

Database Management System

ER-Model & Relational Calculus

RAMNA SATTAR



Generalization

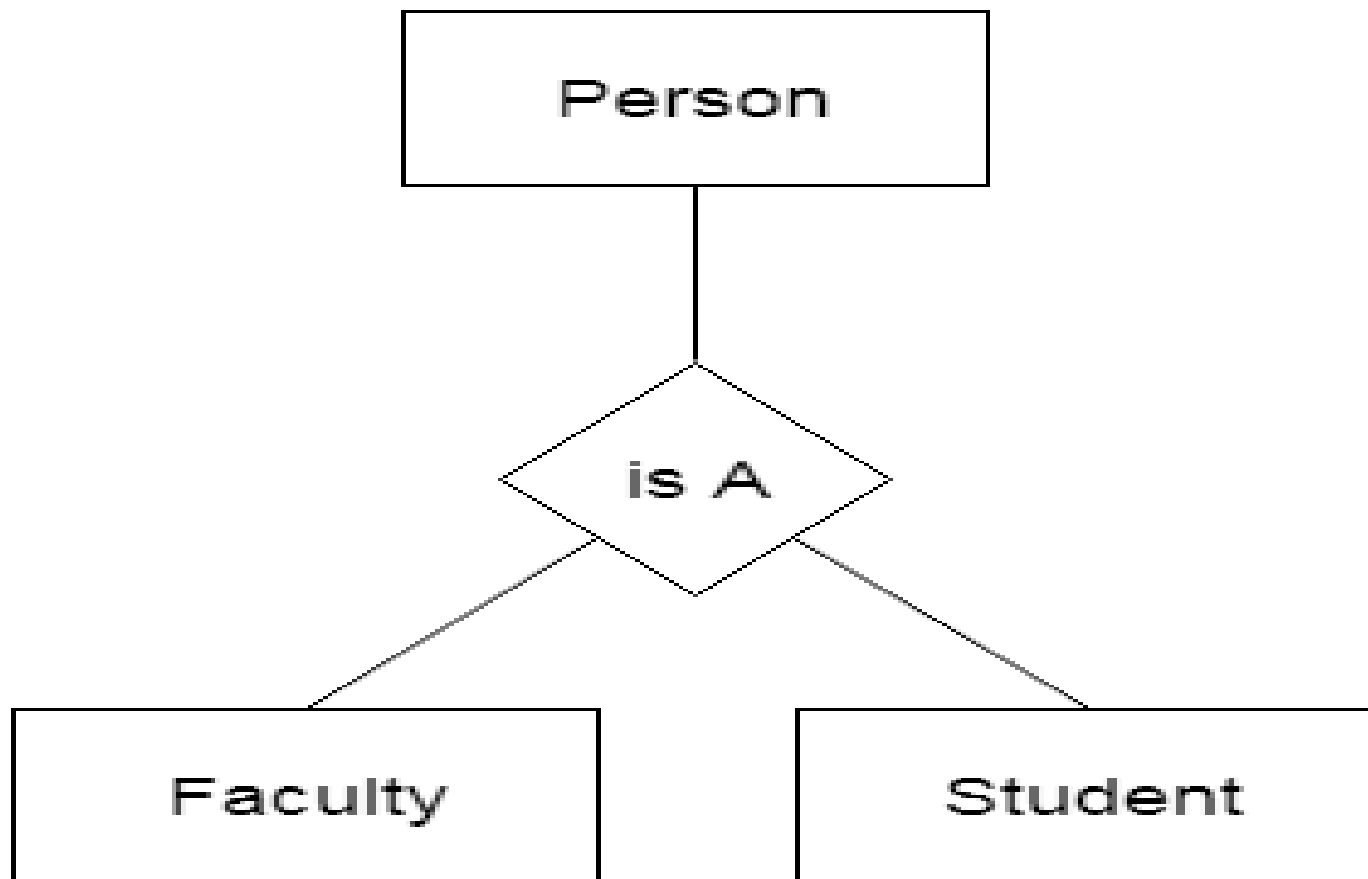
Generalization:

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



Generalization

For example, Faculty and Student entities can be generalized and create a higher level entity Person.



