Course Details

Course Title	:	Database Management System
Semester	:	Third
Course Code	:	313315
TH-ESE	:	70
TH-PA	:	30
PR-ESE	:	25#
PR-PA	:	25
SA-PA	:	25
SLA	:	25

Syllabus Details

Sr.no	Chapter Name	Marks
1	Database System Concepts	12
2	Relational Data Model	12
3	Interactive SQL and Advance SOLSQL Performance Tuning	18
4	PL/SQL Programming	18
5	Database security and Transaction Processing	10
Total		**70**

Unit - I Introduction To Database System

1.1 Database concepts:-Data, Database, Database management system, File system Vs DBMS, Applications of DBMS, Data Abstraction, Data Independence, Database Schema, The Codd's rules, Overall structure of DBMS

1.2 Architecture:- Two tier and Three tier architecture of database.

1.3 Data Models:- Hierarchical, Networking, Relational Data Models

Course Introduction

- Each and every organization like shopping mall, hospital, banking, institutes, industry needs to share huge amount of data in effective manner.

- This course aims to develop skills in students to create, store, modify, manage and extract information from a database.

- Database system can be used as a backend for developing database applications.

What is Database Management System (DBMS)

- ✔ A Database Management System (DBMS) is software designed to store, retrieve, define, and manage data in a database.

- ✔ DBMS software primarily functions as an interface between the end user and the database, simultaneously managing the data, the database engine, and the database schema in order to facilitate the organization and manipulation of data.

How it Works...

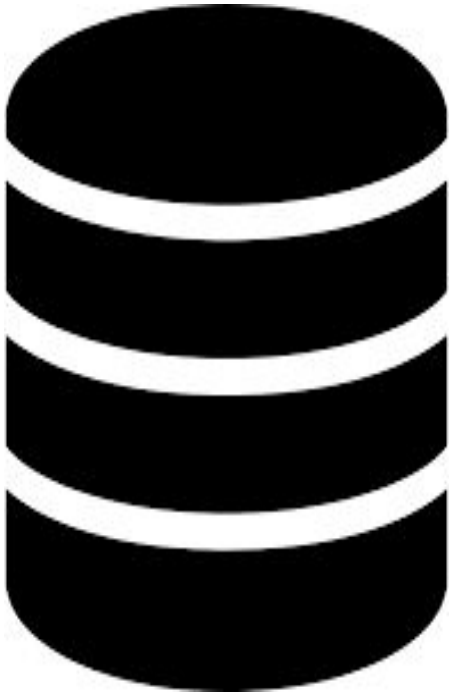
User

User Communicate to DBMS interface for efficient and faster way to fulfilling her requirement's

Software application based on Business Logic

DBMS

Different program to insert , delete , update and query database for information.



Database

Concept of Data , Database , DBMS

- **Data** is the **information** which has been translated into a form that is more convenient to process or move.

- **Database-** The collection of **related data** is termed as Database which is organized in such a way that it can be **easily retrieved and managed**.

- **A Database Management System (DBMS)** is **System software** which manages the data. It can perform various tasks like **creation , retrieval, insertion, modification and deletion** of data to manage it in a systematic way as per requirement.

Purpose of Database System

- A major **purpose** of a **database system** is to provide users with an abstract view of the data. That is, the **system** hides certain details of how the data are stored and maintained.

File Processing System

- File Processing System is a Computer based system in which all the information is stored in various **computer files**.

- It is useful but as the requirement of **data processing** and the **size of data increases** ,the drawback of systems comes.

Disadvantages of Traditional File Processing System

1. Data Redundancy
2. Data Inconsistency
3. Limited Data Sharing
4. Difficulty in Accessing Data
5. Data Dependence
6. Poor Data Control
7. Problem of Security
8. Concurrency Problems
9. Poor Data Modelling of Real World
10. Data isolation
11. Integrity Problems
12. Atomicity Problem

Advantages of DBMS over File Processing System

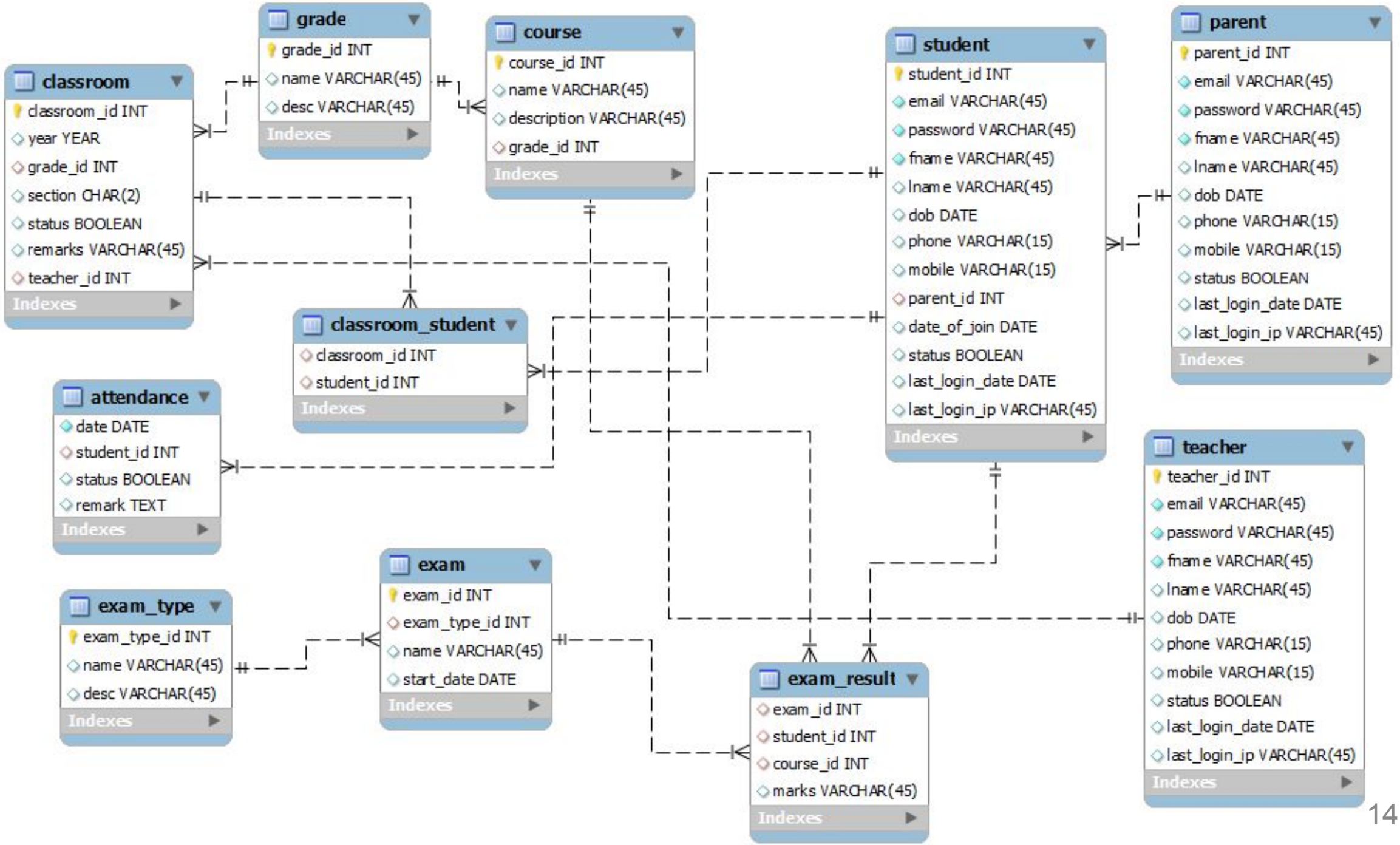
1. Controlling Data Redundancy
2. Data Consistency
3. Sharing of Data
4. Data Independence
5. Data Control
6. Security
7. Control over Concurrency
8. Data Modelling of Real World

Disadvantages of DBMS

1. Increased costs
2. Complexity
3. Size
4. Frequent upgrade/replacement cycle
5. Higher impact of a failure
6. Performance

Application of DBMS

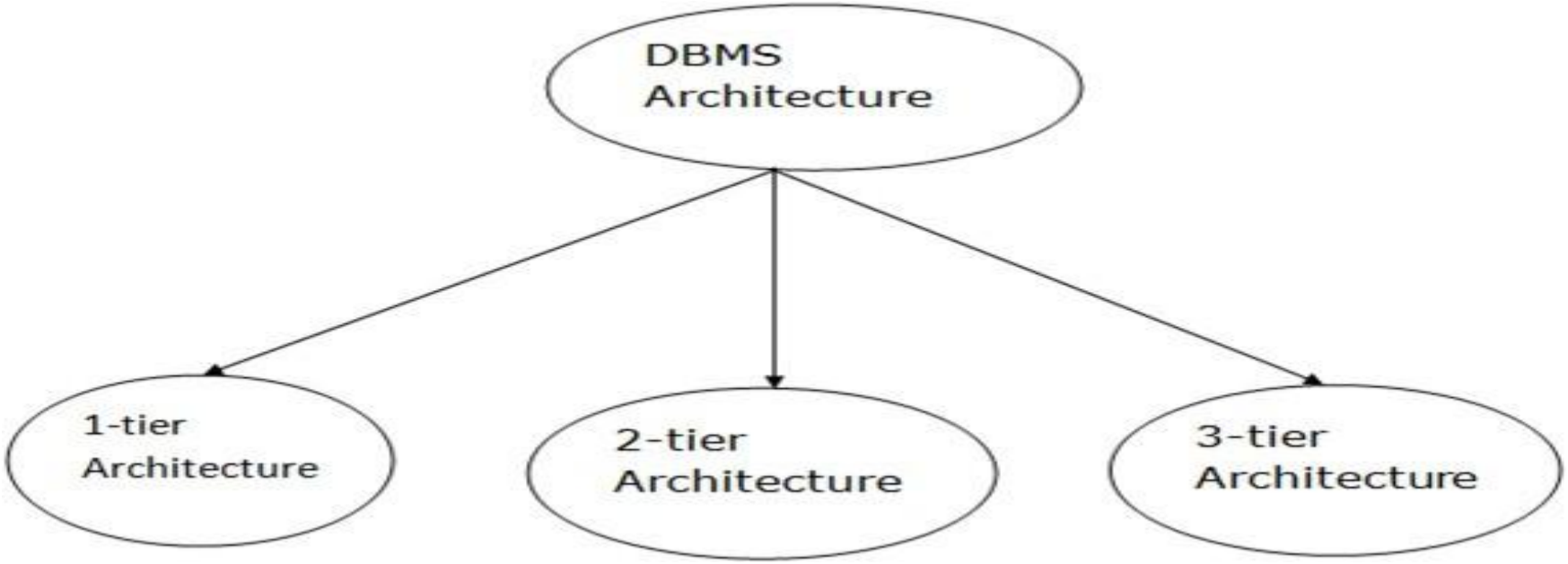
1. Telecom
2. Banking
3. Industry
4. E-Commerce
5. Airlines
6. Education Systems
7. Railway Reservation System
8. Library Management System
9. Social Media Sites


