Introduction to Database (CSE3151)

Lecture - 1

Nitesh Kumar Jha, M.Tech.

niteshkumarjha@soa.ac.in

ITER,S'O'A(DEEMED TO BE UNIVERSITY)

Feb 2024

Course Info

- **Course website**: https://nitesh-jha.onrender.com/
- **Contact**:

  email - niteshkumarjha@soa.ac.in

  cabin – C-227

  webpage - https://nitesh-jha.onrender.com
- **Lecture Materials**:
  - Lecture Slides, Notes
  - Additional readings
- **Reference books**:

1. Database System Concepts, 6th Edition, by Silberschatz, Korth, Sudharsan, Tata Mc-Graw Hills Publication
2. Learning SQL: Master SQL Fundamentals, 2 Edition, by Alan Beaulieu, O Reilly.

Course Info

Course format: 3 class/week (1 hr), 1 PS/week (2 hr)

COURSE OUTCOMES

CO1	To identify and explain the different components and functionalities of DBMS and their interdependence through the database architecture
CO2	To formulate SQL queries and able to develop a database application as per user requirements using SQL and database connectivity.
CO3	To analyze an enterprise schema for given user requirements and apply the conceptual database design principles through ER modelling to construct the ER diagram and corresponding relation schemas.
CO4	To analyze and design relational database schema using decomposition and normalization techniques.
CO5	To apply relational algebra and relational calculus to express queries on relational schemas.
CO6	To interpret the functional issues related to transaction and database recovery along with the concept of storage and database system architecture.

Tentative Syllabus

Introduction to DBMS, Disadvantages of conventional FileSystem, Advantages of Database approach on File system, Three level data abstraction, Database languages: DDL, DML, DBA

Data Models: ER model, Relational Model, etc. DB system architecture(Two tier and three tier architecture). ER model: relationship set, mapping cardinalities, database keys

Foreign key, referential integrity, key attributes of relationship sets, EER features, designing an ER diagram, Relational database, Mapping ER model to relational model

Relational database design, Normalization concept, decomposition method, functional dependencies, attribute closure, extraneous attribute, canonical cover, properties of deco

Normalization (1NF, 2NF, 3NF, BCNF, Multivalued dependency, 4NF), DE-normalization, Relational query language, Relational algebra queries, Tuple relational calculus, Domain relational calculus

Transaction ACID properties, states, scheduling, serializability, recoverability, concurrency, control, two-phase locking protocol, multi-version scheme, recovery system, recovery in deferred database

B+ tree indexing, Hash function, hashed indexing, Query processing, Query optimization, Database system architectures.

Grading Pattern

Grading Pattern 1	ATTENDANCE	**5**
	ASSIGNMENTS/Quiz	20
	MID TERM	15
	TOTAL INTERNAL	**40**
	Final Quiz	15
	THEORY EXAM	45
	TOTAL EXTERNAL	**60**
	TOTAL	**100**

Introduction to Database

What is What?

- Data
  - Set of values representing some information.
  - Ex: age is 21 years, blue shirt, today's temp. is 30°C
  - Salary of ₹20,000, height of john 6'2", . . .
- Database (DB)
  - Is a collection of interrelated data (pertaining to one organization or business house) organized in a meaningful way.
- Database Management System (DBMS)
  - Is a collection of interrelated data and a set of programs to store/retrieve those data.

DBMS Package

- A software **package** designed to define, manipulate, retrieve and manage data in a database in a user friendly environment

