Database Management System (DBMS)

Unit - I

Introduction

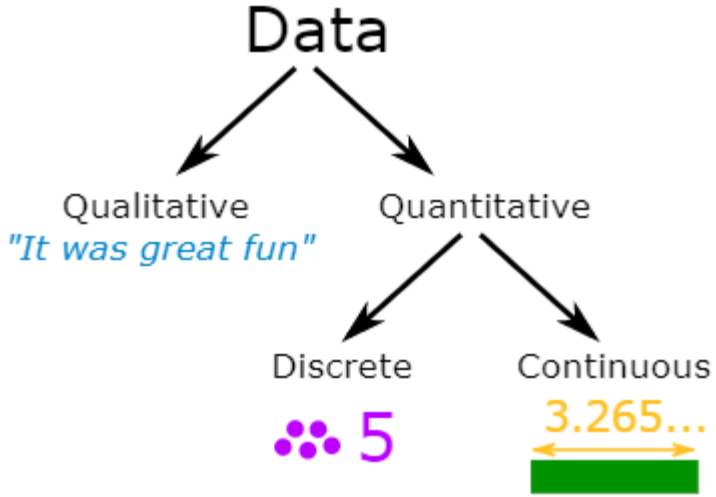
- Database Management System (DBMS) Concepts
- Relational Database Model
- Codd rules
- The Entity-Relationship (ER) Model
- Concepts of Data Independence, Data Sharing, Data Integrity
- Data Protection, System Catalog
- Users associated with database systems and their roles
- Normalization and De-Normalization

Terminologies

- **Data :** Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.
- **Data can be qualitative or quantitative.**
- **Qualitative:** Qualitative data is descriptive information (it describes something)
  - Your favorite holiday destination
  - Behaviour of a person
  - Describe the smell of a perfume
- **Quantitative:**

  Quantitative data is numerical information (numbers)

  Discrete data is counted, Continuous data is measured
  - Petals on a flower (Discrete)
  - Customers in a shop (Discrete)
  - Height (Continuous)
  - Weight (Continuous)



Terminologies

- **Entity :** Real world object that can be easily identified.
  - Student, Product, Book, Department etc.
- **Attribute :** Properties of an entity.
  - Student (Id, Name, Address) Product (Product Id, Name, Price)
  - Book (Book Id, Author, Publisher, No. of pages) , Department (Dept Id, Name)
- **Information :** Data is processed for meaningful to the end users.
  - Attendance report, Product Stock etc.
- **Data Types :** Data having predefined characteristics.

E.g. Integer, Real (Float), Character, String, Date, etc.

  - Integer Values 1, 2, …….
  - Real (Float) Values 1.1, 2.9, 3.85, ……
  - Character Values : A … Z, a ….z , # , @ , ……
  - String Values Ambar, Harsh, Gujarat, India ……..
  - Date Values : 1/1/1980, 12/8/2000 , etc.

Fundamentals of RDBMS

- **Database**
  - It is a computerized record-keeping system.
  - It is a collection of coherent and meaningful data

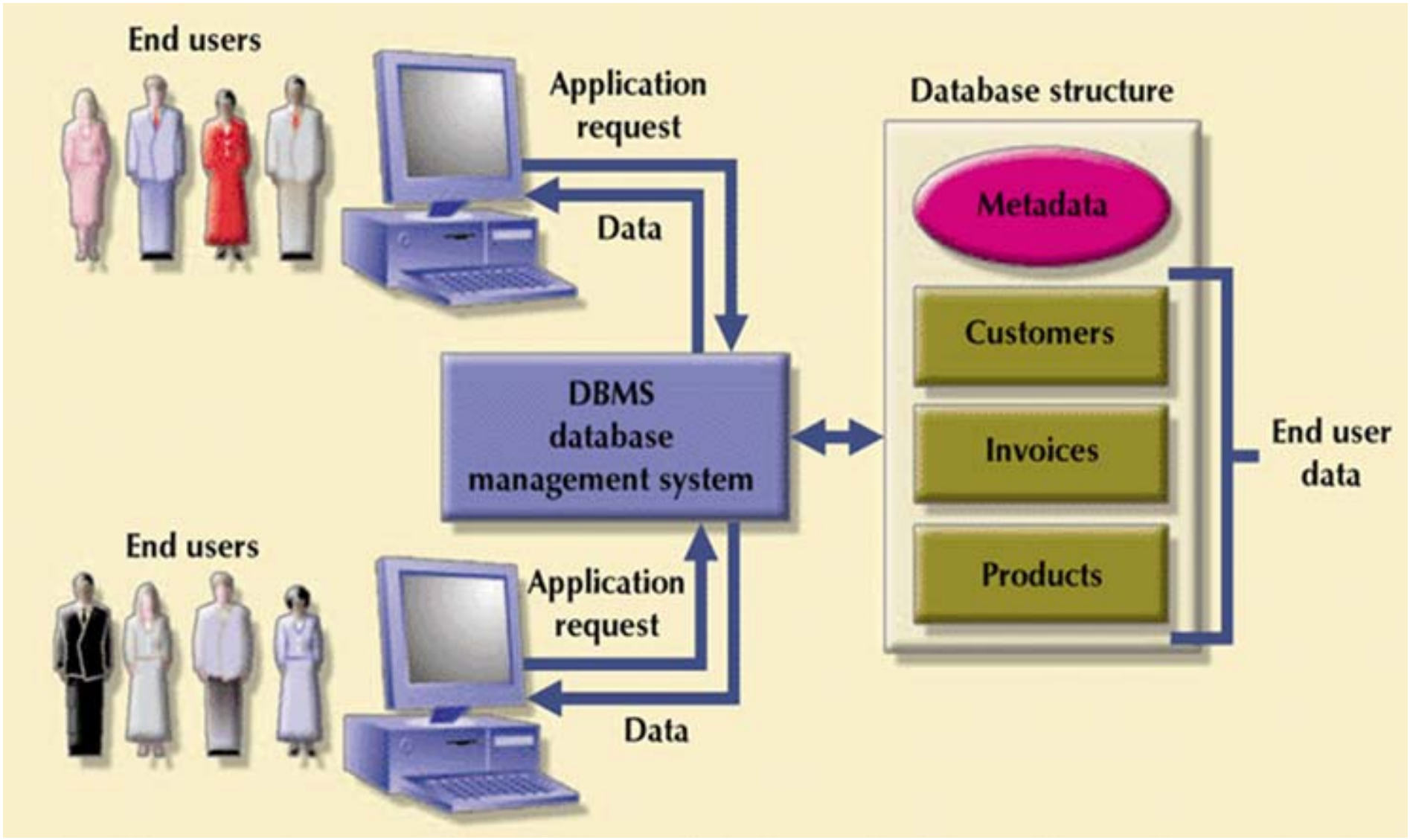
  E.g. A company database : Customers, Invoices and Products
- **DBMS**
  - It is a set of programs designed to create and maintain a database.

  OR
  - A system that allows creating, inserting, deleting, updating and processing of data.

  E.g. Oracle, Sybase, Dbase, FoxPro, MS Access
- **RDBMS**
  - The Relational Database was invented by Edger. F. Codd at IBM in 1970.
  - A DBMS that is based on relational model and stores data in the form of related tables.

  E.g. Oracle, Microsoft SQL Server, Sybase SQL Server, IBM's DB2

DBMS manages interaction between end users and database.



DBMS Benefits

- Controlling Redundancy In Data
- Restricting Unauthorized Access
- Providing Persistent Storage for Program Objects
- Storage Structures for Efficient Query Processing
- Multiple User Interfaces
- Representing Complex Relationships among data
- Enforce Integrity Constraints
- Permitting Interfacing And Actions Using Rules
- Potential For Enforcing Standards
- Reduce Application Development Time
- Flexibility in change the structure of objects.
- Availability of Up-to-Date Information
- Economies of Scale
- Providing Backup and Recovery

DBMS Benefits

- Controlling Redundancy : Redundancy can be reduced.
  - E.g. Students Data with Admin Level, Department Level, & Printing Dept.
- Restricting Unauthorized Access
  - Each user must follow authentication process
  - Each user have the specific rights (Authorization) to access the resources.
- Providing Persistent Storage for Program Objects
  - Objects like (Table, Procedure, Functions etc.) are stored in flash or non-volatile memory to protect against the power failure.
- Providing Storage Structures for Efficient Query Processing :
  - Indexes : It is used to quickly locate data without having to search every row in a database table. First column is index field & second column is pointer to the physical storage of the record.
  - Optimization : To determine the most efficient way to execute a given query by considering the possible query plans. The parameters like CPU Time, No. of Input / Output, No. of joins used for optimization.
- Providing Backup and Recovery
  - Export utility for full or partial back up of data stored in database.
  - Import utility recover the data in database.

DBMS Benefits

- Provides Multiple User Interfaces :
  - The data should be available to users as per their needs.
  - E.g. Employee (Id, Name, DoB, Email, Contact, Basic, HRA, DA, Salary, Designation, Department)
  - Finance Department Needs : Id, Name, Dept., Desi., Basic, HRA, DA, Salary
  - HR Department Needs : Id, Name, DoB, Email, Contact
  - Project Manager Needs Id, Name, Department & Designation
- Representing Complex Relationships among data
  - Data stored in a database is connected with each other and a relationship is made in between data. It provides efficient and accurate data.
- Enforce Integrity Constraints :
  - Data type : Number, Char, Date, Varchar2 etc.
  - Field Length : Length of field according to data placed in it.
  - Primary Key : To uniquely identify each row in a table.
  - Foreign Key : To reference the value from other table.
  - Not NULL : Value for the field is compulsory
  - Check Constraints : Restricted values are stored in field of a table.

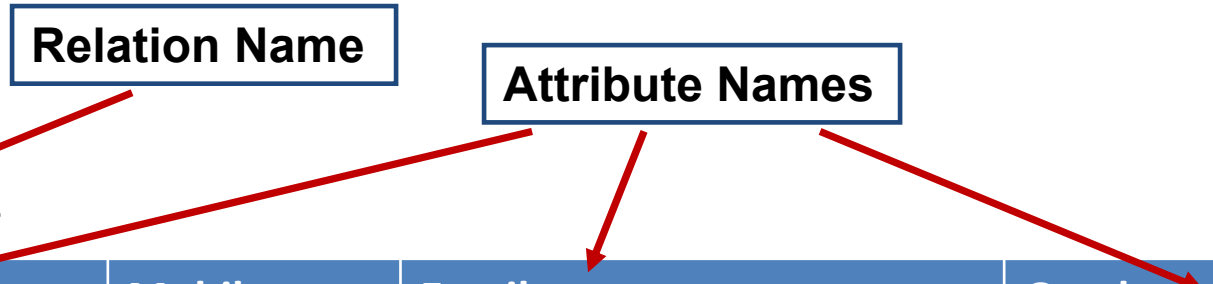
DBMS Benefits

- Permitting Interfacing And Actions Using Rules
  - Triggers, Stored Procedure & Functions
- Potential For Enforcing Standards
  - DBA Define & Enforce Standards Like
    - Names and Formats of data element
    - Report Structures
- Reduce Application Development Time
  - Once the database up & running, Development time is less.
- Flexibility
  - Change the structure of database.
  - Easily add a table and extend it in database.
- Availability of Up-to-Date Information
  - As soon as one users update is available to the database, all the other users of the database can immediately see the update.
  - E.g. Ticket Reservation, Hotel Booking, Bank Passbook etc.
- Economies of Scale :
  - Large scale business invest in more powerful processors, storage devices, communication lines.
  - E.g. Email, Social Media, Super Market, Flipkart, Amazon etc.
  - Reduce overall cost of operation and management.

Relation & Attribute

Relation : It is a table with name. It is set of tuples or rows.

- E.g.



Students

Id	Name	Mobile	Email	Gender
1	Shyam	9089787678	Shyam_parmar@yahoo.com	Male
2	Kalpna	9456453423	Kalpna_shah@gmail.com	Female
3	Abhay	8907898765	Abhay_rathod@yahoo.com	Male

- **Schema of the relation** is Student (Id, Name, Mobile, Email, Gender)

Attribute

- Each attribute of a relation has a name
- Attribute values are **atomic**; that is, indivisible

Domain of an Attribute

The domain of an attribute A, denoted by Dom(A)

It is the set of values that the attribute can take.

Examples of domains:

- Mobile_numbers: The set of ten-digit phone numbers.
- Aadhar Card Numbers: The set of valid twelve-digit numbers.
- Person Name: The set of characters known as string

Datatype of an Attribute

- Domain is represented by a data type.
- For example
  - Mobile_ number — Number(10)
  - Employee_age — Number(3)
  - Department_Name — Char(50)
  - Birth_Date — Date
  - Product_Id — Number(3)
  - Author_Name — Char(50)
  - Joining_Date — Date

Relation Schema & Relation Instance

- A **relation schema R**, denoted by **R(A1, A2, ... , An)** is made up of a relation name R and a list of attributes A1, A2, ..., An.

  E.g. Student (Id, Name, Mobile, Email, Gender)

- A **relation instance** (state) of a relation schema R(A1, …, An), denoted by r(R), is a set of tuples in the table of R at some instance of time.

E.g. r (Student)

Id	Name	Mobile	Email	Gender
1	Shyam	9089787678	Shyam_parmar@yahoo.com	Male
2	Kalpna	9456453423	Kalpna_shah@gmail.com	Female

Schema & Instance Update

- ☛A schema may have different states at different times.

- The **schema of a relation** may change (e.g., adding, deleting, renaming attributes and deleting a table schema) but it is infrequent

- The **state of a relation** may also change (e.g., inserting or deleting a tuple, and changing an attribute value in a tuple) & it is frequent

Relational Model

- It represents the database as a collection of relations.
- Each relation is a table of values **or**
- Each relation is a file of records

- **In the relational model,**
  - A row is called a tuple
  - A column header is called an attribute
  - A table is called a relation.
  - The data type in each column is represented by a domain of possible values.

Relational Model

- It is the primary data model used for data storage and processing.

- **History of Relational Model**

- 1970 E.F. Codd proposed relational model

- Early microcomputer based DBMSs - dBase, R:base, Paradox

- Most popular enterprise DBMSs, all relational
  - Oracle, DB2, Informix, Sybase
  - Microsoft's SQL Server
  - MySQL, PostgreSQL

Difference DBMS & RDBMS

DBMS	RDBMS
•DBMS stands for Database Management System.	RDBMS stands for Relational Database Management System.
Data is stored as file.	Data is stored as tables.
DBMS deals with small quantity of data.	RDBMS deals with large quantity of data.
DBMS supports single user at a time.	RDBMS supports multiple users at a time.
E.g. MSAccess.	E.g. Oracle , SQL Server etc.
•No much Secure	More Secure : Logging at O/S level Command Level & Object level
•Dose not Support Distributed Databases	Support Distributed Databases
• Relationship between two tables or files maintained by programmed.	Relationship between two tables or files can be specified at the time of table creation.

Dr Edgar F. Codd's Rules for RDBMS

1. Information Rule :

   All data should be presented in table form

2. Guaranteed Access Rule

   All data should be accessible without ambiguity.

3. Systematic Treatment of NULL Values

   A field should be allowed to remain empty.

4. Dynamic Online Catalog (System tables)

   A relational database must provide access to its structure through the same tool that are used to access the data.

5. Comprehensive Data Sub-Language Rule

   The database must support at least one language that includes functionality for data definition, data manipulation, data integrity and database transaction control.

6. View Updating Rule

   Data can be presented in different logical combinations.

Dr Edgar F. Codd's Rules for RDBMS

7. High-Level Insert, Update, and Delete Rule

   Data can be retrieved in sets constructed of data from multiple rows and/or multiple tables.

8. Physical Data Independence

   The user is isolated from the physical method of storing and retrieving.

9. Logical Data Independence

   View of data should not be changed when the logical structure (table's structure) of the database changes.

10. Integrity Independence

    The database language (like SQL) should support constrains on user input that maintain database integrity.

11. Distribution Independence

    A user should be unaware of whether database is distributed or not.

