Data base Management Systems

Whether you know it or not,
you’re using a database every day

Introduction

Dr. Parul Madan

Associate Professor

CSE Dept, Graphic Era University, Dehradun

What is Data?

- **Data** is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed. For example: When you visit any website, they might store you IP address, that is data, in return they might add a cookie in your browser, marking you that you visited the website, that is data, your name, it's data, your age, it's data.
- Data becomes **information** when it is processed, turning it into something meaningful. Like, based on the cookie data saved on user's browser, if a website can analyse that generally men of age 20-25 visit us more, that is information, derived from the data collected.

What is a Database?

- A **Database** is a collection of related data organised in a way that data can be easily accessed, managed and updated. Database can be software based or hardware based, with one sole purpose, storing data.
- During early computer days, data was collected and stored on tapes, which were mostly write-only, which means once data is stored on it, it can never be read again. They were slow and bulky, and soon computer scientists realised that they needed a better solution to this problem.
- **Larry Ellison**, the co-founder of **Oracle** was amongst the first few, who realised the need for a software based Database Management System.

What is DBMS?

- A **DBMS** is software that allows creation, definition and manipulation of database, allowing users to store, process and analyse data easily. DBMS provides us with an interface or a tool, to perform various operations like creating database, storing data in it, updating data, creating tables in the database and a lot more.
- DBMS also provides protection and security to the databases. It also maintains data consistency in case of multiple users.
- Here are some examples of popular DBMS used these days:
- MySql
- Oracle
- SQL Server
- IBM DB2
- PostgreSQL
- Amazon SimpleDB (cloud based) etc.

Characteristics of Database Management System

- A database management system has following characteristics:
- **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
- **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows **Normalisation** which divides the data in such a way that repetition is minimum.
- **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
- **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
- **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
- **Security:** The DBMS also takes care of the security of data, protecting the data from un-authorised access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
- DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

Advantages of DBMS

- Segregation of application program.
- Minimal data duplicity or data redundancy.
- Easy retrieval of data using the Query Language.
- Reduced development time and maintenance need.
- With Cloud Datacenters, we now have Database Management Systems capable of storing almost infinite data.
- Seamless integration into the application programming languages which makes it very easier to add a database to almost any application or website.

Disadvantages of DBMS

- It's Complexity
- Except MySQL, which is open source, licensed DBMSs are generally costly.
- They are large in size.

Applications of DBMS

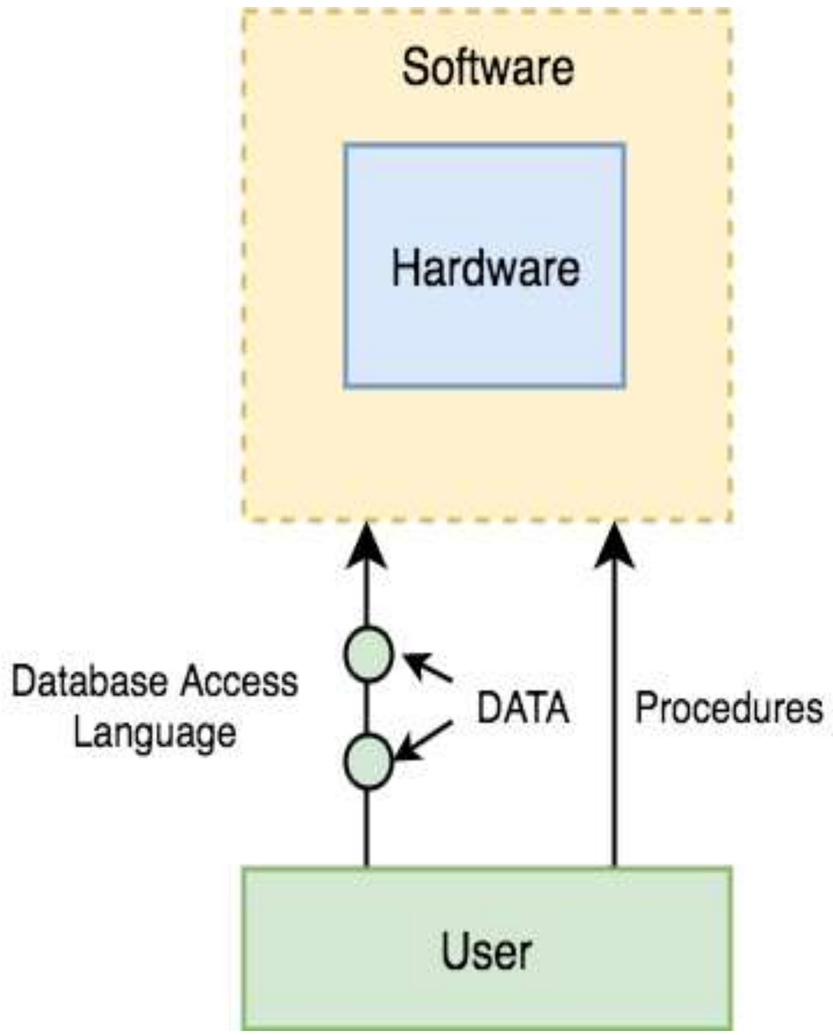
Database is widely used. The some of the representative applications are:

- **Banking:** for customer information, accounts and loans and banking transactions.
- **Universities:** for student registrations and grades.
- **Online shopping:** Everyone wants to shop from home. Everyday new products are added and sold only with the help of DBMS. Purchase information, invoice bills and payment, all of these are done with the help of DBMS.
- **Airlines:** for reservations and schedule information.
- **Credit Card Transactions:** for purchases on credit cards and generation of monthly statements.
- **Library Management System:** maintain all the information relate to book issue dates, name of the book, author and availability of the book.
- **Telecommunications:** for keeping records of call made, generating monthly bills, maintaining balances on prepaid calling cards.
- **Sales:** for customer, product and purchase information.
- **Finance:** for storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
- **Manufacturing:** for management of supply chain and for tracking production of items in factories, inventories of items and orders for items.
- **Human Resource:** for information about employees, salaries, payroll taxes and benefits.

Components of DBMS

- The database management system can be divided into five major components, they are:
- Hardware
- Software
- Data
- Procedures
- Database Access Language
- Let's have a simple diagram to see how they all fit together to form a database management system.

Cont….


Software
Hardware
Database Access Language
DATA
Procedures
User

Cont..

DBMS Components: Hardware

- When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.
- When we run Oracle or MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

DBMS Components: Software

- This is the main component, as this is the program which controls everything. The DBMS software is more like a wrapper around the physical database, which provides us with an easy-to-use interface to store, access and update data.
- The DBMS software is capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the DB.

Cont..

DBMS Components: Data

- Data is that resource, for which DBMS was designed. The motive behind the creation of DBMS was to store and utilize data.
- In a typical Database, the user saved Data is present and **meta data** is stored.

Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it.

