*: Protection from changes in *logical* structure of data.
- *Physical data independence*: Protection from changes in *physical* structure of data.

One of the most important benefits of using a DBMS!

DBMS Components

- Architecture – typically client/server
- DDL, DML parsing, processing
- Physical DB design (files, indexing, access methods)
- Query processing, optimization
- Buffer management
- Concurrency control
- Recovery
- Utilities – backup, archiving, exporting, OLAP, report generator, physical design wizards, access to multiple DBMSs, designer tools, tuning of parameters, etc. etc.

Why Study the Relational Model?

- Most widely used model.
  - Vendors: IBM (DB2, Informix), Microsoft, Oracle, Sybase, etc.
- "Legacy systems" in older models
  - e.g., IBM's IMS fastpath
  - Network models are not used today
- More recently, NoSQL systems
  - Based on different data models
  - Key-store, document store, graph
  - Tradeoffs in functionality ACID vs. CAP (or BASE)

Competitors

- Earlier Competitor: Object-Oriented model
  - ObjectStore, Versant, Ontos
  - a synthesis: *object-relational model*
    - Informix Universal Server, UniSQL, O2, Oracle
- More recent competitors: Map/reduce, NoSQL
  - Works on unstructured data
  - Useful for one-time processing as opposed to data management
  - Overhead is low, however, programming level is high
  - Less expensive, use of commodity machines!!
  - Does not store and manage data

Relational Database: Definitions

- *Relational database:* a set of *relations*.
- *Relation:* made up of 2 parts:
  - *Instance* : a *table*, with rows and columns. #rows = *cardinality*, #fields = *degree or arity*
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.g. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
- Can think of a relation as a *set* of rows or *tuples*. (i.e., all rows are distinct)

Creating Relations (schema) in SQL

- Creates the Students relation. Observe that the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students
(sid: CHAR(20),
name: CHAR(20),
login: CHAR(10),
age: INTEGER,
gpa: REAL)
```

- As another example, the Enrolled table holds information about courses that students take.

```
CREATE TABLE Enrolled
(sid: CHAR(20),
cid: CHAR(20),
grade: CHAR(2))
```

Example Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Cardinality = 3, degree = 5 , all rows distinct
- Do all column names in a relation schema have to be distinct?

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

- *Powerful variants of these commands are available; more later!*

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
  - The key: precise semantics for relational queries.
  - Allows the optimizer to extensively re-order operations, and still ensure that the answer does not change.
- This is in contrast to earlier (hierarchical and network) DBMSs where queries had to be written as programs

The SQL Query Language

- The most widely used relational query language. Current standard is SQL-2011.
- To find all 18 year old students whose gpa is greater than 3.1, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18 AND gpa > 3.1;
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified by the DBA, based on application semantics. DBMS checks for violations.
  - Two important ICs: primary and foreign keys
  - In addition, we *always* have domain constraints.
- Powerful and natural query languages exist.

Relational Model: Summary (contd.)

- ACID properties are important
  - Atomicity, consistency, isolation, and durability
- Concurrency control and Recovery together guarantee ACID properties

Other data models

- The older hierarchical and network models are not used much (although they are in use)
- However, several non-relational data models have been proposed and currently used in specific contexts
- They are called NoSQL DBMSs
  - Key-value store
  - Document
  - graph

Oracle usage

- Log into Omega
- Type sqlplus {username{/password}}
- Prompts for username and password if not given in the above statement
- You can now create tables, insert tuples, drop tables, create views, and run SQL queries in your account
- You can write embedded sql programs, stored procedures, and connect to the database by jdbc or other mechanisms

Oracle Usage (contd.)

SQL> Select *
From cat; //note the semicolon

- Lists all the tables and views defined by the current user.

SQL>Describe sharmac.employee // no semicolon

- Describes the details (attributes and types) of the employee table in the database sharmac
- help command
  - Command is the name for which you want to get help!
  - E.g., SQL>help column

Oracle Usage (contd.)

- You can either enter sql commands at the prompt or put it in a file and submit it.
- If you put it in a file, use the command SQL>@<filename> or start <filename> to execute statements in the file.
- If the file name has .sql extension, you need not specify the extension
- Since you are experimenting, please use a file that you can change and try again!
- Command prompt interfaces for databases are pretty bad!
- The editor is even worse!
- In the shell do
- stty erase ^V <backspace>
  - This will allow you to backspace while writing SQL!

Oracle Usage (contd.)

SQL> Set echo on
SQL> set echo off / will not echo on the command prompt
Sets echo on to echo all the input; especially important if you are submitting a file

SQL> Spool <filename>
SQL> spool off

- Is useful to redirect your output to a file for analysis. Created in the current directory.
- You need to set spool off to see its contents.
- Subsequent runs will append to the same file!

Oracle (contd.)

SQL> host <command>
example: SQL> host ls *.sql

- Executes the command on the host machine. Avoids coming out of sql to issue Unix commands on Omega!
- I Suggest you have 2 windows/logins, one for editing and one for running sql!
- SQL> set linesize 80
- Sets the size of each line to 80; can be increased

Other Useful commands

$wc -l <filename> gives the number of lines in that file
example: $ wc -l imdb-dsc.sql

SQL> column column_name format a15
Uses 15 spaces for each column output

- SQL> ---------- is the comment in SQL ----------

Use that to include English queries and any other comment you want to include

- SQL> set pagesize 0

Will not generate blank lines after each page of output

IMDb Database (International Movies)

- This database is large (similar to a real-world) database
  - Has 7 very large tables
- Has all movies and TV episodes produced since 1920's (last 100 years)
- Has information on movie titles, year of production, director, actors, genre, writers, etc.
- For TV episodes, each episode, seasons, end year of the series, actors, directors, genre etc.
- Has some null values as well.
- Strings are used for attribute values. Have to be careful in how you specify conditions

IMDb Database Schema (or tables)

- title.basics
- name.basics
- title.akas
- title.crew
- title.episode // for TV series
- title.principals
- title.ratings

Project 3 will be on this database

- Show the schema and sample data
- We will ask you to write
  - 2 to 3 aggregate queries (,ulti-line SQL queries with most of the clauses)
  - Aggregate queries are those that use sum, max, min, avg, count and use group by and having clauses
- We May ask you to write
  - 2 to 3 analysis queries
  - Using ROLL UP and CUBE operators

Questions/comments

For more information visit:
http://itlab.uta.edu