MySQL

Microsoft SQL Server

Microsoft Access

PostgreSQL

Firebird

ORACLE®

SQLite

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

Applications

- 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

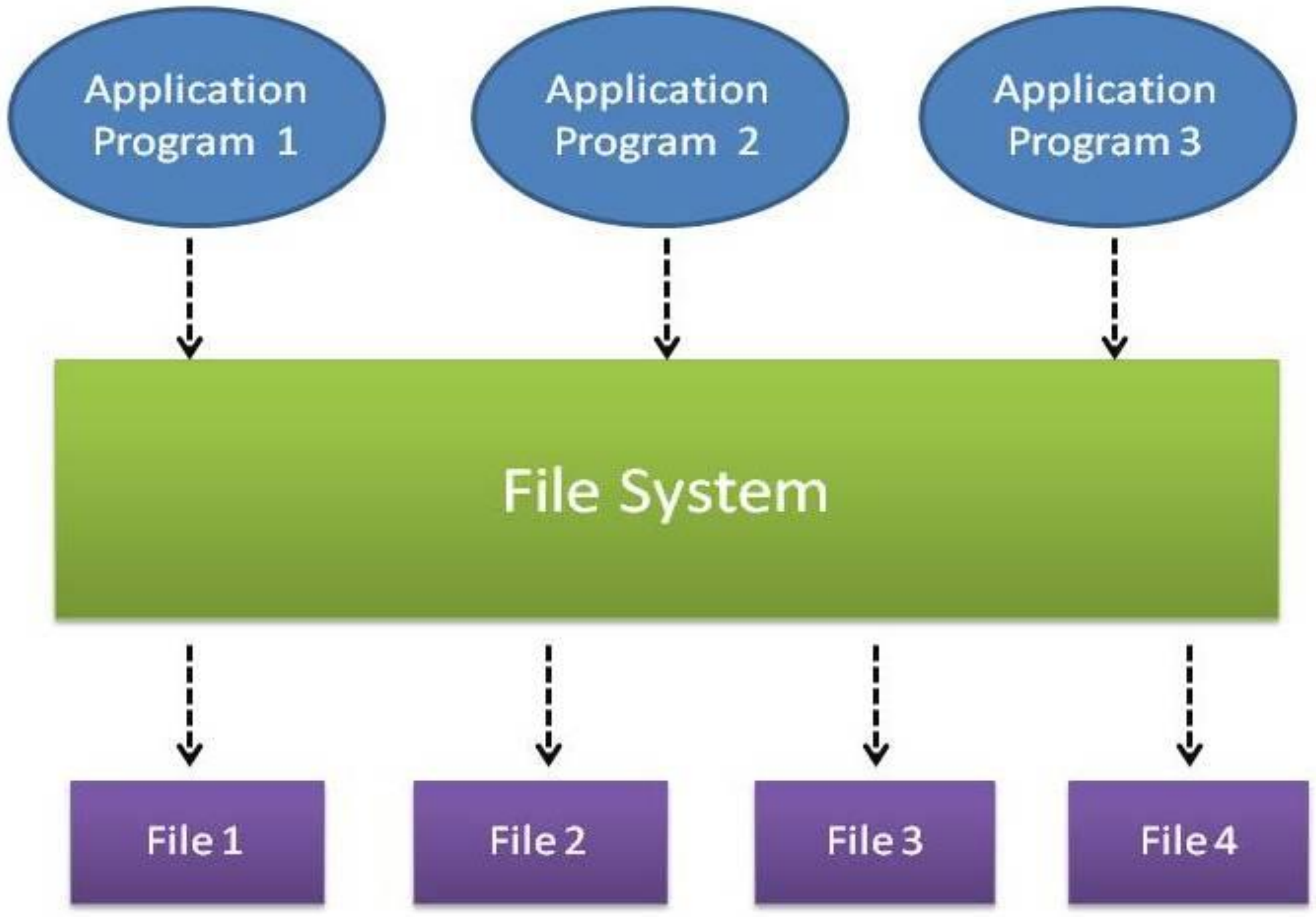
Database Use

- In early days, people used to interact with the database indirectly
  - Printed reports such as credit card statements, bank teller, reservation agents
- Currently, people interact with the database directly
  - ATM
  - E-Shopping
  - E-Banking
  - E-Application

University Database Example

- 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
- In the early days, database applications were built directly on top of file systems

Traditional Approach-file system



Drawbacks of using file systems-I

- Data redundancy and inconsistency
  - Multiple file formats: due to multiple programmers over a period of time.
  - Duplication of information in different files: A student record (name , regno, address, …) is maintained in CSIT, Maths, Phy, ECE depts.
  - If address info is changed but not reflected every where, then data becomes inconsistent.
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
  - Ex: find students from out side odisha?

Drawbacks of using file systems-II

- Data isolation
  - Data scattered in various files possibly of different formats. Writing application program to retrieve required data becomes impossible.
- Integrity problems
  - Integrity constraints are in program code rather than being stated explicitly with data e.g.,
    - account balance > 0
    - Reg No can't be blank
    - Age can't be a negative number
    - etc.
  - Hard to add new constraints or change existing ones as it needs application program modification.

Drawbacks of using file systems-III

- Atomicity problems (of updates)
  - Failures may cause an inconsistent state with partial updates carried out.
  - Ex: Transfer of funds from one account to another should either complete or not happen at all (debited but not credited)
- Concurrent access anomalies (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
  - i.e. defining user roles