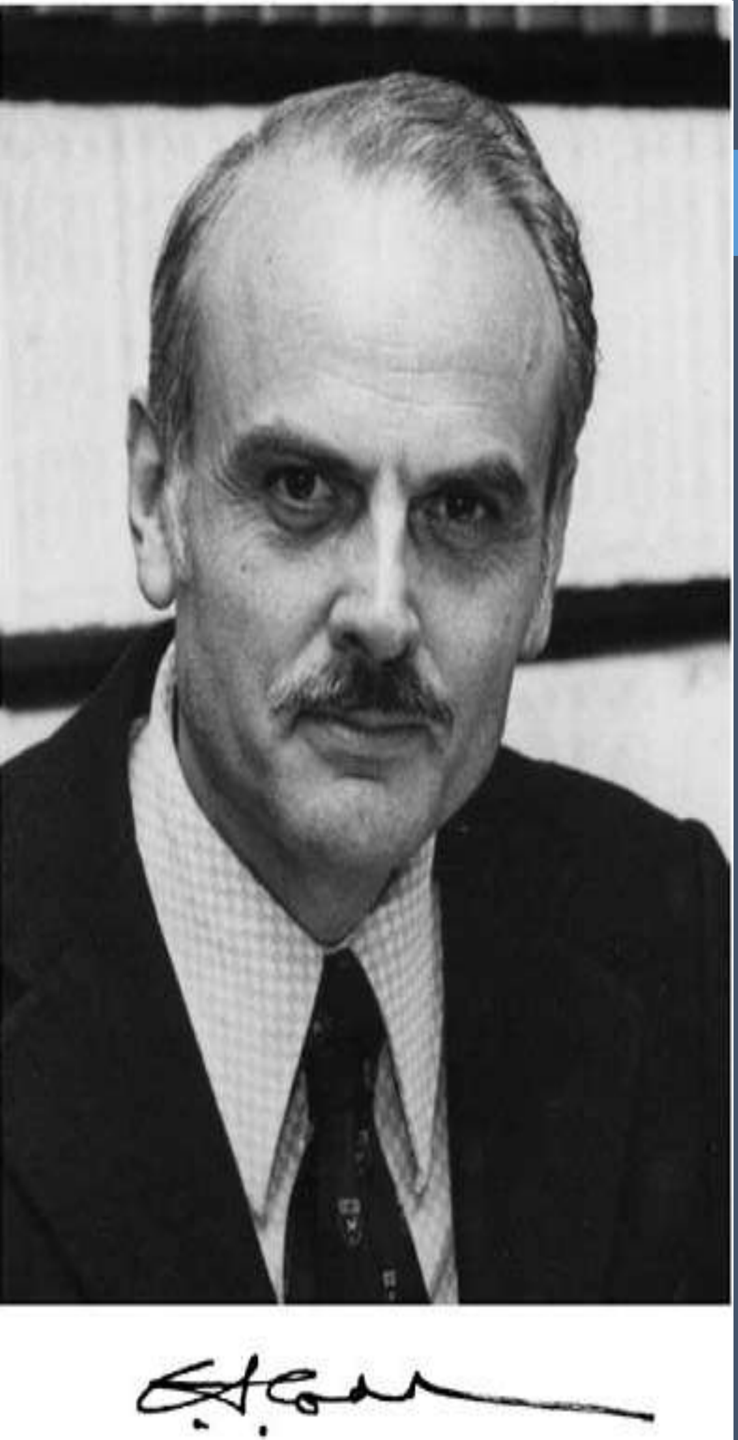
- **For example:** When I store my **Name** in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata.

DBMS Components: Procedures

- Procedures refer to general instructions to use a database management system. This includes procedures to setup and install a DBMS, To login and logout of DBMS software, to manage databases, to take backups, generating reports etc.

DBMS Components: Database Access Language

- Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database.
- A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS.
- User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.



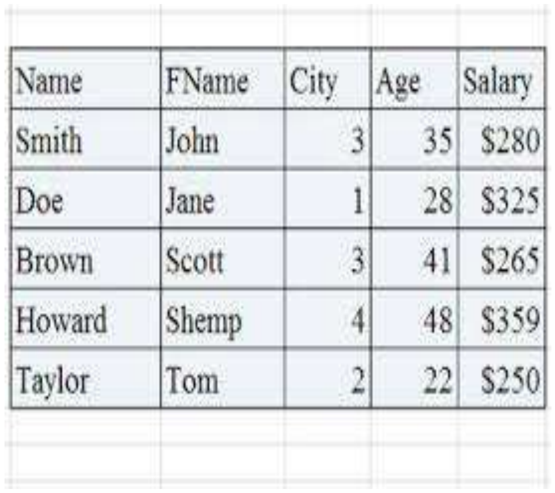
Dr. E. F. Codd’s Rules for RDBMS

Dr. E.F. Codd was an IBM researcher who first developed the relational data model in 1970. In 1985 Dr. E.F. Codd published a list of 12 rules that define an ideal relational database and has provided a guide line for the design of all relational database systems.

Dr. Codd has used the term **guideline** because till date no commercial relational database system fully conforms to all 12 rules.

Rule 1: The Information Rule

All Data should be presented in table form.



Name	FName	City	Age	Salary
Smith	John	3	35	$280
Doe	Jane	1	28	$325
Brown	Scott	3	41	$265
Howard	Shemp	4	48	$359
Taylor	Tom	2	22	$250

Rule No. 2: Guaranteed Access Rule

All Data should be accessible without ambiguity. This can be accomplished through the combination of table name, primary key, and column name.

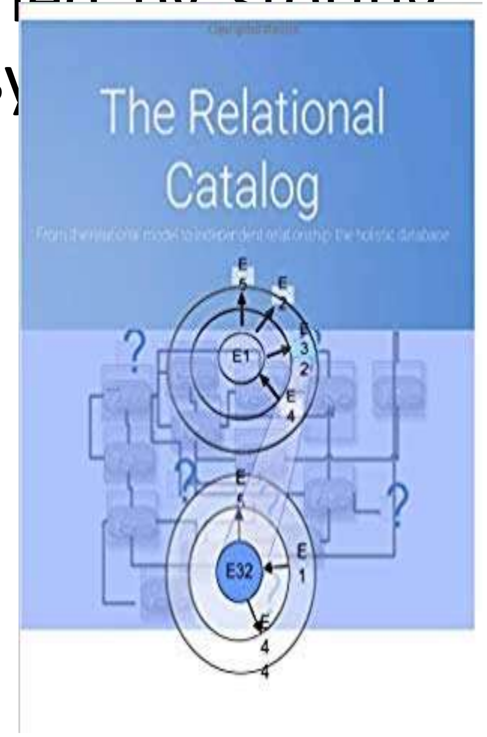
GuaranteedAccess

Rule No. 3: Systematic treatment of null values

A field should be allowed to remain empty. This involves the support of null value, which is distinct from an empty string or a number with a value of zero. Of course this can not apply to primary keys.

Rule No. 4: Dynamic online catalog based on the relational model

A relational database must provide access to its structure through the same tools that are used to access data. This is usually accomplished by storing the structure definition within special sy

