

DATABASE MANAGEMENT SYSTEMS

Authors – Raghu Ramakrishna, Johannes Gehrke

Edition – 3

Unit -1

- History of Database Systems, Database System Applications, database System vs file System – View of Data – Data Abstraction –Instances and Schemas – data Models – the ER Model – Relational Model – Other Models
- Database Languages – DDL, DML –Transaction Management
- Database System Structure – Storage Manager – the Query Processor.
- Database design and E-R diagrams – Beyond E-R Design Entities, Attributes and Entity sets – Relationships and Relationship sets – Additional features of ER Model – Concept Design with the ER Model – Conceptual Design for Large enterprises.

Introduction to DBMS

- The database management systems consist of two parts:
 - Database and
 - Management System

What is Database?

- To find out what database is, we have to start from data, which is the basic building block of any DBMS.
- **Data:** Raw information, Facts, Figures, Statistics etc. having no particular meaning [e.g. 1, ABC, 10 etc]

- **Record:** collection of related data items. In the previous example the three data items had no meaning. But if we organize them in the following way, then they collectively represent meaningful information.

Roll	Name	Age
1	ABC	19

- **Table or Relation:** Collections of related Records.

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

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

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

Table 1

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

Table 2

Roll	Year
1	I
2	II
3	I

Table 3

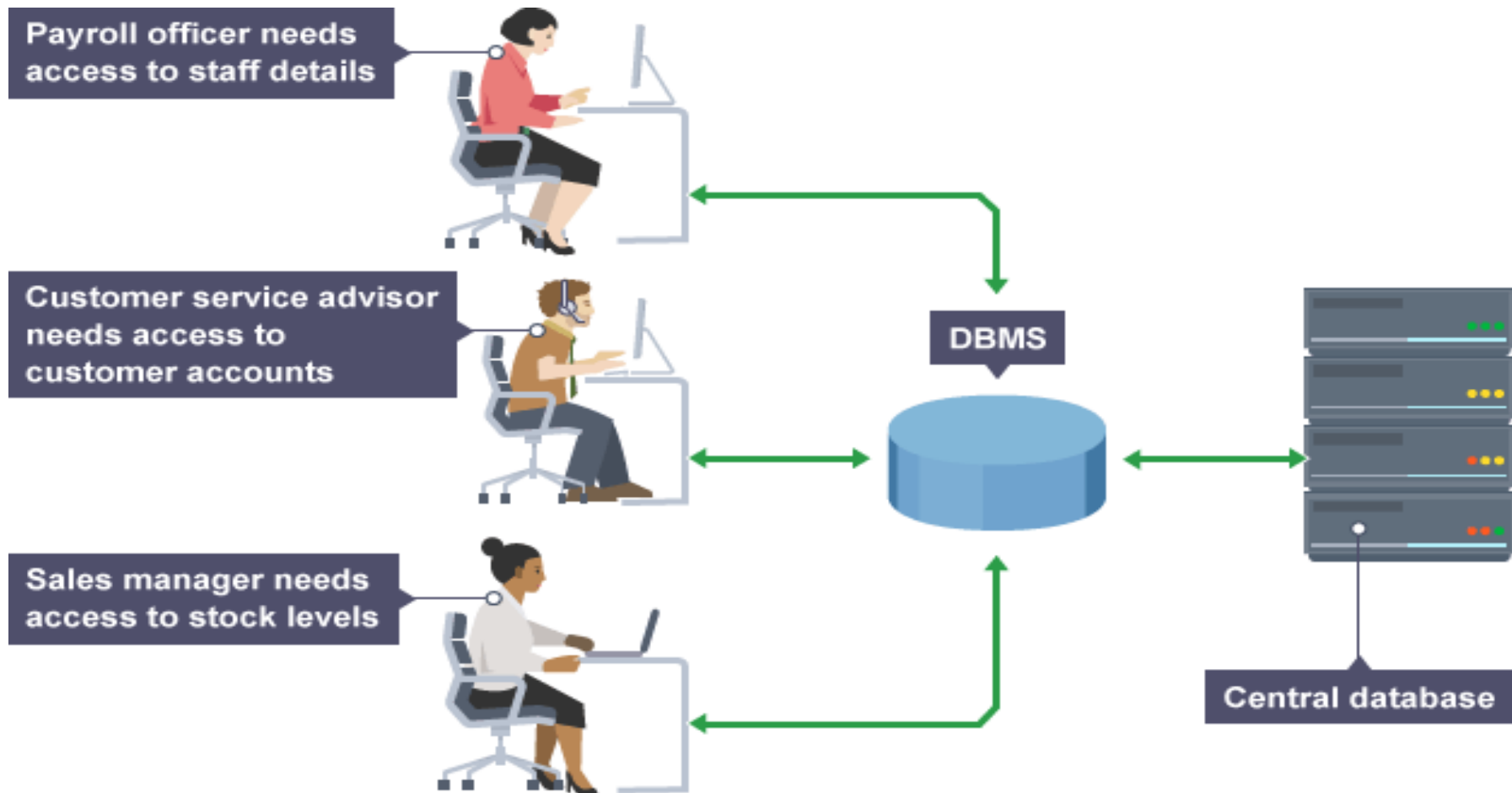
Roll	Address
1	KOL
2	DEL
3	MUM

Table 4

Year	Hostel
I	H1
II	H2

These are called **Related collection**. With this information we can easily get the data of each individuals. For example
Que: “Which hostel does the youngest student live in?”

- A database in a DBMS could be viewed by lots of different people with different responsibilities as well as customers, who each need to see different kinds of data. Each employee in the company will have different levels of access to the database with their own customized **front-end** application.



What is Management Systems?

- The primary goal of a DBMS is to provide a way to **store and retrieve** database information that is both convenient and efficient.
- The management system is important because without the existence of some kind of rules and regulations it is not possible to maintain the database.
- We have to select the particular attributes which should be included in a particular table; the common attributes to create relationship between two tables; if a new record has to be inserted or deleted then which tables should have to be handled etc.
- These issues must be resolved by having some kind of rules to follow in order to maintain the integrity of the database.

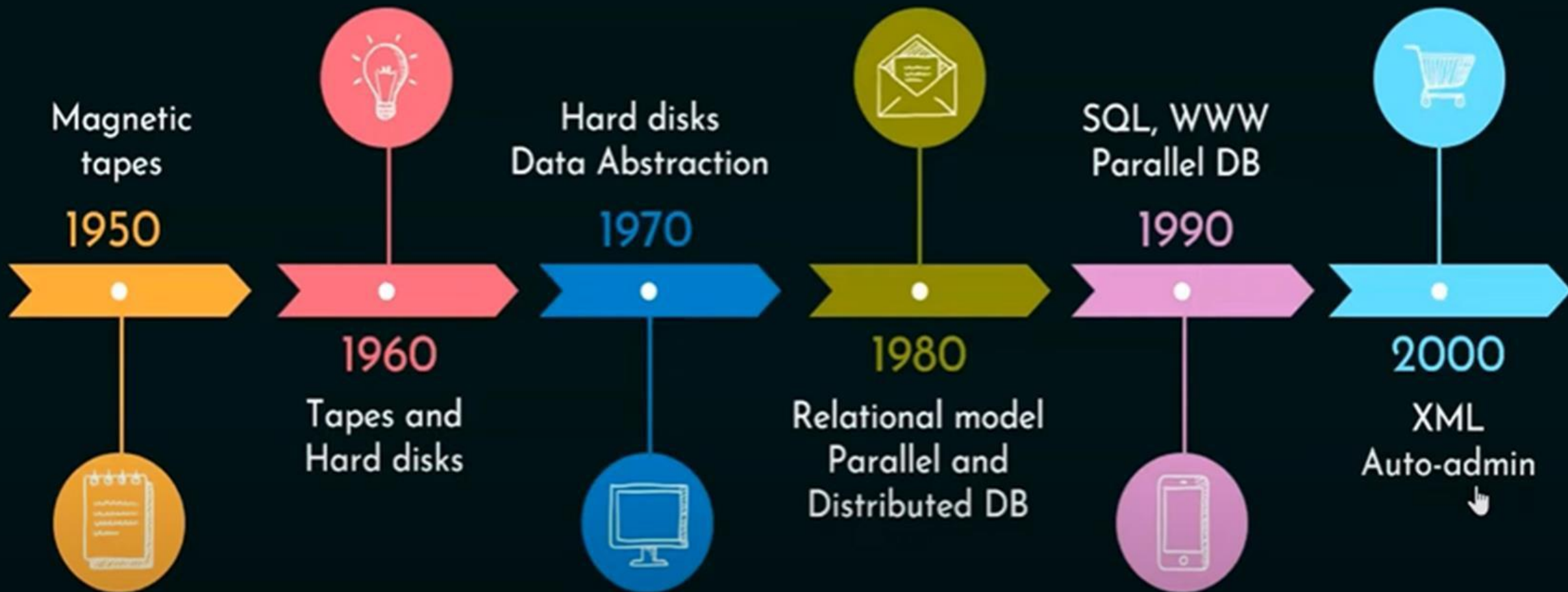
History of Database Systems

- In the early days, computer systems were designed to manage data in a **hierarchical or navigational manner**, where data was stored in a tree-like structure.
- This method of storing data was inefficient and difficult to use, as it required a lot of manual effort to access and manage the data.
- In the late 1960s, the first general-purpose DBMS, designed by **Charles Bachman**, was called the **Integrated Data Store (IDS)** which was based on **network data model** for which he was received the **Turing Award** (The most prestigious award which is equivalent to Nobel prize in the field of Computer Science.).

- In the late 1970s, *Mr. Edgar Codd* proposed a new data representation framework called the *Relational Database Model*. He won the 1981 Turing Award for his seminal work. This model was based on the concept of a table, with rows representing individual records and columns representing individual fields within those records. The relational model allowed for more efficient storage and retrieval of data and was easier to use than the hierarchical or navigational models.
- In the late 1980s IBM developed the *Structured Query Language (SQL)* for relational databases. This system was designed to manage large amounts of data and was used primarily in corporate and government applications. SQL was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).

- In the 1990s, **object-oriented DBMS (OODBMS)** emerged, which were designed to store and manage complex data structures, such as multimedia and other types of non-traditional data. These systems were initially popular in research and academic environments, but their adoption was limited in the commercial sector.
- In the 1997, **XML** applied to database processing. Many vendors begin to integrate XML into DBMS products.
- In the 2000s, web-based applications and cloud computing became more popular, and DBMS systems began to adapt to these new technologies. New DBMS systems were developed to support distributed and web-based applications, including NoSQL databases such as MongoDB and Cassandra.
- Today, DBMS systems continue to evolve, with an emphasis on scalability, performance, and support for cloud-based applications. Some of the most popular DBMS systems in use today include Oracle, Microsoft SQL Server, MySQL, PostgreSQL, and MongoDB.

History of DBMS



Database System Applications

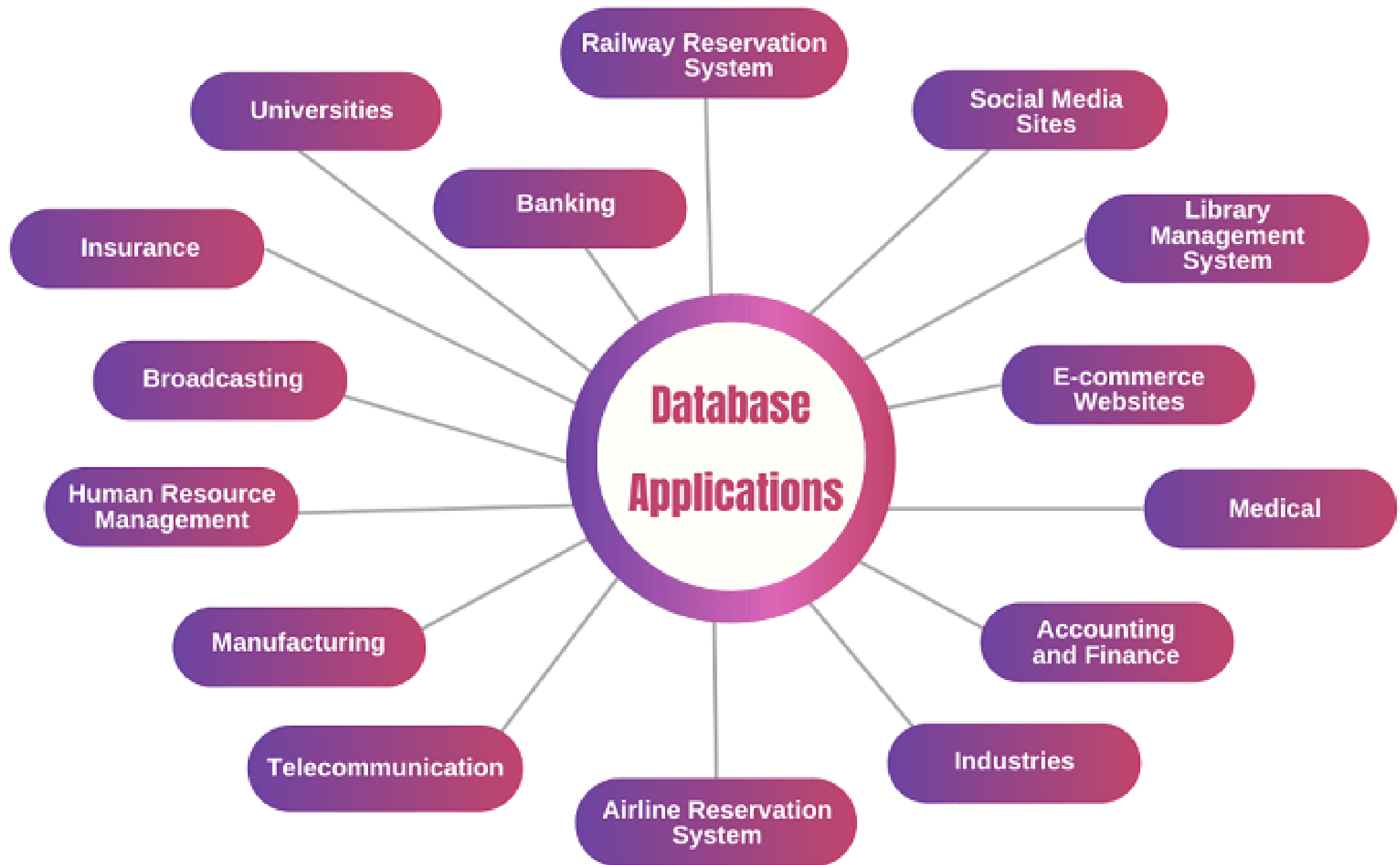
Enterprise Information:

- Sales: For customer, product, and purchase information.
- Accounting: For payments, receipts, account balances, assets and other accounting information.
- Human resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
- Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

Database System Applications

Banking and Finance:

- Banking: For customer information, accounts, loans, and banking transactions.
- Credit card transactions: For purchases on credit cards and generation of monthly statements.
- Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
- Universities: For student information, course registrations, and grades.
- Airlines: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- Telecommunication: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.



Purpose of Database Systems

The purpose of database systems is to manage the following insecurities:

- Data redundancy and Inconsistency,
- Difficulty in accessing data,
- Data Isolation,
- Atomicity of updates,
- Concurrent access,
- Security problems, and
- Supports multiple views of data.

- **Avoid data redundancy and inconsistency:** If there are multiple copies of the same data, it just avoids it. It just maintains data in a single repository.
- **Difficulty in accessing data:** A database system can easily manage to access data. Through different queries, it can access data from the database.
- **Data Isolation:** Data are isolated in several fields in the same database.
- **Atomicity of updates:** In case of power failure, the database might lose data. So, this feature will automatically prevent data loss.
- **Concurrent access:** Users can have multiple access to the database at the same time.
- **Security problems:** Database systems will make the restricted access. So, the data will not be vulnerable.
- **Supports multiple views of data:** It can support multiple views of data to give the required view as their needs. Only database admins can have a complete view of the database. We cannot allow the end-users to have a view of developers.

Database System Vs File System

Database System	File System
DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	File system is a collection of data. In this system, the user has to write the procedures for managing the database.
DBMS gives an abstract view of data that hides the details	File system provides the detail of the data representation and storage of data
DBMS provides a crash recovery mechanism i.e DBMS protects the user from the system failures	File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost.
A database management system is designed to coordinate multiple users accessing the same data at the same time.	A file-processing system is usually designed to allow one or more programs to access different data files at the same time.
DBMS contains a wide variety of sophisticated techniques to store and retrieve the data.	File system can't efficiently store and retrieve the data.
DBMS takes care of Concurrent access of data using some form of locking	In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information
Redundancy is control in DBMS	Redundancy not control in file system
Unauthorized access is restricted and also provides backup and recovery , security.	Not in the file system
A dbms is designed to allow flexible access to data	A file-processing system is designed to allow predetermined access to data (i.e. compiled programs).

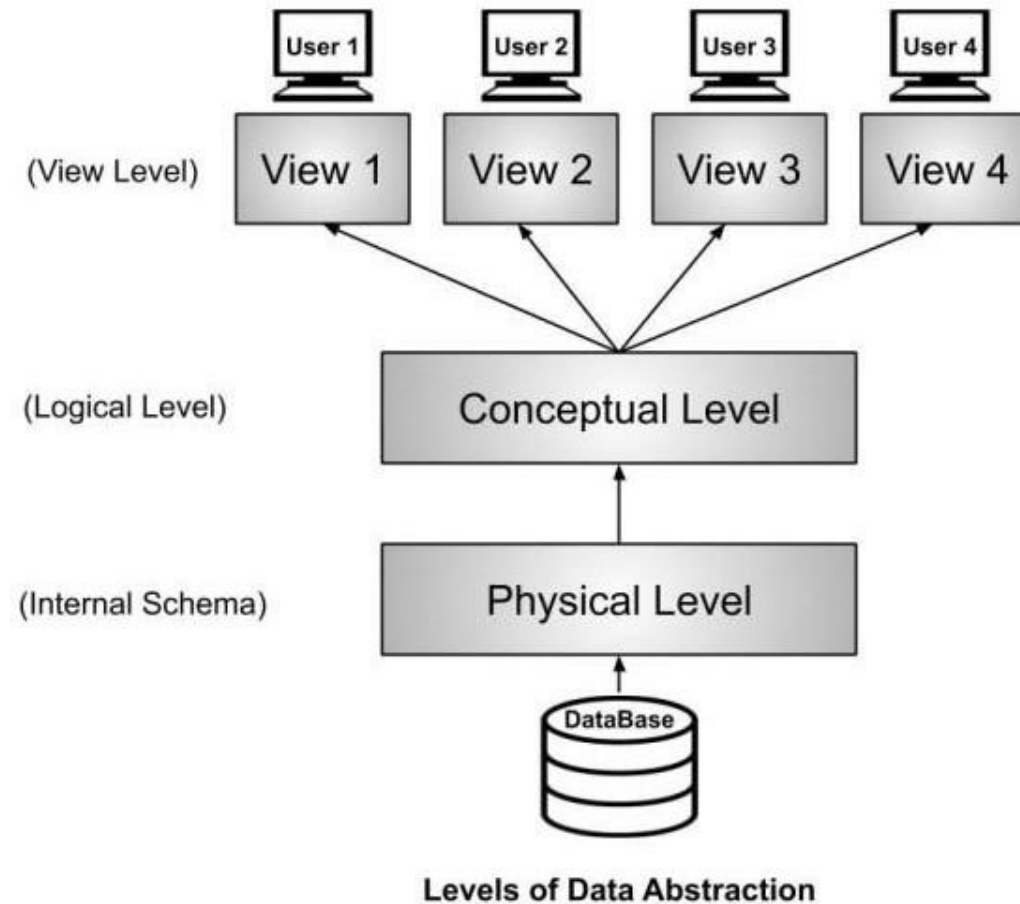
Advantages of a DBMS

1. **Controlling Redundancy:** Data redundancy refers to the duplication of data (i.e storing same data multiple times)
2. **Improved data sharing:** allows a user to share the data in any number of application programs
3. **Data integrity:** Integrity means that the data in the database is accurate. For example: in customer database we can enforce an integrity that it must accept the customer only from Noida and Meerut city
4. **Security:** The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.
5. **Data consistency:** defined by a set of rules that ensure that all data points in the database system are correctly read and accepted. This is achieved by making rules.
6. **Data independence:** DBMS provides interface between the application program and data. When one module is changed then it will not effect to the other modules.

View of Data

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

Data Abstraction:



Levels of Abstraction:

1. Physical Level (Internal Schema):

- The lowest level of abstraction describes *how* the data are actually stored.
- It defines complex data structures in detail, so it is very complex to understand, which is why it is kept hidden from the end user.
- Data Administrators (DBA) decide how to arrange data and where to store data. The Data Administrator (DBA) is the person whose role is to manage the data in the database at the physical or internal level.
- There is a data center that securely stores the raw data in detail on hard drives at this level

- Logical level (Conceptual schema):
 - This level is the intermediate level of data abstraction and 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. The logical level or conceptual level is less complex than the physical level.
 - With the help of the logical level, Data Administrators (DBA) abstract data from raw data present at the physical level.
- View Level (External Schema):
 - View or External Level is the highest level of data abstraction. There are different views at this level that define the parts of the overall data of the database.
 - This level is for the end-user interaction; at this level, end users can access the data based on their queries.

