



Tribhuvan University
B.Sc. CSIT

Butwal Multiple Campus

Database Management System

CSC260

Unit 1: Database and Database Users

Er. Nirmal Thapa

Email: nirmalthapa562@gmail.com

Butwal Multiple Campus

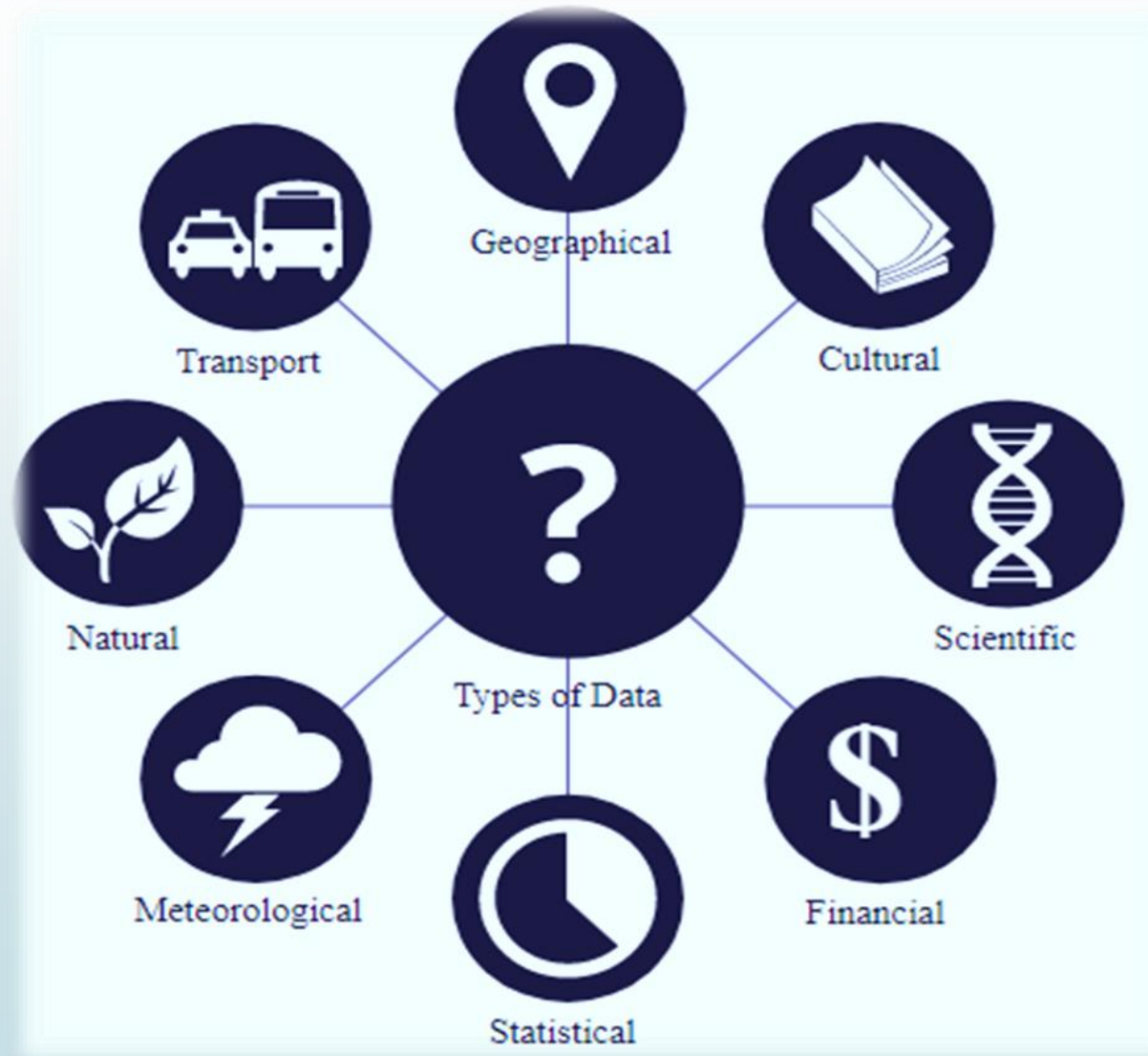
Tribhuvan University

Introduction

Data

- Data is a raw, unanalyzed, unorganized, unrelated, and an uninterrupted entity.
- It is derived from the Latin word “Datum” which means ‘something given’.
- There are multiple forms of data such as numbers, letters, set of characters, images, graphics.
- Measuring units for data are Bits, Nibble, Byte, KB, MB, GB, TB, PT, EB, ZB, YT.
- Data deteriorates as time passes.

Introduction



Introduction

Information

- **Processed data** is information which is accurate, systematize, understandable, relevant, and timely.
- It is derived from the French verb “**informare**” which means ‘to inform’.
- **Information = data + meaning**
- Information is **critical** in sense
- It comes in the second level of the intelligence hierarchy.



