EGCI321: Database System

Lecture 01: Database Architecture

EGCI 321: LECTURE01(WEEK01)

Course Content

Why do we use database?

- Functionality provided by a Database Management System
- Database Models: Rational, Network, OO

How do we use a DBMS?

- Relational model, query languages
- SQL
- Application programming
- Transactions and concurrency

How do we design a database?

- Entity-Relationship (ER) modeling
- Redundancy and normal forms

How do we manage a DBMS?

- Security and authorization
- Physical design/tuning

What is a Database?

Definition (Database)

- A *large* and *persistent* collection of (more-or-less) pieces of information organized in a way that facilitates efficient *retrieval* and *modification*

Examples:

- A file cabinet
- A library system
- A personnel management system

Definition (Database Management System (DBMS))

- A program (or set of programs) that manages details related to storage and access for database.

Files vs. DBMS

- Application must stage large datasets between main memory and secondary storage (e.g., buffering, page-oriented access, 32-bit addressing, etc.)
- Special code for different queries
- Must protect data from inconsistency due to multiple concurrent users
- Crash recovery
- Security and access control

Application of Databases

Original applications

- Inventory control
- Payroll
- Banking and financial systems
- Reservation systems

More recent applications

- Computer aided designed
- Software development
- Telecommunication systems
- E-commerce
- Dynamic/personalized web content

Application of Databases (cont.)

Common Circumstances:

- There is lots of data (mass storage)
- Data is formatted

Requirements:

- Persistence and reliability
- Efficient and concurrent access

Brief History of Data Management: 1950s

First generation 50’s and 60’s

- Batch processing
- Sequential files and tape
- Input on punched cards

Second generation (60’s)

- Disk enabled random access files
- New access methods (hash files)
- Mostly batch with some interactivity
- Independent applications with separate files
- Growing applications base

Brief History of Data Management: 1960s

As the application base grows, we end up with

- Many shared files
- A multitude of files structures
- A need to exchange data among applications

This cause a variety of problems

- Redundancy: multiple copies
- Inconsistency: independent updates
- Inaccuracy: concurrent updates
- Incompatibility: multiple formats
- Insecurity: proliferation
- Inauditability: poor chain of responsibility
- Inflexibility: changes are difficult to apply

Brief History of Data Management: 1960s (cont.)

Hierarchical data model

- IBM's Information Management System (IMS): concurrent access
- Only allows 1:N parent-child relationships (i.e. a tree)
- Hierarchy can be exploited for efficiency
- Queries navigate up and down trees—one record at a time
- Data access language embedded in business processing language
- Difficult to express some queries

Network data model

- Data organized as collections of sets of records
- Separate of physical data representation from users' view of data
- Pointers between records represent relationships
- Set types encoded as lists
- Queries navigate between records—still on record at a time

Database Management System

Idea: Abstracts common functions and creates a uniform well defined interface for applications access data

1. Data Model
   - All data stored in a well defined way
2. Access Control
   - Only authorized people get to see/modify it
3. Concurrency Control
   - Multiple concurrent applications access data
4. Database Recovery
   - Nothing gets accidentally lost
5. Database Maintenance

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.

Three Level Schema Architecture

Definition (schema)

- A schema is a description of the data interface to the database (i.e., how the data is organized)

  1. External schema (view): what the application programs and user see. May differ for different users of the same database.
  2. Conceptual schema: description of the logical structure of ***all*** data in the database.
  3. Physical schema (internal schema): description of physical aspects (selection of files, devices, storage algorithms, etc.)

Definition (Instance)

