Course Introduction & Foundational Concepts

CPS 352: Database Systems

Simon Miner
Gordon College
Last Revised: 1/14/15

Agenda

- Introductions
- Course Syllabus
- Databases
  - Why
  - What
  - Terminology and Concepts
- Design Project

Introductions

- Who are you? (name, background, class, major, minors)
- What excites you about computer science?
- What do you like to do in your free time?
- How is God working in this season of your life?
- What's one interesting thing about you that nobody in the room knows?

Course Syllabus

Feedback Sheets

Why Databases?

Databases are Biblical

- Genesis 2:19
- Good information architecture reflects God’s image
  - Naming and organizing ideas
  - Communicating concepts and information
    - Biblical genealogies, census data, laws, building instructions
  - Fighting against the chaos and entropy brought on by the Fall
- “Eternity in the heart of man., yet no one can fathom what God has done…” Ecclesiastes 3:11
  - “Infinite complexity within perfect structure.” Andrew Pudewa

Why do we need databases?

- Why can’t we just use the file system?
- Store data in files
- Write applications to access and manipulate this data as they are needed
- That’s the way our (grand)parents did it!

Using the File System

- Sometimes using the file system for your applications is just fine
  - Word processing
  - Pictures
  - Games
- As the complexity of the application and the amount of information it works with increase, some disadvantages of solely relying on the file system start to surface..
  - Example – consider a banking system

Why Use a Database Instead of the File System?

File System Disadvantages

- Data redundancy - wasted space
- Update issues – every copy of the data needs to be modified
- Data inconsistency – sometimes every copy is not modified
- Data access issues (getting to just the right data)
  - “There’s no program for that.”
- Data isolation - pulling all the data from disparate sources together)
- Integrity constraints buried in application logic – hard to add to or change
- Atomicity problems – what happens when the system crashes during an important operation?
- Concurrency issues – when multiple users work with the same data at the same time
- Security issues – how to give someone access to some, but not all, of the data

A Database Can Help.

- “Decouples” applications from the files on the file system
  - Programs go through the database to access data stored in the underlying files
- This extra software layer is called the *database management system (DBMS)*

The Data Dictionary Contains Data about the Data.

- In addition to storing data, the DBMS also stores *metadata* – data about the data – in a *data dictionary*
  - A standard name for each data item that applications use to access it
  - Where the data item is stored (which file and where in that file)
  - Security constraints – rules about who is allowed to access which data can be applied at the data item level; these are enforced by the DBMS
  - Integrity constraints – which values are valid for data items; enforced by the DBMS

Databases Have Advantages…

- Atomic *transactions* – either everything in a batch of work completes, or no changes are made
- Concurrency management
- Ad hoc / customized access to the data through a *query language*
  - SQL

…but also Trade-offs.

- The additional DBMS software layer comes with some costs
  - Each application incurs overhead by going through the database to access its data
  - Applications (on their own) cannot optimize access to data stored in the underlying database files
  - Designers and programmers need more (albeit standardized) knowledge of how a DBMS works
  - Additional layer can lead to increased complexity (at least in the short term)
- Database and file systems are not "either / or" solutions
  - More like "both / and"

What is a Database?

Characteristics and Structure

- A *database* consists of an organization's operational data
  - Data is typically interrelated
  - Set of programs to work with the data
  - An environment that is consistent, efficient, and convenient to use
  - Does not necessarily include transient data like input and output streams
- Several aspects from which to look at databases
  - Data description layers / levels of abstraction
  - Data models
  - DBMS Organization
  - Transactions

DBMS Abstraction Layers

- Physical – where the data is actually stored (files)
- Logical (conceptual) – describes data and data relationships in the data
- View – targeted end-user interfaces to database that highlights some data, hides others, and may include virtual fields computed from the data.
- Data independence – changes at one abstraction layer should not impact other layers