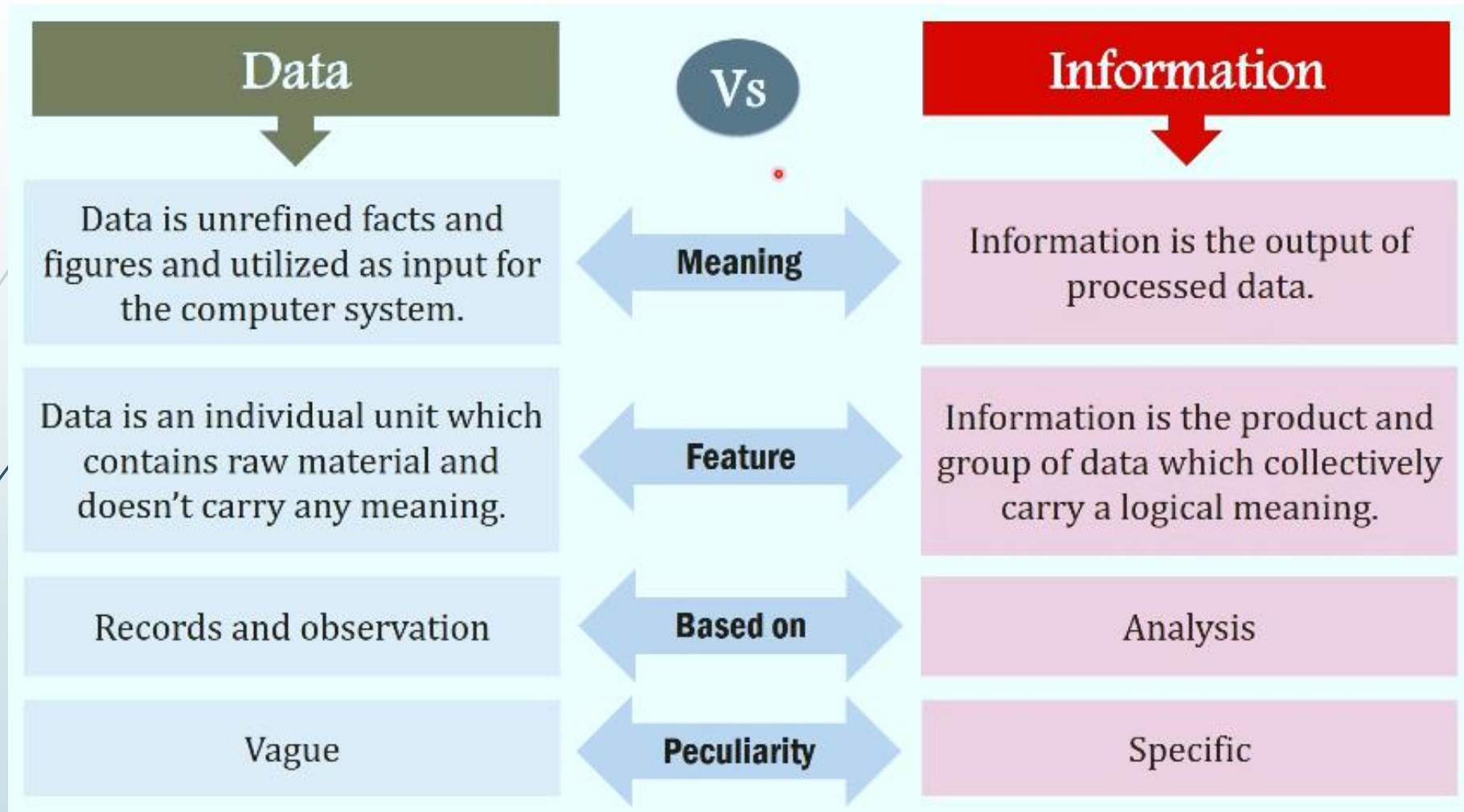
Introduction

Information

Quality of the information depends upon the below given factors:

- Accuracy of the information
- How easily one can understand a piece specific information.
- The relevance for a particular use.
- How up-to-date the information is.

Introduction



Introduction

Database

- A database is a collection of data items stored in one place and having some common base (Background) between them.
- For Example: A college database contains data such as teachers, students, books, canteen etc. college is common (Base) between all above data items.
- So, Data with a common base (Background) is called as Database.
- The database acts as a logical collection of relevant data. It is designed to offer an organized mechanism for storing, managing and retrieving stored information.

Introduction

Concept of DBMS

- A DBMS is a collection of software or programs which help user in creation and maintenance of database (set of information).
- Hence it is also known as a computerized record-keeping system.
- DBMS is the software system that helps in the process of defining, constructing manipulating the database.
- Database management system has become an integral part of the information systems of many organizations as it is used to handle a huge amount of data.
- Computer-based Information Systems (IS) is capable of serving many complex tasks in a coordinated manner, handle large volumes of data, multiple users

Introduction

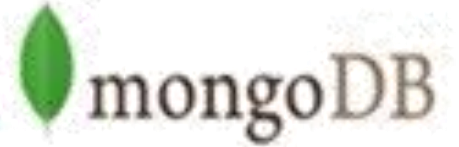
Concept of DBMS

- The heart of an IS is DBMS. This is because most IS have to handle huge amounts of data.
- This core module of an IS is also called as DBMS.
- DBMS provides an interface or a tool to perform various operations:
 1. Defining the database
 2. Inserting data records into the database
 3. Deleting data records
 4. Retrieving data from database
 5. Providing security to the database
- Used almost everywhere and most of the people interact with database either directly or indirectly

Introduction

Commonly used DBMS

ORACLE[®]
DATABASE





Introduction

Applications of DBMS:

1. Banking

2. Airlines

3. Universities

4. Credit card
transactions

5. Telecommunication

6. Finance

7. Sales

8. Manufacturing

9. Human Resources

Introduction

Career Opportunity of Learning DBMS

DBMS is a major skill for software engineers

DBMS is a sought-after skill.

Here are the few most demanding opportunities that you can explore with this skill:

Data Architect: Designs and builds data frameworks based on the product requirements.

Data Engineer: Develops database solutions as designed by the data architect.

Database Manager: Maintains the database, normalizing it and preparing it for expansion.

Data Analyst: Evaluates company's data and taking decisions regarding improving market position.

Data Scientist: Designs and constructs new processes that would improve data mining and data production.

Introduction

Objectives/Needs/Advantages of DBMS

1. Controlling Redundancy:

- In traditional file system, each user group maintain its own files
- This leads to wastage of storage space -> inconsistency
- All the data stored in only one place of the db
- Ensures consistency and saves storage space

2. Restricting Unauthorized Access:

- Multiple user shares a large db -> type of access operation must be controlled
- DBMS must provide security and authorization subsystem
- DBA -> creates accounts and specifies account restrictions

Introduction

Objectives/Needs/Advantages of DBMS

3. Providing persistent storage for program objects:

- In traditional file system, Once the program terminates -> value of program variable -> discarded
- In DBMS -> Non discarded -> Stores objects permanently (persistent object)
- Needed explicit conversion of data structure of programming language in traditional s/m
- DBMS recognizes data structure of programming language (automatically conversion)

Introduction

Objectives/Needs/Advantages of DBMS

4. Providing storage structure for efficient query processing:

- Provide capabilities for efficiently execution queries and updates
- Query processing and optimization module -> responsible

5. Providing Backup and Recovery:

- The backup and recovery subsystem of dbms is responsible recovery -> in case of h/w and s/w failure

Introduction

Objectives/Needs/Advantages of DBMS

6. Providing Multiple User Interface:

- Multiple users -> different level of technical knowledge -> so it provide variety of UI
- Eg: Query language : casual users, Programming language Interface : Application programmers

7. Represent Complex Relationship among data:

- DB may have variety of data -> interrelated in many ways
- DBMS-> capable of representing complex relationship among data
- Retrieve and update related data easily and efficiently

Introduction

Objectives/Needs/Advantages of DBMS

8. Enforcing Integrity Constraints:

- Simplest type of Integrity constraint -> specifying datatypes for each data item
- Another type of integrity constraint -> uniqueness of data item values
- Identify integrity constraints during DB design -> DB designers responsibility

9. Flexibility:

- Allow certain types of changes to the structure of DB without affecting the stored data

