Introduction to Database Management System

Kathmandu University
BE Computer Engineering
Year II/II

Overview

- Database Systems Applications
- Database System versus File systems
- View of Data
- Database Languages
- Database Users and Administrators
- Transaction Management
- Database Architecture

Introduction: Database management System

As the name suggests, the **Database Management System** consists of two parts. They are:

1. **Database** and

2. **Management System**

Database

- What is a **Database**?
- To find out what database is, we have to start from **data**, which is the basic building block of any DBMS.
- **Data**: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).
- **Record**: Collection of related data items, e.g. in the above example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll	Name	Age
1	ABC	19

Database

- **Table or Relation:** Collection of related records

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

- The columns of this relation are called **Fields**, **Attributes** or **Domains**. The rows are called **Tuples** or **Records**.

Database

- **Database**: Collection of related relations. Consider the following collection of tables:

T1

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	28

T2

Roll	Address
1	KOL
2	DEL
3	MUM

T3

Roll	Year
1	I
2	II
3	I

T4

Year	Hostel
I	H1
II	H2

Database

- We now have a collection of 4 tables. They can be called a **“related collection”** because we can clearly find out that there are some common attributes existing in a selected pair of tables.
- Because of these common attributes we may combine the data of two or more tables together to find out the complete details of a student.
- Questions like “Which hostel does the youngest student live in?” can be answered now, although **Age** and **Hostel** attributes are in different tables.

Database Management System

- A **database-management system (DBMS)** is a collection of interrelated data and a **set of programs** to access those data. This is a collection of related data with an implicit meaning and hence is a database.
- The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both **convenient** and **efficient**.
- By data, we mean known facts that can be recorded and that have implicit meaning.

Database Management System

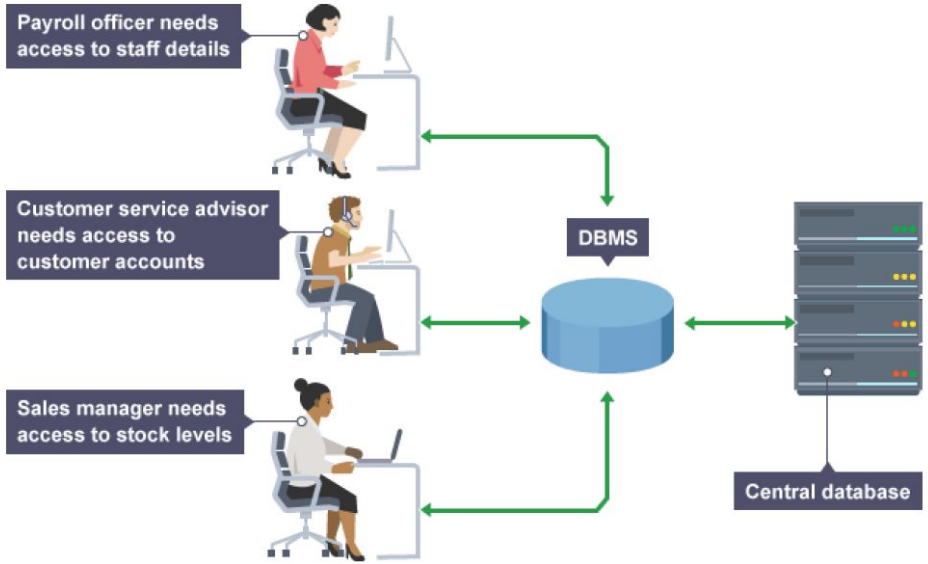
- Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.
- If data are to be shared among several users, the system must avoid possible anomalous results.

Database Management Systems

- 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 systems are used to manage collections of data that are:
  - Highly valuable
  - Relatively large
  - Accessed by multiple users and applications, often at the same time.
- A modern database system is a complex software system whose task is to manage a large, complex collection of data.
- Databases touch all aspects of our lives

Views of DBMS

- A database in a DBMS could be viewed by lots of different people with different responsibilities.



