Database System

LESSON 01: INTRODUCTION

September 2019

Contents

Database Management System (DBMS)

DBMS contains information about a particular enterprise

- Collection of interrelated data
- Set of programs to access the data
- An environment that is both *convenient* and *efficient* to use

Database Applications:

- Banking: transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Databases can be very large.

Databases touch all aspects of our lives

Drawbacks of using file systems to store data

Data redundancy and inconsistency

- Multiple file formats, duplication of information in different files

Difficulty in accessing data

- Need to write a new program to carry out each new task

Data isolation

- Multiple files and formats

Integrity problems

- Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
- Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

Atomicity of updates

- Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all

Concurrent access by multiple users

- Concurrent access needed for performance
- Uncontrolled concurrent accesses can lead to inconsistencies
  - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

Security problems

- Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

Levels of Abstraction

Physical level: describes how a record (e.g., instructor) is stored.

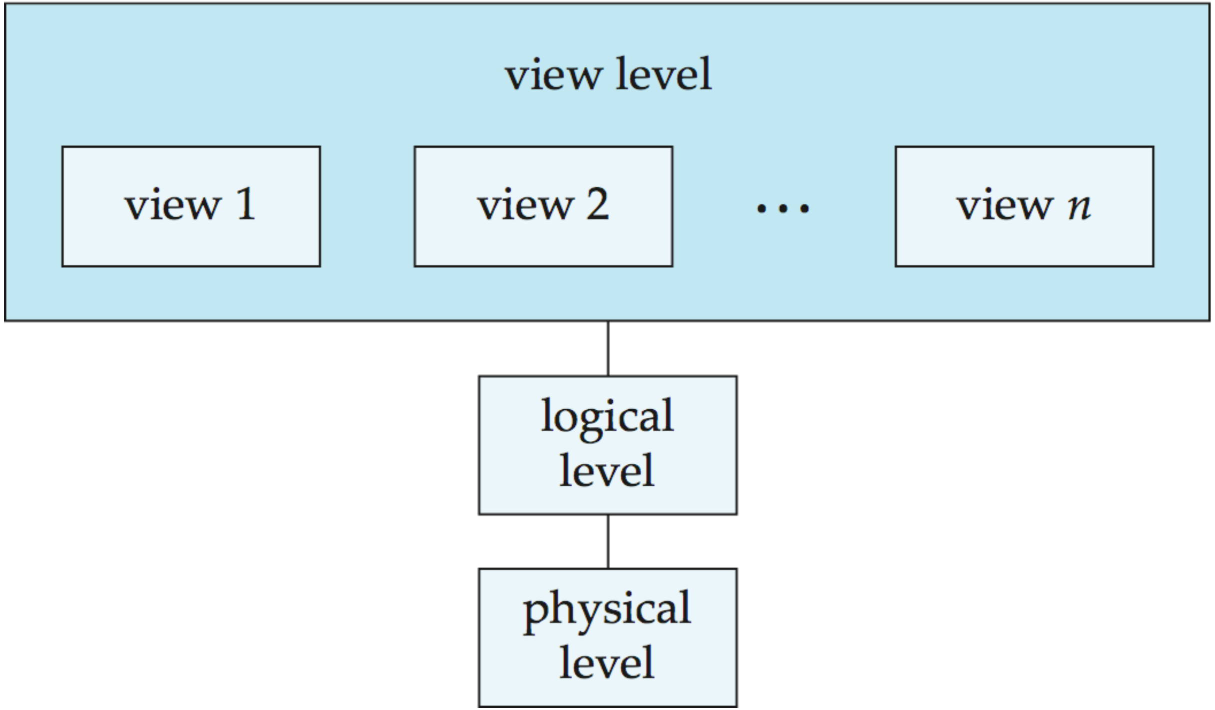
Logical level: describes data stored in database, and the relationships among the data.

```
type instructor = record
        ID : string;
        name : string;
        dept_name : string;
        salary : integer;
    end;
```

View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

Similar to types and variables in programming languages

Logical Schema – the overall logical structure of the database

- Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program

Physical schema– the overall physical structure of the database

Instance – the actual content of the database at a particular point in time

- Analogous to the value of a variable

Physical Data Independence – the ability to modify the physical schema without changing the logical schema

- Applications depend on the logical schema
- In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Data Models

A collection of tools for describing

- Data
- Data relationships
- Data semantics
- Data constraints

Relational model

Entity-Relationship data model (mainly for database design)

Object-based data models (Object-oriented and Object-relational)

Semistructured data model (XML)

Other older models:

- Network model
- Hierarchical model

Relational Model

All the data is stored in various tables.