Introduction

Objectives/Needs/Advantages of DBMS

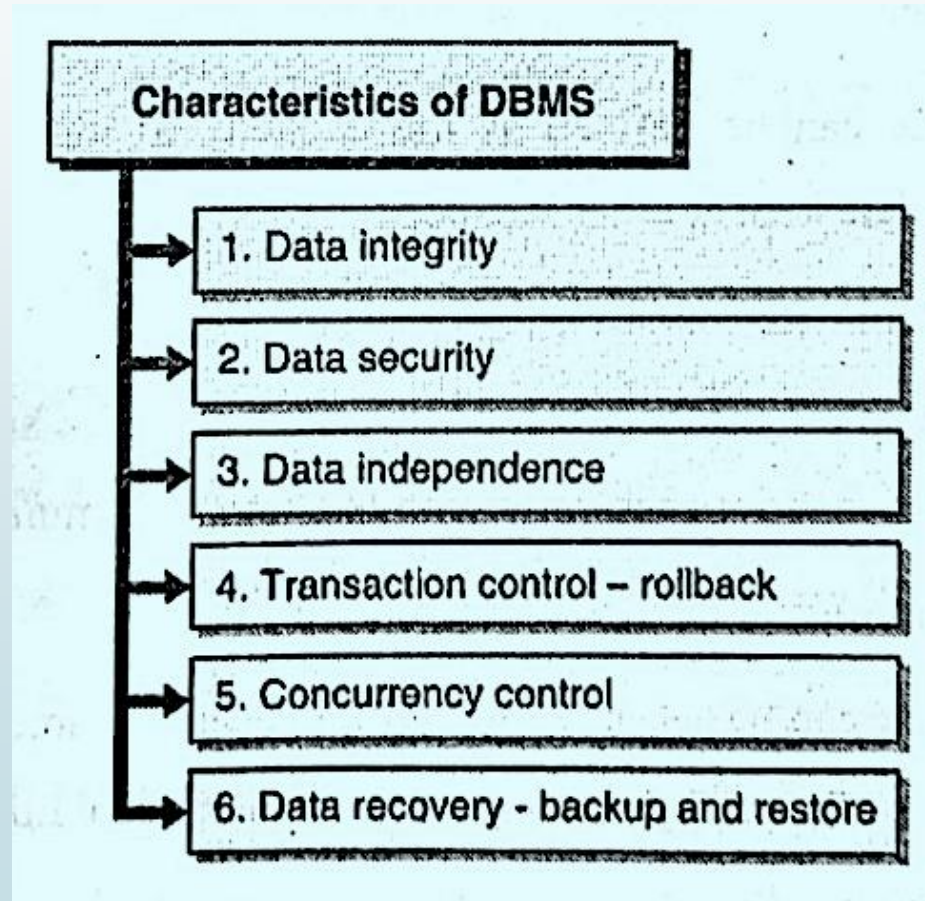
10. Availability of upto-date Information:

- One User's update -> all users can immediately see this update
- Essentials for many Transaction processing application
- Made possible by concurrency control module

Introduction

Characteristics of DBMS

DBMS has many important characteristics due to which database has become an integral part of the software industry:





Introduction

Characteristics of DBMS

1. Data Integrity:

- Integrity constraints provide a way of ensuring that changes made in the database by authorized users that do not result in the loss of data consistency and correctness.
- Database integrity is concerned with the correctness and completeness of data in the database.
- This objective can never be guaranteed, one cannot ensure that every entry made in database is accurate.

Introduction

Characteristics of DBMS

2. Data Security:

- A DBMS system always has a separate system for security which is responsible for protecting database against accidental or unintentional loss, destruction or misuse.
- Data access in database should be given to only authorized users.
- Only authorized users should be allowed to modify data.
- Authorized users are able to access data any time he/she wants.

3. Data Independence:

- Capacity to change data kept at one place without changing data kept at another location



Introduction

Characteristics of DBMS

4. Transaction control- rollback:

- The changes made in the database can be reverted back with help of rollback command
- The changes can be saved successfully with the help of commit data command.

5. Concurrency control:

- The data in database can be accessed by multiple users at same point of time.
- Such operations are allowed by sharing same data between multiple users.

6. Data recovery - Backup and Restore:

- Database recovery is the process of restoring the database to original (correct) state after database failure.

Introduction

When Not to Use DBMS /Disadvantages of DBMS

1. Overhead Cost of using DBMS:

- High Initial Investment
- Overhead for providing security, concurrency control, recovery services

2. DB Applications -> Simple, well defined and no DB changes expected

3. Multiple user access -> not required



Introduction

File Processing system vs Database System

1. Redundancy can be reduced:

- As we are using relational approach for data organization, data is not stored in more than one location.
- Repetition of information can be avoided which in turn saves storage space.

2. Data can be shared

- Multiple users can login at a time into the database to access information.
- They can manipulate the database in a controlled environment

3. Inconsistency can be avoided

- The database assures that all the users access actual or true data present in the database.

Introduction

File Processing system vs Database System

4. Centralized control of data
5. Standards can be enforced
6. Security restrictions can be applied
7. Integrity can be maintained
8. Data independence can be provided

Introduction

File Processing system vs Database System

Database Management System	File Processing System
Computerized record - keeping system is used in DBMS	Collection of individual files accessed by applications programs is called File Processing System
DBMS allows flexible access to data	File processing system is designed to allow predetermined access to data
It co-ordinates both the physical and logical access	It co-ordinates only the physical access to data
Provides multiple user interface	Data is isolated in the file system
Unauthorized access is restricted	Unauthorized access cannot be restricted
Redundancy can be controlled	Redundancy cannot be controlled

Introduction

Evolution of DBMS

1. Database has completed more than 50 years of journey of its Evolution from file system to relational and objects relational systems
2. It has gone through several generations
 - **File-Based**
 - **Hierarchical Data Model (1968)**
 - Files are related in a parent/child manner
 - Lack structural independence
 - Can't easily handle a many-many relationship

Introduction

Evolution of DBMS

- **Network Model (Standardized 1971)**
 - Files are related as owner and members, like to the common network model
- **Relational Database (1970-present)**
 - It is the era of RDBMS
 - Proposed by E.F Codd
 - Has two main terminologies: instance and schema
 - Use the concept like set theory and predicate logic



Introduction

Evolution of DBMS

- **Cloud Database**

- Facilitates us to store, manage and retrieve their structured, unstructured data via cloud platform
- Data is accessible over the Internet
- Also called DBaaS because they are offered as a managed service
- Eg. AWS, Google Cloud Spanner, MS SQL Server, Oracle DB clouds Services