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

A university organization may have several such record types, including

- *department*, with fields *dept_name*, *building*, and *budget*
- *course*, with fields *course_id*, *title*, *dept_name*, and *credits*
- *student*, with fields *ID*, *name*, *dept_name*, and *tot_cred*

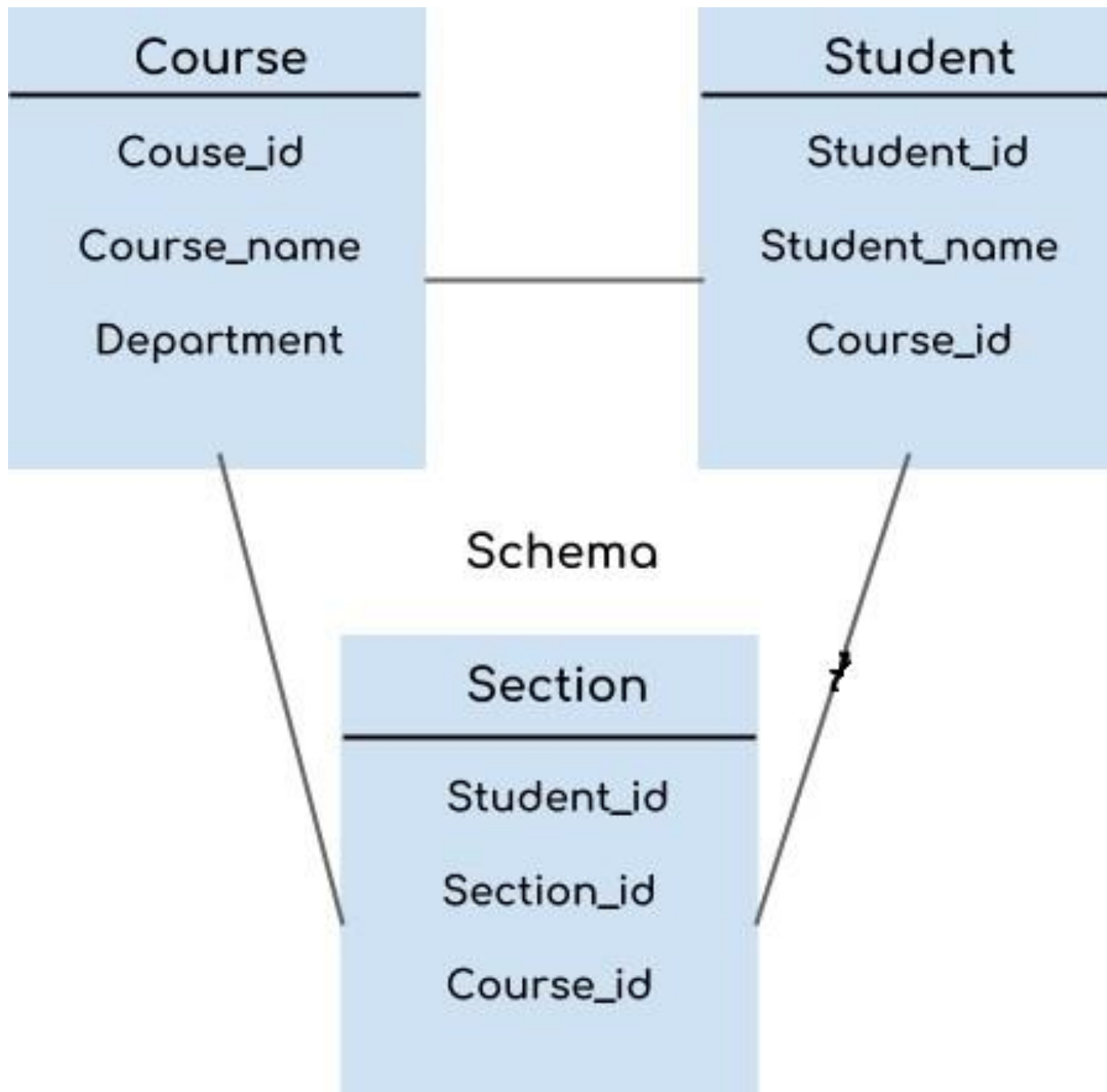
- At the physical level, an instructor, department, or student record can be described as a block of consecutive storage locations. Database administrators may be aware of physical organization of the data.
- At the logical level, each such record is described by a type definition. Programmers using a programming language work at this level of abstraction
- Finally, At the view level, several views of the database are defined, and a database user sees some or all of these views.

DBMS SCHEMA

- Design of a database is called the Schema
- Schema is a Logical representation and it defines the attributes of tables in the database.
- Schema is only a structural view of a database and it doesn't show the data present in those tables.
- For example: An Employee table in database exists with the following attributes:

Employee (Emp_ID, Emp_Name, Emp_address, Emp_contact).

Course(Course_id, Course_name, department)



DBMS Instance

- **Definition of instance:** The data stored in database at a particular moment of time is called instance of database. Database schema defines the attributes in tables that belong to a particular database. The value of these attributes at a moment of time is called the instance of that database. For example:

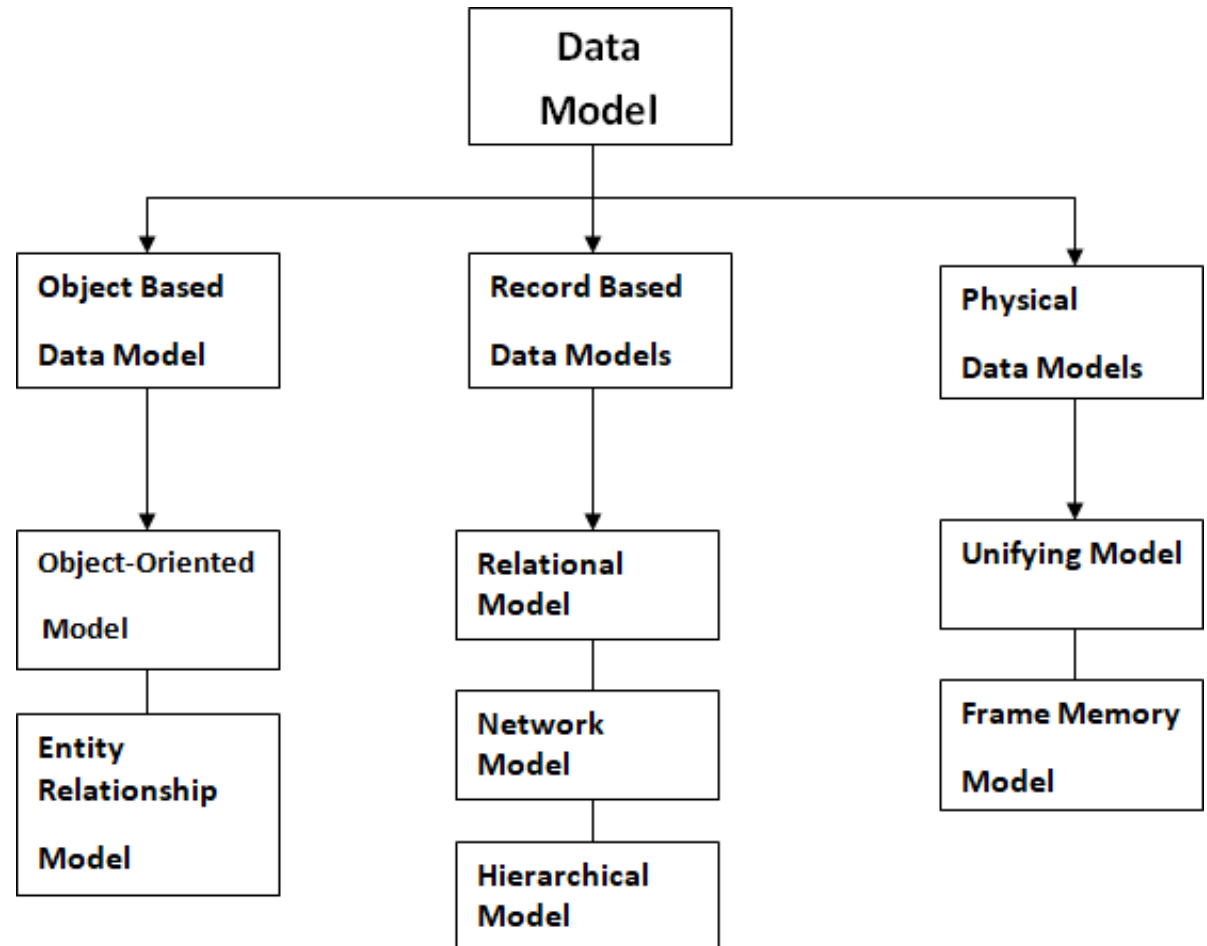
Emp_ID	Emp_Name	Emp_address	Emp_contact
101	Xyz	Noida	1234567890
102	Abc	Delhi	9080123431

Data Models

Data Model is a logical structure of Database. It describes the design of database to reflect entities, attributes, relationship among data, constrains etc.

Types of Data Models

- Object based logical models
 - E-R Model
 - Object oriented model
- Physical Data Models
 - Frame memory models
 - Unifying model
- Record based logical models
 - Relational Model
 - Hierarchical model
 - Network model

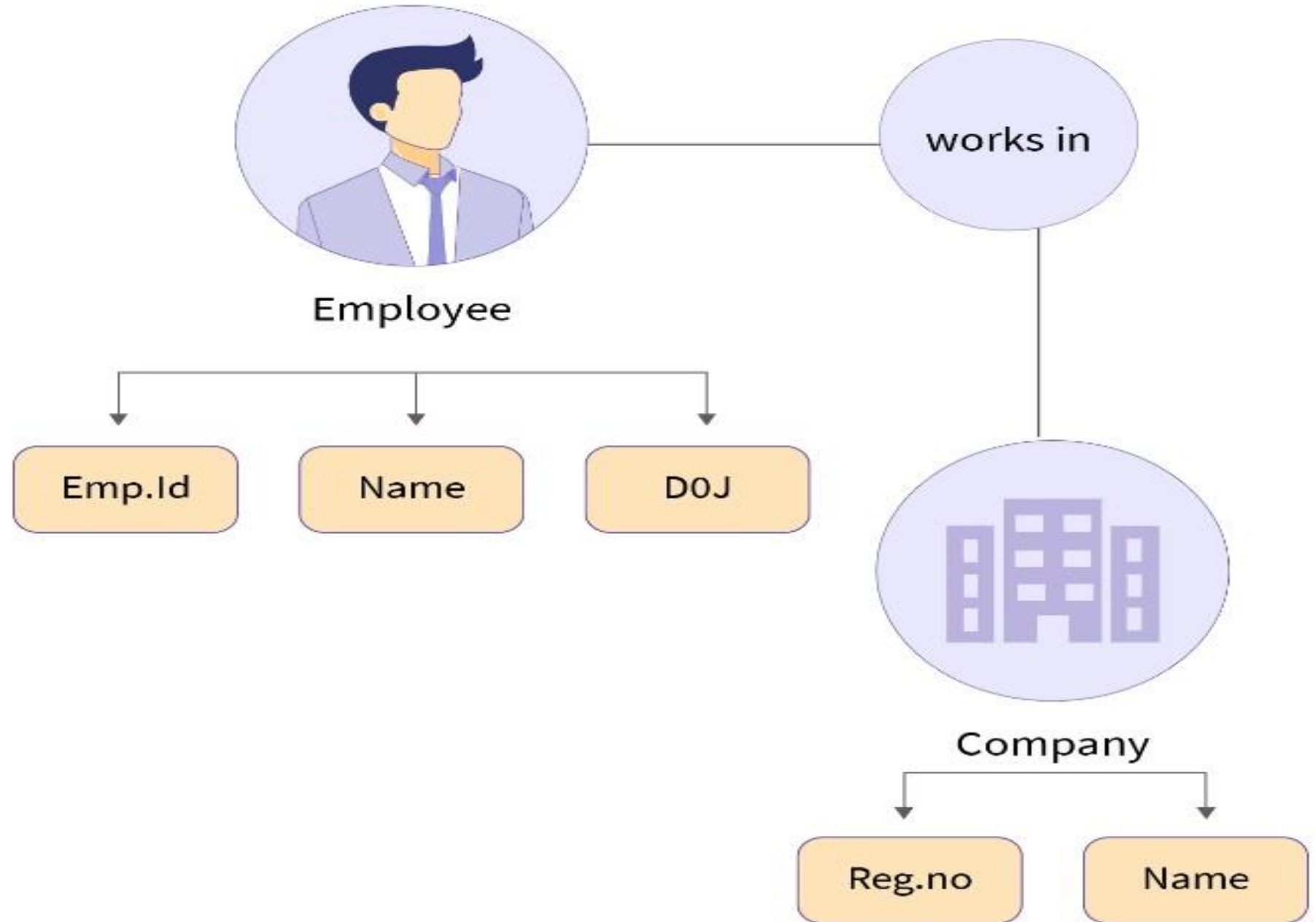


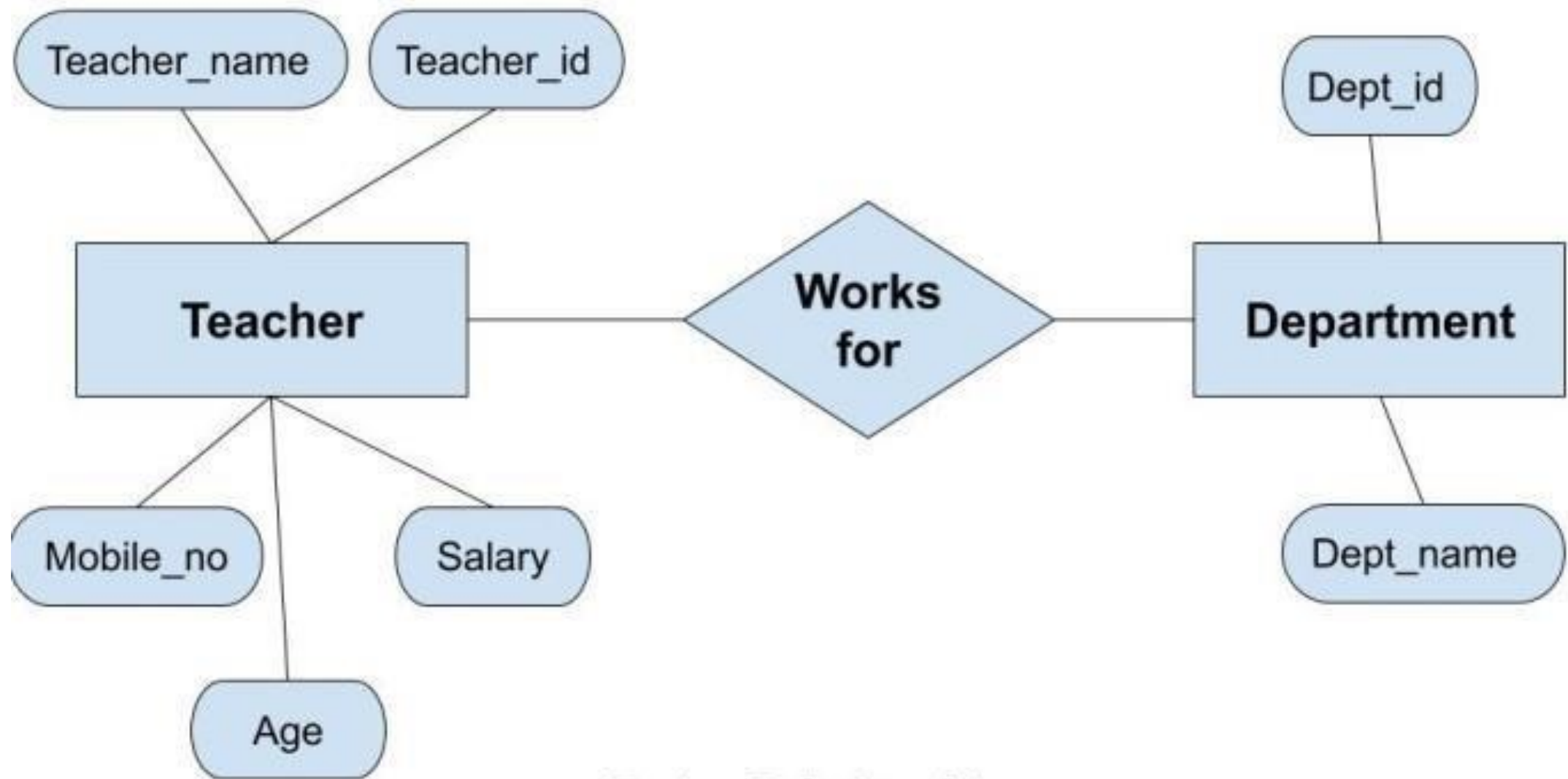
1. Object based logical models – Types of Data Model

1. Entity Relationship Model (ER Model):

1. ER diagram is used to represent logical structure of the database easily.
2. ER model develops a conceptual view of the data hence it can be used as a blueprint to implement the database in the future.
3. **Entity** - Entity is an object and *represented in rectangles in the ER diagram*. For example - Car, house, employee.
4. **Entity Set** - A set of the same type of entities is known as an entity set. For example - Set of students studying in a college.
5. **Attributes** - Properties that define entities are called attributes. *They are represented by an ellipse shape.*
6. **Relationships** - used to describe the association between entities. *They are represented as diamond or rhombus shapes in the ER diagram.*

EXAMPLE:

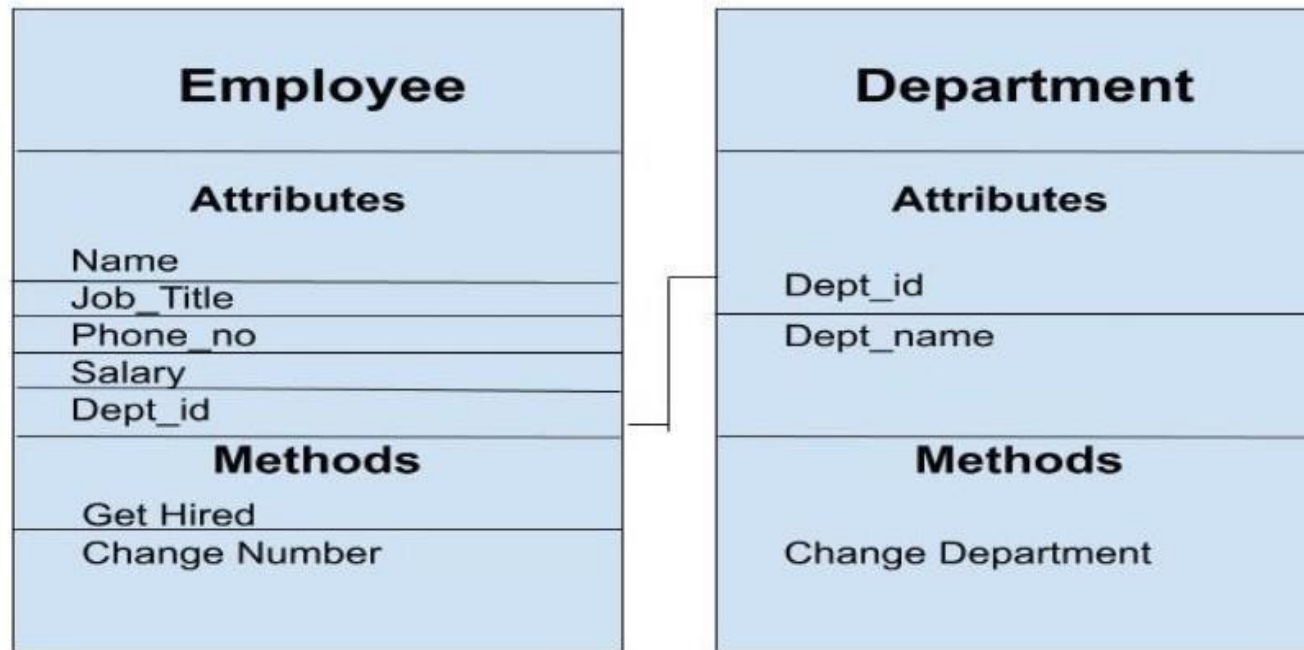




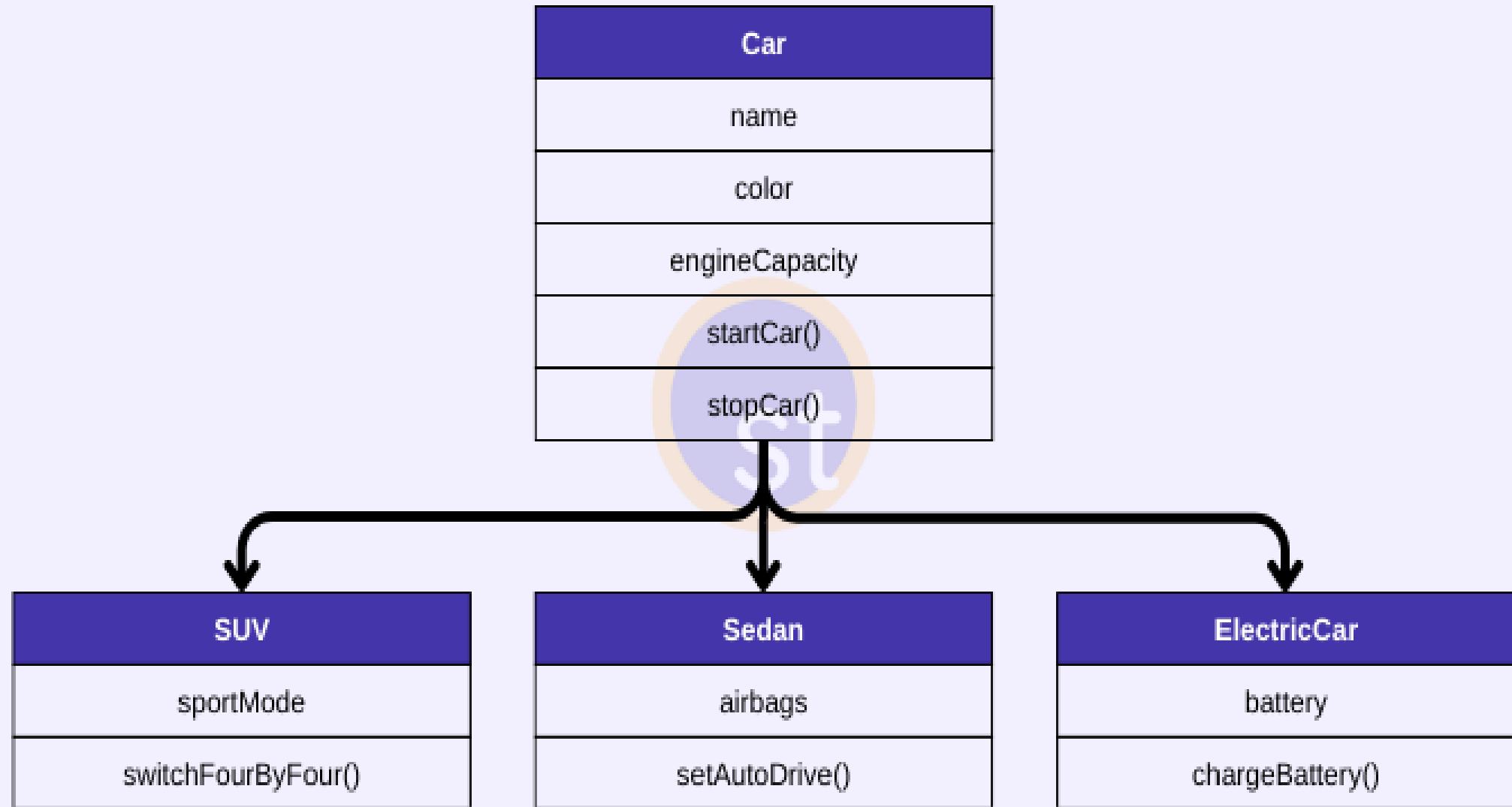
**Entity-Relationship
Model**

2. Object Oriented Model:

- In this model, data is stored in the form of objects.
- The behavior of the object-oriented database model is just like object-oriented programming.
- A very popular example of an Object Database management system or **ODBMS** is **MongoDB** which is also a NoSQL database.



Object_Oriented_Model



2. Record Based Data Model - Types of Data Model

- Like Object based model, they also describe data at the conceptual and view levels. These models specify logical structure of database with records, fields and attributes.

1. Relational Model:

1. This is the **most widely accepted data model**.
2. In this model, the database is represented as a **collection of relations in the form of rows and columns of a two-dimensional table**.
3. Each row is known as a tuple (a tuple contains all the data for an individual record) while each column represents an attribute.

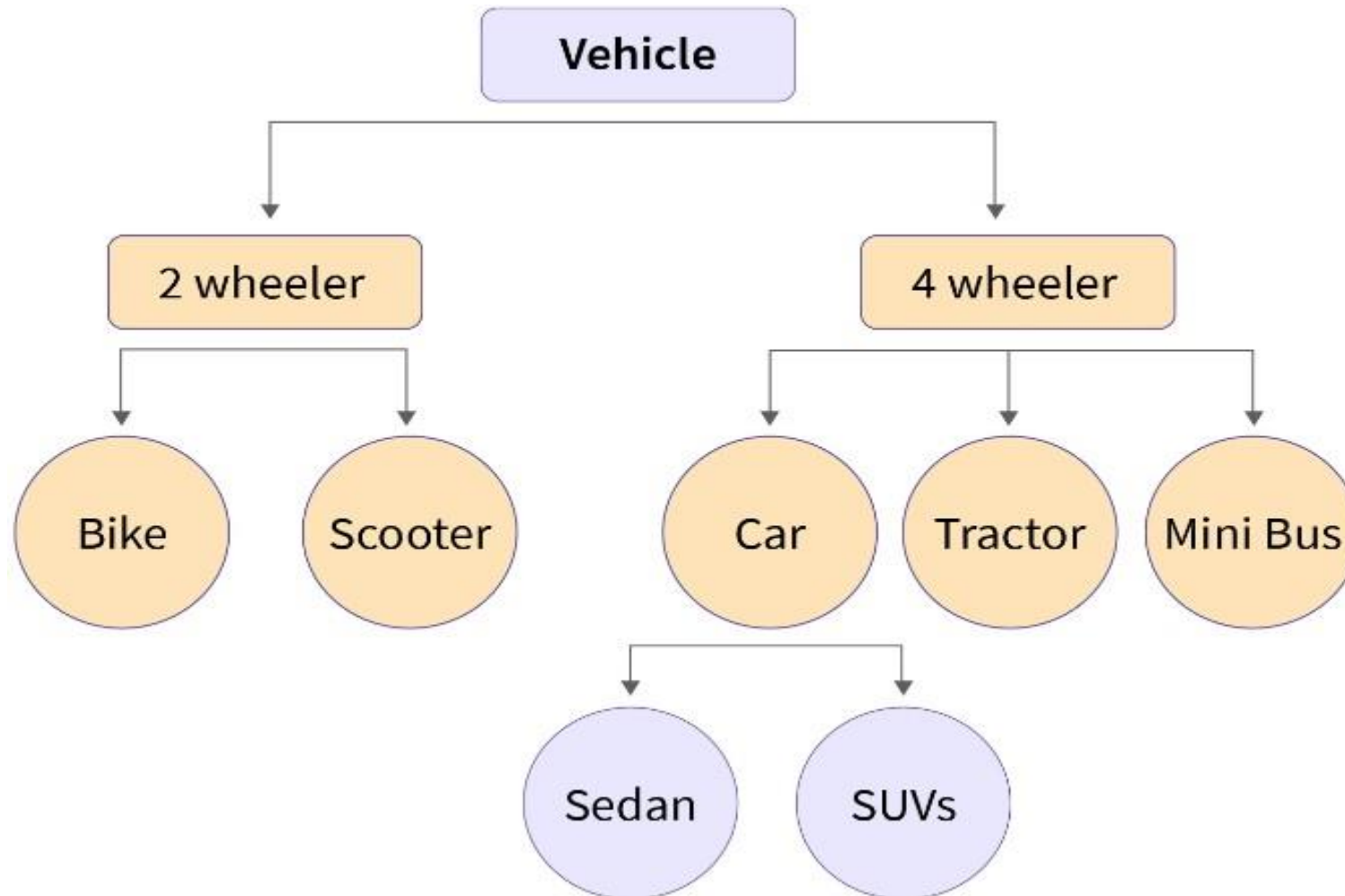
Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Stu. Id	Name	Branch
101	Naman	CSE
102	Saloni	ECE
103	Rishabh	IT
104	Pulkit	ME

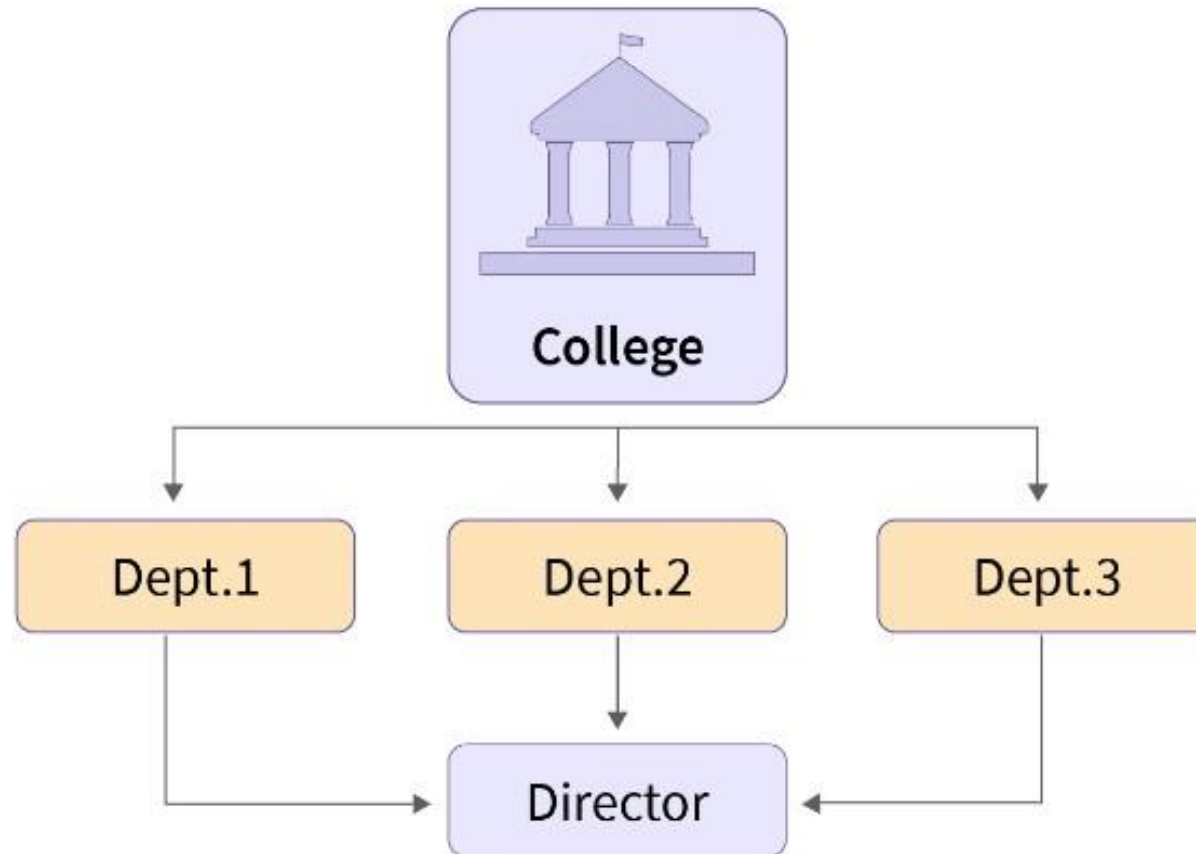
2. Hierarchical Model:

- In this data model, the data is organized in a hierarchical tree-like structure.



3. Network Model:

- Network Model is same as hierarchical model except that it has graph-like structure rather than a tree-based structure. Unlike hierarchical model, this model allows each record to have more than one parent record



3. Physical Data Models - Types of Data Model

1. Frame Memory Model:

- Frame memory is a virtual view of secondary storage that can be implemented with reasonable overhead to support database record storage and accessing requirements

2. Unifying Model:

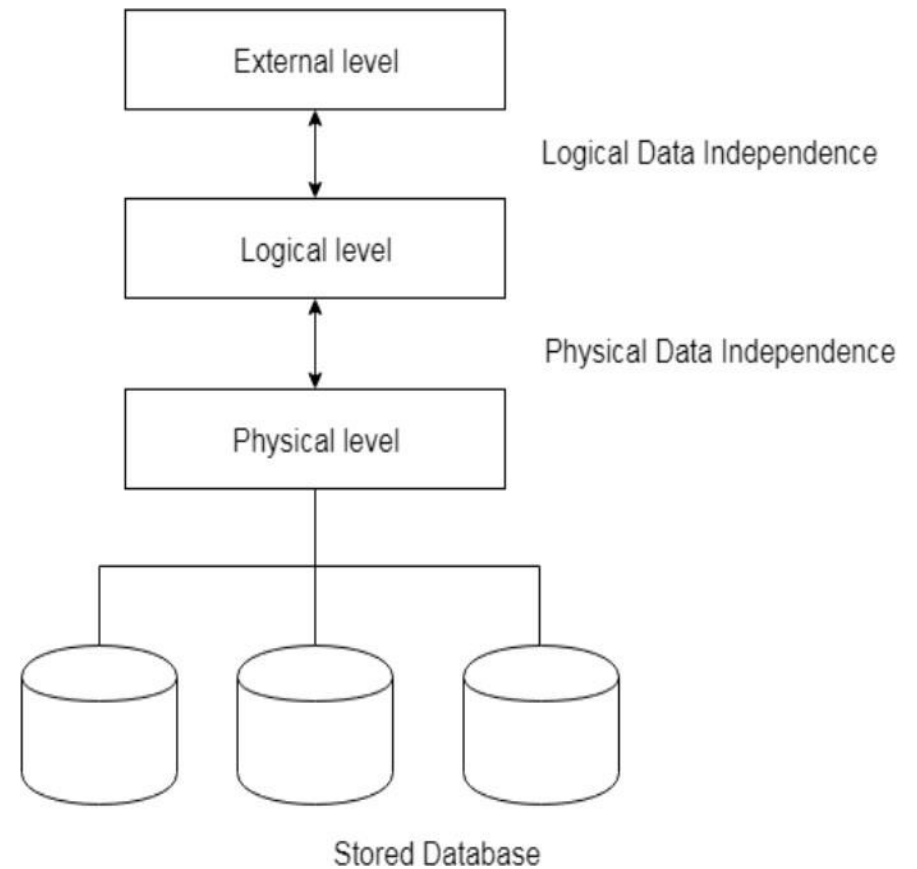
- A unified data model brings together data from different sources and platforms in one place so that all your data is considered when conducting analyses and making decisions.
- Unified data is important because it enables you to look at every collected data point so you can understand the entire data narrative.

Data Independence

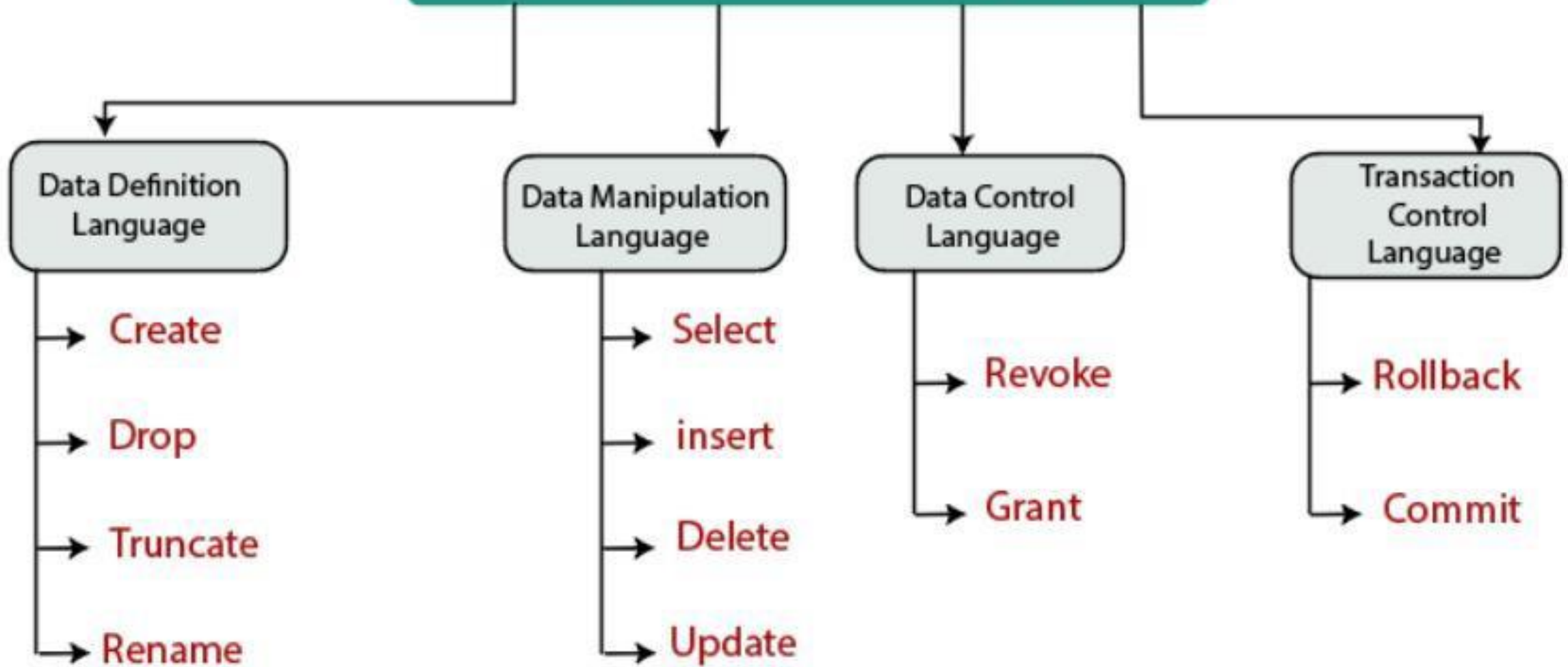
- Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

Types of Data Independence:

1. **Physical Data Independence:** This is defined as the ability to modify the physical schema of the database without the modification causing any changes in the logical/conceptual or view/external level.
2. **Logical Data Independence:** Logical data independence is the ability to modify logical schema without causing any unwanted modifications to the external schema or the application programs to be rewritten.



Types of DBMS Language



1. Data Definition Language:

- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
- Here are some tasks that come under DDL:
 - **Create:** It is used to create objects in the database.
 - **Alter:** It is used to alter the structure of the database.
 - **Drop:** It is used to delete objects from the database.
 - **Truncate:** It is used to remove all records from a table.
 - **Rename:** It is used to rename an object.
 - **Comment:** It is used to comment on the data dictionary.

2. Data Manipulation Language:

- **DML** stands for **Data Manipulation Language**. It is used for accessing and manipulating data in a database. It handles user requests.
- Here are some tasks that come under DML:
 - **Select:** It is used to retrieve data from a database
 - **Insert:** It is used to insert data into a table
 - **Update:** It is used to update existing data within a table
 - **Delete:** It is used to delete all records from a table.
 - **Merge:** It performs UPSERT operation, i.e., insert or update operations
 - **Call:** It is used to call a structured query language or a Java subprogram
 - **Explain Plan:** It has the parameter of explaining data
 - **Lock Table:** It controls concurrency.

3. Data Control Language:

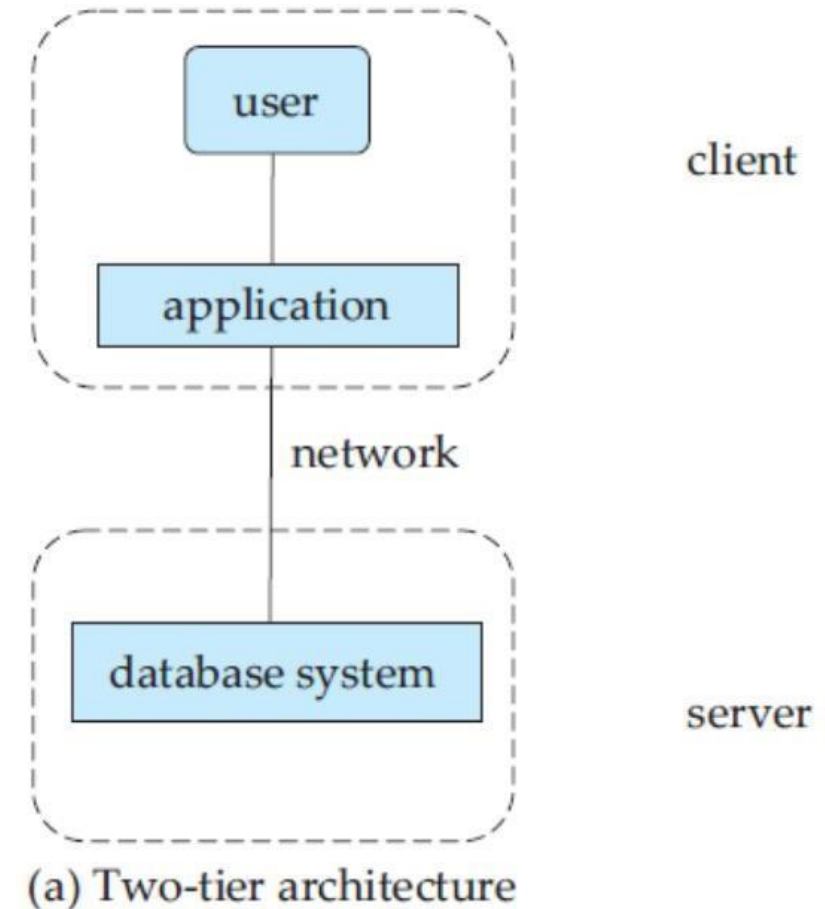
- **DCL** stands for **Data Control Language**. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.
- Here are some tasks that come under DCL:
 - **Grant:** It is used to give user access privileges to a database
 - **Revoke:** It is used to take back permissions from the user

4. Transaction Control Language:

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.
- Here are some tasks that come under TCL:
 - **Commit:** It is used to save the transaction on the database
 - **Rollback:** It is used to restore the database to original since the last Commit.

