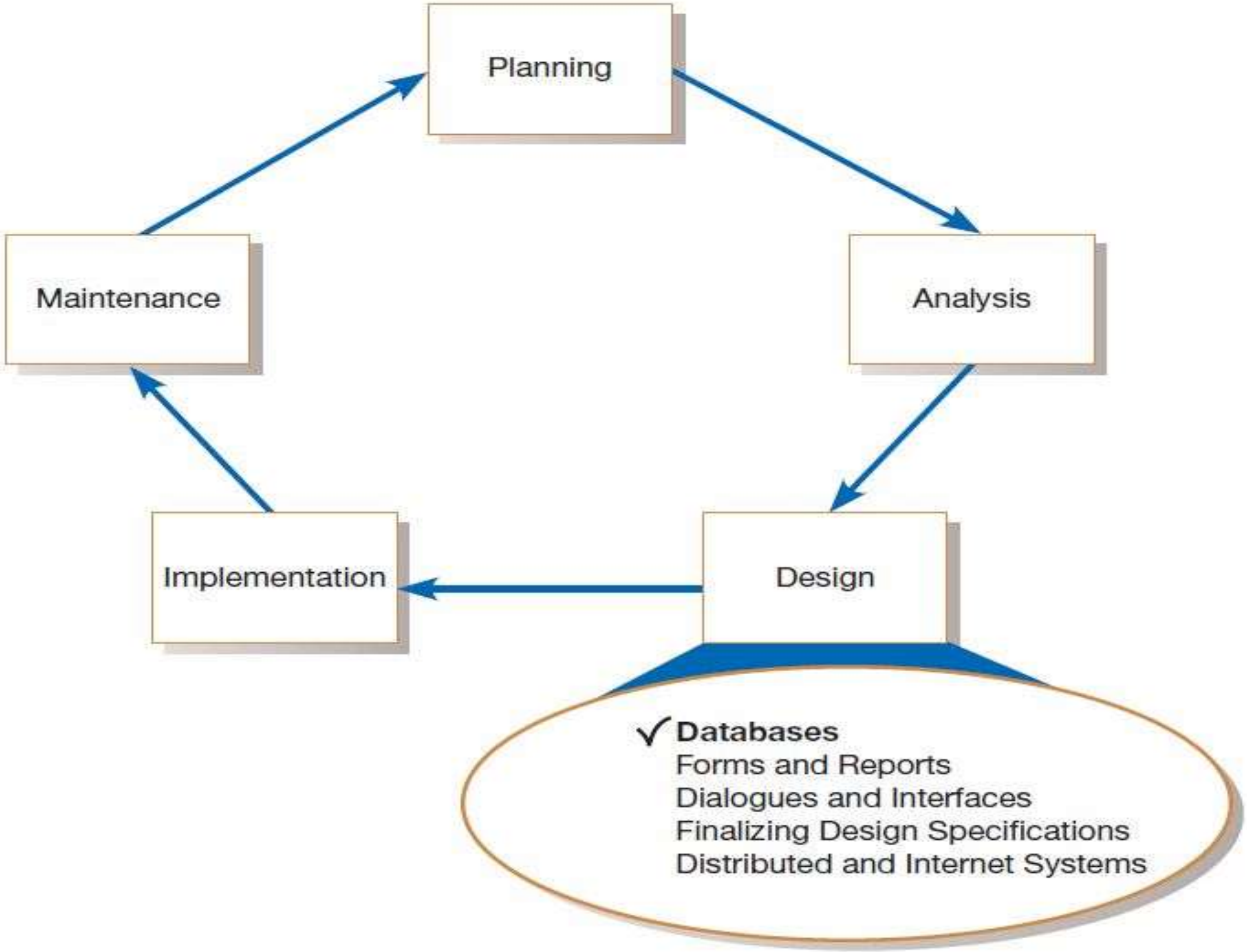
Unit 4.1 Designing Databases

Introduction; Database Design (Process, Deliverables and Outcomes, Relational Database Model, Well-structured Relations); Normalization (Normalization up to 3NF); Transforming E-R Diagrams Into Relations; Merging Relations; Physical File and Database Design; Designing Fields; Designing Physical Tables