Specialization

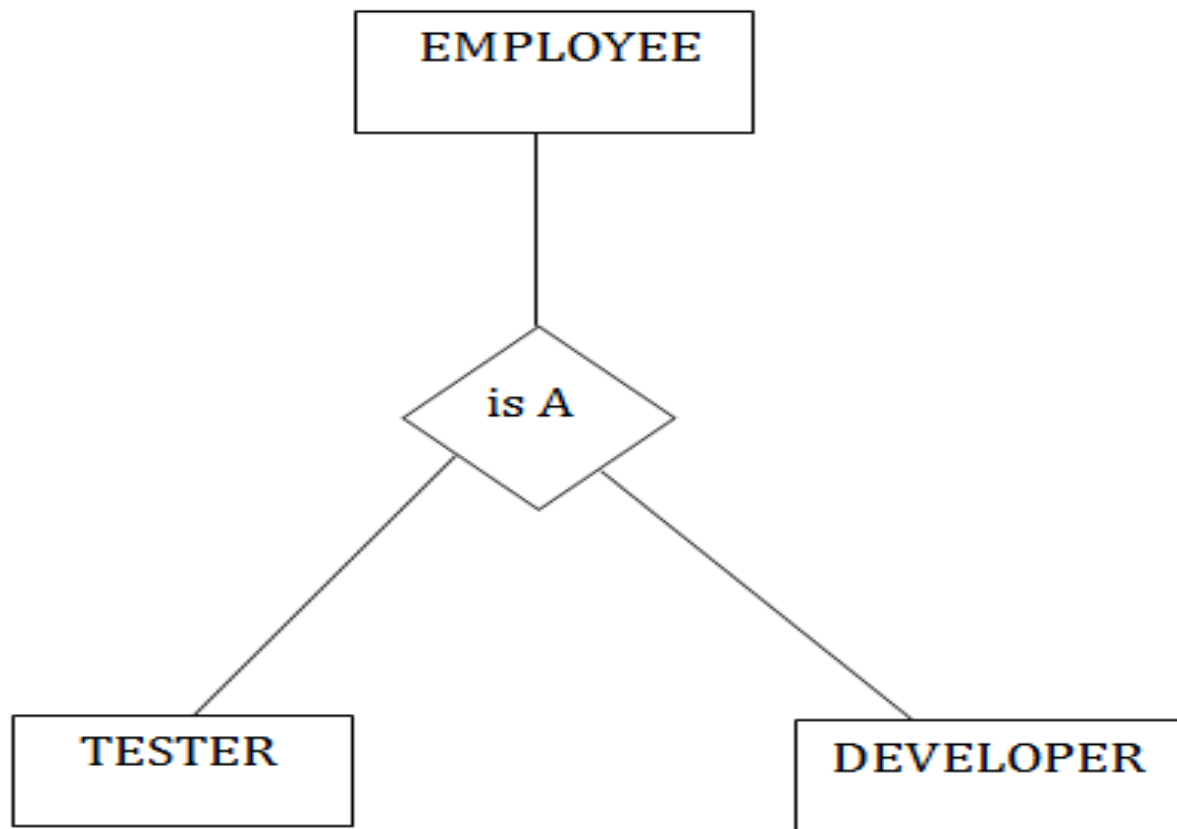
Specialization:

- ❑ Specialization is a top-down approach, and it is opposite to Generalization.
- ❑ In specialization, one higher level entity can be broken down into two or more lower level entities.
- ❑ Specialization is used to identify the subset of an entity set that shares some characteristics.
- ❑ Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.



