21CSC205P -Database Management Systems

Outline of the Presentation

Issues in File Processing System
Need for DBMS
Basic terminologies of Database

Database system Architecture

Various Data models

ER diagram basics and extensions

Construction of Database design using Entity Relationship diagram for University Database,

Entity Relationship diagram for Banking System, Information System

File Processing System

What is File Processing / Management System?

A File Processing / Management system is a DBMS that allows access to single files or tables at a time. In a File System, data is directly stored in set of files.

It contains flat files that have no relation to other files (when only one table is stored in single file, then this file is known as flat file).

Issues in File Processing System

- ✔ Data redundancy
- ✔ Data inconsistency
- ✔ Limited Data Sharing
- ✔ Data Dependency
- ✔ Lack of Data Integrity
- ✔ Limited Security
- ✔ Concurrency Control
- ✔ Scalability Issues
- ✔ Limited Query Capabilities

File Processing System

Issues in File Processing System

- **Data redundancy**

  Problem: Data is often duplicated across various files. This redundancy can lead to inconsistencies and increase the likelihood of data errors.

  Impact: Wasted storage space, increased maintenance efforts, and the risk of inconsistent data.

- **Data inconsistency**

  Problem: Changes made to data in one file may not be reflected in other related files. This inconsistency can lead to inaccurate reporting and decision-making.

  Impact: Unreliable information, potential for errors, and difficulties in maintaining data integrity.

File Processing System

Issues in File Processing System

✔ Limited Data Sharing

Problem: Difficulty in sharing data between different applications or departments. Each department might maintain its own set of files, making it challenging to access and integrate information across the organization.

Impact: Reduced collaboration, increased likelihood of outdated information, and inefficiencies in information retrieval.

✔ Data Dependence

Problem: Programs are often written to access specific files directly. If the structure of a file changes, it may require modifying all programs that use that file.

Impact: High maintenance costs, inflexibility in adapting to changing data structures, and increased risk of errors during updates.

File Processing System

Issues in File Processing System

✔ Lack of Data Integrity

Problem: In the absence of constraints and rules enforced by a database management system, maintaining data integrity becomes a manual task. This increases the risk of entering incorrect or inconsistent data.

Impact: Lower data quality, increased chances of errors, and difficulties in ensuring the accuracy of information.

✔ Limited Security

Problem: File systems often have limited security measures. Access controls are typically basic, and there's a higher risk of unauthorized access.

Impact: Data breaches, unauthorized modifications, and compromised system security.

File Processing System

Issues in File Processing System

- ✔ **Concurrency Control**

  Problem: Ensuring concurrent access to data by multiple users without conflicts is challenging. File systems may lack mechanisms to handle concurrent updates properly.

  Impact: Data corruption, lost updates, and challenges in maintaining data consistency in a multi-user environment.

- ✔ **Scalability Issues**

  Problem: As the volume of data grows, file processing systems may struggle to handle large datasets efficiently. Performance issues can arise.

  Impact: Reduced system performance, longer processing times, and challenges in scaling the system.

- ✔ **Limited Query Capabilities:**

  Problem: File processing systems often lack a query language and sophisticated querying capabilities. Retrieving specific information can be cumbersome.

  Impact: Inefficient data retrieval, increased complexity in generating reports, and challenges in extracting meaningful insights.

Introduction and applications of DBMS

DBMS stands for Database Management System.

- ✔ Database is collection of meaningful interrelated information.
- ✔ DBMS is a collection of set of programs to store and access data in an easy and effective manner from a database.

Need for DBMS:

- ✔ Database systems are developed to deal huge amount of data.
- ✔ Data to be stored and retrieved for data processing in an effective manner.

What is Database Management System ?

Data : It is a RAW FACT (It won't give any meaning)

Ex: 10, RAM, etc.

Information : Which gives meaning for Data

Ex: id = 10 , name = 'RAM' Distance in miles = 200, etc.

Database : Collection of meaningful interrelated information

Ex: DB2,ORACLE, SQL Server, MySQL, etc.

Database Management System (DBMS) :

- Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data which is stored in a database.
- A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.
- DBMS manage the data, the database engine, and the database schema, allowing for data to be manipulated or extracted by users and other programs.
- This helps provide data security, data integrity, concurrency, and uniform data administration procedures.

Introduction and applications of DBMS

Why use DBMS

- To develop software applications in less time.
- Data independence and efficient use of data.
- For uniform data administration.
- For data integrity and security.
- For concurrent access to data, and data recovery from crashes.
- To use user-friendly declarative query language

Advantage of DBMS over File Processing System

Advantage of DBMS over file system

- ✔ No redundant data
- ✔ Data Consistency and Integrity
- ✔ Data Concurrency
- ✔ Data Security
- ✔ Data Privacy
- ✔ Easy access to data
- ✔ Data Recovery
- ✔ Flexible

Applications of DBMS

Domain	Usage of DBMS
Banking	Managing customer information, account activities, payments, deposits, loans, etc.
Transportation	Maintain and Manage the Passenger Manifesto, reservations and schedule information.
Universities	Student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	To store customer details , product details & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
Social Media	Manage the user accounts, Security, Data access

Purpose of database system

Purpose of DBMS:

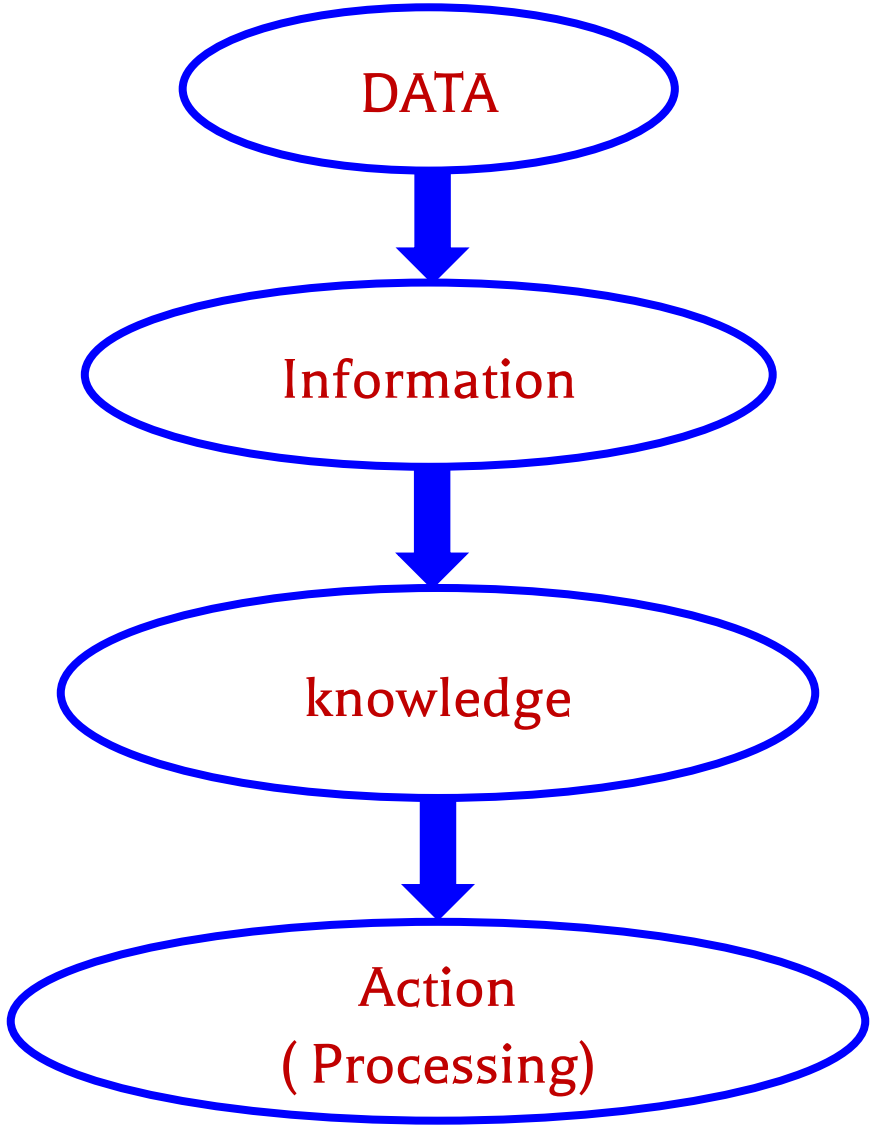
The purpose of DBMS is to transform the following –

- ✔ Data into information.
  - Raw fact is converted into Meaningful information
  - For Example:
    - The data ‘1000’ is converted into INR = ‘1000’
- ✔ Information into knowledge.
  - Using the information or comparing the information , can easily develop knowledge
  - For Example:
    - INR 75 = 1 USD
    - INR100 = 1.17 EURO
- ✔ Knowledge to the action.
  - Can predict the USD , EURO value in the mere future from history of information
  - In 1900 what is the equivalent of INR for USD
  - In 1950, 2000, etc.,
    - What will be the equivalent value in 2030?

Purpose of database system

Purpose of DBMS:

The diagram given explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS



Basic terminologies of Database

Database - A collection of organized data that is stored and can be easily accessed, managed, and updated.

DBMS (Database Management System) - Software that enables users to interact with the database. It provides tools for creating, managing, and querying databases.

Table - A collection of data organized in rows and columns. Tables are the basic structure in a relational database.

Row or Record - A single entry in a database table that contains data related to a specific entity or object.

Column or Field - A vertical section in a database table that represents a specific attribute or property. Columns hold the data for a particular aspect of the entity.

Primary Key - A unique identifier for each record in a table. It ensures that each row can be uniquely identified and retrieved.

Basic terminologies of Database

Foreign Key - A column in a table that refers to the primary key in another table. It establishes a link between two tables, enforcing referential integrity.

Index - A data structure that improves the speed of data retrieval operations on a database table.

Query - A request for data retrieval or manipulation from a database. Queries are typically written in a query language like SQL (Structured Query Language).

Normalization - The process of organizing the data in a database to eliminate redundancy and improve data integrity.

Relational Database - A type of database that uses a tabular structure to organize data, and relationships between tables are defined.

SQL (Structured Query Language) - A programming language used for managing and manipulating relational databases. It includes commands for querying, updating, and managing databases.

Basic terminologies of Database

Transaction - A sequence of one or more database operations treated as a single unit. Transactions ensure the consistency and integrity of a database.

ACID (Atomicity, Consistency, Isolation, Durability) - Properties that guarantee the reliability of database transactions. Atomicity ensures that transactions are treated as a single unit, Consistency ensures that a database remains in a valid state, Isolation ensures that transactions are independent of each other, and Durability ensures that committed transactions are permanent.

Schema - The structure or blueprint that defines the organization of data in a database, including tables, fields, relationships, and constraints.

Data Dictionary - A centralized repository that stores metadata about the database, including information about tables, columns, data types, and relationships.

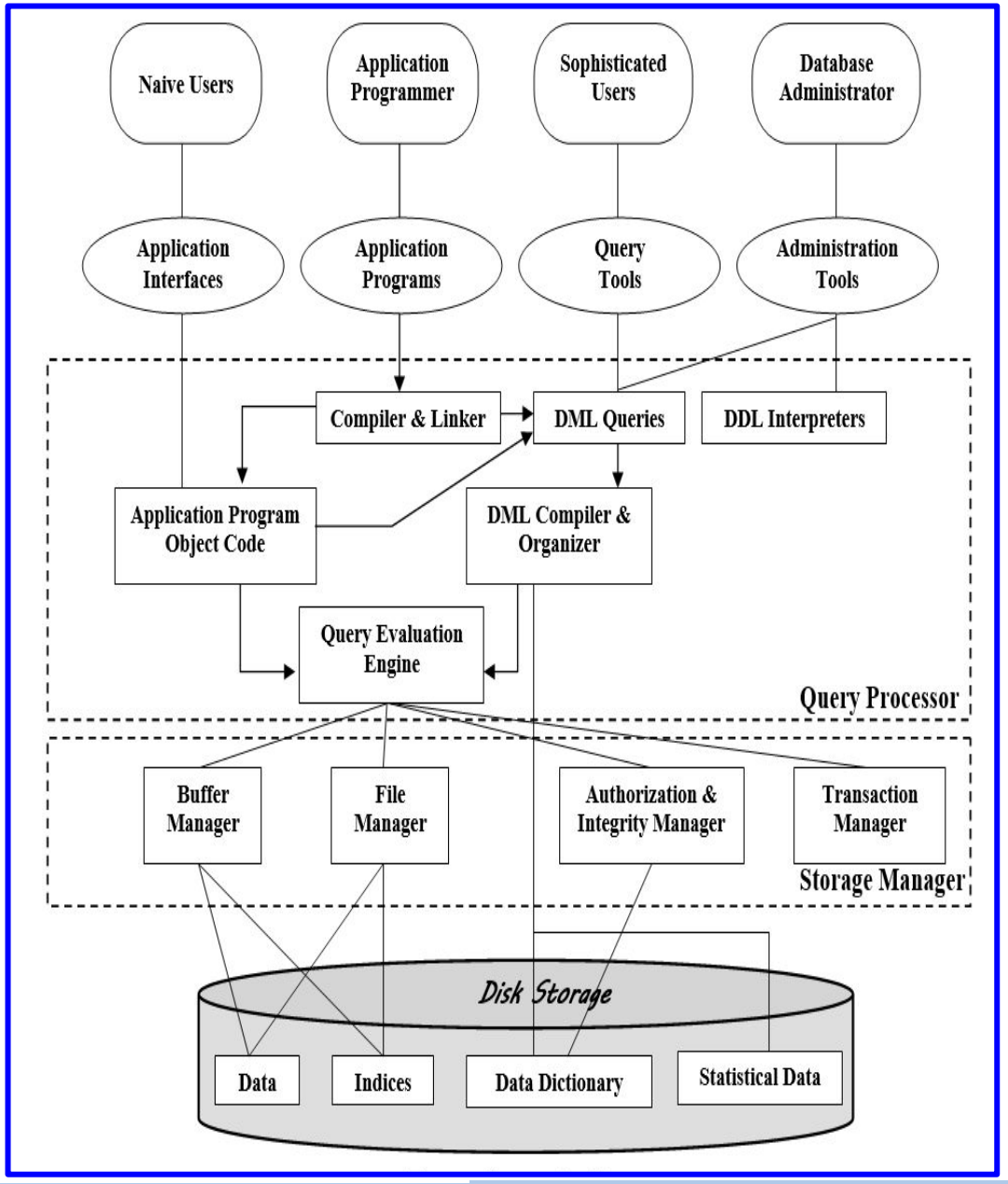
DATABASE SYSTEM ARCHITECTURE

There are four types users accessing / managing the database

- ✔ Naïve Users
- ✔ Application Programmers
- ✔ Sophisticated Users
- ✔ Database Administrators

The database system is divided into three components:

- ✔ Query Processor
- ✔ Storage Manager
- ✔ Disk Storage.



DATABASE SYSTEM ARCHITECTURE

Query Processor :

It interprets the requests (queries) from user(s) via an application program /interface into instructions.

It also executes the user request which is received from the DML compiler.

Query Processor contains the following components

Name of the Component	Purpose of the component
DML Compiler	It processes the DML statements into low level instruction
DDL Interpreter	It processes the DDL statements into a set of table containing meta data
Embedded DML Pre-compiler	It processes DML statements embedded in an application program into procedural calls.
Query Optimizer	It executes the instruction generated by DML Compiler.

DATABASE SYSTEM ARCHITECTURE

Storage Manager :

- ✔ It is an interface between the information stored in the database an and the requests (queries)
- ✔ It is also known as Database Control System
- ✔ It maintains the consistency and Integrity
- ✔ The main responsibility is managing the data manipulation such as addition deletion, modification , etc.,

Storage Manager contains the following components

Components	Purpose of the Components
Authorization Manager	It ensures role-based access control, i.e,. checks whether the particular person is privileged to perform the requested operation or not.
Integrity Manager	It checks the integrity constraints when the database is modified.
Transaction Manager	It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.
File Manager	It manages the file space and the data structure used to represent information in the database.
Buffer Manager	It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

Disk Storage

- ✔ Used to store all the information
- ✔ It contains the following components

Components	Purpose of the 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.
Statistical Data	Contains the statistics of all information

Data Independence

✔Data Independence is an important property of DBMS and also an advantage.

✔There are three levels in database:

1. Physical level / Low level (Disk storage)
2. Conceptual level (query / procedure / logics / etc.,)
3. Logical level / View level (User Interface)

✔Data Independence is used to achieve the changes in physical level without affecting logical level and vice versa.

✔There are two types of Data Independence in DBMS:

1. Physical Data Independence
2. Logical Data Independence

Data Independence

Need of Data Independence

- ✔ To improve the quality of data
- ✔ Easy maintenance of DBMS
- ✔ To achieve database security
- ✔ Developer need not be worry about internal structure
- ✔ Easily making the changes in physical level to improve the performance

Data Independence

Physical Data Independence

It is defined as to make the changes in the structure of the physical level /low level of DBMS without affecting the logical level / view level.

Some of the changes in Physical level

- ✔ Changing the storage devices
- ✔ Changing the file organization techniques
- ✔ Changing the data structures
- ✔ Changing the data access method
- ✔ Modifying indexes
- ✔ Migrating the Database from one drive to another

Data Independence

Logical Data Independence

It is defined as to make the changes in the structure of the logical level / view level of DBMS without affecting the physical / low level.

Some of the changes in Logical level

- ✔ Add a new attribute in an entity set
- ✔ Modify / Delete an attribute
- ✔ Merging records
- ✔ Splitting records

Data Independence

Difference between physical data independence and logical data independence

physical data independence	logical data independence
Concerned with the storage of the data.	Concerned with the structure of data definition.
Easy to retrieve	Difficult to retrieve because of dependent on logical structure
Easy to achieve, compare with logical data independence	Difficult to achieve, compare with physical data independence
Concerned with physical schema	Concerned with logical schema

The evolution of Data Models

To come across the limitations of file systems, there are lot of researchers and software developers designed and developed various data models.

The important and widely accepted models are:

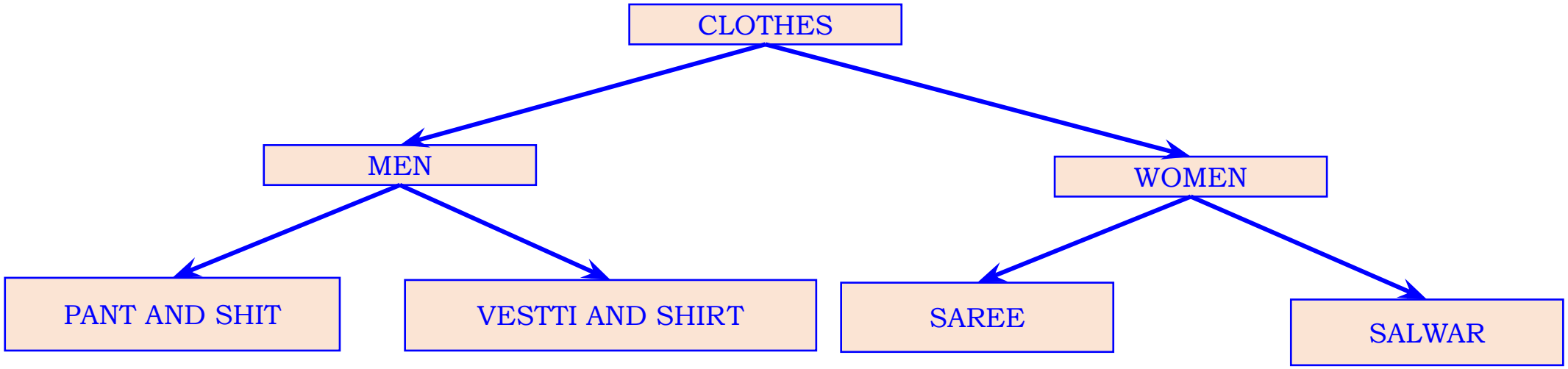
- ✔ Hierarchical
- ✔ Network
- ✔ Entity relationship
- ✔ Relational
- ✔ Object oriented

The evolution of Data Models

Hierarchical Model

- ✔ The first and fore most model of the DBMS.
- ✔ This model organizes the data in the hierarchical tree structure.
- ✔ This model is easy to understand with real time examples site map of a website

Example : For example the following is the representation of relationships present on online clothes shopping



The evolution of Data Models

Features of a Hierarchical Model

- ✔ One-to-many relationship:
- ✔ Parent-Child Relationship
- ✔ Deletion Problem:
- ✔ Pointers

Advantages of Hierarchical Model

- ✔ Simple and fast traversal because of using tree structure
- ✔ Changes in parent node automatically reflected in child node

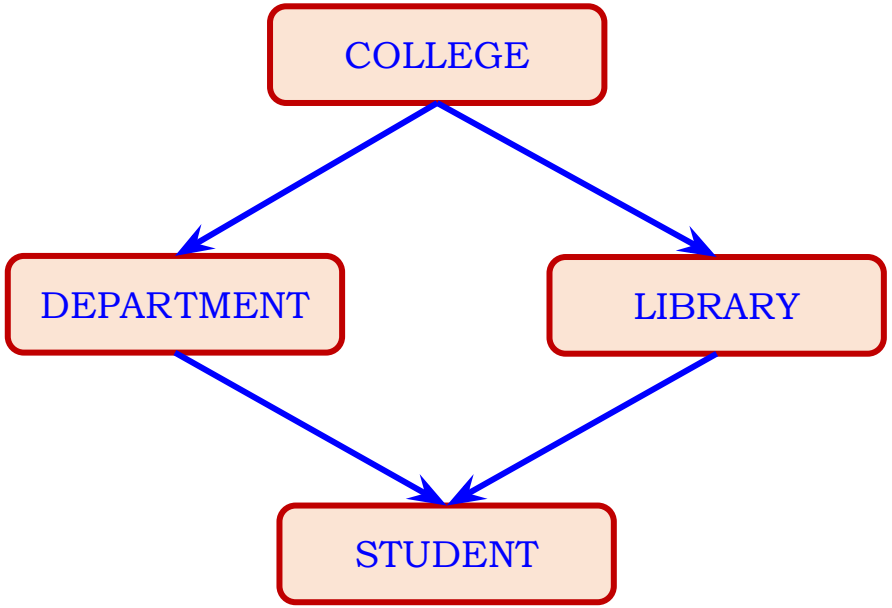
Disadvantages of Hierarchical Model

- ✔ Complexity
- ✔ Parent mode deleted automatically child node will be deleted

The evolution of Data Models

Network Model

- ✔ Network model is an extension of hierarchical model.
- ✔ This model was recommended as the best before relationship model.
- ✔ Same like hierarchical model, the only difference between these two models are a record can have more than one parent
- ✔ For Example consider the following diagram a student entity has more than one parent



The evolution of Data Models

Features of a Network Model

- ✔ Manage to Merge more Relationships
- ✔ More paths
- ✔ Circular Linked List

Advantages of Network Model

- ✔ Data access is faster
- ✔ Because of parent child relationship , the changes in parent reflect in child