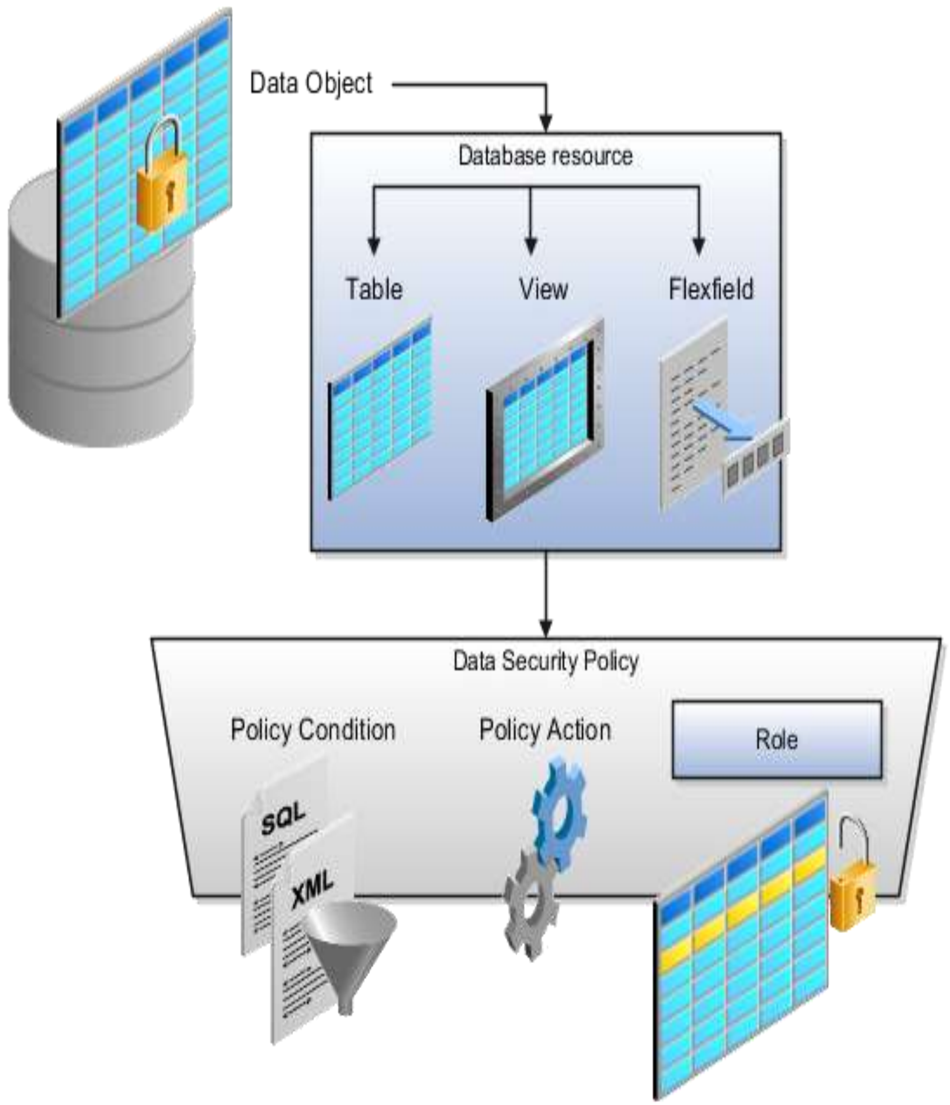


Rule No. 5: Comprehensive Data sub-language Rule

The database must support at least one clearly defined language that includes functionalities for data definition, data manipulation, data integrity and database transaction control. All commercial and free databases use forms of standard SQL as their supported comprehensive language.

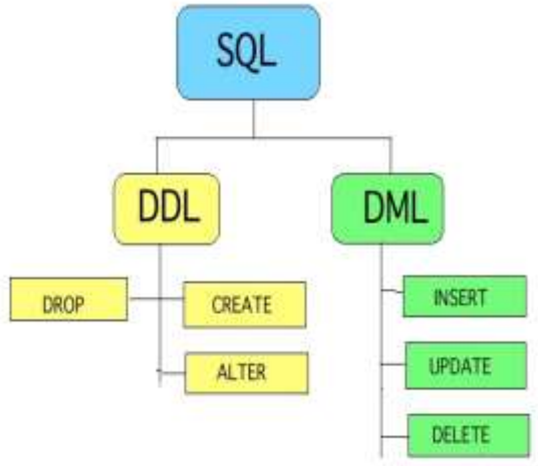
Rule No. 6: View Updating Rule

Data can be presented in different logical combinations called views. Each view should support the same full range of data manipulation that has direct access to a table available.



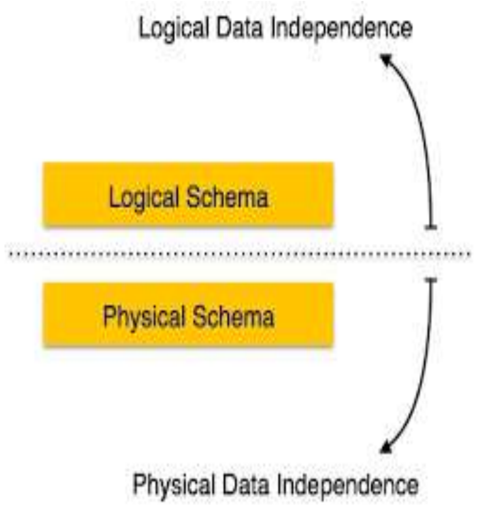
Rule No. 7: High-level Insert, Update, and Delete

Data can be retrieved from a relational database in sets constructed of data from multiple rows and /or multiple tables. This rules states that insert, update and delete operations should be supported for any retrievable set rather than just for a single row in a single table.



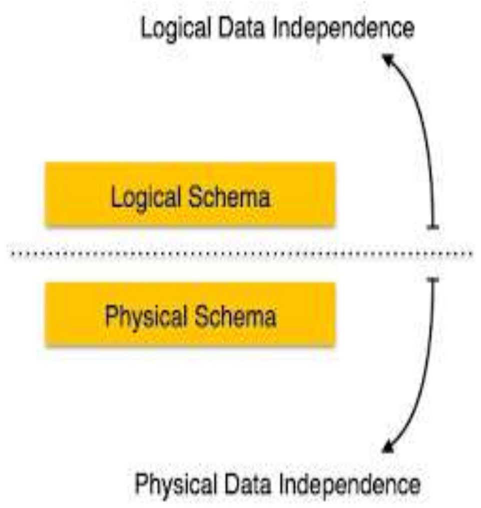
Rule no. 8: Physical Data Independence

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accessed it.



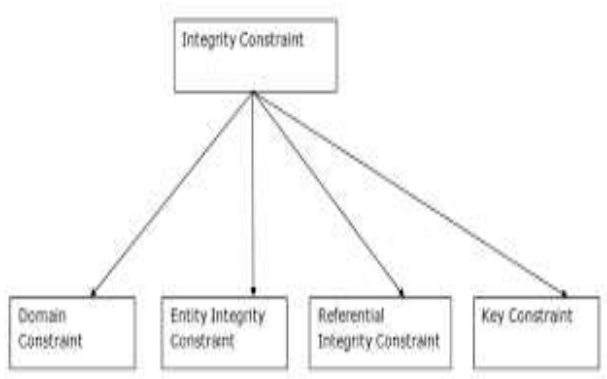
Rule No. 9: Logical Data Independence

How data is viewed should not be changed when the logical structure (table's structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the data viewed and the actual structure of the underlying tables.



Rule 10: Integrity Independence

The database language (like SQL) should support constraints on user input that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all the databases do preserve two constraints through SQL. No components of a primary key can have a null value. If a foreign key is defined in one table, any value in it must exist as a primary key in another table.



Rule no. 11: Distribution independence

A user should be totally unaware of whether the database is distributed(parts of the database exist in multiple locations) or not. A variety of reasons make this rule difficult to implement.

Rule no. 12: Non subversion Rule

There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure.

Database system Vs File System

	File System ***(like data in text file as in C programs)***	**Database System**
Data redundancy (*lead to higher storage and access cost*)	Same information is duplicated in many places eg. Mobile and email id of student in cse_student_detail records and in Btech_student_detail records as well.	Data redundancy can be deal by creating a separate table having student#, mob & email.
Difficulty in Accessing data (*Need to write a new program to carry out each new task*)	If user need all student having GPA>3.5 then we need to create such program or if this program exist then also file system can meet with changing needs as need student having GPA>3.5 and are from Delhi.	Searching required data is easy by writing small queries system get adapted to changing needs

	File System (*like data in text file as in C programs*)	**Database System**
Data Isolation	As data are scattered in different files and file may be of different formats as some in .doc or .txt or .xls need require coding for each of them.	Uniformity in the way data is stored
Integrity problems	Data values must follow some c**onsistency constraint** such as no account should have less than Rs 5000. in File System we need to code it and if in future we want to change it we need to recode it! • *Integrity constraints (e.g. account balance >* 5000*) become part of program code* • *Hard to add new constraints or change existing ones*	New constraint can easily be add, modify & drop easily.
Inconsistency and Atomicity Problem	Computer systems are prone to failures. Suppose a program transferring Rs 5000 from account A to B but in middle system crash then Rs 5000 was removed from but not credited to B. This lead to *inconsistent state*. It is difficult to ensure atomicity in File system. • *Transfer of funds from one account to another should either complete or not happen at all*	Atomicity can easily be maintained, these system have recovery and back up tools.

	File System *(like data in text file as in C programs)*	**Database System**
Concurrent - access anomalies	Consider an account A holding $500, if two customer C1 & C2 withdraw $100 and $50 from A simultaneously then initially C1 & C2 see $500 now whichever write last it either show $400 or $450 while correct is $350. Hence File system have greater challenge as many application programs access same data simultaneously in multi-user system.	Support multi-user system.
Security problem	Enforcing security constraint is difficult. As faculty can upload and see attendance of student, whereas only *class advisor* can modify already uploaded attendance and *students* can only see their own attendance.	Database has internal procedures and commands for this.