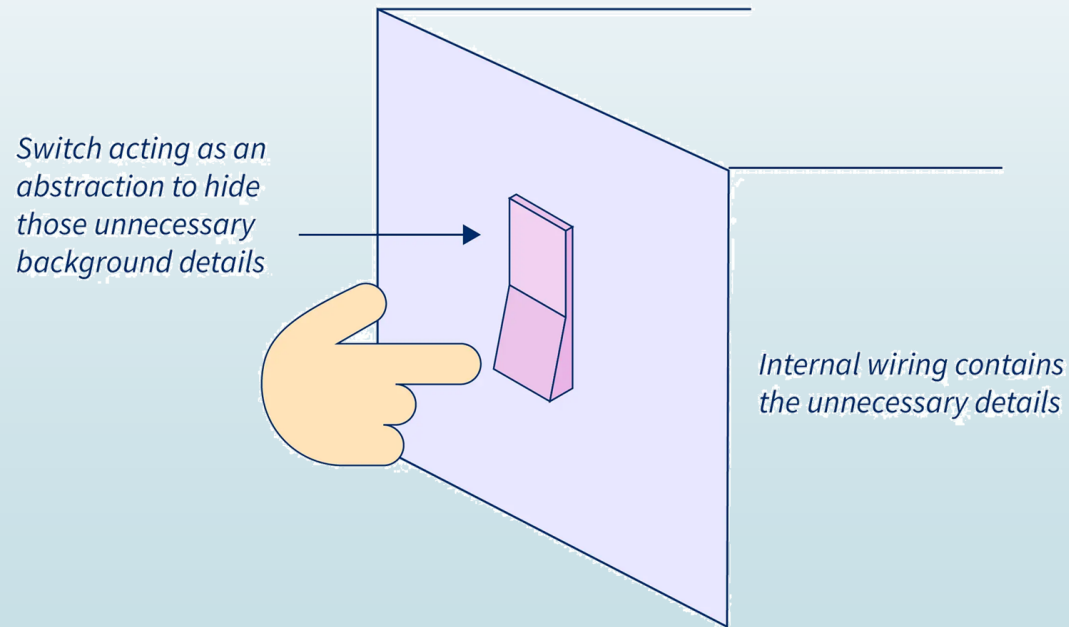
- **NoSQL Database**

- Can accommodate a wide variety of data models
- Alternative to traditional RDB in which data is placed in tables and data is perfectly designed before the database is built
- Useful for large set of distributed data
- Eg. MongoDB, CouchDb (document based), Redis (key value store)

Introduction

Data Abstraction

- Data abstraction is the method of hiding the unimportant details that are present in the database from the end users to make the accessing of data easy and secure.
- **Example**





Introduction

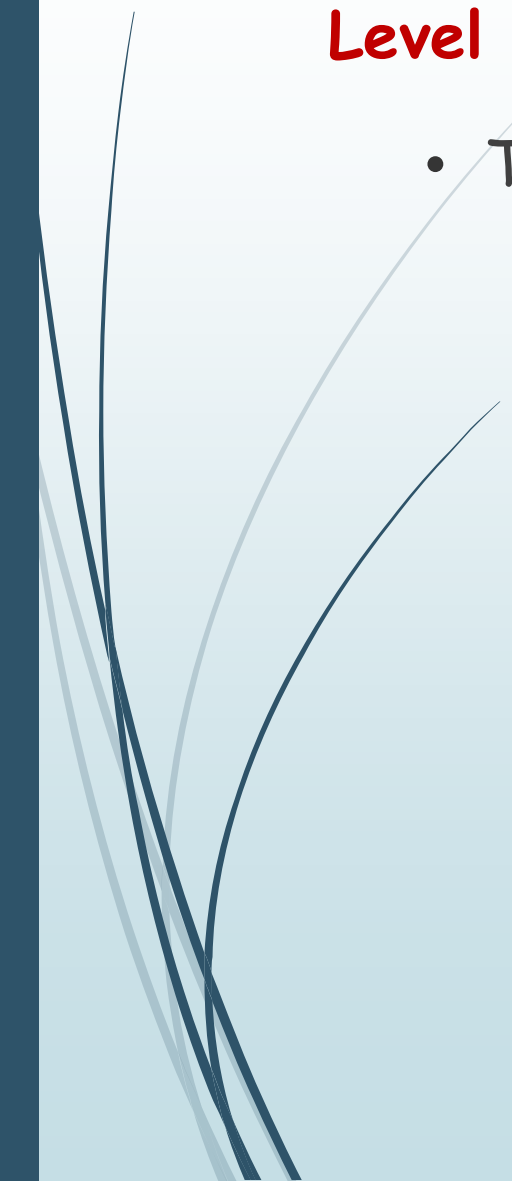
Data Abstraction

- Data abstraction is the method of hiding the unimportant details that are present in the database from the end users to make the accessing of data easy and secure.
- All the DBs have complex data structures which are, in fact, of no use to an end user
- The developers keep away the complex data from the user and remove the complications
- The main purpose of data abstraction is to hide internal irrelevant details and provide an abstract view of the data.
- This is what Data Abstraction is.
- In DBMS, data abstraction is performed in layers which means there are levels of data abstraction

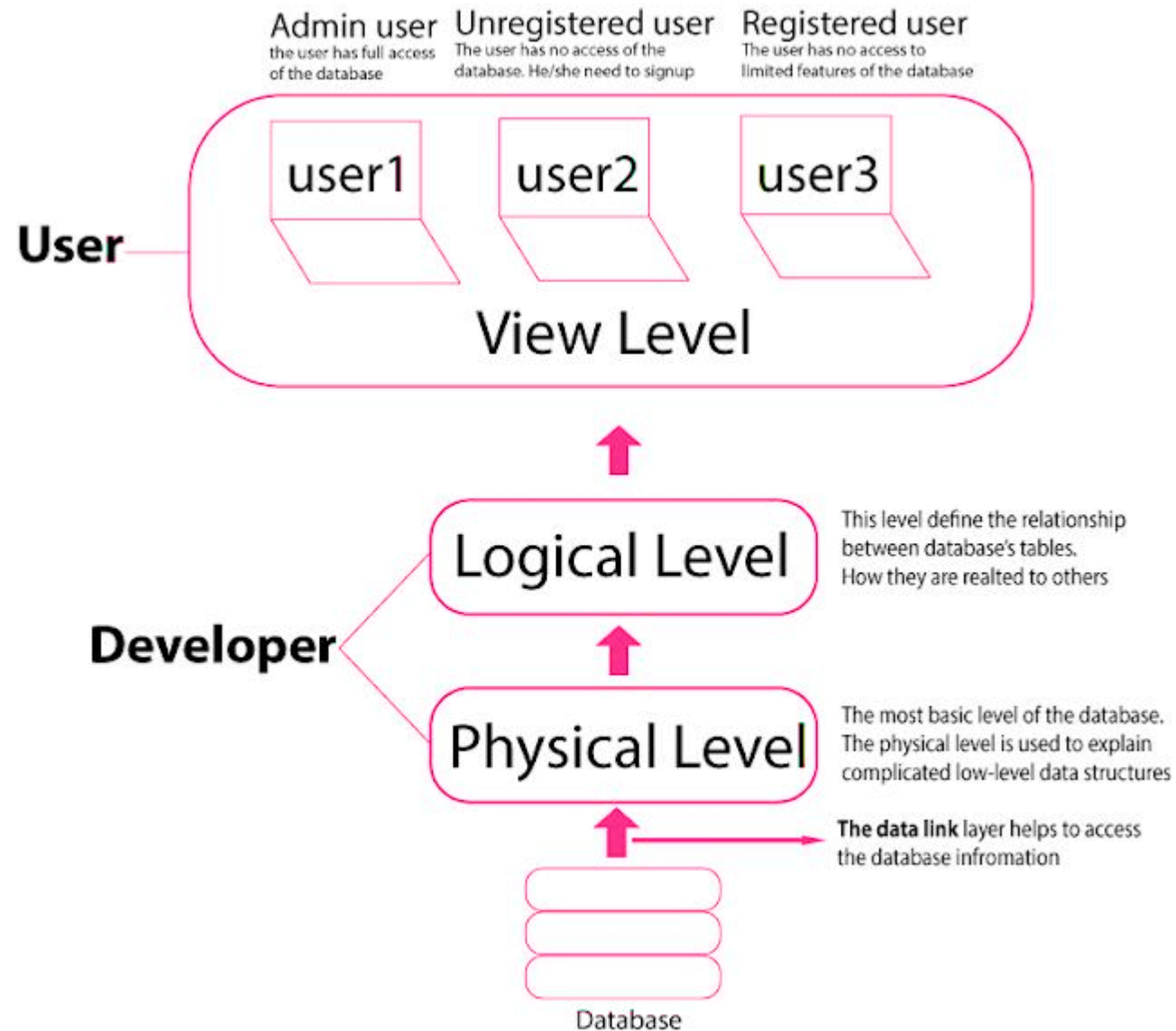


Introduction

Level of Data Abstraction

- The data abstraction in DBMS is implemented in 3 layers:
 - Physical or Internal Level
 - Logical or Conceptual Level
 - View or External Level
- 