grade
grade_id INT
name VARCHAR(45)
desc VARCHAR(45)
Indexes
course
course_id INT
name VARCHAR(45)
description VARCHAR(45)
grade_id INT
Indexes
classroom
classroom_id INT
year YEAR
grade_id INT
section CHAR(2)
status BOOLEAN
remarks VARCHAR(45)
teacher_id INT
Indexes
student
student_id INT
email VARCHAR(45)
password VARCHAR(45)
fname VARCHAR(45)
lname VARCHAR(45)
dob DATE
phone VARCHAR(15)
mobile VARCHAR(15)
parent_id INT
date_of_join DATE
status BOOLEAN
last_login_date DATE
last_login_ip VARCHAR(45)
Indexes
parent
parent_id INT
email VARCHAR(45)
password VARCHAR(45)
fname VARCHAR(45)
lname VARCHAR(45)
dob DATE
phone VARCHAR(15)
mobile VARCHAR(15)
status BOOLEAN
last_login_date DATE
last_login_ip VARCHAR(45)
Indexes
classroom_student
classroom_id INT
student_id INT
Indexes
attendance
date DATE
student_id INT
status BOOLEAN
remark TEXT
Indexes
exam
exam_id INT
exam_type_id INT
name VARCHAR(45)
start_date DATE
Indexes
exam_type
exam_type_id INT
name VARCHAR(45)
desc VARCHAR(45)
Indexes
exam_result
exam_id INT
student_id INT
course_id INT
marks VARCHAR(45)
Indexes
teacher
teacher_id INT
email VARCHAR(45)
password VARCHAR(45)
fname VARCHAR(45)
lname VARCHAR(45)
dob DATE
phone VARCHAR(15)
mobile VARCHAR(15)
status BOOLEAN
last_login_date DATE
last_login_ip VARCHAR(45)
Indexes

Three Level Architecture of Database System

- A data base system is collection of **related data** and **system software** which manages the **data**.
- The data is generally stored in a **detailed and complex manner**.
- It is important to provide an **abstract view** of data to the user.

Three Level Architecture of Database System



Three Level Architecture of Database System

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

Eg:- Ms Access,MS Word.

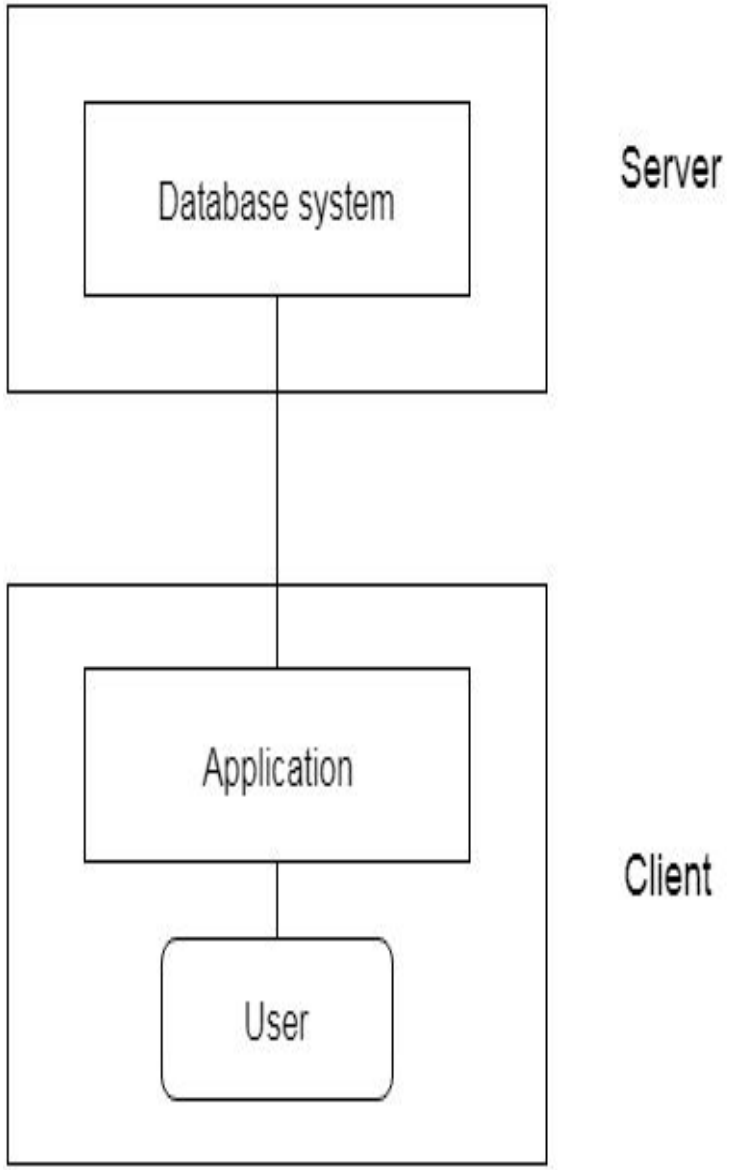
2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.

- The user interfaces and application programs are run on the **client-side**.

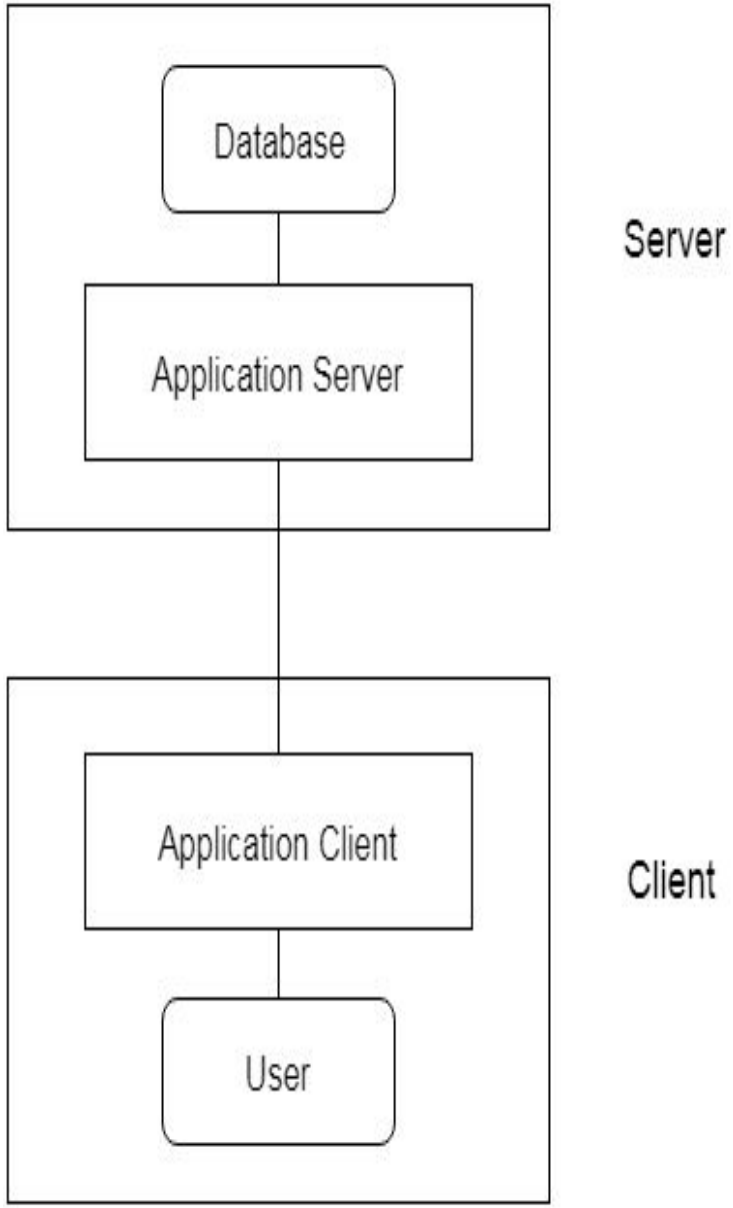
- The server side is responsible to provide the functionalities like: **query processing** and **transaction management.**

- To communicate with the DBMS, client-side application establishes a connection with the server side.