Disadvantages of Network Model

- ✔ More complex because of more and more relations

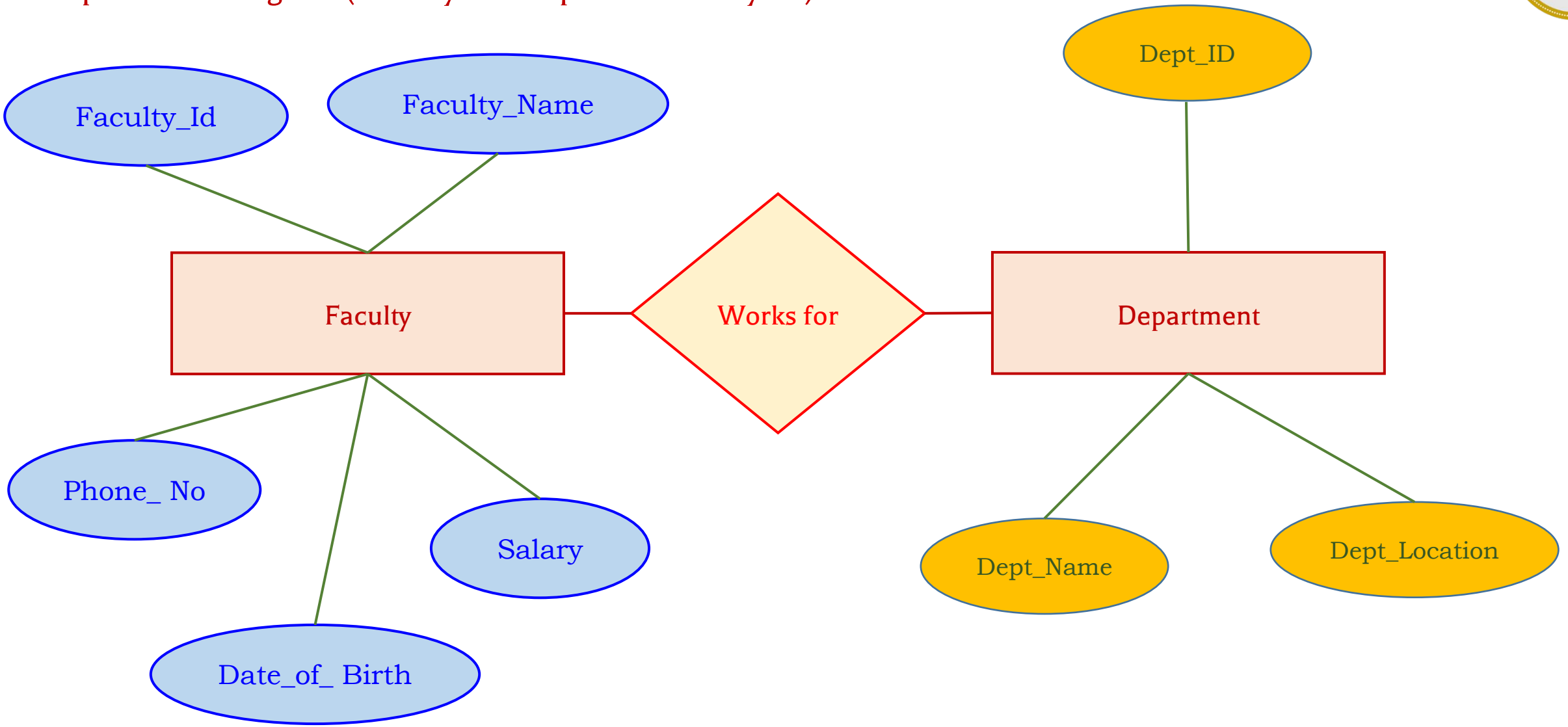
The evolution of Data Models

Entity-Relationship Model (ER Model)

- ✔ This model is a high level data model
- ✔ Represents the real – world problem as a pictorial representation
- ✔ Easy to understand by the developers about the specification
- ✔ It is like a visualization tool to represent a specific database
- ✔ It contains three components
  1. Entities
  2. Attributes
  3. Relationships

The evolution of Data Models

Example for ER Diagram (Faculty and Department entity set)



The evolution of Data Models

In the above example:

- ✔ There are two entities , Faculty and Department
- ✔ The attributes of Faculty entities are
  - Faculty_Id
  - Faculty_Name
  - Phone_No
  - Date_of_birth
  - Salary
- ✔ The attributes of Department entities are
  - Dept_ID
  - Dept_Name
  - Dept_Location
- ✔ Relationship : Faculty works for a department

The evolution of Data Models

Features of ER Model

- ✔ Graphical representation
- ✔ Visualization
- ✔ Good Database design (Widely used)

Advantages of ER Model

- ✔ Very Simple
- ✔ Better communication
- ✔ Easy to convert to any model

Disadvantage of ER Model

- ✔ No industry standard
- ✔ Hidden information

The evolution of Data Models

Relational Model

- ✔ Widely used model
- ✔ Data are represented as row-wise and column-wise (2 Dimensional Array)

Example : EMP (Employee) Table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	-	20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975	-	20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850	-	30
7782	CLARK	MANAGER	7839	09-JUN-81	2450	-	10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000	-	20
7839	KING	PRESIDENT	-	17-NOV-81	5000	-	10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100	-	20
7900	JAMES	CLERK	7698	03-DEC-81	950	-	30
7902	FORD	ANALYST	7566	03-DEC-81	3000	-	20
7934	MILLER	CLERK	7782	23-JAN-82	1300	-	10

The evolution of Data Models

Relational Model

- ✔ Each row is known as RECORD or TUPLE
- ✔ Each Column is known as ATTRIBUTE or FILED
- ✔ The collection of attributes are called as record – An Entity
- ✔ The collection of records are called as Table – Entity Set
- ✔ In the above example:

  Table – EMP

  Attributes – Empno, Ename, Sal,….

The evolution of Data Models

Features of Relational Model

- ✔ Records
- ✔ Attributes

Advantages of Relational Model

- ✔ Simple
- ✔ Scalable
- ✔ Structured format
- ✔ Isolation

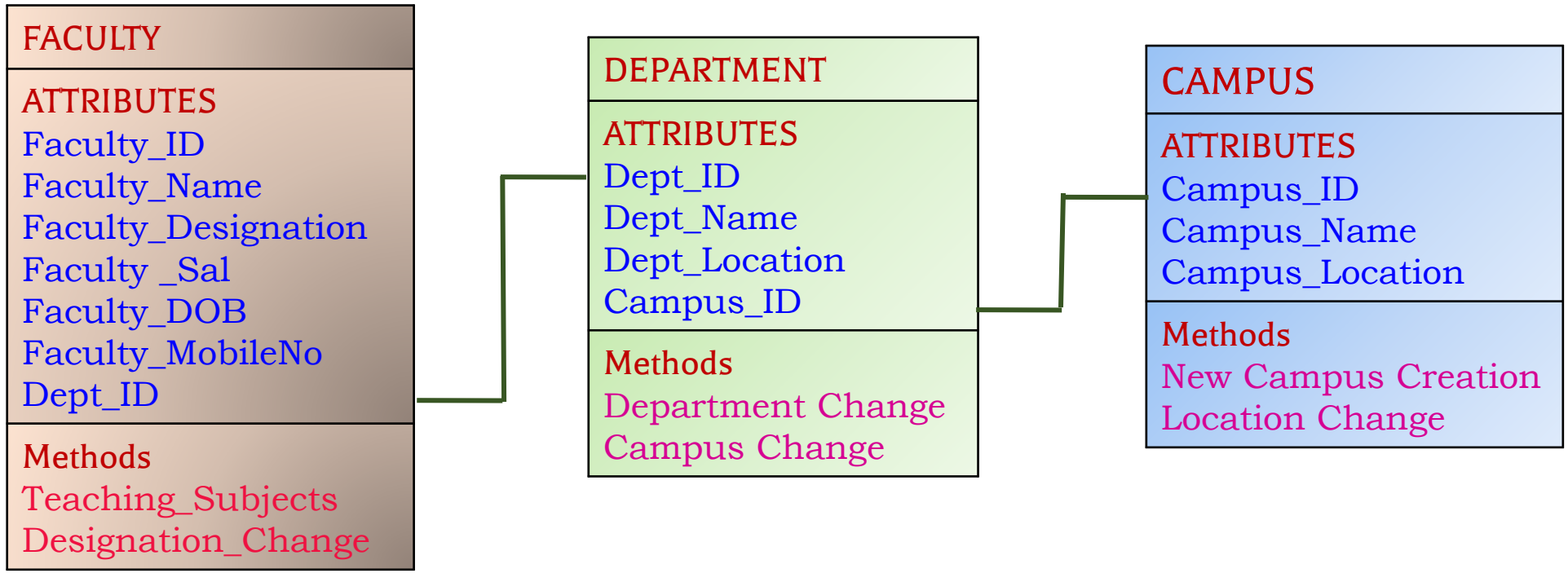
Disadvantages of Relational Model

- ✔ Hardware overheads

The evolution of Data Models

Object Oriented Model

- ✔ The real- time problems are easily represented through object-oriented data model which is an OBJECT.
- ✔ In this Model, the data and its relationship present in the single structure
- ✔ Complex data like images, audio, videos can be stored easily
- ✔ Objects connected through links using common attribute(s)
- ✔ Example : Three Objects Faculty, Department and Campus linked using common attribute



Degrees of Data Abstraction

- ✔ Data abstraction is the idea that a database design begins with a high level view and as it approaches implementation level, the level of detail increases.
- ✔ In 1970, the American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARC) established a framework for database design based on the degrees of abstraction.
- ✔ The ANSI/SPARC architecture is composed of four levels of data abstraction; these levels are external, conceptual, internal, and physical

Reference : https://databasemanagement.fandom.com/wiki/Degrees_of_Abstraction

Degrees of Data Abstraction

- The External Model is the end users' view of the data. The end users view of data usually applies to their specific business needs and those of their organizational unit.

- The Conceptual Model is the database as seen by the specific DBMS. What sets the internal model apart from the external and conceptual is its reliance on its software platform.

- The goal in designing the internal model is to achieve logical independence, where the internal model can be changed without affecting conceptual model.

Reference : https://databasemanagement.fandom.com/wiki/Degrees_of_Abstraction

Degrees of Data Abstraction

- The Physical Model is the final and lowest level of abstraction. This is the model which describe such implementation level design as how the data is stored on media and what media to use. This level of abstraction is reliant on software and hardware.

Note:

- If the rules established by the ANSI/SPARC are followed, the database is easily scalable and upgradeable.
- A common need is for the ease of upgradability in the physical model.
- As technology improves and as the database grows and needs more processing power and space it is important to be able to upgrade the hardware without worrying about needing to redesign parts or the entire database.

Reference : https://databasemanagement.fandom.com/wiki/Degrees_of_Abstraction

Database Users and DBA

Database Users

- ✔ Naive Users
- ✔ Application Programmers
- ✔ Sophisticated Users
- ✔ Native Users
- ✔ Specialized Users
- ✔ Stand-alone Users

Database Users and DBA

Naive Users

- ✔ Those who don't have any knowledge about DBMS
- ✔ Use DBMS applications frequently
- ✔ Mostly using the internet browser as an interface to access the database
- ✔ They don't have any privileges to modify the database, simply use the application
- ✔ Example : Railway booking users, Clerks in bank accessing database

Application Programmers

- ✔ Users who develop DBMS applications.
- ✔ They are backend programmers
- ✔ Programs can be written in any programming languages like C++, JAVA, Python, PHP

Database Users and DBA

Sophisticated Users

- ✔ Having knowledge about database and DBMS
- ✔ They can create their own applications based on requirements
- ✔ They don’t write codes in any programming languages, but able to manage using queries
- ✔ Example : Business Analyst, Researchers

Native Users

- ✔ These are the users, who use the existing database applications
- ✔ They don’t write any codes or queries
- ✔ Example: Library Management Systems, Inventory Control Systems

Database Users and DBA

Specialized Users

- ✔ These are also sophisticated users, but they write special database application programs.
- ✔ They are the developers who develop the complex programs to the requirement.

Stand-alone Users

- ✔ These users will have a stand-alone database for their personal use.
- ✔ These kinds of the database will have readymade database packages which will have menus and graphical interfaces.

Database Users and DBA

Database Administrator (DBA)

- ✔ DBA is a person or a group who define and manage the database in all three levels.
- ✔ DBA can create / modify /remove the users based on the requirements.
- ✔ DBA is the super user having all the privileges of DBMS

Responsibilities of DBA

- ✔ Install the Database
- ✔ Upgrade the Database
- ✔ Design and Implementation
- ✔ Database tuning
- ✔ Migrating the Database
- ✔ User Management
- ✔ Backup and Recovery
- ✔ Security of the Database in all access points
- ✔ Documentation

Design process

The Entity-Relationship (E-R) Model

Entity : Any object in the real world is an entity

Example : Person, Furniture, University / Department

The ER data model uses a collection of entities (objects) and relationships among these entities

Entities in database are described using their attributes / properties

Example 1 : The attributes like dept_id, dept_name, dept_location, etc., describes about a particular department in an university.

Example 2 : The attributes Faculty_id, Faculty_name, Faculty_salary, etc., describes about a faculty works for the particular department.

Note : The attributes dept_id, faculty_id used to identify an entity in an entity set. Like AADHAR CARD number for a person . (Will be discussed later in detail)

Design process

The Entity-Relationship (E-R) Model

Relationship :

- ✔ It is an association among several entities
- ✔ For example , a member is associates as faculty in her/his department.
- ✔ Faculty works for the department.

Entity set : Set of all entities of the same type

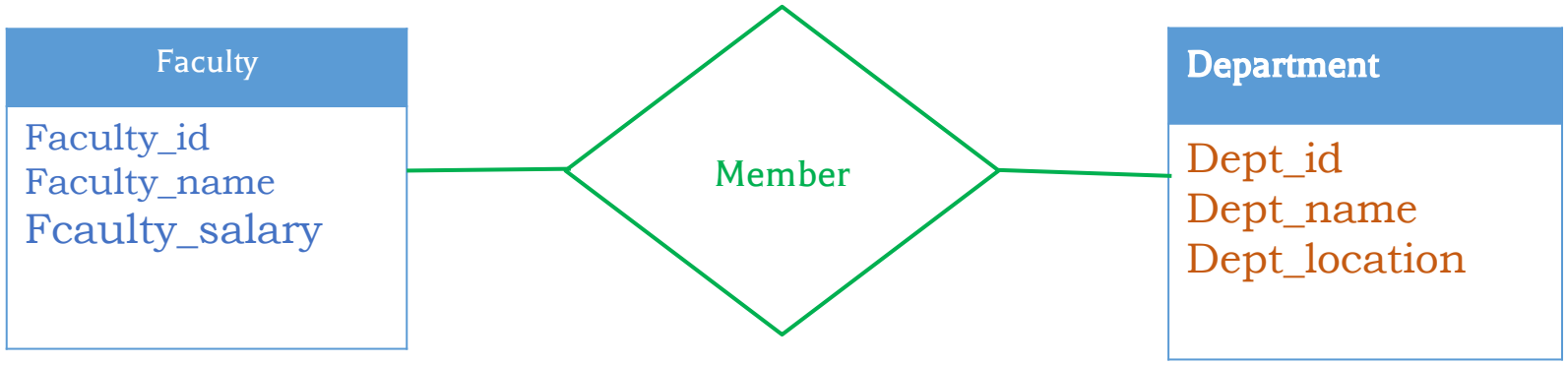
Relationship set : Set of all relationships of the same type

- ✔ The overall logical structure of a database can be represented using graphical notations by an E-R diagram.
- ✔ One of the most popular model is to use UML (Unified Modeling Language)

Design process

The Entity-Relationship (E-R) Model

A Sample E-R Diagram



- ✔ Entity sets are represented by a Rectangle : Faculty and Department
  - Header as Name of the Entity set
  - Attributes are listed below the header
- ✔ Relationship sets are represented as Diamond : Member
- ✔ The above E-R diagram represents the relationship member between faculty and department

Design process

Normalization

- ✔ Normalization is a method to design a relational database
- ✔ It is a process to avoid redundant information and also inability to represent certain information
- ✔ It is used to design a good database without redundant information
- ✔ The most common approach is to use functional dependencies
- ✔ There are several normal forms are available , each normal forms designed using various functional dependencies

Entity Relationship Model

- ✔ Entity – Relationship (E-R) Model is the overall logical structure of database design about a particular enterprise or domain
- ✔ E-R model is very useful in mapping the meaning and interactions of real world enterprises to conceptual schema
- ✔ E-R Model is widely used model in database design
- ✔ E-R Model employs three basic concepts
  - Entity sets
  - Relationship sets
  - Attributes

Entity Relationship Model

Entity Sets

- ✔ Any object in the real world is an entity
- ✔ For example , each faculty in an university is an entity
- ✔ An entity has a set of properties called attributes
- ✔ The values stored in one or more attributes will identify an entity uniquely in an entity sets
- ✔ For example , faculty_id is an attribute hold a unique value of a faculty, similarly the student_Register_no is unique for all students

Entity Relationship Model

Entity Sets

- An entity set is a set of entities of the same type that shares the same attributes.
- The set of people who are faculties at a given university, can be defined as entity set “faculty”
- Similarly the entity set “student” represent all the students in the university.
- The entity sets do not need to be disjoint.
- For example we can create an entity set called “person” can have faculty entity , student entity, both or neither.

Entity Relationship Model

Attributes

- ✔ Attributes are descriptive properties possessed by each member of an entity set.
- ✔ Each entity is represented by a set of attributes.
- ✔ Each attribute of an entity set will store the similar information.
- ✔ Each entity must have its own value for each attribute.
- ✔ Possible attributes for faculty entity set are
  - faculty_id (unique)
  - faculty_name
  - faculty_dept
  - faculty_salary
  - etc.,

Entity Relationship Model

Values

- ✔ Each entity has a value for each attribute
- ✔ For instance , the particular faculty entity may have the following values :
  - faculty_id = 100186
  - faculty_name = ‘Nantha’
  - faculty_ dept = ‘Computing Technologies’
  - faculty_salary = 123456
  - faculty_mobile = 9999955555
- ✔ The faculty_id attribute is used to identity the faculty uniquely , because there is a possibility for more number of faculties will have the same name
- ✔ In general the university use to assign unique id for faculty and students (Reg. No)

Entity Relationship Model

- A database for a university may include a number of entity sets.
- For example , to keeping track of faculty and students , the university also has the information about courses.
- The entity set has the following attributes
  - course_id
  - course_title
  - department_id
  - credits
- In a real setting , university database may keep more number of entity sets.

Entity Relationship Model

Entity sets faculty and student

100186	**Nantha**
100181	Murugan
100199	Ganesh
100201	Senthil
100210	Pradeep
100212	Sivakumar
100300	Chirsty

Entity set : Faculty

RA1911003010001	Koduru siva gowtham reddy
RA1911003010003	Abhinav ranjan
RA1911003010004	Venkata rakesh chowdary .
RA1911003010005	Avi tewari
RA1911003010006	Jayesh jayanandan
RA1911003010007	Ajay samuel victor
RA1911003010008	M p nanda
RA1911003010009	Harshil bhandari
RA1911003010011	Dhanush jayakrishnan nair
RA1911003010012	Rachana komanduri

Entity set : Student

Entity Relationship Model

Relationship Sets

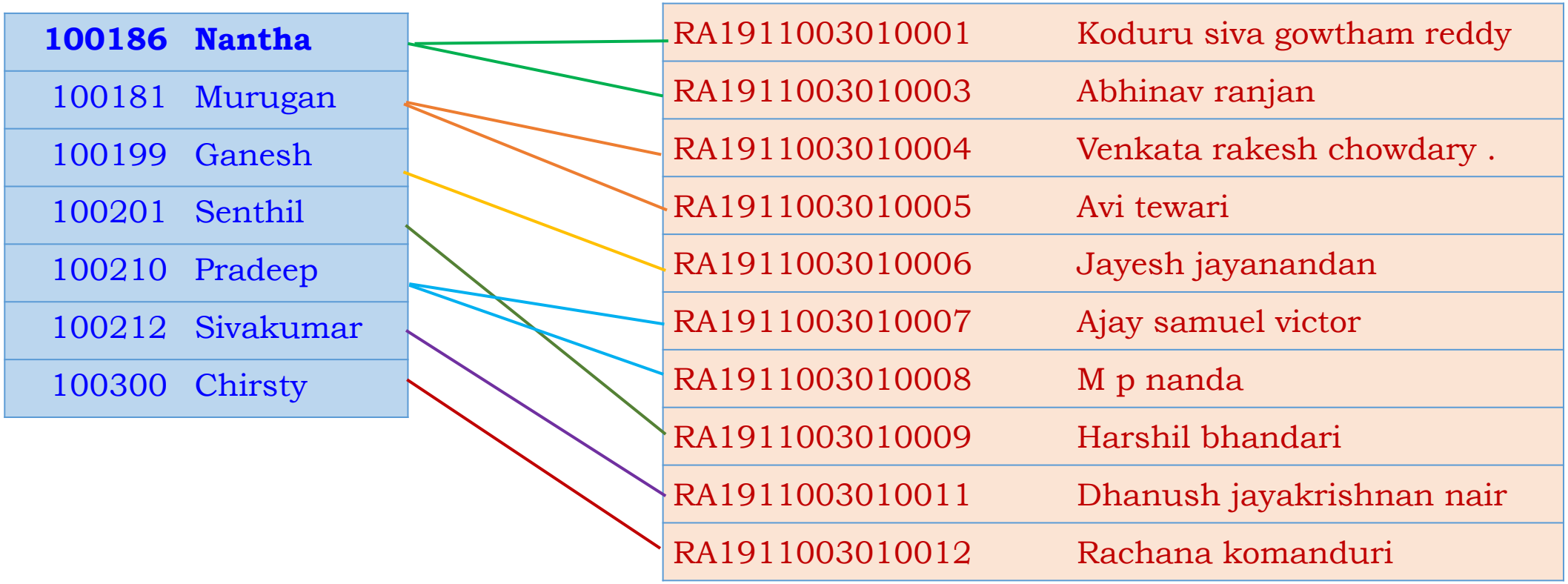
- A relationship is an association among several entities.
- For example , we can define a relationship counselor that associates faculty Nantha with the student Abhinav ranjan
- The realtionship specifies that Nantha is a counselor to student Abhinav ranjan.
- A relationship set is a set of relationships of the same type.
- Formally, it is a mathematical relation on n ≥ 2 (possibly nondistinct) entity sets.
- If E1, E2,..., En are entity sets, then a relationship set R is a subset of

$$\{(e1,e2,\ldots,en) \mid e1 \in E1, e2 \in E2, \ldots, en \in En\}$$

where (e1,e2,...,en) is a relationship.

Entity Relationship Model

- ✔ Consider the two entity sets Faculty and Student (Ref : Slide No 21)
- ✔ We define the relationship set counselor to denote the association between faculty and students.
- ✔ The following figure represents this association



Entity Relationship Model

- The association between entity sets is referred to as participation.
- The entity sets E1, E2,..., En participate in relationship set R.
- A relationship instance in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled.
- To explain this, the individual faculty entity Nantha, who has faculty_id 100186, and the student entity Abhinav ranjan who has student_regno RA1911003010003 participate in a relationship instance counselor.
- This relationship instance represents that in the university, the faculty Nantha is counseling student Abhinav ranjan.