Applications of DBMS

- **Enterprise Information**
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- **Manufacturing**: management of production, inventory, orders, supply chain.
- **Banking and finance**
  - customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- **Universities**: registration, grades

Applications of DBMS ...

- **Airlines**: reservations, schedules
- **Telecommunication**: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- **Web-based services**
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- **Document databases**
- **Navigation systems:** For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

University Database Example

- In this course we will be using a university database to illustrate most of the concepts
- Data consists of information about:
  - Students
  - Instructors
  - Classes
- 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

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 (Michael Stonebraker) begins Ingres prototype
  - Oracle releases first commercial relational database
- High-performance (for the era) transaction processing

History of Database Systems ...

1980s:

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

1990s:

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

History of Database Systems ...

2000s

- Big data storage systems
  - Google BigTable, Yahoo PNuts, Amazon,
  - “NoSQL” systems.
- Big data analysis: beyond SQL
  - Map reduce

2010s

- SQL reloaded
  - SQL front end to Map Reduce systems
  - Massively parallel database systems
  - Multi-core main-memory databases

File System ?

- What is a file system?
- How is it used to store data/information?
- What are the advantages and disadvantages of file systems ?
- Compare File System with DBMS ?

File System

In computing, File System or filesystem (often abbreviated to fs) is a method and data structure that the operating system uses to control how data is stored and retrieved.

File system **organizes** the files and helps in **retrieval** of files when they are required. File systems consists of different files which are grouped into **directories**. The directories further contain other **folders** and **files**. File system performs basic operations like **management**, **file naming**, **giving access rules** etc.

File System

The **FAT** (short for File Allocation Table) file system is a general purpose file system that is compatible with all major operating systems (Windows, Mac OS X, and Linux/Unix).

Examples of file systems:

Windows: **NTFS**

Mac Os: **APFS**

Linux/Unix: **Ext4**

Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- **Data redundancy and inconsistency**: data is stored in multiple file formats resulting in 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

Purpose of Database Systems

- **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 be complete or not happen at all
- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
  - Ex: 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

Advantages of DBMS

- Reducing Data Redundancy
- Sharing of Data
- Data Integrity
- Data Security
- Privacy
- Backup and Recovery
- Data Consistency

File System Vs DBMS

Basis	File System	DBMS
Structure	File system is a software that manages and organizes the files in a storage medium within a computer	DBMS is a software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
. Query processing	There is no efficient query processing in file system.	Efficient query processing is there in DBMS.

File system Vs DBMS ...

Basis	File System	DBMS
Consistency	There is less data consistency in file system.	There is more data consistency because of the process of normalization.
Complexity	It is less complex as compared to DBMS	It is more complex as compared to File System
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file system
Cost	Relatively less expensive	Relatively more expensive than File system.