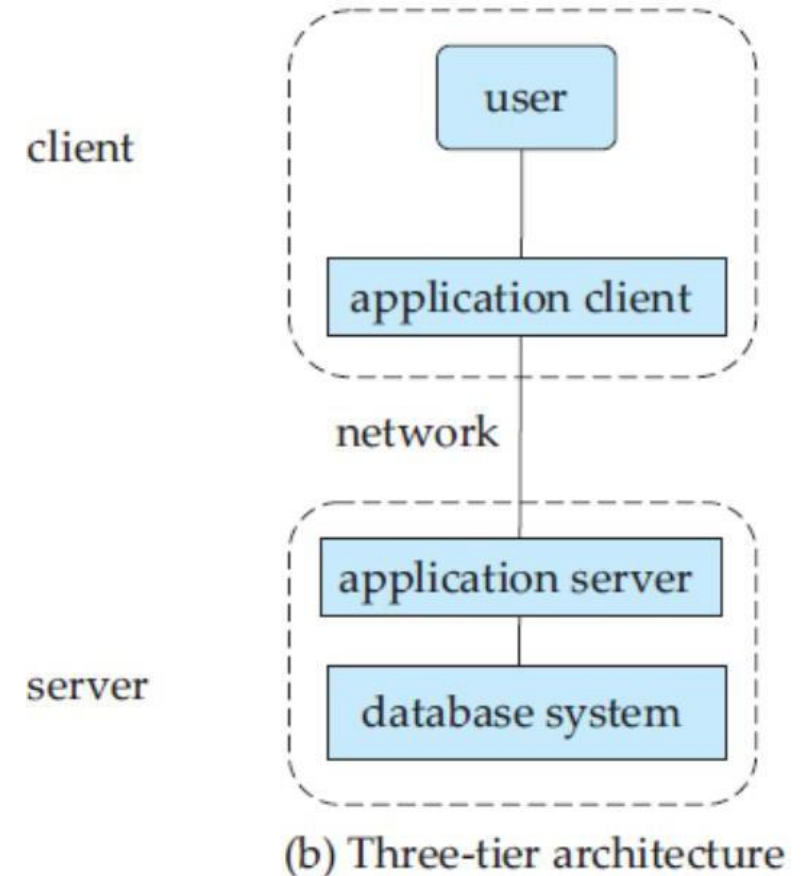
Two – tier architectures

- A two-tier architecture, also known as a client-server architecture, consists of two main layers: a client layer and a server layer.
- The client layer is responsible for interacting with the user and presenting data to them, while the server layer is responsible for processing requests and managing the application's data and business logic



Three – tier architectures

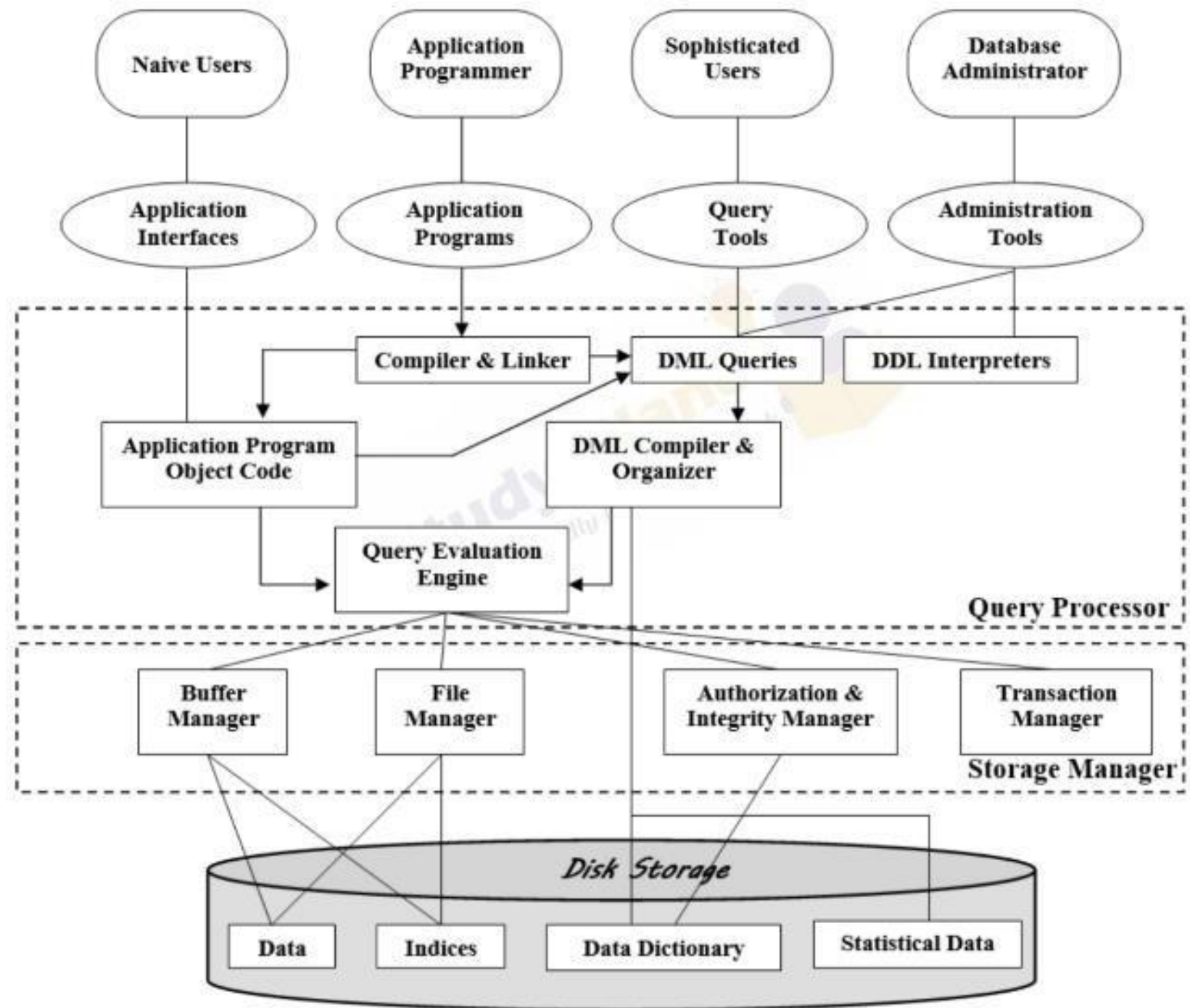
- A three-tier architecture, also known as a multi-tier architecture, consists of three main layers: a presentation layer, an application layer, and a data layer.
- The presentation layer is responsible for interacting with the user and presenting data to them, the application layer is responsible for processing requests and managing the application's business logic, and the data layer is responsible for managing the application's data.



Database System Structure

- The typical structure of DBMS is based on Relational data model.
- The top part of the architecture shows. application interfaces used by naive users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator.
- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the storage manager and the query processor components.
- The storage manager is important because databases typically require a large amount of storage space.
- The query processor is important because it helps the database system simplify and facilitate access to data.

Database System Structure



Query Processor: The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL interpreter, DML compiler and query evaluation engine.

The following are various functionalities and components of query process:

- **DDL interpreter:** This is basically a translator which interprets the DDL statements in data dictionaries.
- **DML compiler:** It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.
- **Query evaluation engine:** It executes the low-level instructions generated by the DML compiler.

Storage manager:

- The storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include
- **Authorization and integrity manager:** Validates the users who want to access the data and tests for integrity constraints.
- **Transaction manager:** Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
- **File manager:** Manages allocation of space on disk storage and representation of the information on disk.
- **Buffer manager:** Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.

Disk Storage: It contains the following components –

- **Data Files:** It stores the data.
- **Data Dictionary:** It contains the information about the structure of any database object. It is the repository of information that governs the metadata.
- **Indices:** It provides faster retrieval of data item.

Transaction Management

- A **transaction** is a set of logically related operations. For example, you are transferring money from your bank account to your friend's account, the set of operations would be like this:
- **Simple Transaction Example**
- 1. Read your account balance
 2. Deduct the amount from your balance
 3. Write the remaining balance to your account
 4. Read your friend's account balance
 5. Add the amount to his account balance
 6. Write the new updated balance to his account

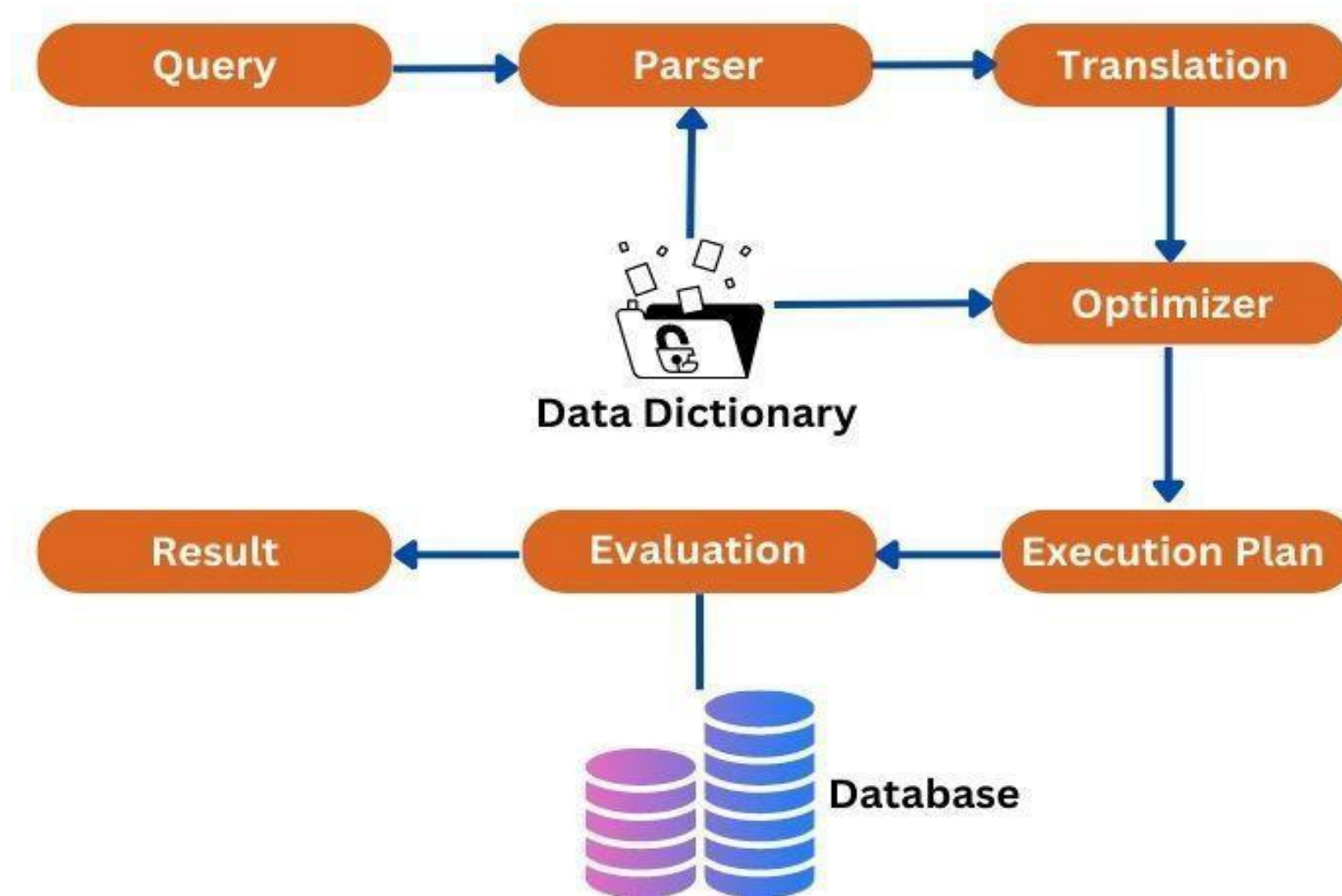
In DBMS we will
write like this

```
1. R(A);  
2. A = A - 10000;  
3. W(A);  
4. R(B);  
5. B = B + 10000;  
6. W(B);
```

Query Processing in DBMS

- **Query: A query is a kind of request** that is sent to the data base for retrieval data.
- **Query Processing:**
 - Query processing is a process of translating a user query into an executable form. It helps to retrieve the results from a database.
 - In query processing, it converts the high-level query into a low-level query for the database.
 - The steps involved are:
 - Parsing and Translation
 - Optimization
 - Evaluation

Steps in Query Processing



1. Parsing and Translation:

- The first step is **Parsing and Translation**. The fired queries undergo lexical, syntactic, and semantic analysis.
- Essentially, the query gets broken down into different tokens and white spaces are removed along with the comments (**Lexical Analysis**).
- In the next step, the query gets checked for the correctness, both syntax and semantic wise. The query processor first checks the query if the rules of SQL have been correctly followed or not (**Syntactic Analysis**).
- Finally, the query processor checks if the meaning of the query is right or not. (**Semantic Analysis**).

Example: **select emp_name from Employee where salary>10000;**

The above query would be divided into the following tokens:

SELECT, emp_name, FROM, employee, WHERE, salary, >, 10000.

○ $\sigma_{\text{salary} > 10000} (\pi_{\text{salary}} (\text{Employee}))$

Relational Algebra

○ $\pi_{\text{salary}} (\sigma_{\text{salary} > 10000} (\text{Employee}))$

2. Optimization:

1. After doing query parsing, the DBMS starts finding the **most efficient way to execute the given query**.
2. The optimization process follows some factors for the query. These factors are indexing, joins, and other optimization mechanisms. So, query optimization tells the DBMS what the best execution plan is for it. The main goal of this step is to retrieve the required data with minimal cost in terms of resources and time.

3. Evaluation:

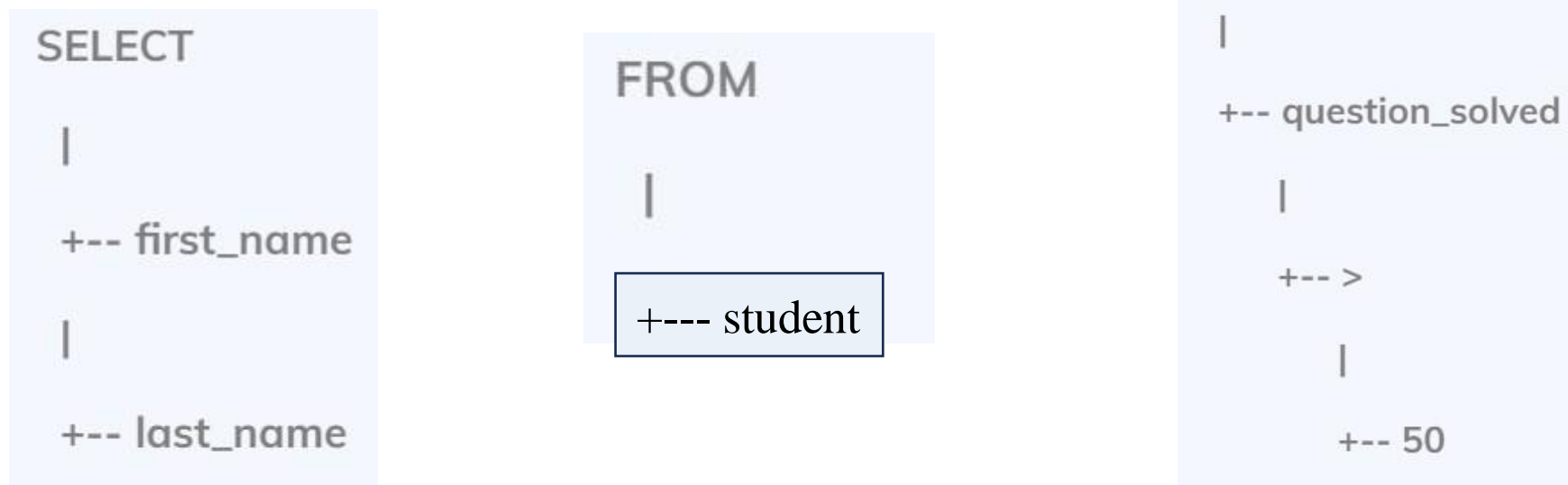
- After finding the best execution plan, the DBMS starts the execution of the optimized query. And it gives the results from the database.
- In this step, DBMS can perform **operations on the data**. These operations are selecting the data, inserting something, updating the data, and so on.

Example:

Let's consider an example to show you the query processing steps. Suppose we have a database with a table “student”. It contains the `stud_id`, `first_name`, `last_name`, and `ques_solved`. The following SQL query is used to retrieve the names of all students whose `ques_solved` is greater than 50:

```
select first_name, last_name from student where ques_solved >50;
```

Parsing: Firstly, the query will be parsed. This will also check whether the syntax is correct or not. Then this query will be converted into a parse tree. This tree will look like this:



- **Optimization:**

- The DBMS determines the most efficient way to execute the query by considering factors such as whether an index exists on the ques_solved field. In this case, the DBMS might use an index on the ques_solved field to efficiently retrieve the matching rows.

- **Evaluation:**

- The DBMS executes the optimized query. It retrieves the results from the database. Then it returns the first_name and last_name of all ninjas whose ques_solved is greater than 50.

Database Design - Entity Relationship(ER) Modelling

Database design process can be divided into 6 steps. The ER model is most relevant to the first three steps:

- **Requirement Analysis:** involves understanding user needs, analyzing existing systems, and identifying frequent operations for performance requirements.
- **Conceptual database design:** employed to create a broad overview of the data and associated constraints for the database. Typically, the (ER) model is used in this step.
- **Logical database design:** In this step we convert Conceptual database design (ER) into a relational DBMS.