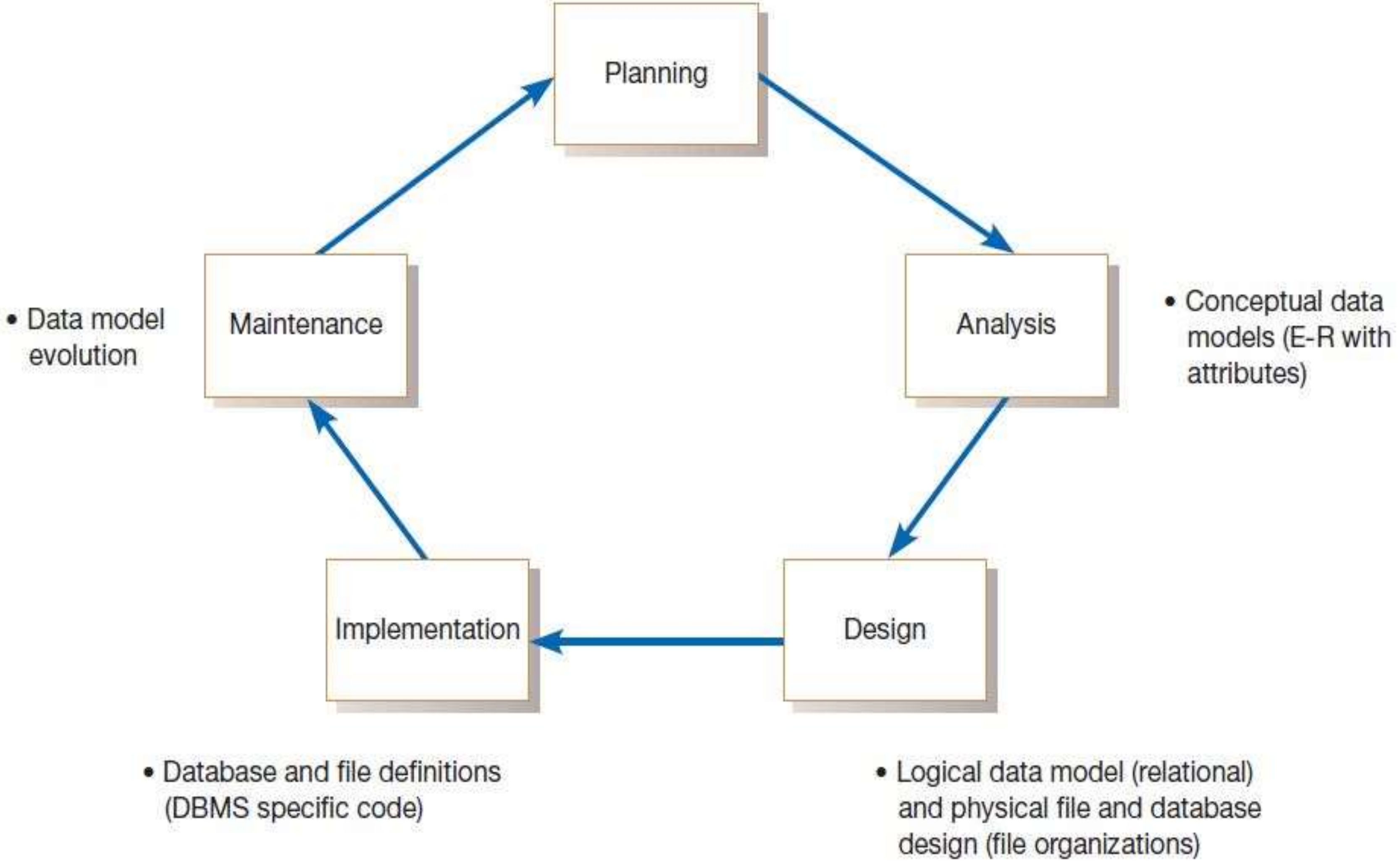
Database design has five purposes:

- Structure the data in stable structures, called normalized tables, that are not likely to change over time and that have minimal redundancy.
- Develop a logical database design that reflects the actual data requirements that exist in the forms (hard copy and computer displays) and reports of an information system. This is why database design is often done in parallel with the design of the human interface of an information system.
- Develop a logical database design from which we can do physical database design. Because most information systems today use relational database management systems, logical database design usually uses a relational database model, which represents data in simple tables with common columns to link related tables.