Specialization

For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



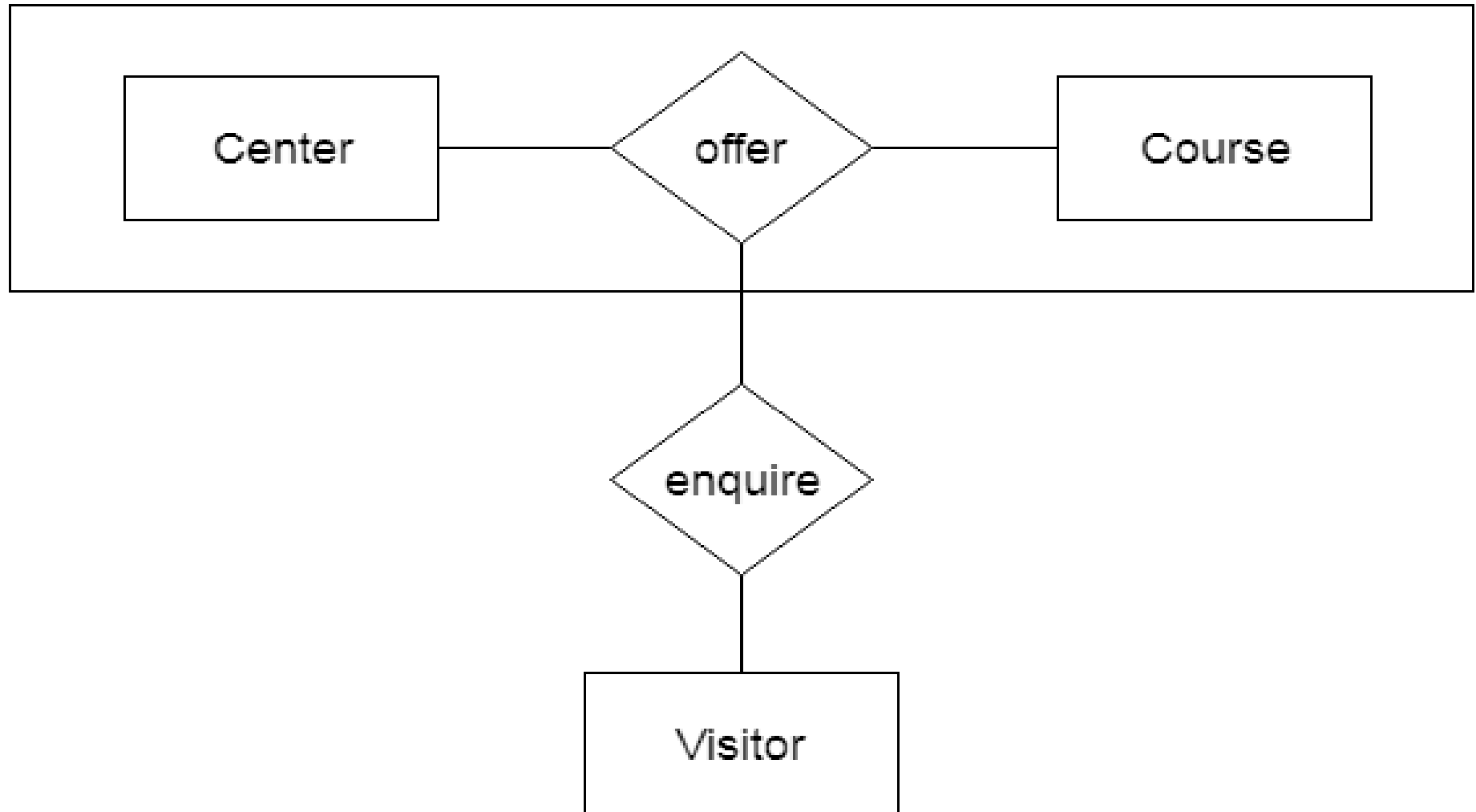
Aggregation

Aggregation:

- ❑ In aggregation, the relation between two entities is treated as a single entity.
- ❑ In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
- ❑ **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