12. Non-Subversion Rule

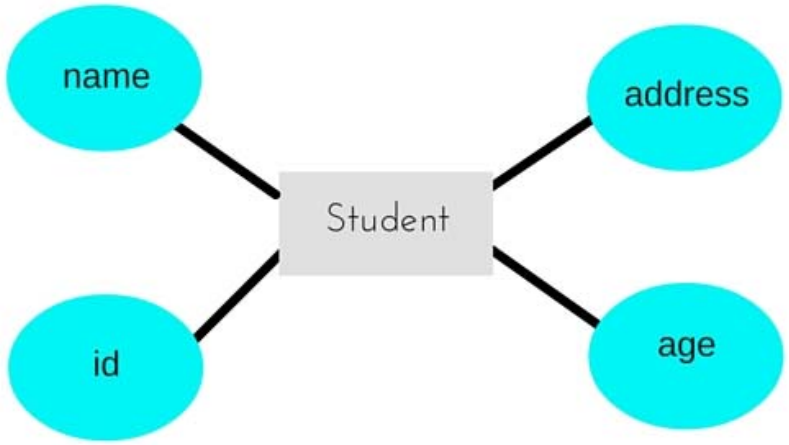
    There should be no way to modify the database structure other than database language (like SQL).

The Entity Relationship Model

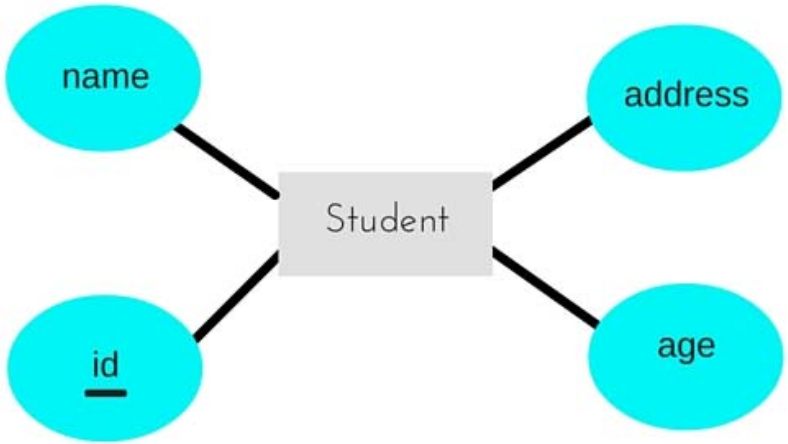
- Introduced by Peter Chen in 1976
- Entity relationship diagram (ERD)
  - Uses graphic representations to model database components
  - Entity is mapped to a relational table
- ER diagrams are created based on three basic concepts:
  - Entities,
  - Attributes and
  - Relationships
- Connectivity labels types of relationships
  - Diamond connected to related entities through a relationship line

Notations for ERD

- **Attribute:** An **Attribute** describes a property or characteristic of an entity. For example, Name, Age, Address etc can be attributes of a Student. An attribute is represented using eclipse.



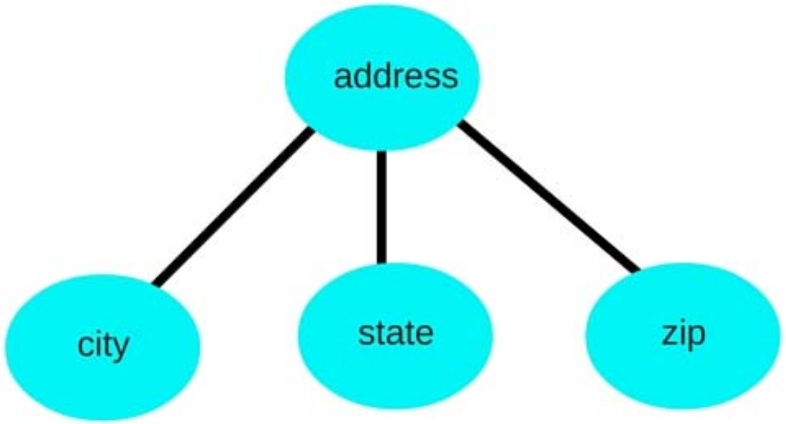
- **Key Attribute:** Key attribute represents the main characteristic of an Entity. It is used to represent Primary key. Ellipse with underlying lines represent Key Attribute.



- **Simple Attribute:** It can not be meaningfully subdivided further.
  - E.g. Emp_No, Birthdate , Salary

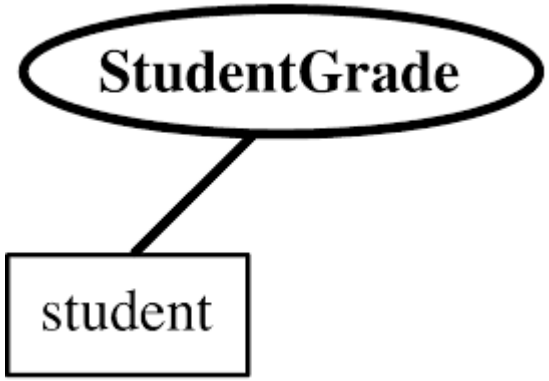
Composite Attribute: An attribute can be subdivided into composite parts.

- E.g. Address (City, State, Zip) , Name (Fname, Mname, Lname)

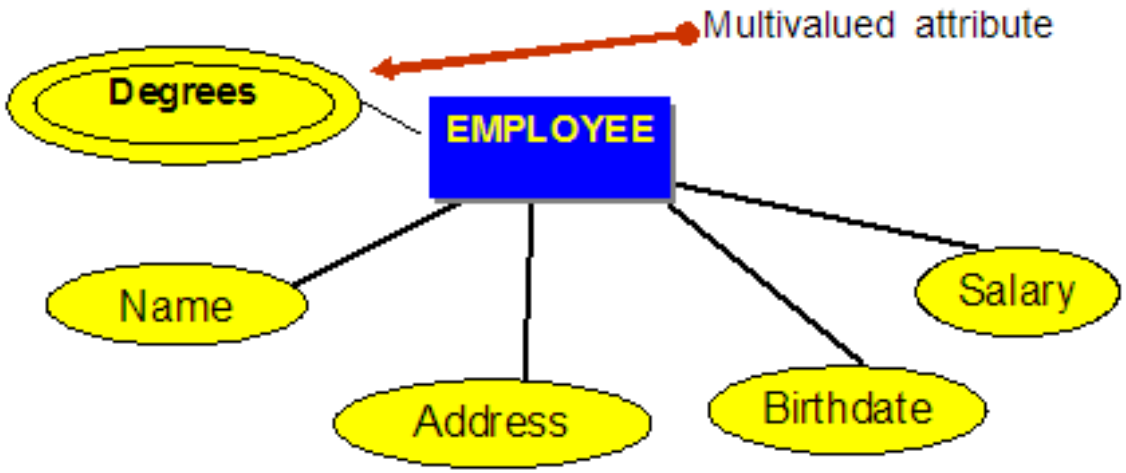


Single-value Attribute :

A field of a single-value attribute can contain only a single value



Multivalue Attribute : A field of a **multivalued attribute** can contain either a single value or multiple values.

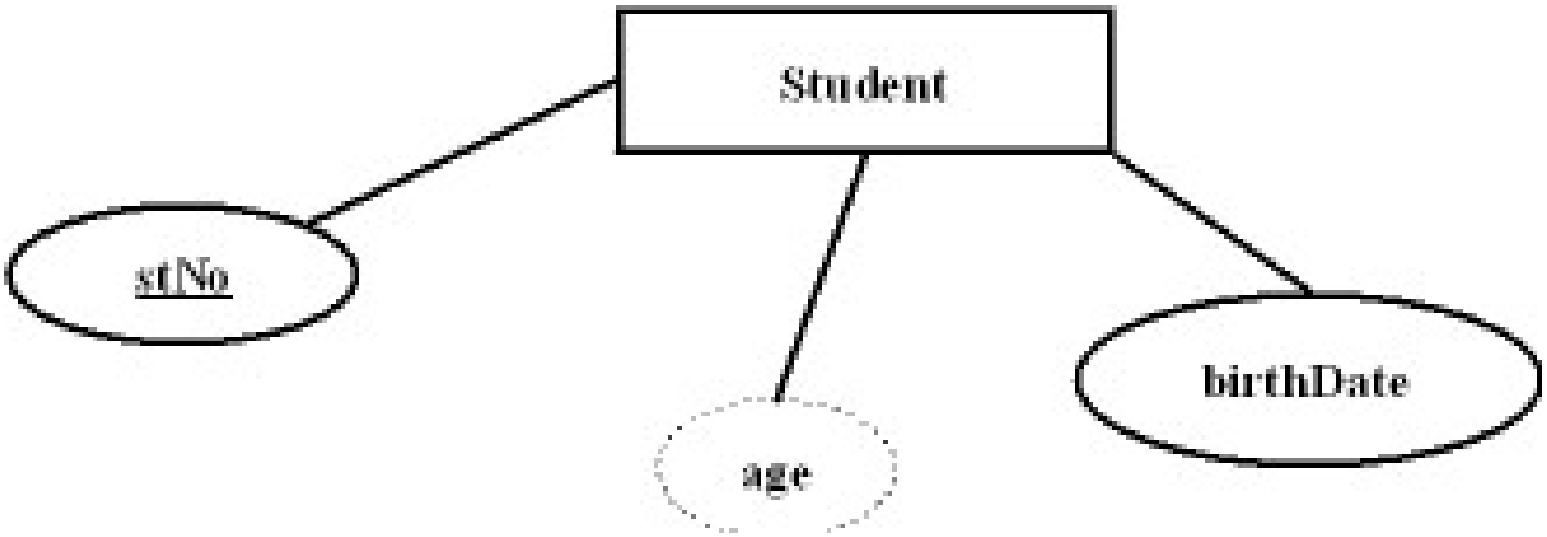


Stored Attributes:

Attribute that is used to derive other attribute. E.g. Birthdate

Derived Attributes:

An attribute of an entity is derived from other attribute. E.g. Age



Relationship: A Relationship describes relations between **entities**. Relationship is represented using diamonds.

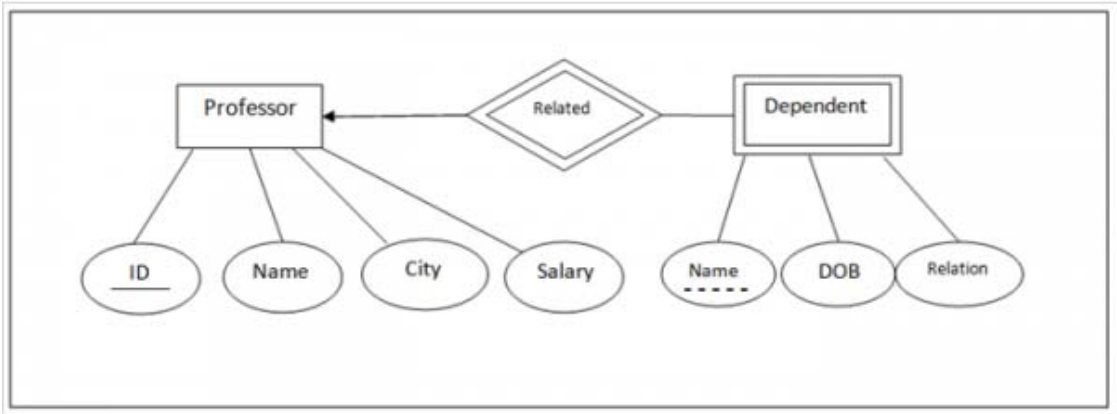
Strong Entity :

An entity that has a primary key is called as Strong entity set.
Strong Entity is represented by a single rectangle

Weak Entity :

- The weak entity do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.
- Weak Entity is represented by double rectangle

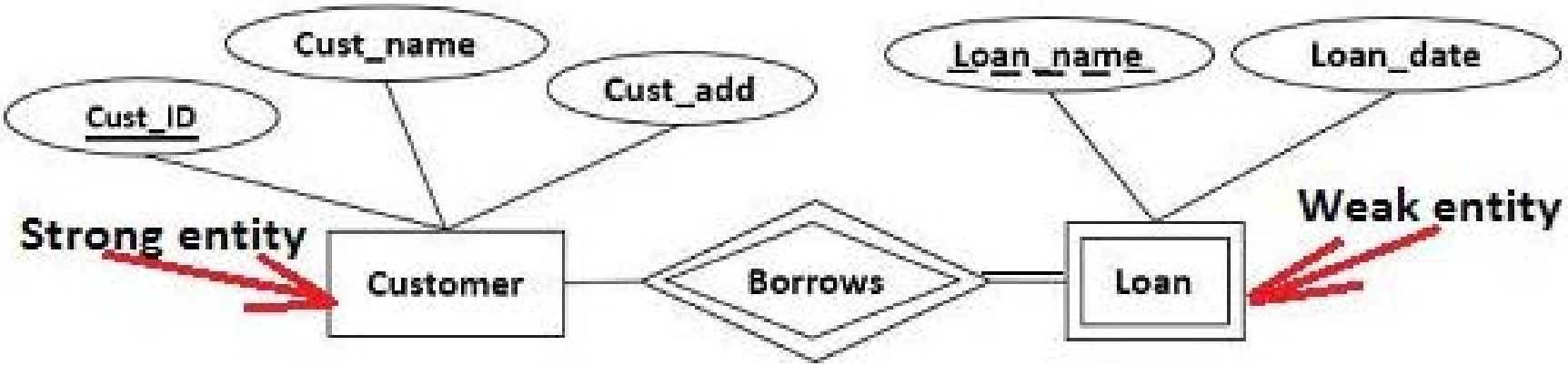
E.g.



•The Strong Entity is Professor, whereas Dependent is a Weak Entity.

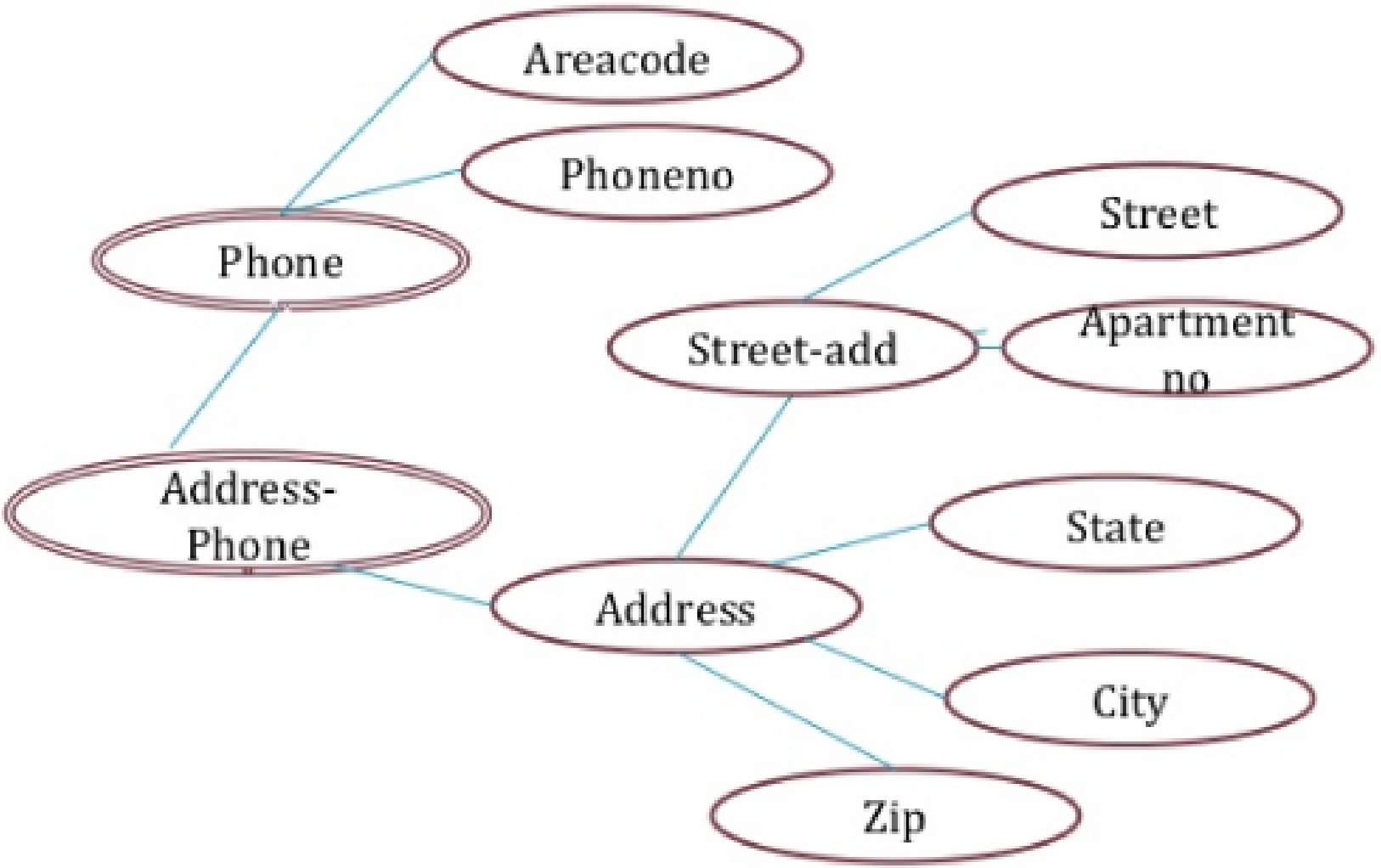
•ID is the primary key (represented with a line) and Name in Dependent entity is called Partial Key (represented with a dotted line).