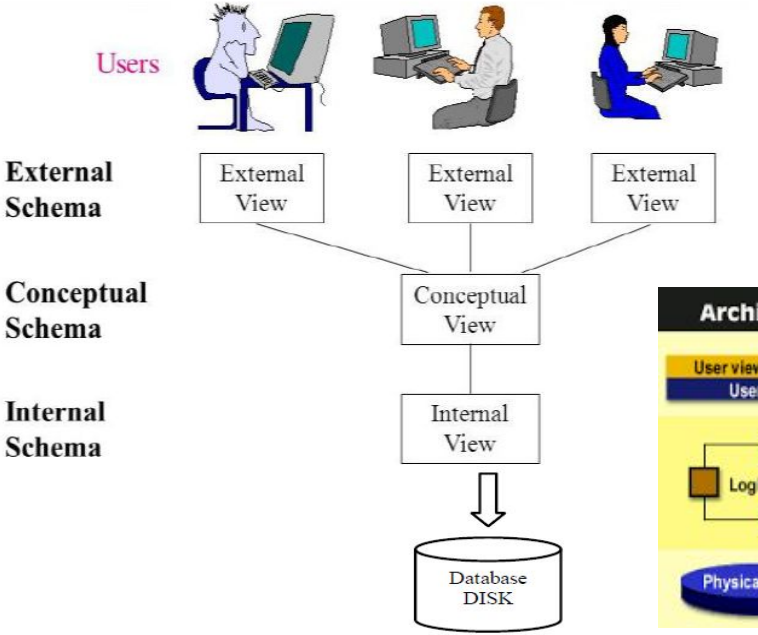
View of Data

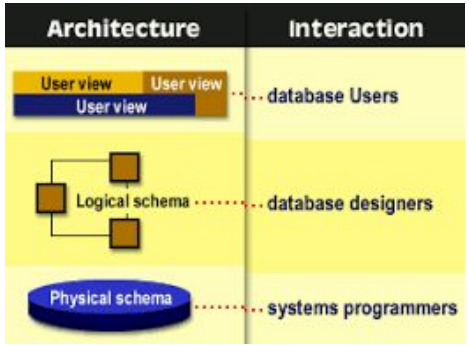
- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained.
  - **Data models**
    - A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - **Data abstraction**
    - Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

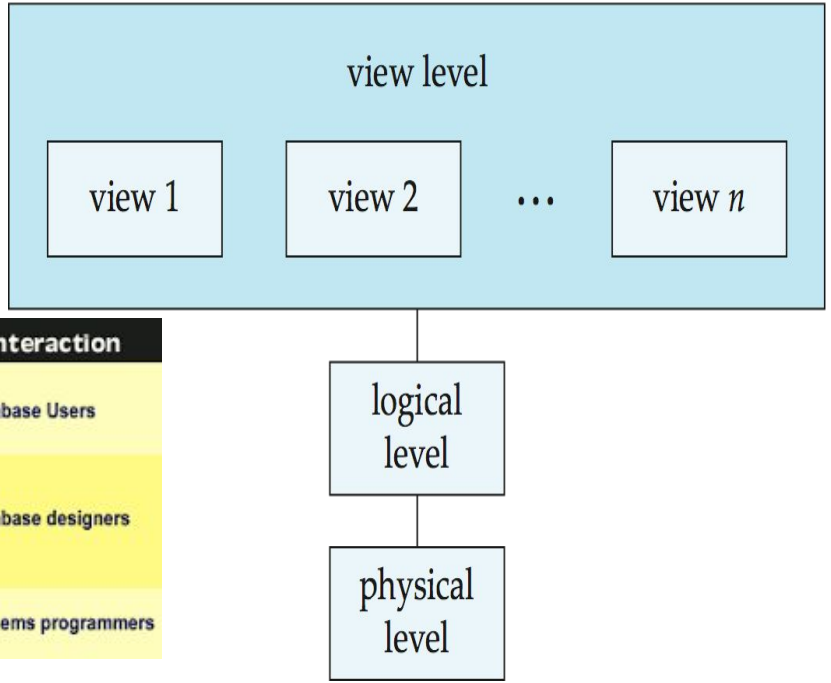
Data Abstraction

- For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database.
- Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

Data Abstraction/View of Data







Physical Level (Internal View/Schema)

- **Physical level** (or Internal View / Schema): The lowest level of abstraction describes how the data are actually stored.
- The physical level describes complex low-level data structures in detail.

Logical Level (Conceptual View/Schema)

- **Logical level (or Conceptual View / Schema)**: The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures.
- Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as **physical data independence**.
- Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.

Physical Data independence

- **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.

View level (or External View / Schema):

- **View level (or External View / Schema):** The highest level of abstraction describes only part of the entire database.
- Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database.
- The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

Data Types and Level of Abstraction Programming Vs DBMS: an Analogy

- Many high-level programming languages support the notion of a structure type. For example, we may describe a record as follows:

```
type instructor = record
ID : char (5);
name : char (20);
dept name : char (20);
salary : numeric (8,2);
end;
```

- This code defines a new record type called instructor with four fields. Each field has a name and a type associated with it. A university organization may have several other such record types.

Physical level of abstraction: Programming vs DBMS

- At the **physical level**, an instructor (department, or student) record can be described as a block of consecutive storage locations. The compiler hides this level of detail from programmers.
- Similarly, the database system hides many of the lowest-level storage details from database programmers. Database administrators, on the other hand, may be aware of certain details of the physical organization of the data.