Database systems offer solutions to all the above problems

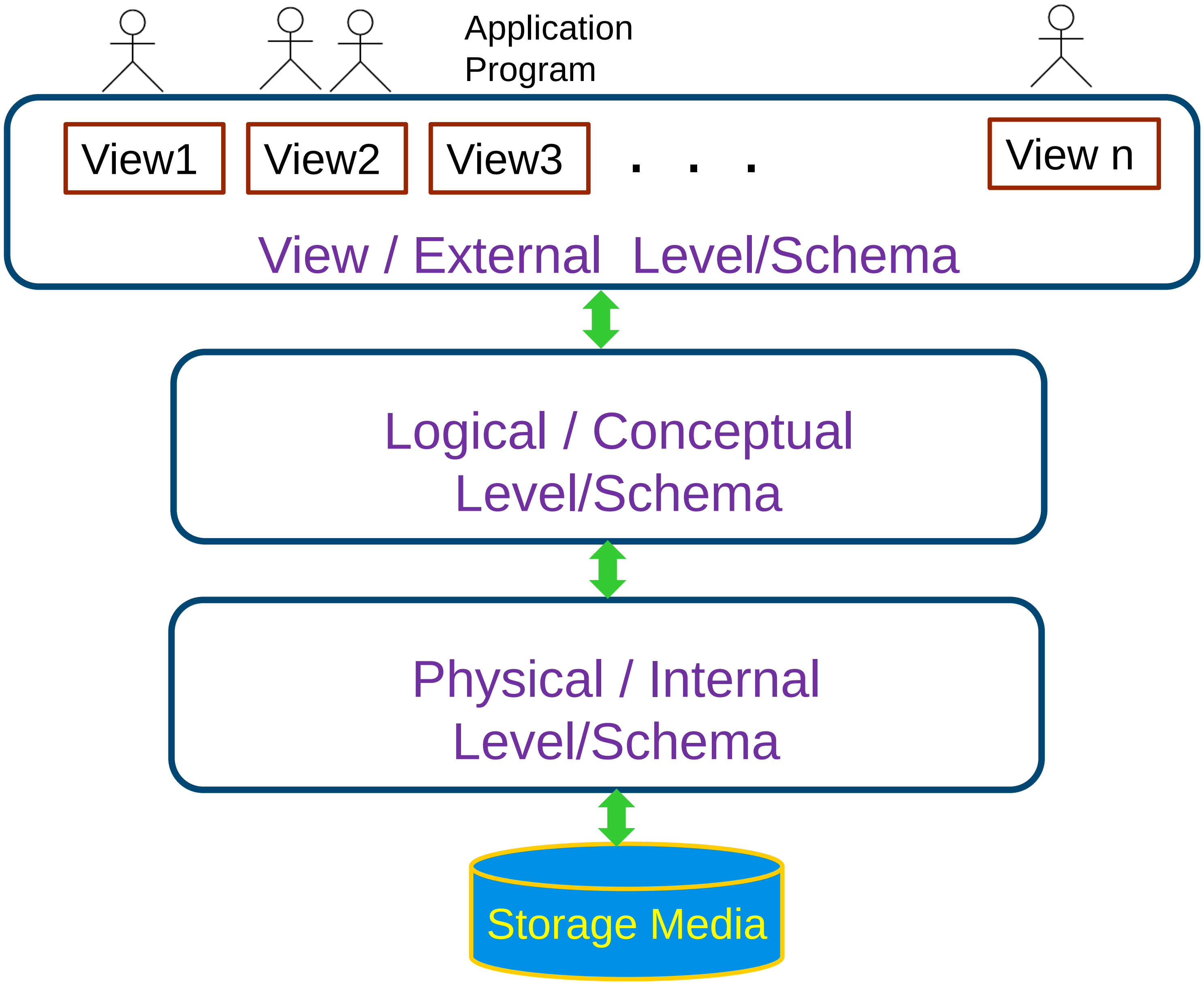
Data Abstraction

- A major purpose of database system is to provide users with an abstract view of the data.
- That is the system hides certain details how the data are stored and maintained.
- The abstraction helps avoiding mishandling of data by normal database users and makes it simple and
- Makes it convenient to access and understand the information

Example Abstraction

- Map
  - is an abstraction of a geographical region (easy to understand the world)
- Riding a bike
  - No abstraction: Read and understand the mechanism of IC engine, gear mechanism, brake system, electrical section . . .
  - Abstraction: Understand only the use of accelerator, gear and brake leavers. (life is simple)

Three Schema architecture of DBMS



Data abstraction in Database

- Database system provide three different levels of abstraction
- **Physical level / Internal Schema**:
  - The lowest level that describes how data are actually stored.
  - It describes the complex low-level data structures in detail.
  - Used by package developers

Levels of Abstraction

- **Logical level / Conceptual Schema:** describes what data are stored in database, and the relationships among the data.

  type *instructor* = **record**

  > *ID* : string;
  > *name* : string;
  > *dept_name* : string;
  > *salary* : integer;

  end;

- Deals with simple structures (tables) to store database information.
- Database administrators use this abstraction without knowing the details of complex physical structure called physical data independence.

Levels of Abstraction

- **View level / External Schema:** highest level of abstraction that presents a specific portion of the database view for different users.
- This level may contain many different views of the database required by different users.
- All these views are mapped on a single unified conceptual level.
- Application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.
- User access layer that is used to simplify the interaction between a user and the database.

Instances and Schemas

- **Database Instance** (variables values)
  - The collection of database information at a given time is called the instance of the database of that time. (Actual content of the database)
  - Ex: Reg: 001, Name: john, Branch: CSIT, Sem: 4, . . .
  - With database operations like, insertion, deletion, updation, the instance changes
  - Ex: Reg: 001, Name: john, Branch: CSIT, Sem: 5, . . .
  - Reg: 002, Name: Smith, Branch: CSIT, Sem: 5, . . .
- **Database Schema** (types, variables)
  - It refers to the overall design of the database
  - Depending upon the level of abstraction, there exists 3 types of database schemas

Three level Schemas

- **Physical schema**– the overall physical structure of the database
- **Logical Schema** – the overall logical structure of the database
  - Ex: 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
- **External Schema / View** – At the external level, we find multiple views of the database referred as sub-schemas
- Schema changes infrequently than instances.

Data Independence

- **Data Independence** – Any changes made to the lower-level schema does not affect the higher level schema.
- **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.
- **Logical Data Independence** – Any change made to the conceptual level schema, do not require any change modification to the external schema.

Database Language

- The language used for database is broadly classified into two categories of languages.
- **Data Definition Language (DDL)**– basically used for defining/modifying the logical schema.
- **Data Manipulation Language (DML)** – used to access and manipulate the database instance.
- In practice, these are not two separate languages, rather they are the two parts of the same commercial database language called Structural Query Language (SQL)
- That is SQL integrates all

Data Definition Language (DDL)

- Specification notation for defining the database schema (create, alter, drop, rename, …)

  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
  - Operations supported: insert, delete, update, retrieve
- 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-II

- DML can be of two types
- **Procedural DML**: Requires user to specify what data are needed and how to get those data?
- **Declarative / non-procedural DML**: Requires user to specify what data are needed? without specifying how to get those data.
  - It is easy to use and popular
  - When DML is executed, the meta data present in the data dictionary are referred to check the validity of the intended DML operation.

