Database Management Systems

Reagan Shirk

August 31, 2020

View of Data

- You have a logical level and a physical level of data
- The physical level is telling the management system how you want your data to physically be stored

Relational Model

- A collection of tables. . . ?
  - I didn't totally catch what she was saying here but it's in Chapter 2 of the book and I'll go look and try to elaborate
  - Oh I think it's a collection of similar data, for example she has a table of instructors and a table of departments
- When talking about the logical schema, you give the datatype for each column in the data
  - i.e. if you have an ID column and a Name column, the ID would be an int and the Name would be a string

Data Definition Language (DDL)

- Language used for defining the database schema
- Relational database uses SQL

```
create table instructor(
    ID        char(5),
    name      varchar(20),
    dept_name varchar(20),
    salary    numeric(8, 2))
)
```

- If you're curious, the difference between `char` and `varchar` is that `char` is of fixed length and `varchar` is of variable length
- The DDL compiler creates a set of table templates stored in a data dictionary (this data dictionary is metadata - data about data)
- There are integrity constraints; primary key and referential integrity
  - For example, the `dept_name` value in any `instructor` tuple must appear in the `department` relation

SQL (Chapters 3, 4, 5)

- SQL is a widely used non-procedural language
- Applications generally access databases through either:
  - language extensions that allow embedded SQL
  - interface which allows for SQL queries to be sent to a database

Database Design

- A longwinded way of saying that you don't want redundancy in your data. Why would the department budget need to be in a datatable of professor information? It wouldn't.

Design Approaches

Entity Relationship Model

- A collection of *entities* and *relationships* (I'm shocked)
  - An entity is a thing or object in the enterprise that is distinguishable from other objects, it is described by a set of attributes
  - A relationship is an association among several entities
    - An instructor is a member of a department, that is their entity-relationship model
- This is **not an implementation model**, it is just for design
  - An implementation model is something that has been/can be implemented

Storage Management

- A storage manager is a program module that linkes the low-level data and the application programs/queries submitted to the system
- It is responsible for:
  - interaction with the file manager
  - Efficient storing, retrieving, and updating
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

Query Processing

- You have your query and your DBMS
  - You submit your query and it goes into the query processor to be parsed and translated
  - I got distracted. Looks like it goes into an optimizer after doing something with algebra? I'll have to come rewatch this part of lecture
  - After it gets optimized, the execution is planned and evaluation is performed, then you get your output

Transaction Management

- What if the system fails, what if more than one user is updating the same data at the same time?
- This is when transaction management comes into play
- Something about a transaction having a beginning and an end
- You can recover a database if the transaction fails in the middle
- A transaction is a collection of operations that performs a logical function
- Transaction management ensures that the database remains consistent and correct despite system and transaction failures
- Concurrency control controls the interaction among concurrent transactions to ensure consistency

Database Administrator (a person, not a code thing)

- a person who has central control over the system. They take care of:
  - schema definitions (logical design)
  - storage structure and access-method definitions (physical design)
  - schema and physical-organization modification
  - granting authorization for access
  - maintenance
  - backing up the database
  - ensuring enough free disk space
  - monitoring jobs running and performance

Database Architecture

- The architecture of a database system is influenced by the computer system that it's running on
- You got:
  - centralized
  - client server
  - parallel (multi-processor)
  - distributed

History Time

- 50's and 60's - AKA the stone age
  - Data processing using magnetic tapes for storage - these only provided sequential access
  - Punch cards used for input
- 60's and 70's - still the stone age
  - Hard disks (had to read that twice) allowed for direct access to data
  - Network and heirarchical data models were used frequently
  - Ted Codd (mathematician from England who worked for IBM) defines the relational data model (the not implementation model we talked about earlier)
    - Won the ACM turing award in 1981 for being a badass
    - He did something with set operations like union and intersection

When to not use a DBMS

- Woah our DBMS professor isn't telling us that we need to use one all the time? I'm shook
- Don't use one when theres:
  - too much cost associated due to
    - overhead of concurrency control, recovery, integrity, etc
    - High investment in hardware, software, and training
    - generality
  - additional problems may pop up, like:
    - database is not properly designed
    - database application is not properly implemented
  - Could be more desireable for a regular file system when:
    - db and application are simple and well defined, not expected to change
    - time critical applications
    - something I missed