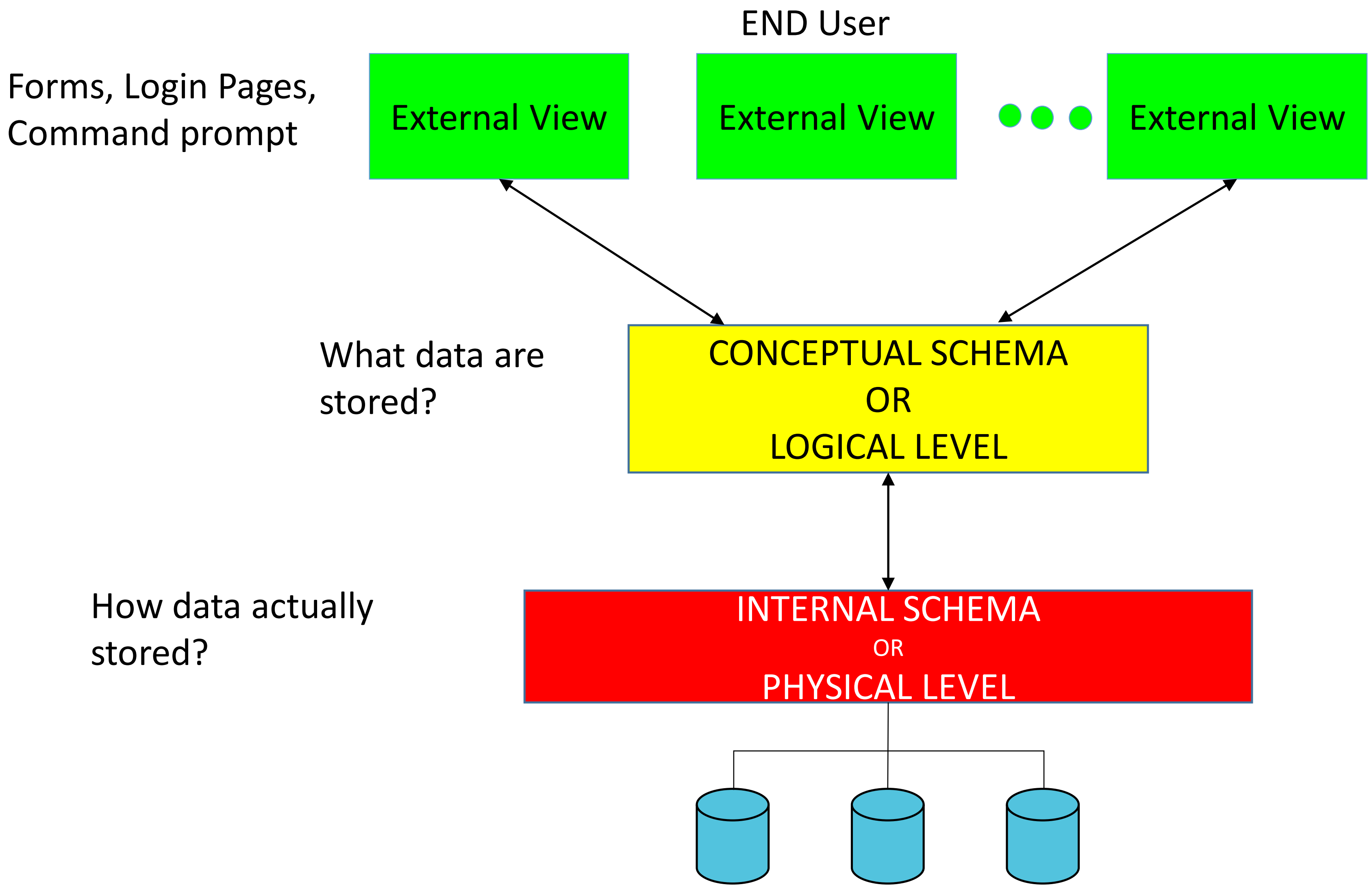
Schema and Instance

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - the overall design of the database
  - e.g., the database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **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
  - Analogous to the value of a variable

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.
- Developers hide the complexity from users through several levels of abstraction to simplify users’ interactions with the system.
- It follows three Schema Architecture:
  - **Physical Level**
  - **Logical Level**
  - **View level**

Three Schema Architecture


END User
Forms, Login Pages, Command prompt
External View
External View
External View
What data are stored?
CONCEPTUAL SCHEMA OR LOGICAL LEVEL
How data actually stored?
INTERNAL SCHEMA OR PHYSICAL LEVEL

- Physical level
  - Has an internal schema
  - Describes the physical storage structure of the database
  - Data structure used to store data
  - Access paths for database
  - The lowest level of abstraction describes **how** the data are actually stored.

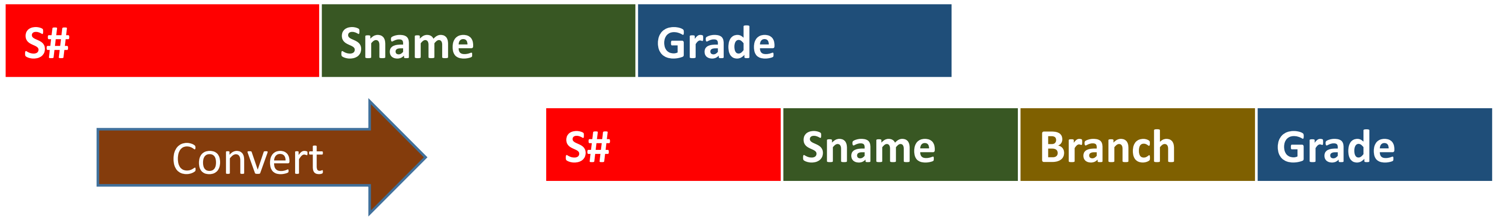
- Conceptual level
  - The next-higher level of abstraction describes **what** data are stored in the database, and what relationships exist among those data.
  - DBA uses this level
  - Hides details of physical storage structures
- External View or View level
  - The highest level of abstraction describes only part of the entire database.
  - Application programs
  - Describe part of the database that a particular user group is interested in
    - ✓ Hides rest of database from that user group
    - ✓ hide information (e.g., salary) for security purposes.

Data Independence

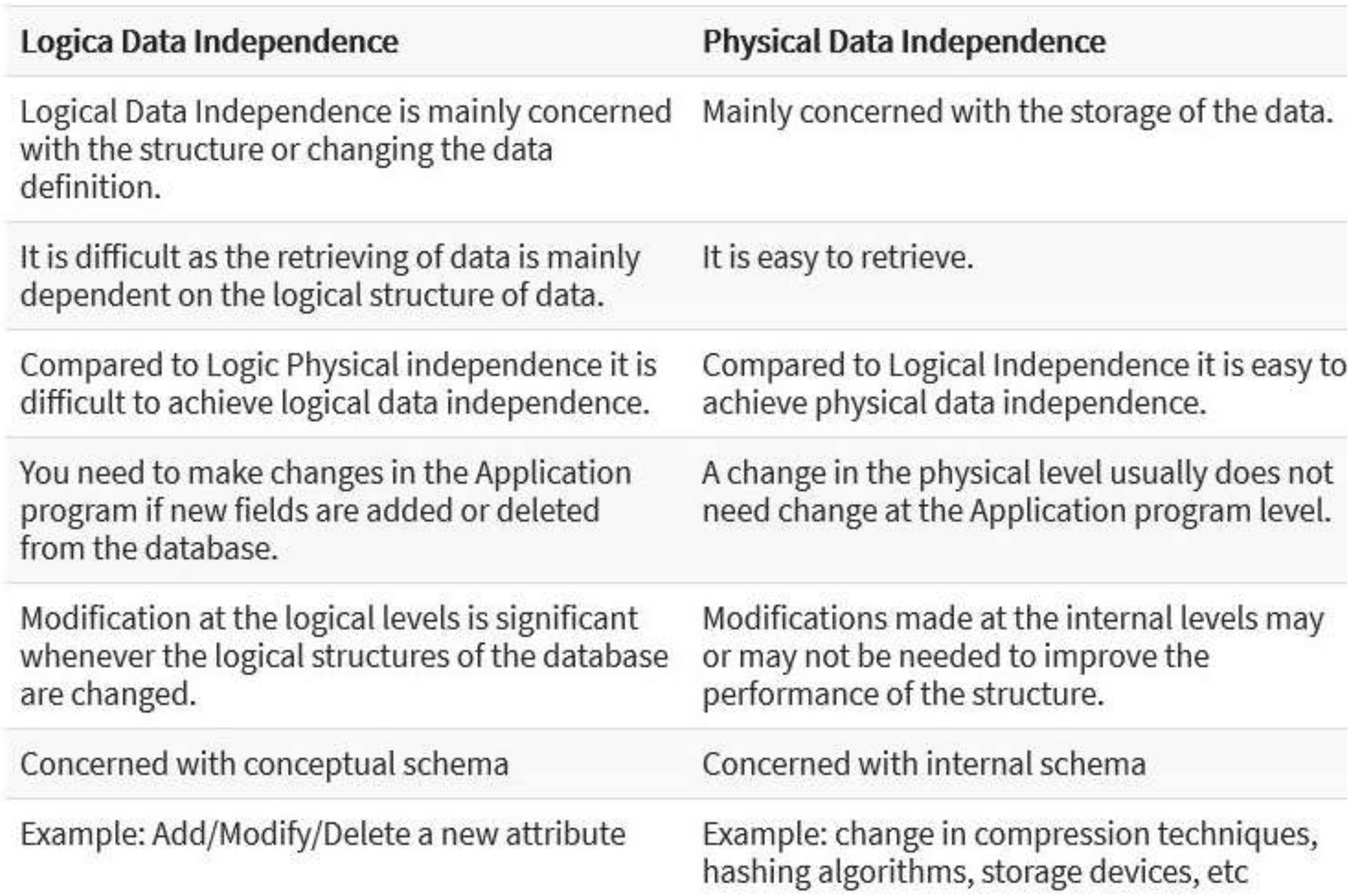
- Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Data independence helps you to keep data separated from all programs that make use of it.
- In DBMS there are two types of data independence
  - Physical data independence
  - Logical data independence.