Database Users

- **Naive users**
  - The don't have much knowledge on databases
- **Application Programmers**
  - Those have developed the application.
- **Sophisticated users**
  - They are not direct users, they are analyst, knowledge workers.
- **Specialized users** (Database administrators)
  - One of the primary objective of database system is to have centralized control over the entire database and the application programes. It is achieved by database administrators.

Database Administrators (DBA)

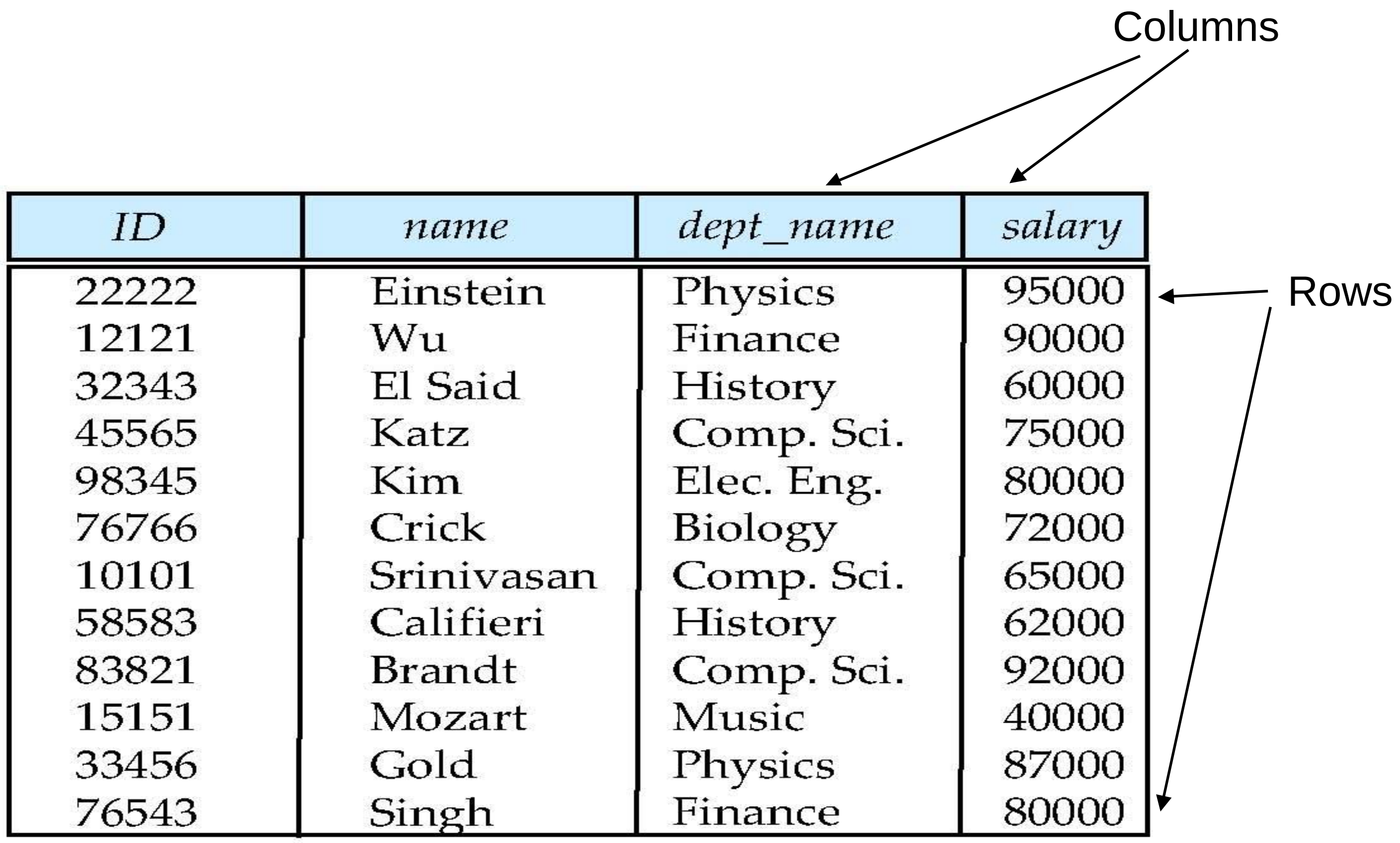
- DBA may be a person or a group of persons in an organization acts as the manager of the database and take full responsibility of the database of that organization.
- The important roles and responsibilities of DBA are
  - Defining the Database schema
  - Defining the storage structure and access methodology
  - Modifying the database schema and storage organization
  - Granting authorization and ensuring security
  - Monitoring the database performance
  - Routine maintenance of database

Data Model

- A collection of conceptual tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- The data models can be classified into following four different categories
  - Relational model
  - Entity-Relationship data model (mainly for database design)
  - Object-based data models (Object-oriented and Object-relational)
  - Semi-structured 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

