Chapter 1: Introduction

Database System Concepts, 6th Ed.

Outline

- Data base definitions, applications and types,modes etc
- Purpose of Database
- The Need for Databases or functionality of data base
- View of data
- Data Models
- Relational Databases
- Data base Languages
- Database Design
- Storage Manager
- Query Processing
- Transaction Manager

Basic Definitions

- DBMS - DB + programs to access,manage this data in a convenient and efficient manner.
- Data Base: DB: collection of interrelated data
- Data :known facts that can be recorded, processed, transmitted and have implicit meaning.
- Ex: data of measurements , statistics etc
- Use of data : reasoning, discussion, calculation

Basic Definitions

- **Database Implicit Properties: DB Represents Mini-world:**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university, Hospital , Library system
  - DB is built and populated with data – for specific purpose & for intended users
  - DB with different size and complexity – Amazon.com -20 million items – Occupies over 2 Terabytes & stored on 200 servers – 15 million visitors /day – items r added to the inventory & stock quantities are updated frequently
- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database. – Defining, Constructing/populating, Manipulating, and Sharing databases, protecting – among users and applications,

    System protection – against H/w-S/w malfunction
  - Meta-data / DB catalog/ DB Dictionary
- **Database System:**
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

Summary

- DBMS are used to manage collection of data tat are ighly valuable, large and are used by multiple people and applications, often at same time.
- Hence it is a complex system.
- Any complex system design is managed thro abstraction.
- So, todays DBMS are also using abstraction concept.

DBMS Applications Example

- 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
  - Social media: users ,connections, posts rating etc.
  - Online retailers: orders, sales etc
  - Document data bases: papers,articles patents etc
  - Navigational systems: routes,roadstrais,buses etc
  - Online advertisements: click stream data, product suggestions etc
- Databases can be very large. Applications are also very large.
- Databases touch all aspects of our lives

Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
    - **system** designed to capture, store, manipulate, analyze, manage, and present spatial or **geographic** data
  - Data Warehouses
    - central repositories of integrated **data** from one or more disparate sources
    - used for reporting and **data** analysis
  - Real-time and Active Databases
    - Weather forcasting system

Modes of DB usage

- OLTP online transactional processing

More users interact with data base, with each user working on retrieving small amount of data and performing smaller updates

- OLAP online analytical processing

Data processing to draw conclusions, decision making for business etc

Purpose of DBS

- In the early days, database applications were built directly on top of file systems which store data.
- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

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[correctness and completeness of data w.r.to DB]
  - Integrity constraints
  - 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

Typical DBMS Functionality

- *Define* a particular database in terms of its data types, structures, and constraints
- *Construct* or Load the initial database contents on a secondary storage medium
- *Manipulating* the database:
  - Retrieval: Querying, generating reports
  - Modification: Insertions, deletions and updates to its content
  - Accessing the database through Web applications
- *Processing* and *Sharing* by a set of concurrent users and application programs – yet, keeping all data valid and **consistent**

Typical DBMS Functionality

- Other features:
  - Protection or Security measures to prevent unauthorized access
  - Presentation and Visualization of data
  - Maintaining the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