3-Tier Architecture

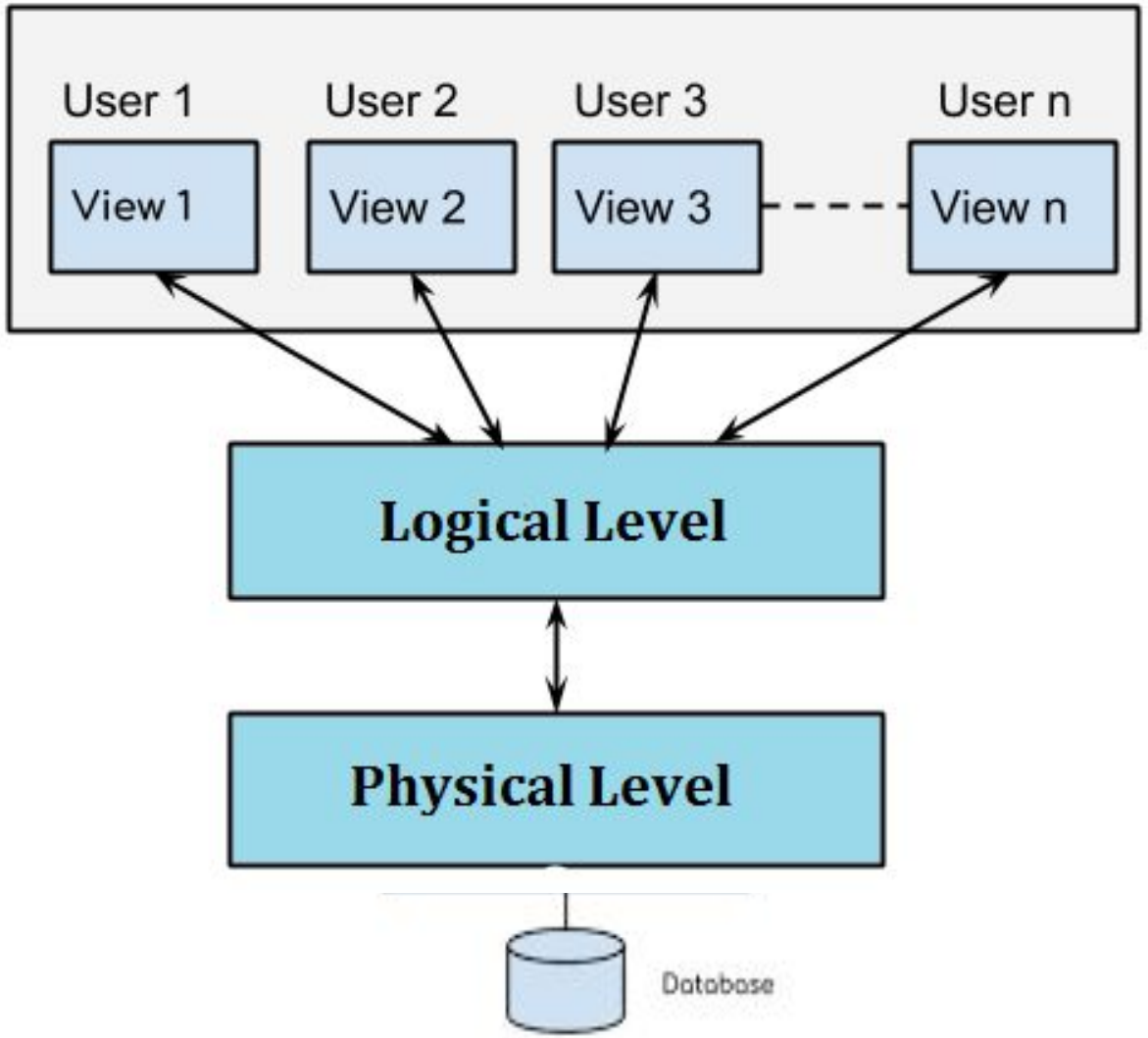
- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.



Data Abstraction

- **Extracting the important** data by **ignoring the remaining** irrelevant details is known as **abstraction**.
- This process of **hiding irrelevant** details from user is called **data abstraction.**
- The **complexity of database** can be hiding from user by using different **level of abstraction.**

Different Levels of Data Abstraction



Different Levels of Data Abstraction cont.

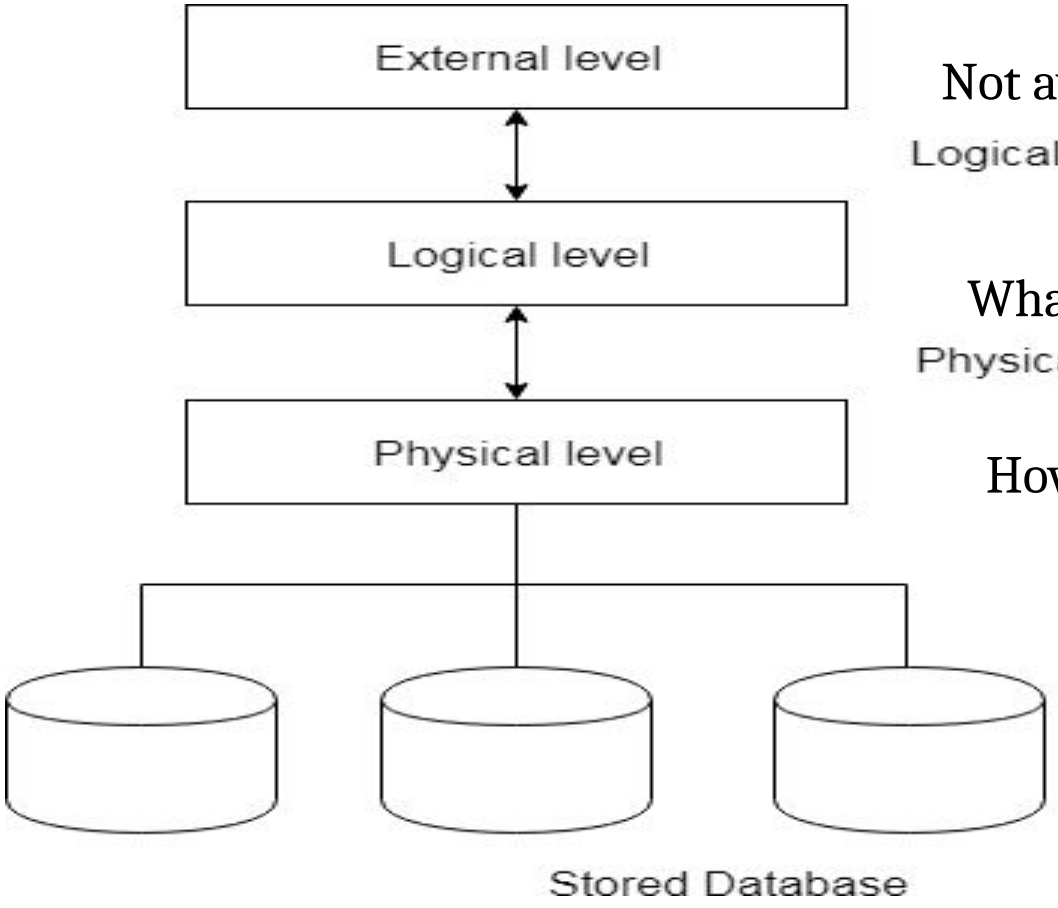
1. **Physical Level –**This is the **lowest level** in the three level architecture. The physical level describes **how data is actually stored** in the database. In the lowest level, this data is stored in the **external hard drives** like hard disk, magnetic tapes etc.

2. **Logical Level-** This is the **next higher level** of abstraction which is used to describe **what data the database stores**, and what **relationships exist** in between the data items. Database administrators use the logical level of abstraction to decide **what information to keep** in a database.

3. **View Level-** It is the **highest level** of data abstraction. This level describes the **user interaction with database** system. End user interacts with system with the help of **GUI** and enters the details at the screen at view level. User is not aware of **how the data is stored** and **what data is stored**; such details are hidden from them.

DBMS Abstraction

Example:- IRCTC



Not aware of storing the user just interact with database.