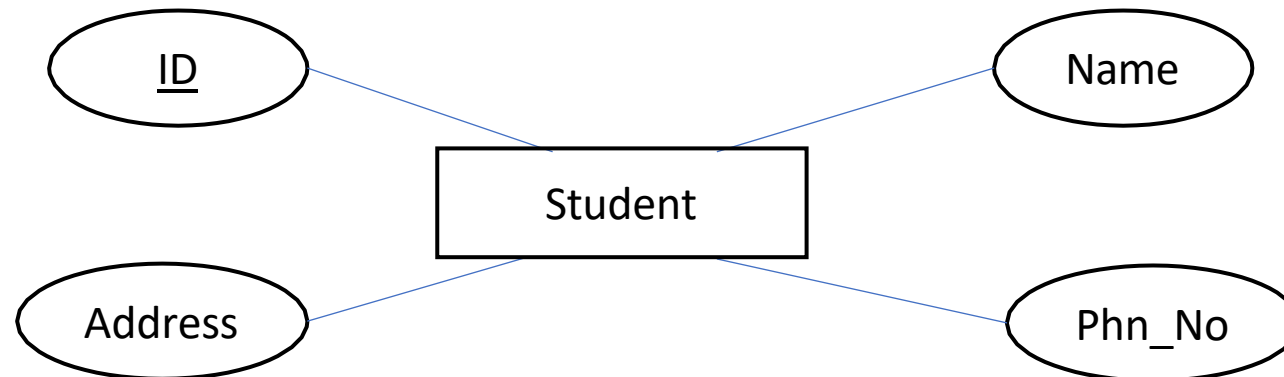
Database Design - Entity Relationship(ER) Modelling

Beyond ER Design,

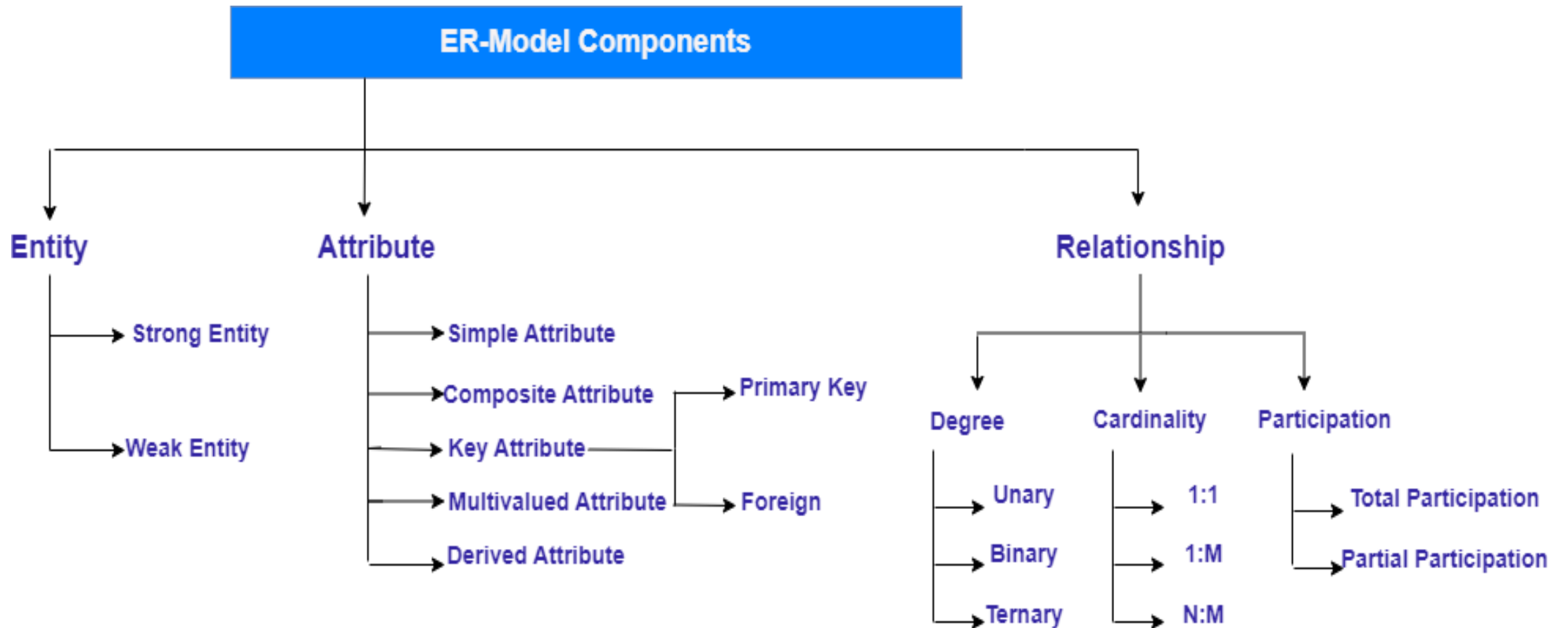
- **Schema Refinement:** involves analyzing the relations in the relational database schema to identify and address potential issues through a more objective and theory-guided process.
- **Physical Database Design:** It is important to consider anticipated workloads in the database design process, refining it to meet desired performance criteria
- **Application and Security Design:** This step involves recognizing distinct user groups and their respective roles. It requires specifying which parts of the database each group can access, implementing restrictions to ensure authorized access, and safeguarding against unauthorized access to sensitive areas.

ER (Entity Relationship) Model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc.



Components of ER diagram



1. **Entities:** An entity is anything in the real world, such as an object, class, person, or place.

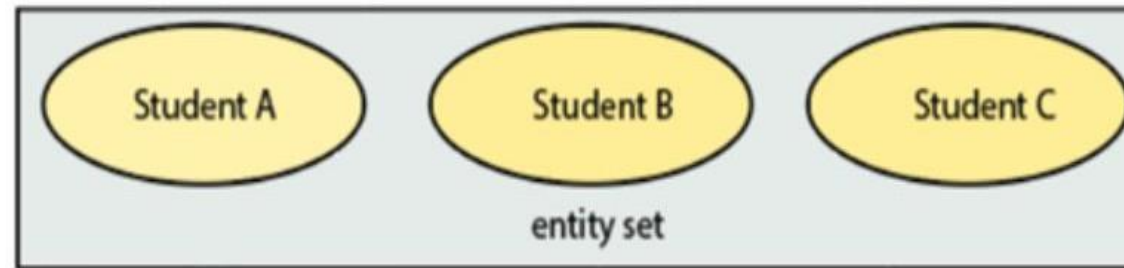
Example:

Company

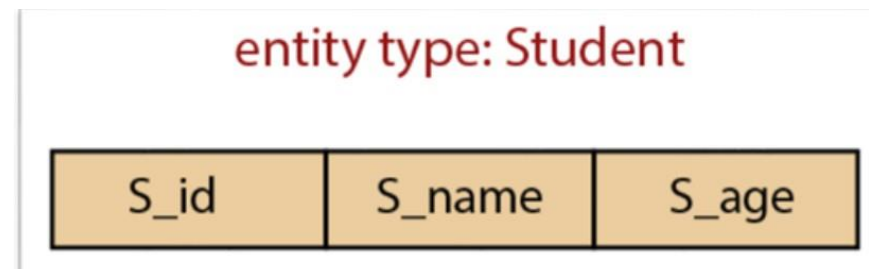
Loan

STUDENT

- **Entity set:** is a group of entities of similar kinds. It can contain entities with attributes that share similar values.



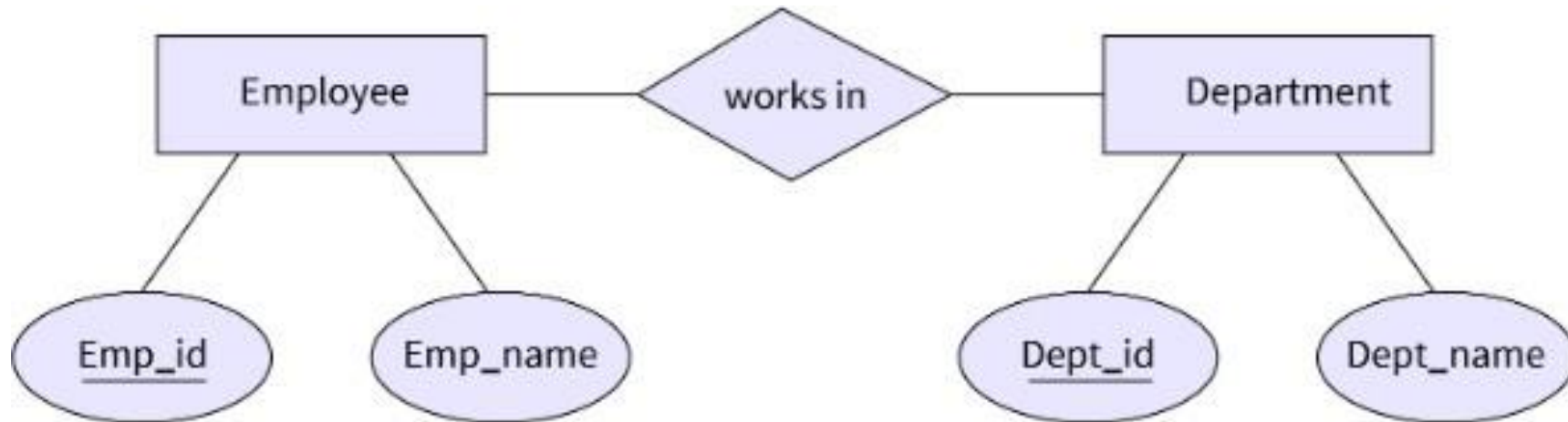
- **Entity types:** are the basic building blocks for describing the structure of data



Types of Entity:

1. Strong Entity:

- A **strong entity** is not dependent on any other entity in the schema.
- A strong entity will always have a primary key.
- Strong entities are represented by a single rectangle.
- The relationship of two strong entities is represented by a single diamond.



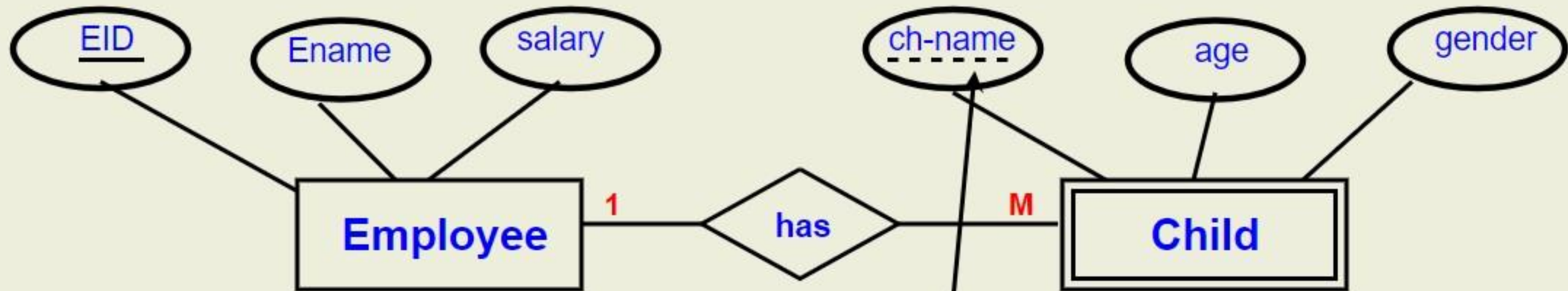
2. Weak Entity:

- A **weak entity** is dependent on a strong entity to ensure its existence.
- A weak entity does not have any primary key.
- It instead has a partial discriminator key.
- A weak entity is represented by a double rectangle.
- The relation between one strong and one weak entity is represented by a double diamond.
- This relationship is also known as **identifying relationship**.



Weak entity Notation

Example given here: is for CHILD entity related to an EMPLOYEE entity, where each employee may have one or more child and each child must have a parent employee.



Always all weak entities,

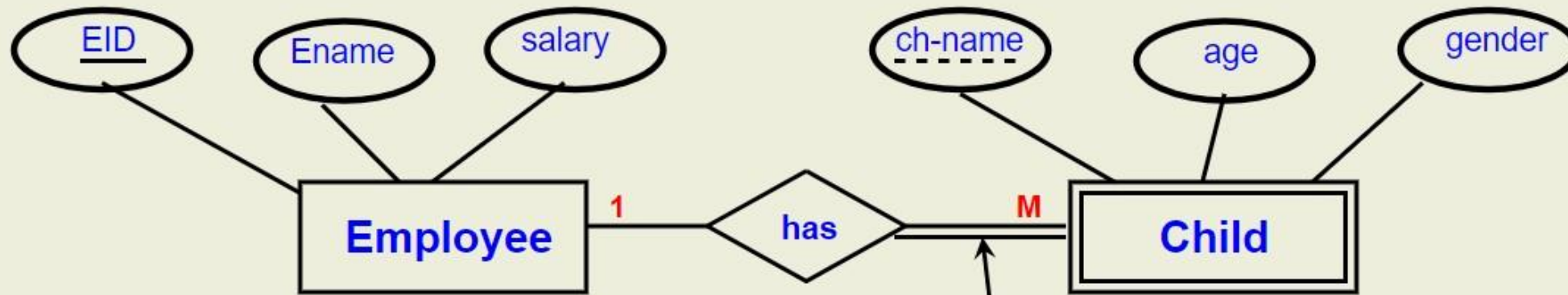
(e.g. Child entity) -----> has no primary key attribute

weak Child entity -----> only has partial key (Ch-name)

denoted by -----> dotted underline

Weak entity Notation

Example given here: is for CHILD entity related to an EMPLOYEE entity, where each employee may have one or more child and each child must have a parent employee.



- As weak Child entity has no PK **So,**
- **Child entity** should depend on-----> **the strong Employee entity** for unique identification.
- By relating **every instance of Child entity** -----> with **one corresponding instance of Employee entity**.
- This means that **weak Child entity** should -----> **totally participate in Has relationship**.
- This **total participation** -----> is denoted by **double lines**

Using real data example for weak entity illustration

Employee table, contains 3 employees (Ahmed, Hazem, Khaled) each of them has unique ID

Child table, contains 6 children (Ali, Nada, Ali, Sara, Morad, Nada) but no unique ID for anyone.

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000



<u>CH-name</u>	age	gender
Ali	12	male
Nada	9	female
Ali	12	male
sara	6	female
Morad	10	male
Nada	9	female

Employee is strong entity, because every employee has unique ID highlighted by yellow

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000

has

<u>CH-name</u>	age	gender
ALi	12	male
Nada	9	female
Ali	12	male
sara	6	female
Morad	10	male
Nada	9	female

Child is **weak entity**, No unique ID for children (e.g., two different children Ali & Ali both having the same data of age & gender)

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000

<u>CH-name</u>	age	gender
ALi	12	male
Nada	9	female
Ali	12	male
sara	6	female
Morad	10	male
Nada	9	female



Now, How to differentiate between children??????

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000

<u>CH-name</u>	age	gender
ALi	12	male
Nada	9	female
Ali	12	male
sara	6	female
Morad	10	male
Nada	9	female



The answer is, by their fathers

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000



<u>CH-name</u>	age	gender
ALi	12	male
Nada	9	female
Ali	12	male
sara	6	female
Morad	10	male
Nada	9	female

For example first child Ali is the son of employee Ahmed of ID=100

For example second child Ali is the son of employee Hazem of ID=200

<u>E-id</u>	E-name	salary
100	Ahmed	3000
200	Hazem	7000
300	Khaled	5000

has

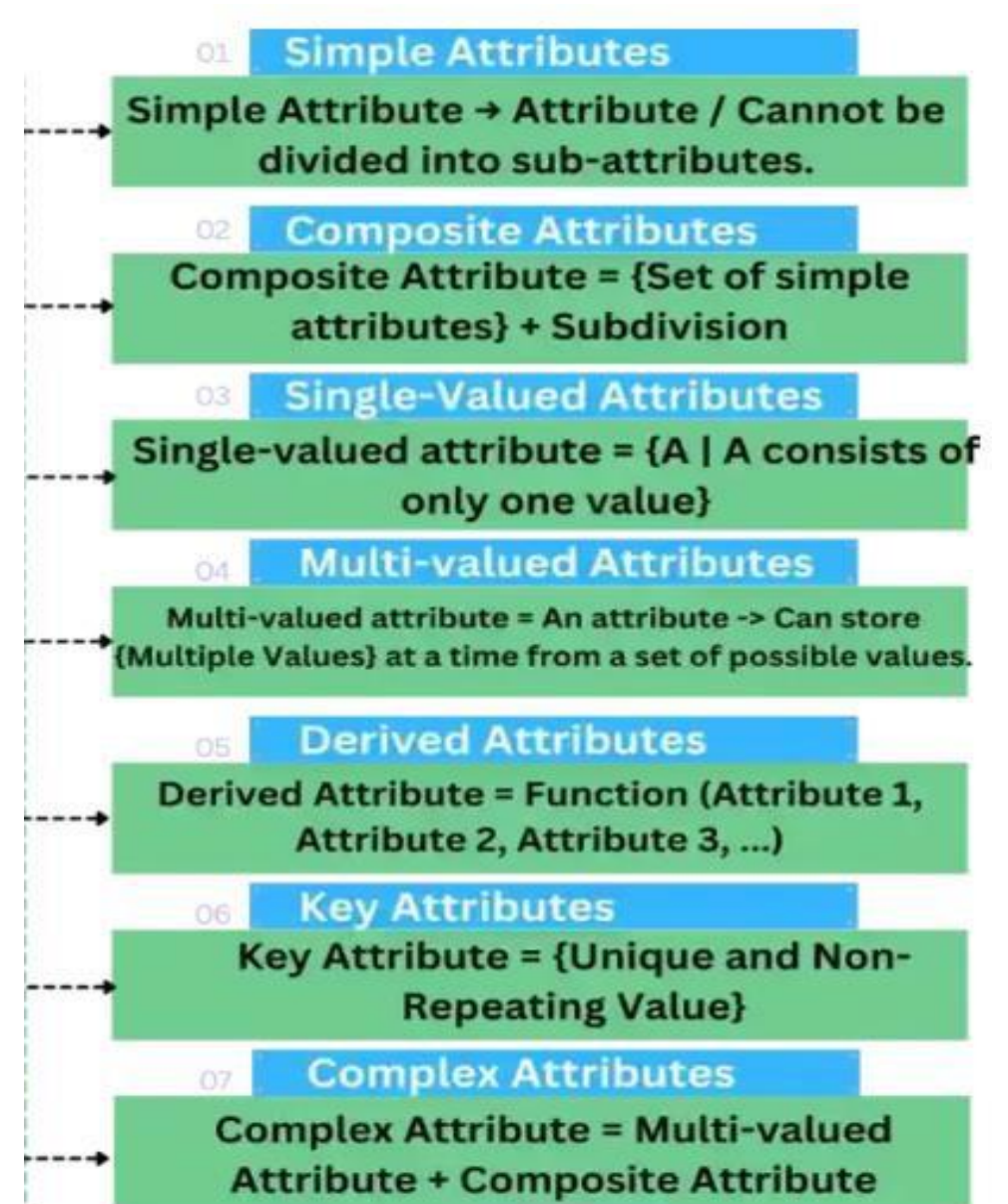
<u>CH-name</u>	age	gender	<u>E-id(FK)</u>
ALi	12	male	100
Nada	9	female	100
Ali	12	male	200
sara	6	female	200
Morad	10	male	200
Nada	9	female	300

Finally the (PK) of Child entity= partial key of Child entity + (FK) of Employee entity

Note: CH-name used alone as a partial primary key because it can only differentiate between the kids of the same father (i.e., no father has two kids with the same name)

2. Attributes:

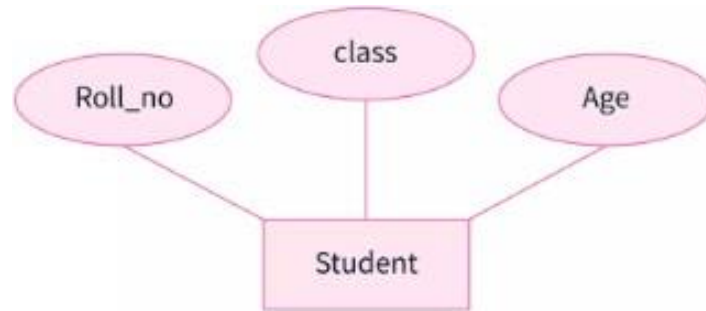
- An attribute in an Entity-Relationship Model describes the properties or characteristics of an entity.
- It is represented by an **oval** or **ellipse** shape in the ER diagram.
- Every oval shape represents one attribute and is directly connected to its entity which is in the rectangle in shape.
- For example, **employee_id**, **employee_name**, **Gender**, **employee_age**, **Salary**, and **Mobile no.** are the attributes which define entity type **Employee**



Types of attributes:

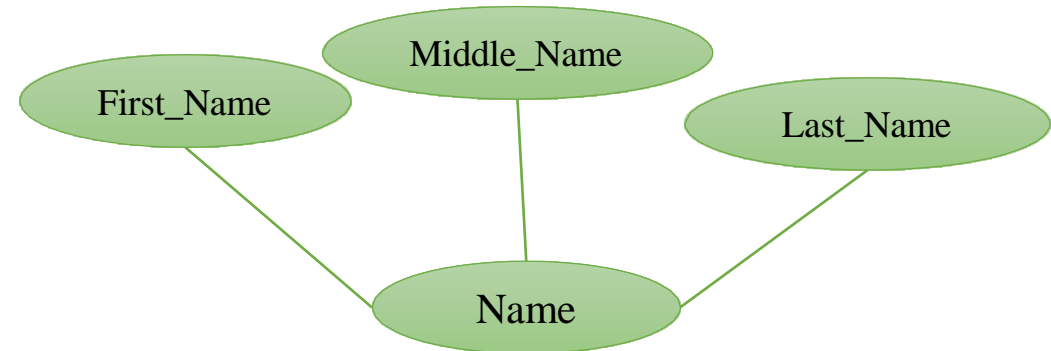
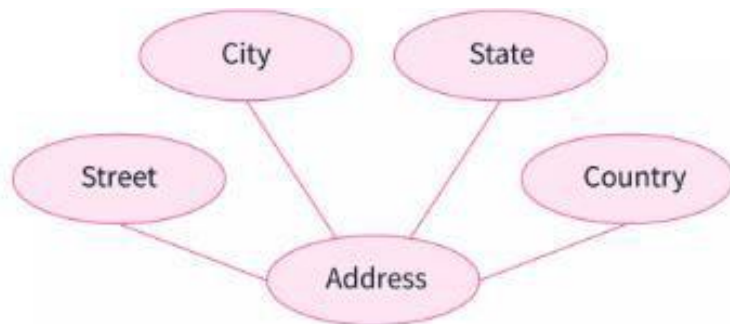
- **Simple Attributes:** Cannot be divided into sub-attributes. Also called atomic

Example:



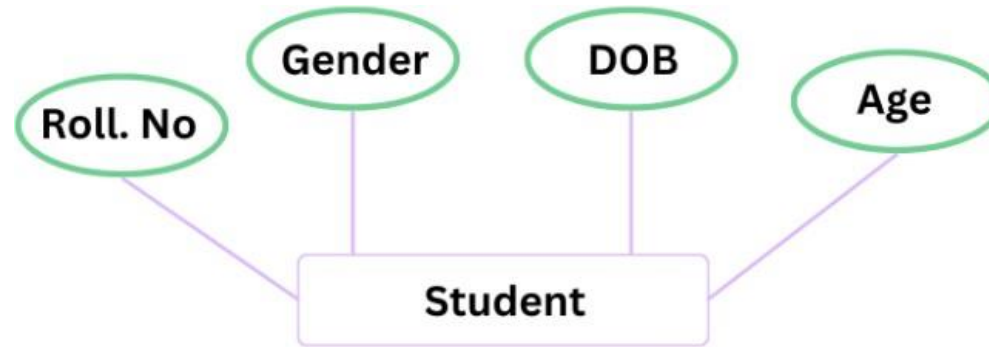
- **Composite Attributes:** {Set of simple attributes} + Subdivision.