Levels of Abstraction

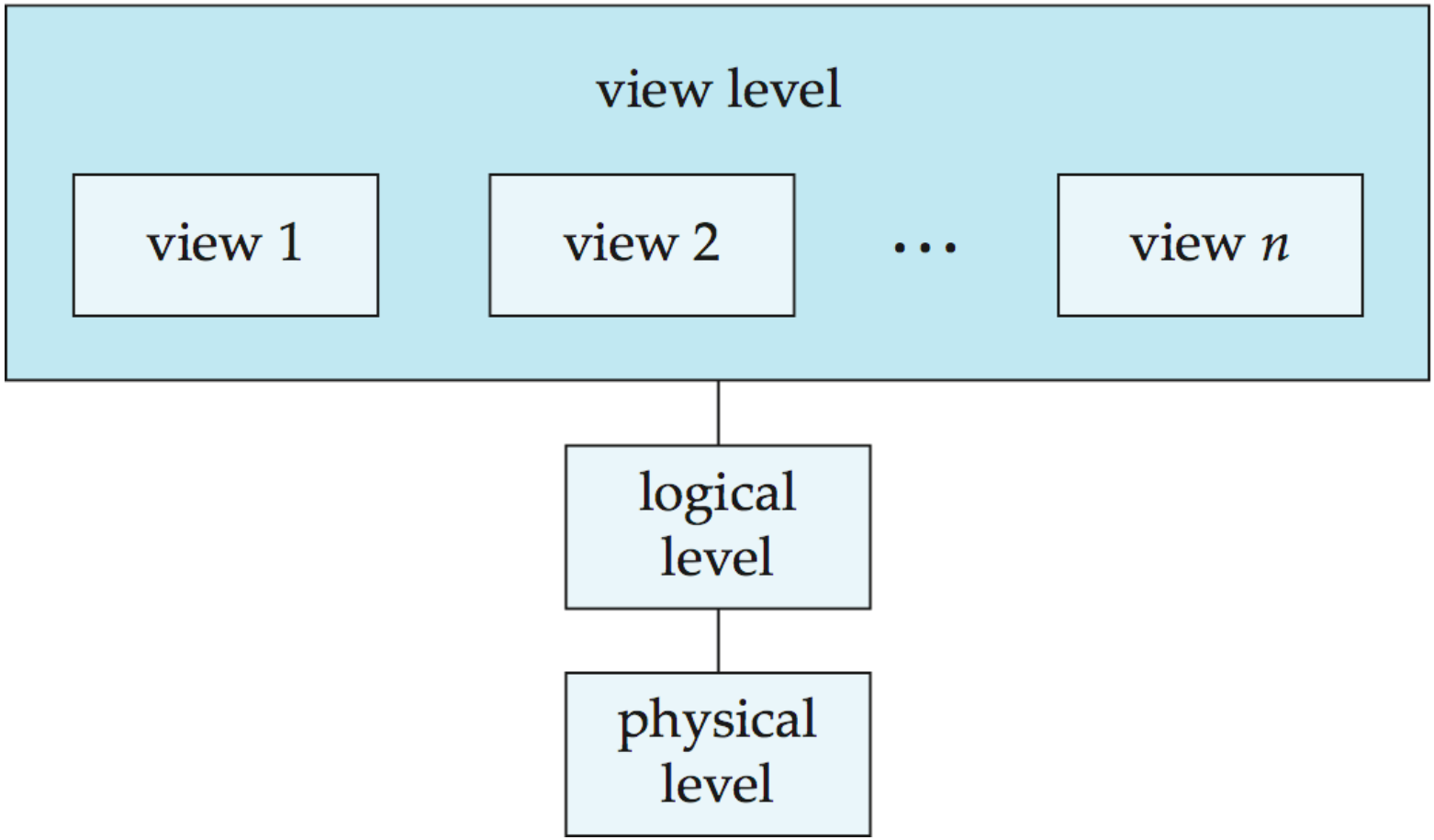
- **Physical level:** describes how a record (e.g., instructor) is stored. (internal)
- **Logical level:** describes data stored in database, and the relationships among the data.(conceptual)

```
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. (external)

View of Data

An architecture for a database system



Instances and Schemas

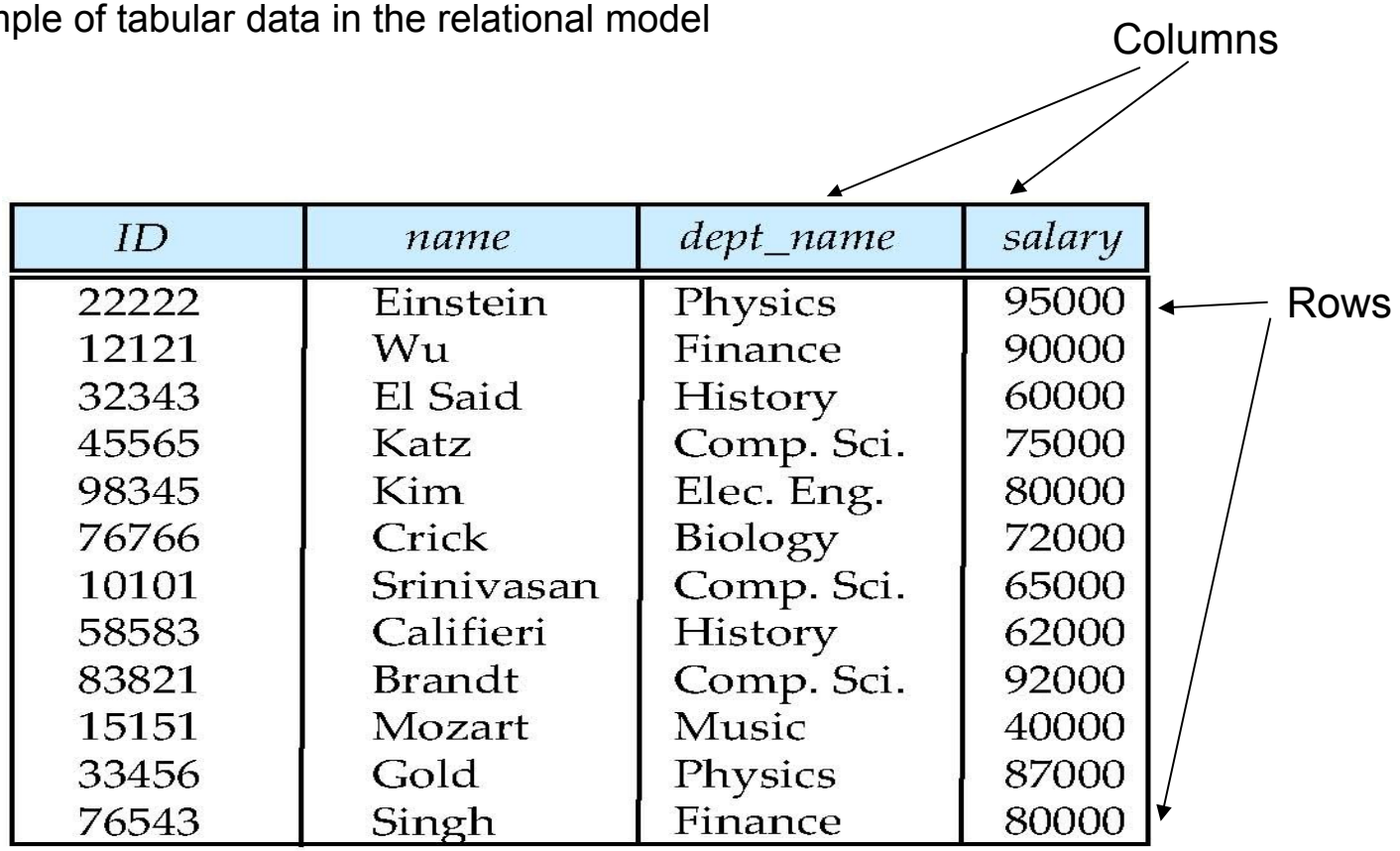
- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable 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: Ex: MySQL, PostgreSQL, Oracle Database.