Strong and Weak Entity Example

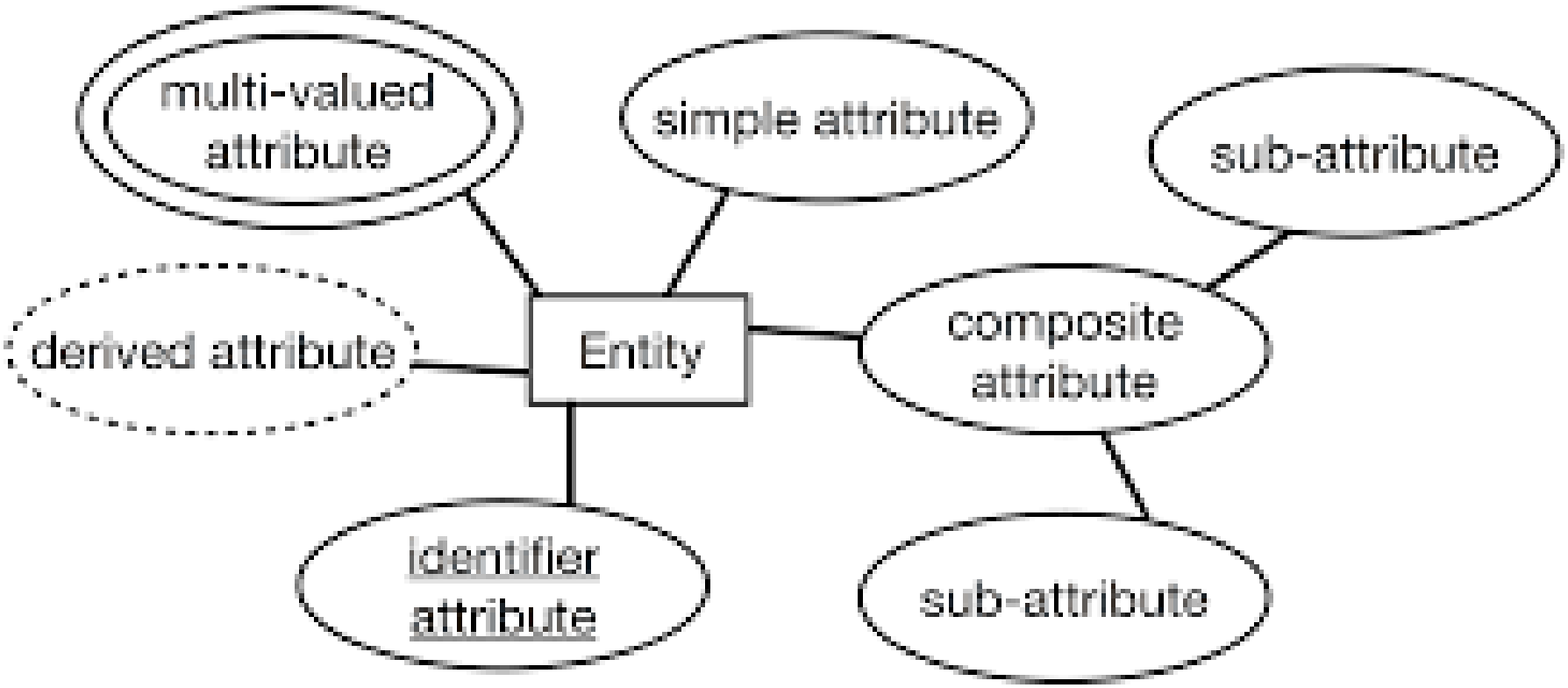


- COMPLEX ATTRIBUTE

Composite and mutivalued attribute can also be nested arbitrarily to form complex key.



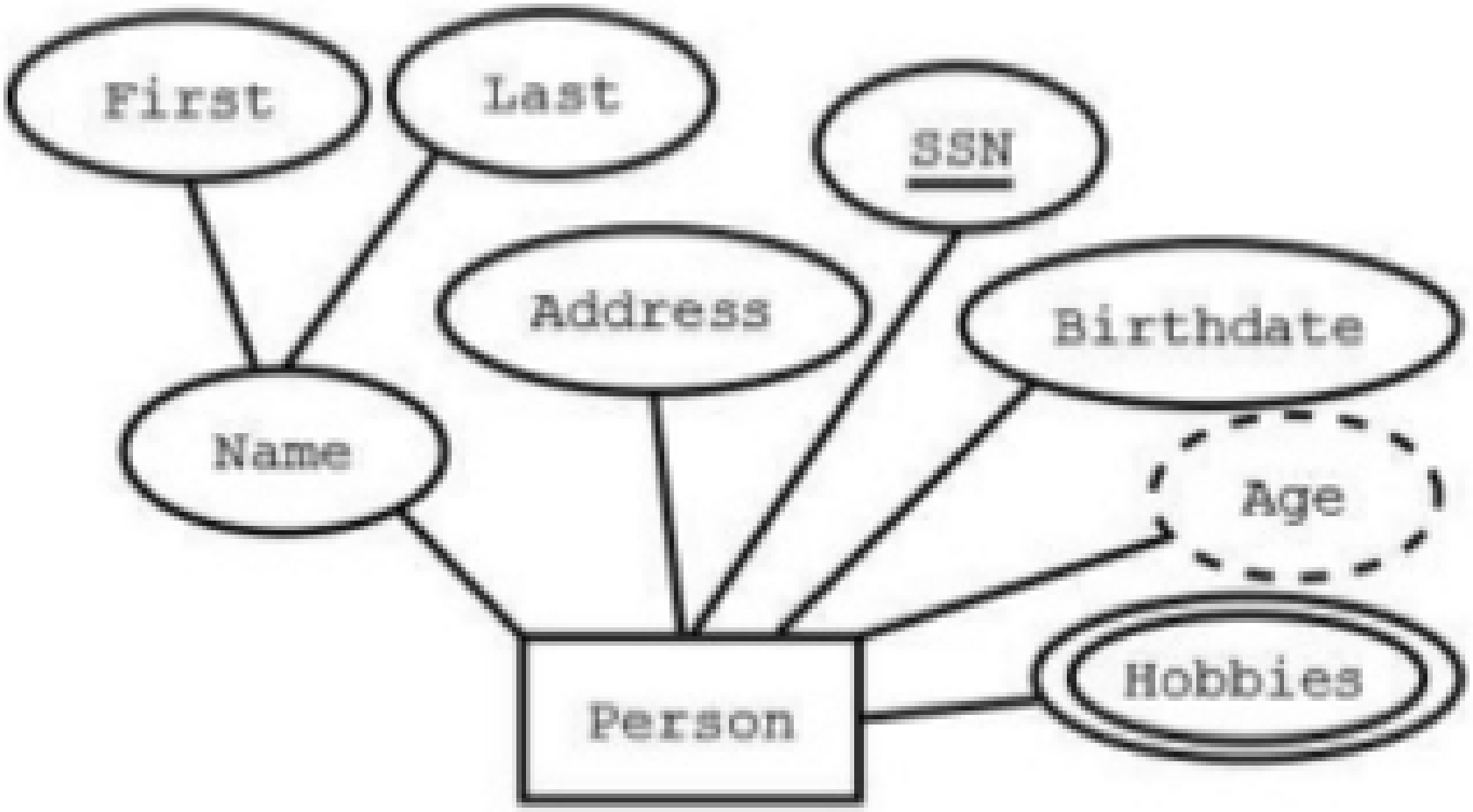
Overview of E-R Notation



Attribute Type

Types of Attributes	Definition	Example
Simple attribute	Cannot be divided into simpler components	**Gender** of the employee
Composite attribute	Can be split into components	**Date of joining** of the employee
Single valued	Can take on only a single value for each entity instance	**Age** of the employee
Multi-valued	Can take up many values	**Skill set** of the employee
Stored Attribute	Attribute that need to be stored permanently	**Date of joining** of the employee
Derived Attribute	Attribute that can be calculated based on other attributes.	**Years of service** of the employee

Example : Types of Attributes



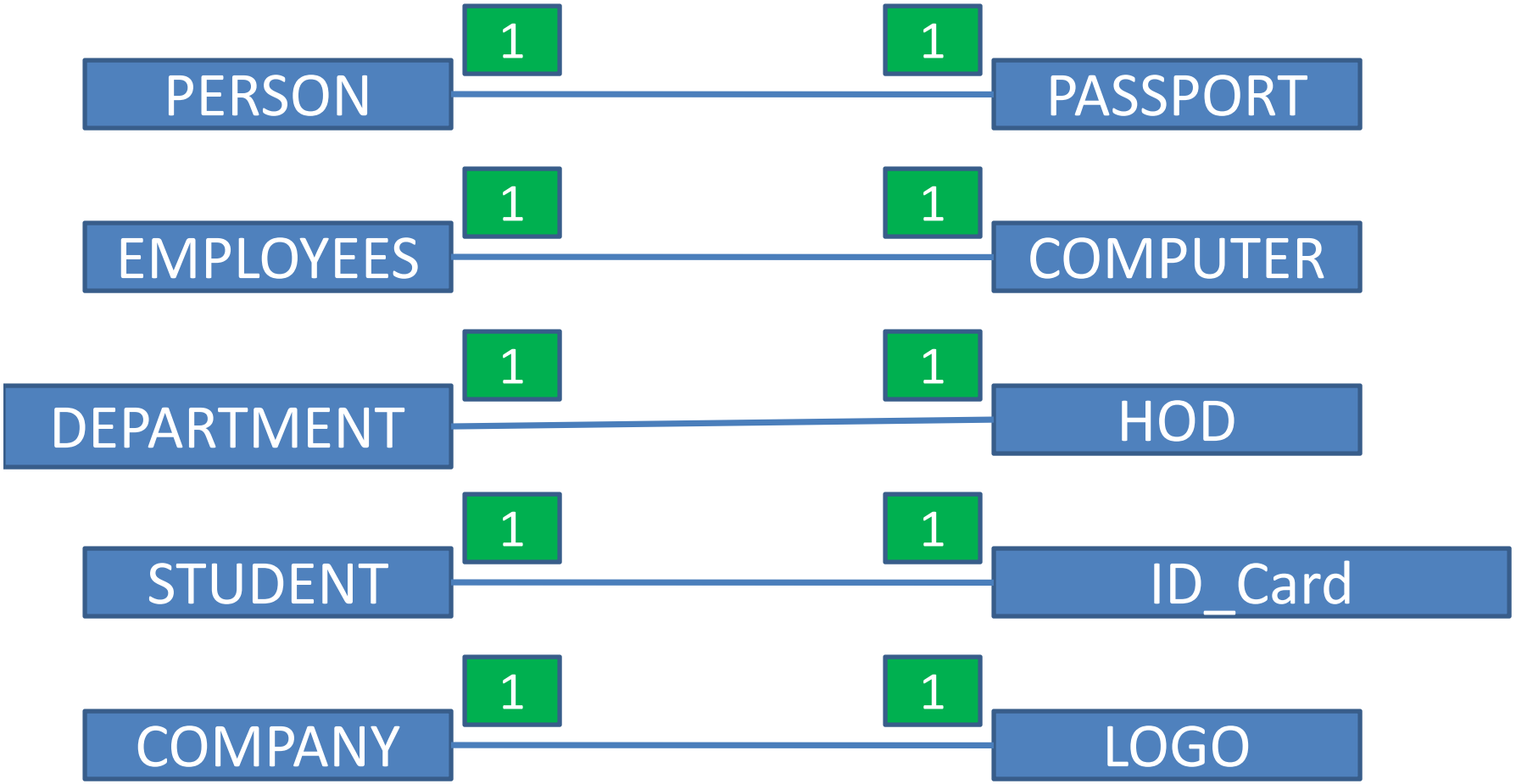
Relationship in ER Diagram

- A **relationship** type represents the association between entity types.
- In **ER diagram**, **relationship** type is represented by a diamond and connecting the entities with lines.
- E.g. **'Enrolled in'** is a **relationship** type that exists between entity type Student and Course.
- Relationship may be a
  - **One to One (1:1)**
  - **One to Many (1:M) or**
  - **Many to Many (M:N)**.
- A relationships connectivity is represented by a **1**, **M** or **N** next to the related entity.

Relationship in ER Diagram

- **One to One (1:1)**
  - When only one instance of an entity is associated with the relationship, it is marked as '1:1'.
  - Let us assume that a person have only one License and a license indicate only one person. So the relationship is one to one.

Examples of One-to-One Relationship



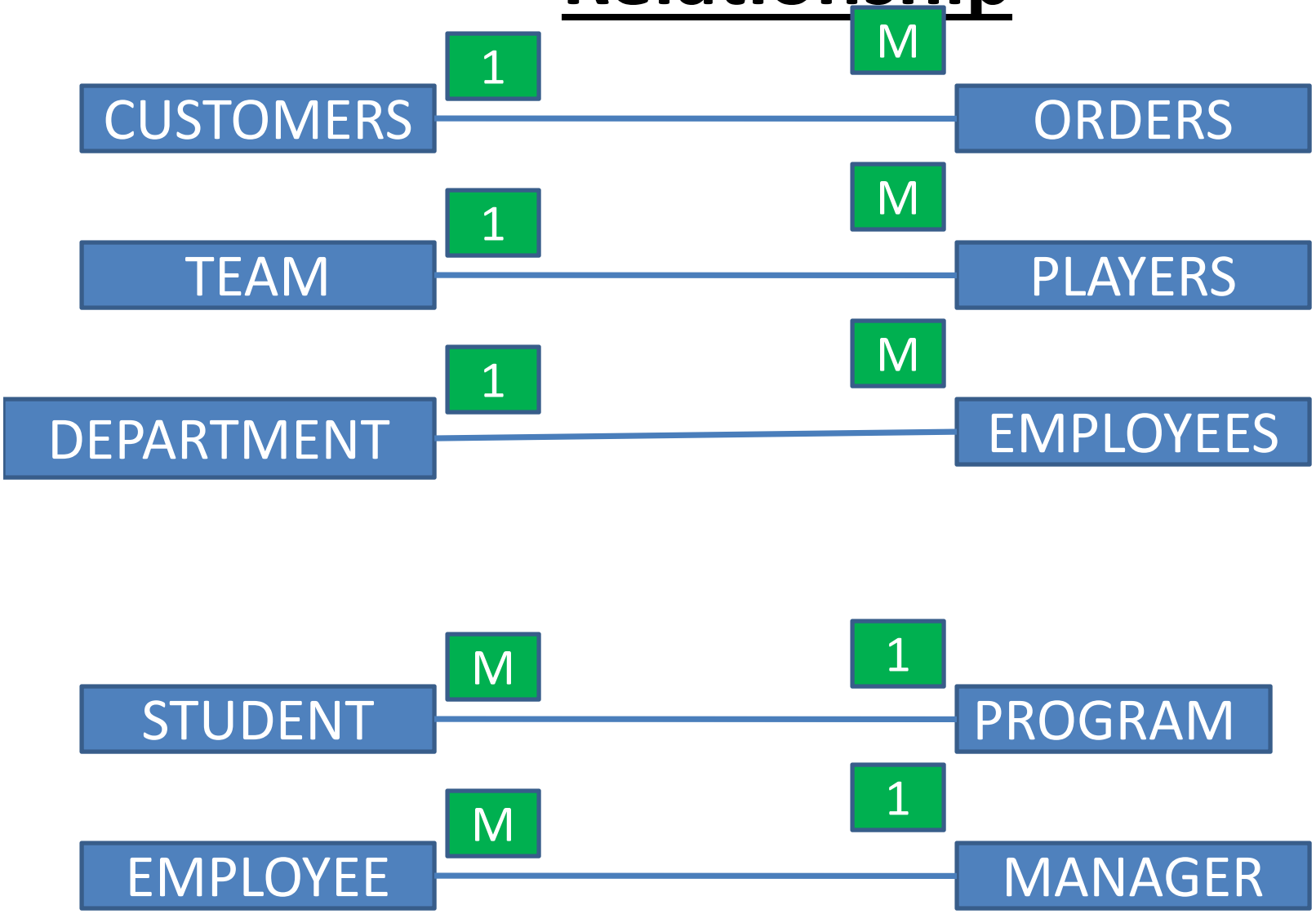
Relationship in ER Diagram

One to Many (1:M) OR Many to One (M:1)

- **One to Many (1:M)**
  - When more than one instance of an entity is associated with a relationship, it is marked as '1:N'.
  - The following example reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship.