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.
  - Numeric(7) to numeric(10)
- ***Logical Data Independence-*** the ability to modify the logical schema without having to change external schema or application programs.



Logical Data Independence vs Physical Data Independence



Logica Data Independence	Physical Data Independence
Logical Data Independence is mainly concerned with the structure or changing the data definition.	Mainly concerned with the storage of the data.
It is difficult as the retrieving of data is mainly dependent on the logical structure of data.	It is easy to retrieve.
Compared to Logic Physical independence it is difficult to achieve logical data independence.	Compared to Logical Independence it is easy to achieve physical data independence.
You need to make changes in the Application program if new fields are added or deleted from the database.	A change in the physical level usually does not need change at the Application program level.
Modification at the logical levels is significant whenever the logical structures of the database are changed.	Modifications made at the internal levels may or may not be needed to improve the performance of the structure.
Concerned with conceptual schema	Concerned with internal schema
Example: Add/Modify/Delete a new attribute	Example: change in compression techniques, hashing algorithms, storage devices, etc

Instances 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.
- Database systems have several schemas, partitioned according to the levels of abstraction.
- The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called **subschemas**, that describe different views of the database.

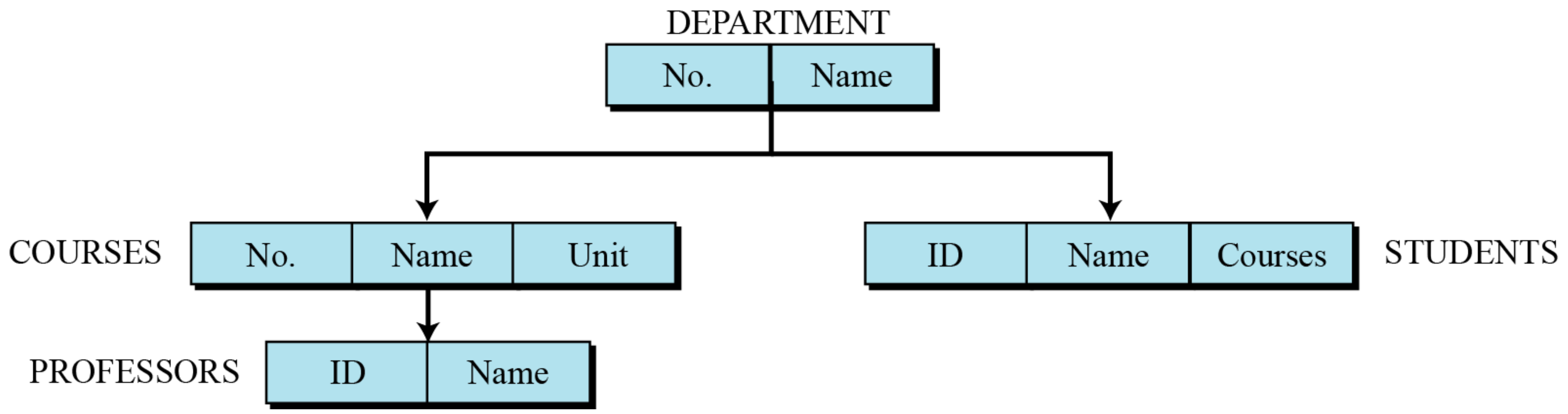
Data Models

Data Models

- A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraint.
- A data model provides a way to describe the design at physical, logical & view level.
- Various data models are:
  1. The entity- relationship model
  2. The object oriented model
  3. Relational model
  4. Network model and
  5. Hierarchical model

Hierarchical database model

In the hierarchical model, data is organized as an inverted tree. Each entity has only one parent but can have several children. At the top of the hierarchy, there is one entity, which is called the root.



An example of the hierarchical model representing a university

Network database model

In the network model, the entities are organized in a graph, in which some entities can be accessed through several paths (Figure).

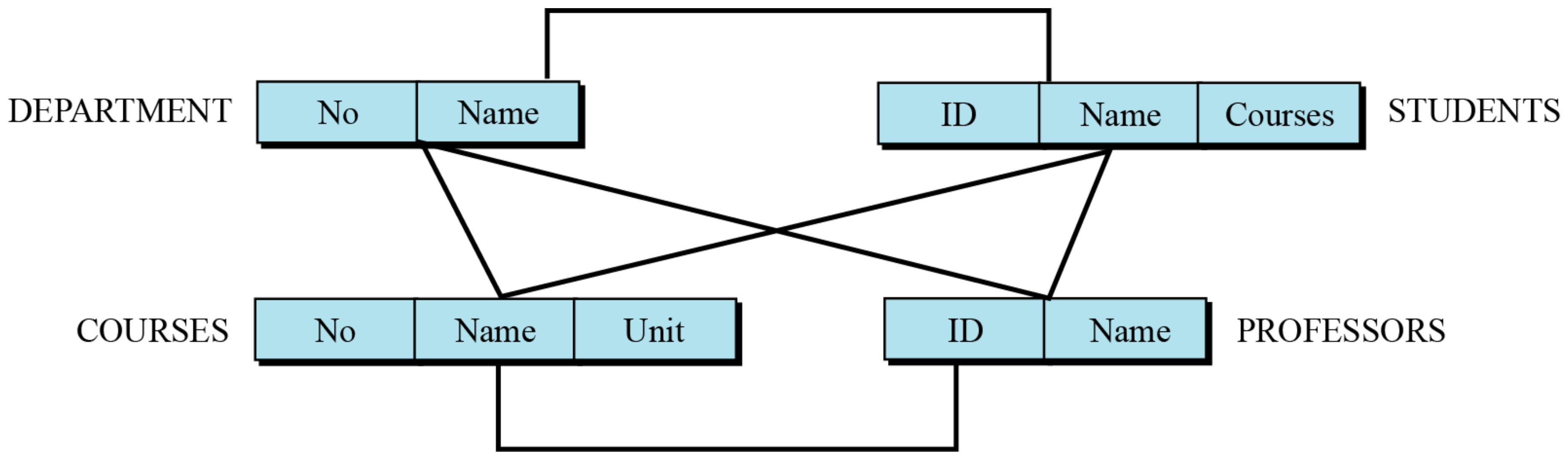


Figure An example of the network model representing a university

Relational database model

In the relational model, data is organized in two-dimensional tables called relations. The tables or relations are, however, related to each other.