Example: Name -> First_name, Last_name, Middle_name

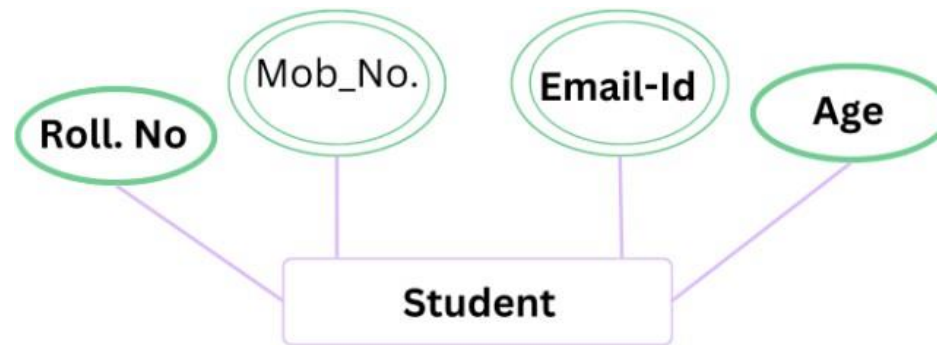


- **Single Valued Attributes:** Single-valued attributes are attributes that have only one value for each instance of an entity and cannot store more than one value.

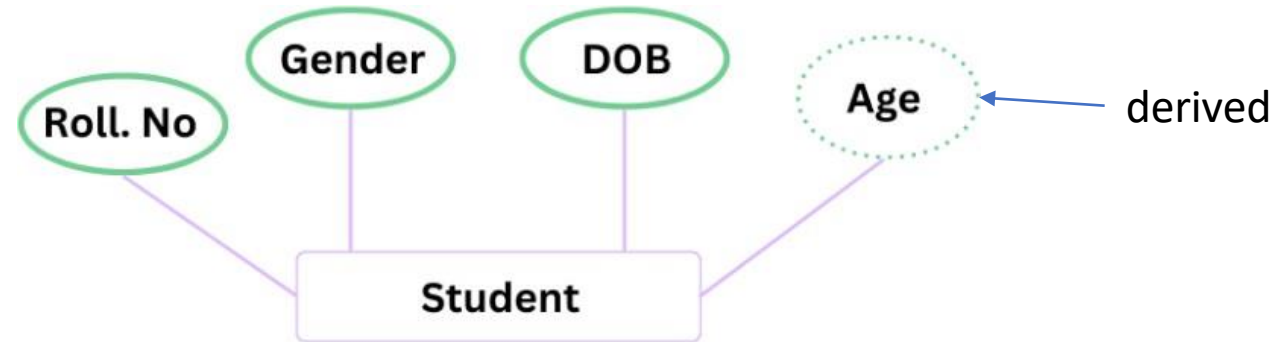
Example: Single-valued attributes are like your name – you can only have one name, and it stays the same for you all your life.



- **Multi Valued Attributes:** An attribute -> Can store {Multiple Values} at a time from a set of possible values.



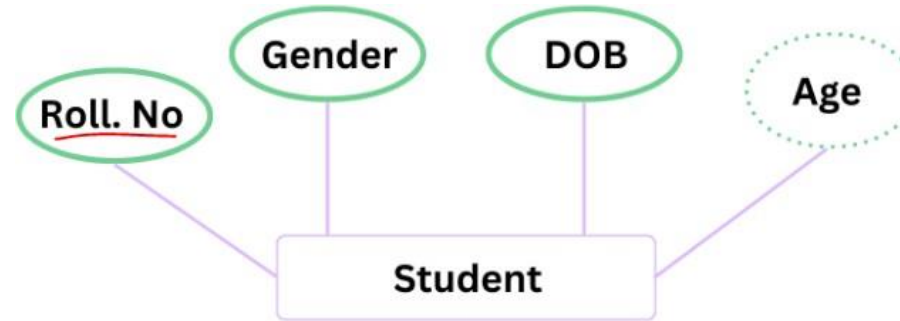
- **Derived attributes** : Values that can be derived from other attributes and are always dependent on other attributes for their value.



- **Stored attributes**: Values of stored attributes remain constant and fixed for an entity instance and also, and they help in deriving the derived attributes.

Example: **For example**, the **Age** attribute can be derived from the **DOB attribute**, and also, the **DOB attribute** has a fixed and constant value throughout the life of an entity. Hence, the **DOB attribute** is a **stored attribute**.

- **Key Attributes:** Attributes that serve as the primary key for an entity, allowing us to uniquely identify an entity from a set of entities. {Unique and Non-Repeating Value}.



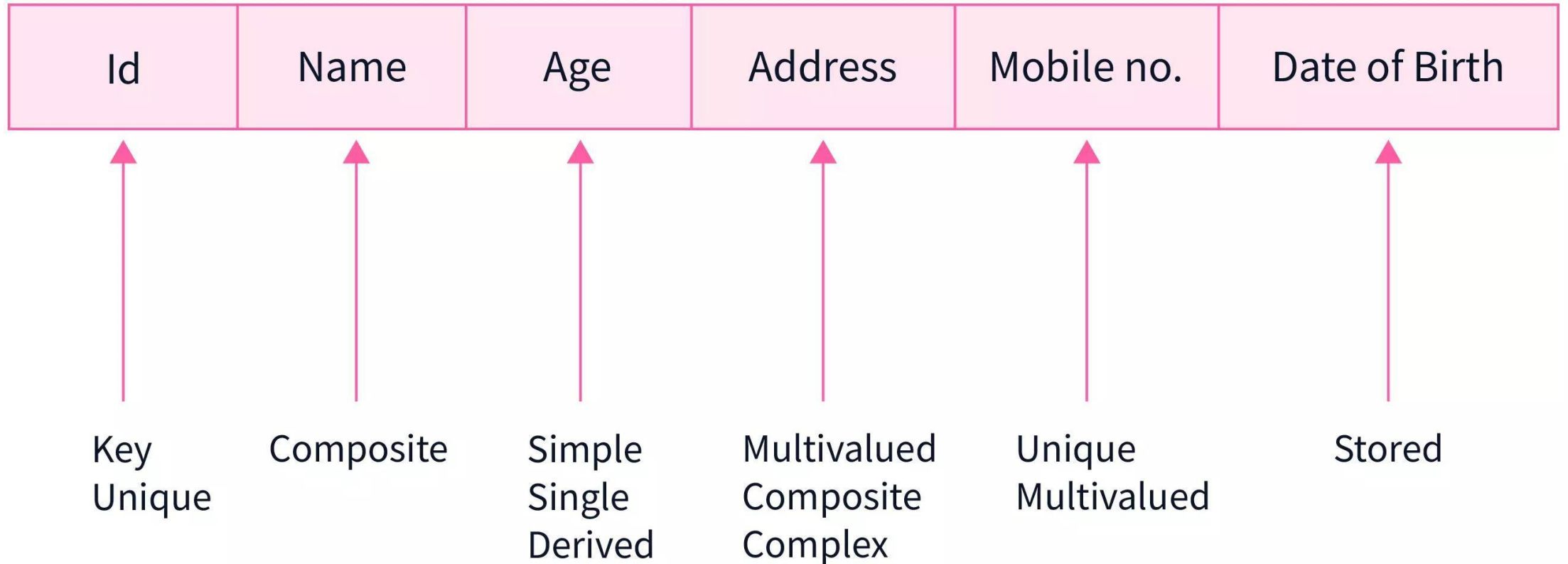
- **Complex Attributes:** Formed by combining multi-valued & composite attributes, resulting in a value with many sub-sections.

Multi-valued Attribute + Composite Attribute

Example:

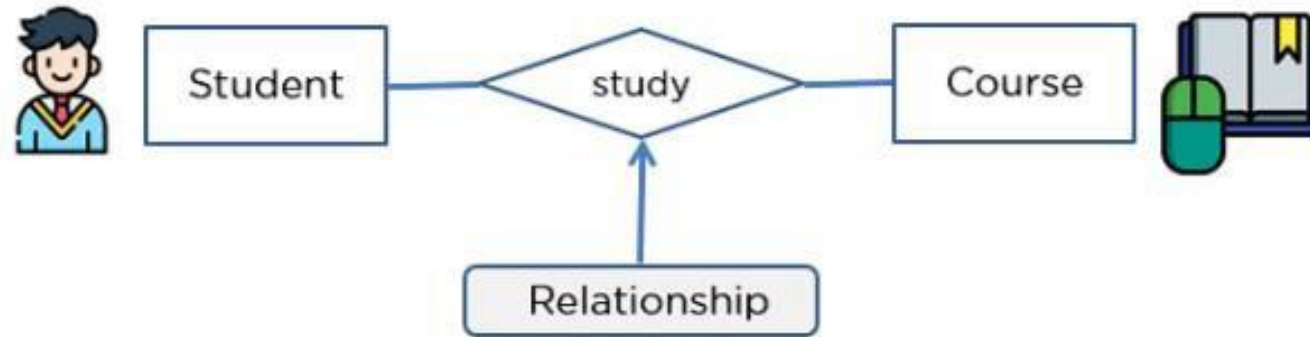
Address attribute (can have more than one address – multi valued
can be sub divided like city,street etc – composite)

Example



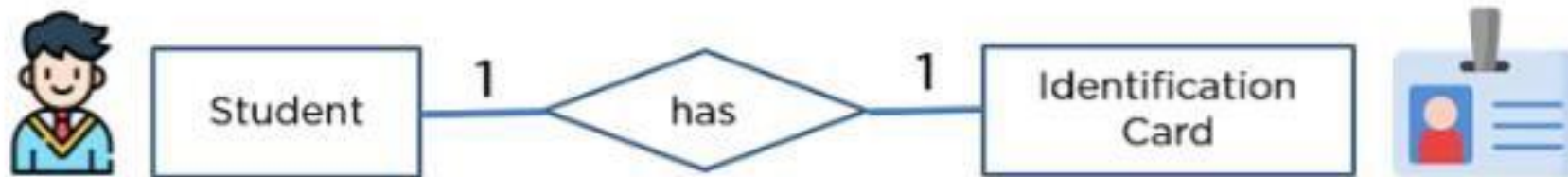
3. Relationships:

- The diamond shape showcases a relationship in the ER diagram.
- It depicts the relationship between two entities.



Types of Relationships:

- **One to One relationship (1: 1)** : When a single element of an entity is associated with a single element of another entity, it is called a one-to-one relationship



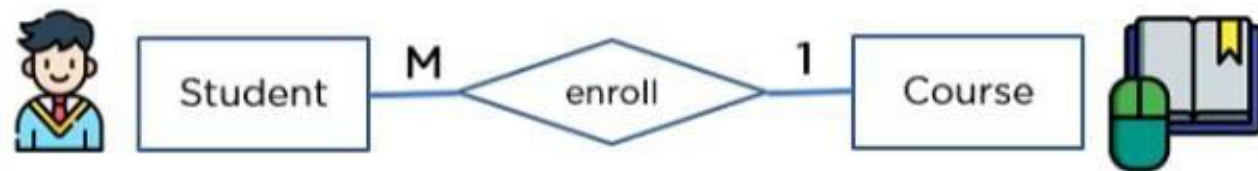
- **One to Many relationship (1: M):** When a single element of an entity is associated with more than one element of another entity, it is called a one-to-many relationship.



- **Many to One relationship (M: 1):** When more than one element of an entity is related to a single element of another entity, then it is called a many-to-one relationship.



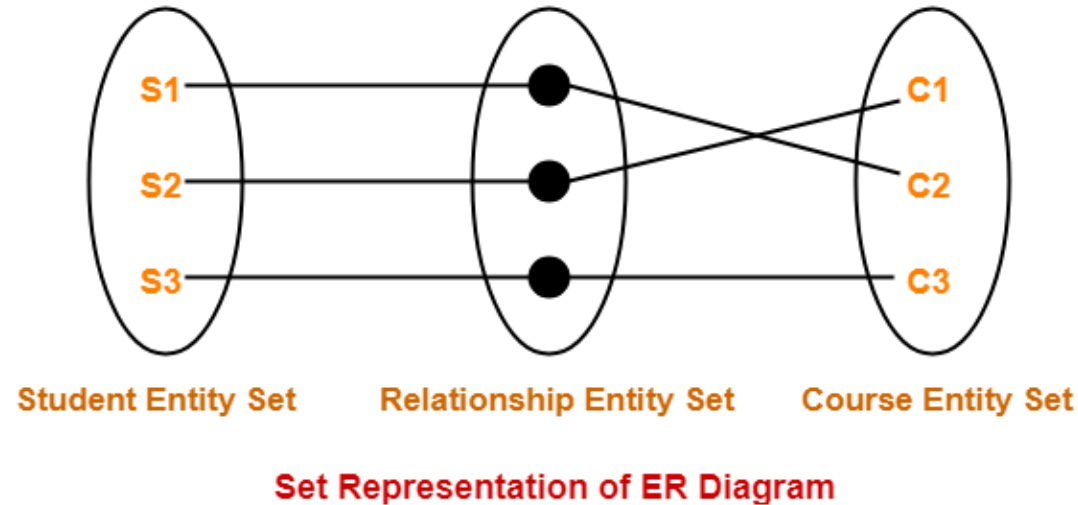
- **Many to Many relationship (M: M):** When more than one element of an entity is associated with more than one element of another entity, this is called a many-to-many relationship.



4. Relationship Set:

- A relationship set is a set of relationships of same type.

Example:



Degree of Relationship set:

Degree of Relationship set = No. of entity sets participating in a relationship set.

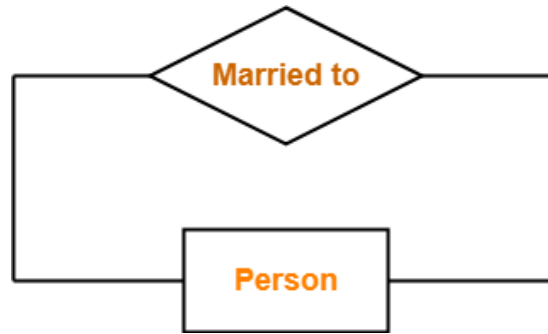
Types of Relationship Set:

- Unary relationship set
- Binary relationship set
- Ternary relationship set
- N – array relationship set

- **Unary relationship set:** Unary relationship set is a relationship set where only one entity set participates in a relationship set.

Example: One person can marry only one person

one person can have only one name on the certificate

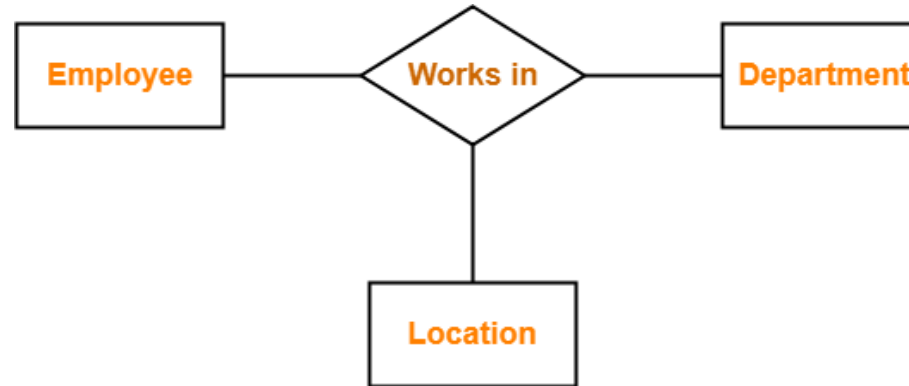


- **Binary relationship set:** a relationship set where two entity sets participate in a relationship set.

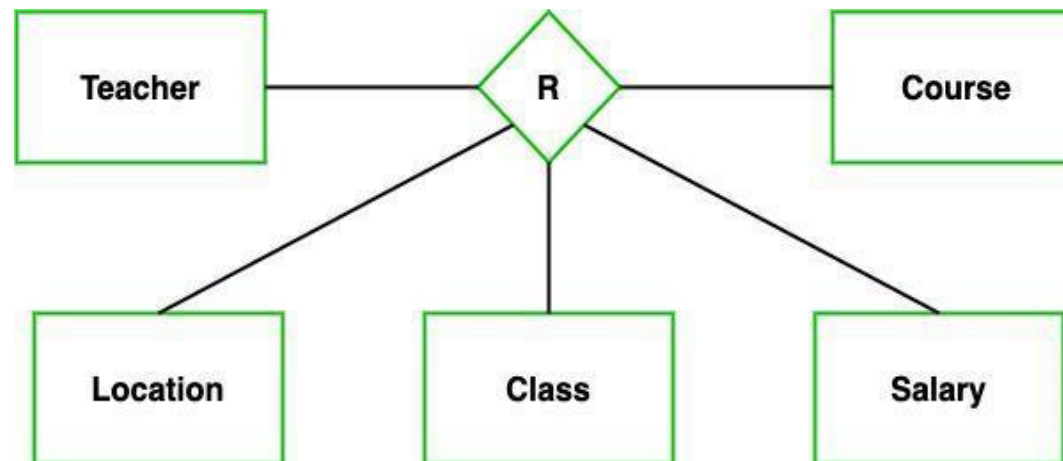
Example: Student is enrolled in a course



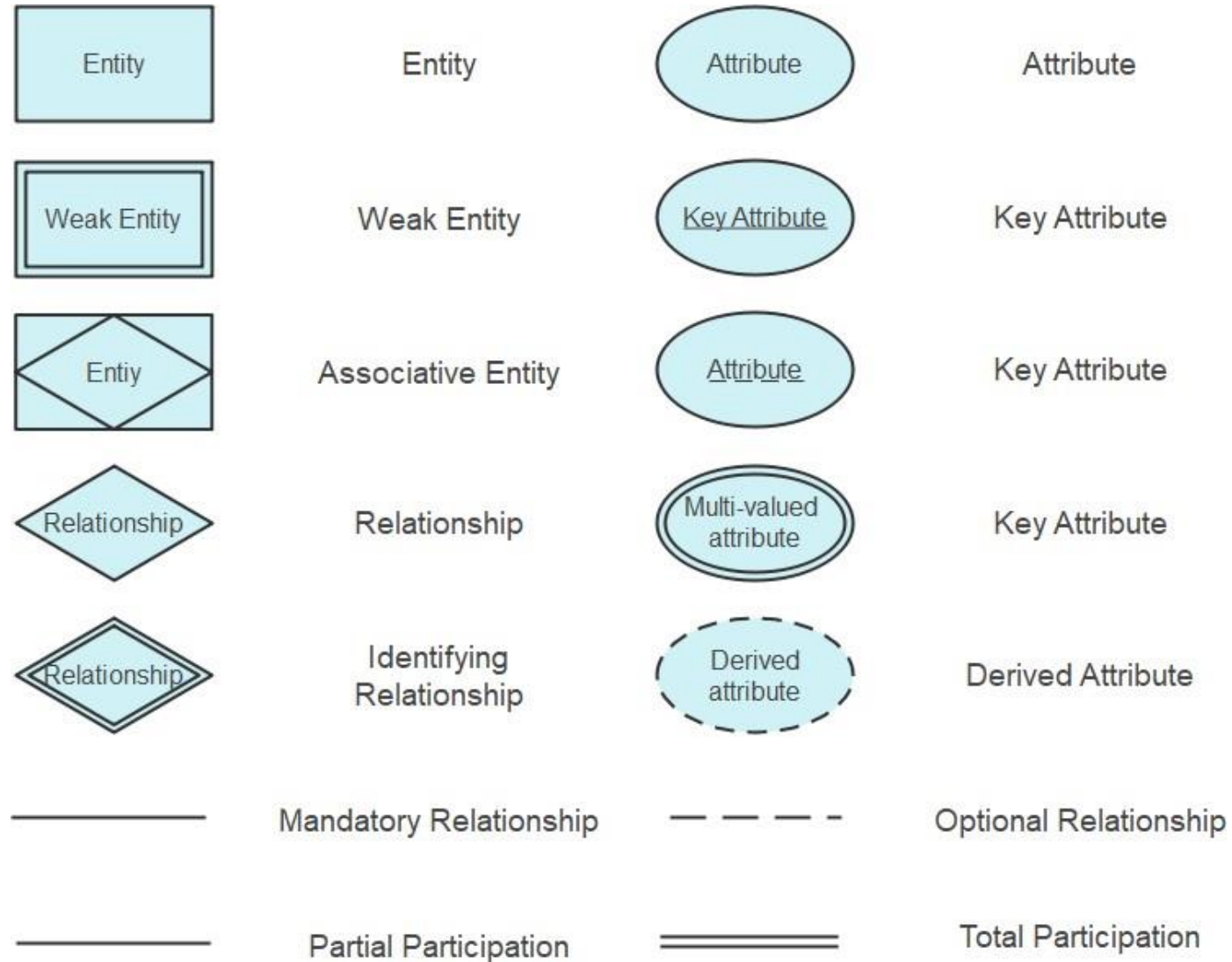
- **Ternary relationship set:** a relationship set where three entity sets participate in a relationship set.