Logical Level of Abstraction: Programming Vs DBMS

- At the logical level, each such record is described by a type definition, as in the previous code segment, and the interrelationship of these record types is defined as well.
- Programmers using a programming language work at this level of abstraction. Similarly, database administrators usually work at this level of abstraction.

View Level of Abstraction: Programming vs DBMS

- At the **view level**, computer users see a set of application programs that hide details of the data types.

- At the view level, several views of the database are defined, and a database user sees some or all of these views. In addition to hiding details of the logical level of the database, the views also provide a security mechanism to prevent users from accessing certain parts of the database.

- For example, clerks in the university registrar office can see only that part of the database that has information about students; they cannot access information about salaries of instructors.

Instance and Schemas

- Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database. The overall design of the database is called the **database schema.** Schemas are changed infrequently, if at all.
- **Schema** – the logical structure of the database
  - **Physical schema**: database design at the physical level
  - **Logical schema:** database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
- **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.

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

Data Model

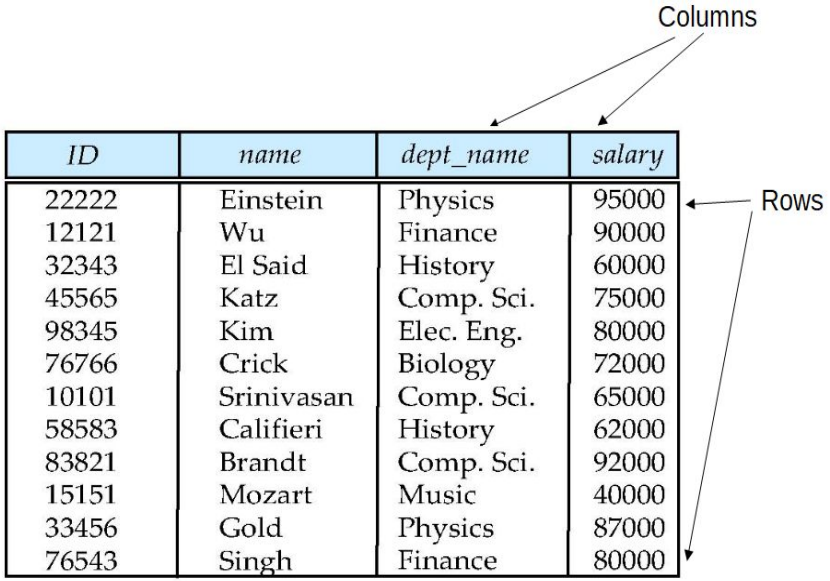
- Underlying structure of a database is the **data model**. It is the a collection of conceptual tools for describing
  - Data
  - data relationships
  - data semantics
  - consistency constraints.
- A data model provides a way to describe the design of a database at the physical, logical, and view levels.
- The data models can be classified into different categories:
  - Relational model
  - Entity Relationship Model
  - Object Based Model
  - Semi Structured Data Model

  Other older models: Network model, Hierarchical model, etc.

Relational Model

- The **relational model** uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a **record-based model**.
- Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular types. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.
- The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