Level of Data Abstraction





Introduction

Physical Level or Internal Level

- This is the layer of data abstraction where the raw data is physically stored as files
- This layer contains all the complex data structures and the data accessing methods defined.
- The physical layer is the lowest level of data abstraction in a DBMS.
- It is the DBA who decides how the data is to be stored in these physical hard drives.

Introduction

Physical Level or Internal Level

Example:

- When accessing data we may get a single data. we visualize a table of rows and columns.
- But at a physical level, these tables are stored in hard drives which are located at a very secure data center.





Introduction

Logical or Conceptual Level

- The structure of the data is defined at the logical or conceptual level.
- It explains what data is going to be stored in the database and what the relationship is between them.
- It describes the structure of the entire data in the form of tables.
- It is less complex than the physical level.
- With the help of the logical level, DBA abstract data from raw data present at the physical level.
- This layer does not have any information about how the end user will view the data.



Introduction

Logical or Conceptual Level

Example

- We have data of a few products like **product id**, **product name**, and **manufacturing date**, and we have another set of data of customers containing **customer id**, **customer name**, and **customer address**.
- Now, we need to frame this data in proper tables of **products** and **customers**.
- After that, we can even frame a join to show which product has been ordered by which customer.

Introduction

View Level or External Level

- This is what an end-user gets to see.
- He/she does not get the entire database, but depending on the queries made from the front-end the user gets to see the data.
- It may be a single data from the entire database or a collection of data in tabular format.
- Multiple views of the same data are available to the user, the representation can be:
 - A table, a graph, or a pie chart.
- View Level is the highest level of data abstraction in DBMS.

Introduction

View Level or External Level

Example:

- let us say a customer wants to view the order history, he gets to see only the orders he had made in the past.
- Now, let us say a shop owner needs to see the products that are on the order list.
- He gets to see a table containing all the info about the products and the customers to whom they need to be delivered

Introduction

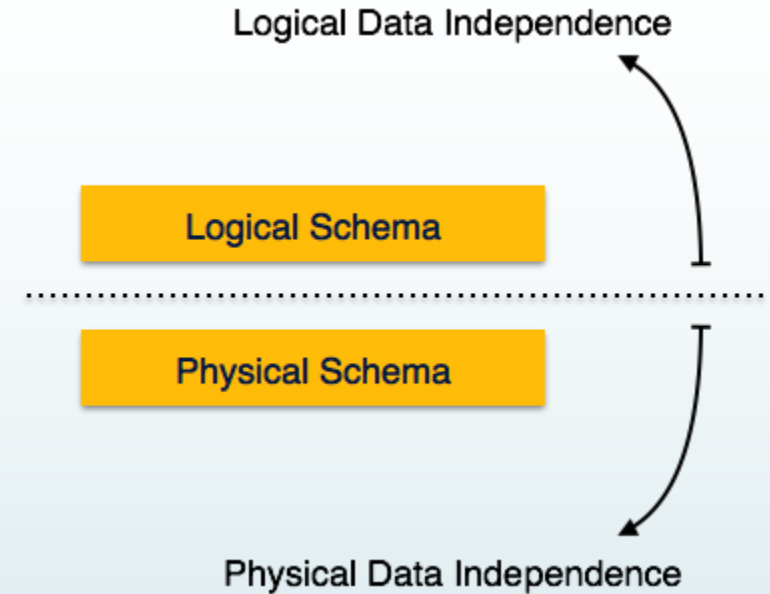
Data Independence

- A database system normally contains a lot of data in addition to users' data.
- It stores data about data, known as metadata, to locate and retrieve data easily.
- Difficult to modify or update metadata once it is stored in the database
- But as a DBMS expands, it needs to change over time to satisfy the requirements of the users.
- If the entire data is dependent, it would become a tedious and highly complex job.
- Data independence can be explained using the three-schema architecture.
- Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

Introduction

Data Independence

- There are two types of data independence:
 - Logical data independence
 - Physical data independence



- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level.
- This data is independent but mapped to each other.