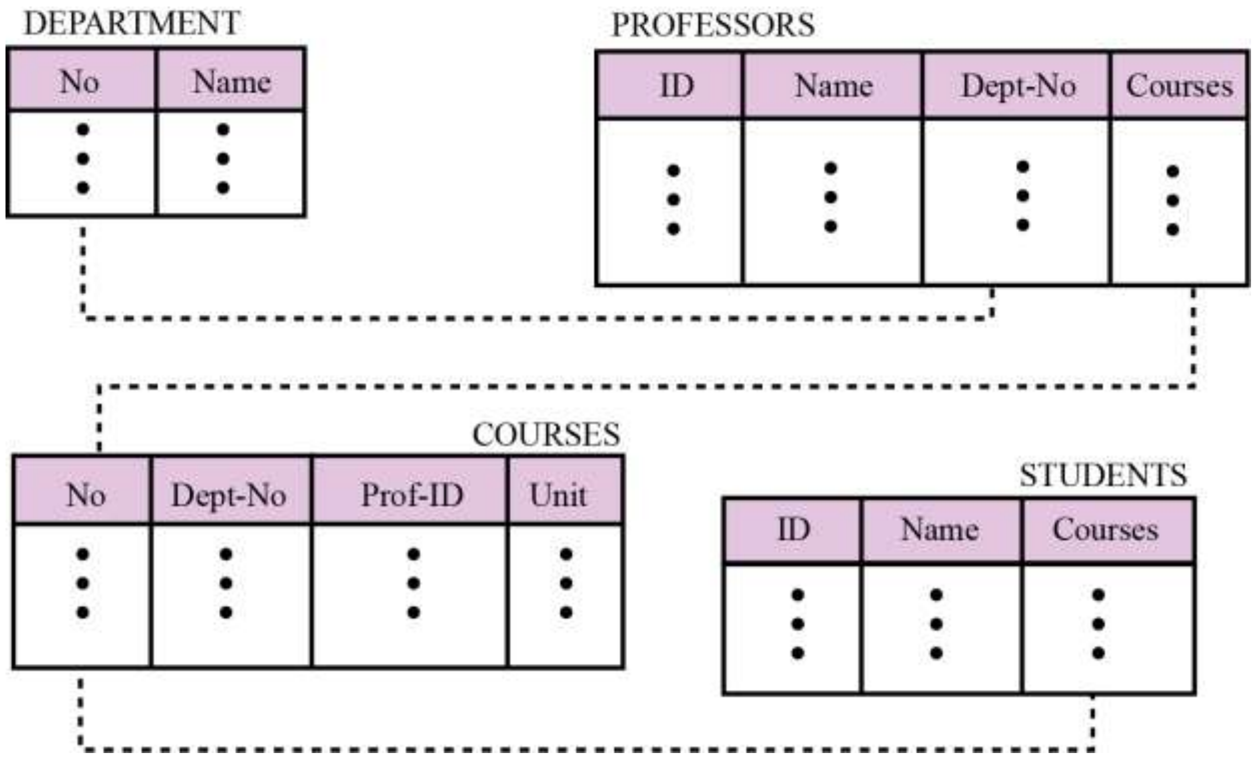
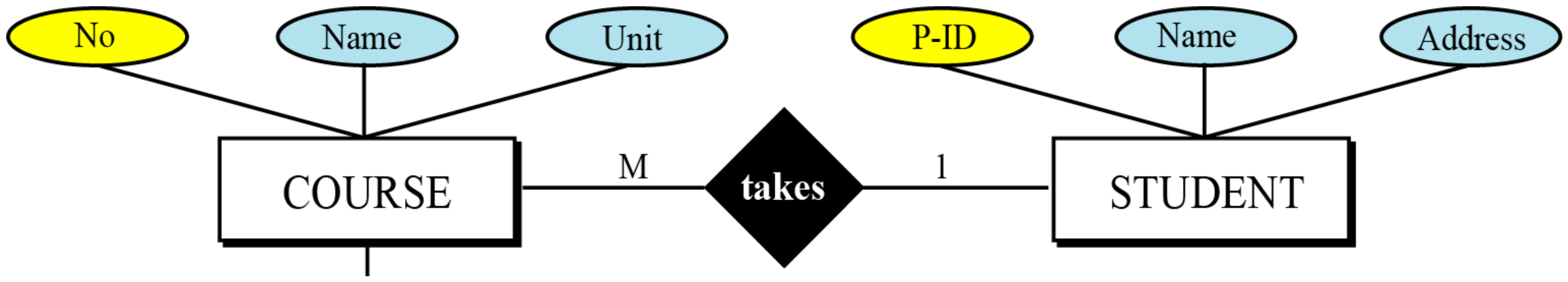


Figure 14.5 An example of the relational model representing a university

The Entity-Relationship Model

- Entity is real world object or thing that is distinguishable from others
- The entities for which information needs to be stored and the relationship among these entities. E-R diagrams uses several geometric shapes, but we use only a few of them here:



Object Oriented Model

- This is extended ER model
- Having notion of ***encapsulation,*** ***methods (functions)*** & ***object identity.***
- 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.

Database languages

Data Definition Language

- Data Definition Language **(DDL)** statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:
- CREATE – used to create objects in the database
- ALTER – used to alters the structure of the database
- DROP – used to delete objects from the database
- TRUNCATE – used to remove all records from a table, including all spaces allocated for the records are removed
- COMMENT – used to add comments to the data dictionary
- RENAME – used to rename an object

Data Manipulation Language

- A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects.
- Two types:
  - Procedural DMLs: require a user to specify what data are needed and how to get those data.
  - Declarative DMLs: (non-procedural DMLs) require a user to specify what data are needed without specifying how to get those data.eg SQL

- Here are the lists of tasks that come under DML:
- SELECT – It retrieves data from a database
- INSERT – It inserts data into a table
- UPDATE – It updates existing data within a table
- DELETE – It deletes all records from a table, the space for the records remain
- MERGE – UPSERT operation (insert or update)
- CALL – It calls a PL/SQL or Java subprogram
- EXPLAIN PLAN – It explains access path to data
- LOCK TABLE – It controls concurrency

Data Control Language

- There is another two forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Database. To perform any operation in the database, such as for creating tables, sequences or views we need privileges. Privileges are of two types,
- System – creating a session, table etc are all types of system privilege.
- Object – any command or query to work on tables comes under object privilege. DCL is used to define two commands. These are:
- Grant – It gives user access privileges to a database.
- Revoke – It takes back permissions from the user.

Transaction Control Language

- Transaction Control statements are used to run the changes made by DML statements. It allows statements to be grouped together into logical transactions.
- COMMIT – It saves the work done
- SAVEPOINT – It identifies a point in a transaction to which you can later roll back
- ROLLBACK – It restores database to original since the last COMMIT
- SET TRANSACTION – It changes the transaction options like isolation level and what rollback segment to use

Understanding DBMS Architecture

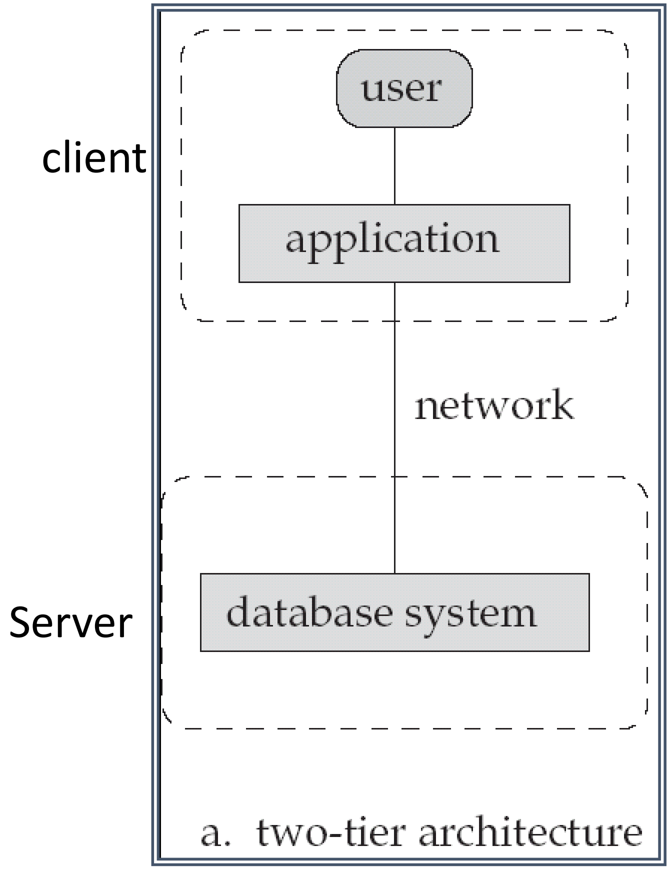
- A Database Management system is not always directly available for users and applications to access and store data in it. A Database Management system can be **centralized** (all the data stored at one location), **decentralized** (multiple copies of database at different locations) or **hierarchical**, depending upon its architecture.
- **1-tier DBMS** architecture also exist, this is when the database is directly available to the user for using it to store data. Generally such a setup is used for local application development, where programmers communicate directly with the database for quick response.