Logical Data Independence

What data is stored in the database

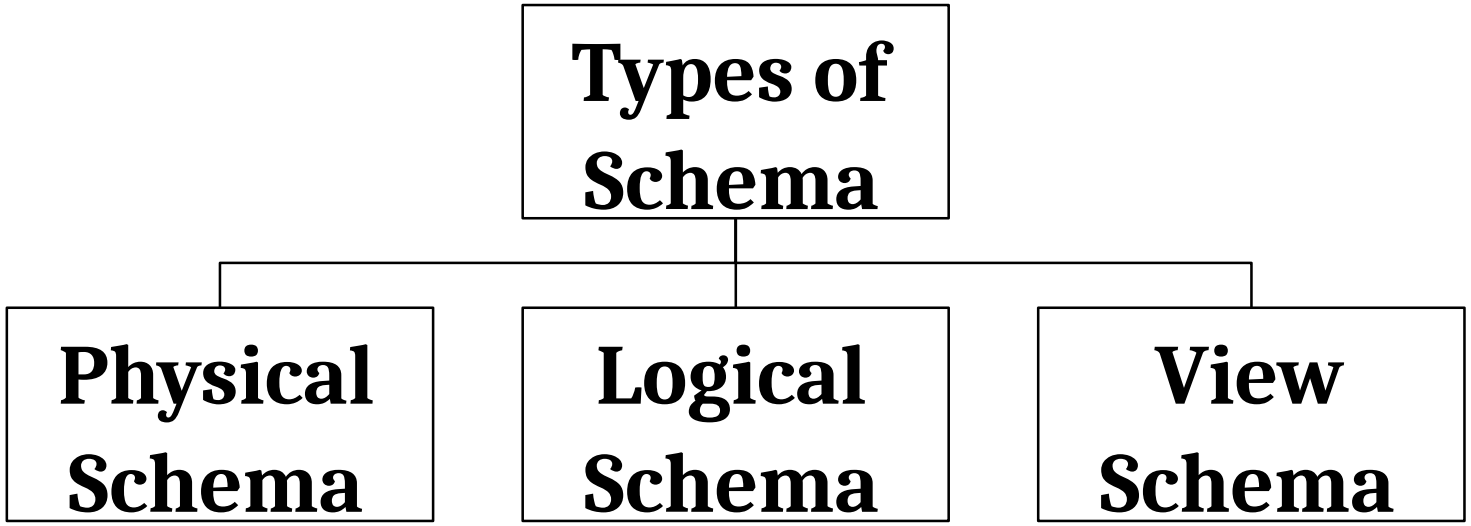
Physical Data Independence

How data is actually stored in the database

Instance and Schema

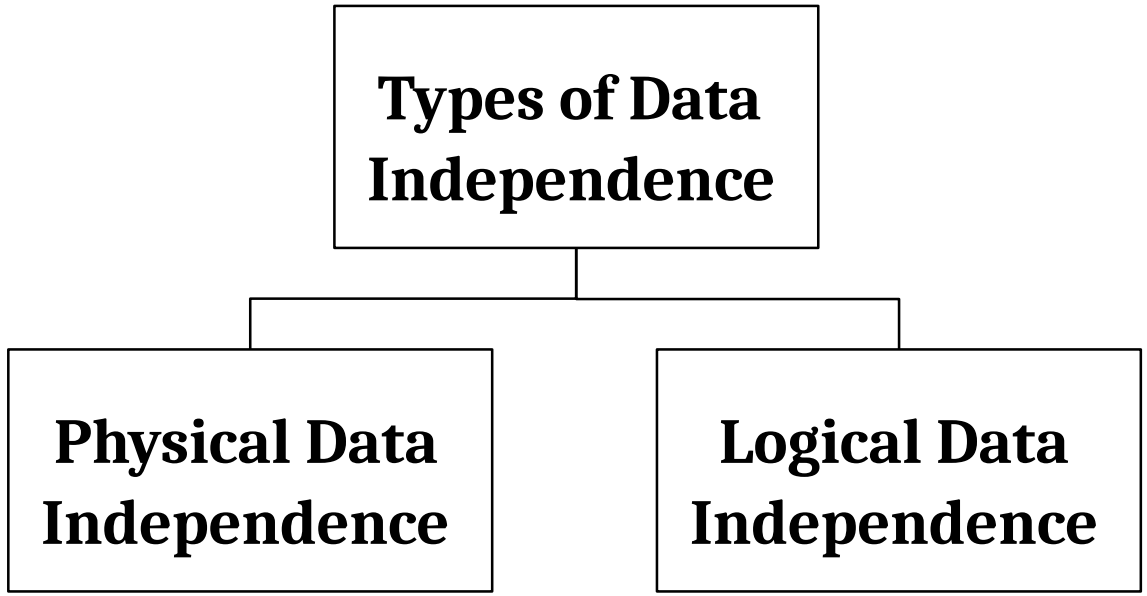
- **Instance-**The data is stored in the database **at particular moment** is called as **instance** of the database.

- **Schema-** The design of a database is called the **schema.**

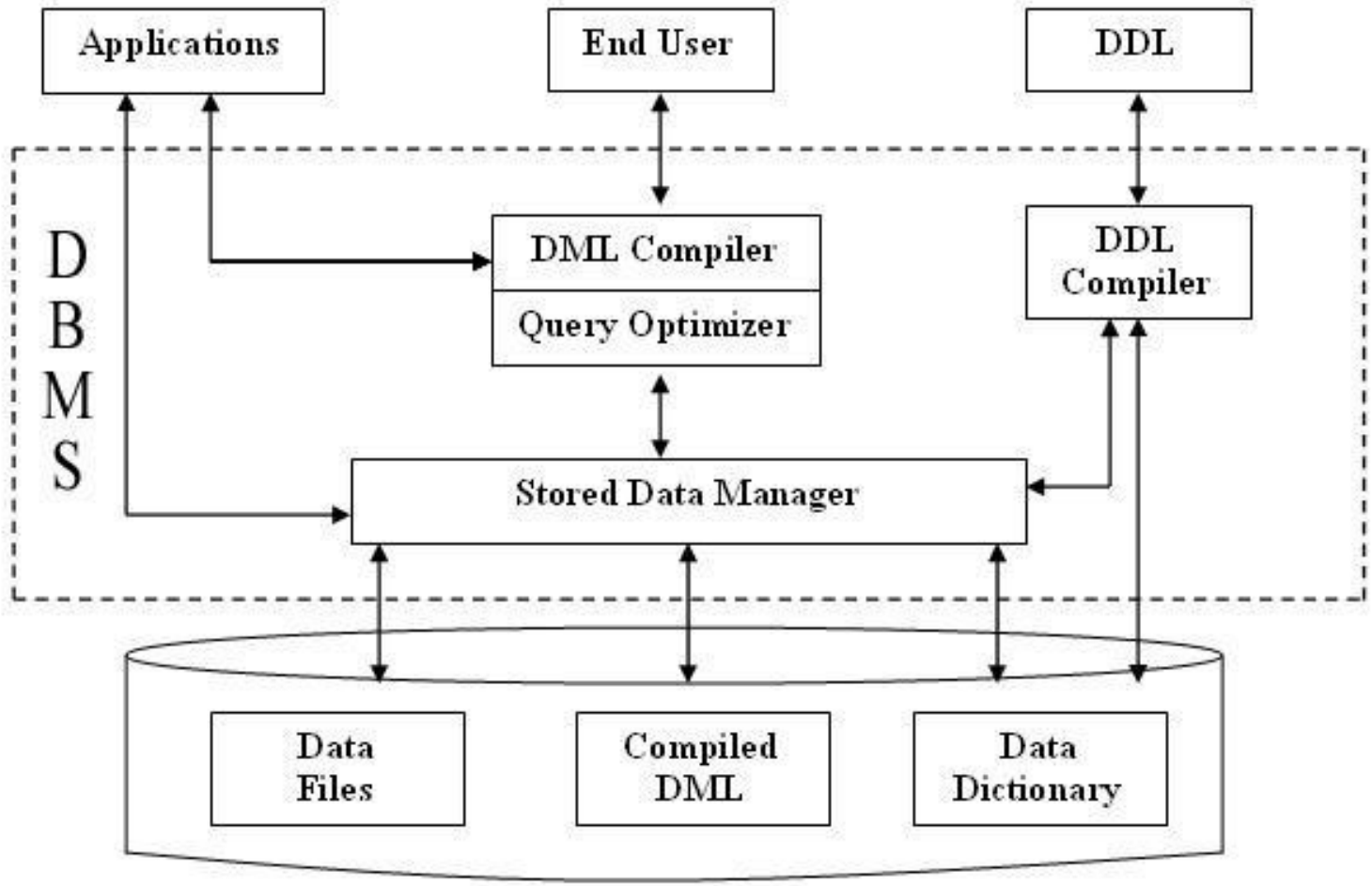


Data Independence

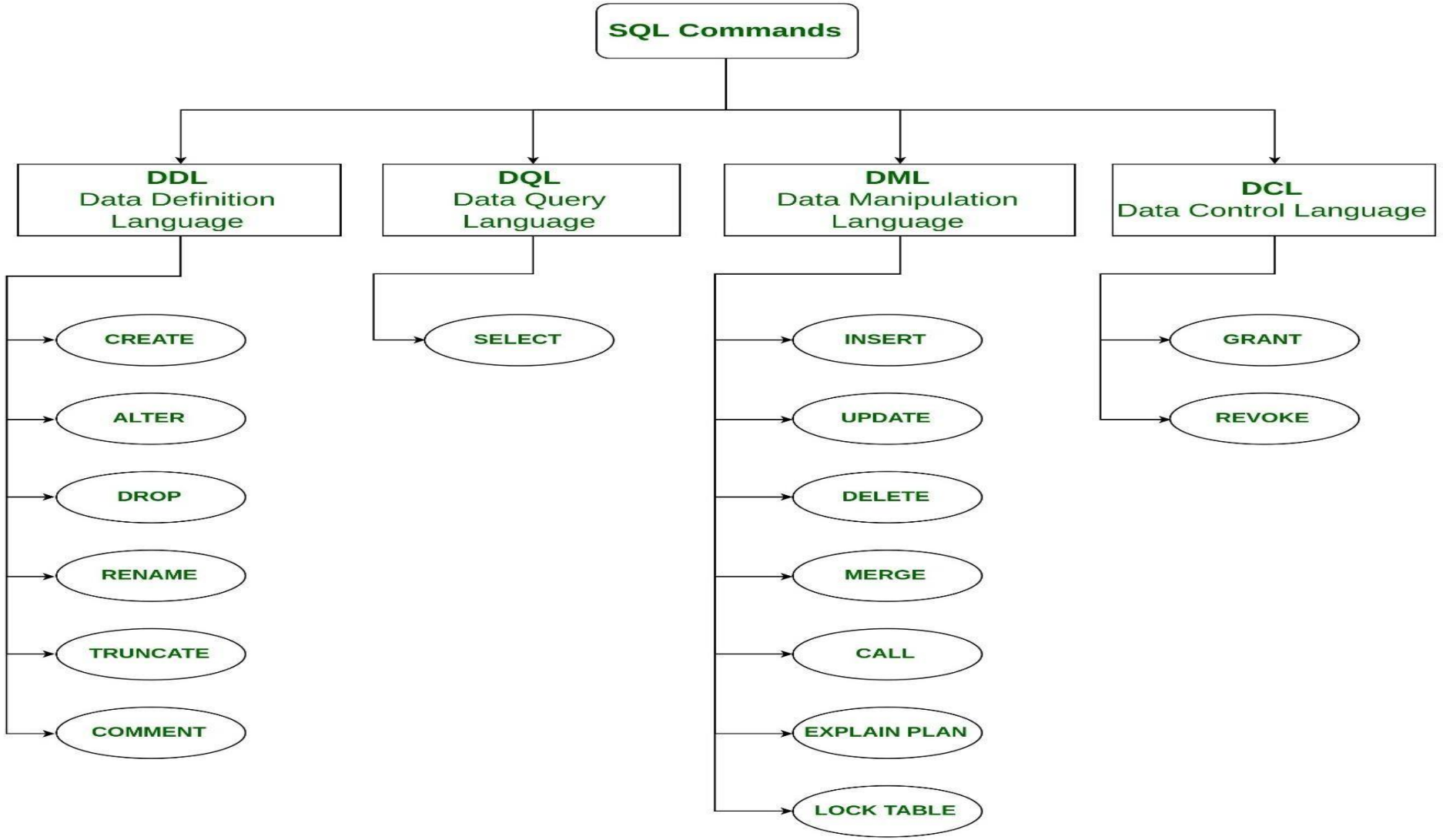
Data Independence -The ability to modify schema definition in one level without affecting schema definition in the next higher level is called data independence.



Structure of DBMS



Types of SQL Commands



Structure of DBMS

1. **DDL Compiler:** It converts the DDL commands into set of table containing **metadata** stored in data dictionary.

2. **DML Compiler:** It receives the DML commands from application program and converts DML commands into **object code** for understanding of database.

3. **Query Optimizer:** It optimized the **object code** to execute query in best way and then send to store data manager.

4. **Storage Data Manager:** It is a program module which is responsible for storing, retrieving and updating data in the database. It receives the request from query optimizer to machine understandable form.it makes the actual request inside the database.

Structure of DBMS

1. **Data Files:** It stores the **database** itself.