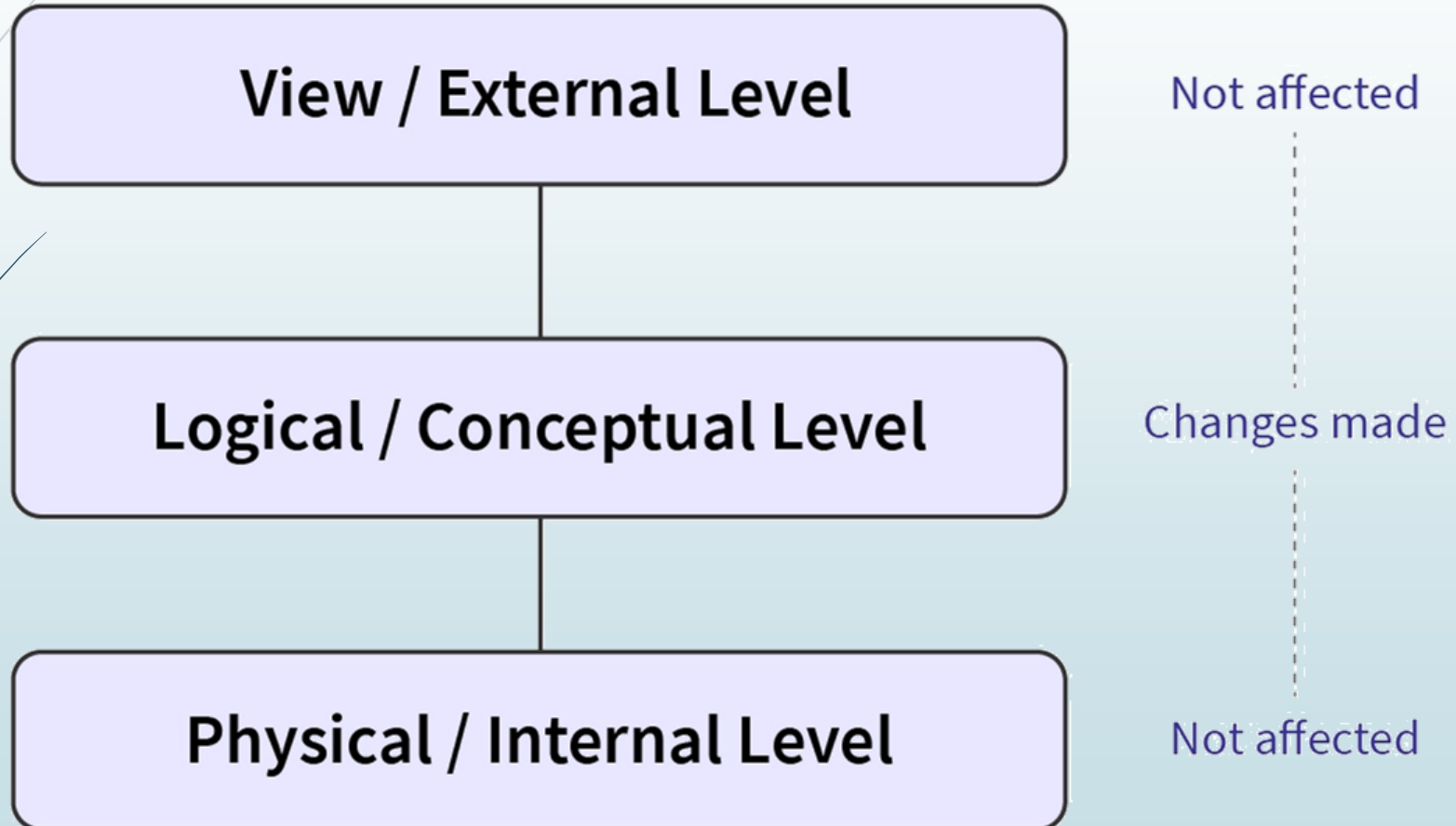
Introduction

Logical Data Independence

- Is the ability to modify logical schema without causing any unwanted modifications to the external schema (user's view)
- It is data about database, that is, it stores information about how data is managed inside.
- **For example**, a table (relation) stored in the database and all its constraints, applied on that relation.
- If we do some changes on table format, it should not change the data residing on the disk.

Introduction

Logical Data Independence



Introduction

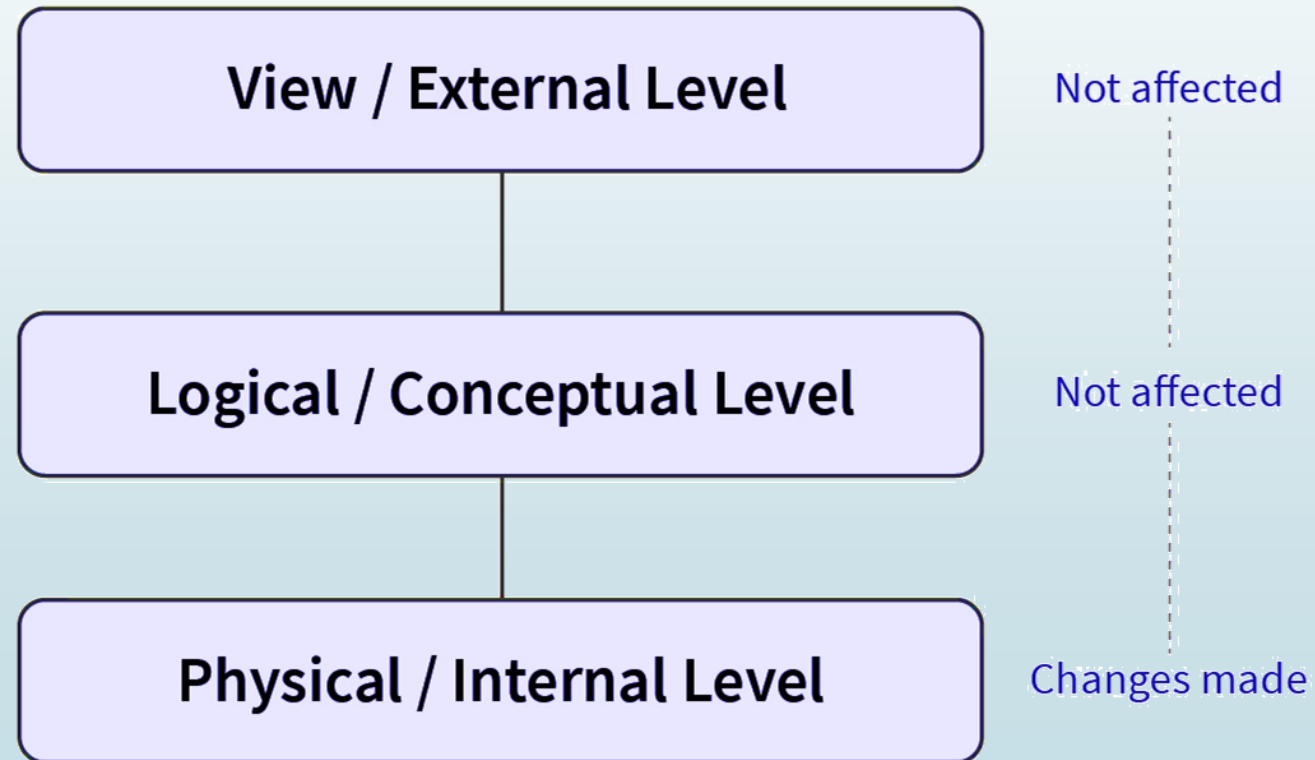
Logical Data Independence

- Changed to constraint can be applied to conceptual schema without affecting external schema (user's view)
- Logical re-organisation -> external schema must work as before

Introduction

Physical Data Independence

- Ability to modify the physical schema of the database without the changing the logical/conceptual or view/external level.





Introduction

Physical Data Independence

- It allows us:
 - to distinguish between conceptual and internal/physical levels.
 - to describe the database logically without needing to identify physical structures.
 - to modify physical storage structures or devices without affecting the conceptual model of the database.

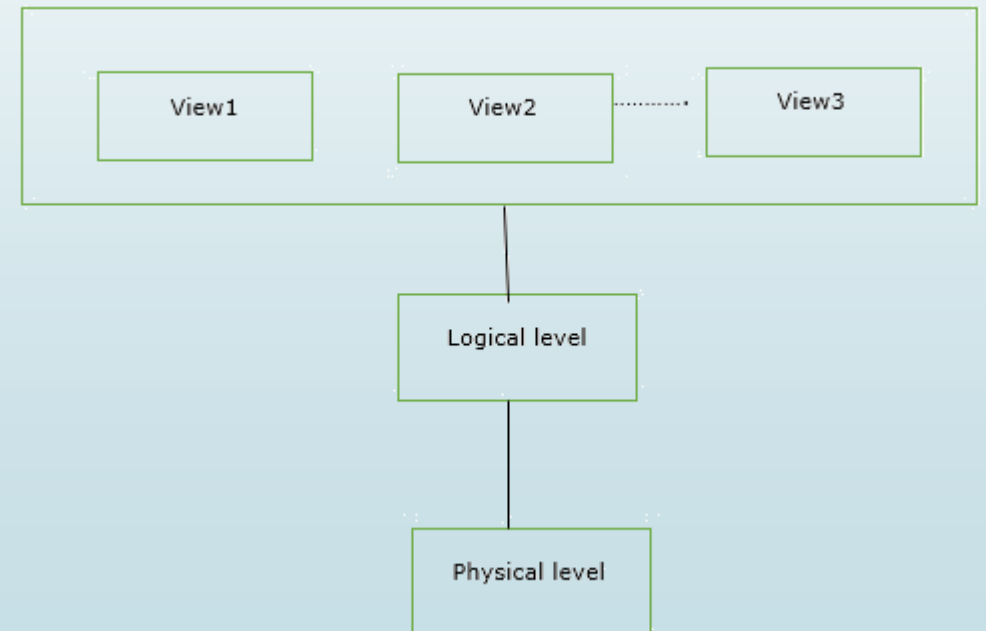
Examples:

- Changing from one data structure to another.
- Making use of new storage technology, such as a hard drive or magnetic tapes
- Change the location of the database from one drive to another.
- Changing the database's file organization.

Introduction

Schema and Instances

- The overall design of the database is called database schema.
- Schema will not be changed frequently.
- It is the logical structure of a database.
- It does not show the data in the database.



Introduction

Schema and Instances

- Overall logical representation of the database that helps us understand how the actual database looks like
- Schema provide us with the information like what are the entities and how they are associated and so on
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram.

Introduction

Schema and Instances

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

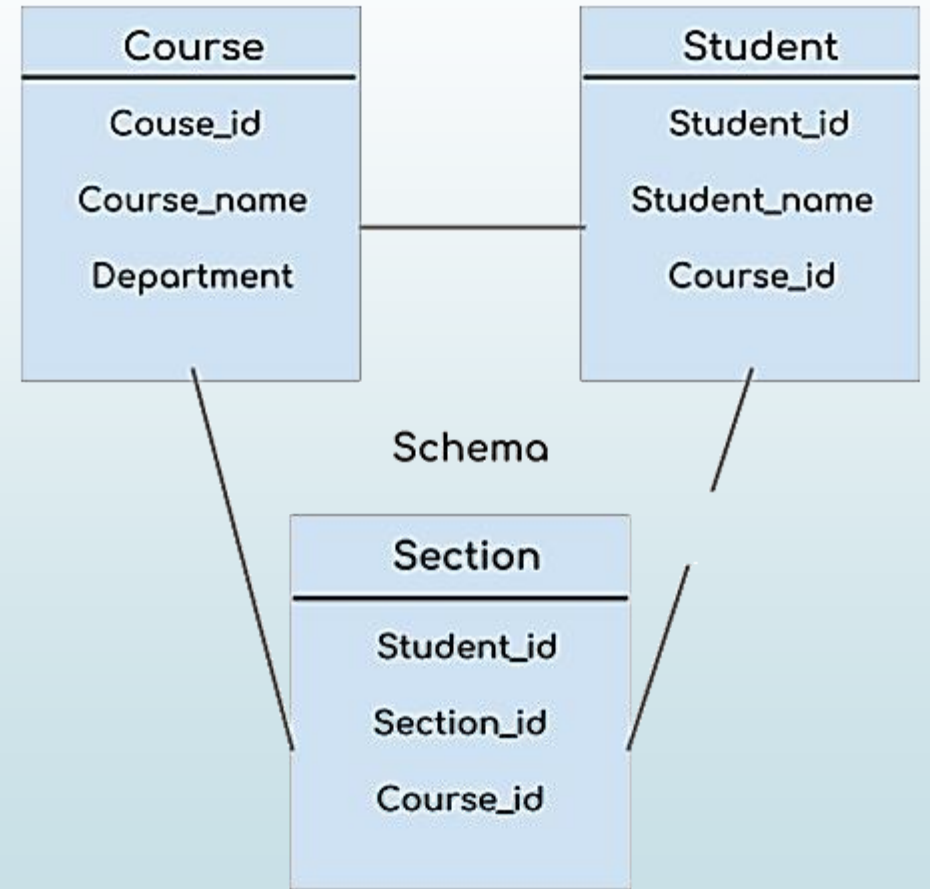
Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------



Introduction

Types of Schema

- Physical schema
 - It is a database design at the physical level.
 - It is hidden below the logical schema and can be changed easily without affecting the application programs.
- Logical schema
 - It is a database design at the logical level.
 - Programmers construct applications using logical schema.
- External
 - It is schema at view level.
 - It is the highest level of a schema which defines the views for end users.

Introduction

Instance

- When the schema framework is filled in with data item values, it is referred as an **instance**
- Information stored in the database at a given point of time.
- It is a dynamic value which keeps on changing.
- The data in the database at a particular moment of time is called a **database state** or **snapshot**
- Database schema defines the attributes in tables that belong to a particular database.