Entity Relationship Model

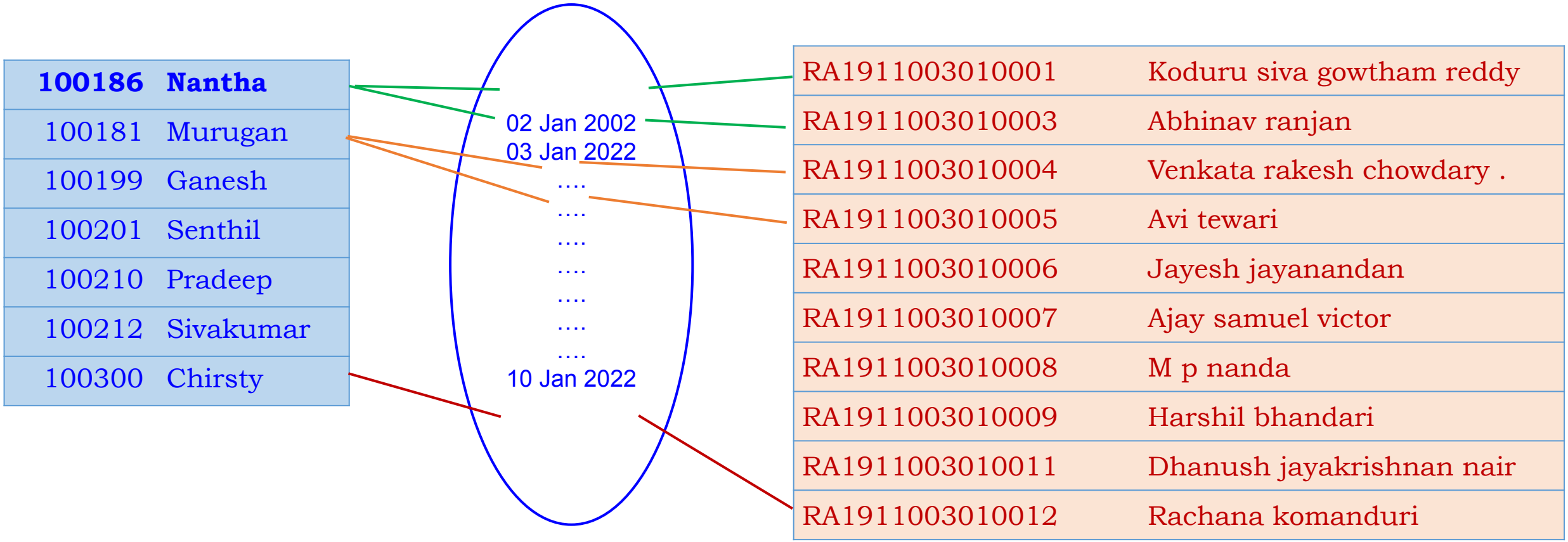
- ✔ The function that an entity plays in a relationship is called that entity's role.
- ✔ Since entity sets participating in a relationship set are generally distinct, roles are implicit and are not usually specified.
- ✔ The same entity set participates in a relationship set more than once, in different roles.
- ✔ In this type of relationship set, sometimes called a recursive relationship set, explicit role names are necessary to specify how an entity participates in a relationship instance.
- ✔ Example:
  - Consider the "course" entity set, which contains all about the courses offered in the university.
  - One course C2 , has a prerequisite course C1
  - The relationship set prereq that is modeled by pairs of course entities.
  - All relationships of prereq are characterized by (C1,C2) pairs, but (C2,C1) pairs are excluded

Entity Relationship Model

- A relationship may also have attributes called descriptive attributes.
- Consider a relationship set “counselor” with entity sets Faculty and Student.
- The attribute date can be associate with that relationship to specify the date when the faculty became the counselor of a student.
- The advisor relationship among the entities corresponding to faculty Nantha and student Abhinav ranjan has the value “3 Jan 2022” for attribute date, which means that Nantha became Abhinav ranjan’s counselor on 3 Jan 2022.

Entity Relationship Model

- The following figure shows the relationship set counselor with a descriptive attribute date.
- Faculty Nantha counsel two students with two different counseling dates.



Entity Relationship Model

Binary relationship set

- One entity set involves in two entity sets is known as Binary relationship set.

Example

- The faculty and student entity sets participate in relationship set counselor.

- In addition each student must have another faculty who works as department counselor (Co-ordinator)
- Then the faculty and student entity sets may participate in another relationship set, dept counselor.

Entity Relationship Model

Attributes

- For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute.
- For example the domain attribute of student_regno might be the set of all text strings of a certain length.
- Similarly the domain attribute of dept_name might be strings from the set
  { CSE,IT, MECH,ECE, EEE, BT,….}
- An attribute of an entity set is a function that maps from the entity set into a domain.
- An entity set may have several attributes, Each entity is described by a set of (Attribute, Data Value) Pairs.
- For example , A particular ,the Faculty entity may be described by a set
  { (faculty_id, 100186), (faculty_name, Nantha), (dept_name, cse), (salary, 123456) }

Entity Relationship Model

Attribute types

Simple : Values can not be divided into subparts

Example : Faculty_salary, Dept_name, etc.,

Attributes like salary, deptname can't be divided further

Composite : Values can be divided into subparts

Example : Faculty_name, Faculty_address

Faculty_name can be divided into first_name, middle_name, last_name

Faculty_address can be divided into Door_no, Street_name, City_name, State_name, Pincode

Entity Relationship Model

Types of Values	Description
Single value	• Only one value can be stored • Example : Faculty_id, DOB
Multiple value	• More values are possible • Example : Faculty_Phone_no
Derived value	• The values which is derived from existing value • Example : AGE • The values keep on changing is not advisable to store in the database • Normally the values will be derived from existing value of another attribute. • AGE will be changing continuously. • It can be derived from DOB (DOB never change)
Null value	• NULL values are unknown undeclared • An attribute does not have a value for a particular entity in an entity set

Entity Relationship Model

Constraints

- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.
- This is achieved using
  - Mapping Cardinalities
  - Participation Constraints

Entity Relationship Model

Mapping Cardinalities

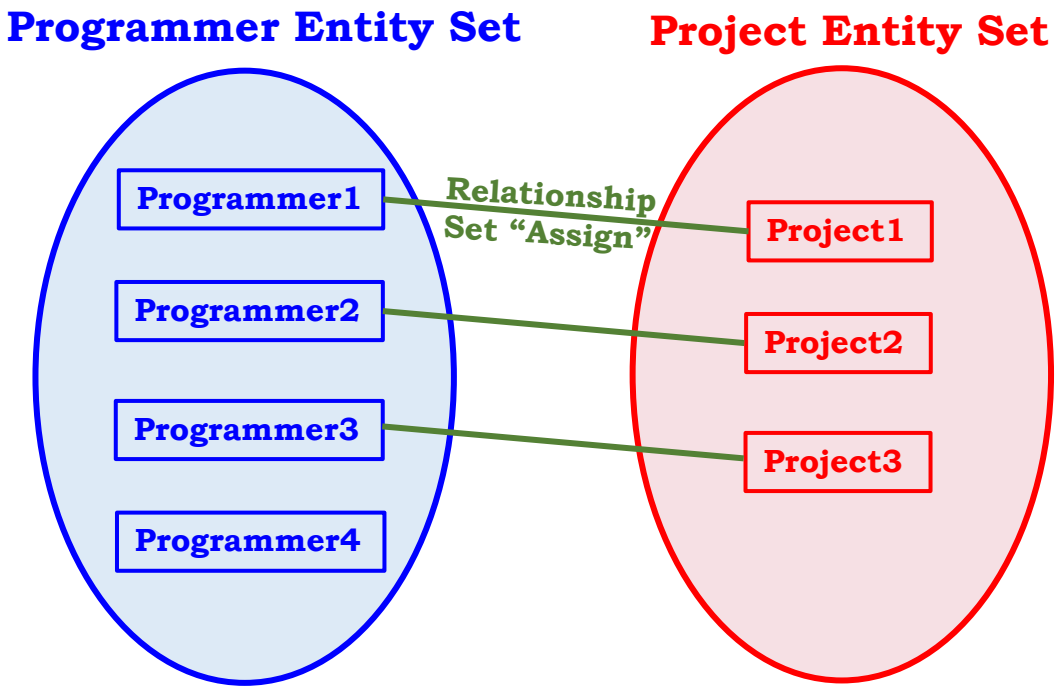
- ✔ Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
- ✔ Mapping cardinalities are most useful in describing binary relationship sets.
- ✔ For a binary relationship set “Assign” between entity sets Programmer and Project the mapping cardinality must be one of the following.
  - One-to-One (1:1)
  - One-to-Many (1:M)
  - Many-to-One (M:1)
  - Many-to-Many (M:M)

Entity Relationship Model

Mapping Cardinalities

One-to-One (1:1)

- ✔ An entity in Programmer is associated with at most one entity in Project, and an entity in Project is associated with at most one entity in Programmer.
- ✔ The following figure depicts 1:1 mapping cardinality

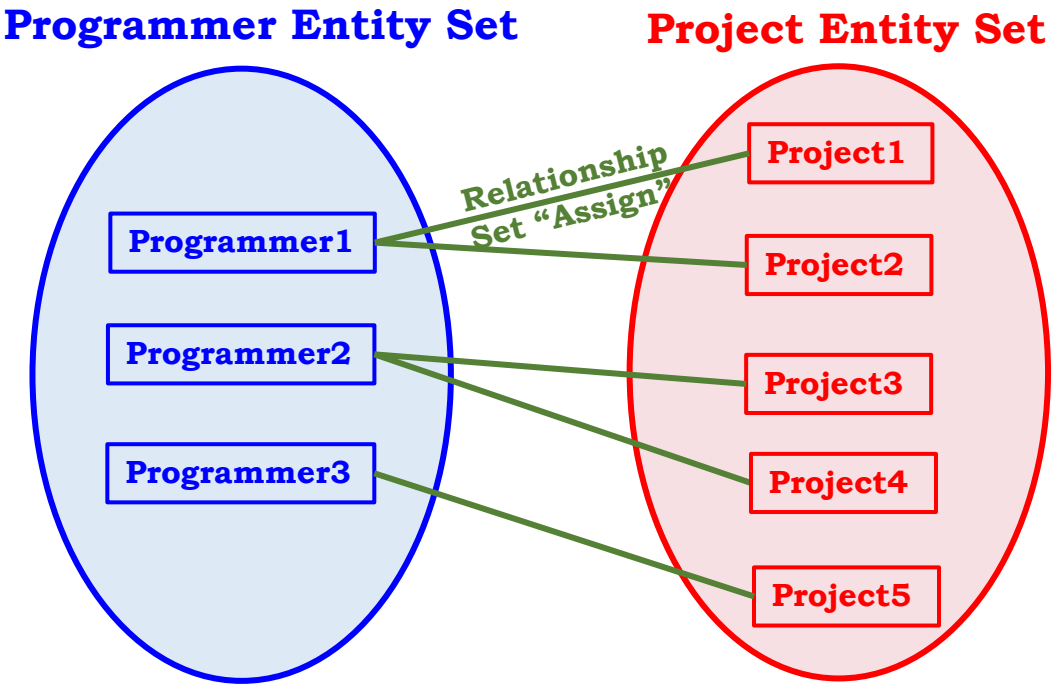


Entity Relationship Model

Mapping Cardinalities

One-to-Many (1:M)

- ✔ One-to-many. An entity in Programmer is associated with any number (zero or more) of entities in Project. An entity in Project, however, can be associated with at most one entity in Programmer.
- ✔ The following figure depicts mapping cardinality 1:M

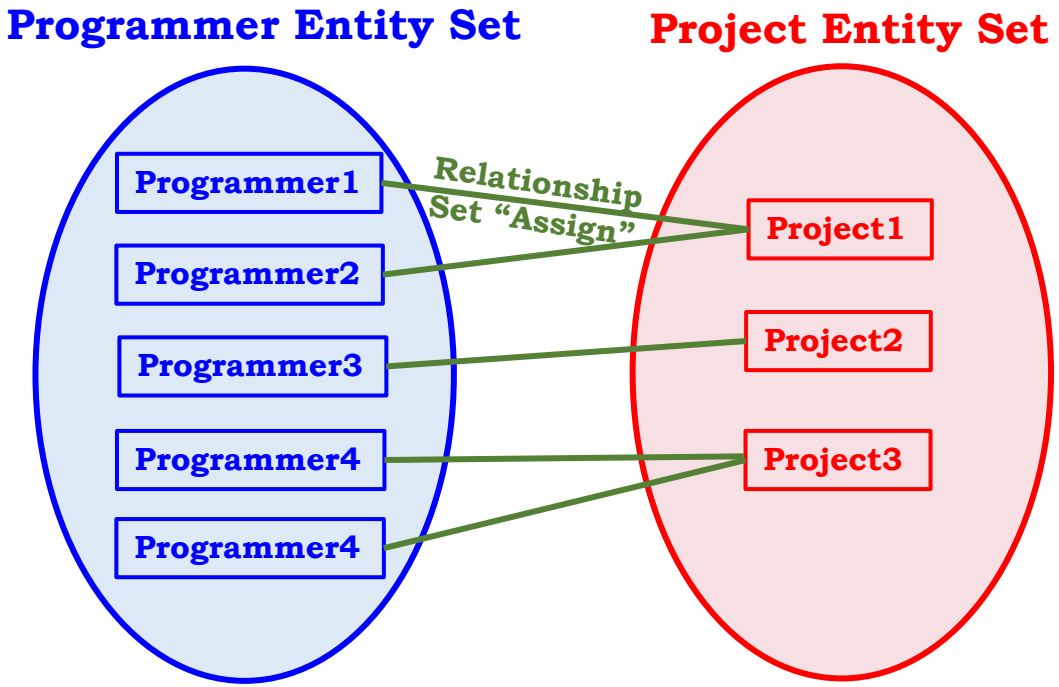


Entity Relationship Model

Mapping Cardinalities

Many-to-One (M:I)

- ✔ An entity in Programmer is associated with at most one entity in Project. An entity in Project, however, can be associated with any number (zero or more) of entities in Programmer.
- ✔ The following figure depicts mapping cardinality 1:M

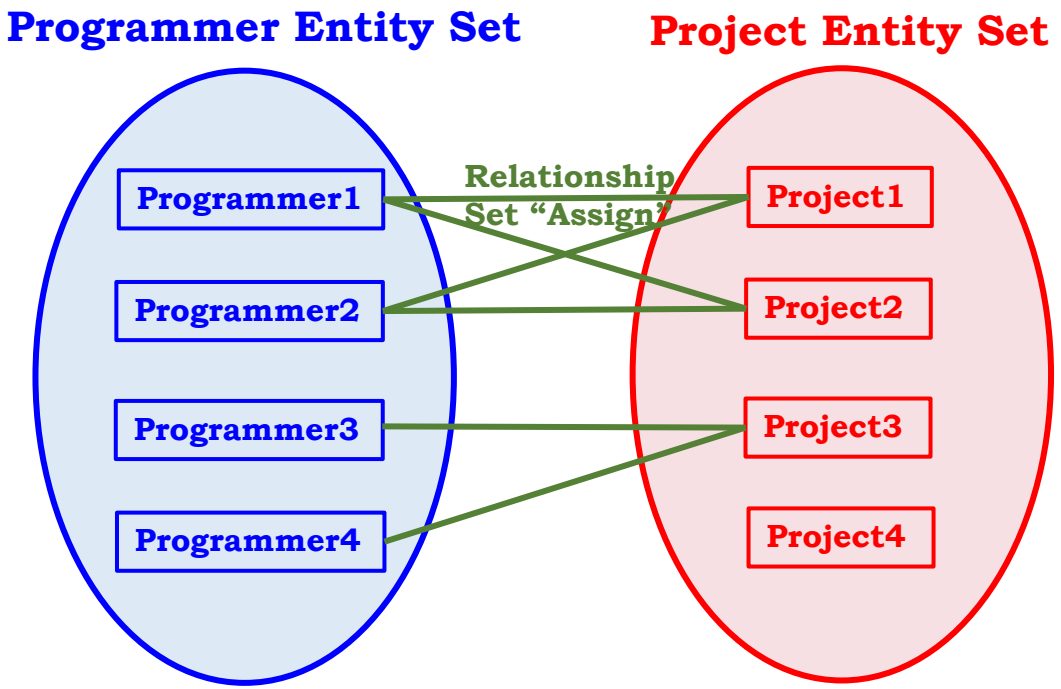


Entity Relationship Model

Mapping Cardinalities

Many-to-Many (M:M)

- ✔ An entity in Programmer is associated with any number (zero or more) of entities in Project, and an entity in Project is associated with any number (zero or more) of entities in Programmer.
- ✔ The following figure depicts mapping cardinality M:M



Entity Relationship Model

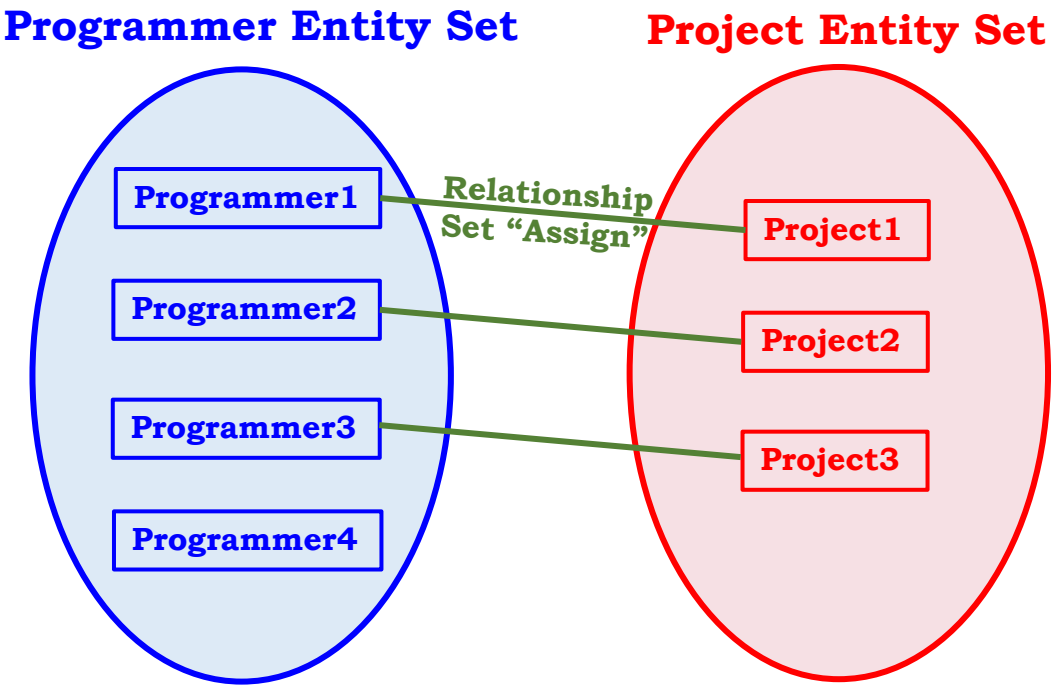
Participation Constraints

Total Participation :

The participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

Partial Participation :

If only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be partial.



Entity Relationship Model

Participation Constraints

Example :

- ✔ In Figure : A, the participation of Project Entity Set in the relationship set is total while the participation of A in the relationship set is partial.
- ✔ In Figure : B, the participation of both Programmer Entity Set and Project Entity Set in the relationship set are total.