Example of tabular data in the relational model

Columns

Rows

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

Data Definition Language (DDL)

Specification notation for defining the database schema

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

DDL compiler generates a set of table templates stored in a ***data dictionary***

Data dictionary contains metadata (i.e., data about data)

- Database schema
- Integrity constraints
  - Primary key (ID uniquely identifies instructors)
- Authorization
  - Who can access what

Data Manipulation Language (DML)

Language for accessing and manipulating the data organized by the appropriate data model

- DML also known as query language

Two classes of languages

- **Pure** – used for proving properties about computational power and for optimization
  - Relational Algebra
  - Tuple relational calculus
  - Domain relational calculus
- **Commercial** – used in commercial systems
  - SQL is the most widely used commercial language

SQL

The most widely used commercial language

To be able to compute complex functions SQL is usually embedded in some higher-level language

Application programs generally access databases through one of

- Language extensions to allow embedded SQL
- Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

Need to come up with a methodology to ensure that each of the relations in the database is “good”

Two ways of doing so:

- Entity Relationship Model
  - Models an enterprise as a collection of *entities* and *relationships*
  - Represented diagrammatically by an *entity-relationship diagram:*
- Normalization Theory
  - Formalize what designs are bad, and test for them

Database Engine

Storage manager

Query processing

Transaction manager

Storage Management

Storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible to the following tasks:

- Interaction with the OS file manager
- Efficient storing, retrieving and updating of data

Issues:

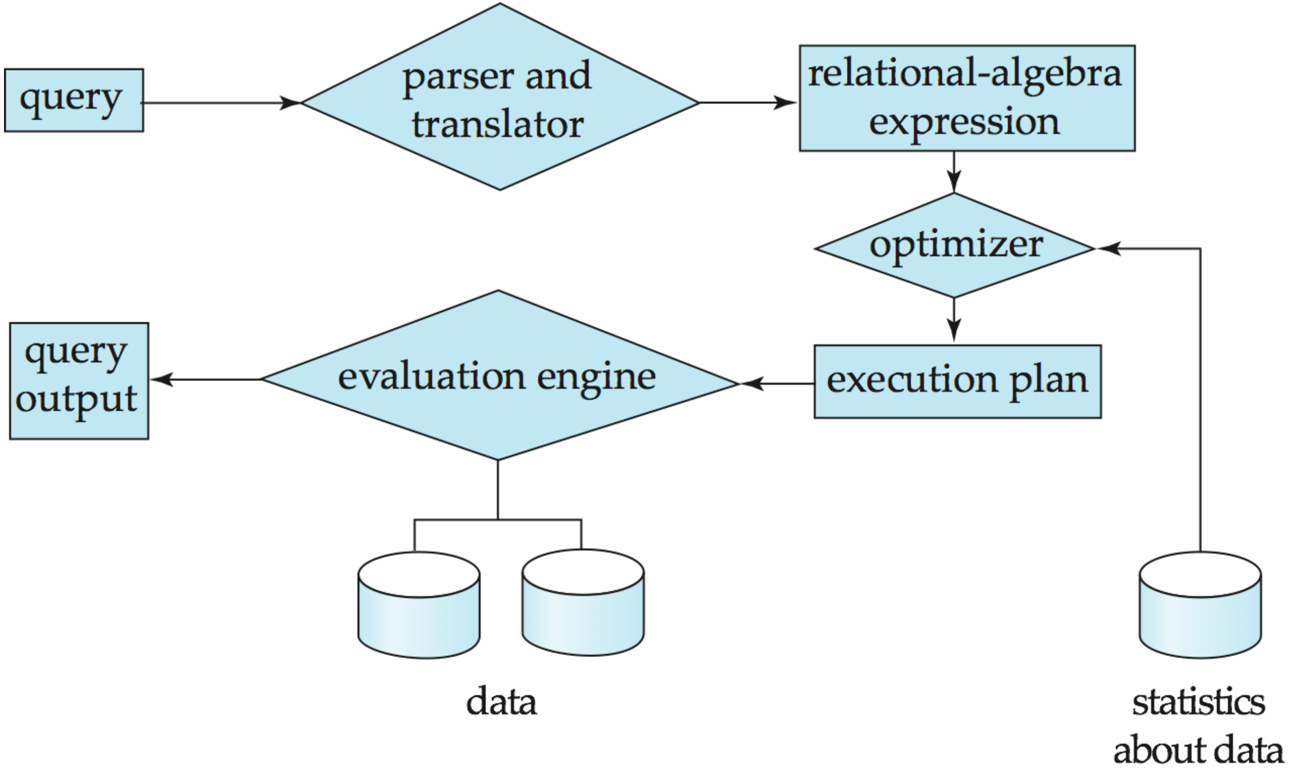
- Storage access
- File organization
- Indexing and hashing

Query Processing

Parsing and translation

2. Optimization

3. Evaluation



Query Processing (Cont.)

Alternative ways of evaluating a given query

- Equivalent expressions
- Different algorithms for each operation

Cost difference between a good and a bad way of evaluating a query can be enormous

Need to estimate the cost of operations

- Depends critically on statistical information about relations which the database must maintain
- Need to estimate statistics for intermediate results to compute cost of complex expressions

Transaction Management

What if the system fails?