- **Many to One (M:1)**
  - When more than one instance of an entity is associated with a relationship, it is marked as '1:N'.
  - The following example reflects that only one instance of entity on the right and more than one instance of an entity on the left can be associated with the relationship.

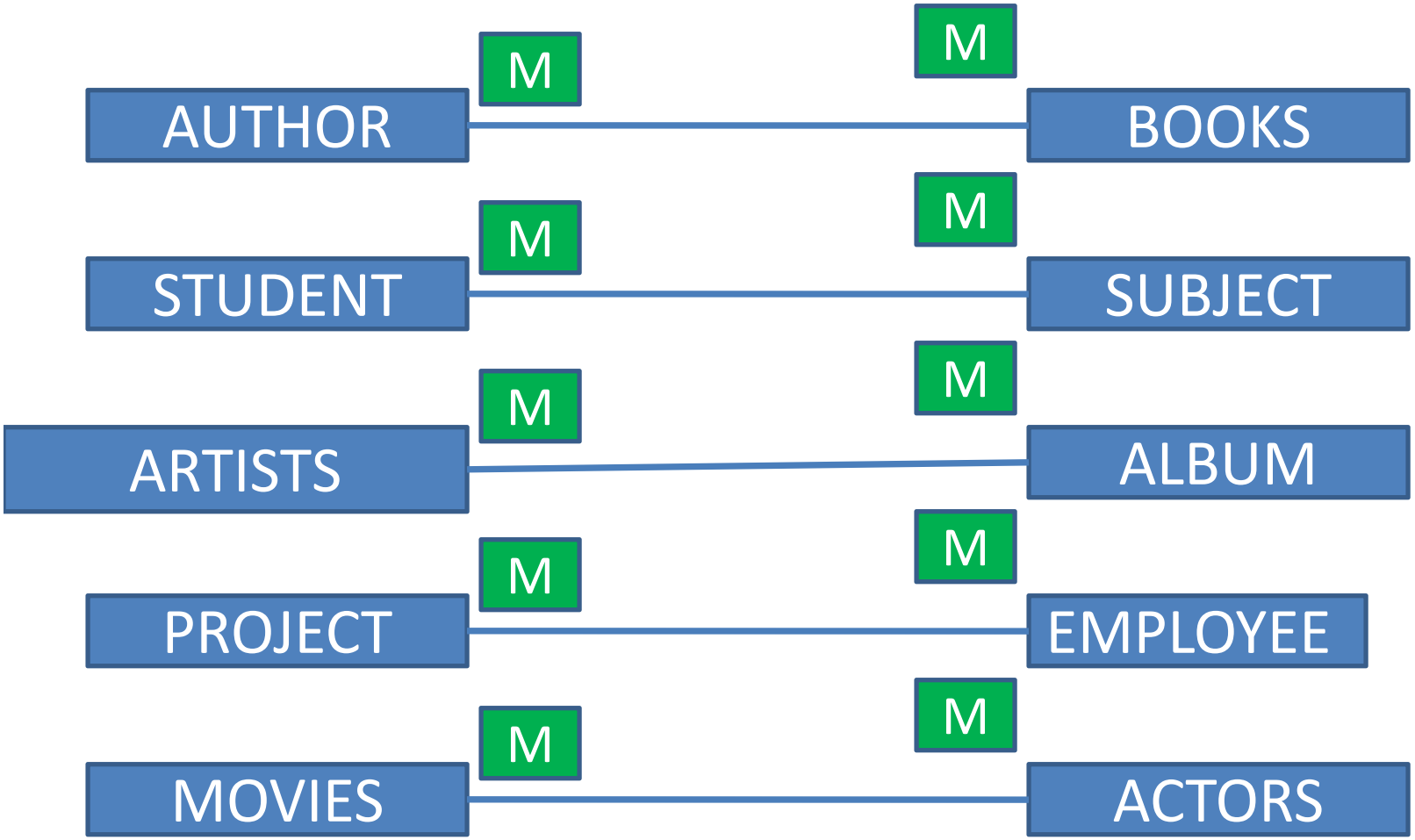
Examples of One-to-Many & Many-to-One Relationship



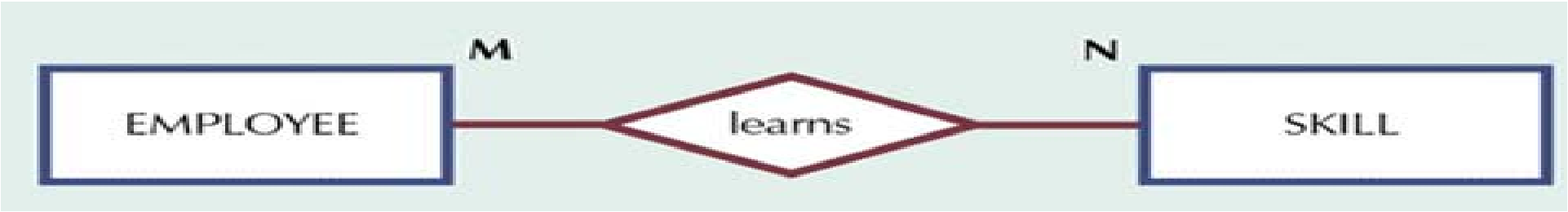
Relationship in ER Diagram

- **Many to Many (M:N)**
  - More than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship.

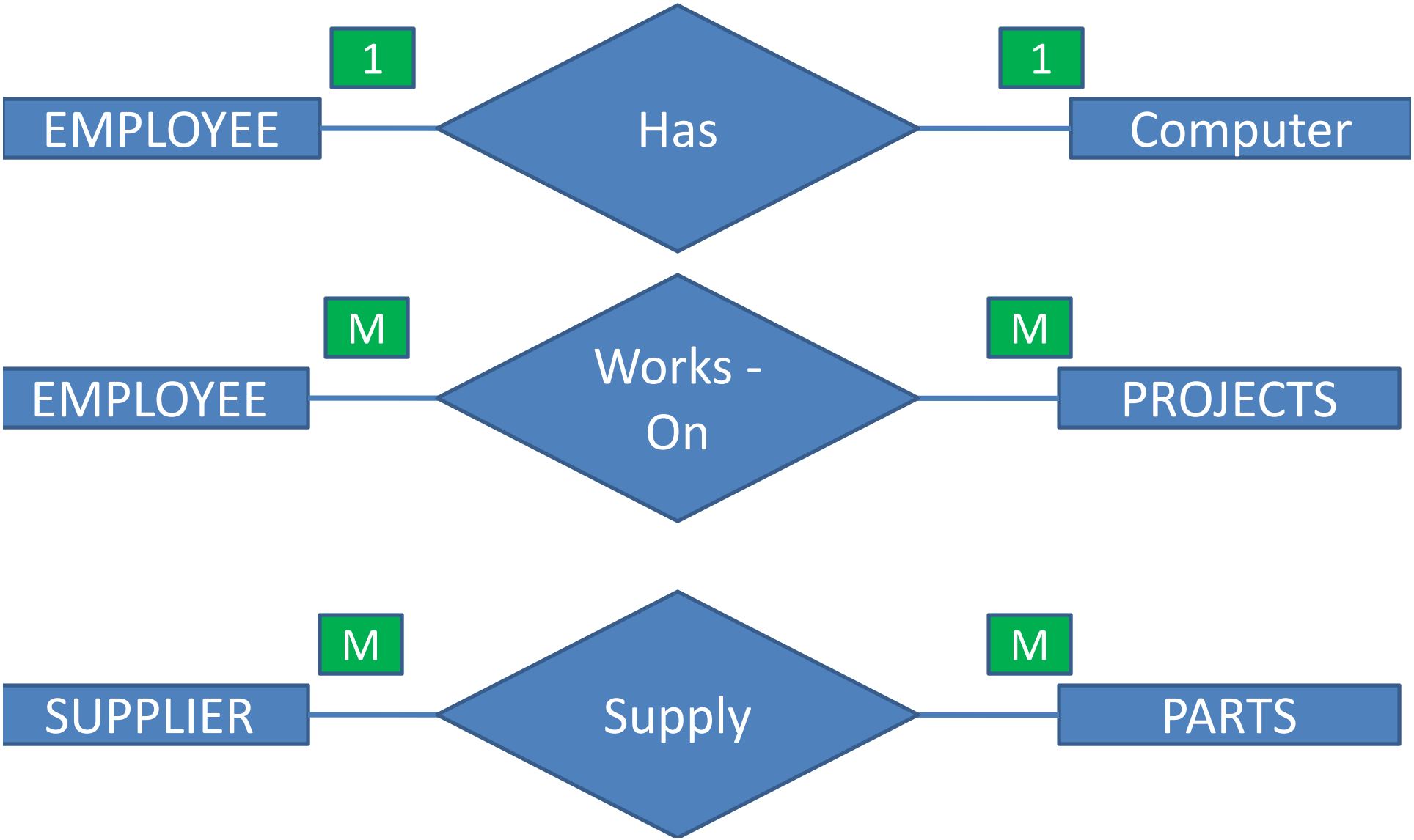
Examples of Many-to-Many Relationship



Examples of Relationship

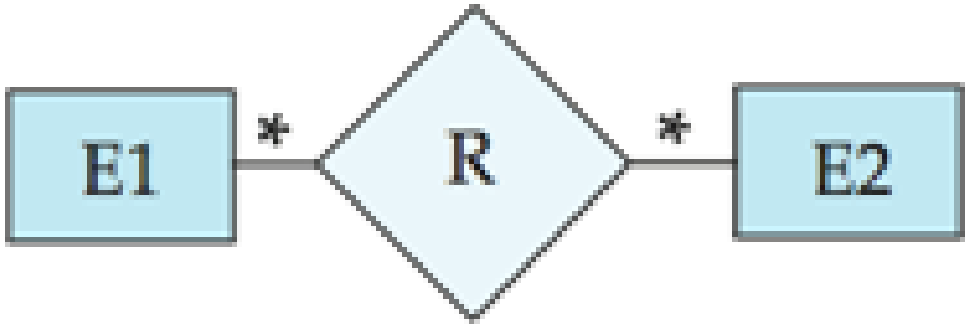


Examples of Relationship

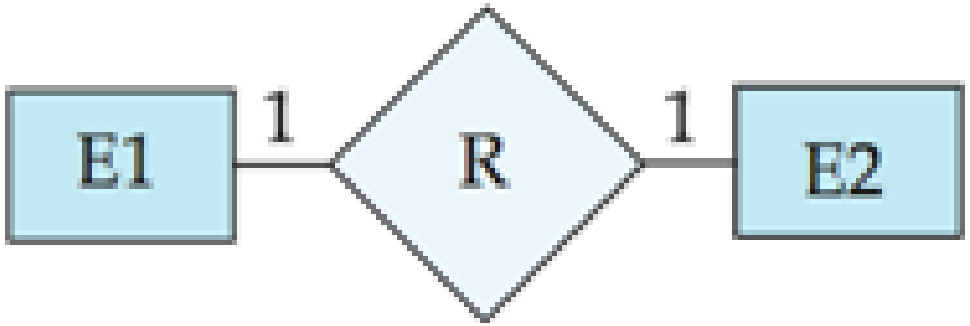


Alternative ER Notation

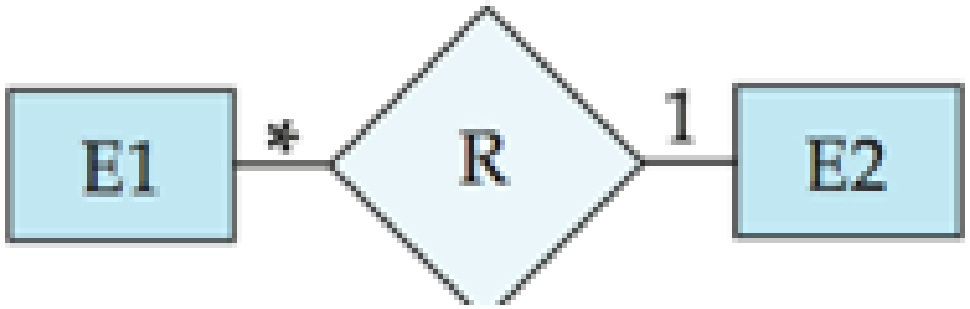
many-to-many relationship



one-to-one relationship

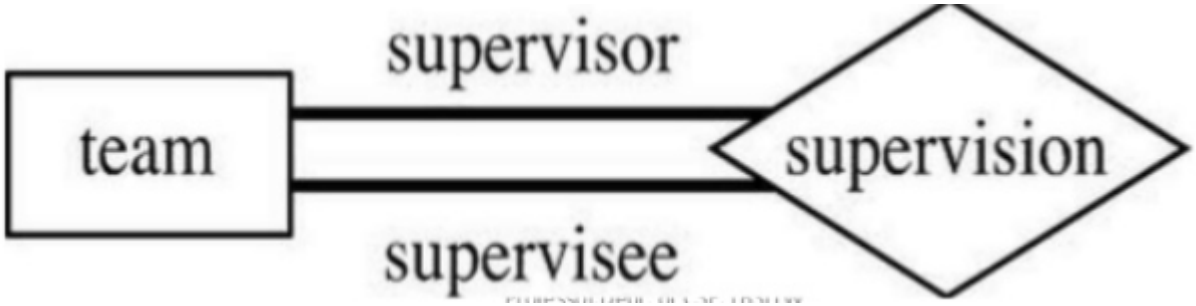


many-to-one relationship

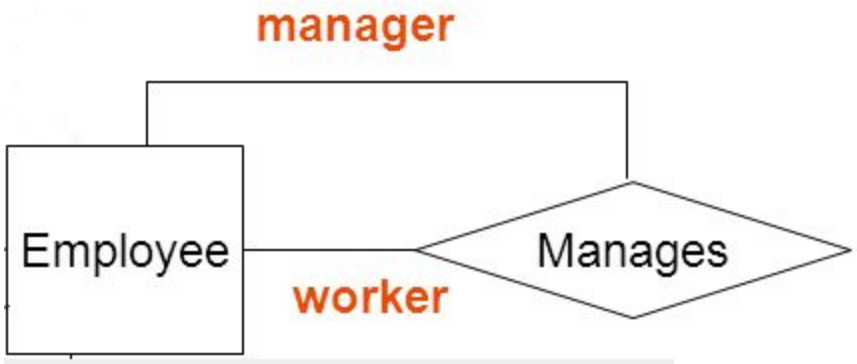


Recursive Relationships and Role Names

- **Recursive Relationships**
  - Relationship is between different instances of the same entity. An entity set relating to itself.
- **Role Names**
  - Specify the exact role in which the entity participates in the relationships.
- **E.g.** An employee can supervise multiple employees. Hence, this is a recursive relationship of entity employee with itself. This is a 1 to many recursive relationship as one employee supervises many employees.