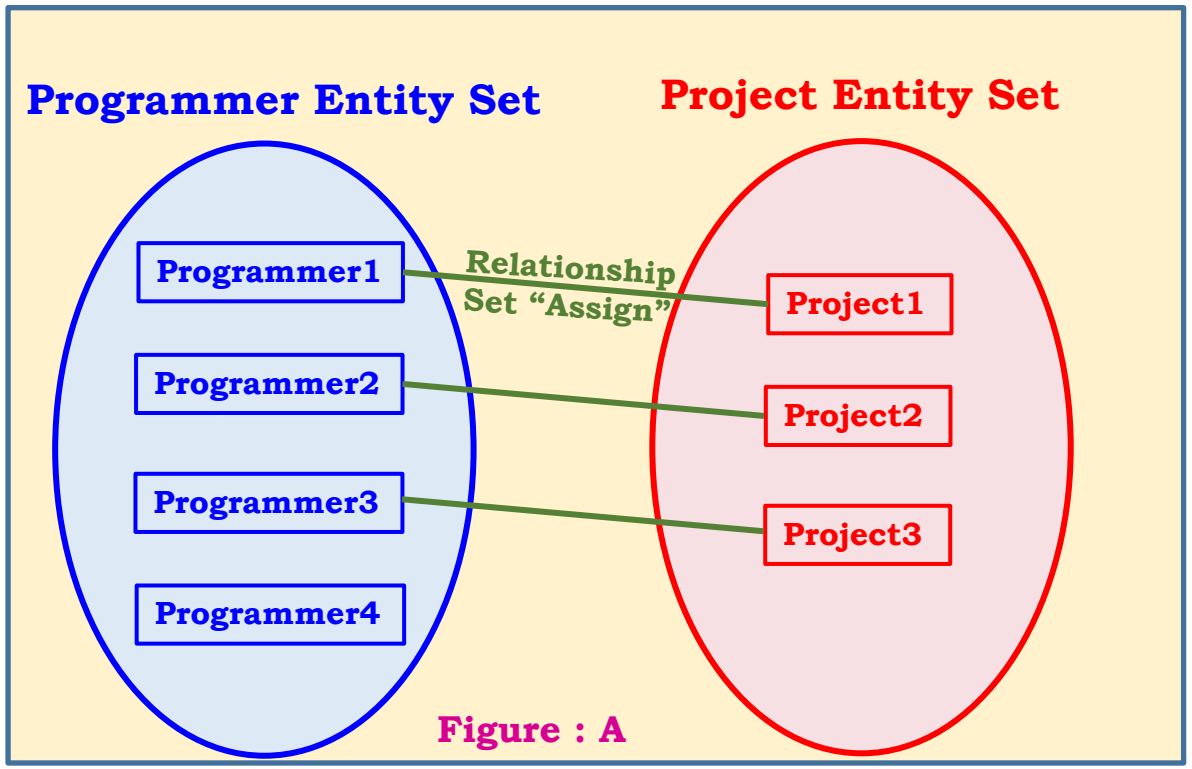


Figure : A

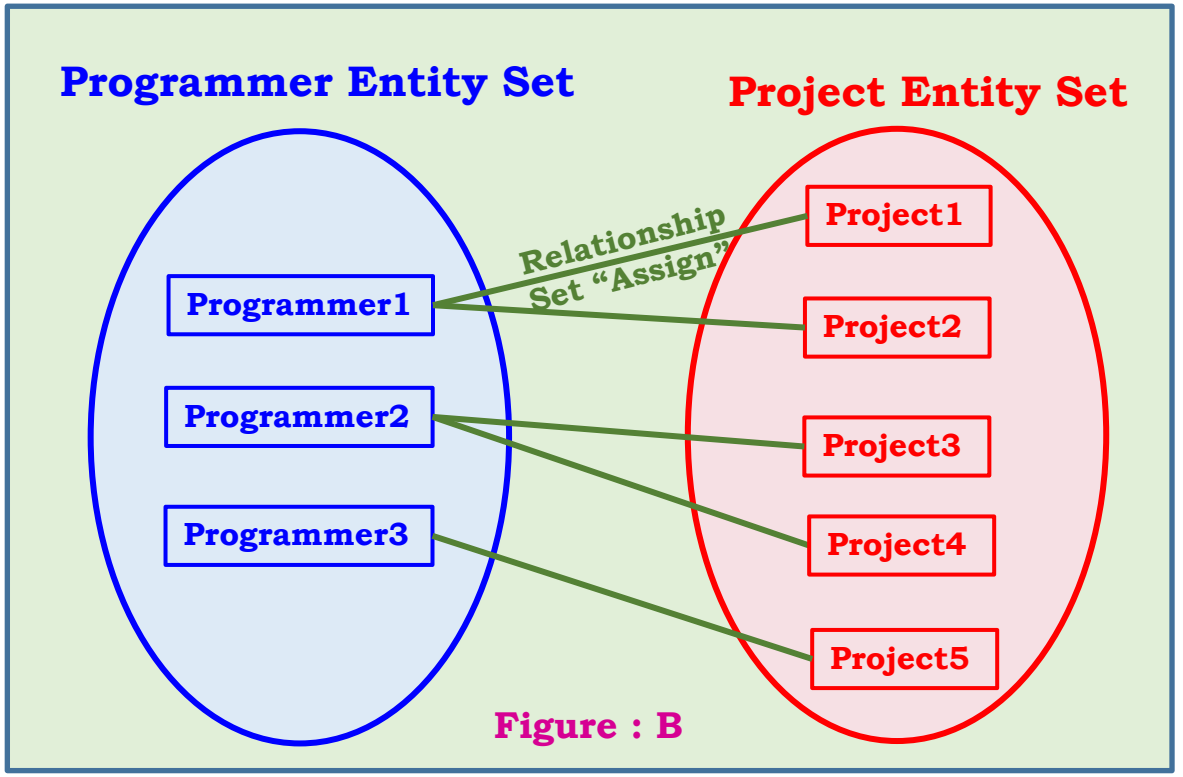
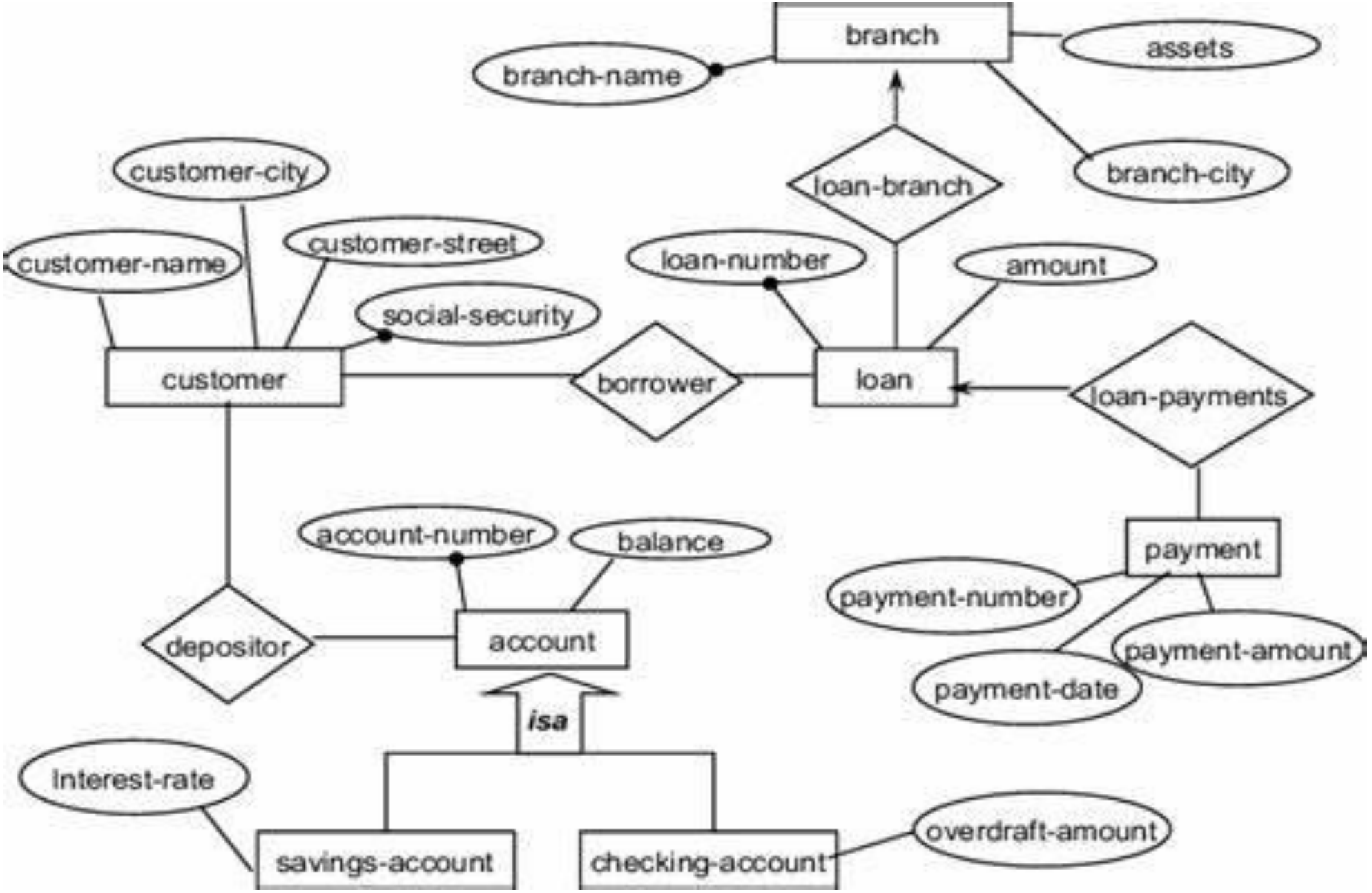
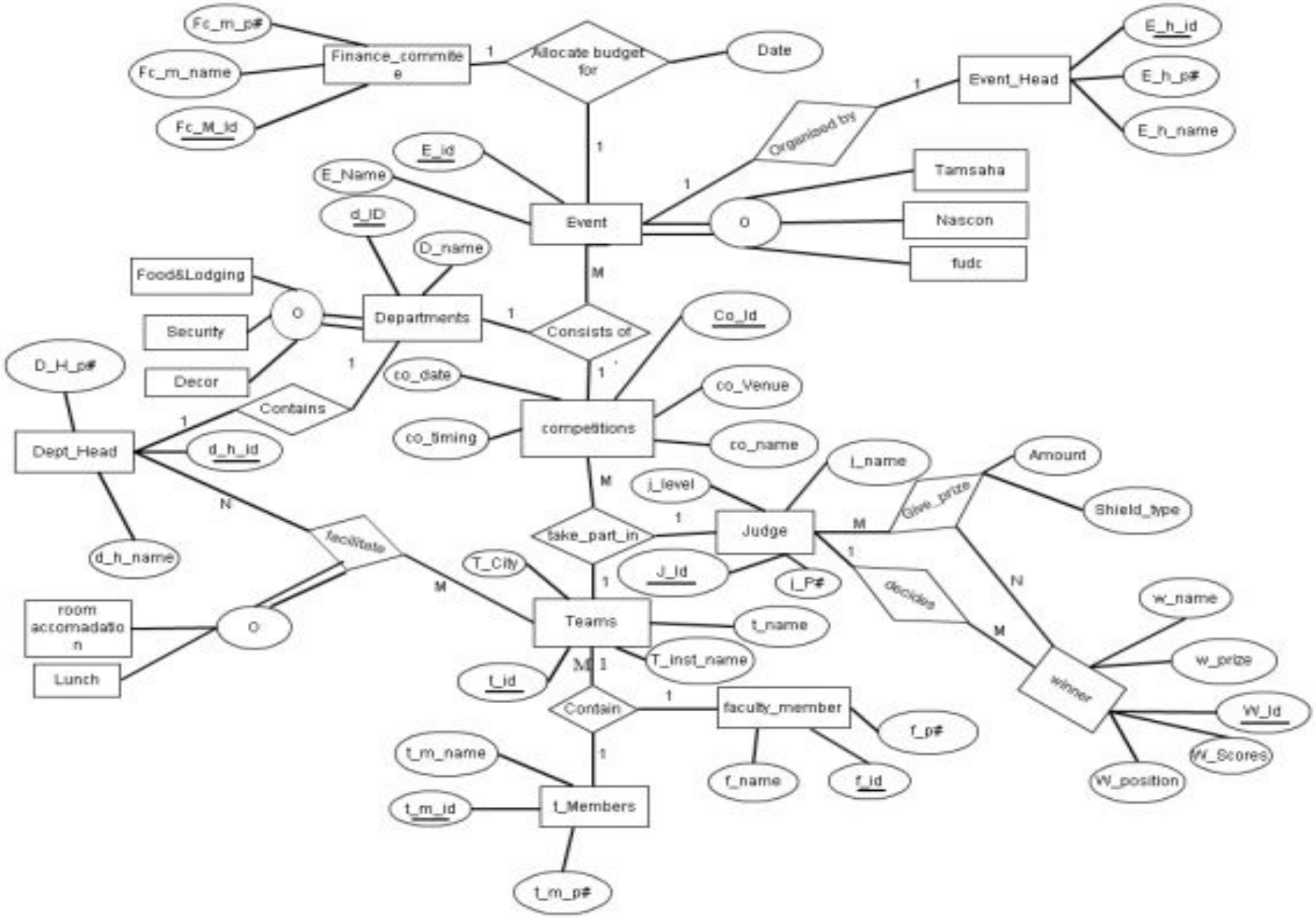


Figure : B


branch
assets
branch-name
branch-city
loan-branch
customer-city
customer-name
customer-street
social-security
customer
loan-number
amount
borrower
loan
loan-payments
payment
account-number
balance
payment-number
depositor
account
payment-amount
payment-date
isa
Interest-rate
savings-account
checking-account
overdraft-amount


Fc_m_p#
Fc_m_name
Fc_M_Id
Finance_committe e
1
Allocate budget for
Date
E_id
E_Name
d_ID
D_name
Event
1
1
Organised by
Event_Head
E_h_id
E_h_p#
E_h_name
O
Tamsaha
Nascon
fudc
M
Food&Lodging
Security
Decor
O
Departments
1
Consists of
Co_Id
1
1
D_H_p#
Contains
co_date
co_Venue
competitions
co_timing
co_name
1
Dept_Head
d_h_id
N
M
j_level
i_name
Amount
Give_prize
Shield_type
Judge
M
1
take_part_in
1
d_h_name
facilitate
T_City
J_Id
i_P#
decides
N
M
w_name
w_prize
W_Id
W_Scores
W_position
winner
room accomadatio n
Lunch
O
M
Teams
t_name
T_inst_name
t_id
M
1
Contain
1
faculty_member
f_p#
f_name
f_id
1
t_m_name
t_m_id
t_Members
t_m_p#

ER diagram

- ✔ E-R diagram can express the overall logical structure of a database graphically.
- ✔ E-R diagrams are simple and easy to understand

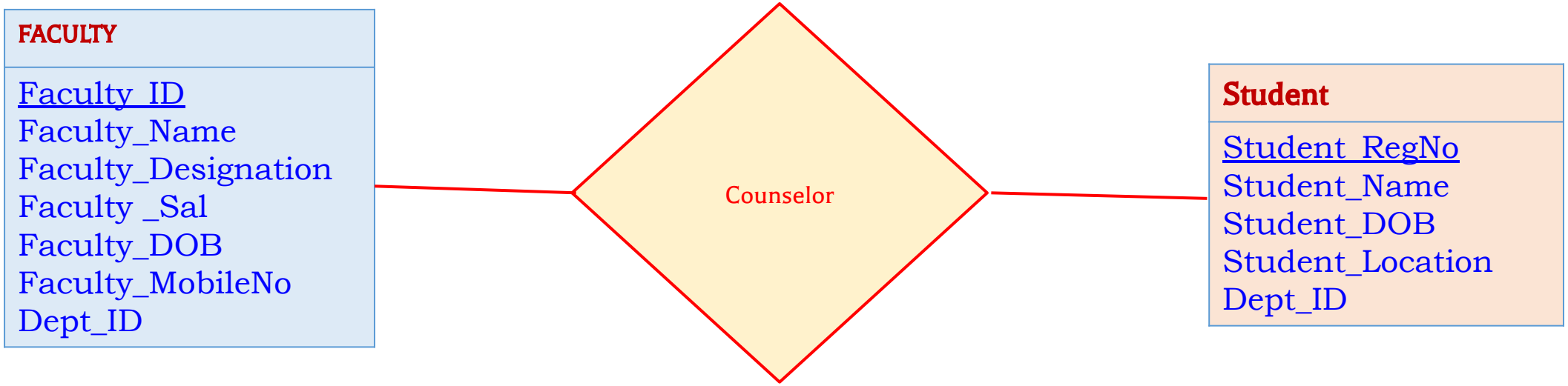
Basic Structure

E-R Diagram consists of following major components

- ✔ Rectangles divided into two parts represent entity sets. The first part contains the name of the entity set. The second part contains the names of all the attributes of the entity set.
- ✔ Diamonds represent relationship sets.
- ✔ Undivided rectangles represent the attributes of a relationship set. Attributes that are part of the primary key are underlined.
- ✔ Lines link entity sets to relationship sets.
- ✔ Dashed lines link attributes of a relationship set to the relationship set.
- ✔ Double lines indicate total participation of an entity in a relationship set.
- ✔ Double diamonds represent identifying relationship sets linked to weak entity sets

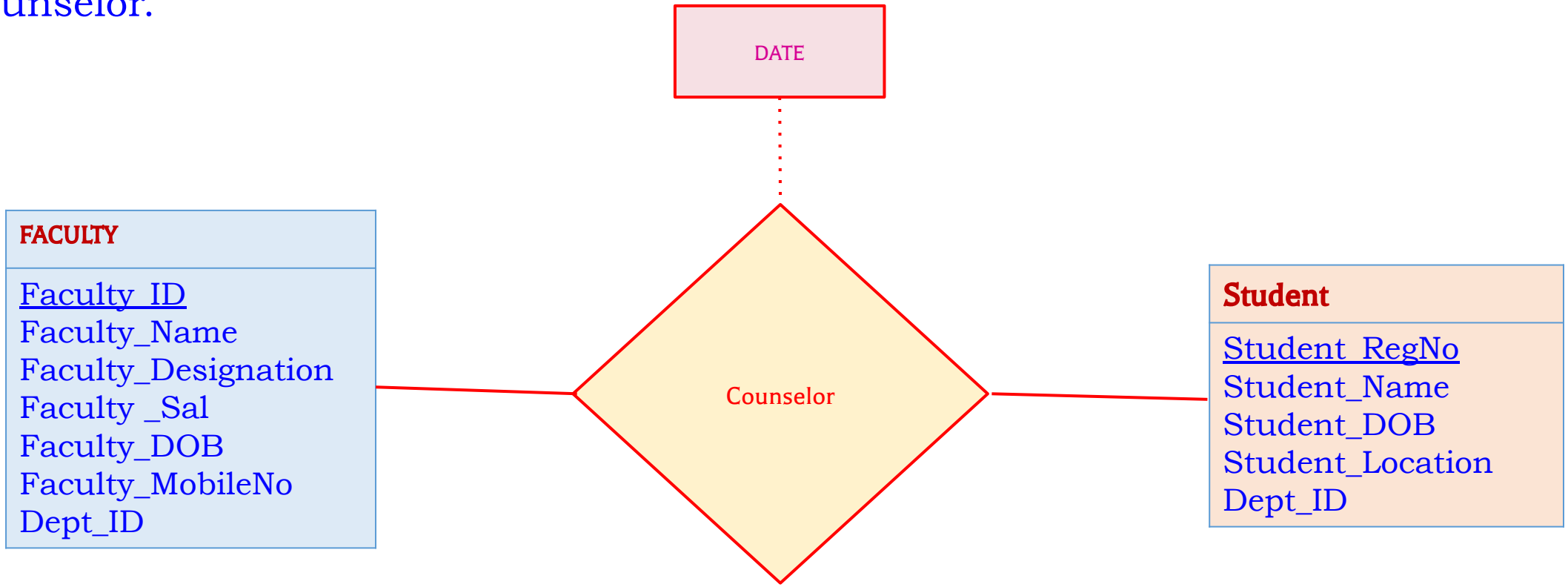
ER diagram

- Consider the E-R diagram in following figure, which consists of two entity sets, faculty and student related through a binary relationship set counselor.
- The attributes associated with faculty are Faculty_ID, Faculty_Name, Faculty_Designation, Faculty _Sal, Faculty_DOB ,Faculty_MobileNo, Dept_ID
- The attributes associated with student are Student_RegNo, Student_Name, Student_DOB, Student_Location, Dept_ID
- Attributes of an entity set that are members of the primary key are underlined.



ER diagram

- If a relationship set has some attributes associated with it, then we enclose the attributes in a rectangle and link the rectangle with a dashed line to the diamond representing that relationship set.
- For example, in the given figure, the date descriptive attribute attached to the relationship set counselor to specify the date on which the faculty became the counselor.



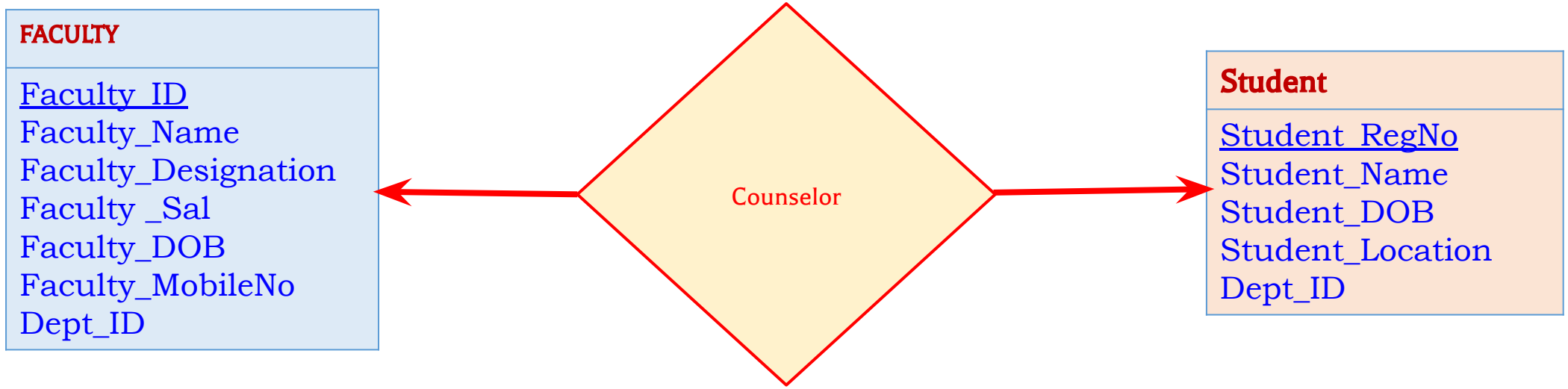
ER diagram

Mapping Cardinality

- ✔ The relationship set counselor, between the faculty and student entity sets may be one-to-one, one-to-many, many-to-one, or many-to-many.
- ✔ To distinguish among these types, we draw either a directed line (→) or an undirected line (—) between the relationship set and the entity.

One-to-one:

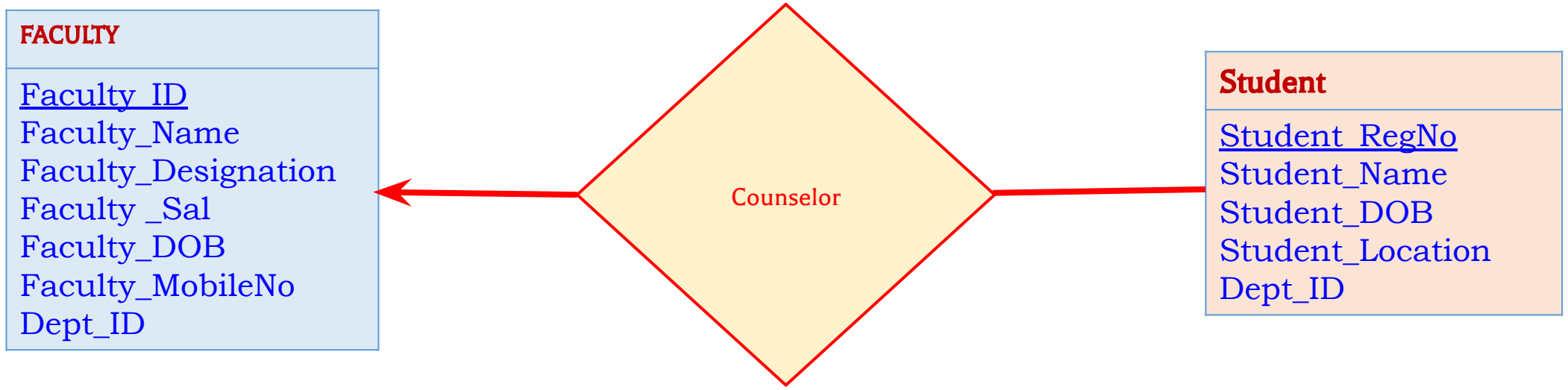
Line from the relationship set counselor to both entity sets faculty and student as given in the figure below. This indicates that a faculty may counsel at most one student, and a student may have at most one counselor.



Mapping Cardinality

One-to-many:

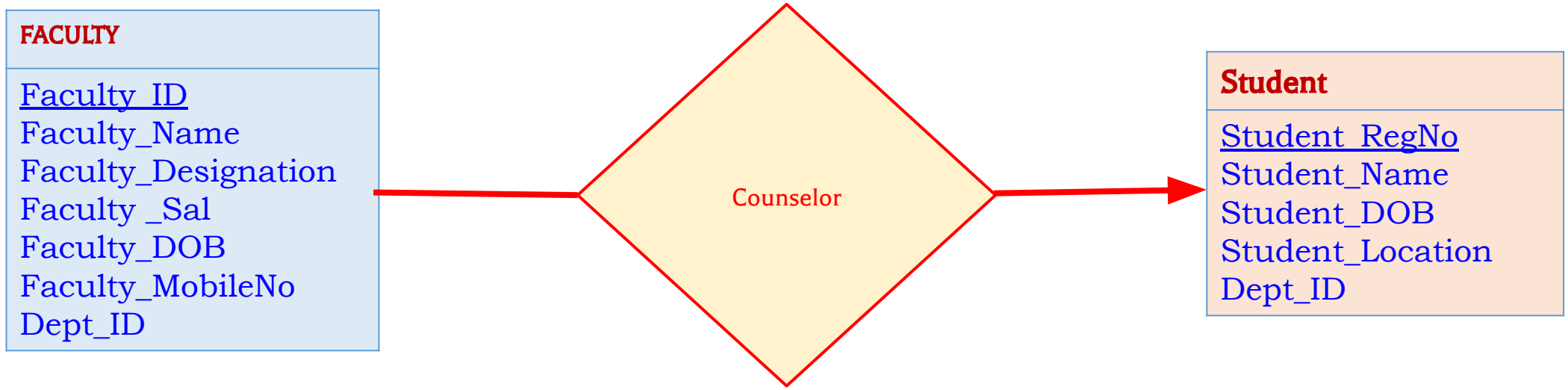
A directed line from the relationship set counselor to the entity set faculty and an undirected line to the entity set student as shown in the below figure, indicates that a faculty may counsel many students, but a student may have at most one counselor.