A Sample Relational Database



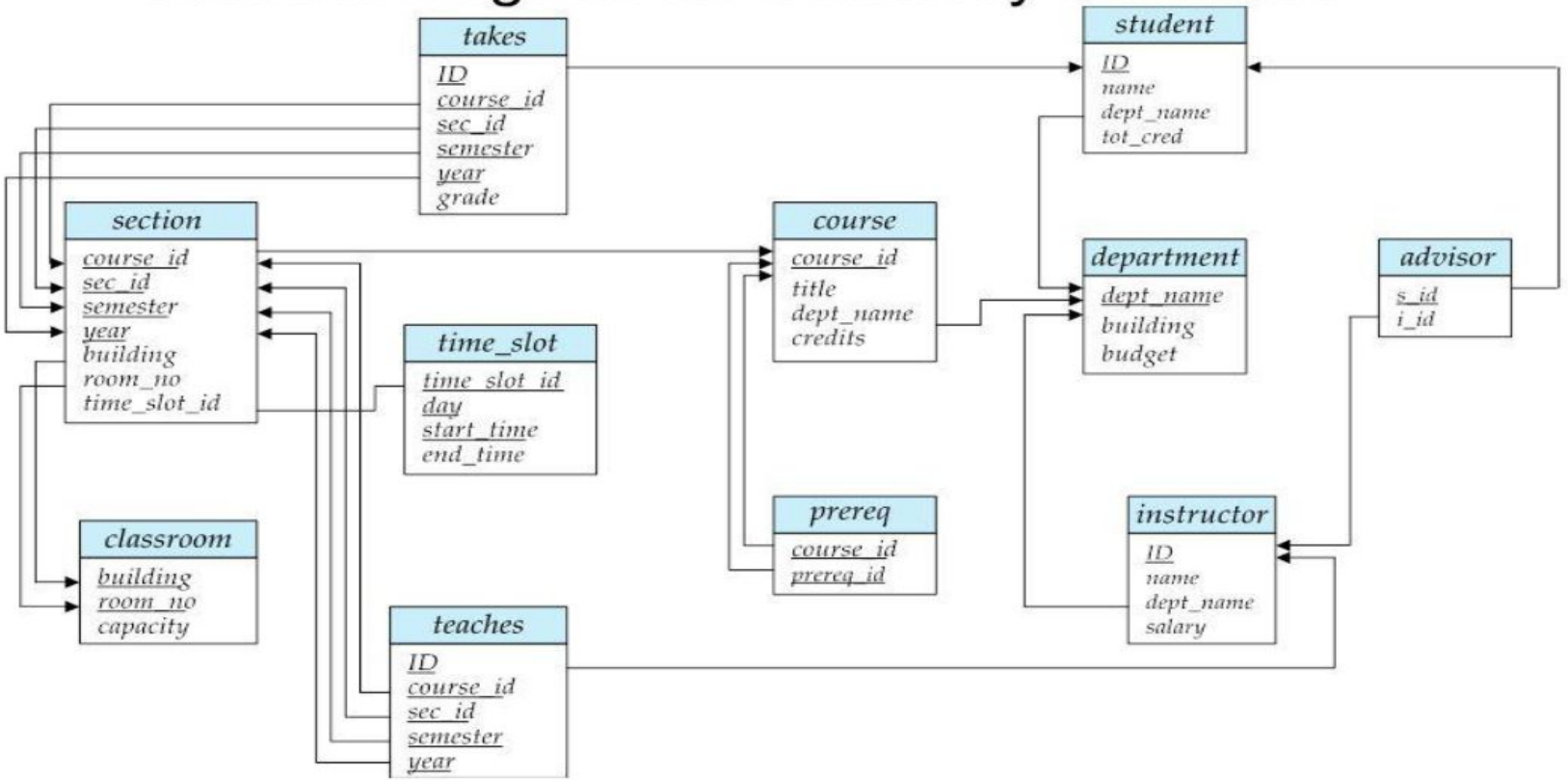
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music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

Schema Diagram for University Database



Database languages

- A database system provides a data-definition language to specify the database schema and a data-manipulation language to express database queries and updates.
- In practice, the **data definition language** and **data-manipulation languages** are not two separate languages; instead they simply form parts of a single database language,
- such as the widely used SQL language.

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 is 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
- Data Manipulation Language enables users to access or manipulate data as organized by the appropriate data model.
- The types of access are:
  - Retrieval of information stored in the database
  - Insertion of new information into the database
  - Deletion of information from the database
  - Modification of information stored in the database

Data Manipulation Language ...

- There are basically two types of data-manipulation language

  Procedural DML -- require a user to specify what data are needed and how to get those data.

  Declarative DML -- require a user to specify what data are needed without specifying how to get those data.

- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a **query language.**

SQL Query Language

- The most widely used commercial language

  Example to find all instructors in Comp. Sci. dept

  ```
  select name
  from instructor
  where dept_name = 'Comp. Sci.'
  ```

- 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 Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs** -- are programs that are used to interact with the database in this fashion.

Data Dictionary

- We can define a **data dictionary** as a DBMS component that stores the definition of data characteristics and relationships. You may recall that such "**data about data**" were labeled metadata.
- The DBMS data dictionary provides the DBMS with its self describing characteristic. In effect, the data dictionary resembles and X-ray of the company's entire data set, and is a crucial element in the data administration function.
- The two main types of data dictionary exist, **integrated** and **stand alone**. An integrated data dictionary is included with the DBMS. For example, all relational DBMSs include a built in data dictionary or system catalog that is frequently accessed and updated by the RDBMS. Other DBMSs especially older types, do not have a built in data dictionary instead the DBA may use third party stand alone data dictionary systems.

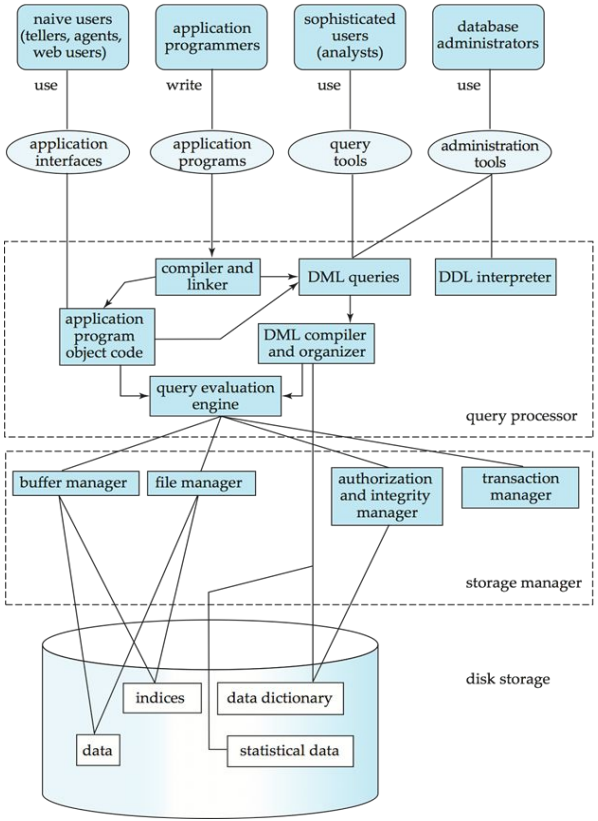
Database Design

- 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 Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
  - The storage manager,
  - The query processor component,
  - The transaction management component.

Database System Internals



Storage Manager

- A 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 for the interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

Storage Manager

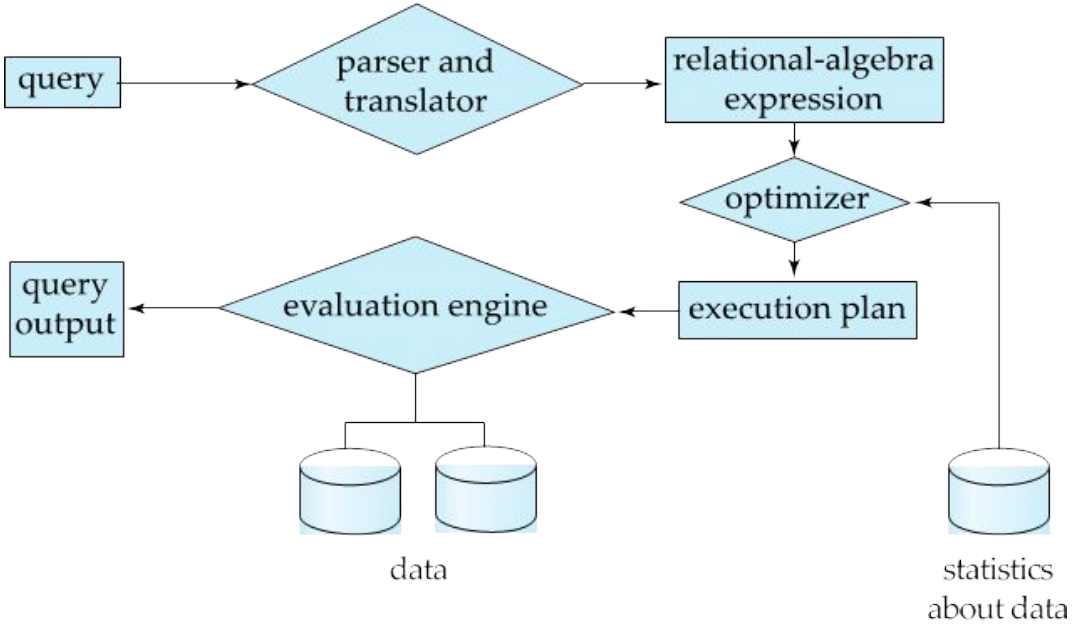
The storage manager components include:
Authorization and integrity manager, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
Transaction manager, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
File manager, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
Buffer manager, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Qwery Processor