- **N- array relationship set:** a relationship set where 'n' entity sets participate in a relationship set.

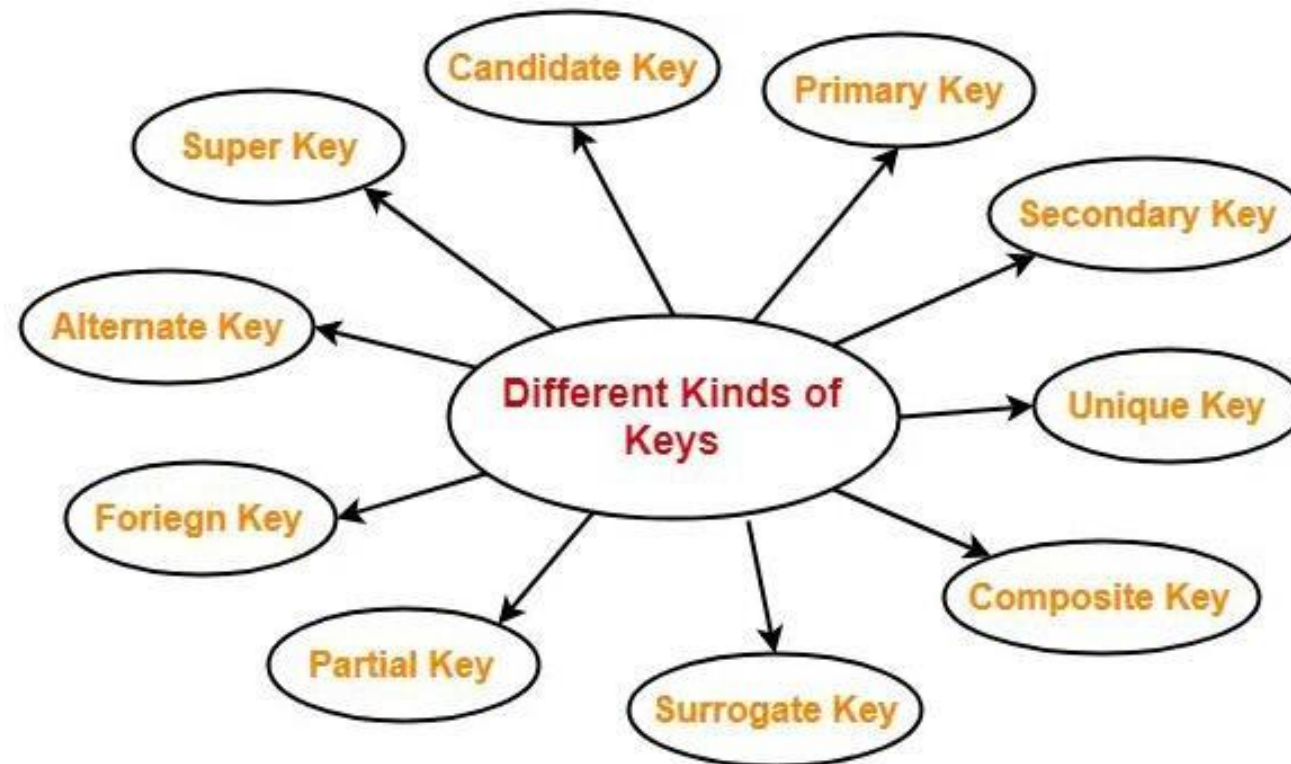


NOTATIONS OF ER DIAGRAM



Additional features of ER model

1. Key Constraints: A key in DBMS is an attribute or a set of attributes that help to uniquely identify a tuple in a relation (or table). Keys are also used to establish relationships between the different tables and columns of a relational database. Individual values in a key are called key values.



Super Key: A super key is a group of single or multiple keys which identifies rows in a table.

- Example:

Emp_ID	Emp_Name	Emp_Aadhar	Emp_Salary	Emp_phone	Emp_Email
101	HARI	0000000000	45000	123456	ABC@gmail.com
102	RAMU	1111111111	50000	543432	NULL
103	HARI	2222222222	45000	NULL	aec@gmail.com
104	RAJU	3333333333	47000	534332	afg@gmail.com

{Emp_ID}, {Emp_ID, Emp_phone}, {Emp_Aadhar}

{Emp_ID, Emp_Name}, {Emp_ID, Emp_Salary}

{Emp_ID, Emp_Aadhar}, {Emp_ID, Emp_Name, Emp_Aadhar}

{Emp_ID, Emp_Name, Emp_Aadhar, Emp_Salary}

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

Set of super keys obtained

{ Emp_SSN }

{ Emp_Id }

{ Emp_email }

{ Emp_SSN, Emp_Id }

{ Emp_Id, Emp_name }

{ Emp_SSN, Emp_Id, Emp_email }

{ Emp_SSN, Emp_name, Emp_Id }

Roll No.	Name	Age	Phone
1	Aryan	21	7491901521
2	Sachin	25	870904365
3	Prince	20	784600652
4	Anuj	21	9876534523

Diagram illustrating Super Keys for the second table:

- Roll No. is a Super Key.
- Name is a Super Key.
- Phone is a Super Key.
- Roll No. and Name together form a Super Key.

{ Roll No }

{ Phone }

{ Roll No, Name }

{ Roll No, Name, Phone }

Candidate Keys: Minimal set of super Key. The candidate key is chosen from the set of super keys.

- Example:

Emp_ID	Emp_Name	Emp_Aadhar	Emp_Salary	Emp_phone	Emp_Email
101	HARI	0000000000	45000	123456	ABC@gmail.com
102	RAMU	1111111111	50000	543432	NULL
103	HARI	2222222222	45000	NULL	aec@gmail.com
104	RAJU	3333333333	47000	534332	afg@gmail.com

- {Emp_ID}
- {Emp_Aadhar}

Example

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

The candidate keys *Stud ID*, *Roll No.*, and *Email* allow us to identify each student record individually.

Primary Key: Any key from the candidate key but it should be not null or updated / combinations in candidate key.

- A value must be present in the Primary Key column, and the Primary Key field cannot be left NULL.
- No two rows in the table may have the same values in that particular column.
- No value can be changed or modified in this primary key column.
- **Example:**

Emp_ID	Emp_Name	Emp_Aadhar	Emp_Salary	Emp_phone	Emp_Email
101	HARI	0000000000	45000	123456	ABC@gmail.com
102	RAMU	1111111111	50000	543432	NULL
103	HARI	2222222222	45000	NULL	aec@gmail.com
104	RAJU	3333333333	47000	534332	afg@gmail.com

- {Emp_ID} / {Emp_Aadhar}

Alternate Key: An alternate is a secondary candidate key that is capable of identifying a row uniquely. However, such a key is not used as a primary key because, out of all the generated candidate keys, only one key is selected as the primary key. Thus, the other remaining keys are known as **Alternate Keys** or **Secondary Keys**.

Emp_ID	Emp_Name	Emp_Aadhar	Emp_Salary	Emp_phone	Emp_Email
101	HARI	0000000000	45000	123456	ABC@gmail.com
102	RAMU	1111111111	50000	543432	NULL
103	HARI	2222222222	45000	NULL	aec@gmail.com
104	RAJU	3333333333	47000	534332	afg@gmail.com

- If {Emp_ID} is a primary key then {Emp_Aadhar} is an alternate key
- If {Emp_Aadhar} is an primary key then {Emp_ID} is an alternate key

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

Candidate Keys :

Emp_SSN

Emp_Id

Emp_email

Example

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

Alternate Key

Primary Key

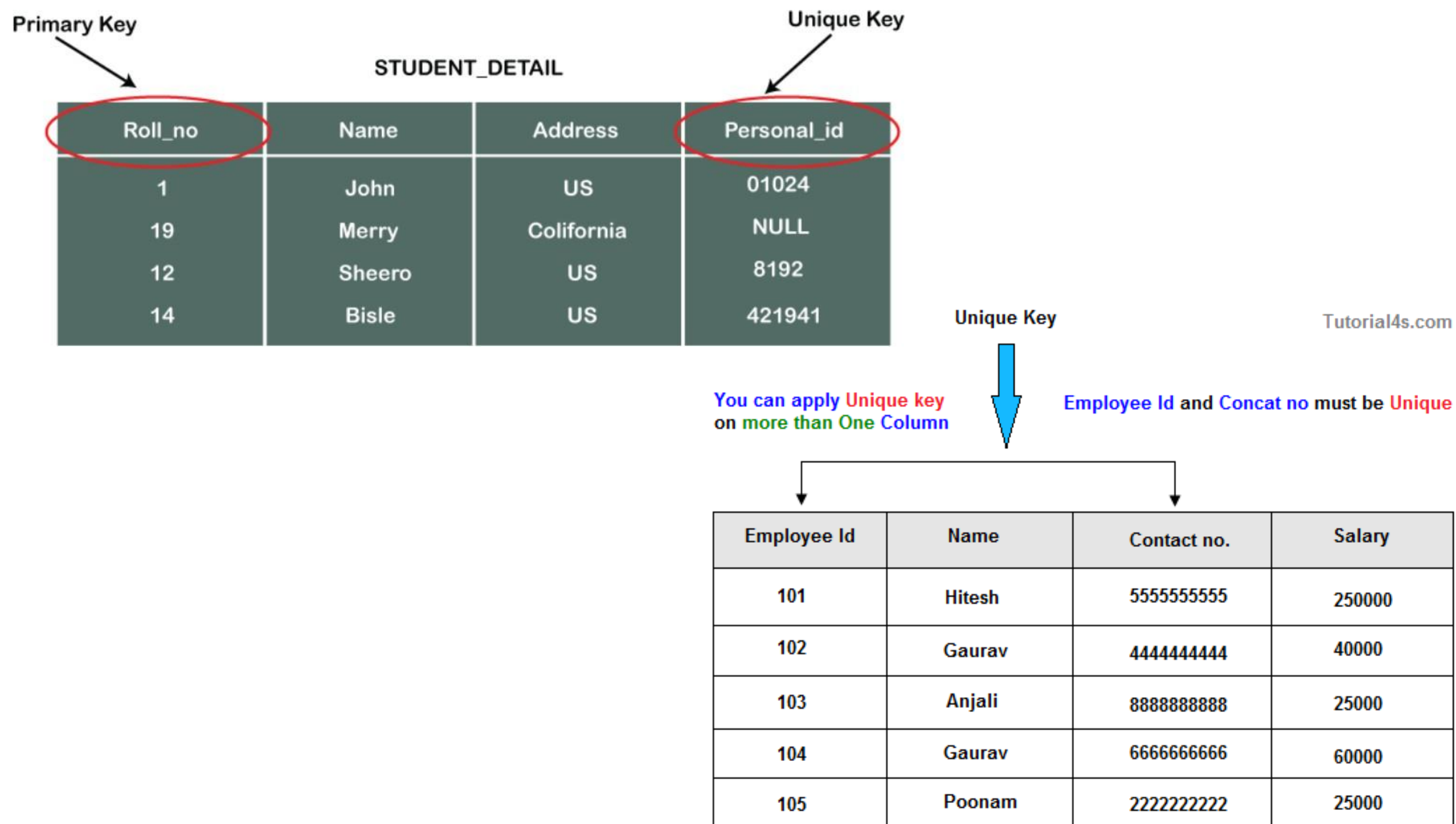
Alternate Key

Unique Key: A unique key is used to remove the duplicity of values in a table. However, the usage of a primary key is the same, but there is a difference between both keys. A primary key cannot take a **NULL** value, but a unique key can have one **NULL** value as its value.

Emp_ID	Emp_Name	Emp_Aadhar	Emp_Salary	Emp_phone	Emp_Email
101	HARI	0000000000	45000	123456	ABC@gmail.com
102	RAMU	1111111111	50000	543432	NULL
103	HARI	2222222222	45000	NULL	aec@gmail.com
104	RAJU	3333333333	47000	534332	afg@gmail.com

Example:

- **Emp_phone** and **Emp_Email** can be unique key.



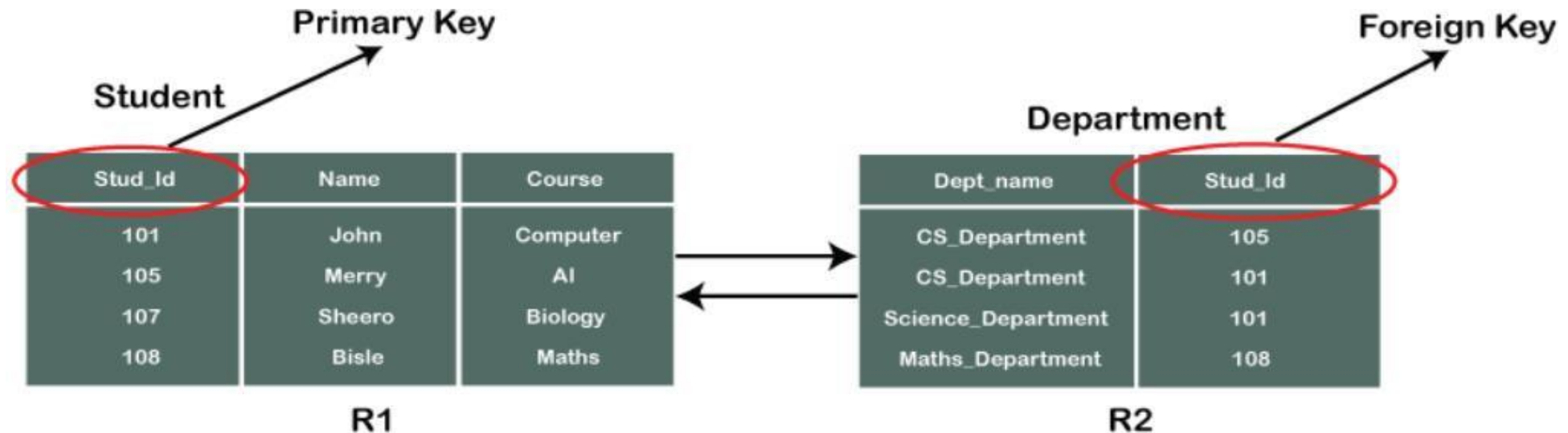
Foreign Key: A foreign key is different from a super key, candidate key or primary key because a foreign key is the one that is used to link two tables together or create connectivity between the two.

Student table

Stud_Id	Name	Course
101	John	Computer
105	Merry	AI
107	Sheero	Biology
108	Bisle	Maths

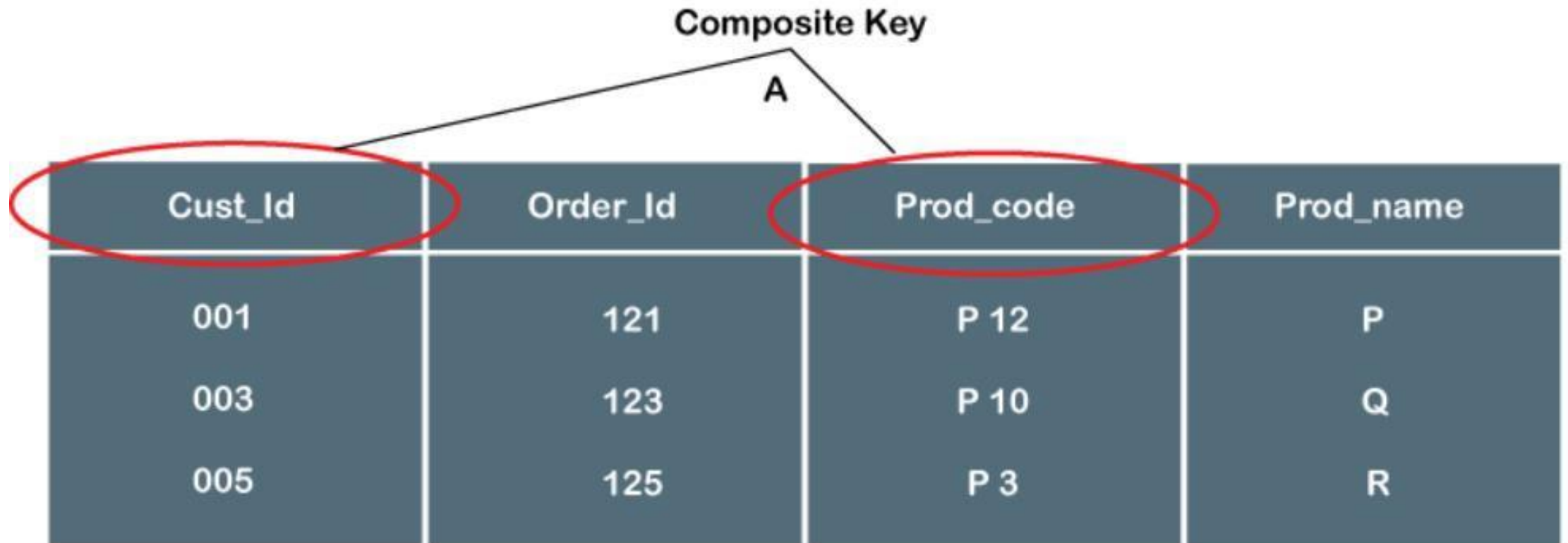
Department table

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Math_Department	108



Composite Key: Two or more attributes together form a composite key that can uniquely identify a tuple in a table. We need to find out such table columns combination that can form a candidate key and hence a composite key.

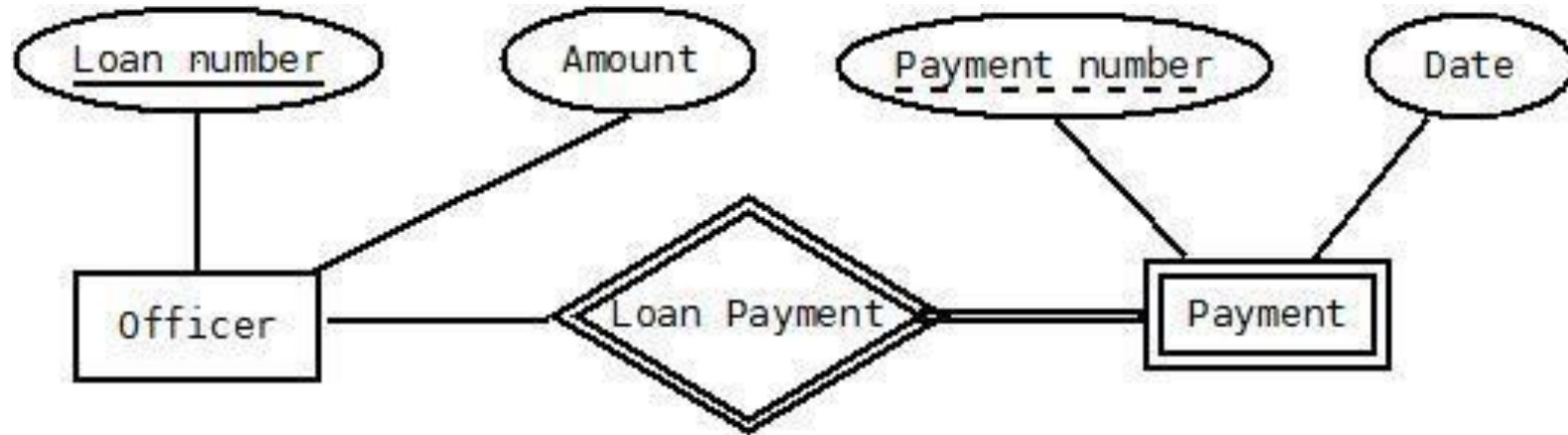
Example:



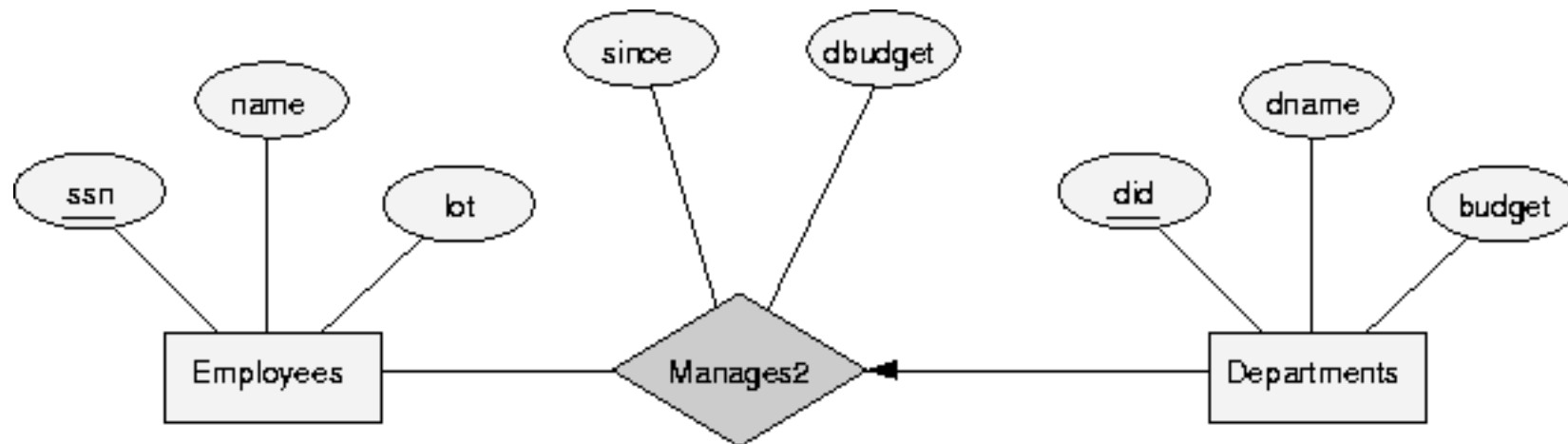
The diagram illustrates a composite key 'A' formed by the combination of 'Cust_Id' and 'Prod_code' columns. These two columns are circled in red, and a line labeled 'A' points to them from the text 'Composite Key' above. The table below shows three rows of data, where each row has a unique combination of 'Cust_Id' and 'Prod_code'.