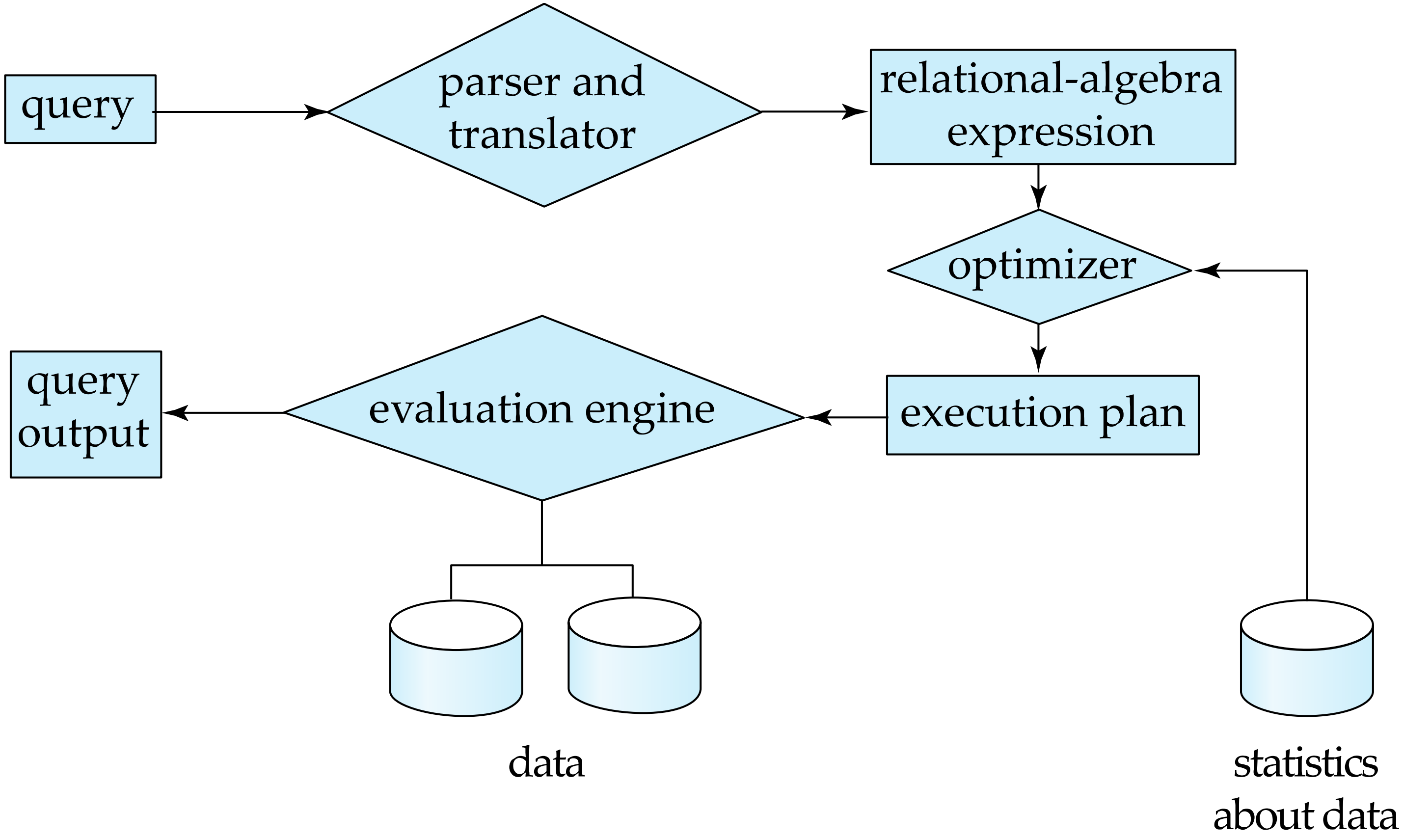
Data Models

- Relational model
- Entity-relationship model
- Aggregate data models (NoSQL)
  - Key-value data model
  - Document data model
  - Column-family stores
- Graph model
- Other models
  - Object-based
  - Semi-structured (XML) models
  - Hierarchical
  - Network

DBMS Components

- Storage manager
  - Interface between the applications and queries using the system and the low-level data
  - Manages interaction with file system
  - Facilitates efficient storing, retrieving, and updating data
- Query processor
  - Parses and executes queries efficiently

Query Processor Diagram



A Transaction is a Complete Unit of Work with the Database.

- Unit of work with the following (ACID) properties
  - Atomic
  - Consistent
  - Isolated
  - Durable
- Transaction management involves coping with system failures as well as concurrent users

Terminology and Concepts

Covered So Far…

- Database
- Metadata
- Query language
- Data description/abstraction layers (3 of them)

Important Concerns

- Data redundancy
- Data inconsistency
- Security constraints
- Integrity constraints
- Concurrency constraints

Database Building Blocks

- Schema
- Instance
- Entity
- Table
- Row (record)
- Column (attribute)

Database Languages

- Data definition language (DDL)
- Data manipulation language (DML)
- Query Language
  - Structured Query Language (SQL)

Design Project