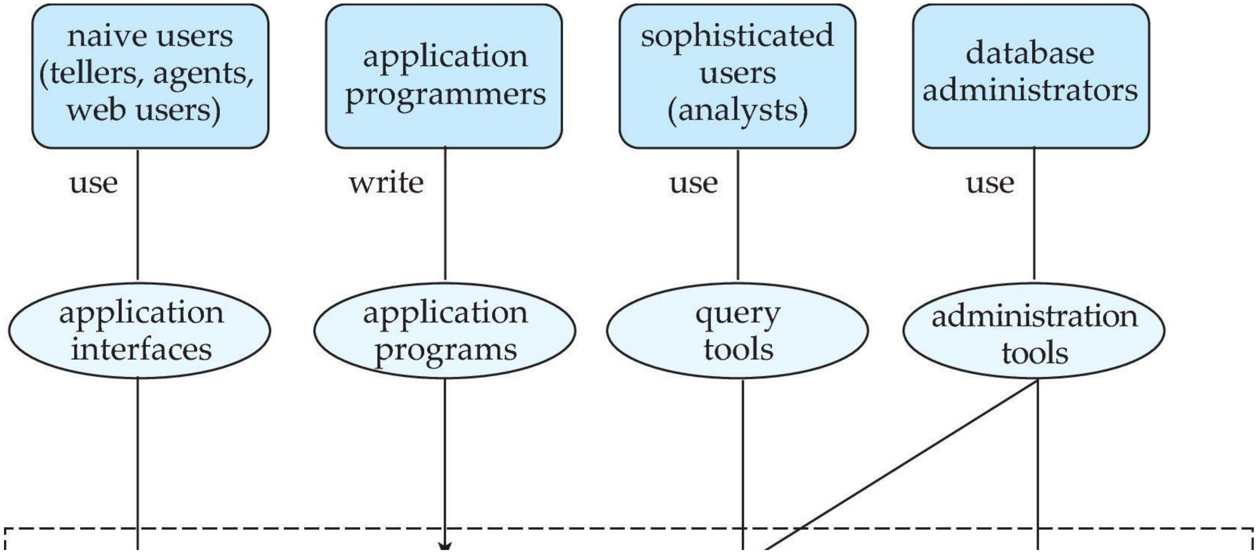
What if more than one user is concurrently updating the same data?

A **transaction** is a collection of operations that performs a single logical function in a database application

Transaction-management component ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

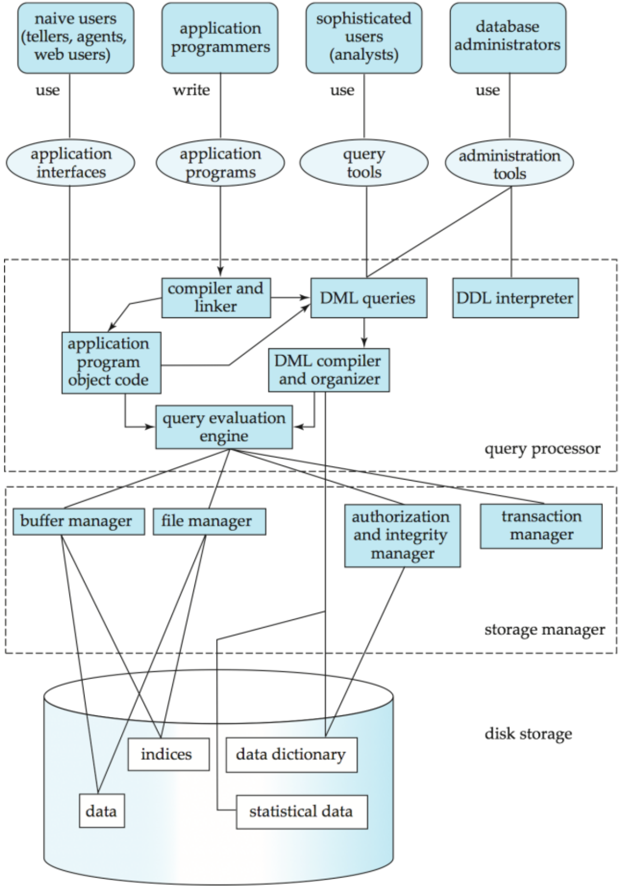
Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

Database Users and Administrators



Database

Database System Internals



Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed

History of Database Systems

1950s and early 1960s:

- Data processing using magnetic tapes for storage
  - Tapes provided only sequential access
- Punched cards for input

Late 1960s and 1970s:

- Hard disks allowed direct access to data
- Network and hierarchical data models in widespread use
- Ted Codd defines the relational data model
  - Would win the ACM Turing Award for this work
  - IBM Research begins System R prototype
  - UC Berkeley begins Ingres prototype
- High-performance (for the era) transaction processing

History (cont.)

1980s:

- Research relational prototypes evolve into commercial systems
  - SQL becomes industrial standard
- Parallel and distributed database systems
- Object-oriented database systems

1990s:

- Large decision support and data-mining applications
- Large multi-terabyte data warehouses
- Emergence of Web commerce

Early 2000s:

- XML and XQuery standards
- Automated database administration

Later 2000s:

- Giant data storage systems
  - Google BigTable, Yahoo PNuts, Amazon, ..