- **E.g.** Manager manages worker. Here Manager and Worker are both Employee of the organization.



Steps to create an ER Diagram:

1. Entity Identification
2. Relationship Identification
3. Cardinality Identification
4. Identify attributes of each entity

E.g. In a university, a Student enrolls in Courses (Subjects). A student must be assigned to at least one or more Courses (Subjects). Each course (Subject) is taught by a single Professor. To maintain instruction quality, a Professor can deliver only one course.

ER Diagram

1. **Entity Identification**

Example have three entities :

Student

Course &

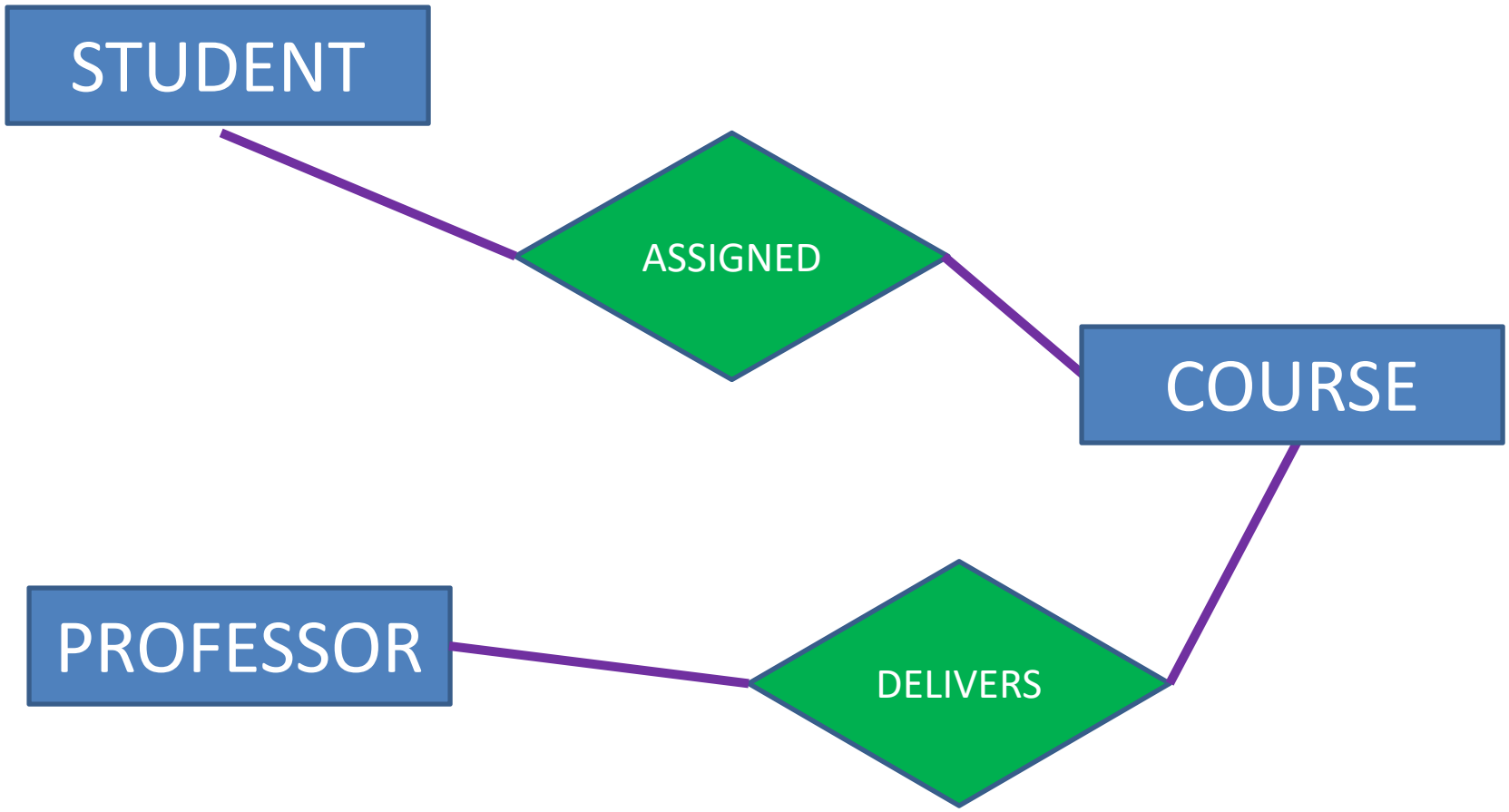
Professor

2. Relationship Identification

Example have two relationships:

The Student assigned in course

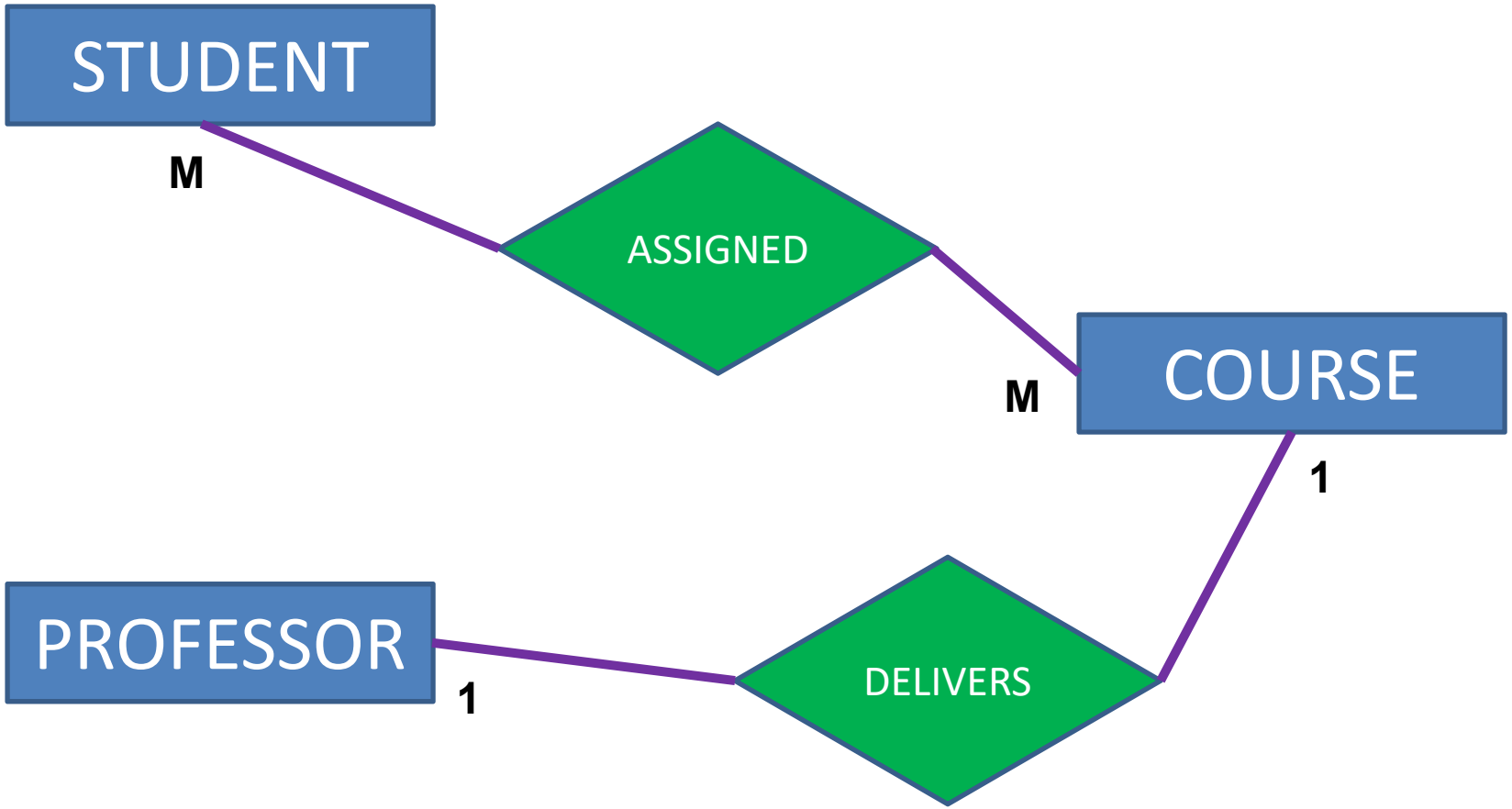
Professor delivers a course



3. Cardinality Identification

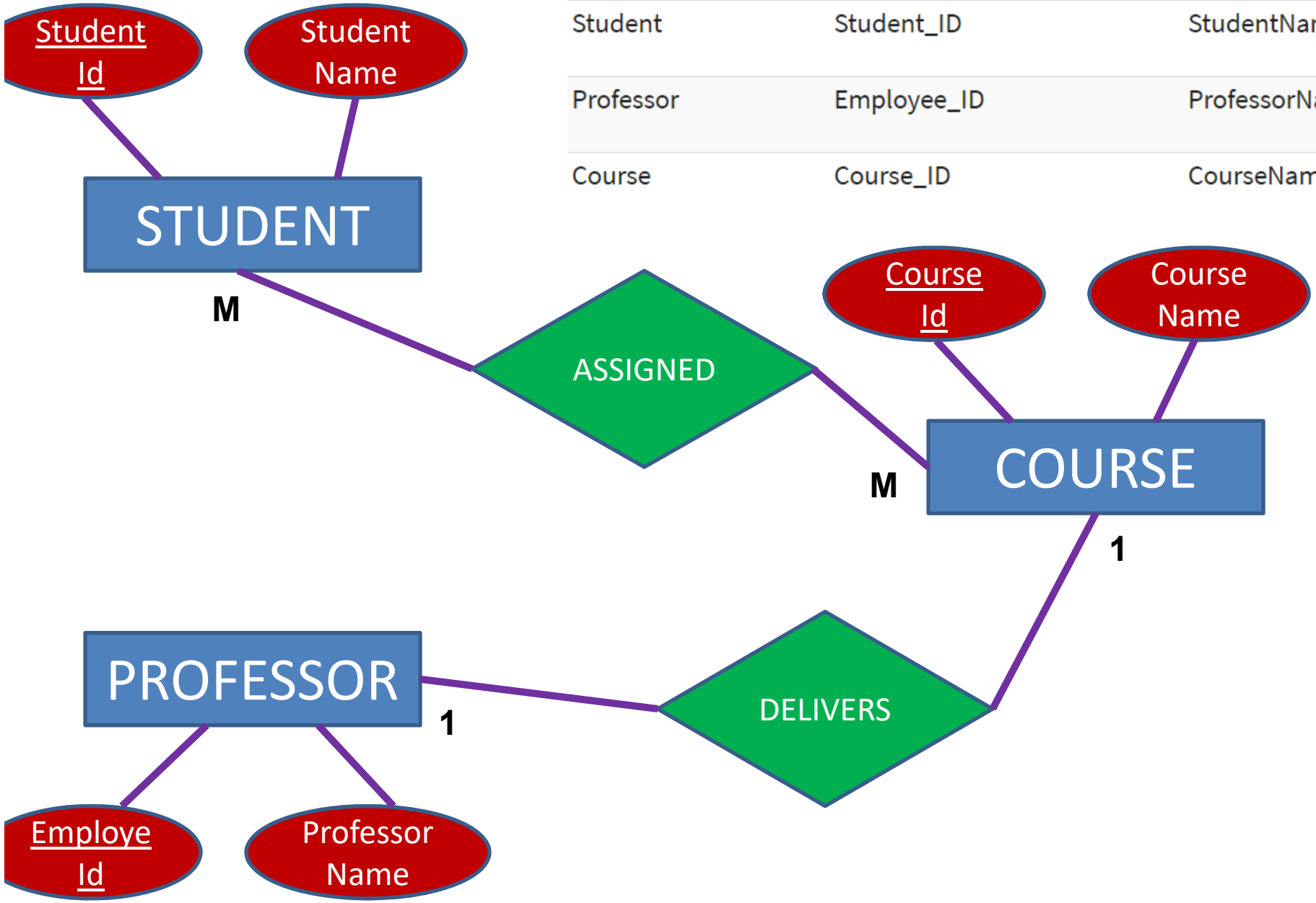
A student can be assigned multiple courses

A Professor can deliver only one course



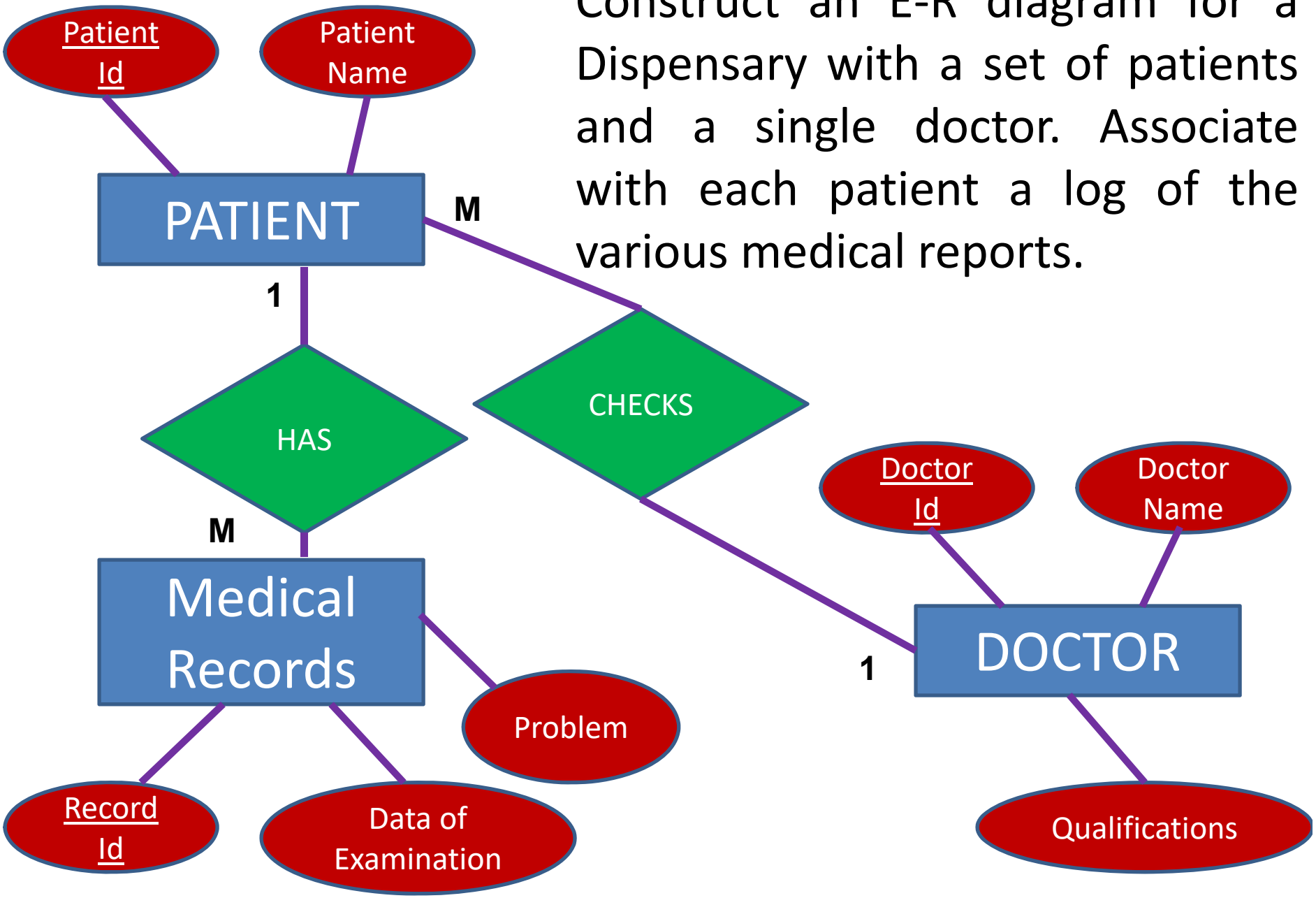
4. Identify Attributes

Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



ER Diagram

Construct an E-R diagram for a Dispensary with a set of patients and a single doctor. Associate with each patient a log of the various medical reports.