- The query processor components include
  - **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
  - **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs query optimization, that is, it picks the lowest cost evaluation plan from among the alternatives.
  - **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



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.

Transaction manager

- A transaction is a collection of operations that performs a single logical function in a database application.
- Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.
- Transaction - manager ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.

Database Architecture

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

- **Centralized**
  - A centralized database is a database that is **located, stored, and maintained in a single location**. This location is most often a central computer or database system, for example **a desktop or server CPU, or a mainframe computer**.
- **Client-server**
  - A client-server database is **one where the database resides on a server**, and client applications are written to access the database.
- **Parallel (multi-processor)**
  - A parallel database system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries. ... Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel
- **Distributed**

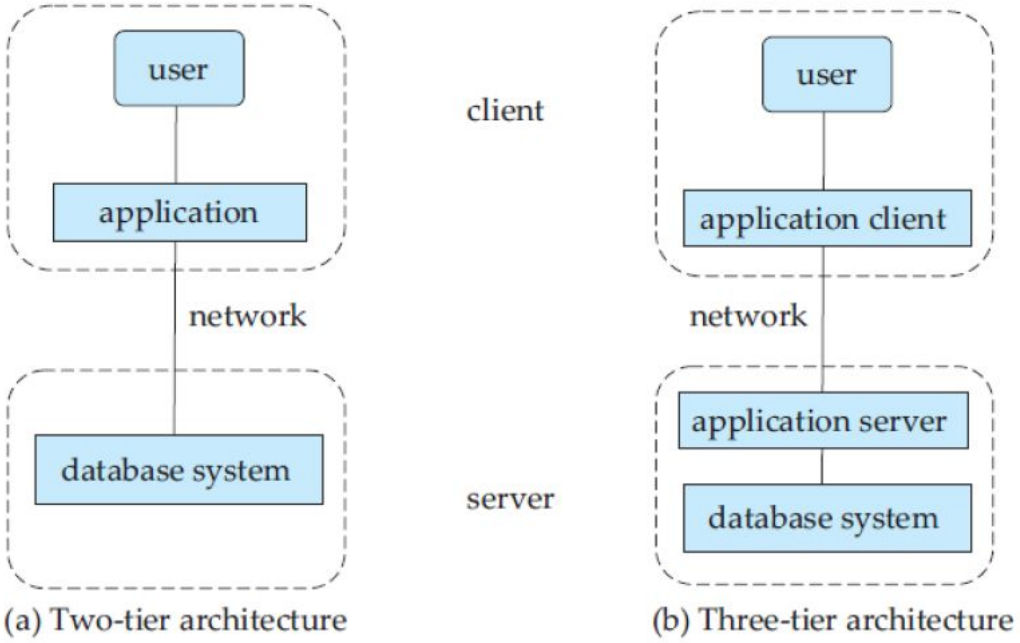
Database Architecture

- Database applications are usually partitioned into two or three parts. In a two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements.
- Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.
- In contrast, in a three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface.
- The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients.
- Three-tier applications are more appropriate for large applications, and for applications that run on the WorldWideWeb.