Mapping Cardinality

Many-to-one:

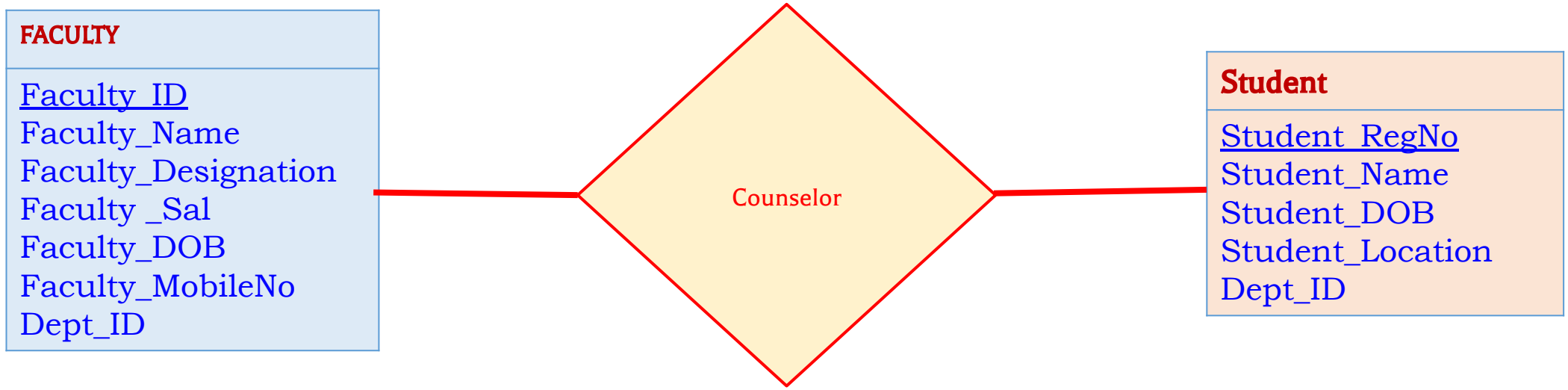
An undirected line from the relationship set counselor to the entity set faculty and a directed line to the entity set student as shown in the below figure, indicates that a faculty may counsel at most one student, but a student may have many counselors.



Mapping Cardinality

Many-to-many:

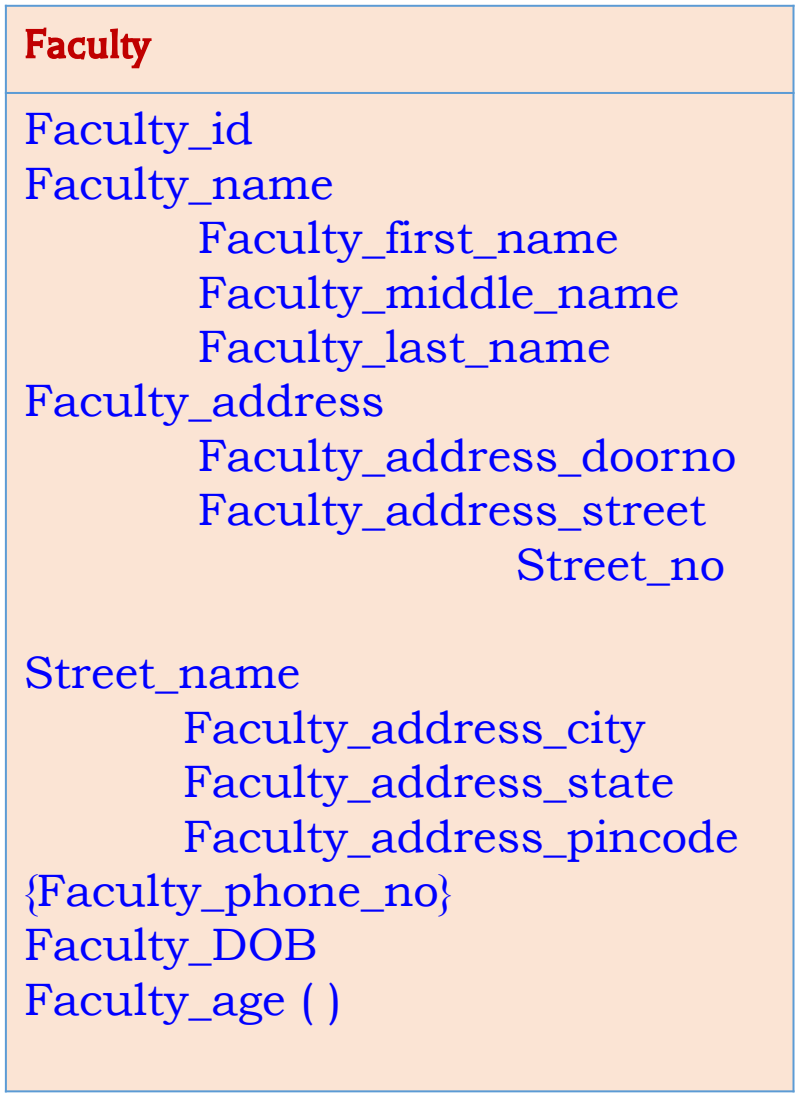
- We draw an undirected line from the relationship set counselor to both entity sets faculty and student as shown in the below figure, indicates that a faculty may counsel many students, and a student may have many counselor.



ER diagram

Complex Attributes

- ✔ Figure shows how composite attributes can be represented in the E-R notation.
- ✔ Here, a composite attribute Faculty_name, with component attributes Faculty_first_name, Faculty_middle_name, and Faculty_last_name replaces the simple attribute name of Faculty.
- ✔ As another example, An address to the Faculty entity-set. The address can be defined as the composite attribute Faculty_address with the attributes street, city, state, and pincode.
- ✔ The attribute street is itself a composite attribute whose component attributes are Faculty_street_no and Faculty_street name.
- ✔ The given figure also illustrates a multivalued attribute phone number, denoted by "{ Faculty_phone_no }".
- ✔ A derived attribute age, depicted by a "Faculty_age ( )".



Keys , Attributes and Constraints

Keys

- An entity should be identified in an entity set uniquely.
- It is expressed in terms of their attributes
- The values hold by attributes must identify the record / tuple uniquely.
- No two records in relation are not allowed to hold exactly the same values for all attributes.

Superkey

- A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely a record in the relation.
- For example, the Faculty_ID attribute of the relation faculty is sufficient to distinguish one faculty record from another.
- Here Faculty_ID is the superkey.
- The Faculty_name attribute of Faculty, on the other hand, is not a superkey, because many faculty might have the same name.

Keys , Attributes and Constraints

Superkey

- Let R denote the set of attributes in the schema of relation r. If we say that a subset K of R is a superkey for r.
- We are restricting consideration to instances of relations r in which no two distinct tuples have the same values on all attributes in K.
- That is, if t1 and t2 are in r and t1 = t2, then t1.K = t2.K.
- A superkey may contain extraneous attributes. For example, the combination of Faculty_ID and Faculty_name is a superkey for the relation Faculty.
- Minimal of Superkeys are called as Candidate key.
- It is possible that several distinct set of attributes could serve as a Candiadate key

Keys , Attributes and Constraints

Superkey

- Suppose that a combination of Faculty_name and Dept_name is sufficient to distinguish among members of the Faculty relation.
- Then, both {Faculty_ID} and {Faculty_name, Dept_name} are candidate keys.
- Although the attributes Faculty_ID and Faculty_name together can distinguish faculty tuples, their combination, {Faculty_ID, Faculty_name}, does not form a candidate key, since the attribute Faculty_ID alone is a candidate key.
- The term primary key is to denote a candidate key.
- A key (whether primary, candidate, or super) is a property of the entire relation, rather than of the individual tuples.
- The designation of a key represents a constraint in the real-world enterprise being modeled.

Keys , Attributes and Constraints

Superkey

- The Primary key should be selected with special care.
- As we discussed the name of the person is obviously not sufficient to identify uniquely a person , because many persons can have the same name.
- In India , now the Aadhar card number attribute would be a primary key / candidate key.
- Non resident of India will not have the Aadhar number .
- An alternative is to use some unique combination of other attributes as a key.

Keys , Attributes and Constraints

Superkey

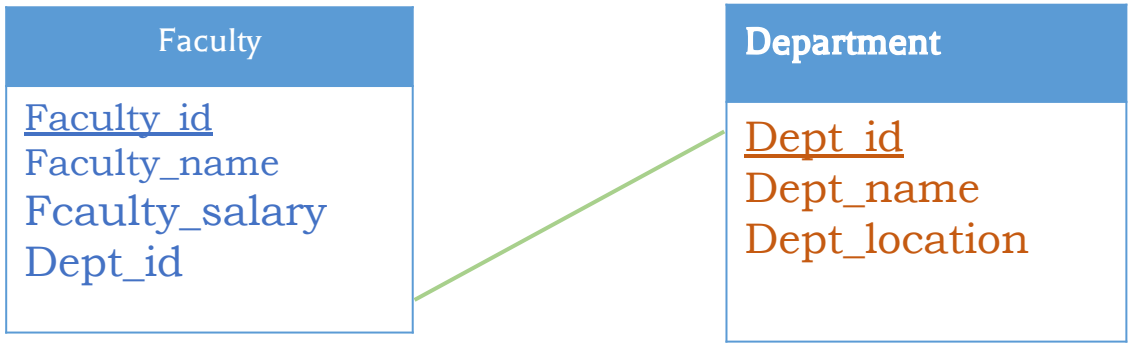
- The primary key should be chosen such that its attribute values are never,or very rarely, changed.
- For example , the address field should not be a primary key or part of primary key, since it is likely to change but, Aadhar number guaranteed never to change.
- To represent the primary key , the primary key attributes are underlined
- A relation, say r1, may include among its attributes the primary key of an other relation, say r2. This attribute is called a foreign key from r1, referencing r2.
- The relation r1 is also called the referencing relation of the foreign key dependency, and r2 is called the referenced relation of the foreign key.

Note : A primary key for a particular relation/ table is act as an referential key in another table (s) is called foreign key , it known as referential integrity constraints

Keys , Attributes and Constraints

Superkey

- Consider the two entitiy sets named : Faculty and Department
- For Faculty entity set the primary key is : Faculty_id
- For Department entity set the primary key is : Dept_id
- In this relations, Dept_id in the Department relation , is the referential key or foreign key for the Faculty relation.
- Primary key in a relations is underlined
- Only one primary key is possible for a relation
- One or more attributes can be combined and declared as a primary key , known as composite primary key. (Note : Maximum 16 Columns are allowed)



Keys , Attributes and Constraints

Attributes

- ✔ Attributes are the properties of an entity
- ✔ Attributes are used to describe about an entity
- ✔ The type of attributes are
  - Simple attributes
  - Composite attributes
  - Single valued attributes
  - Multi valued attributes
  - Derived attributes
  - Key attributes

Keys , Attributes and Constraints

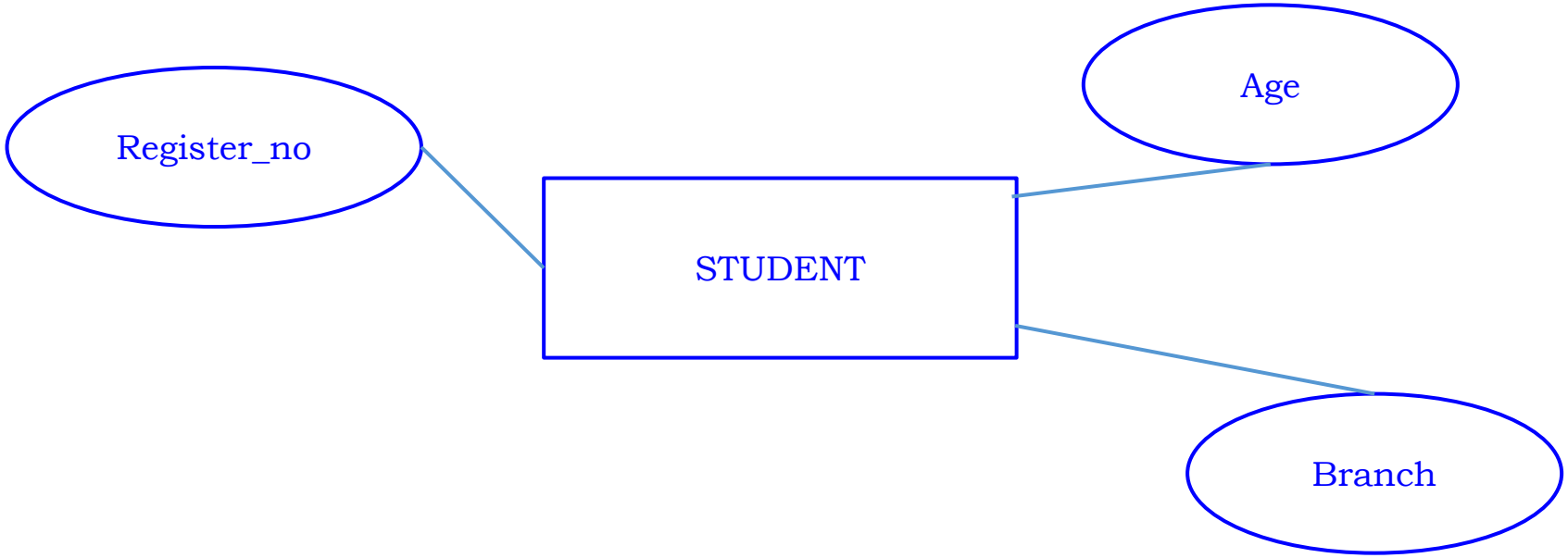
Simple attributes

It can not be divided further

All the simple attributes will hold the atomic values

Example :

Student = { Register_no, Name, ... }

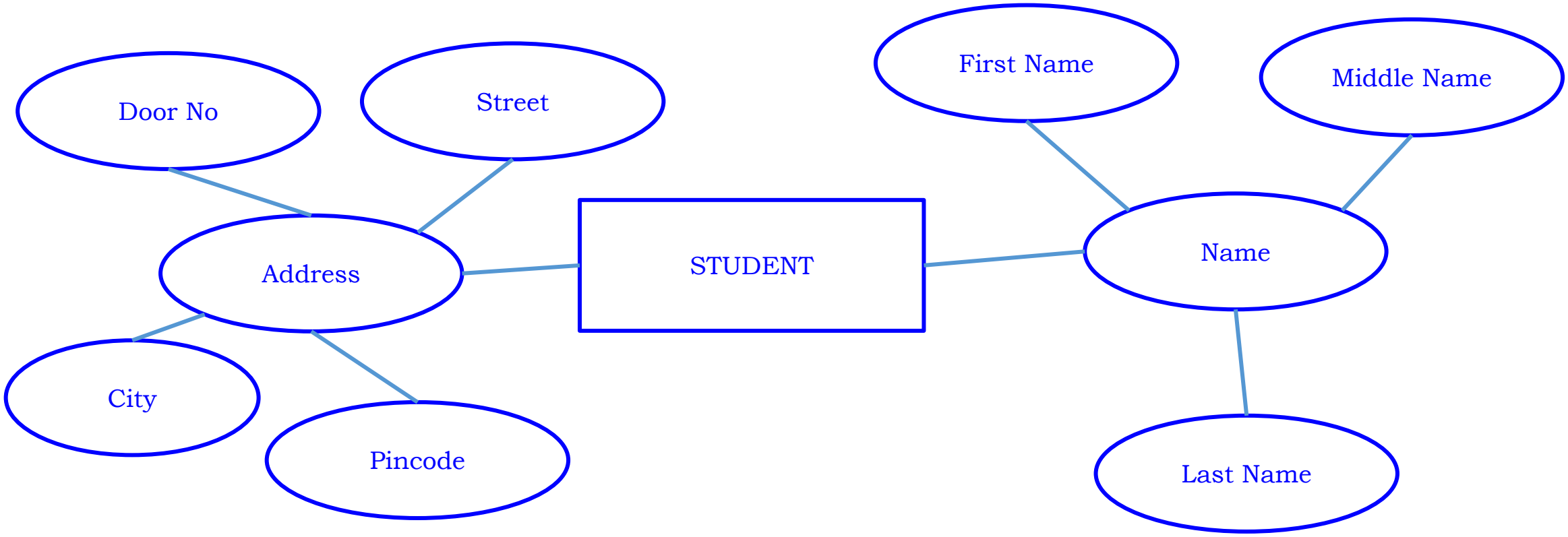


Keys , Attributes and Constraints

Composite attributes

Composed by many other simple attributes

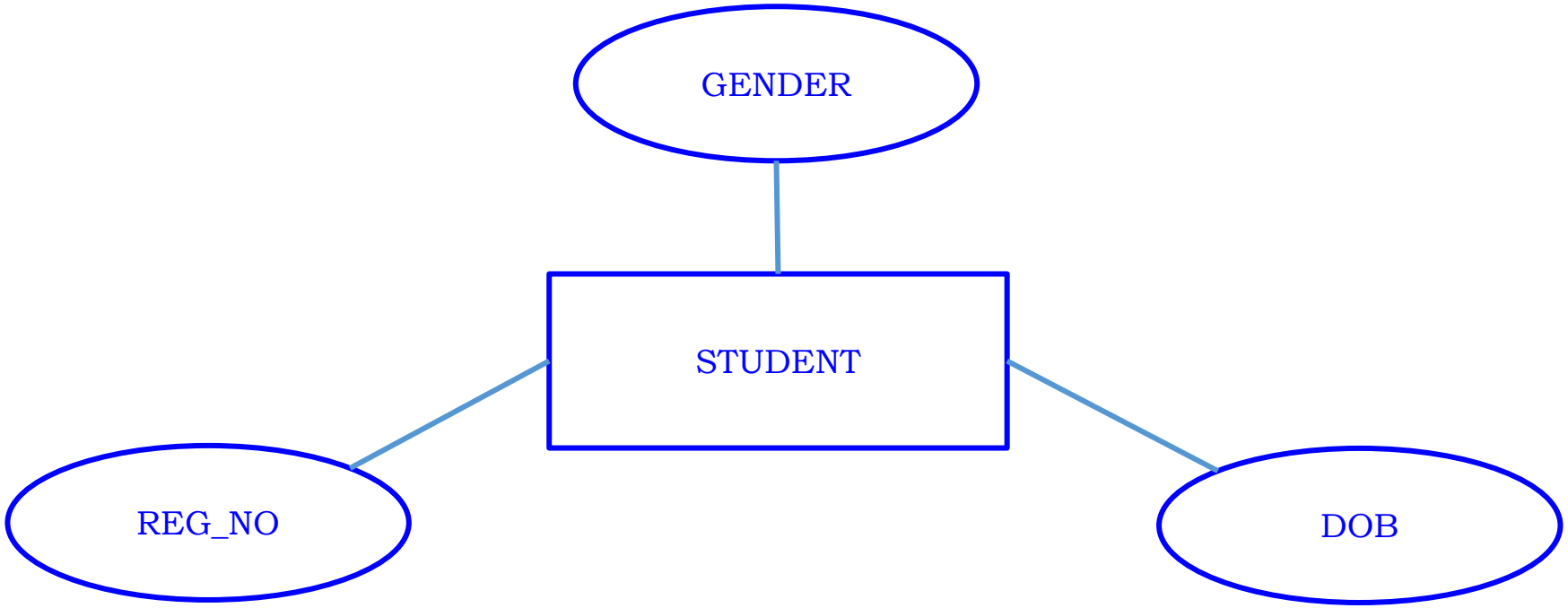
Example : Address , Name , etc.,



Keys , Attributes and Constraints

Single valued attributes

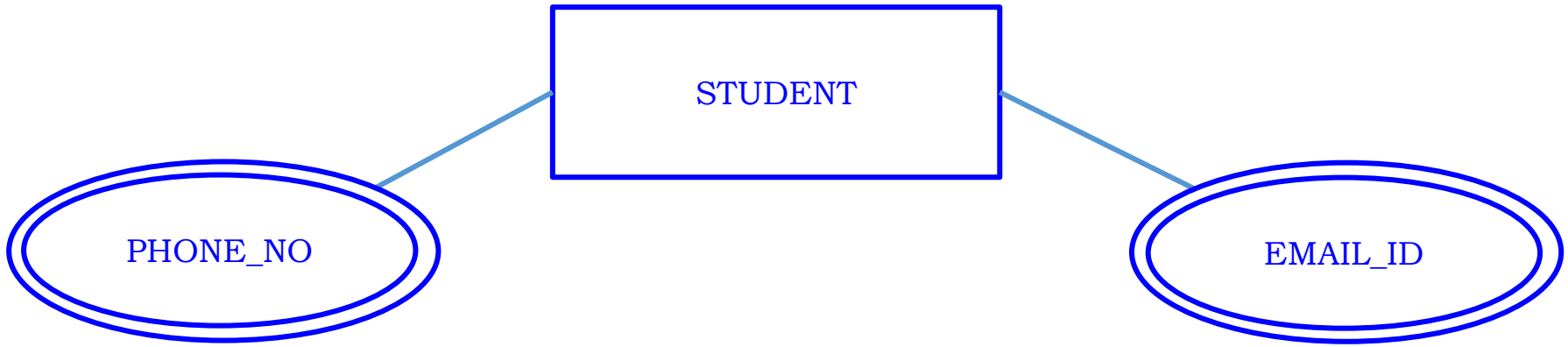
- Single valued attributes are those attributes which can take only one value for a given entity from an entity set.
- Example : Gender , DOB, Reg_No



Keys , Attributes and Constraints

Multi valued attributes

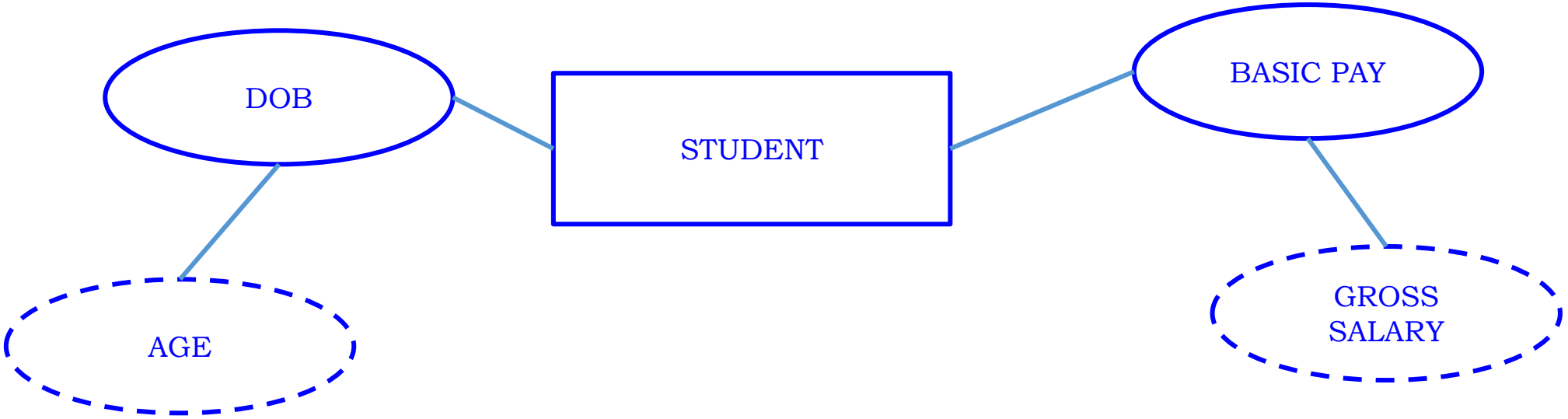
- ✔ Attributes can hold more than one values are called multi valued attribute
- ✔ Example : Phone_no, Email_id



Keys , Attributes and Constraints

Derived attributes

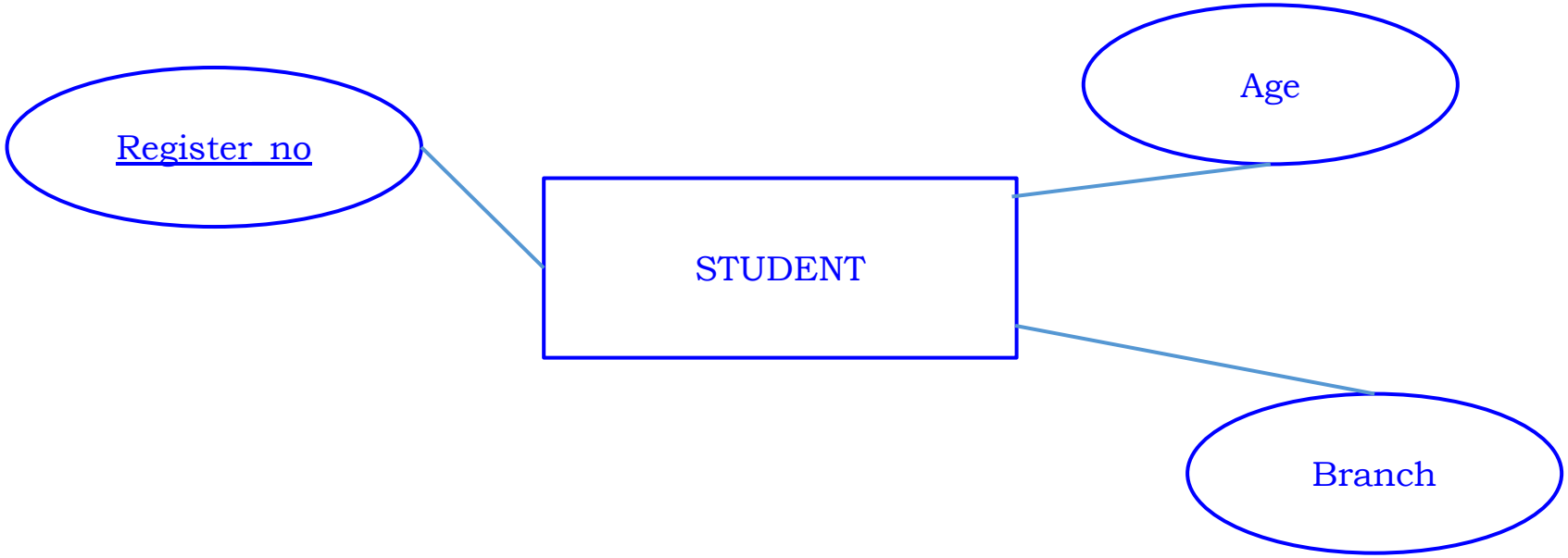
- A value which is derived from already existing value.
- It is not advisable to store such kind of values in database.
- The derived attributes represented by ellipse using dotted lines
- Example : Age , Gross Salary
- In the given figure below, Age is derived from DOB and Gross Salary derived from Basic Pay



Keys , Attributes and Constraints

Key attributes

- Attributes which is used to identify an entity in an entity set is called Key attributes
- Key attributes are represented by underline the name of the attribute.
- In the given figure , In Student entity the attribute Register_no is key attribute used to identity each student uniquely.