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented(db4o) and Object-relational)
- Semistructured data model (XML/JSON)
- The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes.
- Other models:
  - Graph model (Neo4j)
  - Network model(CODASYL)
  - Hierarchical model(IMS)
  - Document data model(CouchDB,MongoDB)

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



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

A Sample Relational Database

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

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Normal DDL 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)
- Also additional properties of data(Data Storage DL)
  - Database schema implementation detail
  - Integrity constraints, consistency constraints etc
    - Primary key (ID uniquely identifies instructors), Domain,referential etc
  - Authorization
    - Who can access what: read,insert,delete,update etc

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
- Query: is a Statement for retrieval of data.
  - DML commonly known as query language(QL) though technically QL is subset of DML
- 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

DML classes/types

- Procedural: what to do, how to do
- Declarative: what to do

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent 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 (Chapter 7)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 8)
    - Formalize what designs are bad, and test for them

Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

Database Engine

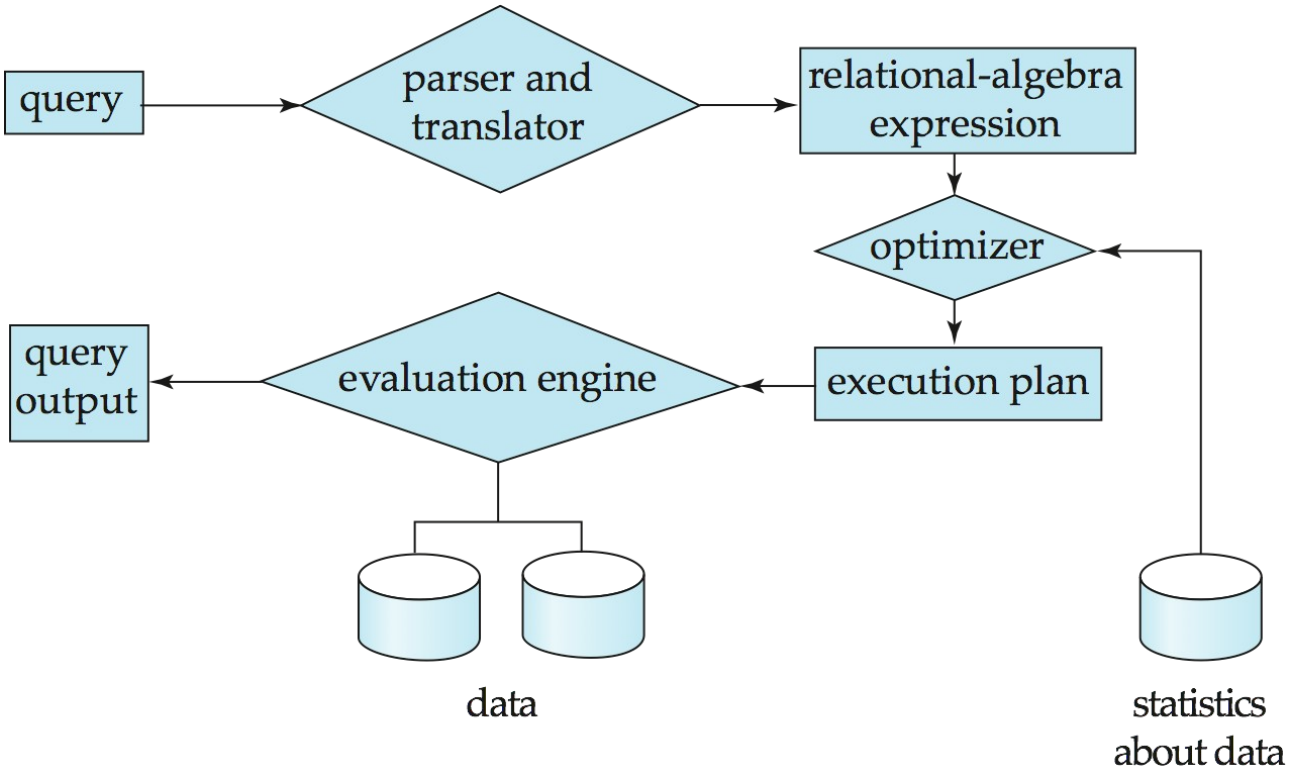
- Storage manager
- Query processing
- Transaction manager