Introduction

Instance: Example

P

P#	PNAME	PRODUCT	PRICE	CITY
----	-------	---------	-------	------

C

C#	CNAME	ADD	CITY
----	-------	-----	------

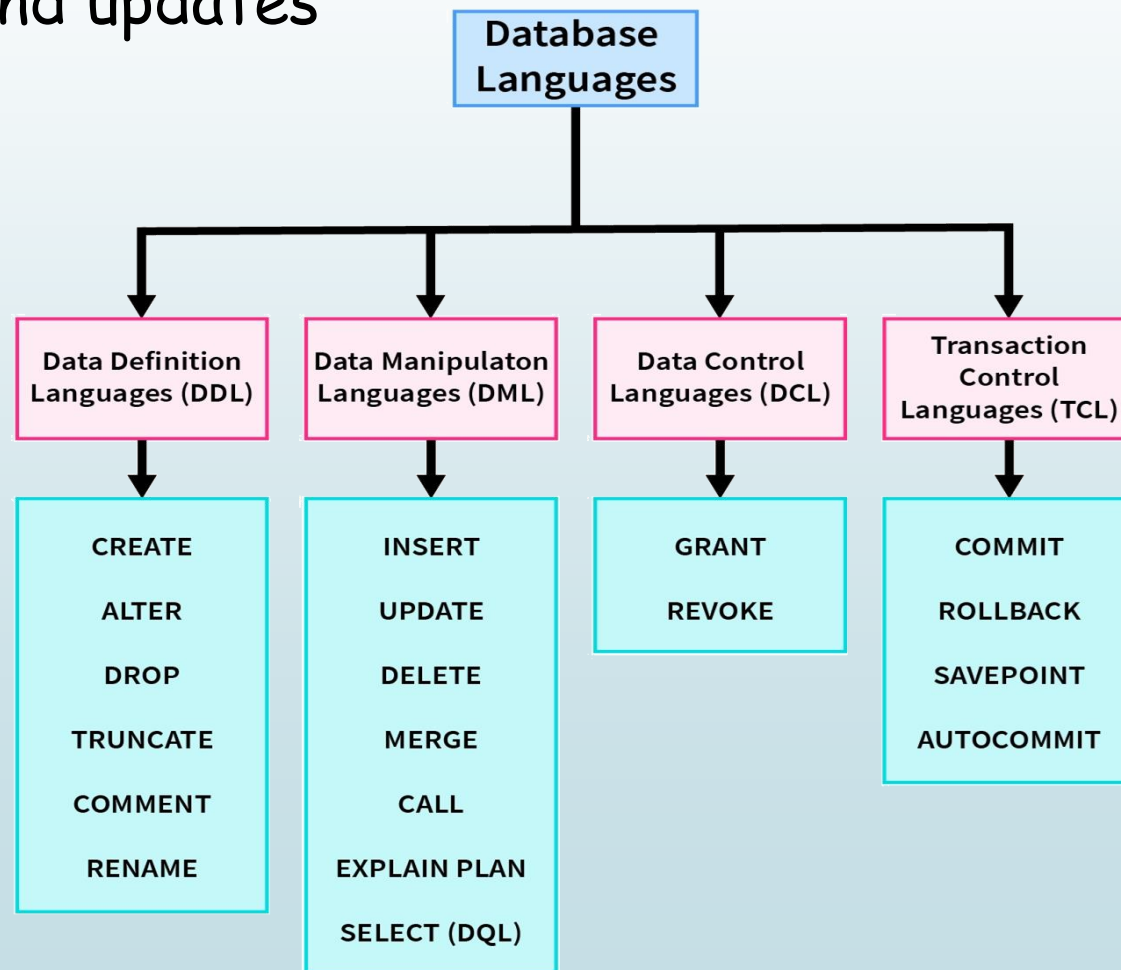
PC

P#	C#	CITY
----	----	------

P#	PNAME	PRODUCT	PRICE	CITY
P1	Rahat Computers	Printer	5000	Patiala
P2	Ruhani info system	Monitor	6000	Jalandhar
P3	IBM	Keyboard	1200	Qadian

Database Languages: Concept of DDL, DML and DCL

A database system provides a **data definition language** to specify the database schema and a **data manipulation language** to express database queries and updates



Database Languages: Concept of DDL, DML and DCL

Data Definition Language (DDL)

- It is a set of special commands that allows us to define and modify the structure and the metadata of the database
- These commands can be used to **create**, **modify**, and **delete** the database structures such as schema, tables, indexes, etc.
- DDL commands can alter the structure of the whole database

Database Languages: Concept of DDL, DML and DCL

Data Definition Language (DDL)

- To create the database instance - **CREATE**
- To alter the structure of database - **ALTER**
- To drop database instances - **DROP**
- To delete tables in a database instance - **TRUNCATE**
- To rename database instances - **RENAME**
- To drop objects from database such as tables - **DROP**
- To Comment - **Comment**

Database Languages: Concept of DDL, DML and DCL

Data Manipulation Language (DML)

- It is a set of special commands that allows us to access and manipulate data stored in existing schema objects.
- These commands are used to perform certain operations such as insertion, deletion, updation, and retrieval of the data from the database.
- These commands deal with the user requests as they are responsible for all types of data modification.
- The DML commands that deal with the retrieval of the data are known as Data Query language.

Database Languages: Concept of DDL, DML and DCL

Data Manipulation Language (DML)

- To read records from table(s) - **SELECT**
- To insert record(s) into the table(s) - **INSERT**
- Update the data in table(s) - **UPDATE**
- Delete all the records from the table - **DELETE**

Database Languages: Concept of DDL, DML and DCL

Data Control Language (DCL)

- It is a set of special commands that are used to control the user privileges in the database system.
- The user privileges include *ALL*, *CREATE*, *SELECT*, *INSERT*, *UPDATE*, *DELETE*, *EXECUTE*, etc.
- We require data access permissions to execute any command or query in the database system.
- Which is controlled using the DCL statements.
- These statements are used to grant and revoke user access to data or the database.
- DCL commands are transactional i.e., these commands include rollback parameters. These commands include:
 - *Grant* : It is used to provide user access to the database or its objects
 - *Revoke* : It is used to revoke user access to the database system

Database Manager and users

- A database typically has many types of users, each of whom may require a different perspective or view of the database
- Based on their interaction, the DB users are divided into
 1. **Actors on the scene:**
 - Whose jobs involve using large database everyday
 2. **Workers behind the scene**
 - Whose work is to maintain the database system environment
 - Development, design and operation of database system
 - Not interest in the database itself

Database Manager and users

1. Actors on the scene:

1. Database Administrators (DBA)

- In a database environment, the primary resource is the database itself, and the secondary resource is the DBMS and related software.

Responsibilities:

- Administering these resources
- Authorizing access to the database
- Coordinating and monitoring use of database
- Acquiring software and hardware resources as needed.
- Accountable for problems such as security and troubleshooting

Database Manager and users

Actors on the scene:

2. Database designers

- Responsible for identifying the data to be stored in the database
- Choosing appropriate structures to represent and store this data
- To communicate with all prospective database users
- Understand their requirements and create a design that meets these requirements
- Develop views of the database that meet the data and processing requirements of these groups.
- The final database design must be capable of supporting the requirements of all user groups.

Database Manager and users

Actors on the scene:

3. End Users

- End users are the people whose jobs require access to the database for querying, updating, and generating reports
- The database primarily exists for their use.
- There are several categories of end users:
 - **Casual end users:**
 - Occasionally access the database
 - They may need different information each time.
 - They use a sophisticated database query interface to specify their requests
 - typically middle- or high-level managers or other occasional browsers.

Database Manager and users

Actors on the scene:

3. End Users

- Naïve or parametric end users:
 - Constantly querying and updating the database, using standard types of queries and updates called canned transactions
 - Eg. Bank customers and tellers check account balances and post withdrawals and deposits.
 - Reservation agents or customers for airlines, hotels, check availability for a given request and make reservations.
 - Employees at receiving stations for shipping companies enter package information
 - Social media users post and read items on social media Web sites.

Database Manager and users

Actors on the scene:

3. End Users

- **Sophisticated end users:**
 - Include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS
 - To implement their own applications to meet their complex requirements.
- **Standalone users:**
 - Maintain personal databases by using ready-made program packages
 - That provide easy-to-use menu-based or graphics-based interfaces.
 - An example is the user of a financial software package that stores a variety of personal financial data.

Database Manager and users

Actors on the scene:

4. System analyst and Application programmers (Software Engineers)

- They determine the requirements of end users, especially naive and parametric end users
- Develop specifications for standard canned transactions that meet these requirements.
- Application programmers implement these specifications as programs; then they test, debug, document, and maintain these canned transactions.
- Such analysts and programmers: commonly referred to as software developers or software engineers
- should be familiar with the full range of capabilities provided by the DBMS to accomplish their tasks.



Database Manager and users

2. Workers behind the scene:

1. DBMS System designers and implementers

- Design and implement DBMS modules and Interface as a packages

2. Tool developers

- Persons who design and implement tools and software packages

3. Operators and maintenance personal

- Responsible for actual runtime and maintenance of the hardware and software



Questions ?



Thank You!