Keys , Attributes and Constraints

- Constraints
- It is a condition to manage the consistency as well integrity of the values stored in an attribute.
- Constraints specified at the time of designing relations is good choice
- There are two types of Constraints
- Domain Constraints
  - Not Null
  - Check
  - Unique
  - Primary key
- Integrity Constraints
  - Referential key or Foreign key

Keys , Attributes and Constraints

Domain Constraints

Not Null :

(NOTE : By default ,an attribute hold NULL values)

If an attribute holds not null constraint

- ✔ The value should be inserted
- ✔ It will not accept “NULL” values
- ✔ It will accept Duplicate values
- ✔ N number of not null constraints is possible in a relation
- ✔ While inserting a new record the not null must be entered otherwise , insertion of new record is not possible
- ✔ Example : Student entity defined with not null constraint for an attribute Register_no

```
CREATE TABLE STUDENT (
                    Register_no Number(10) NOT NULL,
                    LastName varchar(25) ,
                    FirstName varchar(25),
                    DOB Date );
```

Keys , Attributes and Constraints

Domain Constraints

Check :

- ✔ Check Constraints check the condition specified in the create statement.
- ✔ If the condition satisfied then the value will be inserted , otherwise will not be permitted.
- ✔ It allows NULL values
- ✔ It allows duplicate values
- ✔ Example : The emp entity created with check constraint for an attribute “Salary” should be greater than 10000.

```
CREATE TABLE emp ( empno number (10) Not null,
                   Ename varchar2(25),
                   ……,
                   ……,
                   Salary number(10,2) Check (Salary > 10000);
```

Keys , Attributes and Constraints

Domain Constraints

Unique:

- ✔ To maintain the distinct values in an attribute of an entity set , UNIQUE constraint is used.
- ✔ It will not accept duplicate values.
- ✔ It will accept NULL values .
- ✔ It will accept N number of null values , because two null values are always not equal.
- ✔ A relation can have N number of unique constraints.
- ✔ Example : A Student entity is created with unique constraint for an attribute Register_no

```
CREATE TABLE STUDENT (
    Register_no Number(10) Unique,
    LastName varchar(25) ,
    FirstName varchar(25),
    DOB Date );
```

Note : An attribute can hold one or more constraints

```
CREATE TABLE STUDENT (
    Register_no Number(10) Not null Unique,
    LastName varchar(25),FirstName varchar(25), DOB Date );
```

Keys , Attributes and Constraints

Domain Constraints

Primary key

- ✔ Minimal of super key is known as Candidate key.
- ✔ Candidate key represented as PRIMARY KEY
- ✔ A relation can have only one primary key
- ✔ Combination of one or more (Maximum 16 Nos) attributes can be declared as primary key.
- ✔ It will not accept both null values and duplicate values.
- ✔ Primary key is the combination of Not null and Unique constraints.
- ✔ Primary key can act as a referential key for another table called child table.
- ✔ Example: A Student entity created with primary key constraint for an attribute Register_no

```
CREATE TABLE STUDENT (
    Register_no Number(10) Primary key,
    LastName varchar(25) ,
    FirstName varchar(25),
    DOB Date );
```

Keys , Attributes and Constraints

Integrity Constraints

Referential Integrity / Foreign key Constraints

- ✔ A primary key will be a referential key for another table is called as referential integrity / foreign key constraints.
- ✔ Foreign key allows only the values available in referential key (Primary key).
- ✔ It allows duplicate values and null values.
- ✔ It allows N number of null values.
- ✔ Example : An entity emp created with foreign key constraint referencing dept entity primary key attribute dept_id.

```
CREATE TABLE emp ( empno number (10) Primary key,
                   Ename varchar2(25),
                   ……….,
                   ……….,
                   Salary number(10,2) Check (Salary > 10000),
                   Dept_id references DEPT (DEPT_ID);
```

Note : The geiven emp entity , primary key attribute is empno and foreign key is dept_id which is the primary key in dept entity.

Keys , Attributes and Constraints

An overview of Constraints

CONSTRAINTS	NULL VALUES	DUPLICATE VALUES	CHECKING THE CONDITION	REFERENTIAL KEY
NOT NULL	NO	YES	YES	NO
CHECK	YES	YES	YES	NO
UNIQUE	YES	NO	YES	NO
PRIMARY KEY	NO	NO	YES	YES
FOREIGN KEY	YES	YES	YES	NO

Mapping Cardinalities

- ✔ Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.
- ✔ Mapping cardinalities are most useful in describing binary relationship sets.
- ✔ For a binary relationship set “Assign” between entity sets Programmer and Project the mapping cardinality must be one of the following.
  - One-to-One (1:1)
  - One-to-Many (1:M)
  - Many-to-One (M:1)
  - Many-to-Many (M:M)

Extended ER - Generalization, Specialization and Aggregation

Extended ER Features

- ✔ Basic ER Model is more than enough to model most of the Database Features.
- ✔ Extended ER model developed for some aspects of Database features more suitably expressed
- ✔ The followings are the Extended ER Features
  - Specialization
  - Generalization
  - Higher and lower level entity sets
  - Attribute inheritance
  - Aggregation
- ✔ To explain the above concepts, slightly more elaborate the schema for the university, by considering an entity set “person” with attributes “id”, “name”, and “address”

Extended ER - Generalization, Specialization and Aggregation

Specialization

- An entity set may include subgroupings of entities that are distinct in some way from other entities in the set.
- a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set.
- The E-R model provides a means for representing these distinctive entity groupings.
- The Entity set person may be further classified as one of the following:
  - Employee
  - Student

Extended ER - Generalization, Specialization and Aggregation

Specialization

- Both employee and student is described by a set of attributes that includes all the attributes of entity set person plus possibly additional attributes.
- For example, employee entities may be described further by the attribute salary, whereas student entities may be described further by the attribute fees.
- The process of designating subgroupings within an entity set is called specialization.
- The specialization of person allows us to distinguish among person entities according to whether they correspond to employees or students:
- In general, a person could be an employee, a student, both, or neither.

Extended ER - Generalization, Specialization and Aggregation

Specialization

- As another example, suppose the university divides students into two categories: Under graduate and Post graduate.
- Under graduate students have an office assigned to them. Post graduate students are assigned to a residential college.
- Each of these student types is described by a set of attributes that includes all the attributes of the entity set student plus additional attributes.
- The university could create two specializations of student, namely under graduate and post graduate.

Extended ER - Generalization, Specialization and Aggregation

Specialization

- ✔ We can apply specialization repeatedly to refine a design.
- ✔ For instance, university employees may be further classified as one of the following:
  - Faculty
  - Secretary
- ✔ Each of these employee types is described by a set of attributes that includes all the attributes of entity set employee plus additional attributes.
- ✔ For example, faculties entities may be described further by the attribute designation while secretary entities are described by the attribute hours per week.
- ✔ Further, secretary entities may participate in a relationship secretary for between the secretary and employee entity sets, which identifies the employees who are assisted by a secretary.

Extended ER - Generalization, Specialization and Aggregation

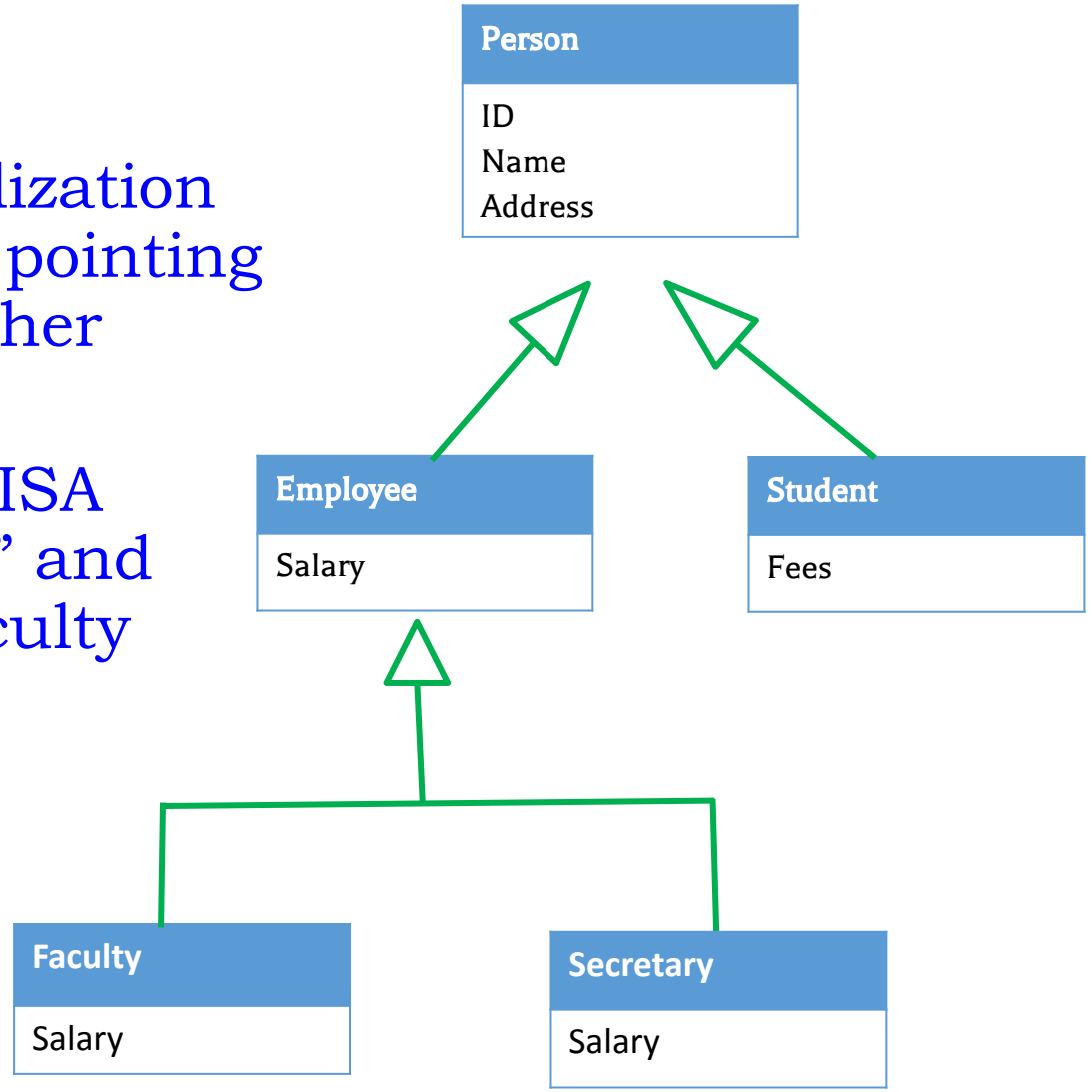
Specialization

- An entity set may be specialized by more than one distinguishing feature.
- In our example, the distinguishing feature among employee entities is the job the employee performs.
- Another, coexistent, specialization could be based on whether the person is a temporary employee or a permanent employee?
- Resulting in the entity sets temporary employee and permanent employee.

Extended ER - Generalization, Specialization and Aggregation

Specialization

- In terms of an E-R diagram, specialization is depicted by a hollow arrow-head pointing from the specialized entity to the other entity
- We refer to this relationship as the ISA relationship, which stands for “is a” and represents, for example, that an faculty “is a” employee.



Extended ER - Generalization, Specialization and Aggregation

Specialization

- Specialization represents in an E-R diagram depends on whether an entity may belong to multiple specialized entity sets or if it must belong to at most one specialized entity set.
- Multiple sets permitted is called overlapping specialization
- At most one permitted is called disjoint specialization.
- For an overlapping specialization (refer the figure in slide number 87 for student and employee as specializations of person), two separate arrows are used.
- For a disjoint specialization (refer the figure in slide number 87 for faculty and secretary as specializations of employee), a single arrow is used.
- The specialization relationship may also be referred to as a superclass-subclass relationship.

Extended ER - Generalization, Specialization and Aggregation

Generalization

- The refinement from an initial entity set into successive levels of entity subgroupings represents a top-down design process in which distinctions are made explicit.
- The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.
- The database designer may have first identified:
  - Faculty entity set with attributes Faculty_id, Faculty_name, Faculty_salary, and Faculty_Desig.
  - Secretary entity set with attributes secretary_id, secretary_name, secretary_salary, and hours_per_week.

Extended ER - Generalization, Specialization and Aggregation

Generalization

- There are some similarities are between the Faculty entity and Secretary entity, means several attributes that are conceptually the same across the two entity sets.
- For example, the identifier, name, and salary attributes are common between Faculty and Secretary entities.
- This commonality can be expressed by Generalization.
- Generalization is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets.
- In given example (slide number 85) ,employee is the higher-level entity set and faculty and secretary are lower-level entity sets.
- Higher- and lower-level entity sets also may be designated by the terms superclass and subclass, respectively.
- The person entity set is the superclass of the employee and student subclasses.

Extended ER - Generalization, Specialization and Aggregation

Attribute Inheritance

- An important property of the higher- and lower-level entities created by specialization and generalization is attribute inheritance.
- The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.
  - Example, student and employee inherit the attributes of person.
- Student entity is described by its ID, name, and address attributes, and additionally a fees attribute.
- Employee is described by its ID, name, and address attributes, and additionally a salary attribute.
- Attribute inheritance applies through all tiers of lower-level entity sets.
  - Example : Faculty and Secretary, which are subclasses of employee, inherit the attributes ID, name, and address from person, in addition to inheriting the attribute salary from employee.

Extended ER - Generalization, Specialization and Aggregation

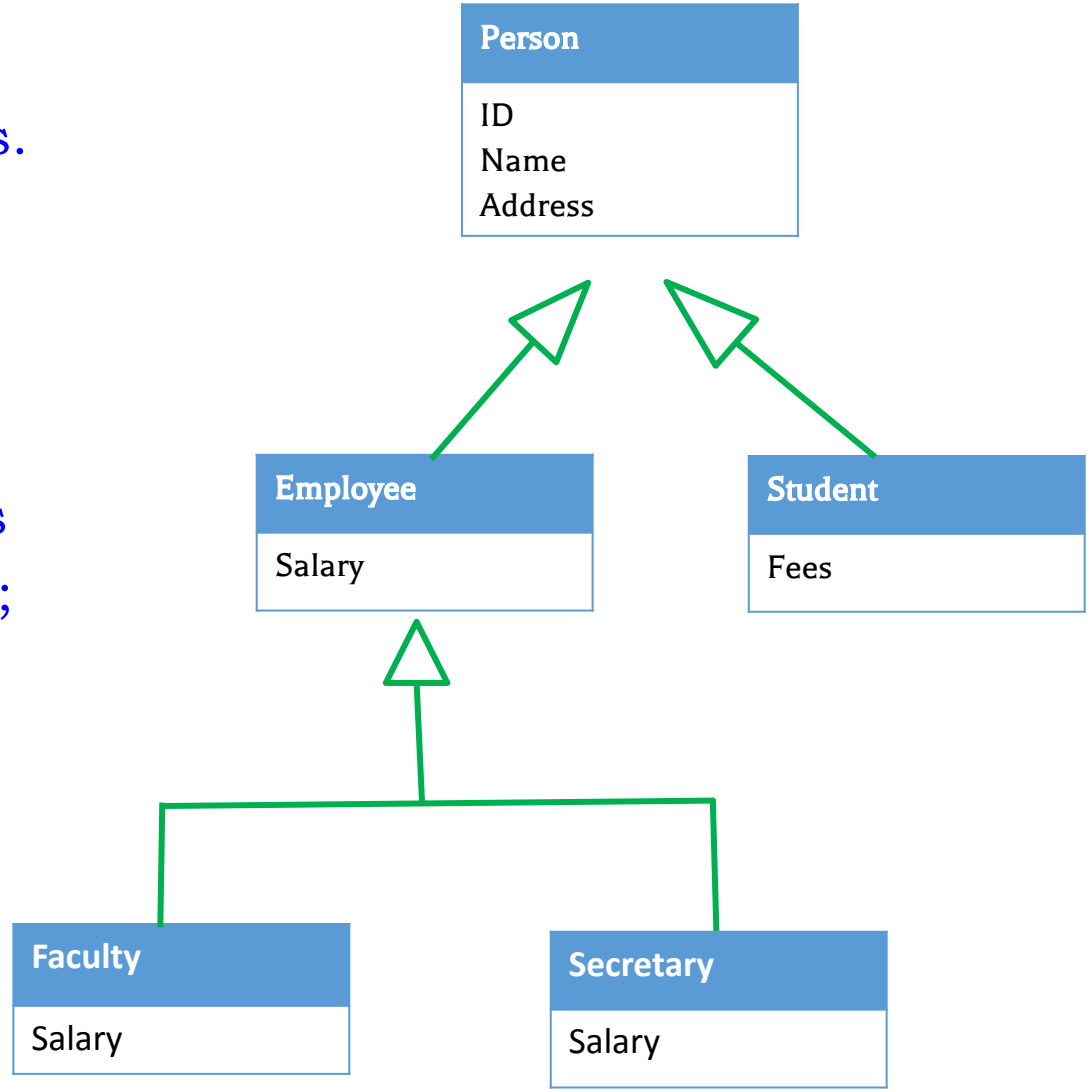
Attribute Inheritance

- An E-R model was arrived at by specialization or generalization, the outcome is basically the same:
  - A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets.
  - Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set.

Extended ER - Generalization, Specialization and Aggregation

Attribute Inheritance

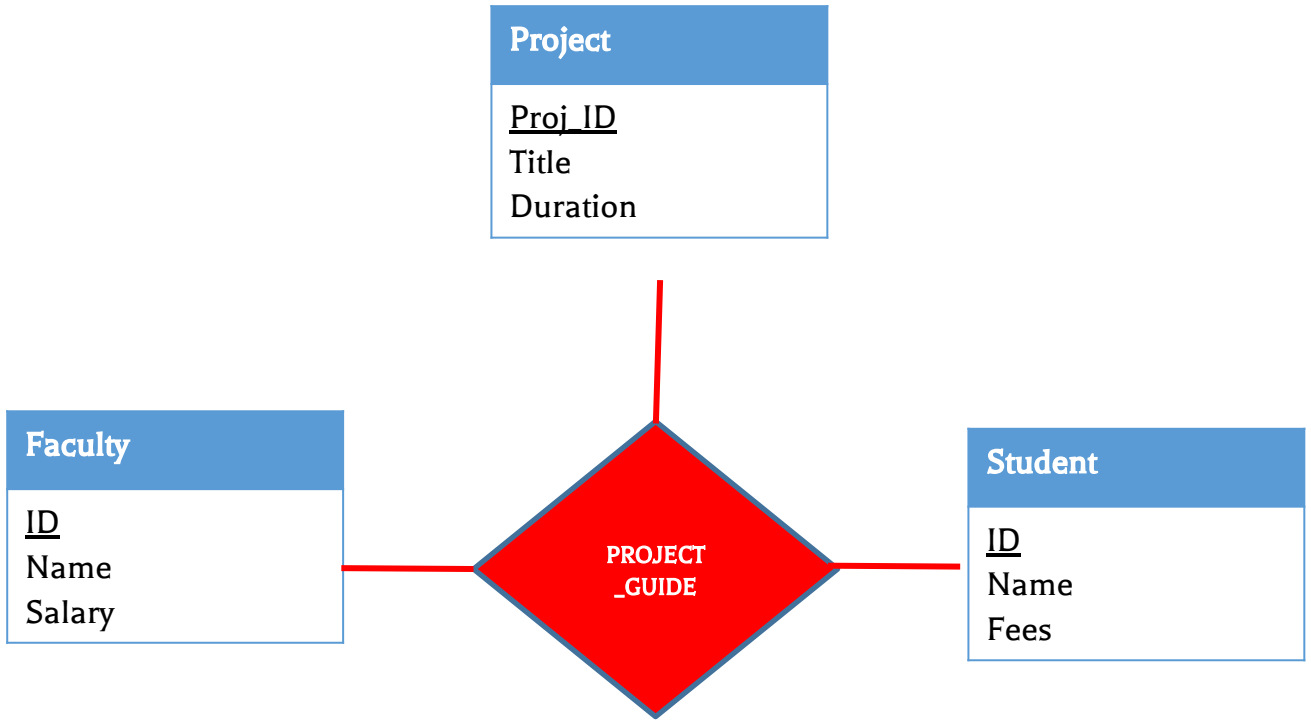
- ✔ The given Figure describes a hierarchy of entity sets.
- ✔ In the figure, employee is a lower-level entity set of person and a higher-level entity set of the faculty and secretary entity sets.
- ✔ In a hierarchy, a given entity set may be involved as a lower- level entity set in only one ISA relationship; that is, entity sets in this diagram have only single inheritance.
- ✔ If an entity set is a lower-level entity set in more than one ISA relationship, then the entity set has multiple inheritance, and the resulting structure is said to be a lattice.