Aggregation



Inheritance

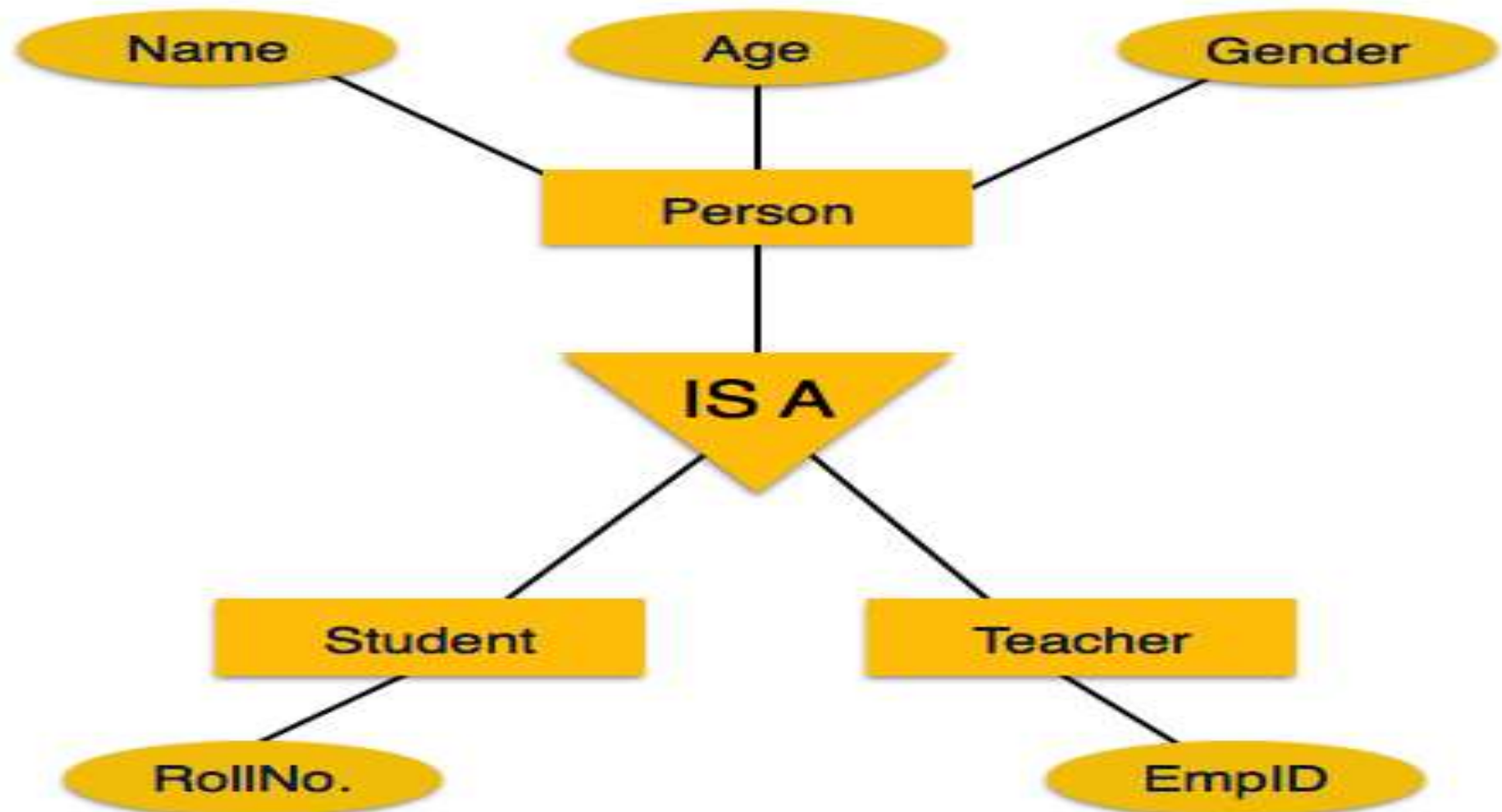
Inheritance:

- ❑ We use all the above features of ER-Model in order to create classes of objects in object-oriented programming.
- ❑ The details of entities are generally hidden from the user; this process known as **abstraction**.
- ❑ Inheritance is an important feature of Generalization and Specialization.
- ❑ It allows lower-level entities to inherit the attributes of higher-level entities.



Inheritance

For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.



Relational Calculus

Relational Calculus:

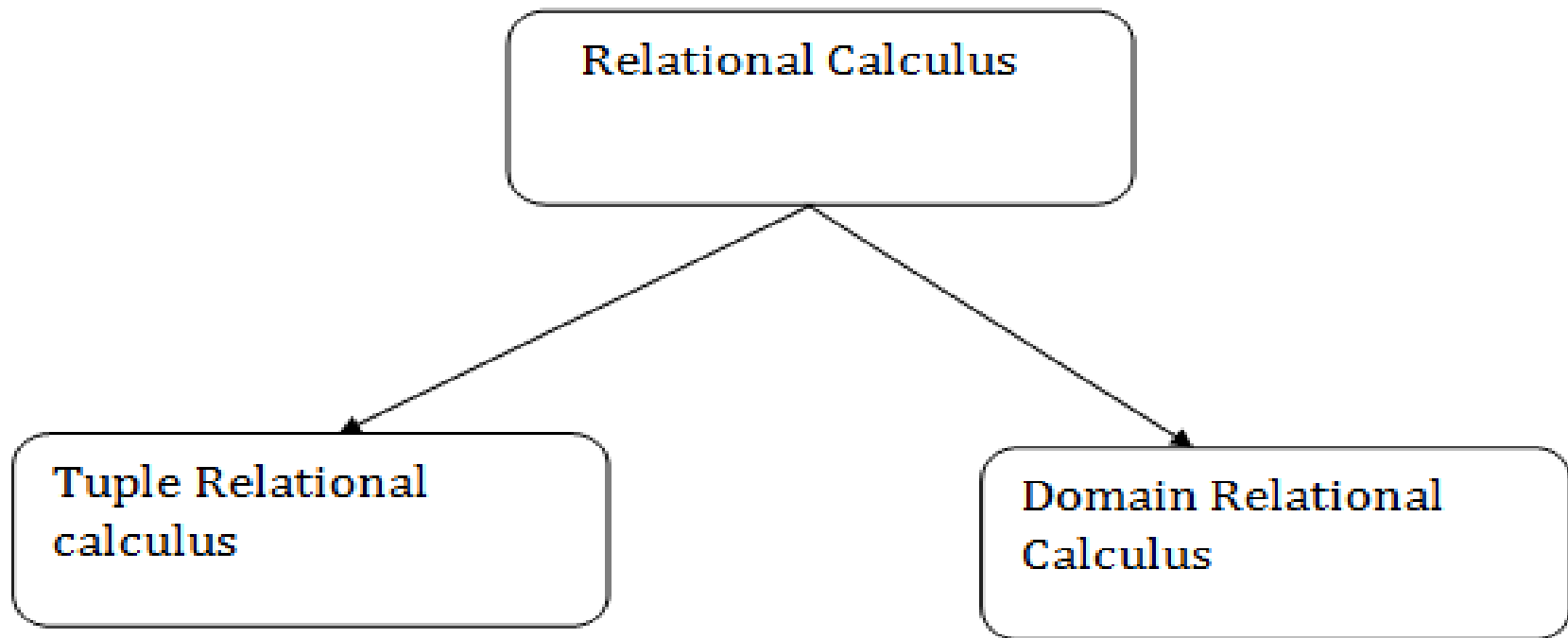
- ❑ Relational calculus is a non procedural query language.
- ❑ It uses mathematical predicate calculus instead of algebra.
- ❑ It provides the description about the query to get the result where as relational algebra gives the method to get the result.
- ❑ It informs the system what to do with the relation, but does not inform how to perform it.



Relational Calculus

Relational Calculus exists in two forms:

- ❑ Tuple Relational Calculus (TRC)
- ❑ Domain Relational Calculus (DRC)



Relational Calculus

Tuple Relational Calculus:

- ❑ A tuple relational calculus is a non procedural query language which specifies to select the tuples in a relation.
- ❑ It can select the tuples with range of values or tuples for certain attribute values etc.
- ❑ The resulting relation can have one or more tuples.

It is denoted as below:

$\{t \mid P(t)\}$ or $\{t \mid \text{condition}(t)\}$ -- this is also known as expression of relational calculus

Where t is the resulting tuples, $P(t)$ is the condition used to fetch t .
The variable which is used in the condition is called **tuple variable**.



Relational Calculus

Tuple Relational Calculus

Example:

□ $\{t \mid \text{EMPLOYEE}(t) \text{ and } t.\text{SALARY} > 10000\}$ - implies that it selects the tuples from EMPLOYEE relation such that resulting employee tuples will have salary greater than 10000. It is example of selecting a range of values.

□ $\{t \mid \text{EMPLOYEE}(t) \text{ AND } t.\text{DEPT_ID} = 10\}$ – this select all the tuples of employee name who work for Department 10.



Relational Calculus

Domain Relational Calculus:

- ❑ In contrast to tuple relational calculus, domain relational calculus uses list of attribute to be selected from the relation based on the condition.
- ❑ It is same as TRC, but differs by selecting the attributes rather than selecting whole tuples.

It is denoted as below:

$$\{ \langle a_1, a_2, a_3, \dots a_n \rangle \mid P(a_1, a_2, a_3, \dots a_n) \}$$

Where $a_1, a_2, a_3, \dots a_n$ are attributes of the relation and P is the condition.