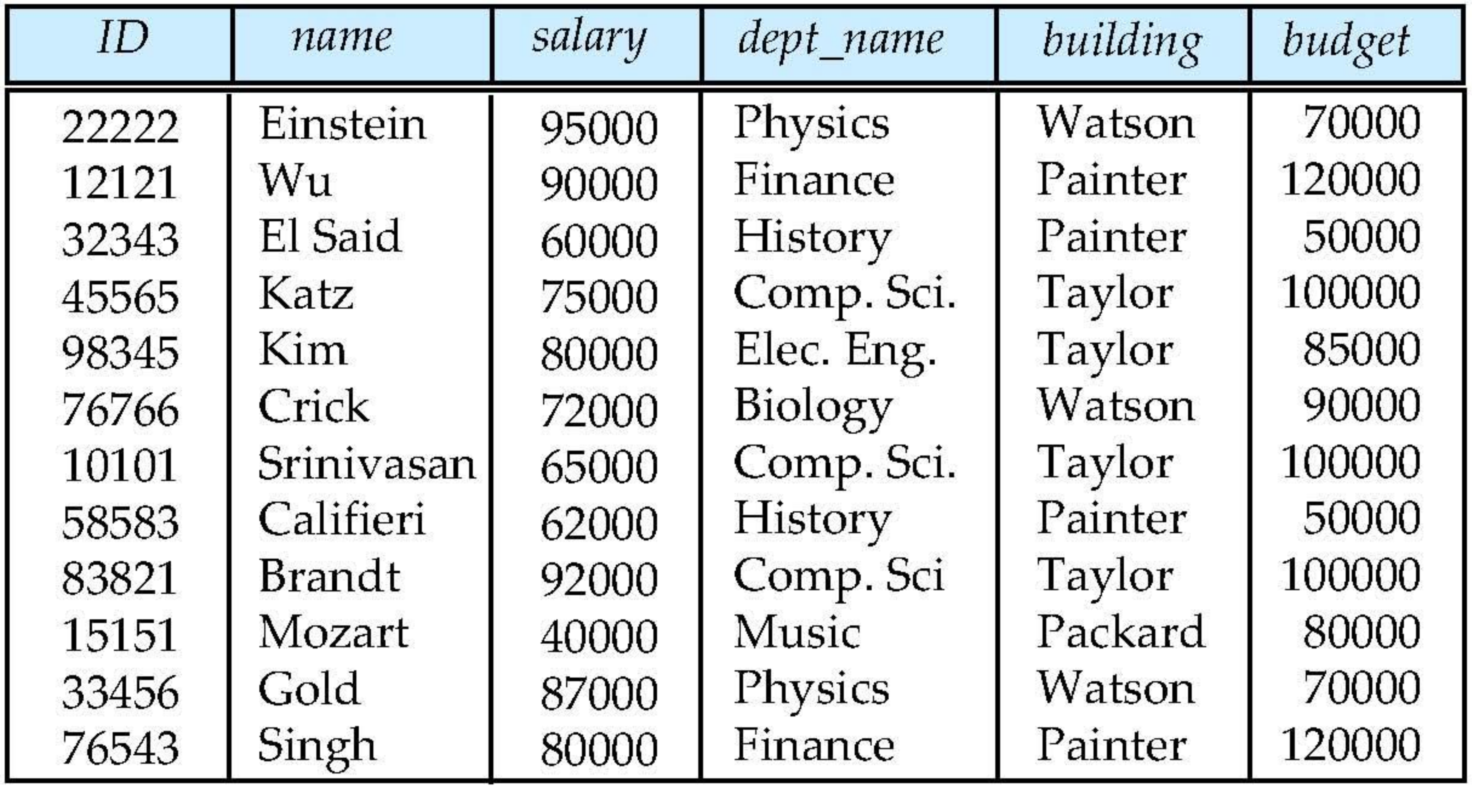

Columns
Rows

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

Database Design (Cont.)

- Is there any problem with this



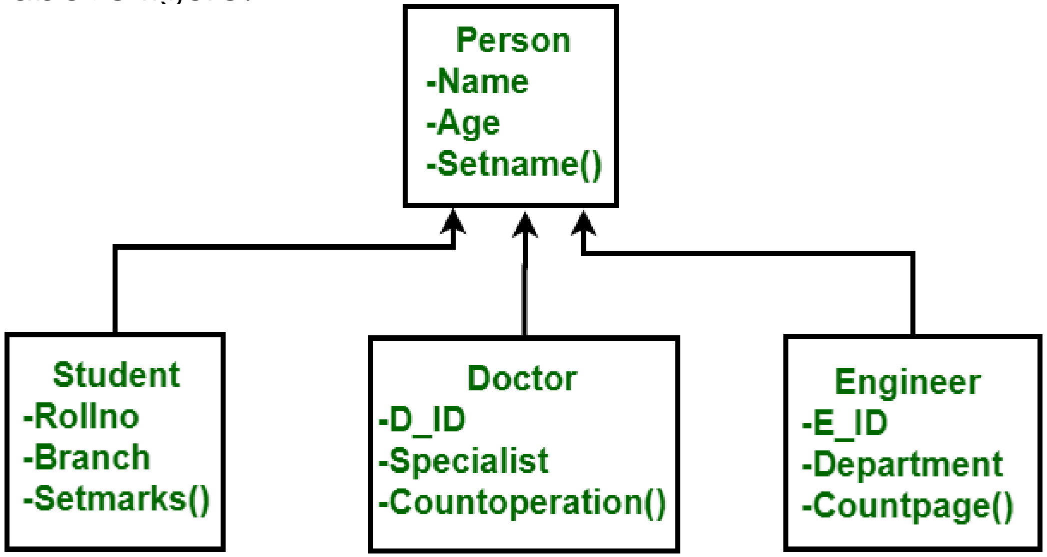
ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	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

Object-Relational Data Models

- To represent the complex real world problems there was a need for a data model that is closely related to real world. Object Oriented Data Model represents the real world problems easily.
- Object Relational Data Model: In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object in this data model. In this, real world problems are represented as objects with different attributes. All objects have multiple relationships between them. Basically, it is combination of Object Oriented programming and Relational Database Model.