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

1. Parsing and translation
2. Optimization
3. Evaluation



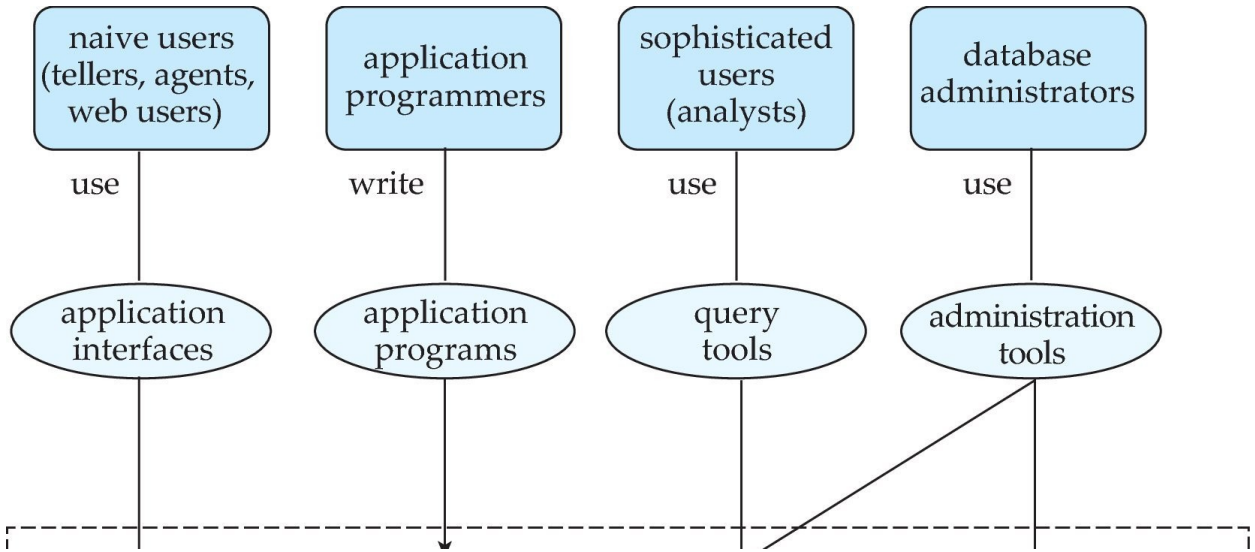
Query Processing (Cont.)

- 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

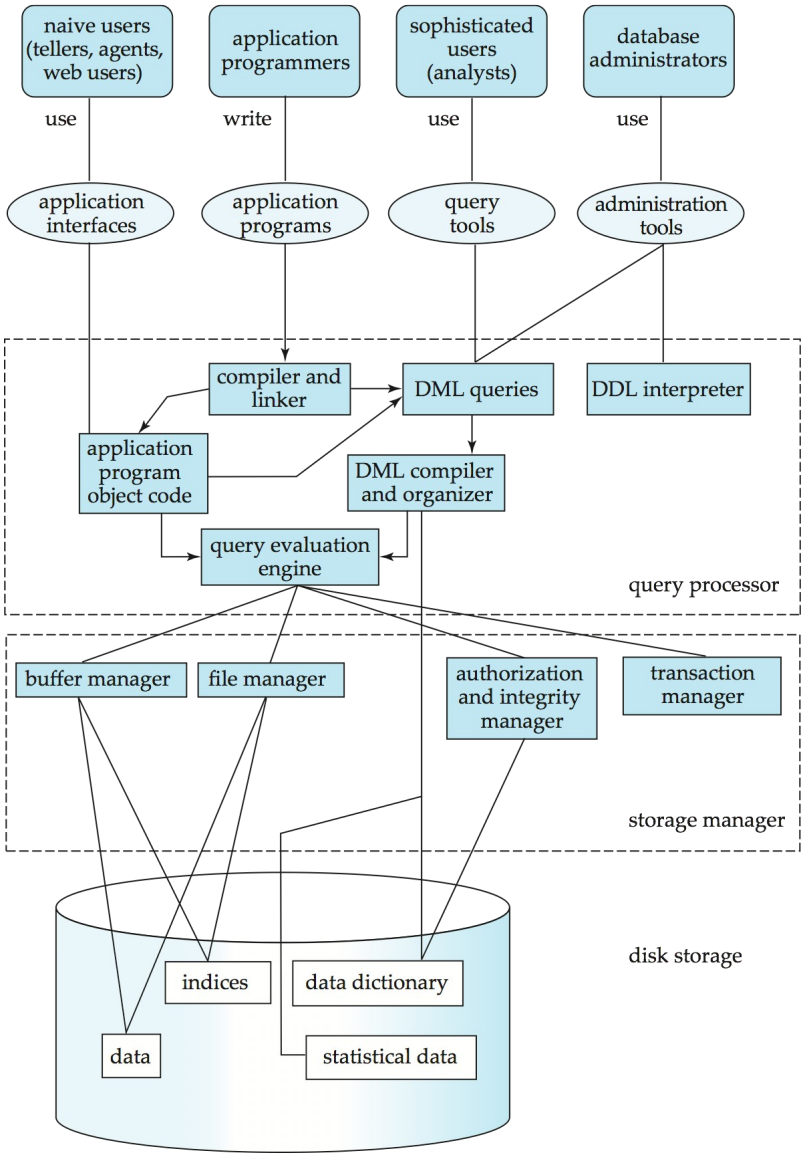
Transaction Management

- 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 System Internals



Simplified database system environment

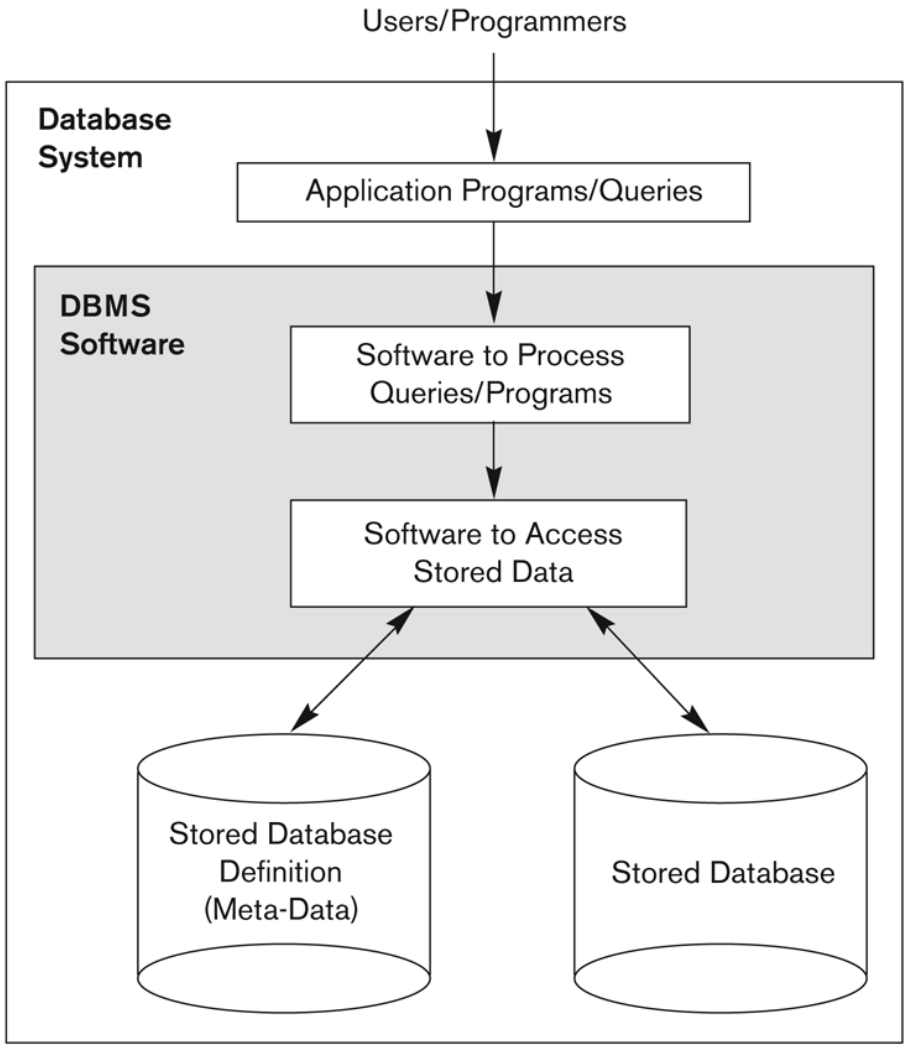


Figure 1.1
A simplified database system environment.

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, ..

End of Chapter 1