Relational Calculus

Domain Relational Calculus

Example:

❑ select EMP_ID and EMP_NAME of employees who work for department 10

$\{ \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \mid \langle \text{EMP_ID}, \text{EMP_NAME} \rangle ? \text{EMPLOYEE} \wedge \text{DEPT_ID} = 10 \}$

❑ Get name of the department name that Alex works for.

$\{ \text{DEPT_NAME} \mid \langle \text{DEPT_NAME} \rangle ? \text{DEPT} \wedge ? \text{DEPT_ID} (\langle \text{DEPT_ID} \rangle ? \text{EMPLOYEE} \wedge \text{EMP_NAME} = \text{Alex}) \}$

Here green color expression is evaluated to get the department Id of Alex and then it is used to get the department name form DEPT relation



Relational Calculus Summary

Relational Query Language					
Relational Algebra			Relational Calculus		
	Operator	Symbol	Operator	Symbol	Example
Fundamental Operators	SELECT	σ	Tuple Relational Calculus (TRC)	$\{t \mid \text{condition}(t)\}$	$\{t \mid \text{EMPLOYEE}(t) \text{ AND } t.DEP_ID = 10\}$
					$\{ \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \mid \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \in \text{EMPLOYEE} \wedge \text{DEPT_ID} = 10 \}$
	PROJECT	Π	Domain Relational Calculus (DRC)	$\{ \langle a_1, a_2, a_3, \dots, a_n \rangle \mid \theta(a_1, a_2, a_3, \dots, a_n) \}$	$\{ \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \mid \langle \text{EMP_ID}, \text{EMP_NAME} \rangle \in \text{EMPLOYEE} \wedge \text{DEPT_ID} = 10 \}$
	RENAME	ρ			
	UNION	\cup			
	SET DIFFERENCE	$-$			
	CARTESIAN PRODUCT	\times			
Non-Fundamental Operators	SET INTERSECTION	\cap			
	ASSIGNMENT	\leftarrow			
	Natural JOINS	\bowtie			
	Left Outer Join				
	Right Outer Join				
	Full Outer Join				
	Division	\div			

Database Management System

Normalization

RAMNA SATTAR



Normalization

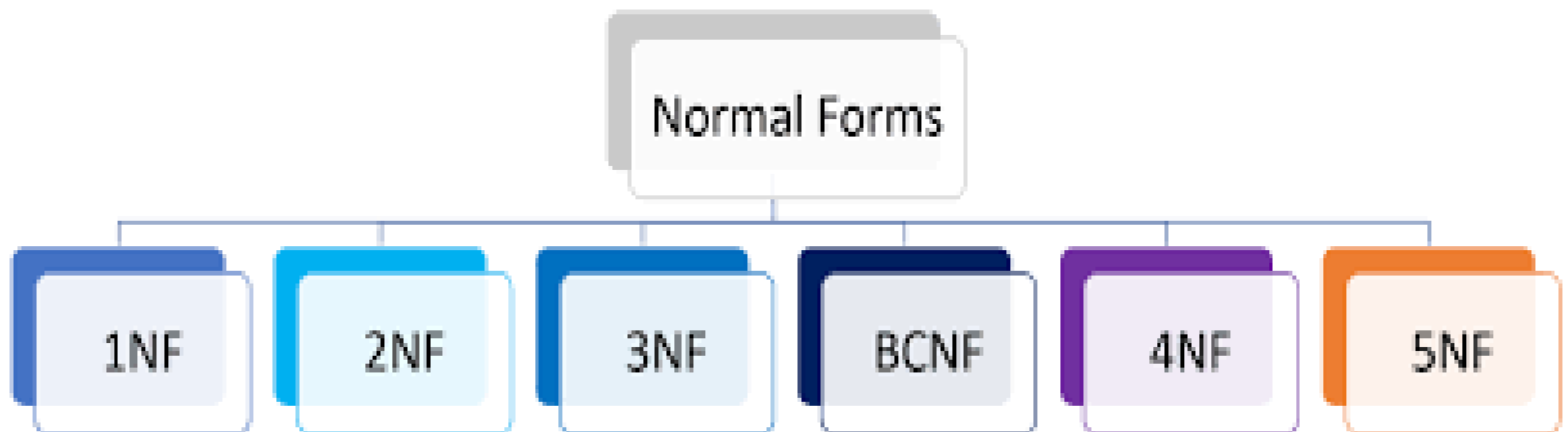
What is Normalization?

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.

It divides larger tables to smaller tables and links them using relationships.