Object-Relational Data Models

- Objects – An object is an abstraction of a real world entity or we can say it is an instance of class. Objects encapsulates data and code into a single unit which provide data abstraction by hiding the implementation details from the user. For example: Instances of student, doctor, engineer in above figure.
- Attribute – An attribute describes the properties of object. For example: Object is STUDENT and its attribute are Roll no, Branch, Setmarks() in the Student class.
- Methods – Method represents the behavior of an object. Basically, it represents the real-world action. For example: Finding a STUDENT marks in above figure as Setmarks().
- Class – A class is a collection of similar objects with shared structure i.e. attributes and behavior i.e. methods. An object is an instance of class. For example: Person, Student, Doctor, Engineer in above figure.



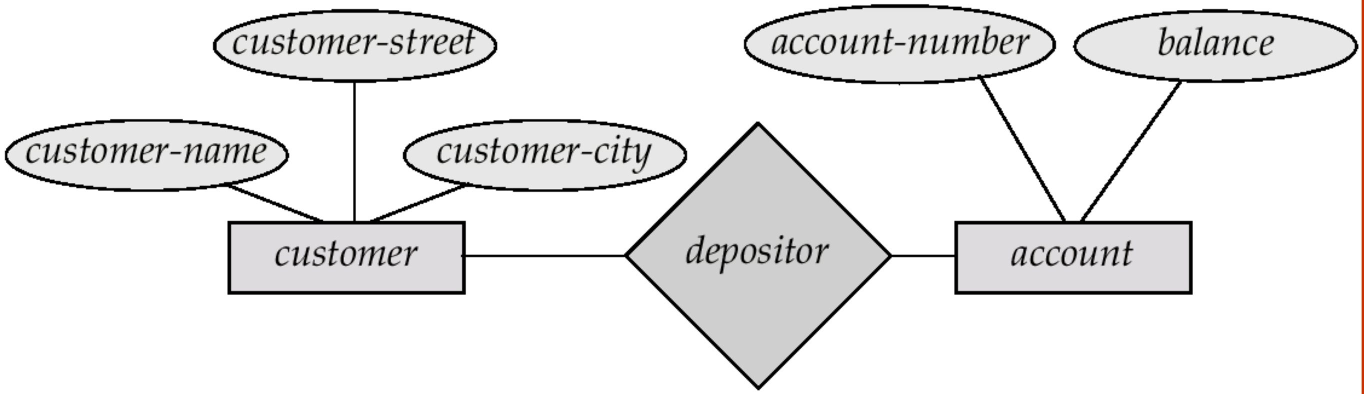
Semi structured model: XML, JSON

- Semi-structured data is a type of data that is not purely structured, but also not completely unstructured. It contains some level of organization or structure, but does not conform to a rigid schema or data model, and may contain elements that are not easily categorized or classified.
  - Semi-structured data is typically characterized by the use of metadata or tags that provide additional information about the data elements. For example, an XML document might contain tags that indicate the structure of the document, but may also contain additional tags that provide metadata about the content, such as author, date, or keywords.
  - Other examples of semi-structured data include JSON, which is commonly used for exchanging data between web applications, and log files, which often contain a mix of structured and unstructured data.

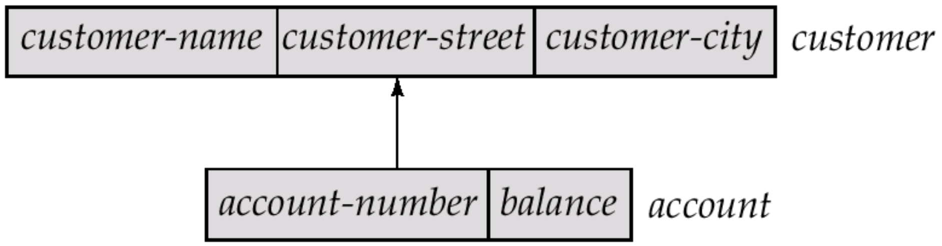
Hierarchical model

- A hierarchical database consists of a collection of *records* which are connected to one another through *links*.
- a record is a collection of fields, each of which contains only one data value.
- A link is an association between precisely two records.
- The hierarchical model differs from the network model in that the records are organized as collections of trees rather than as arbitrary graphs.
- The schema for a hierarchical database consists of
  - boxes, which correspond to record types
  - lines, which correspond to links
- Record types are organized in the form of a rooted tree.
  - No cycles in the underlying graph.
  - Relationships formed in the graph must be such that only
    one-to-many or one-to-one relationships exist between a parent and a child.