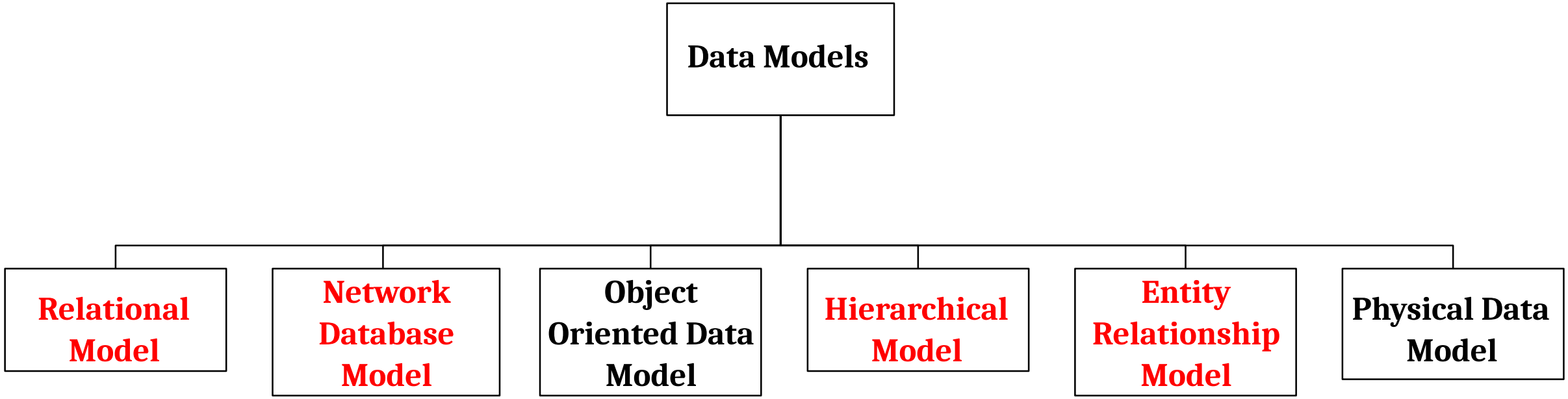
2. **Compiled DML:** The DML Compiler converts the high level Queries into low level file access commands known as **Compiled DML.**

3. **Data Dictionary:** It stores **metadata**, i.e. file, storage , attributes , access path , etc.

Data Models

- The process of analysis of data object and their relationships to other data objects is known as **data modeling.**
- It is the **conceptual representation** of data in database.
- It is the first step in **database designing**.
- Data models define **how data is connected to each other** and **how they are processed** and **stored inside the system**.
- A data model provides a way to describe the **design of a database** at the **physical, logical and view levels.**

Types of Data Models



1.Relational Model

1. Record Based Logical Model/Relational Model-

- Developed by **E.F.Codd.**
- Tables are used and also known as **relations.**
- Record are known as **tuples** and fields are known as **attributes.**
- Every record must have a **unique identification or key.**
- **Advantages-**
  1. Support SQL
  2. Flexible

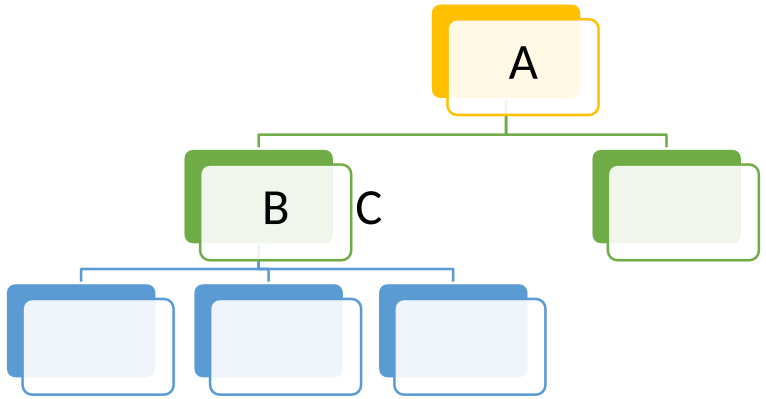
Key

Tuple →

Stud_ID	Stud_Name	DOB
101	Prajakta	03/03/1995
102	Rakesh	13/01/1996
103	Rahul	16/08/1995

2.Hierarchical Model

2. Hierarchical Model



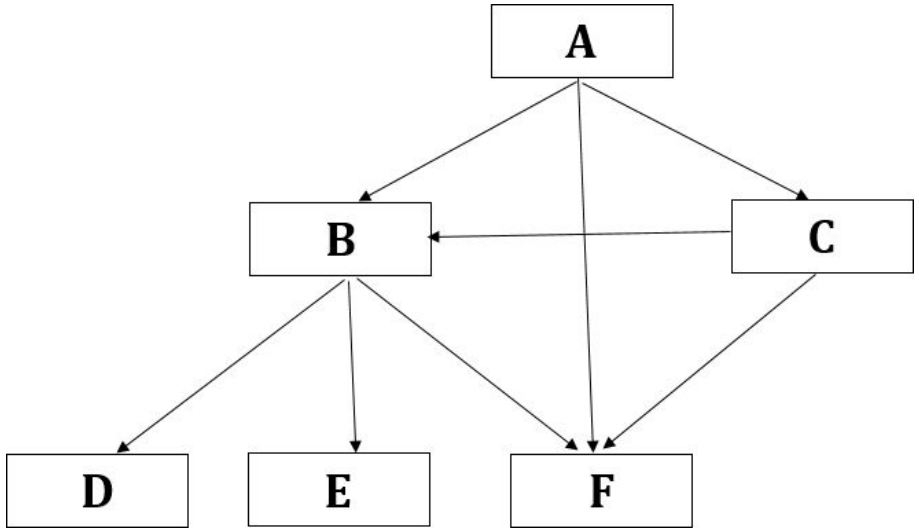
- Organized into a **tree structure.**
- **Parent-child** relationship.
- Data is stored in the form of **Record.**
- Record is collection of **fields** and it contains only **one value.**
- One **parent can have many child nodes** but **one child node can have only one parent node.**
- **Advantages-**
  1. Simple to Understand.
  2. Database Integrity 3.Efficient

3.Network Database Model

3. Network Database Model

- Extended type of Hierarchical data Model but any **child can have multiple parent.**
- **No need of Parent-child** relationship.
- Allows multiple **Records linked in same file.**
- **Advantages-**

1. Design is Simple.
2. Capability to handle various relationship.
3. Easy to access.

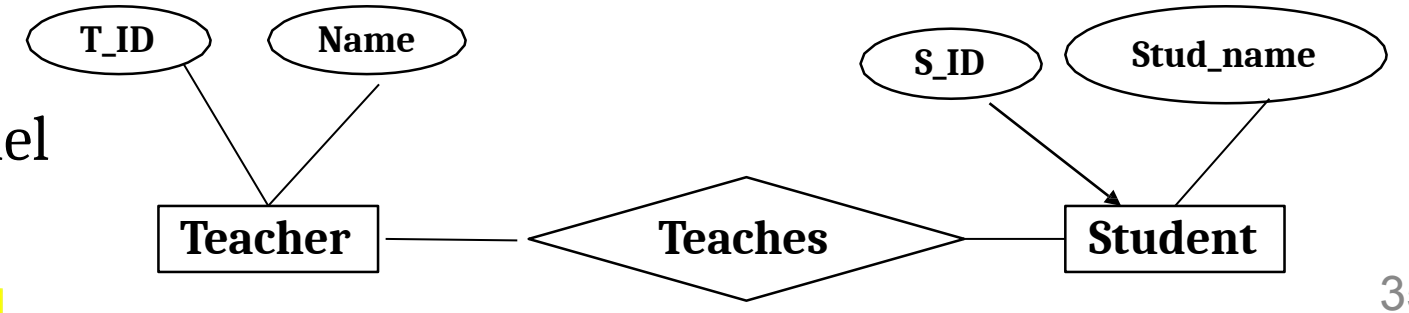


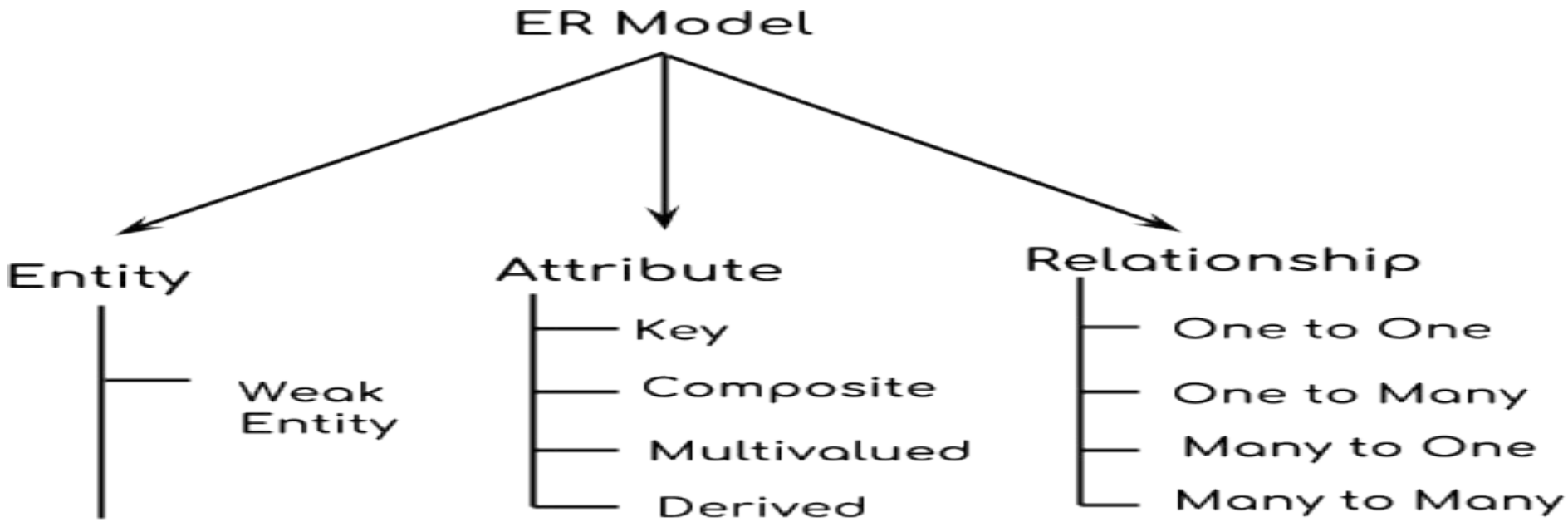
4.Model/E-R Model

4. Entity Relationship Model/E-R Model

- Represents overall structure of database.
- **Design technique of database**.
- Real world object can be easily mapped with entities of E-R model.
- Diagrams are used which known as Entity relationship digram,ER digrams or ERD's.
- Concept of ER model-Entity,attribute,Relationship,constraints and keys.
- **Advantages-**