Normalization was developed by IBM researcher E.F. Codd In the 1970s.

Normalization increases the clarity in organizing data in Database.



Normalization

Normalization Anomalies:

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.

Update anomalies: If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

Deletion anomalies: We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.

Insert anomalies: We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.



Normalization

1NF (First Normal Form) Rules:

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be of the same domain
- All the columns in a table should have unique names.
- And the order in which data is stored, does not matter.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.



Normalization

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Normalization

2NF (Second Normal Form) Rules:

- In the 2NF, relational must be in 1NF.
- And, it should not have Partial Dependency.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key.

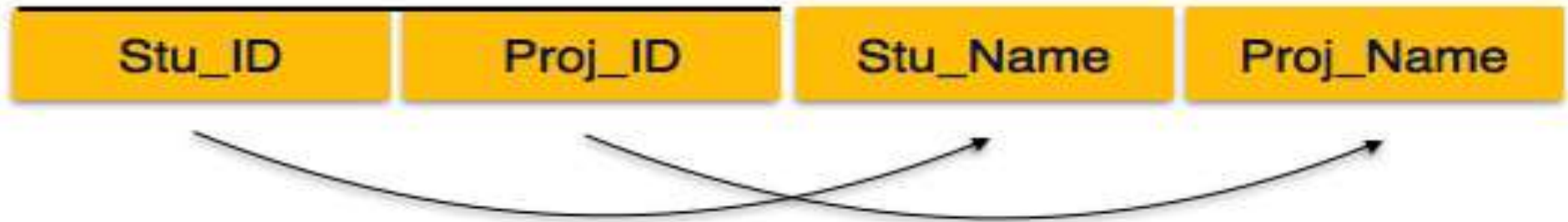
Partial Dependency:

Partial Dependency occurs when a non-key attribute is functionally dependent on part of the primary key.



Normalization

Student_Project



Student



2NF

Project



Normalization

EmpId	EmployeeName	Gender	Salary	DeptName	DeptHead	DeptLocation
1	Sam	Male	4500	IT	John	London
2	Pam	Female	2300	HR	Mike	Sydney
3	Simon	Male	1345	IT	John	London
4	Mary	Female	2567	HR	Mike	Sydney
5	Todd	Male	6890	IT	John	London

Problems of Data Redundancy

1. Disk Space Wastage
2. Data Inconsistency
3. DML queries can become slow

Table Design in Second Normal Form

DeptId	DeptName	DeptHead	DeptLocation
1	IT	John	London
2	HR	Mike	Sydney

EmpId	EmployeeName	Gender	Salary	DeptId
1	Sam	Male	4500	1
2	Pam	Female	2300	2
3	Simon	Male	1345	1
4	Mary	Female	2567	2
5	Todd	Male	6890	1

Normalization

What are transitive functional dependencies?

A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.

Consider the table 1. Changing the non-key column Full Name may change Salutation.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

Change in Name

*May Change
Salutation*

Normalization

3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
- Rule 2- Has no transitive functional dependencies

Student_Detail

Stu_ID	Stu_Name	City	Zip
--------	----------	------	-----



Student_Detail

Stu_ID	Stu_Name	Zip
--------	----------	-----

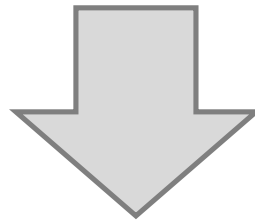
ZipCodes

Zip	City
-----	------

3NF

Normalization

2NF				
STUDENT_ID	STUDENT_NAME	STREET	CITY	ZIP
100	Joseph	Alaiedon Village Drive	Alaiedon Township	48035
101	Allen	Fraser Village Drive	Fraser Township	48036
102	Chris	Lakeside Village Drive	Clinton Township	48038
103	Patty	Troy Village Drive	Troy	48039



3NF					
STUDENT			ZIPCODE		
STUDENT_ID	STUDENT_NAME	ZIP	ZIP	STREET	CITY
100	Joseph	48035	48035	Alaiedon Village Drive	Alaiedon Township
101	Allen	48036	48036	Fraser Village Drive	Fraser Township
102	Chris	48038	48038	Lakeside Village Drive	Clinton Township
103	Patty	48039	48039	Troy Village Drive	Troy

Normalization

Boyce-Codd Normal Form (BCNF):

Every determinant must be a candidate key.