Extended ER - Generalization, Specialization and Aggregation

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- To illustrate the need for such a construct, consider the ternary relationship project_guide, between an faculty, student and project



Extended ER - Generalization, Specialization and Aggregation

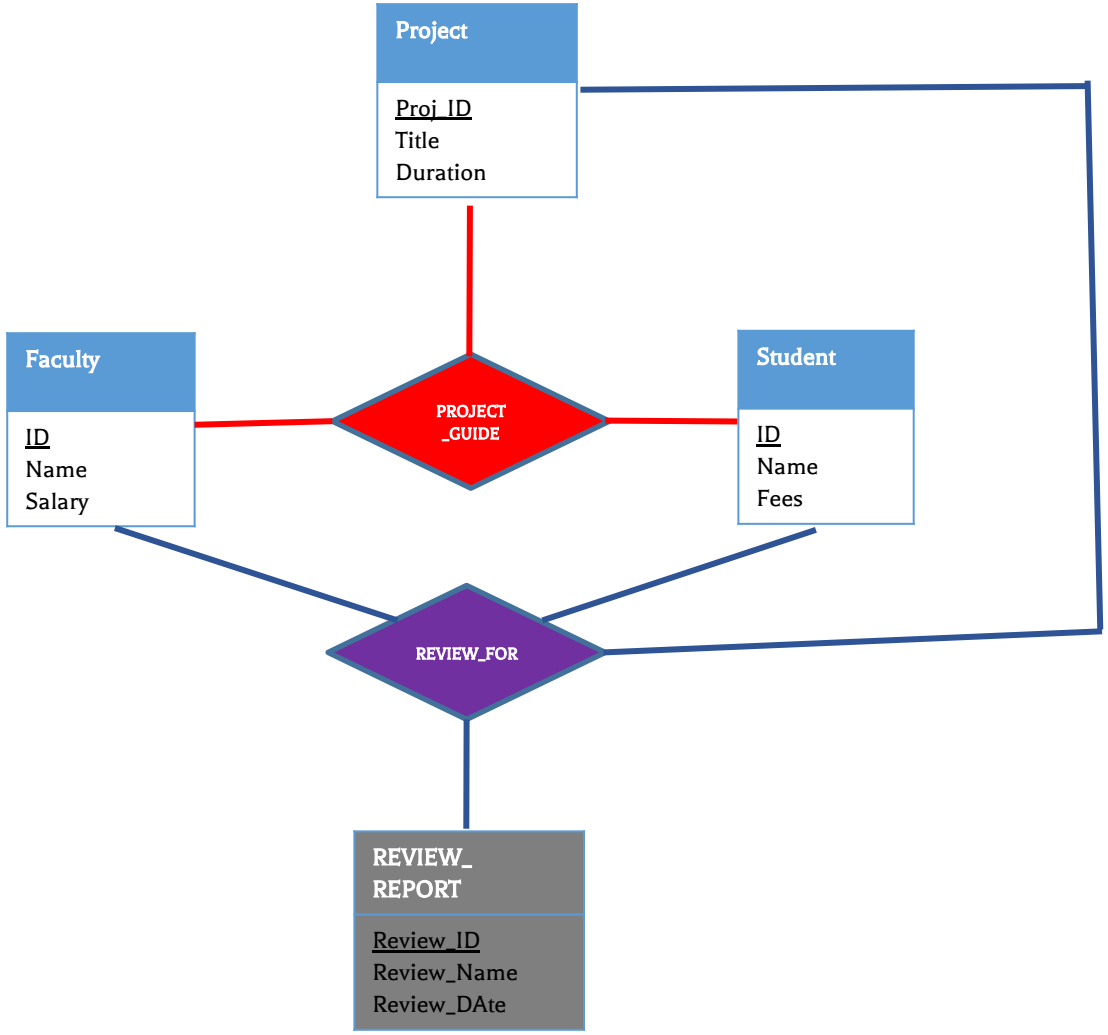
Aggregation

- Consider that, each faculty guiding a student on a project is required to file a monthly review report.
- We model the review report as an entity review_report, with a primary key review_id.
- One alternative for recording the (student, project, faculty) combination to which a review corresponds is to create a quaternary (4-way) relationship set review_for between faculty , student, project, and review_report evaluation.
- A quaternary relationship is required—a binary relationship between student and review report, for example, would not permit us to represent the (project, faculty) combination to which a review_ report corresponds.

Extended ER - Generalization, Specialization and Aggregation

Aggregation

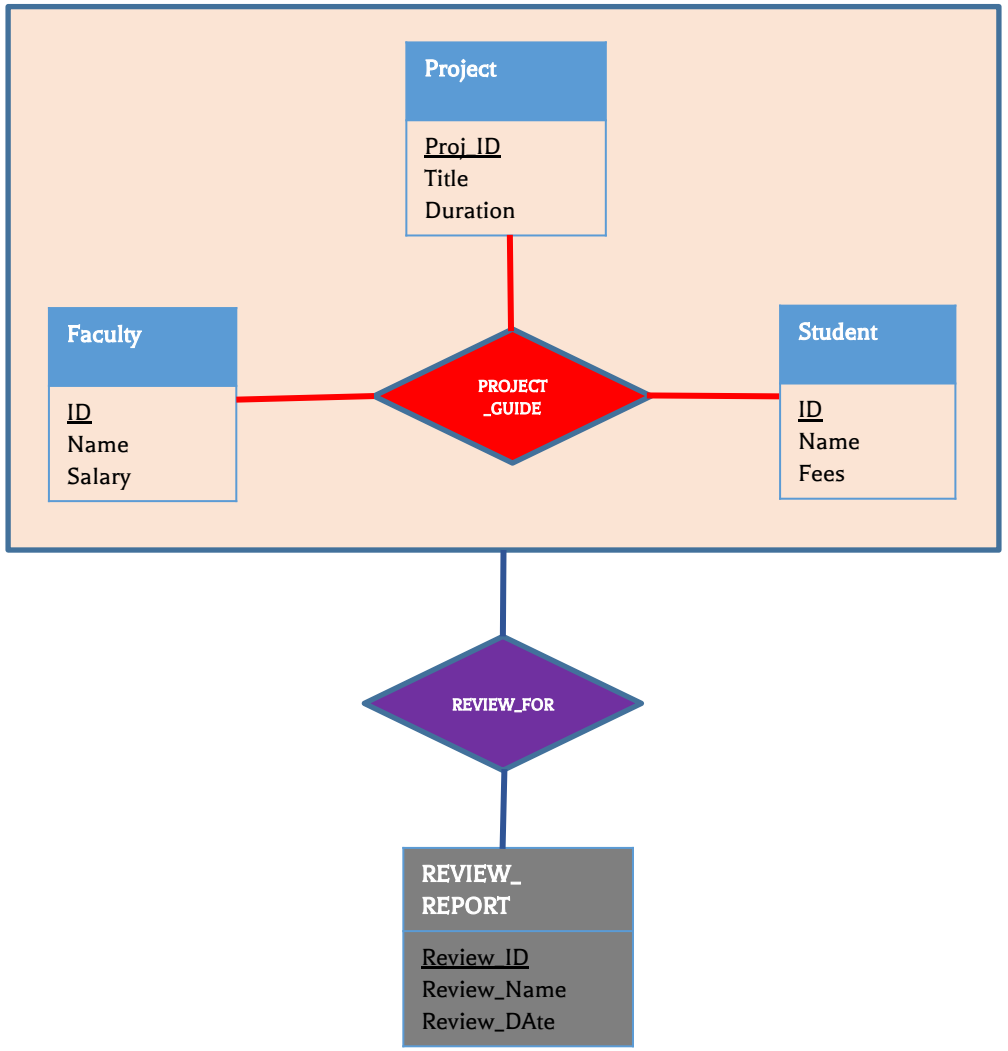
- ✔ Using the basic E-R modeling constructs, the following E-R diagram for the above constraints is obtained
- ✔ This diagram with redundant relationships



Extended ER - Generalization, Specialization and Aggregation

Aggregation

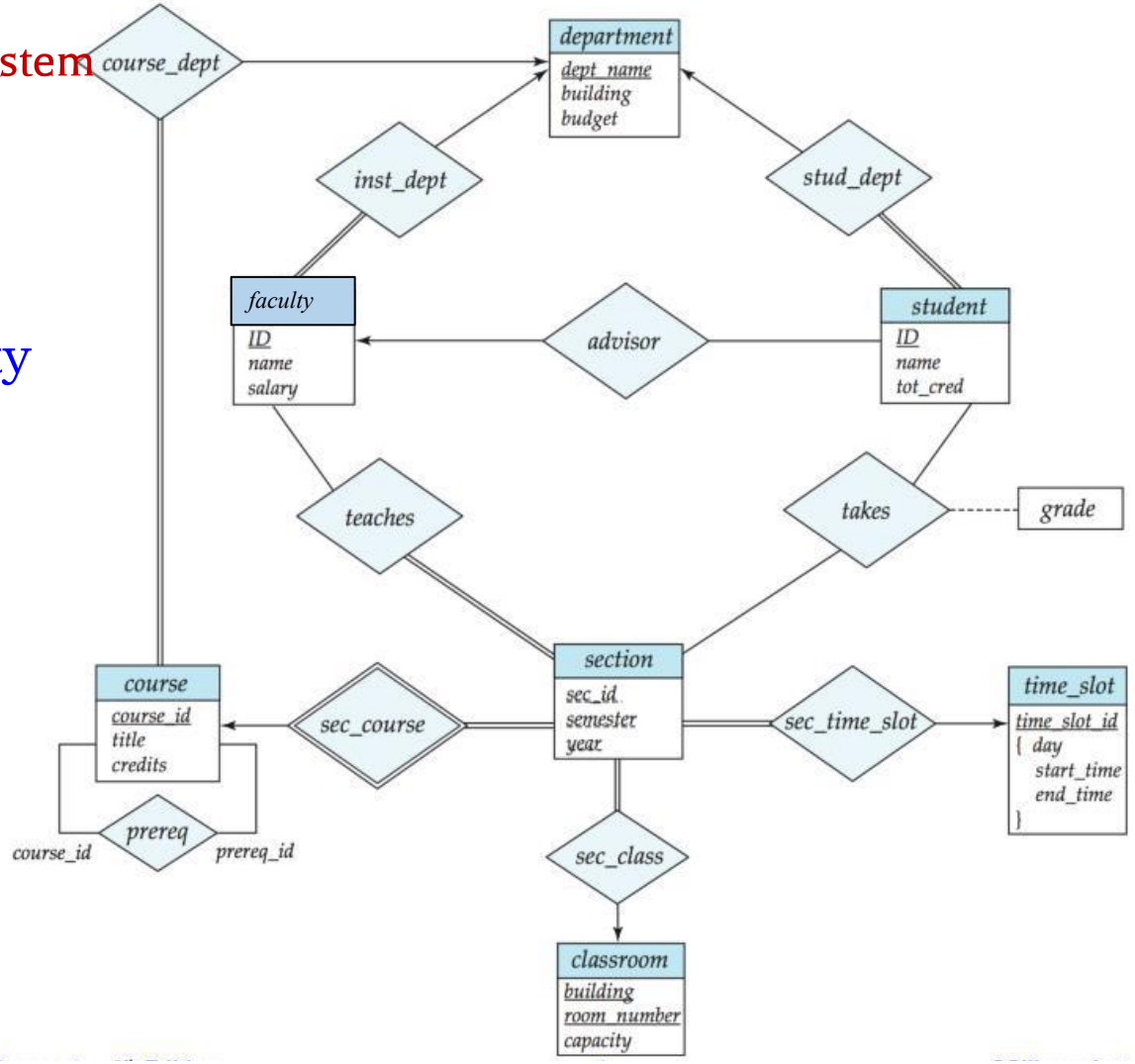
- The best way to model a situation such as the one just described is to use aggregation.
- Aggregation is an abstraction through which relationships are treated as higher-level entities.
- In the given example, the relationship set project_ guide (relating the entity sets faulty, student, and project) as a higher-level entity set called project_guide.
- Such an entity set is treated in the same manner as is any other entity set.
- We can then create a binary relationship review_for between project_guide and review report to represent which (student, project, faculty) combination an review_report is for.
- Figure shows a notation for aggregation commonly used to represent this situation.



Construct a ER Model for the application to be constructed to a Database

Sample ER Diagram for University Management System

- ✔Discuss briefly about E-R Diagram
- ✔Give the E-R Diagram Notations
- ✔List the schema participated in university
- ✔List the relationship sets
- ✔List the required constraints
- ✔Draw the E-R Diagram



ER Diagram Issues

- ✔ The notions of an entity set and a relationship set are not precise.
- ✔ It is possible to define a set of entities and the relationships among them in a number of different ways.
- ✔ The followings are the basic issues in ER Diagram
  - Use of Entity Sets versus Attributes
  - Use of Entity Sets versus Relationship Sets
  - Binary versus n-ary Relationship Sets
  - Placement of Relationship Attributes

ER Diagram Issues

Use of Entity Sets versus Attributes

- ✔ Consider the entity set faculty with the additional attribute phone_no , (Figure a)
- ✔ The considering phone as a separate entity , with attributes phone_no and location.
- ✔ The location may be office or home or mobile
- ✔ In this case , the attribute phone_no do not add to the faculty entity
- ✔ The following may consider
  - A phone entity set with attributes phone number and location.
  - A relationship set faculty_phone, denoting the association between faculty and the phones that they have. (Figure b)

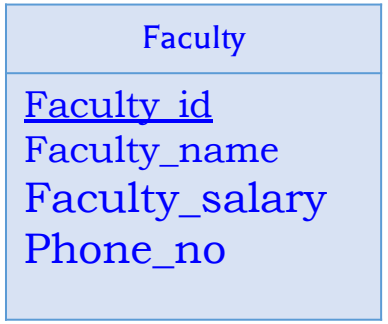


Figure a

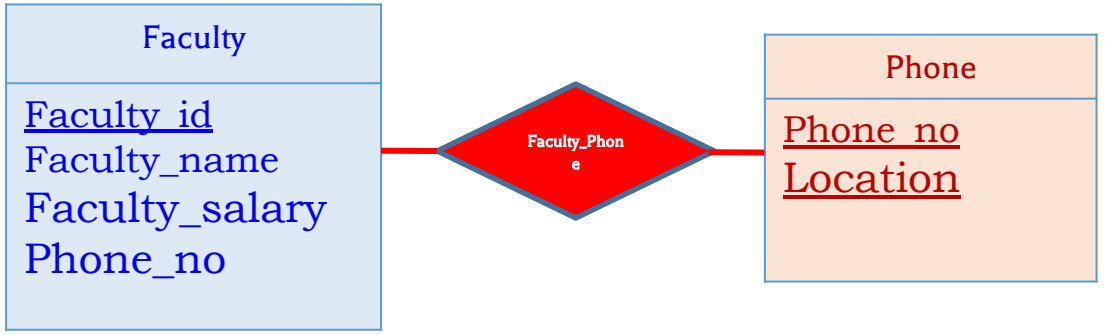


Figure b

ER Diagram Issues

Use of Entity Sets versus Attributes

- Treating a phone as an attribute phone number implies that faculty have precisely one phone number each.
- Treating a phone as an entity phone permits faculty to have several phone numbers (including zero) associated with them.
- However, we could instead easily define phone number as a multivalued attribute to allow multiple phones per faculty.
- The main difference then is that treating a phone as an entity better models a situation where one may want to keep extra information about a phone, such as its location, or its type like mobile, office, old phone, etc.,

ER Diagram Issues

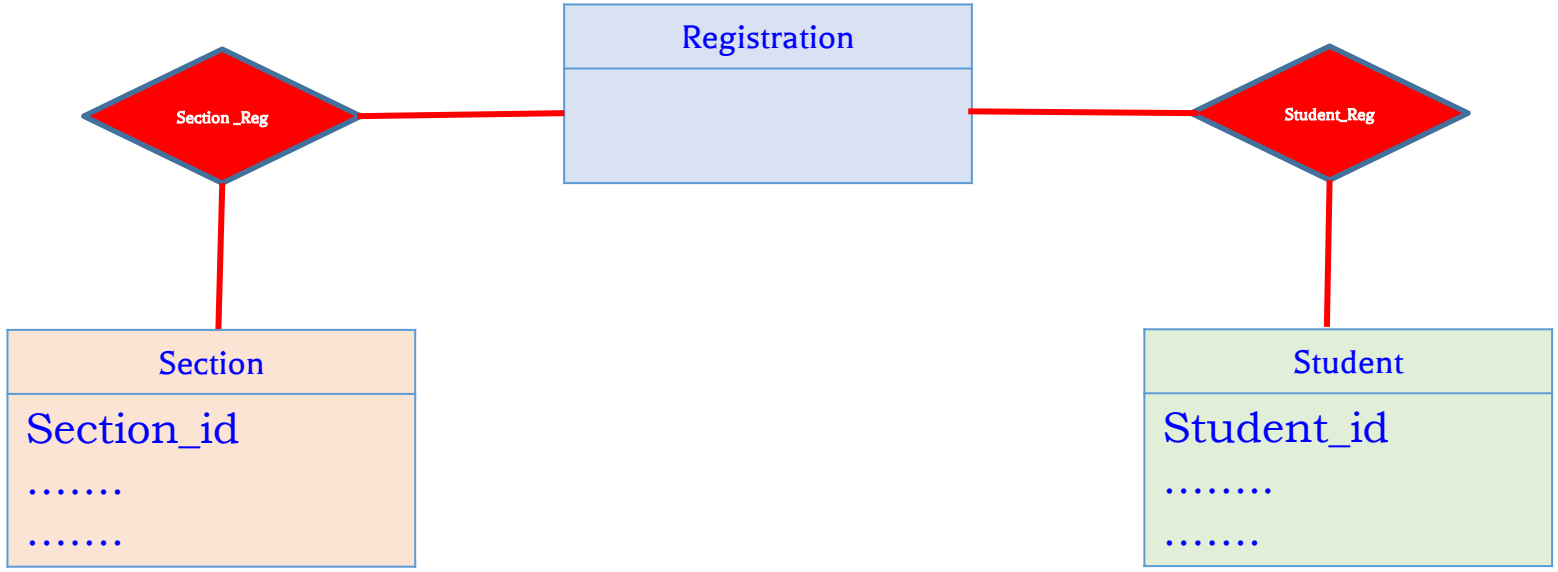
Use of Entity Sets versus Relationship Sets

- It is not always clear whether an object is best expressed by an entity set or a relationship set.
- In ER diagram for University Management system, we used the takes relationship set to model the situation where a student takes a (section of a) course.
- An alternative is to imagine that there is a course-registration record for each course that each student takes.
- Then need to have an entity set to represent the course-registration record.
- Let us call that entity set registration. Each registration entity is related to exactly one student and to exactly one section,
- Have two relationship sets, one to relate course registration records to students and one to relate course-registration records to sections.

ER Diagram Issues

Use of Entity Sets versus Relationship Sets

- In the given Figure , we show the entity sets section and student from ER diagram for University Management System with the takes relationship set replaced by one entity set and two relationship sets:
  - registration, the entity set representing course-registration records.
  - section reg, the relationship set relating registration and course.
  - student reg, the relationship set relating registration and student.



ER Diagram Issues

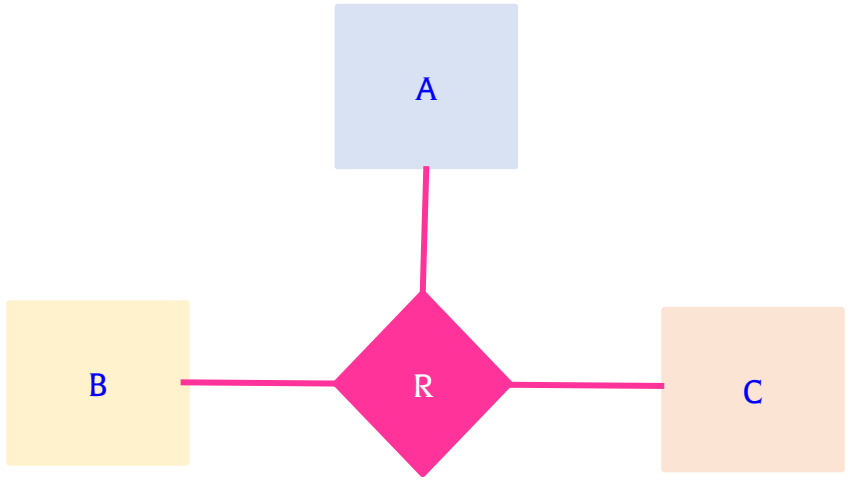
Use of Entity Sets versus Relationship Sets

- Relationships in databases are often binary.
- Some relationships that appear to be nonbinary could actually be better represented by several binary relationships.
- For instance, one could create a ternary relationship parent, relating a child to his/her mother and father.
- However, such a relationship could also be represented by two binary relationships, mother and father, relating a child to his/her mother and father separately.
- it is always possible to replace a nonbinary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets.

ER Diagram Issues

Use of Entity Sets versus Relationship Sets

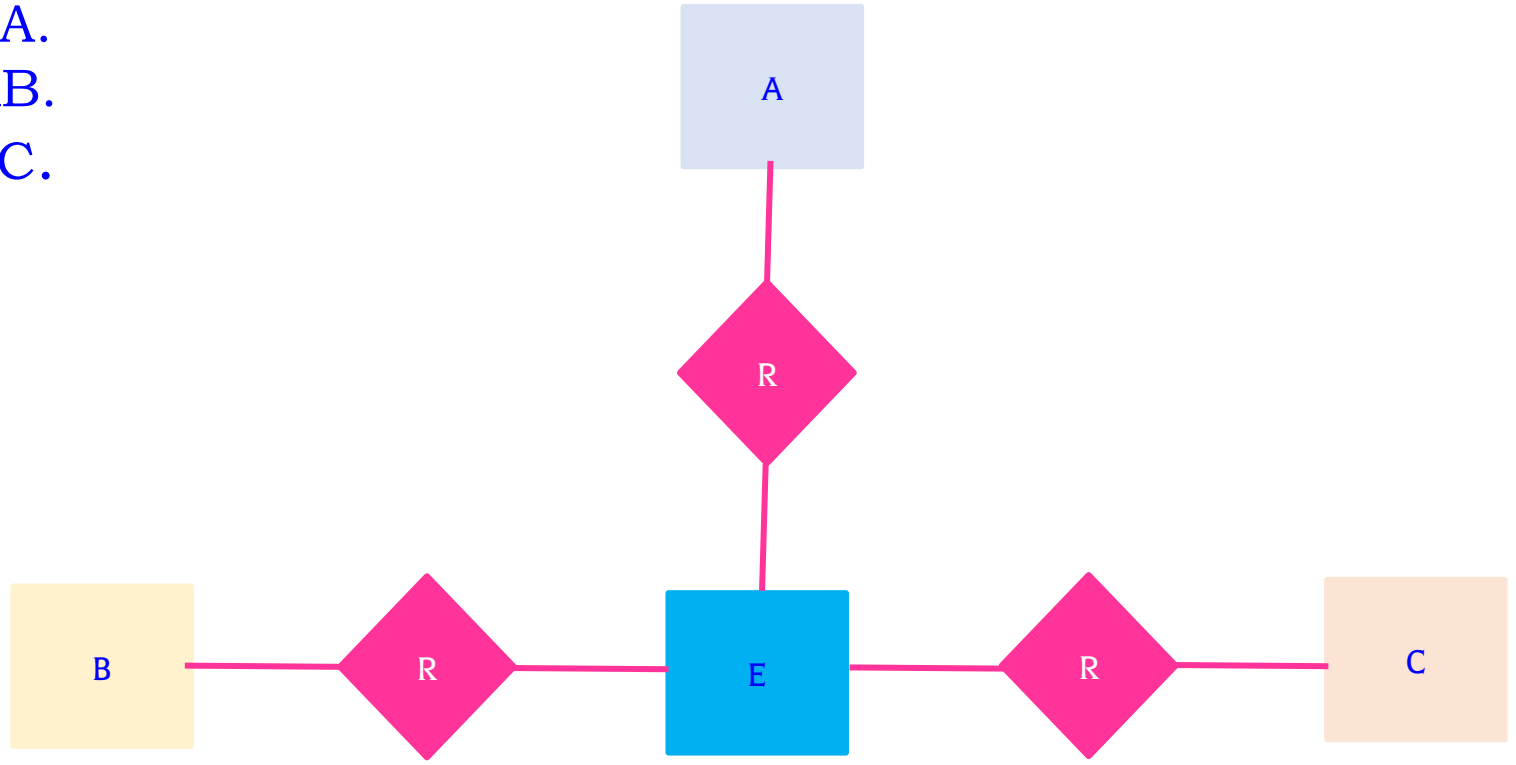
- Consider the abstract ternary (n = 3) relationship set R, relating entity sets A, B, and C. We replace the relationship set R by an entity set E, and create three relationship sets as shown in Figure below.
  - RA, relating E and A.
  - RB, relating E and B.
  - RC, relating E and C.