Database Architecture is logically of two types:

- **2-tier DBMS architecture**
- **3-tier DBMS architecture**

Two – tier Architecture

- Application is partitioned into two components:
  - Server: runs query language statements
  - Clients: provide user interface and local processing
- JDBC & ODBC are used for interaction between the client and the server.



2-tier DBMS Architecture

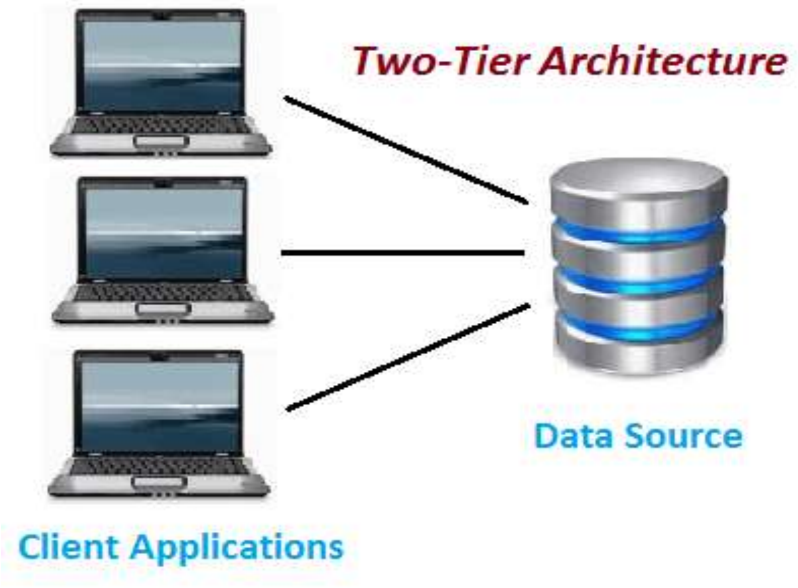
2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

An application interface known as **ODBC**(Open Database Connectivity) provides an API that allow client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.

Cont..

Two-Tier Architecture

- The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.

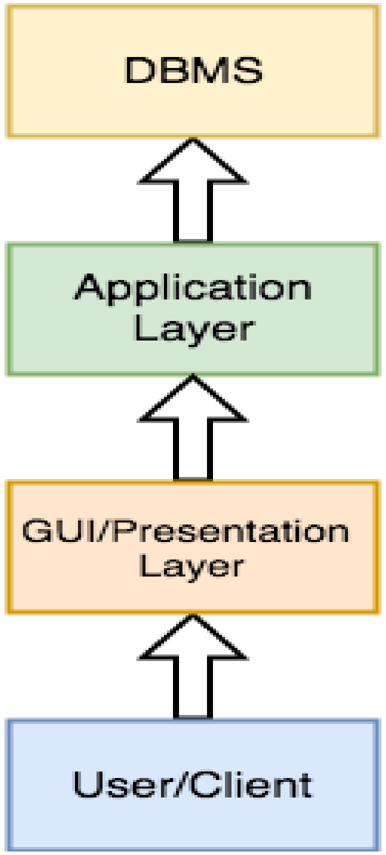


- The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.
- Let's take a look of real life example of Railway Reservation two-tier architecture:
- Let's consider that first Person is making Railway Reservation for Mumbai to Delhi by Mumbai Express at Counter No. 1 and at same time second Person is also try to make Railway reservation of Mumbai to Delhi from Counter No. 2
- If staff from Counter No. 1 is searching for availability into system & at the same staff from Counter No. 2 is also looking for availability of ticket for same day then in this case there is might be good change of confusion and chaos occurs. There might be chance of lock the Railway reservation that reserves the first.
- But reservations can be making anywhere from the India, then how it is handled?
- So here if there is difference of micro seconds for making reservation by staff from Counter No. 1 & 2 then second request is added into queue. So in this case the Staff is entering data to Client Application and reservation request is sent to the database. The database sends back the information/data to the client.
- In this application the Staff user is an end user who is using Railway reservation application software. He gives inputs to the application software and it sends requests to Server. So here both Database and Server are incorporated with each other, so this technology is called as "***Client-Server Technology***".
- The Two-tier architecture is divided into two parts:
- **1) Client Application (Client Tier)**
  2) Database (Data Tier)
- On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.

- **Advantages:**
- Easy to maintain and modification is bit easy
- Communication is faster
- **Disadvantages**:
- In two tier architecture application performance will be degrade upon increasing the users.
- Cost-ineffective

3-tier DBMS Architecture

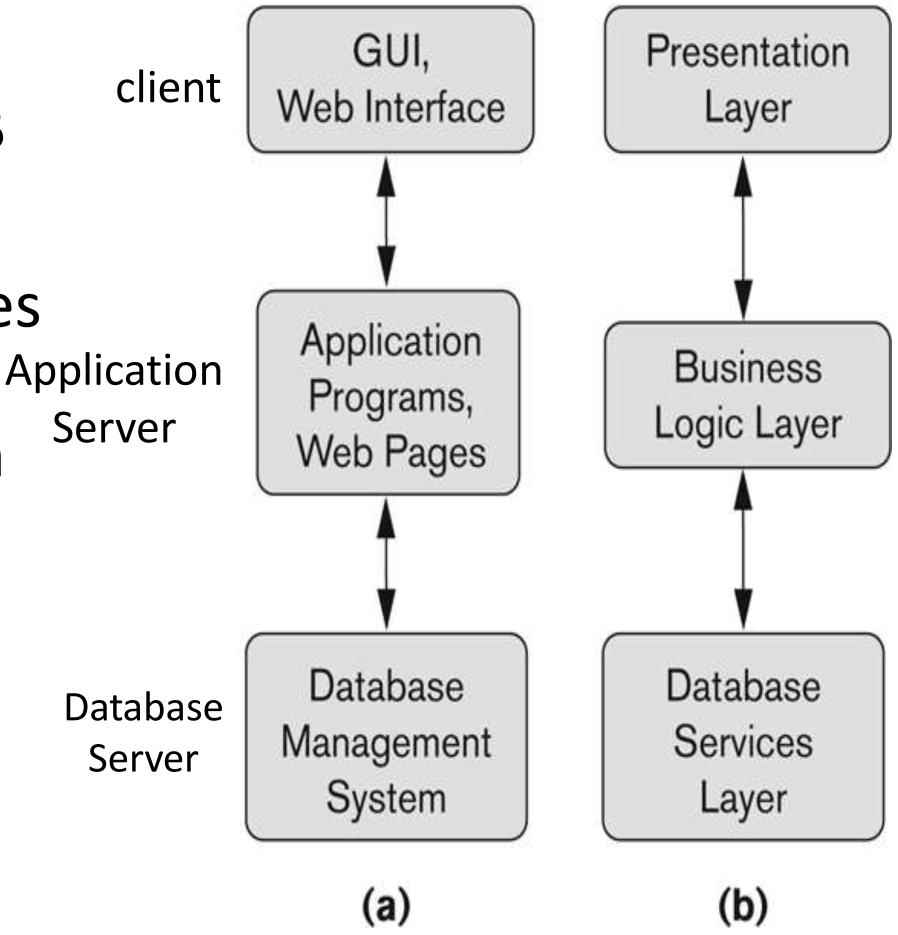
3-tier DBMS architecture is the most commonly used architecture for web applications.