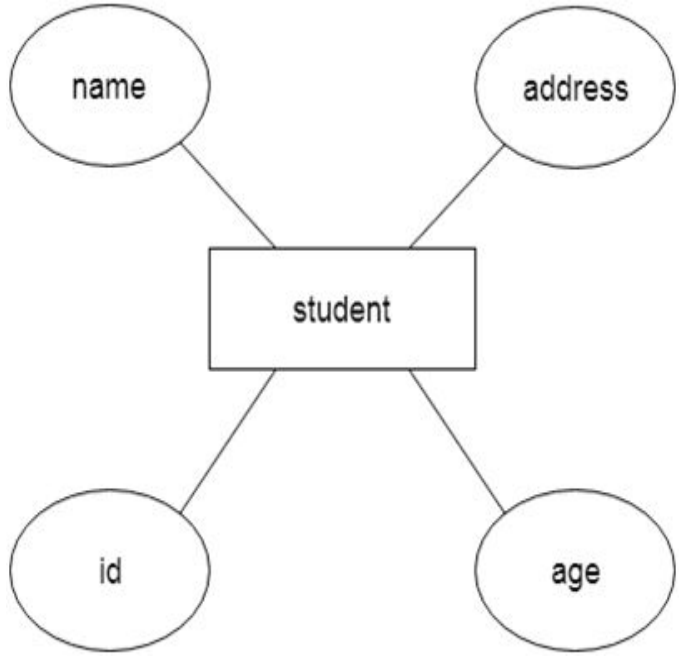
1. Design is Simple.
2. Effective representation
3. Connected with Relational Model





Components of ER Diagram

Entity and Entity Set

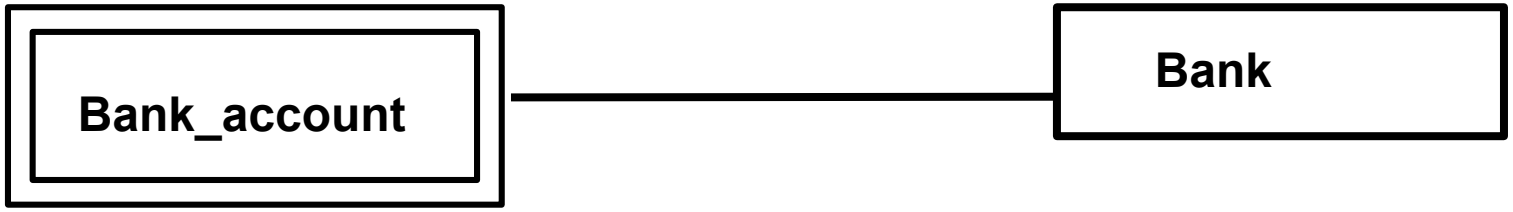


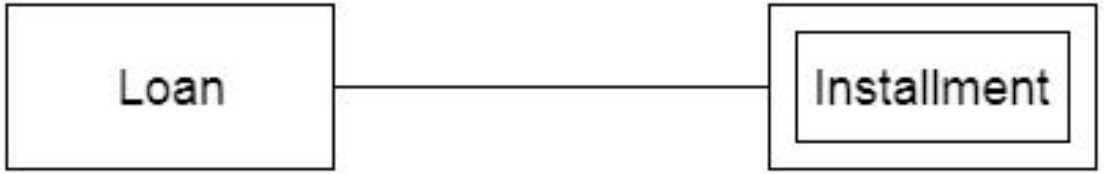
- An entity is a thing that exists either **physically or logically.**
- An entity is nothing but a thing having **its own properties.**
- These properties helps to **differentiate** the **object from other objects**.
- An **entity set** is a set of entities which share the **same properties.**

Types of Entity

1.Strong Entity or Regular Entity- If an entity having it's **own key attribute** specified then it is a strong entity. Key attribute is used to identify that entity uniquely among set of entities in entity-set. Strong entity is denoted by a single rectangle.

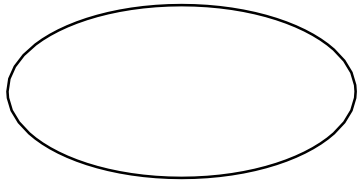
2.Weak Entity- The entity which **does not have any key attribute** is known as weak entity. The weak entity has a partial discriminator key. Weak entity depends on the strong entity for its existence. Weak entity is denoted with the double rectangle.

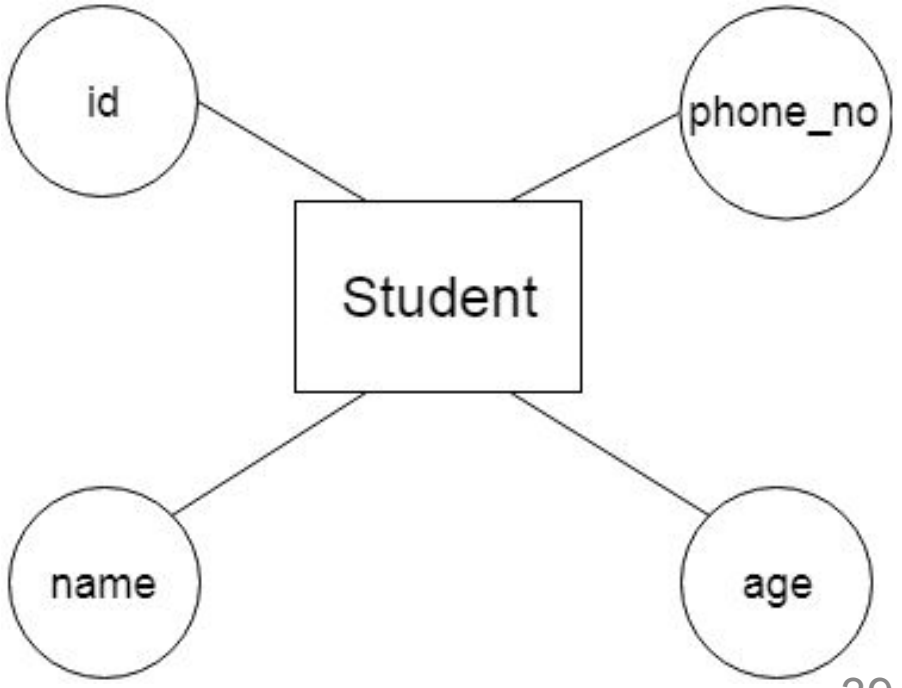




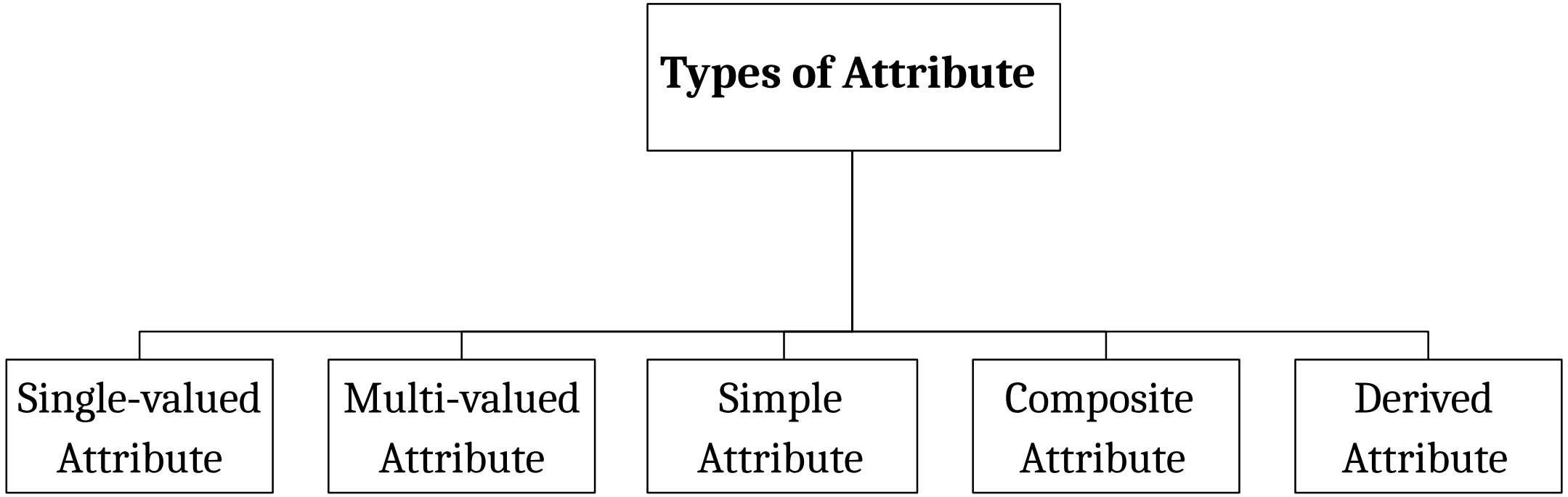
Attribute

- An attribute is a **characteristic** of an entity.
- Entities are represented by means of their attributes.
- All attributes have their **own specific values.**
- Attribute is denoted by a Ellipse.



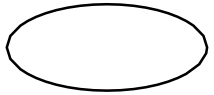


Types of Attribute



Types of Attribute

1. **Single- valued Attribute-** A Single-valued attribute is the attribute which **can hold single value** for the single entity.



2. **Multi-valued Attribute -**A multi-valued attribute is the attribute which **can hold multiple values** for the single entity.