Single Relationships

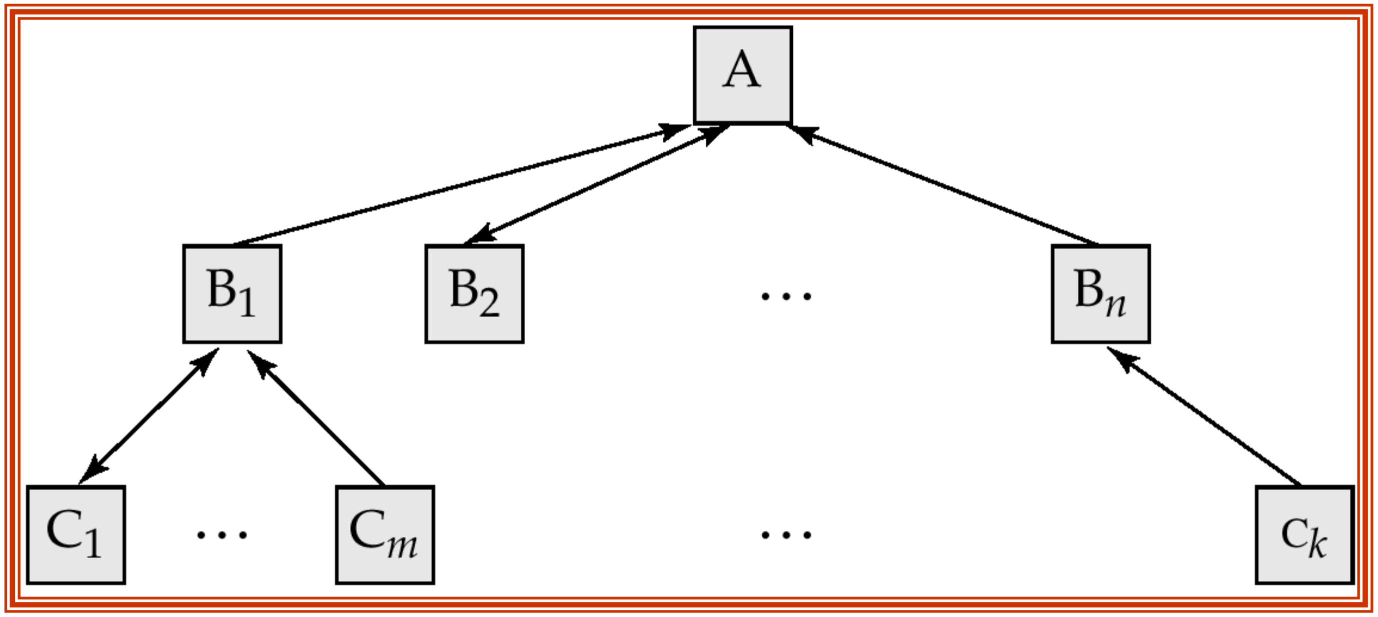


(a) E-R diagram



(b) Tree-structure diagram

General Structure



- A parent *may* have an arrow pointing to a child, but a child *must* have an arrow pointing to its parent.

Network model

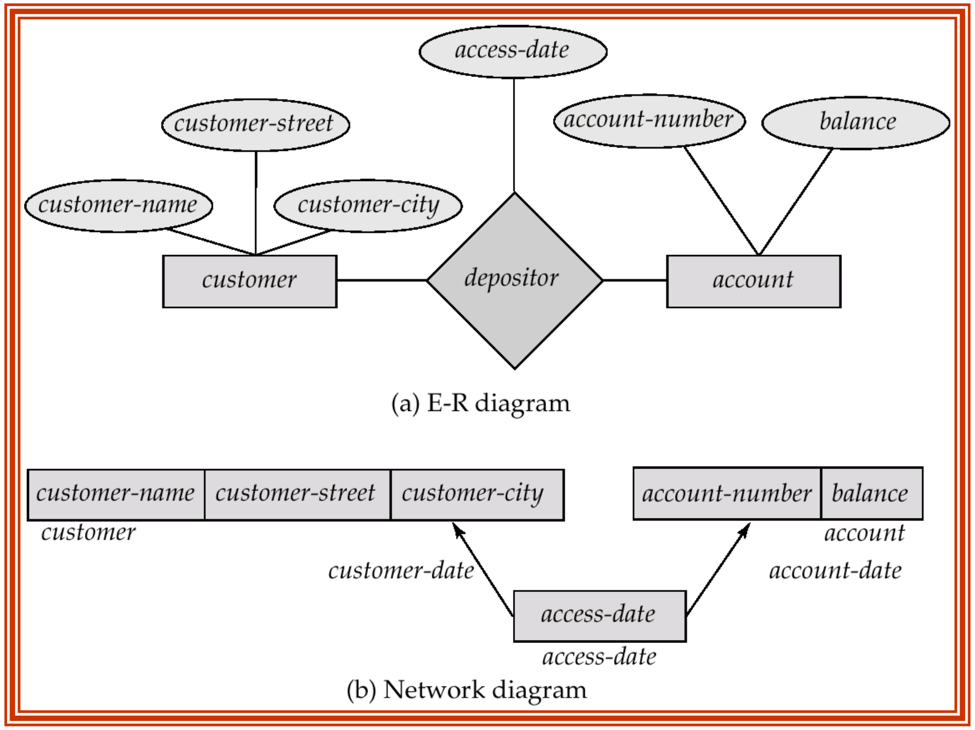
- It is the advance version of the hierarchical data model. To organize data it uses directed graphs instead of the tree-structure. In this child can have more than one parent. It uses the concept of the two data structures i.e. Records and Sets.
- Data are represented by collections of records.
  - similar to an entity in the E-R model
  - Records and their fields are represented as record type

-
  ```
  type customer = record            type     account = record
        customer-name: string;               account-number: integer;
        customer-street: string;             balance: integer;
        customer-city: string;
  ```

-
  ```
  end                                end
  ```
- Relationships among data are represented by links
  - similar to a restricted (binary) form of an E-R relationship
  - restrictions on links depend on whether the relationship is many-many, many-to-one, or one-to-one.