Cont..

Three tier Architecture

- *Client*: merely a user interface (form interface) but no direct DB calls,
- *Application Server*: communicates with database system to access data. Had coding for what action to carried out under what conditions.
- *Database Server*: handle data access for query receive by Application Server



It is an extension of the 2-tier architecture. In the 2-tier architecture, we have an application layer which can be accessed programmatically to perform various operations on the DBMS. The application generally understands the Database Access Language and processes end users requests to the DBMS.

In 3-tier architecture, an additional Presentation or GUI Layer is added, which provides a graphical user interface for the End user to interact with the DBMS.

For the end user, the GUI layer is the Database System, and the end user has no idea about the application layer and the DBMS system.

If you have used **MySQL**, then you must have seen **PHPMyAdmin**, it is the best example of 3-tier DBMS architecture.

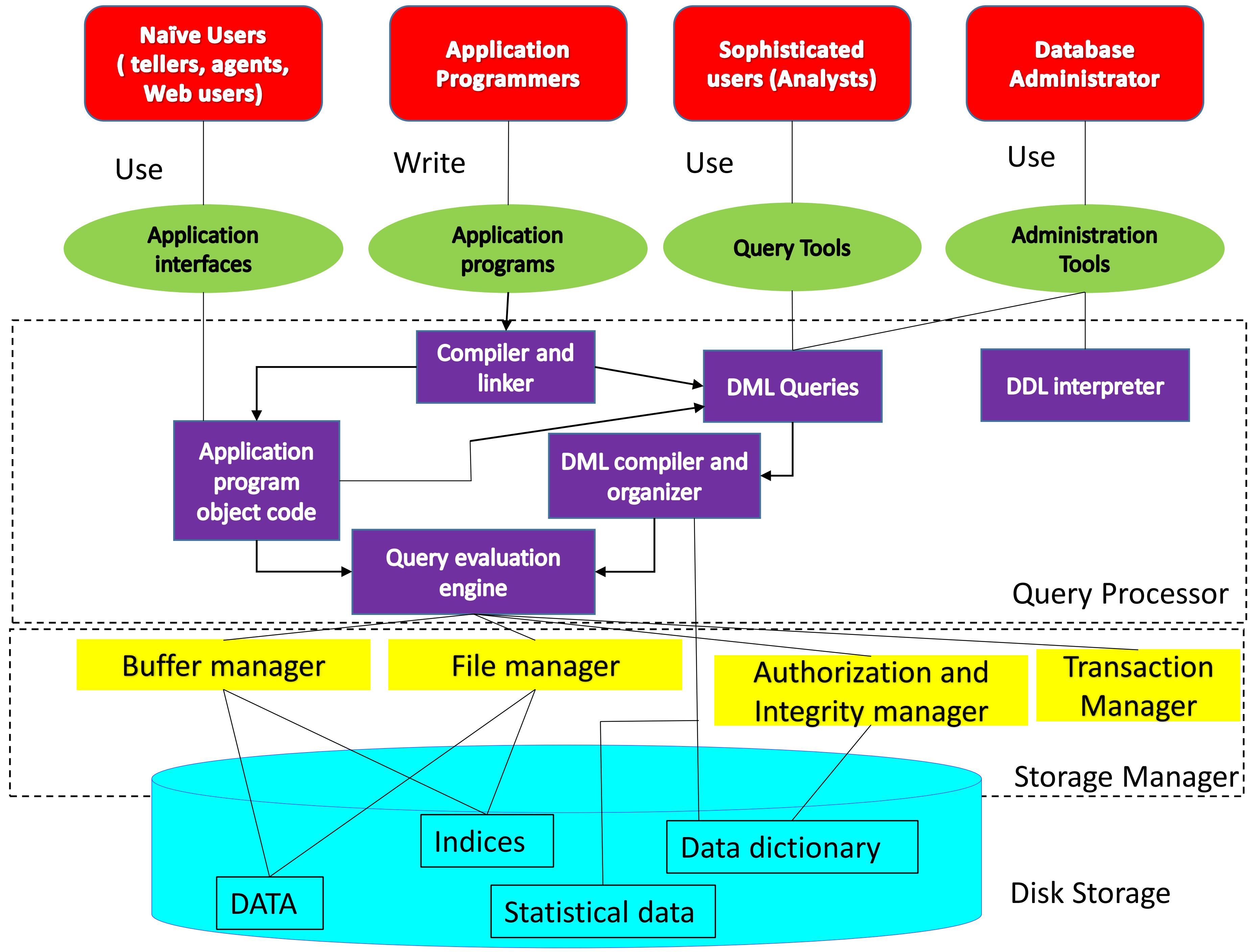
Three-Tier Architecture

- **Three-tier architecture** typically comprise a presentation tier, a business or data access tier, and a data tier. Three layers in the three tier architecture are as follows:
- **1) Client layer**
  2) Business layer
  3) Data layer
- **1) Client layer:**
- It is also called as *Presentation layer* which contains UI part of our application. This layer is used for the design purpose where data is presented to the user or input is taken from the user. For example designing registration form which contains text box, label, button etc.
- **2) Business layer:**
- In this layer all business logic written like validation of data, calculations, data insertion etc. This acts as a interface between Client layer and Data Access Layer. This layer is also called the intermediary layer helps to make communication faster between client and data layer.
- **3) Data layer:**
- In this layer actual database is comes in the picture. Data Access Layer contains methods to connect with database and to perform insert, update, delete, get data from database based on our input data.

- **Advantages**
- High performance, lightweight persistent objects
- Scalability – Each tier can scale horizontally
- Performance – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
- High degree of flexibility in deployment platform and configuration
- Better Re-use
- Improve Data Integrity
- Improved Security – Client is not direct access to database.
- Easy to maintain and modification is bit easy, won't affect other modules
- In three tier architecture application performance is good.
-
- **Disadvantages**
- Increase Complexity/Effort

Database Architecture

Overall database mgmt. structure



Naïve Users (tellers, agents, Web users)
Application Programmers
Sophisticated users (Analysts)
Database Administrator
Use
Write
Use
Use
Application interfaces
Application programs
Query Tools
Administration Tools
Compiler and linker
DML Queries
DDL interpreter
Application program object code
DML compiler and organizer
Query evaluation engine
Query Processor
Buffer manager
File manager
Authorization and Integrity manager
Transaction Manager
Storage Manager
Indices
Data dictionary
DATA
Statistical data
Disk Storage

Database Users

Users are differentiated by the way they expect to interact with the system

- *Application programmers* – interact with system through DML calls, writes application programs
- *Sophisticated users* – form requests in a database query language eg analysts
- *Specialized users* – write specialized database applications that do not fit into the traditional data processing framework
- *Naïve users* – invoke one of the permanent application programs that have been written previously
  - E.g. people accessing database over the web, bank tellers, clerical staff

Database Administrator (DBA)

- Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.
- Database administrator's duties include:
  - Schema definition- by executing statements in DDL
  - Storage structure and access method definition
  - Schema and physical organization modification
  - Granting user authority to access the database
  - Specifying integrity constraints eg. Primary key
  - Back up data
  - Ensure enough disk space for smooth performing of DB
  - Monitors jobs running on DB

Storage Manager

- Provide interface between low level data in database and application program.
- Components:
  - *Authorization & integrity Manager*: define role and responsibility for users, and provide Integrity checks.
  - *Transaction manager*: ensure DB consistency in concurrent access
  - *File Manager*: manage allocation of disk space and Data structure used for storing data
  - *Buffer Manager*: decide what data to *cache*

Disk Storage

- *Data Files*: which stores database itself
- *Data Dictionary*: stores metadata about the structure of the database
- *Indices:* provide fast access to data items

Query Processor

- *DDL interpreter*: which interprets DDL statements and records the definitions in the data dictionary
- *DML Compiler*: translate DML statements to evaluation plan. *Also perform query optimization. (pick the lowest cost evaluation plan from among the alternatives)*
- *Query Evaluation Engine*: executes low level instructions generated by DML compiler.