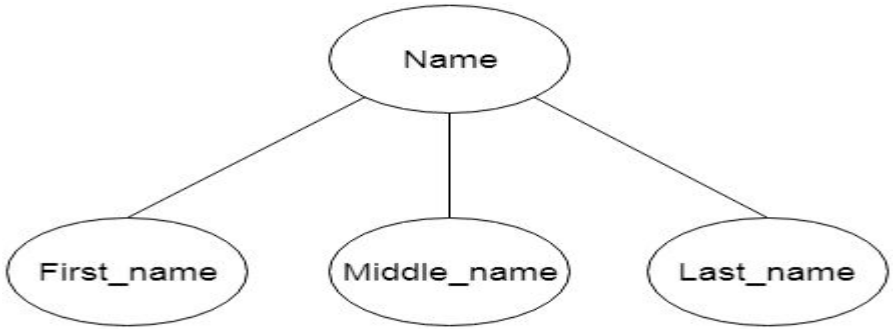
For example, a student can have more than one phone number.

3. Simple Attribute- An attribute whose value **cannot be further divided** is known as simple attribute. That means it is atomic in nature.

Eg GST NO can be Divided.

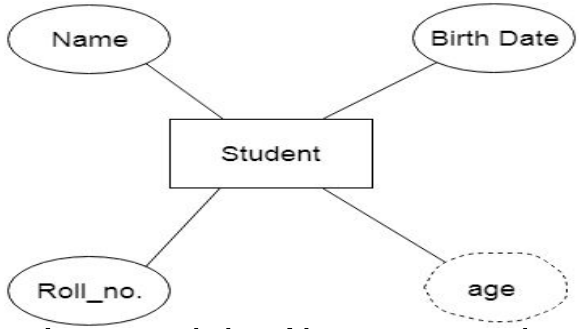
Types of Attribute

4. Composite Attribute- The composite attributes are the attributes which **can be further divided** into sub parts. These sub parts represent the basic entities with their independent meaning.

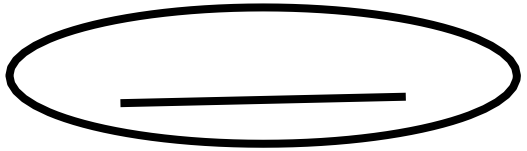


5. Derived Attribute -The attribute which is **not physically exist in database**, but its value can be calculated from the other present attributes is known as derived attribute.

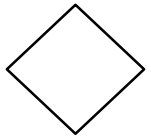
It can be represented by a dashed ellipse. For example, A person's age changes over time and can be derived from another attribute like Date of birth.



6. Key Attribute-The key attribute is used to represent the main characteristics of an entity. It represents a **primary** key. The key attribute is represented by an **ellipse** with the text underlined.



Relationship

- The association between two different entities is called as **relationship.** It is denoted by Diamond. 
- Line is used to links attributes to entity and entity set to relationship sets. 

Mapping Cardinality in E-R Diagram

- Mapping Cardinality in E-R Diagram

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

2. One-to-many

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

3. Many-to-one

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.

4. Many-to-many

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

Component of ER Model

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

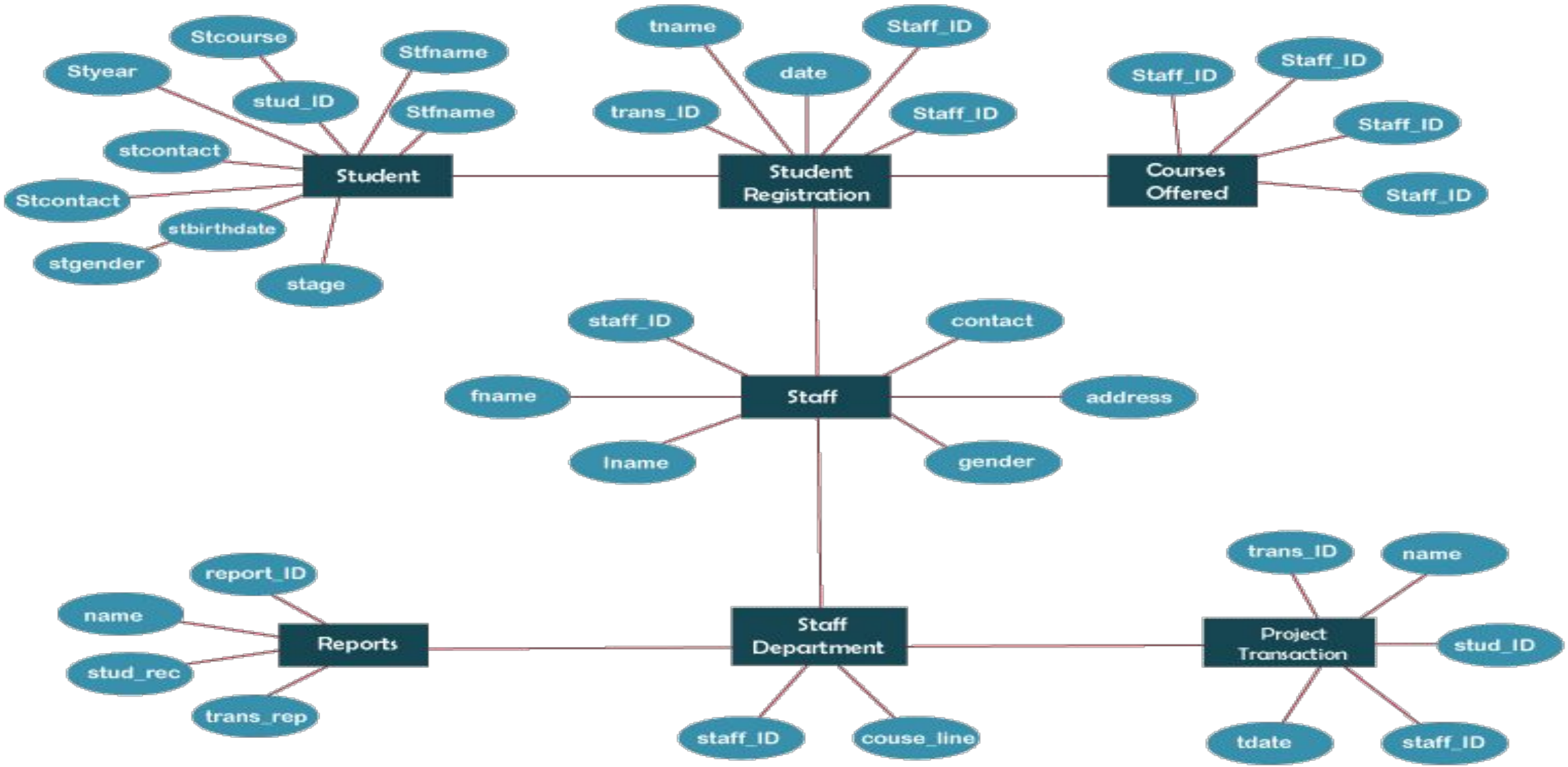
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

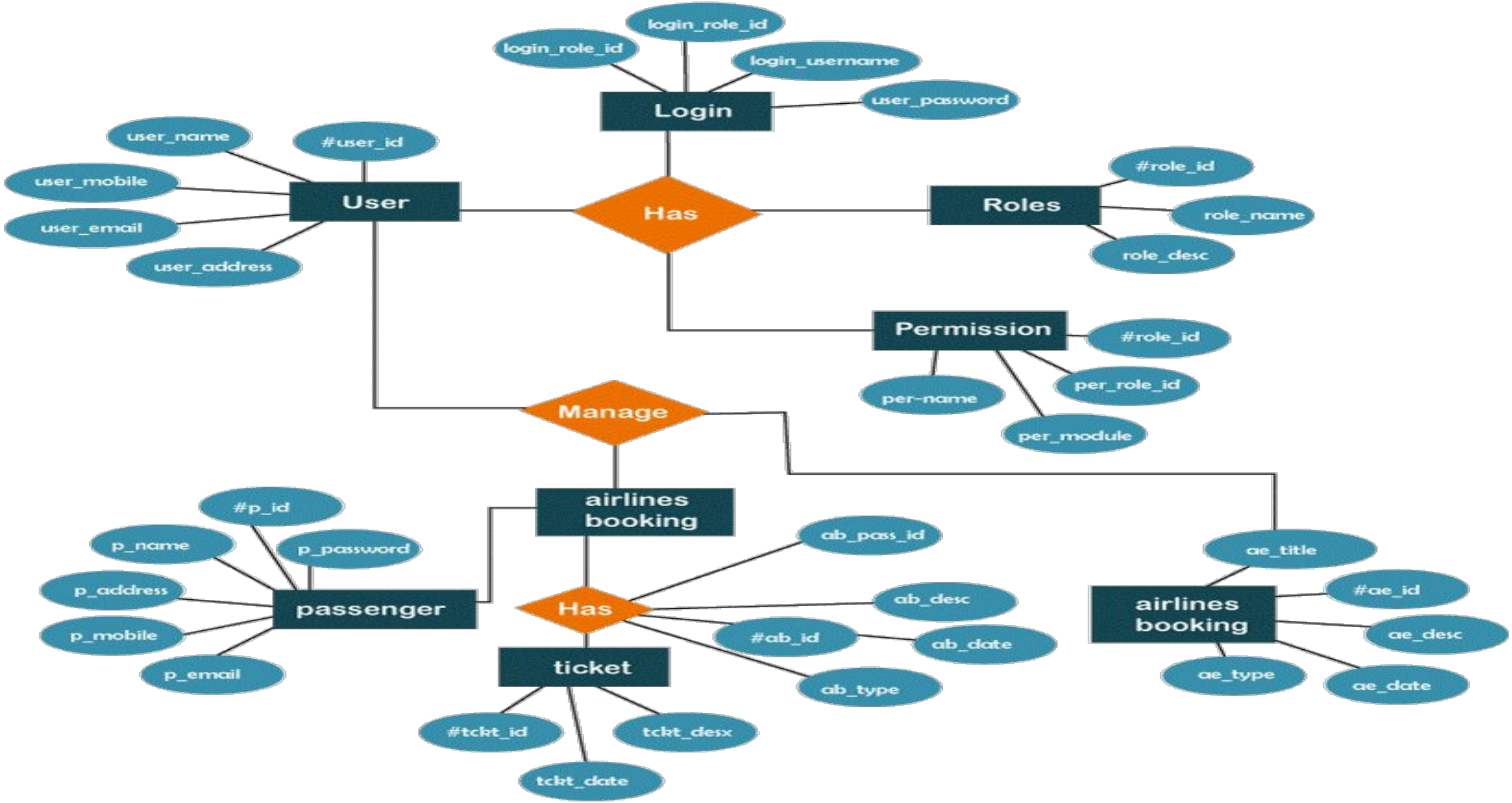
Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