Sometimes BCNF is also referred as **3.5 Normal Form**.

4NF (Fourth Normal Form) Rules:

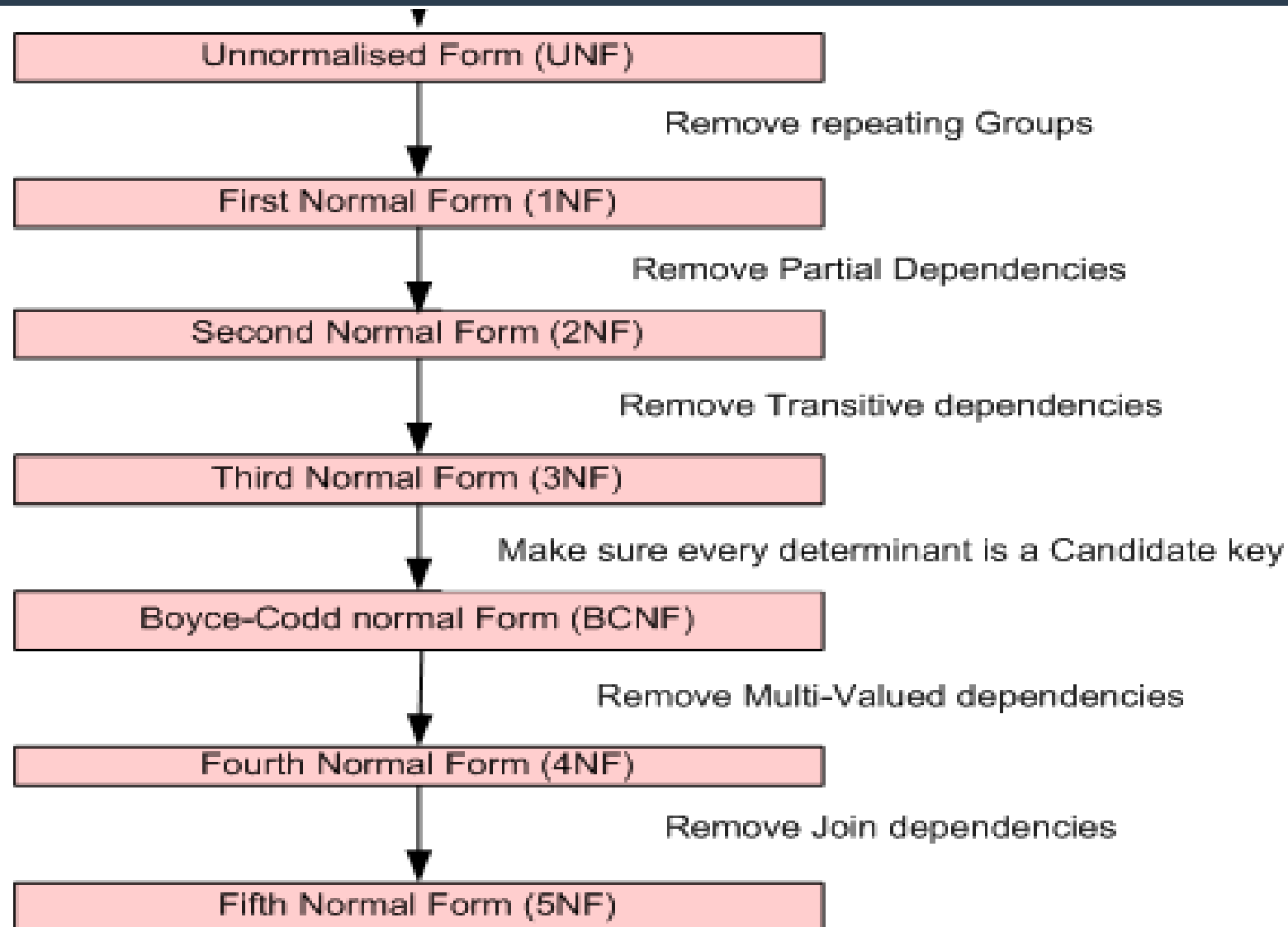
Tables cannot have multi-valued dependencies on a Primary Key.

5NF (Fifth Normal Form) Rules:

All join Dependencies are removed.



Normalization



THANK YOU

ANY QUERY???