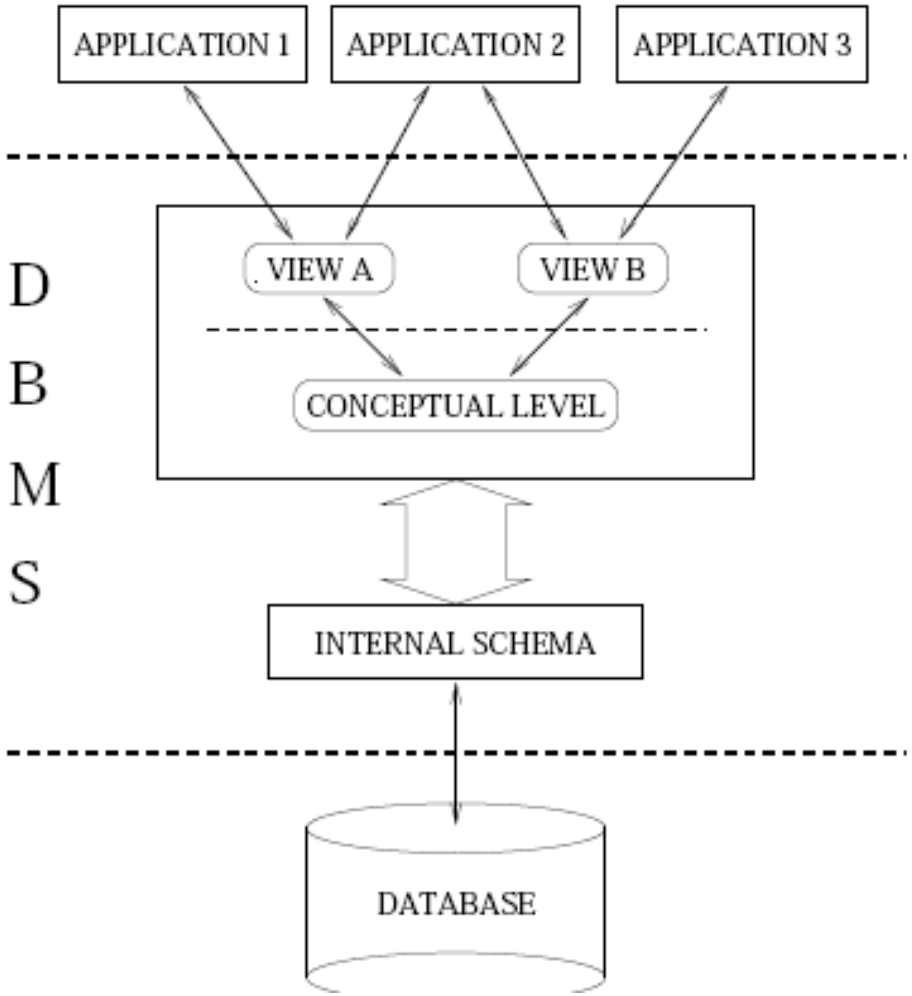
- A **database instance** is a database (real data) that conforms to a given schema

Example

- Please name column A, B, C, and D
- What is the table's name?
- Define data model and data schema

A	B	C	D
Bob	Cat	New York	502-100-3004
Mary	Lamb	Seattle	450-600-2300
Jame	Fox	Florida	620-700-5678

Three-level Schema Architecture (cont.)



Data Independence

Idea

- *Applications do not access data directly* but, rather through an abstract data model *provided by the DBMS*

Two kinds of data independence:

- ***Physical:*** applications immune to change in storage structures
- ***Logical:*** application immune to change in data organization

Note: One of the most important reasons to use a DBMS!

Interfacing to the DBMS

Data Definition Language (DDL): for specifying schemas

- Have different DDLs for external schema, conceptual schema, internal schema
- Information is stored in the data dictionary, or catalogue

Data Manipulation Language (DML): for specifying queries and updates

- Navigational (procedural)
- Non-navigational (declarative)

Types of Database Users

End user:

- Accesses the database indirectly through forms or other query-generating applications, or
- Generates *ad-hoc queries* using DML

Application developer:

- Designs and implements applications that access the database

Database administrator (DBA):

- Manages conceptual schema.
- Assists with application view integration.
- Defines internal schema.
- Loads and reformats database.
- Responsible for security and reliability

Transactions

When multiple applications access the same data, undesirable results occur.

Example:

```
withdraw(AC,1000)              withdraw(AC,500)
  Bal := getbal(AC)
                                 Bal := getbal(AC)
  if (Bal>1000)                  if (Bal>500)
    <give-money>                   <give-money>
    setbal(AC,Bal-1000)
                                   setbal(AC,Bal-500)
```

Idea

- Every application may think it is the sole application accessing the data. The DBMS should guarantee correct execution

Transaction (cont)

Definition (Transaction)

- An application-specified atomic and durable unit of work.

Properties of transactions ensured by the DBMS:

- Atomic: a transaction occurs entirely, or not at all
- Consistency: each transaction preserves the consistency of the database
- Isolated: concurrent transactions do not interfere with each other
- Durable: once completed , a transaction's changes are permanent

Recent History of Data Management

- Development of commercial relational technology
  - IBM DB2, Oracle, Informix, Sybase
- SQL standardization efforts through ANSI and ISO
- Object-oriented DBMSs
  - Persistent objects
  - Object id's, methods, inheritance
- Continued expansion of SQL and system capabilities

Recent History of Data Management (cont.)

New application areas:

- The internet
- Online analytic processing (OLAP)
- Data warehousing
- Embedded systems
- XML
- Data streams

Summary

Using a DBMS to manage data helps:

- To remove common code from applications
- To provide uniform access to data
- To guarantee data integrity
- To manage concurrent access
- To protect against system failure
- To set access policies for data

Reference

- Ramakrishnan R, Gehrke J., Database management systems, 3rd ed., New York (NY): McGraw-Hill, 2003.

Any Questions?

:O)

Thank you