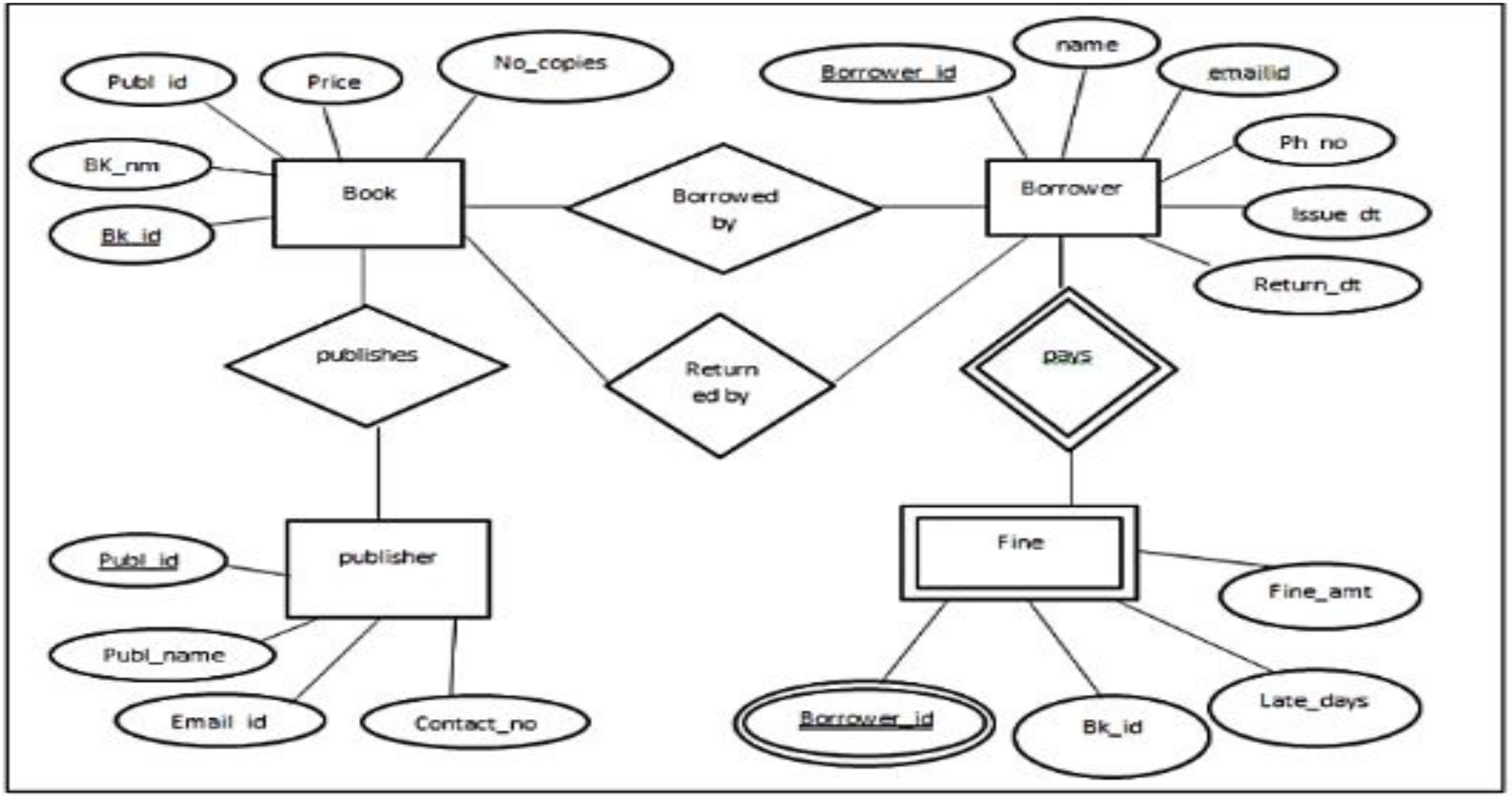
Double Rectangles: Weak Entity Sets

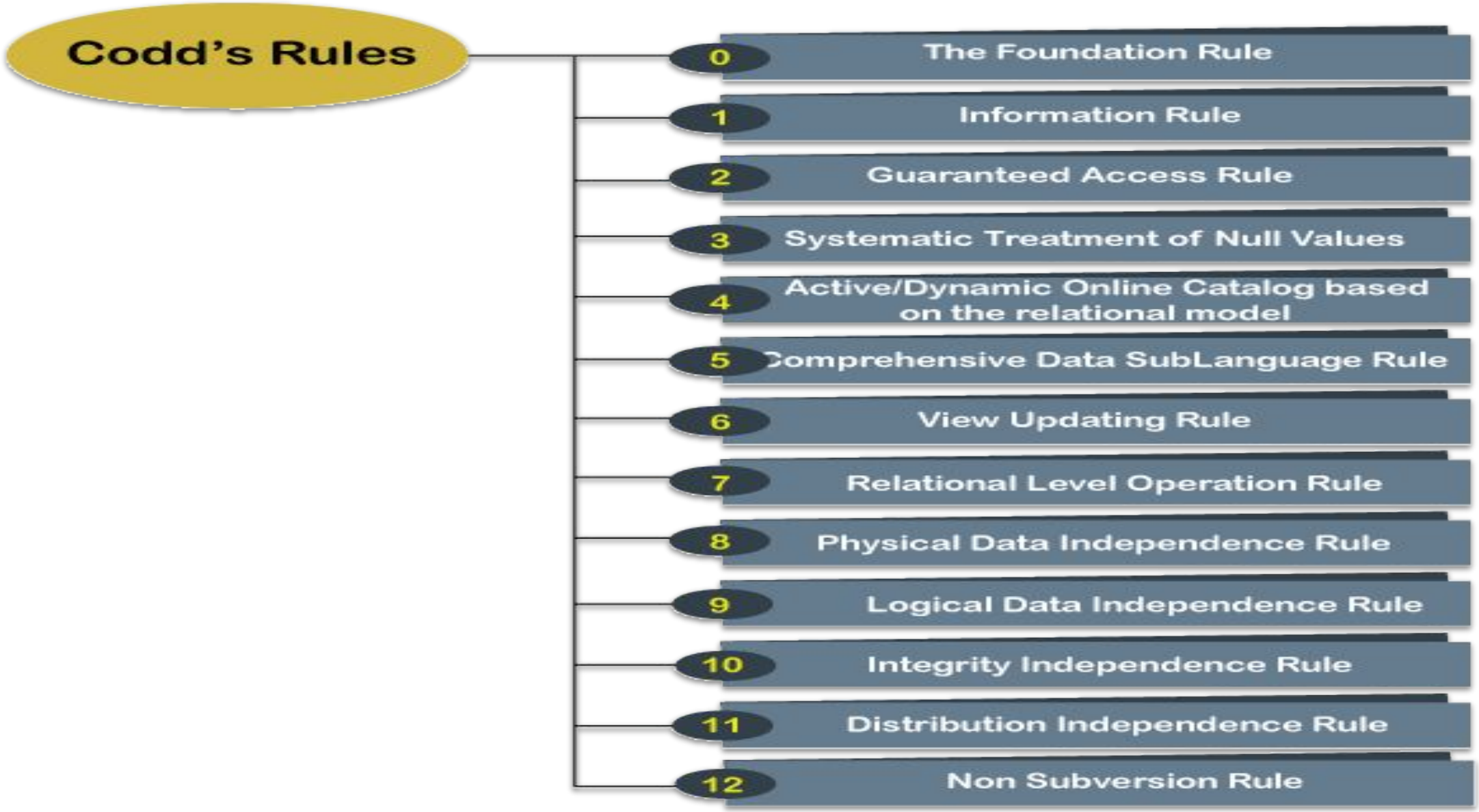
STUDENT MANAGEMENT SYSTEM




login_role_id
login_role_id
login_username
Login
user_password
user_name
#user_id
user_mobile
User
user_email
user_address
Has
#role_id
Roles
role_name
role_desc
Permission
#role_id
per_role_id
per-name
per_module
Manage
airlines booking
#p_id
p_name
p_password
p_address
passenger
p_mobile
p_email
Has
ab_pass_id
ab_desc
#ab_id
ab_date
ab_type
ticket
#tckt_id
tckt_desx
tckt_date
ae_title
airlines booking
#ae_id
ae_desc
ae_type
ae_date

Draw an E-R diagram of library management system considering issue and return, Fine calculation facility. Consider appropriate entities.




Codd's Rules
0
The Foundation Rule
1
Information Rule
2
Guaranteed Access Rule
3
Systematic Treatment of Null Values
4
Active/Dynamic Online Catalog based on the relational model
5
Comprehensive Data SubLanguage Rule
6
View Updating Rule
7
Relational Level Operation Rule
8
Physical Data Independence Rule
9
Logical Data Independence Rule
10
Integrity Independence Rule
11
Distribution Independence Rule
12
Non Subversion Rule

Rule 0: The Foundation Rule

The database must be in relational form. So that the system can handle the database through its relational capabilities.

Rule 1: Information Rule

A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns.

Rule 2: Guaranteed Access Rule

Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: Systematic Treatment of Null Values

This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null.

Rule 4: Active/Dynamic Online Catalog based on the relational model

It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Rule 5: Comprehensive Data SubLanguage Rule

The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database.

Rule 6: View Updating Rule

All views table can be theoretically updated and must be practically updated by the database systems.

Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule

A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Rule 8: Physical Data Independence Rule

All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application. If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database.

Rule 9: Logical Data Independence Rule

It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application.

Rule 10: Integrity Independence Rule

A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application.

Rule 11: Distribution Independence Rule

The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries.

Rule 12: Non Subversion Rule

The non-submersion rule defines RDBMS as a **SQL** language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.