Data-Structure Diagrams (Cont.)

- Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links.



Overall DBMS Architecture

The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The query processor is important because it helps the database system to simplify and facilitate access to data. The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of the system. It is the job of the database system to translate updates and queries written in a non-procedural language, at the logical level, into an efficient sequence of operations at the physical level.

The storage manager is the component of a database system that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

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.

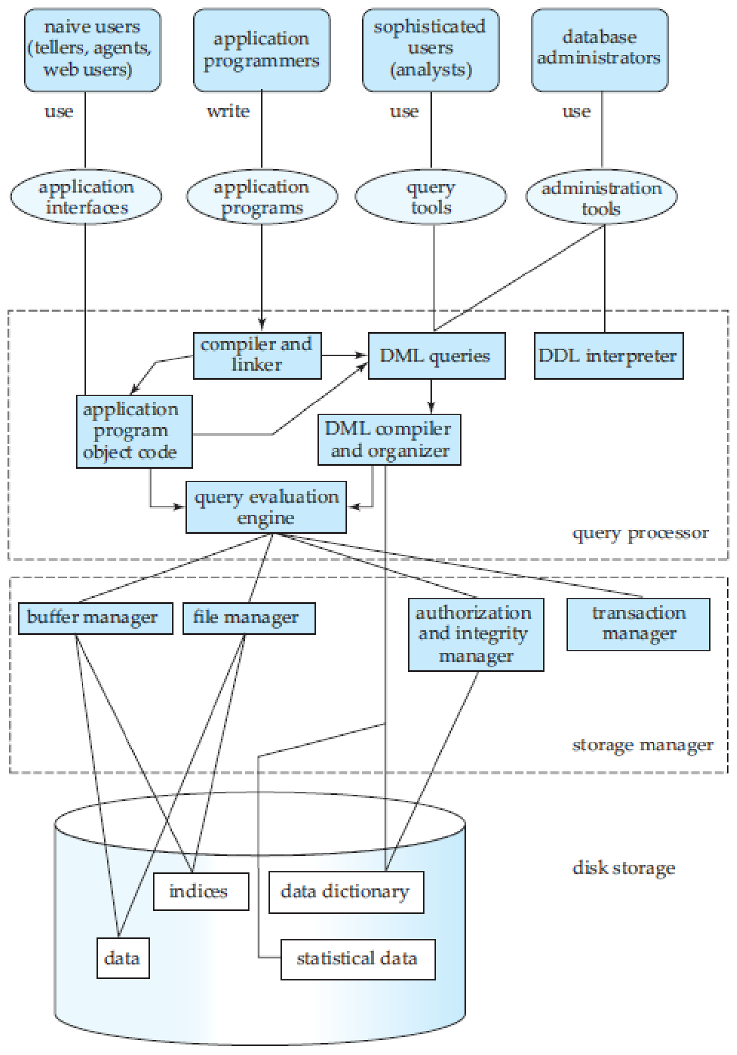


Figure 1.5 System structure.

Two-tier and Three-tier Architecture

Database applications are usually partitioned into two or three parts, as in Below Figure 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.

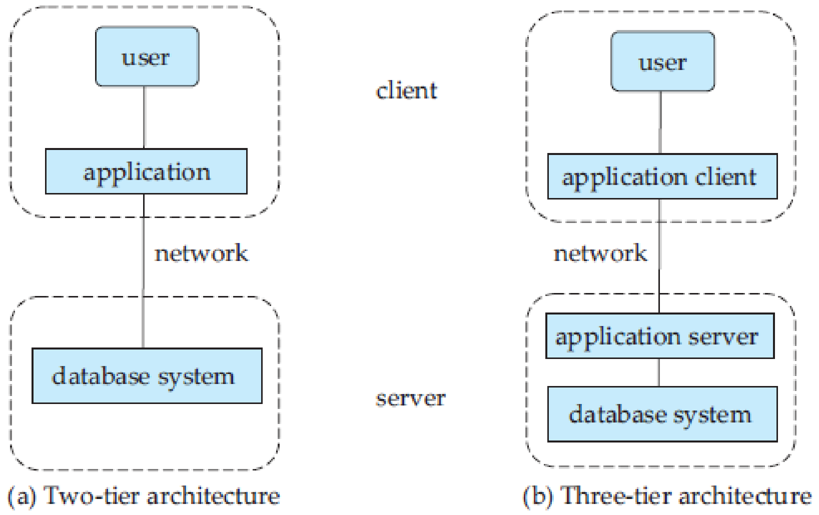


Figure 1.6 Two-tier and three-tier architectures.

End of Chapter 1

Thank You