Database Applications

- Database applications are usually partitioned into two or three parts.
  - **Two-tier architecture** -- the application resides at the client machine, where it invokes database system functionality at the server machine
  - **Three-tier architecture** -- the client machine acts as a front end and does not contain any direct database calls.
    - The client end communicates with an application server, usually through a forms interface.
    - The application server in turn communicates with a database system to access data.

Two-tier and Three tier Architecture



(a) Two-tier architecture

(b) Three-tier architecture

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

Entity Relationship Model

- The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects.
- An entity is a “thing” or “object” in the real world that is distinguishable from other objects. The entity relationship model is widely used in database design.

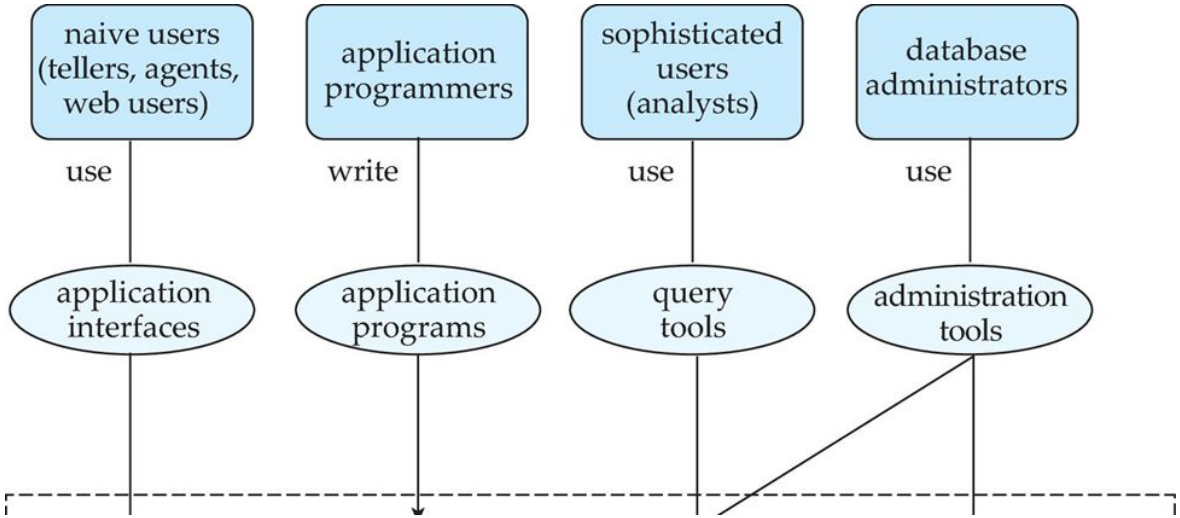
Semi Structured Data Model

- The semi-structured data model permits the specification of data where
- individual data items of the same type may have different sets of attributes.
- This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The
- Extensible Markup Language (XML) is widely used to represent semi-structured data.

Object Based Model

- Object-oriented programming (especially in Java, C++, or C#) has become the
- dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.
- The object-relational data model combines features of the object-oriented data model and relational data model.

Database Users and Administrators



Database Users

- **Naive users** -- unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- **Application programmers** -- are computer professionals who write application programs.
- **Sophisticated users** -- interact with the system without writing programs
  - using a database query language or
  - by using tools such as data analysis software.
- **Specialized users** --write specialized database applications that do not fit into the traditional data-processing framework. For example, CAD, graphic data, audio, video.

Database Administrator

A person who has central control over the system is called a **database administrator DBA**, whose functions are:

- Schema definition
- Storage structure and access-method definition
- Schema and physical-organization modification
- Granting of authorization for data access
- Routine maintenance
- Periodically backing up the database
- Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
- Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users

References

1. Database Systems Concepts, Abraham Silberschatz, Henry F. Korth, S. Sudarshan, McGraw-Hill Higher Education
2. Database Management Systems, 2nd Edition, Raghu Ramakrishnan, Johannes Gehrke