Cust_Id	Order_Id	Prod_code	Prod_name
001	121	P 12	P
003	123	P 10	Q
005	125	P 3	R

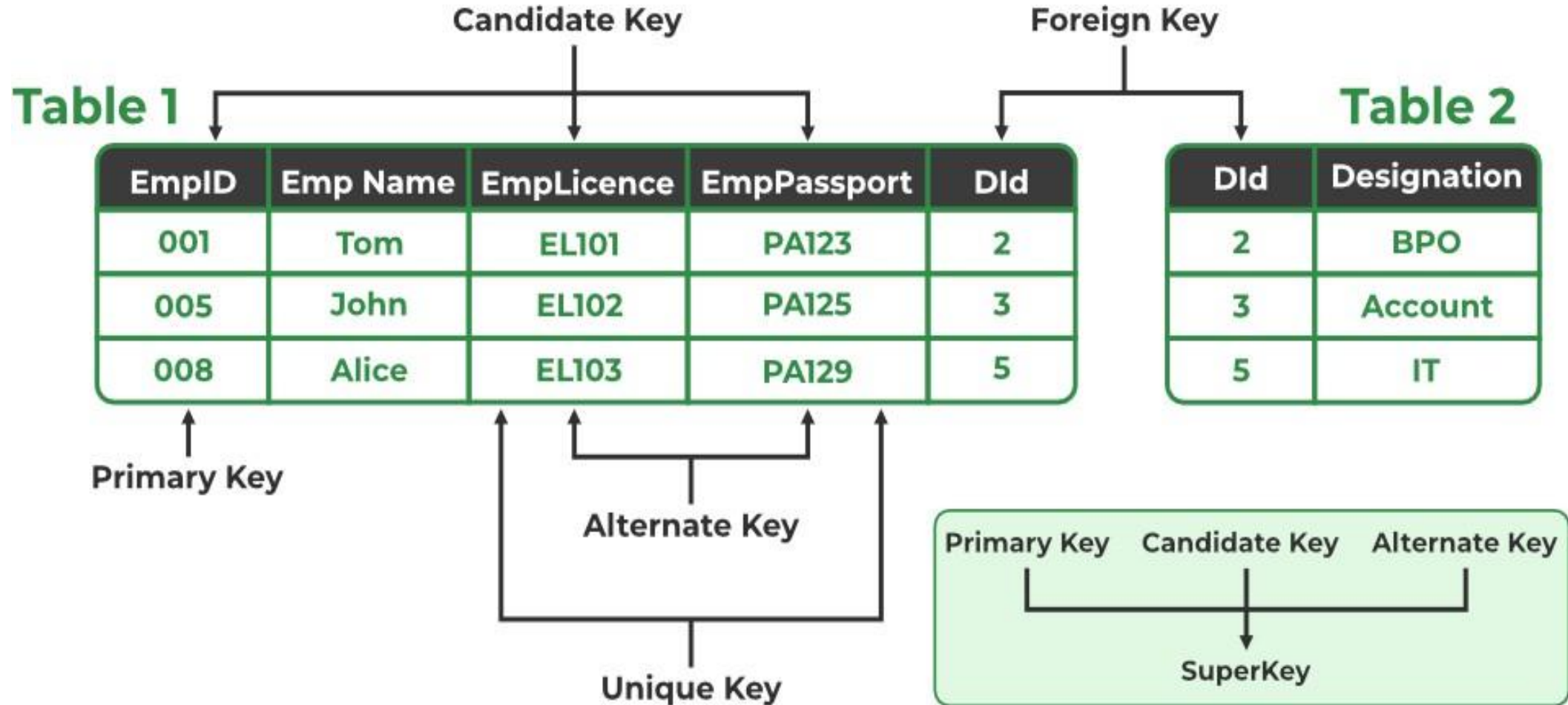
Partial Key: The set of attributes that are used to uniquely identify a weak entity set is called the Partial key. The partial Key of the weak entity set is also known as a discriminator.



Example



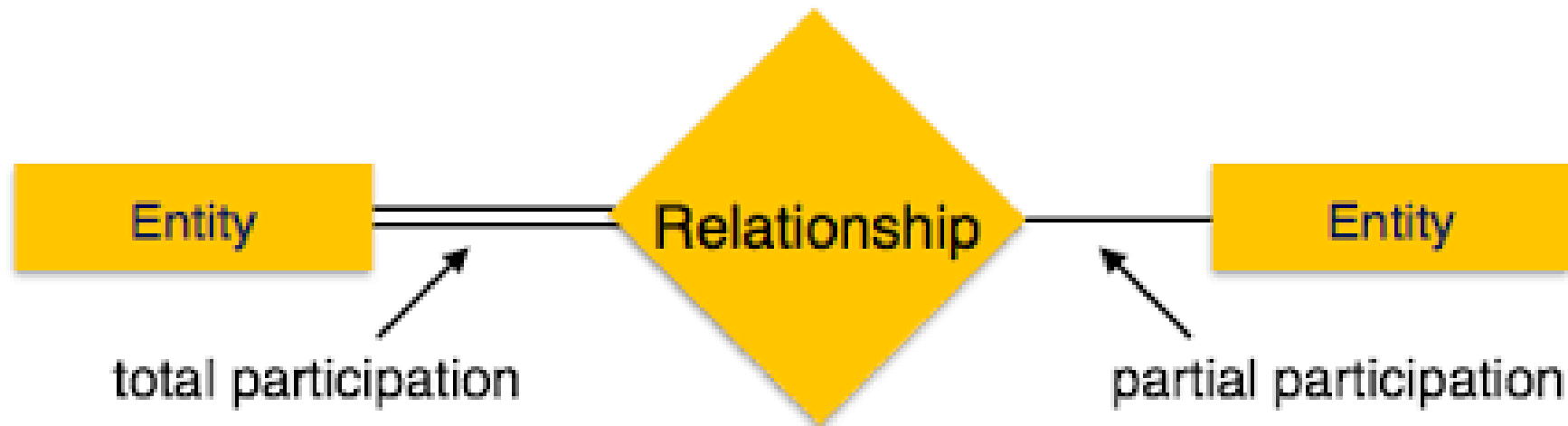
EXAMPLE



Additional features of ER model

2. Participation Constraints:

- Total Participation: Each entity is involved in the relationship. Total participation is represented by double lines.
- Partial Participation: Not all entities are involved in the relationship. Partial participation is represented by single lines.

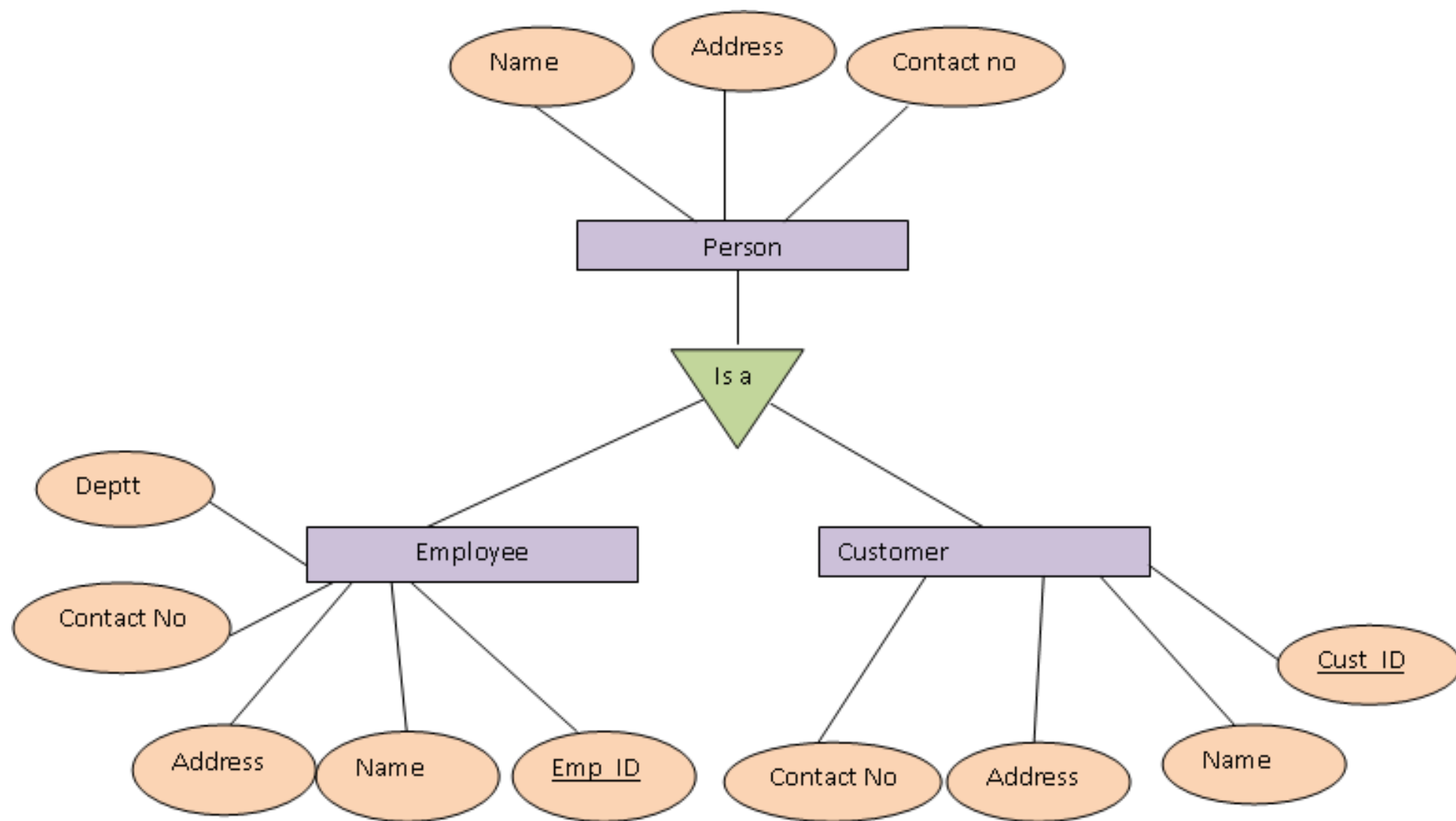


Additional features of ER model - 3. Generalization, Specialization and Aggregation:

Generalization:

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- Generalization is more like subclass and superclass system, but the only difference is the approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

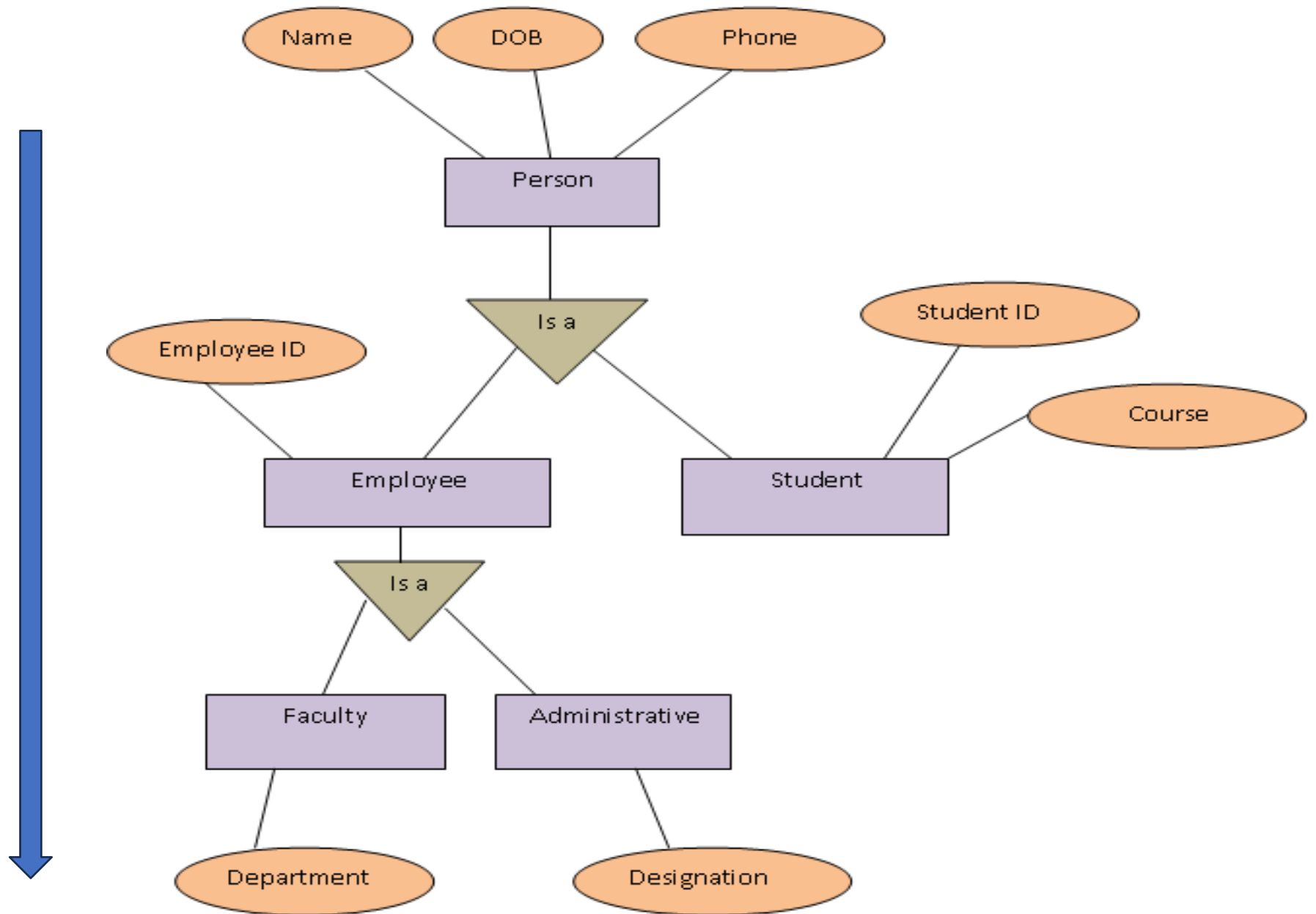
Bottom- Up
Approach



Specialization:

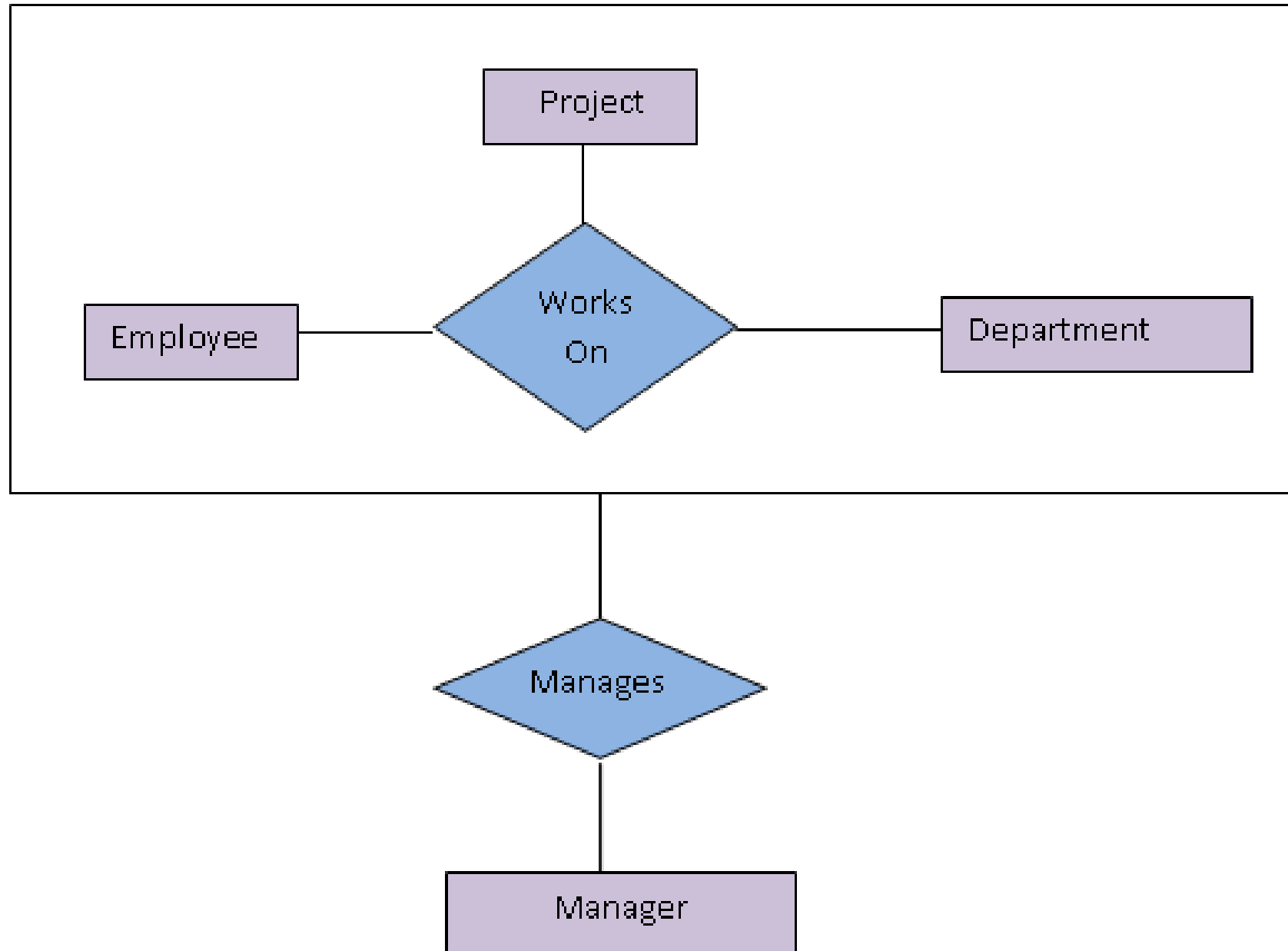
- It's a Top Down approach to design ER model where subclasses of entity types are defined that inherits attributes of another entity type.
- The entity type which passes on attributes to other entities is called the super class of the specialization.
- In other words, Specialization is the process of classifying class of objects into more specialized subclasses. It is a conceptual refinement.

TOP- DOWN Approach



Aggregation:

- Aggregation is an abstraction through which we can represent relationships as a higher level entity.
- It is an abstraction concept for building composite objects from their component objects.
- There is a limitation in ER modeling that we don't have a way to represent relationship among relationships.
- To overcome this limitation aggregation is the solution.



CONCEPTUAL DATABASE DESIGN WITH THE ER MODEL

- Entity versus Attribute
- Entity versus Relationship
- Binary versus Ternary Relationships
- Aggregation versus Ternary Relationships