ER Model Conversion to Relations

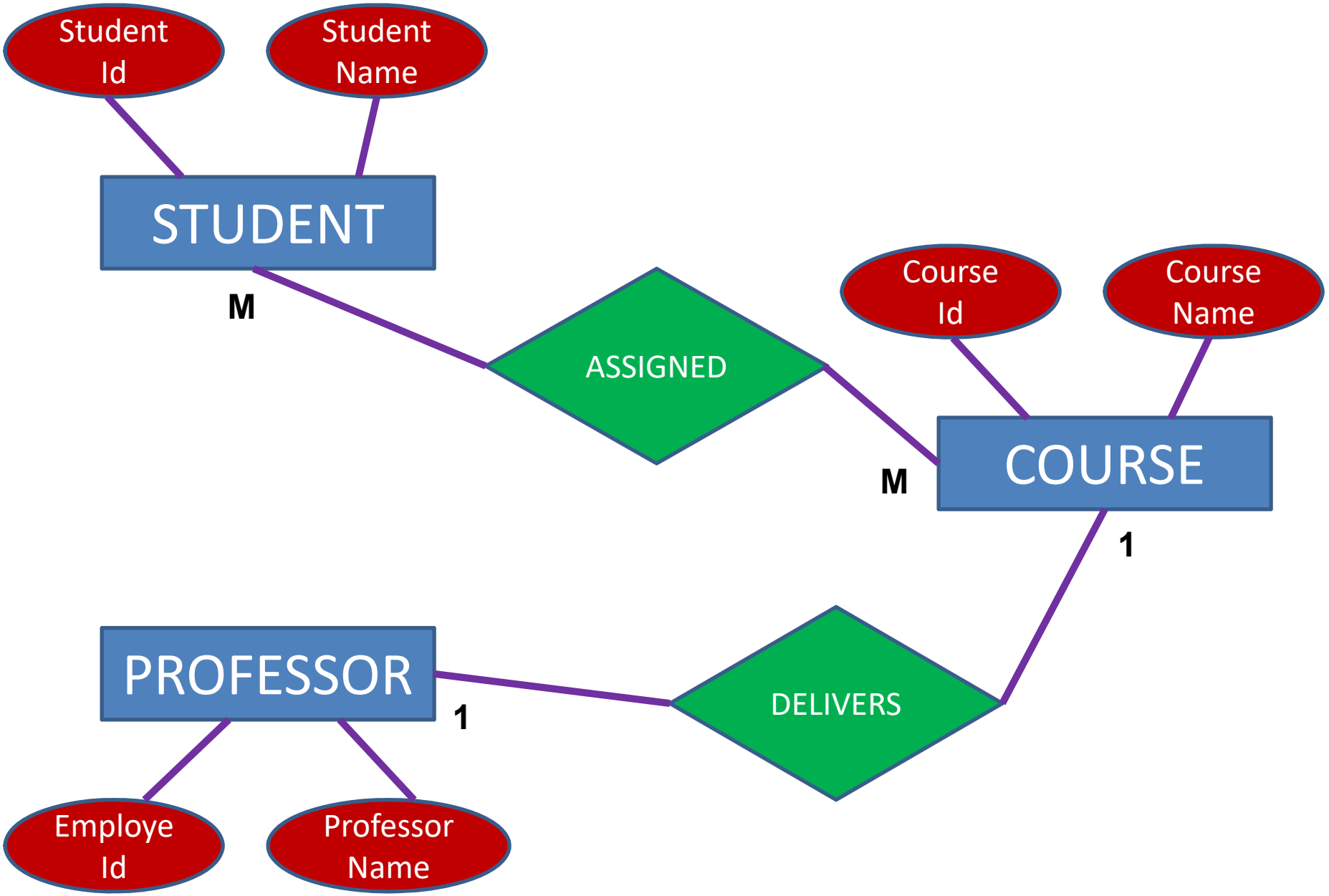
- **Steps:**

1. Create table for each entity.

2. Entity's attributes should become fields of tables with their respective data types. Also declare primary key.

3. Create table for a relationship. Add the primary keys of all participating Entities as fields of table with their respective data types.

Example : ER Diagram To Relation



ER Model Conversion to Relations

1. Student table corresponds to Student Entity with key as Student-Id.
Similarly **Course table** corresponds to Course Entity with key as Course_Id and **Professor table** corresponds to Professor Entity with key as Employee_Id.

2. Student table **attributes** are Student_Id and StudentName. Student_id is uniquely identify each student, so student_id becomes primary key. And student table is as below.

Student (**Student_id** NUMBER (3), **StudentName** CHAR (50),
PRIMARY KEY (Student_Id)

Similarly

Course (**Course_id** NUMBER (3) , **CourseName** CHAR (50) ,
PRIMARY KEY (Course_Id) **&**

Professor (**Employee_id** NUMBER (3) , **ProfessorName** CHAR (50) ,
PRIMARY KEY (Course_Id)

ER Model Conversion to Relations

3. Stud_Course table represents relationship between Student and Course (Which student assigned which Course). So it will take attribute Student_Id from Student and Course_Id from Course.

Stud_Course (**Student_Id** NUMBER (3) , **Course_Id** NUMBER(50) ,
PRIMARY KEY (Student_Id, Course_Id)

Similarly

Prof_Course table represents relationship between Professor and Course (Which Professor Delivers which Course). So it will take attribute Employee_Id from Professor and Course_Id from Course.

Prof_Course (**Employee_Id** NUMBER (3) , **Course_Id** NUMBER(50) ,
PRIMARY KEY (Employee_Id, Course_Id)

ER Model Conversion to Relations

RELATIONS (TABLES)

Student

Stud_Id	NUMBER(3)
StudName	CHAR(100)

Course

Course_ID	NUMBER(2)
CourseName	CHAR(50)

Professor

Employee_Id	NUMBER(3)
ProfessorName	CHAR(100)

Stud_Course

Student_Id	NUMBER (3)
Course_Id	NUMBER(50)

Prof_Course

Employee_Id	NUMBER (3)
Course_Id	NUMBER(50)

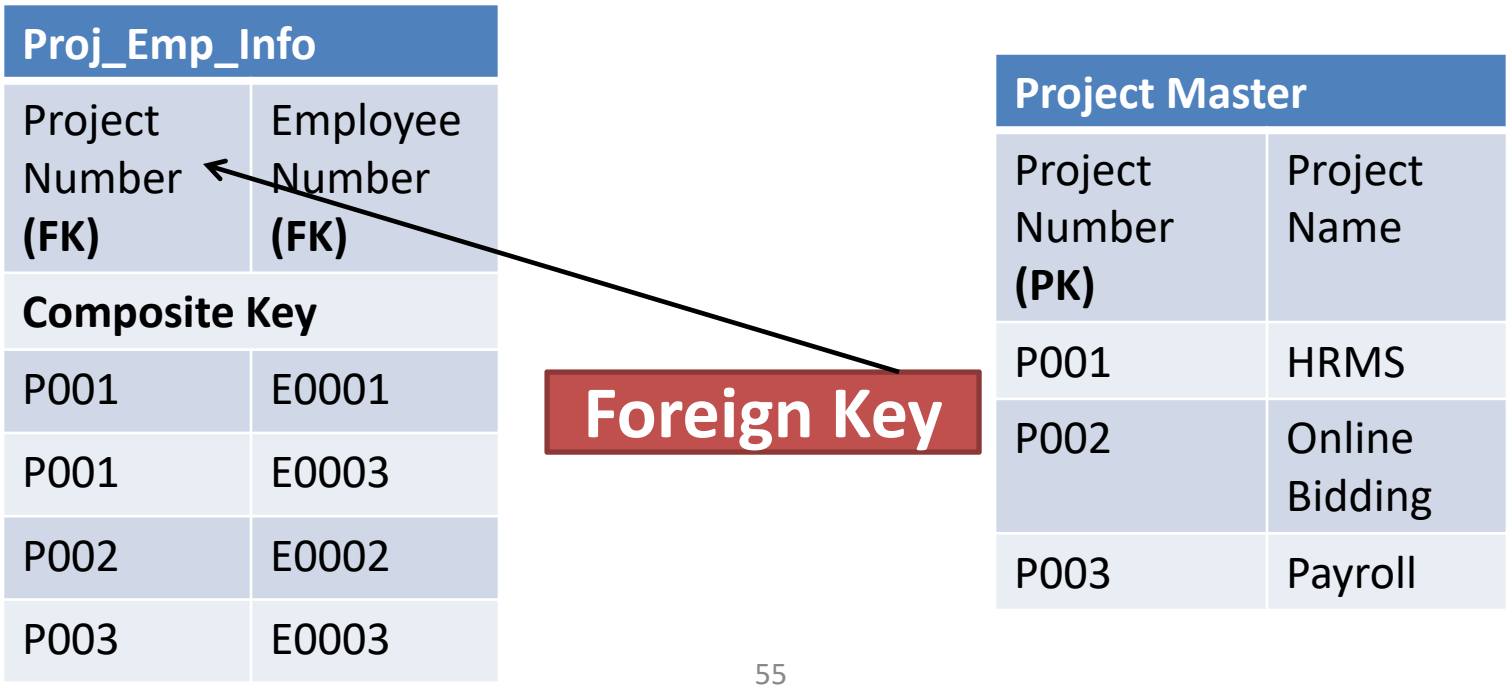
Primary Key

Project Master	
Project Number **(PK)**	Project Name
P001	HRMS
P002	Online Bidding
P003	Payroll

Employee Master		
Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer

Foreign Key (Referential Integrity)

- **Foreign key** must match actual values and data types with the related **Primary Key.**



Foreign Key & Composite Primary Key

Composite Key : A primary key composed of one or more columns

Project Master	
Project Number **(PK)**	Project Name
P001	HRMS
P002	Online Bidding
P003	Payroll

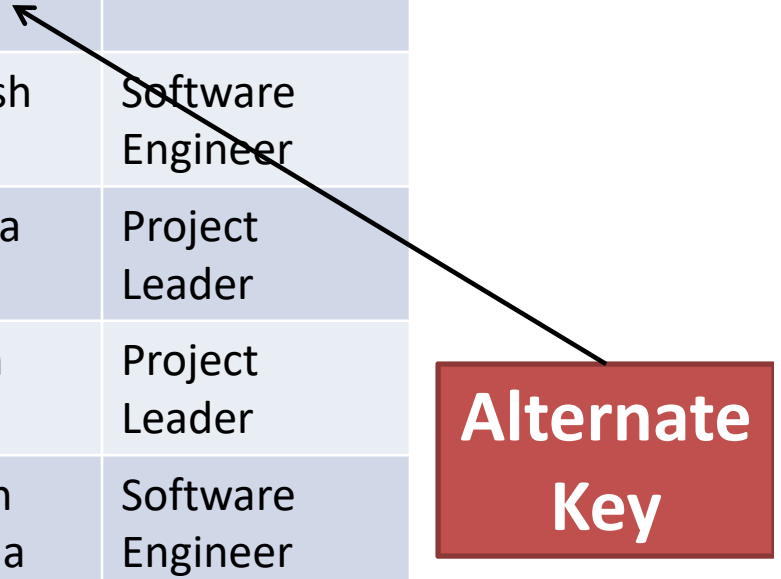
Employee Master		
Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer

Proj_Emp_Info	
Project Number **(FK)**	Employee Number **(FK)**
Composite Key	
P001	E0001
P001	E0003
P002	E0002
P002	E0004
P002	E0005
P003	E0001

Alternate Key

- A **key** associated with one or more columns whose values uniquely identify every row in the table, but which is not the primary **key**.
- **E.g.** Employee Master having Employee Name as Alternate key but it is not primary key.

Employee Master		
Employee Number **(PK)**	Employee Name	Designation
E0001	Ramesh Shah	Software Engineer
E0002	Krishna Patel	Project Leader
E0003	Jayesh Vyas	Project Leader
E0004	Mitesh Sharma	Software Engineer
E0005	Vishal Shah	Software Engineer



Primary Key & Foreign Key

- A **Primary key** is a key that uniquely identifies a row in each table.
- It is denoted with its first two letters, namely, **PK**.
- Employee table is uniquely identified by Employee_No field than it is called a primary key of a table.

- A **Foreign key** is a key borrowed from another related table (that's why its foreign) in order to make the relationship between two tables.
- It is normally denoted with its first two letters, namely, **FK**.
- E.g. Bill table contains the product_id of the product. In Product table product_id is primary key. In Bill table product_id is foreign key.

Users Associated with Database System

- For small database like address book one person may define, construct and manipulates it and there is no sharing.
- For large database many peoples are involved in design, construct and manipulates.
- **Types of Users**
  - Database Administrators
  - Database Designers
  - End Users
  - Software Engineers
    - (System Analysts and Application Programmers)

Role of DBA User

- Database Administrator
  - To Oversee and manage database and database softwares
  - Authorizing access to database
  - Coordinating and Monitoring its use
  - Purchase new Software and Hardware
  - Solution of security issues
  - Track system response time

Role of Database Designer

- Database Designer
  - Interact with each potential group of users and develop views of the database.
  - Each view is analyzed and integrated with the views of other user groups.
  - Choosing appropriate structure to store and retrieve the data

Role of End Users

- End Users
  - Access the database for querying, updating and generating reports
  - Casual end users (Manager Level)
    - Occasionally access the database.
    - Need different information each time.
  - Naïve or Parametric end users
    - Constantly querying and updating database
      - Bank Cashier
      - Reservation at ticket counter
      - Clerks of shipping companies
  - Sophisticated end users (Engineer, Scientist, Analyst)
    - Familiar with the facilities of DBMS to meet their complex requirements
  - Standalone users
    - Maintain personal database by using ready made program packages with menu and graphical interfaces

Role of Software Engineer

- Software Engineer
  - System analyst
    - Determine end user requirements and develop specifications that meets these requirements
  - Application Programmers
    - Implement these specification as application programs
    - Test, Debug, Document and maintain it

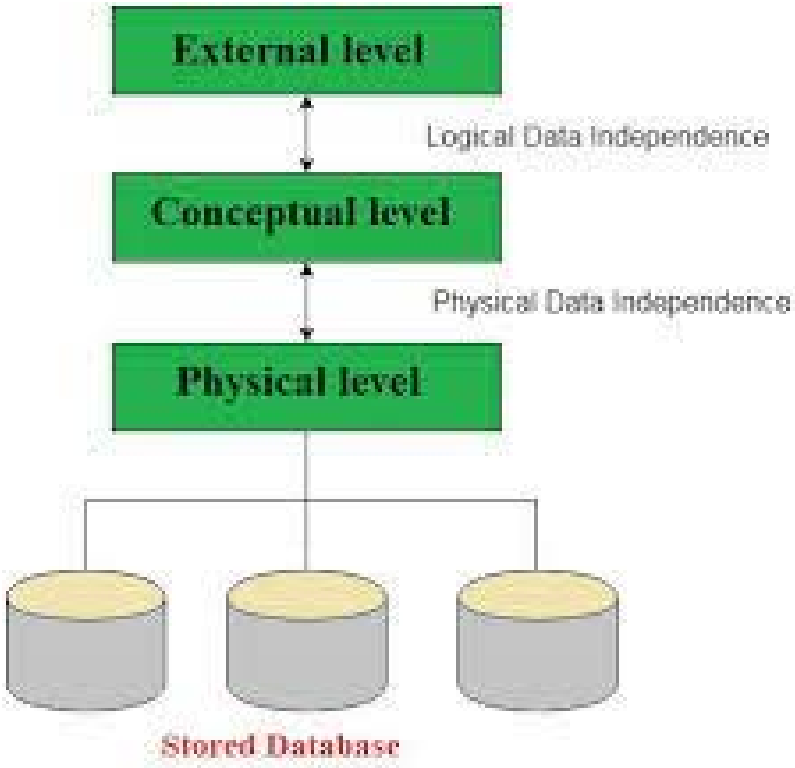
SYSTEM CATALOG / DATA DICTIONARY

- Why System Catalog?
  - As database having lots of objects (like tables, views, constraints, indexes, users etc.)
    - It is very difficult for any user or designer to remember all information.
    - For new designer of the database, it is necessary to understand the system thoroughly.