ER Diagram Issues

Use of Entity Sets versus Relationship Sets

- ✔ If the relationship set R had any attributes, these are assigned to entity set E.
- ✔ Further, a special identifying attribute is created for E For each relationship (ai, bi, ci) in the relationship set R, we create a new entity ei in the entity set E.
- ✔ Then, in each of the three new relationship sets, we insert a relationship as follows:
  - (ei, ai) in RA.
  - (ei, bi) in RB.
  - (ei, ci) in RC.



ER Diagram Issues

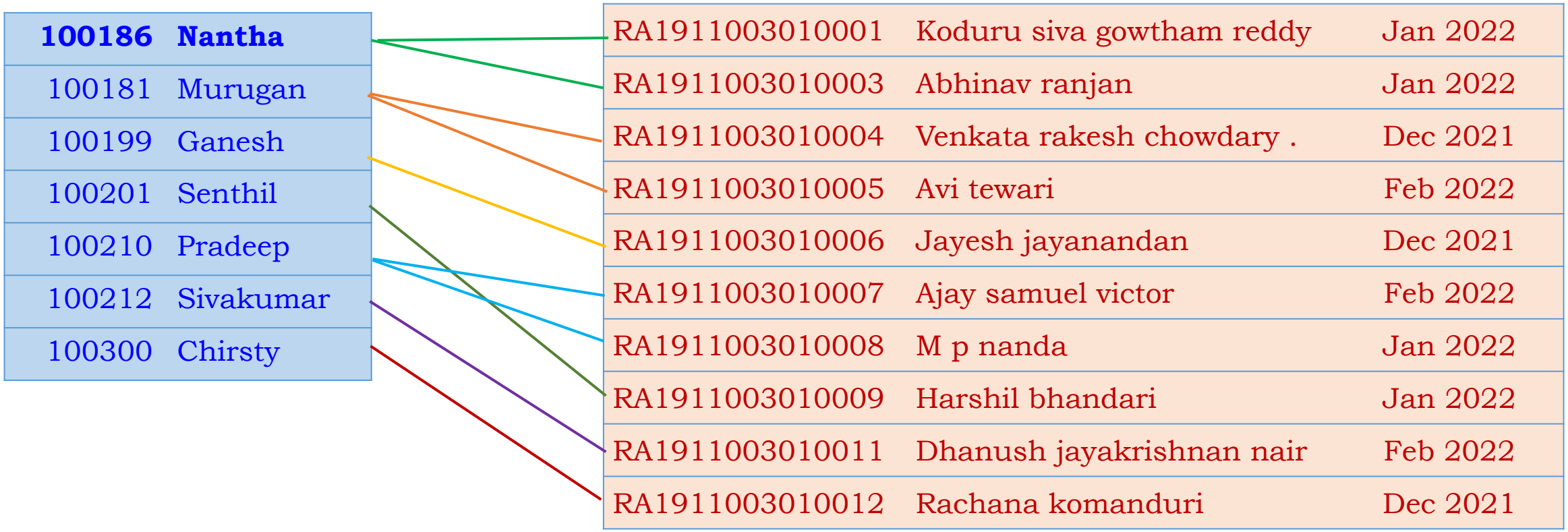
Placement of Relationship Attributes

- The cardinality ratio of a relationship can affect the placement of relationship attributes.
- Thus, attributes of one-to-one or one-to-many relationship sets can be associated with one of the participating entity sets, rather than with the relationship set.
- For instance, let us specify that counselor is a one-to-many relationship set such that one faculty may advise several students, but each student can be counseled only a single faculty.

ER Diagram Issues

Placement of Relationship Attributes

- In this case, the attribute date, which specifies when the faculty became the counselor of a student, could be associated with the student entity set, as Figure below depicts.



100186	**Nantha**
100181	Murugan
100199	Ganesh
100201	Senthil
100210	Pradeep
100212	Sivakumar
100300	Chirsty

RA1911003010001	Koduru siva gowtham reddy	Jan 2022
RA1911003010003	Abhinav ranjan	Jan 2022
RA1911003010004	Venkata rakesh chowdary .	Dec 2021
RA1911003010005	Avi tewari	Feb 2022
RA1911003010006	Jayesh jayanandan	Dec 2021
RA1911003010007	Ajay samuel victor	Feb 2022
RA1911003010008	M p nanda	Jan 2022
RA1911003010009	Harshil bhandari	Jan 2022
RA1911003010011	Dhanush jayakrishnan nair	Feb 2022
RA1911003010012	Rachana komanduri	Dec 2021

Relational Model

- The relational model is today the primary data model for commercial data processing applications.
- It attained its primary position because of its simplicity, which eases the job of the programmer / developer.
- It is simple and easy to understand compared to earlier data models such as the network model or the hierarchical model.
- The followings should be consider for Relational Model
  - Structure of Relational Databases
  - Database Schema
  - Keys
  - Schema Diagrams
  - Relational Query Languages
  - Relational Operations

Relational Model

Structure of Relational Databases

- A relational database consists of a collection of tables.
- Each table will have a unique name (unique identification)
- For example, consider the faculty table in the given figure, which stores information about faculty.
- This table contains four attributes (columns) named faculty_id, faculty_name, dept_name and salary

Faculty_id	Faculty_Name	Dept_Name	Salary
100186	Nantha	CSE	12345
100181	Murugan	DSBS	23456
100199	Ganesh	DSBS	12456
100201	Senthil	CSE	34213
100210	Pradeep	BT	23457
100212	Sivakumar	MECH	12567
100300	Chirsty	ECE	23425

Relational Model

Structure of Relational Databases

- Consider the following table Course, which stores the information about course details like course_code, title, dept_name, credits

Course_code	Title	Dept_Name	Credits
18CSC303J	Database Management Systems	Computing Technology	4
18CSE456T	Distributed Operating System	Computing Technology	3
18CSE390T	Computer Vision	Data Science and Business Systems	3
18CSC205J	Operating Systems	Data Science and Business Systems	4
18CSE344T	Cloud Architecture	Networking and Communications	3
18CSC305J	Artificial Intelligence	Computing Intelligence	4
18CSE459T	Service Oriented Architecture	Computing Intelligence	3

Structure of Relational Databases

- Consider the table, prereq, which stores the prerequisite courses for each course.
- The table has two attributes, course_code and prereq_code.

Course_code	Prereq_code
18CSC303J	18CSC161J
18CSE456T	18CSC205J
18CSE390T	18CSE353T
18CSC205J	18CSC161J
18CSE344T	18CSE378T
18CSC305J	18CSE388T
18CSE459T	18CSC302J

Relational Model

Structure of Relational Databases

- A row in a table represents a relationship among a set of values.
- A table is a collection of such relationships,
- In mathematical terminology, a tuple is simply a sequence (or list) of values.
- A relationship between n values is represented mathematically by an n-tuple of values, i.e., a tuple with n values, which corresponds to a row in a table.
- In relational model the term relation is used to refer to a table
- The term tuple is used to refer to a row.
- The term attribute refers to a column of a table.
- For each attribute of a relation, there is a set of permitted values, called the
- Domain of that attribute.
- The domains of all attributes of relation be atomic.
- The null value is a special value that signifies that the value is unknown or does not exist.

Database Schema

- The database schema, which is the logical design of the database.
- Database instance, which is a snapshot of the data in the database at a given instant in time.
- The concept of a relation corresponds to the programming-language notion of a variable.
- The concept of a relation schema corresponds to the programming-language notion of type definition.
- A relation schema consists of a list of attributes and their corresponding domains.

Relational Model

Database Schema

- Consider the Department relation

Dept_name	Location	Budget
Computing Technologies	Techpark	7000000
Networking and Communication	Techpark	4000000
Computing Intelligence	University Building	6000000
Data Science and Business Systems	University Building	3000000
Mechatronics	Hitech	3500000
Electrical Engineering	Main Building	2000000

- The schema for that relation is
  department (dept_name, location, budget)

Database Schema

- Consider the university database example (Slide Number : 96)
  - Each course in a university may be offered multiple times, across different semesters, or even within a semester.
  - A relation to describe each individual offering, or section, of the class.
  - The schema is:

    section (course_code, sec id, semester, year, location, room number, time slot id)
  - To describe the association between faculty and the class sections that they teach.

    teaches (faculty_id, course id, sec id, semester, year)

Database Schema

Section relation

Course_code	Sec_id	Semester	Year	Location	Room_no	Time_slot_id
18CSC303J	A1	EVEN	2022	Techpark	TP801	A
18CSE456T	A1	ODD	2021	Techpark	TP706	B
18CSE390T	B1	EVEN	2022	University Building	UB4001	C
18CSC205J	B1	EVEN	2022	University Building	UB5002	B
18CSE344T	B1	ODD	2021	Techpark	TP403	D
18CSC305J	A1	ODD	2021	University Building	UB1201	E
18CSE459T	A1	EVEN	2022	University Building	UB1210	G

Database Schema

Teaches Relation

Faculty_id	Course_Code	Sec_id	Semester	Year
100186	18CSC303J	A1	EVEN	2022
100181	18CSE456T	A1	ODD	2021
100199	18CSE390T	B1	EVEN	2022
100201	18CSC205J	B1	EVEN	2022
100210	18CSE344T	B1	ODD	2021
100212	18CSC305J	A1	ODD	2021
100300	18CSE459T	A1	EVEN	2022

Database Schema

- The other relations of University database is given below
  - student (reg_no, name, dept name, fees)
  - counselor (faculty_id, reg_no)
  - takes (reg_no, course_code, sec_id, semester, year, credits)
  - classroom (location, room number, capacity)
  - time_slot (time_slot_id, day_order, start_time, end_time)

Relational Model

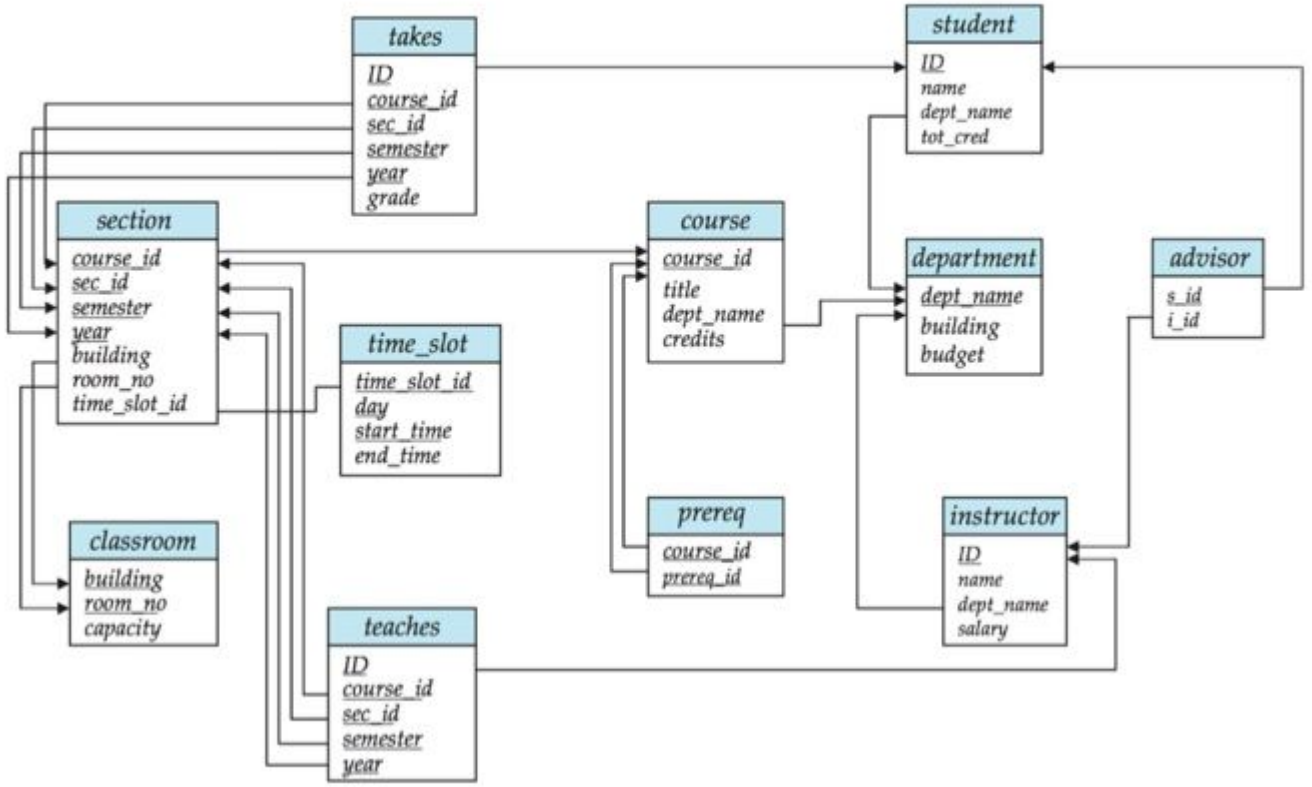
Keys

- One or more attributes used to identify an entity uniquely in an entity set if known as key attributes .
- Key attributes are called Super Key
- Minimal of Super key is Candidate Key
- Candidate key is also known as Primary key
- A primary key for a particular relation will be act as a referential key for another table is known as Foreign key

Relational Model

Schema Diagram

- ✔ The given figure is for University Database
- ✔ A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams.
- ✔ Each relation given as relation name and list of attributes
- ✔ Primary key attributes are underlined
- ✔ Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation.
- ✔ Referential integrity constraints other than foreign key constraints are not shown explicitly in schema diagrams.

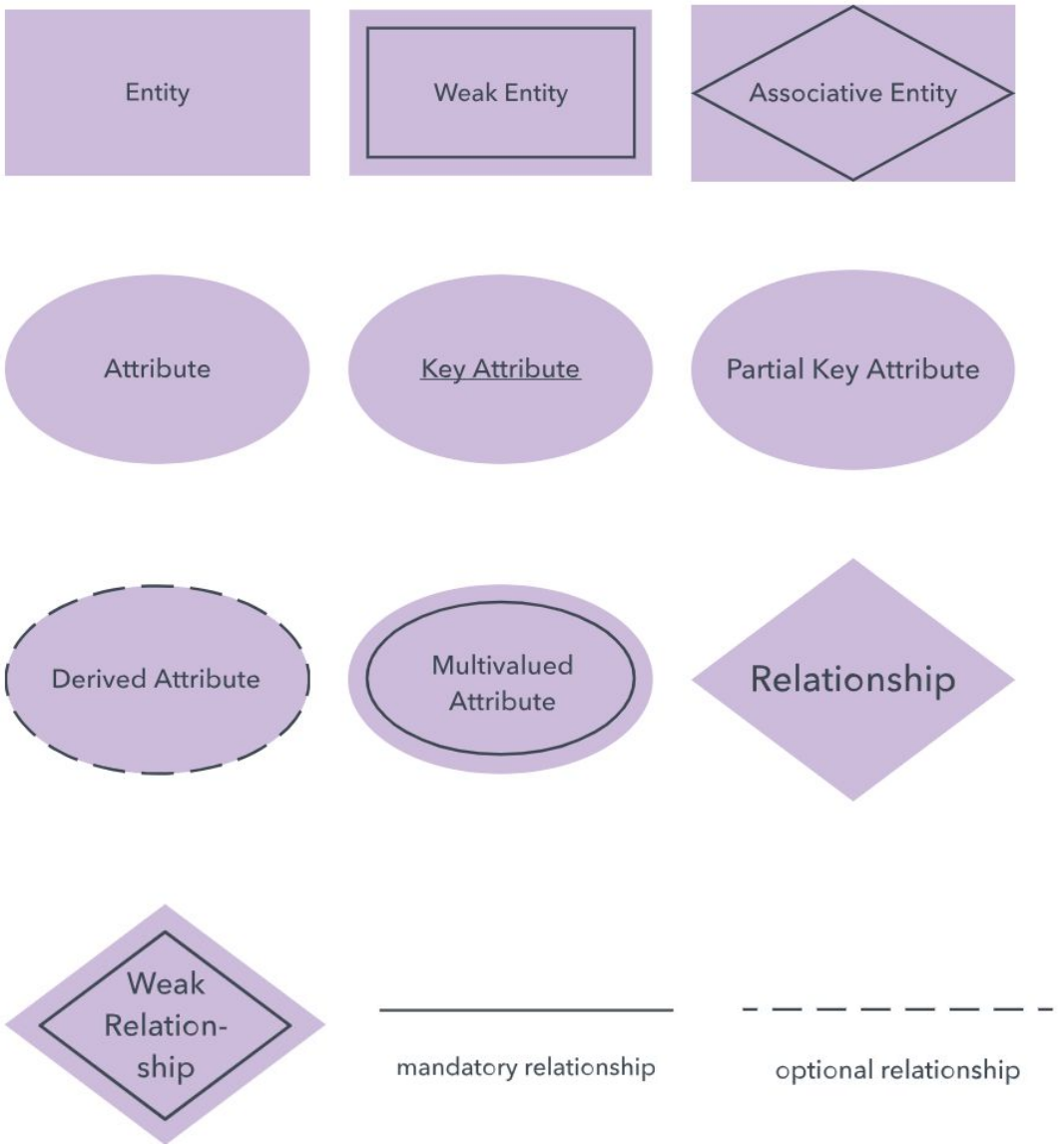


Relational Model

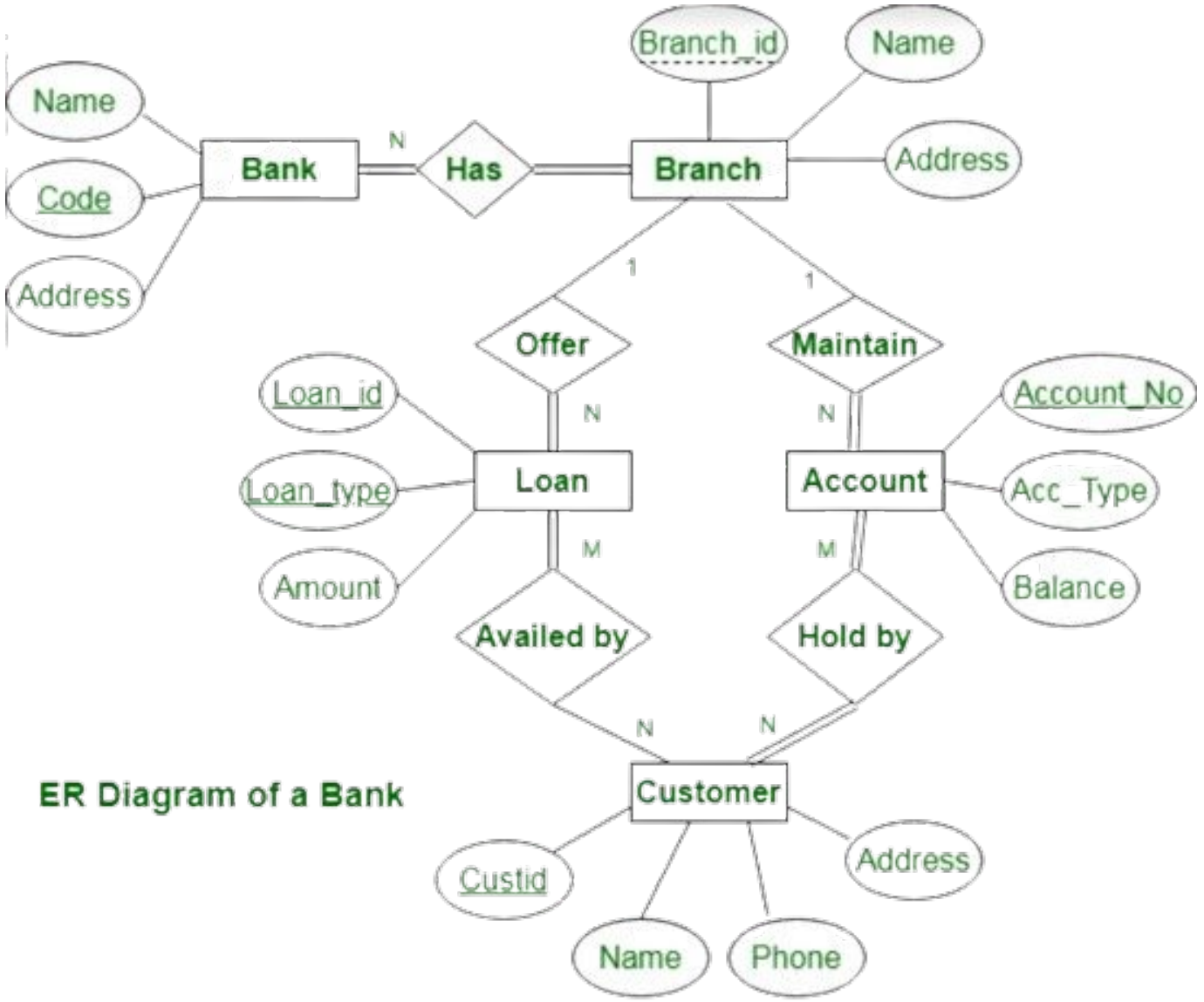
Relational Query Languages

- A query language is a language in which a user requests info
- These languages are usually on a level higher than that of a standard programming.
- Query languages can be categorized as either procedural or nonprocedural.
- In a procedural language, the user instructs the system to perform a sequence of operations on the database to compute the desired result.
- In a nonprocedural language, the user describes the desired information without giving a specific procedure for obtaining that information.
- There are a number of “pure” query languages.
- The relational algebra is procedural.
- The tuple relational calculus and domain relational calculus are nonprocedural.

ER Diagram Symbols



Entity Relationship diagram for Banking System



ER Diagram of a Bank

THANK YOU