- What is System Catalog?
  - “It is a data about data. It contains complete information about the database objects (like tables, views , constraints, indexes, users etc.) ”.
  - It is automatically inserted / updated / deleted by the RDBMS.

SYSTEM CATALOG / DATA DICTIONARY

- What is Metadata?
  - Every database stores following details about an objects. (Objects may be table, view, index etc.)
    - Structure,
    - Definition,
    - Purpose,
    - Storage,
    - Number of columns and rows (records)
    - Dependencies
    - Access rights
    - Owners etc.



- Changes in Logical schema may include
  - Addition or Removal of attributes
  - Addition or Removal of relationship
- Changes in physical schema may include
  - Using new storage device
  - Change the data files
  - Modifying indexes

- Data Independence
  - Lower-level schema changes will not effect upper-level schemas and applications
- Physical Data Independence
  - if physical schema changes do not affect application programs.
- Logical Data Independence
  - if logical schema changes do not affect application programs.

Data Sharing

- Without Data Sharing
  - File based system : Same data in multiple files.
  - E.g.
    - Student data with Admin Section for fees collection
    - Student data with Faculty for attendance
    - Student data with Exam section for Result

- Data Sharing
  - Three types of data sharing
    - Sharing data between functional units
    - Sharing data between different levels of users
    - Sharing data between different location.

Sharing Data Between Functional Units

- The data sharing suggests that people in different functional areas use common pool of data, each of their own applications.
- Without Data Sharing
  - Admission Section Department & Exam section may have their own data files.
  - Each group benefits only from its own data.
- With Data Sharing
  - Combined data are more valuable
  - Access rights for data

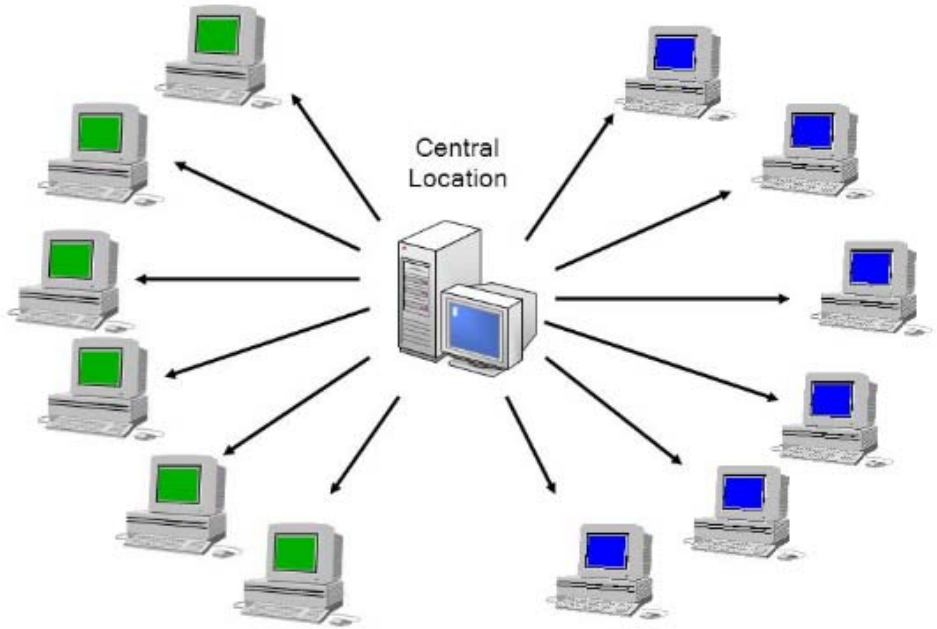
Sharing data between different levels of users

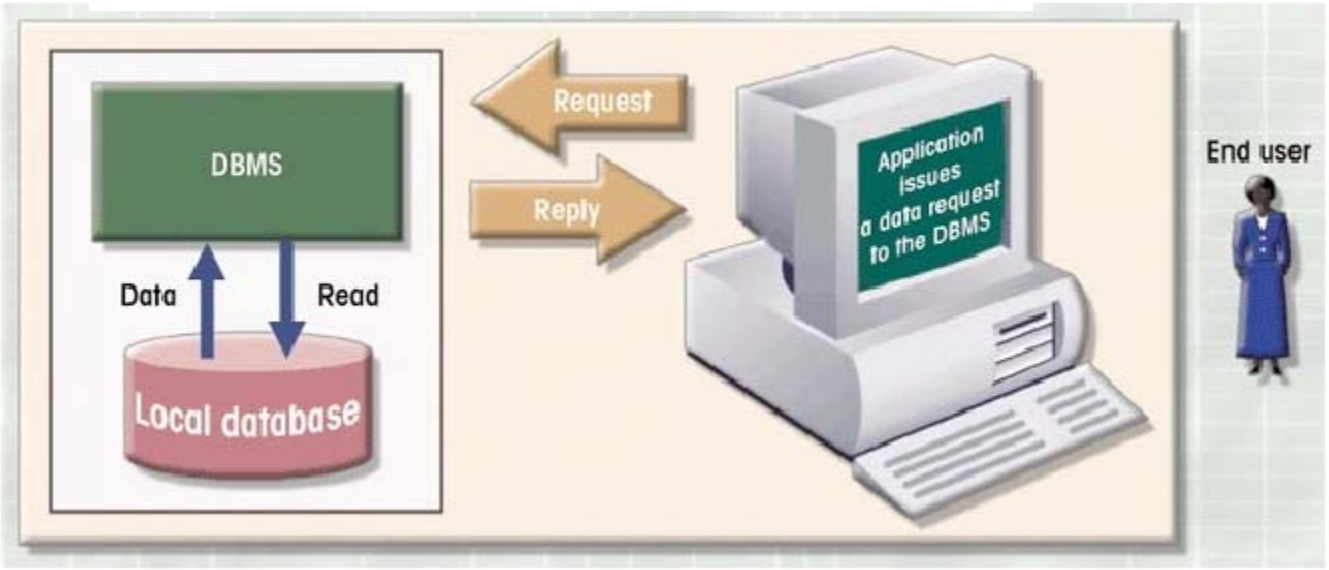
- Three different levels of users :
  - Operations
  - Middle management and
  - Executive
- Level of Users : Operations
  - Its basic characteristics include:
    - A focus on data, storage and flows at the operational level.
    - Efficient transaction processing.
- Level of Users : Middle Management
    - Inquiry and Report Generation
    - Integration and Planning of functionality
- Level of Users : Executive
    - Managers or Decision makers
    - Strategic reports & Quick response

Sharing data between different locations

- A company with several locations has important data distributed over a wide geographical area.
- Two ways of data sharing
  - A Centralized Database
  - A Distributed Database
- A **centralized database**
  - It is physically confined to a single location, controlled by a single computer.

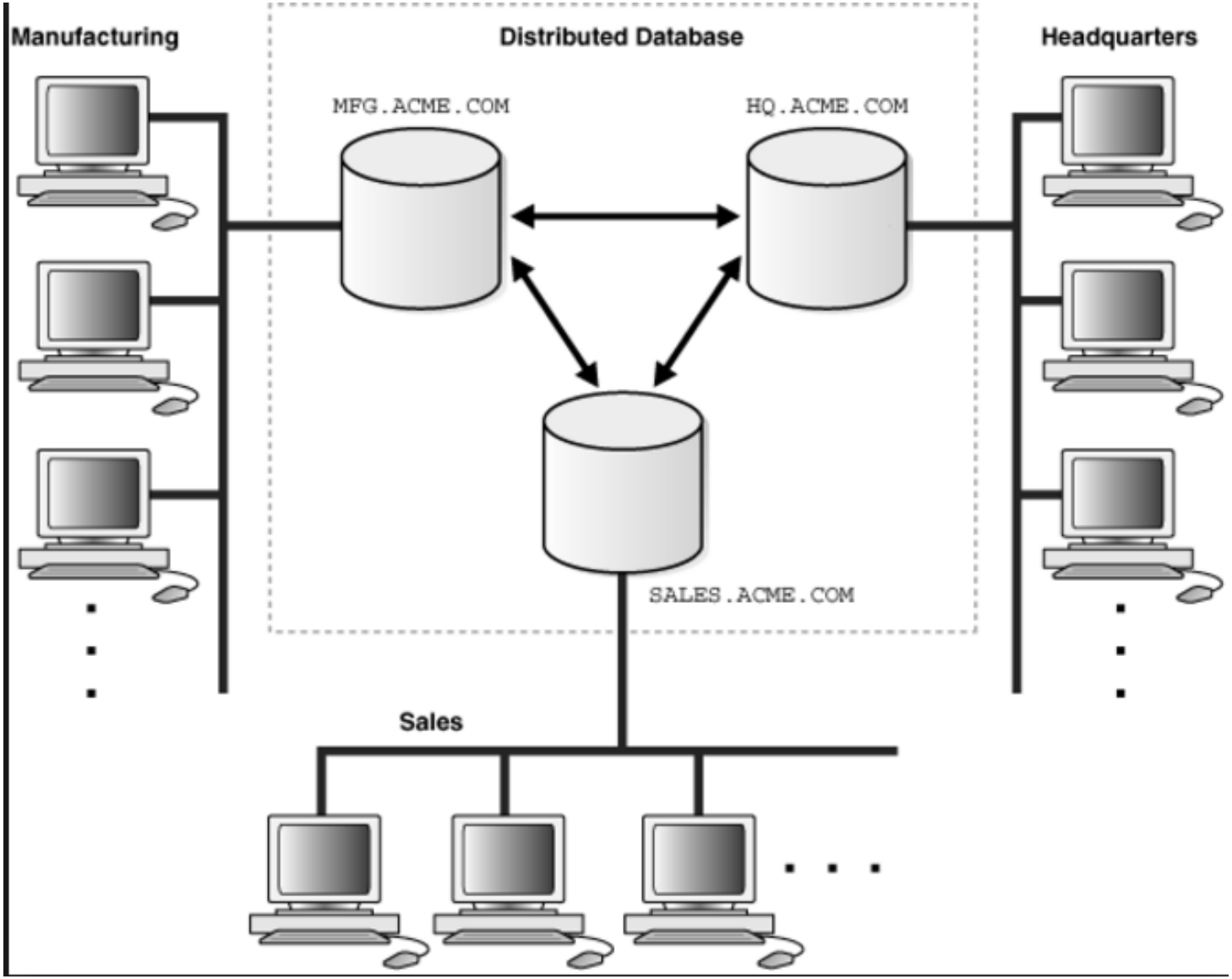
A Centralized Database





Distributed Database

A set of databases in a distributed system that can appear to applications as a single data source.



Difference between Centralised and Decentralized Database

Centralised Database	Distributed Database
A single database located at only one site of a network	It consists of more data files located at different sites.
Easy to get complete view of data	It is very complex to get complete view of data
Easy to Manage, Update and Backup data	Not Easy to Manage, Update and Backup data
No duplicate (replication) of data	Duplicate (Replication) of data
As the number of users increase the performance is down.	As the number of users increase the performance is balanced
All request handle by a single server so data speed may be slow	All requests are distribute among multiple server with multiple locations. Data accessing speed is fast
If single server is fail, whole system is down.	If one server is fail, other servers helps to restore data and system works fine.

Data Integrity

- It is the assurance that data is always be correct, consistent and accessible during storage, transfer and retrieval.
- Ways to ensure data integrity
  - Domain Integrity
  - Entity Integrity Constraint
  - Referential Integrity
  - Database consistency
- **Domain Integrity**
  - It means choosing the correct data type and length for a column.
  - It ensures that all the data items in a column fall within a defined set of valid values.
  - Each column in a table has a defined set of values, such as
    - For Mobile Number , Set of all numbers and length is 10 digit
    - For Person Name , Set of all characters and length is 50 char. and
    - For Birth Date , Set of all dates and length is 8

Data Integrity

- **Entity Integrity Constraints**
  - Each row in a relation must be unique identified.
  - **Primary key** shows the uniqueness of a rows, it cannot be NULL
- **Referential integrity**
  - Involves prevention of errors in the **relationship between a foreign key and primary key**.
  - If a table has a foreign key, then a values of the foreign key must be exist in the referenced table
  - E.g. Each product mentioned in bill must be exist in product master table .
- **Database consistency**
  - Must be **consistent before and after a transaction**
  - All database integrity constraints are satisfied

Data Protection

- **Database Security**
  - All data must be protected from types of threats
    - Accidental threats – such as
      - Operator carelessness
      - Power failure
      - Disk crashes and fire.
    - Intentional threats – such as
      - Unauthorized access to database
      - Human, to exploit weaknesses in the system for personal gain.

Security Measures

- Authentication : Identity the user
- Authorization : Access rights to the user
- Encryption : Secret writing

Data Protection

- **Database Backup and Recovery**
  - Use Export and Import utilities
  - Backup creates a copy of the database
  - Mainly two types of backup
    - Entire Backup
    - Incremental Backup

Anomalies

- **Anomalies** are problems that can occur in poorly planned, un-normalised databases where all the data is stored in one table.

- **Three types of Anomalies**
  - Insert Anomalies
  - Update Anomalies
  - Delete Anomalies

Anomalies

E.g. Company sells household products, The SALES table keeps track of it.

SALES					
Customer_ID	**Customer_Name**	**Cust_Addr**	**Store_Id**	**Product**	**Price**
1001	Jaya	Nana Bazar	S1	Soap	20
1007	Kavita	Mota Bazar	S2	Toothpaste	50
1010	Shreya	Mogari	S3	Shampoo	80
1001	Jaya	Nana Bazar	S4	Hair Oil	60

Insert Anomaly : Add Conditioner product at a price of Rs.100.

You can't insert this data to the SALES table until a customer buys Conditioner.

Update Anomaly: Update the address of Jaya must update the two rows.

Actually Jaya address should be changes only at single row but here it should update all rows where Jaya name comes. Update is not properly.

Delete Anomaly : Delete the record with customer with id = 1001

Deleting one fact (Customer 1001 bought Soap), you also delete another fact(Price of Soap and Price of Hair Oil from the database.)

Anomalies

- **Insertion Anomaly** - An Insert Anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

- **Update Anomaly** – It occurs when update value of an attribute affect the other values in attribute which make data inconsistent.

- **Delete Anomaly** – It occurs when delete an unimportant information the important information is also delete from the relation.

Anomalies

Insert Anomaly : Add conditioner product at a price of Rs.100.

Update Anomaly: Update the address of Jaya

Delete Anomaly : Delete the record with customer with id = 1007

Solution of Anomalies

PRODUCT		
Product_id	**Product_Name**	**Price**
P1	Soap	20
P2	Toothpaste	50
P3	Shampoo	80
P4	Hair Oil	60
P5	**Conditioner**	**100**

CUSTOMER		
Customer_ID	**Customer_ Name**	**Cust_Addr**
1001	**Jaya**	**Nana Bazar**
1007	Kavita	Mota Bazar
1010	Shreya	Mogari
1001	Jaya	Nana Bazar

CUST_SALES		Store_Id
Customer_ID	**Product_id**	**Store_id**
1001	**P1**	**S1**
1007	**P2**	**S2**
1010	P3	S3
1001	**P4**	**S4**

Normalization

- **Normalization** is the process of splitting relations into well structured relations that allow users to insert, delete, and update tuples (rows) without introducing database inconsistencies.
- **Step-by-step process :** At each step, a test (called a normal form) is applied to the current database design. If the design passes the test, it is said to confirm to the normal form.

- **<u>1NF (First Normal Form):</u>**
- **“A relation R is in First Normal Form (1NF) if it contains only atomic values. “**

OR

- **“The domain of an attribute must include only atomic (simple, indivisible) values and the value of any attribute in a tuple must be a single value from that domain. “**

Normalization : 1 NF

Example-1:

Emp (EmpNo, EmpName, DeptNo, Designation, Salary, PhoneNumbers)

Emp

EmpNo	EmpName	DeptNo	Designation	Salary	PhoneNumbers
101	Abc	1	Officer	10000	+91-2692-234567
102	Def	2	Executive	5000	(0265)2323232, 9898989898
103	Ghi	1	Manager	20000	(02692)212121, (0265)2424242
104	Jkl	2	Manager	20000	(079)21234567
105	Abc	3	Accountant	15000	9998999898

Normalization : 1 NF

Solution-1:

Emp (EmpNo, EmpName, DeptNo, Designation, Salary)

EmployeePhones (EmpNo, PhoneNumber)

Emp

EmpNo	EmpName	DeptNo	Designation	Salary
101	Abc	1	Officer	10000
102	Def	2	Executive	5000
103	Ghi	1	Manager	20000
104	Jkl	2	Manager	20000
105	Abc	3	Accountant	15000

Problem : Phone Number having multiple parts

EmployeePhones

EmpNo	PhoneNumber
101	+91-2692-234567
102	(0265)2323232
102	9898989898
103	(02692)212121
103	(0265)2426927
104	(079)21234567
105	9998999898

Normalization : 1 NF

Solution-2:

Emp (EmpNo, EmpName, DeptNo, Designation, Salary)

EmployeePhones (EmpNo, CountryCode, STDCode, PhoneNumber)

Emp

EmpNo	EmpName	DeptNo	Designation	Salary
101	Abc	1	Officer	10000
102	Def	2	Executive	5000
103	Ghi	1	Manager	20000
104	Jkl	2	Manager	20000
105	Abc	3	Accountant	15000

EmployeePhones

EmpNo	CountryCode	STDCode	PhoneNumber
101	91	2692	234567
102	91	265	2323232
102	91	0	9898989898
103	91	2692	212121
103	91	265	2424242
104	91	79	21234567
105	91	0	9998999898

Normalization : 1 NF

Example-2:
Emp (EmpNo, Name, FullAddress)

Emp

EmpNo	EmpName	FullAddress
101	Abc	Asas as as as, Kalawad Road, Rajkot, Gujarat, India
102	Def	203, Rajkot Seva Samaj Hostel, Vallabh Vidyanagar, Gujarat
103	Ghi	Dfdfd,d fdfdfd ,d fdfdfd, Hyderabad, Andhra Pradesh, India
104	Jkl	Adad, dfdfdfd, dfdfdfd, Hyderabad, Sindh, Pakistan
105	Abc	Asasasas, asdssdsdsd, sdsdsds, Delhi, India
106	Dfd	Fffgfgg, rfrrgrgr, rfrfrgr, New Delhi, India

Normalization : 1 NF

Solution:
Emp (EmpNo, Name, Addr1, Addr2, Landmark, Area, City, State, Country)

Emp

Emp No	EmpName	Addr1	Addr2	Landmark	Area	City	State	Country
101	Abc	Asas as	as as		Kalawad Road	Rajkot	Gujarat	India
102	Def	203, Rajkot Seva Samaj Hostel				Vallabh Vidyanagar	Gujarat	India
103	Ghi	Dfdfd	D	Fdfdfd	d fdfdfd	Hyderabad	Andhra Pradesh	India

Normalization : 1 NF

Example-3:

DeptEmp (DeptNo, Name, {Employees(EmpNo, Name, Salary)})

DeptEmp

DeptNo	Name	Employees		
1	Finance	EmpNo	Name	Salary
		101	aaa	10000
		102	bbb	20000
		103	ccc	30000
2	HR	EmpNo	Name	Salary
		104	ddd	15000
		105	eee	17000

Normalization : 1 NF

Solution:

Dept(DeptNo, Name)

Emp(EmpNo, Name, DeptNo, Salary)

Dept

DeptNo	Name
1	Finance
2	HR

Emp

EmpNo	Name	DeptNo	Salary
101	Aaa	1	10000
102	Bbb	1	20000
103	Ccc	1	30000
104	Ddd	2	15000
105	Eee	2	17000

Normalization : 1 NF

Example-4:

SuppliersItems (SupNo, SName, {Items (ItemNo, IName, Price)})

SuppliersItems

SupNo	SName	Items		
1	A Ltd.	**ItemNo**	**IName**	**Price**
		101	aaa	100
		102	bbb	200
		103	Ccc	300
2	B Ltd.	**ItemNo**	**IName**	**Price**
		104	ddd	150
		101	aaa	100

Normalization : 1 NF

Solution-1:

Suppliers (SupNo, SName)
Items (ItemNo, IName, SupNo, Price)

Suppliers

SupNo	SName
1	A Ltd.
2	B Ltd.

Items

ItemNo	IName	SupNo	Price
101	Aaa	1	100
102	Bbb	1	200
103	Ccc	1	300
104	Ddd	2	150
101	Aaa	2	100

Normalization : 1NF

- **How to make a relational design compliant with 1NF:**
  - If there is a **multi-valued attribute**, move it to a new relation of its own and copy the primary key of this relation to the new relation. **E.g. Emp , EmpContacts**
  - If there is a **composite attribute**, decompose it into multiple attributes, each holding one meaningful/separately identifiable subpart of that attribute. **E.g. Emp (Address)**
  - When we have **nested relations with one-many relationship**, we move the inner relation to a new independent relation of its own and copy the primary key of the outer relation to the newly created relation as its foreign key.

    E.g. Dept (DeptNo), Emp (EmpNo,DeptNo)
  - When we have **nested relations with many-many relationship**, we create **two separate relations** for the entity represented by the outer relation and the entity represented by the inner relation. We **also** create a **"relationship relation"** that contains the primary keys of both the relations. **E.g. Suppliers (SupNo) & SupplierItem(SupNo, ItemNo)**

Normalization : Functional Dependencies (FDs):

"The attributes of a table is said to be dependent on another attribute of the same table."

- **E.g. Student** (**Stud_Id**, Stud_Name, Stud_Contact, Stud_Email)
- Stud_Id attribute uniquely identifies the Stu_Name attribute of student table. **This is functional dependency** and in words we can say **Stud_Name is functionally dependent on Stud_Id**.
- **A functional dependency between two sets of attributes X and Y of a relation R, denoted by X → Y. X → Y means** If we know the value(s) in X, we can tell the value(s) in Y

- **Possible FDs :** Stud_Id -> Stud_Name
  Stud_Id->Stud_Contact,
  Stud_Id->Stud_Email

Normalization : Functional Dependencies (FDs):

SuppliersItems (SupNo, SName, ItemNo, IName, Price)

SuppliersItems

SupNo	SName	ItemNo	IName	Price
1	A Ltd.	101	aaa	100
1	A Ltd.	102	bbb	200
1	A Ltd.	103	Ccc	300
2	B Ltd.	104	ddd	150
2	B Ltd.	101	aaa	100

Normalization:
1NF: Satisfied
2NF: Not satisfied

Key:
(SupNo, ItemNo)

FDs:
(SupNo,ItemNo) -> Sname
(SupNo,ItemNo) -> Iname
(SupNo,ItemNo) -> Price

Normalization : Functional Dependencies (FDs):

- **Partial Functional Dependencies:**

  "When a non-key attribute is determined by a part, but not the whole, of a composite primary key."

  OR

  "A functional dependency X → Y is said to be a partial functional dependency if the functional dependency X → Y continues to hold even after removing at least one attribute from X"

SuppliersItems (SupNo, SName, ItemNo, IName, Price)

Key:
(SupNo, ItemNo)

SuppliersItems

SupNo	SName	ItemNo	IName	Price
1	A Ltd.	101	aaa	100
1	A Ltd.	102	bbb	200
1	A Ltd.	103	Ccc	300
2	B Ltd.	104	ddd	150
2	B Ltd.	101	aaa	100

Partial FDs:

(SupNo,ItemNo) -> Sname
SupNo -> Sname

(SupNo,ItemNo) -> Iname
ItemNo -> Iname

(SupNo,ItemNo) -> Price
ItemNo -> Price

Normalization : Functional Dependencies (FDs):

- **Full Functional Dependencies:**

  A functional dependency X → Y is said to be a **full functional dependency,** if removing any attribute from X means that the functional dependency X → Y no longer holds.

Suppliers

SupNo	SName
1	A Ltd.
2	B Ltd.

FFDs:

Suppliers (SupNo, SName)
Items (ItemNo, IName, Price)
SupplierItem(SupNo, ItemNo)

Items

ItemNo	IName	Price
101	Aaa	100
102	Bbb	200
103	Ccc	300
104	Ddd	150

Normalization:
1NF: Satisfied
2NF: Satisfied

SupplierItem

SupNo	ItemNo
1	101
1	102
1	103
2	104
2	101

Normalization : 2NF (Second Normal Form)

- **A relation R is in 2NF if**
  - **R is in 1 NF (Atomic)**
  - **Non-key attributes are FFD on Primary key.**

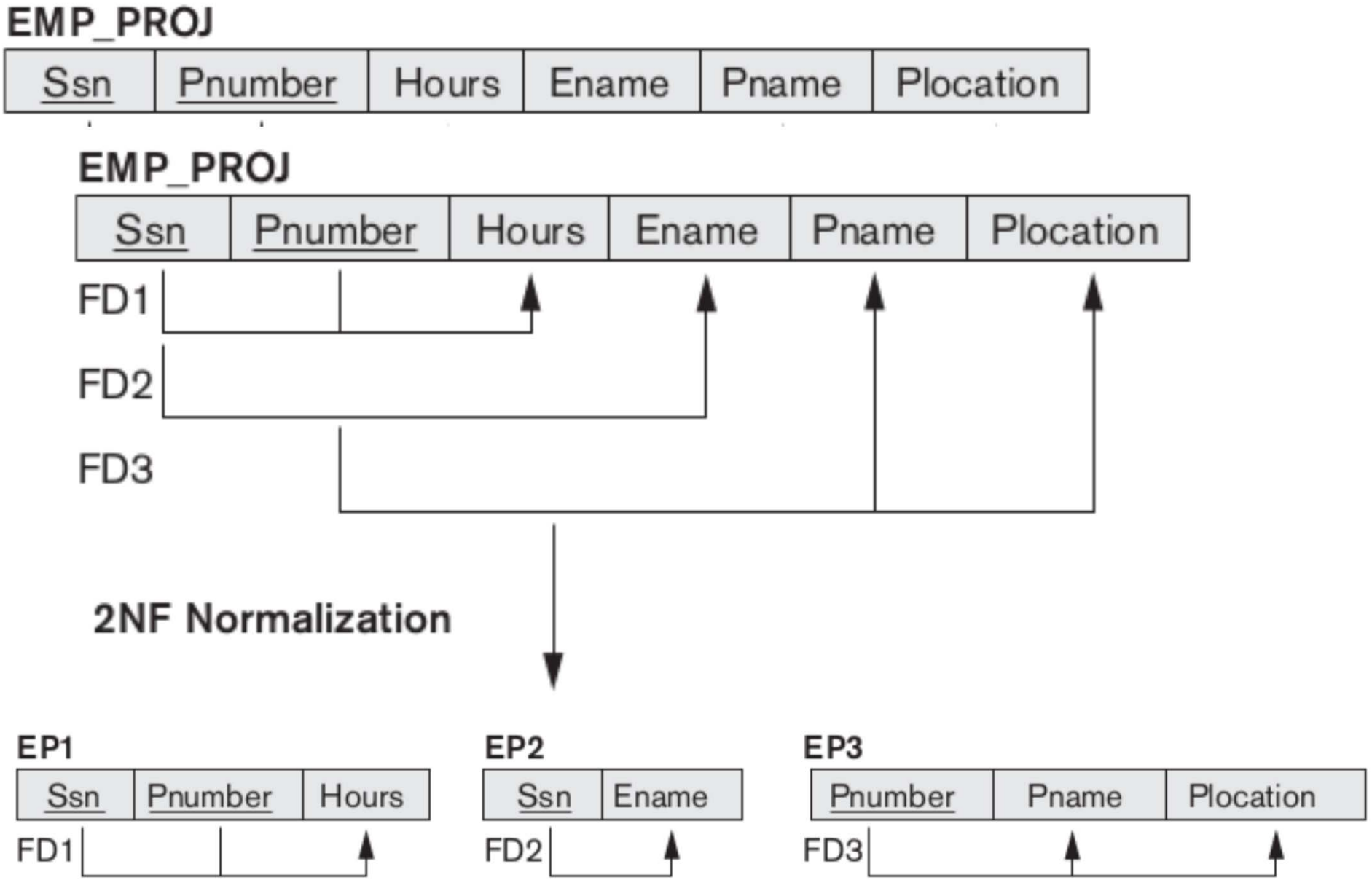
OR

- "Every non-prime attribute of the relation must be **fully functionally dependent** on the key(s)"

OR

- "There must **not be partial dependencies** of non-prime attributes on any key"

Exercise : Normalize Upto 2NF



Normalization :Functional Dependencies

- **Transitive Functional Dependencies :**
  A transitive dependency can occur only in a relation that has **three or more attributes.**

- **A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.**

- **E.g.** Let **X, Y, and Z** designate **three distinct attributes** in the relation. Suppose all three of the following conditions hold:

  - **X → Y**
  - **Y does not → X**
  - **Y → Z**

  Then X → Z is a transitive dependency.

Normalization : Transitive Functional Dependencies :

Book	Genre	Author	Author Nationality
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne	French
Journey to the Center of the Earth	Science Fiction	Jules Verne	French
Leaves of Grass	Poetry	Walt Whitman	American
Anna Karenina	Literary Fiction	Leo Tolstoy	Russian
A Confession	Religious Autobiography	Leo Tolstoy	Russian

- **Book → Author**
- **Author does not → Book**
- **Author → Author Nationality**
- **Transitive dependency occurred because a non-key attribute (Author)** was **determining** another **non-key attribute (Author Nationality).**
- Therefore **Book → Author Nationality is a transitive dependency.**

Normalization : 3NF (Third Normal Form):

- **A relation R is in 3 NF if**
  - R is in 2 NF
  - **There must not be any transitive dependencies of a non-prime attribute on a key.**

Normalization : 3 NF

Book	Genre	Author
Twenty Thousand Leagues Under the Sea	Science Fiction	Jules Verne
Journey to the Center of the Earth	Science Fiction	Jules Verne
Leaves of Grass	Poetry	Walt Whitman
Anna Karenina	Literary Fiction	Leo Tolstoy
A Confession	Religious Autobiography	Leo Tolstoy

Normalization:
1NF: Satisfied
2NF: Satisfied
3NF : Satisfied

Author	Author Nationality
Jules Verne	French
Walt Whitman	American
Leo Tolstoy	Russian

De-Normalization

- **De-Normalization** is the inverse process of normalization where the redundancy is added to the data to improve the performance of the specific application.

OR

- De-Normalization is the technique of combining the data into a single table to make data retrieval faster.
- The join query is expensive.
- **Example**

EMP_ZIP (emp_id,emp_name , emp_zip, emp_state, emp_city, emp_district)

Normalization up to BCNF is

EMP_MST(emp_id, emp_name , emp_zip)
ZIP_MST (emp_zip, emp_state, emp_city, emp_district)

- **Query**

```
SELECT E.emp_id, E.emp_name, E.emp_zip, Z.emp_state, Z.emp_state,
       Z.emp_city, Z.emp_district
FROM Emp_Zip  E, Zip_Mst  Z
WHERE E.emp_zip = Z.emp_zip;
```

It is a frequent and important query in the company. It takes more time to execute, so performance is down.

- **Query**

```
SELECT  E.emp_id, E.emp_name, E.emp_zip, Z.emp_state, Z.emp_state,
       Z.emp_city, Z.emp_district FROM Emp_Zip
```

It takes less time to execute, so performance is increase.

De-Normalization

- **Disadvantages of De-Normalization**
  - **Space wasted**
    - But space is cheap nowadays
  - **Anomalies when data changes**
    - But zip codes and city, state, district are unlikely to change
- **Advantage of De-Normalization**
  - **Improve Performance**