- Translate a relational database model into a technical file and database design that balances several performance factors.
- Choose data storage technologies (such as Read/ Write DVD or optical disc) that will efficiently, accurately and securely process database activities



Systems development life cycle with design phase highlighted

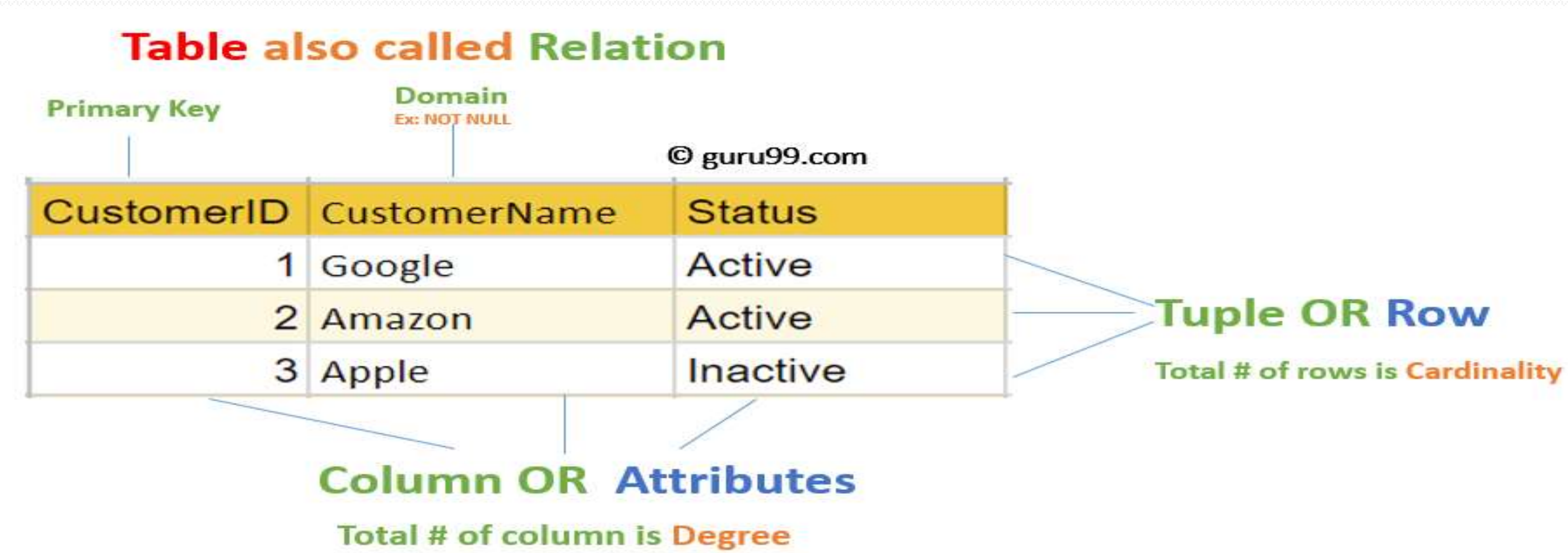


Relationship between data modeling and the SDLC

- There are four key steps in logical database modeling and design:
  - Develop a logical data model for each known user interface (form and report) for the application using normalization principles.
  - Combine normalized data requirements from all user interfaces into one consolidated logical database model; this step is called view integration.
  - Translate the conceptual E-R data model developed without explicit consideration of specific user interfaces, into normalized data requirements.
  - Compare the consolidated logical database design with the translated E-R model and produce, through view integration, one final logical database model for the application.

RELATIONAL MODEL (RM)

- Represents the database as a collection of relations.
- A relation is nothing but a table of values.
- Every row in the table represents a collection of related data values.
- These rows in the table denote a real-world entity or relationship.
- The table name and column names are helpful to interpret the meaning of values in each row.
- The data are represented as a set of relations.
- In the relational model, data are stored as tables.



Attribute: Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.

Tables – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

Tuple – It is nothing but a single row of a table, which contains a single record.

Relation Schema: A relation schema represents the name of the relation with its attributes.

Degree: The total number of attributes which in the relation is called the degree of the relation.

Cardinality: Total number of rows present in the Table.

Column: The column represents the set of values for a specific attribute.

Relation instance – Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

Relation key - Every row has one, two or multiple attributes, which is called relation key.

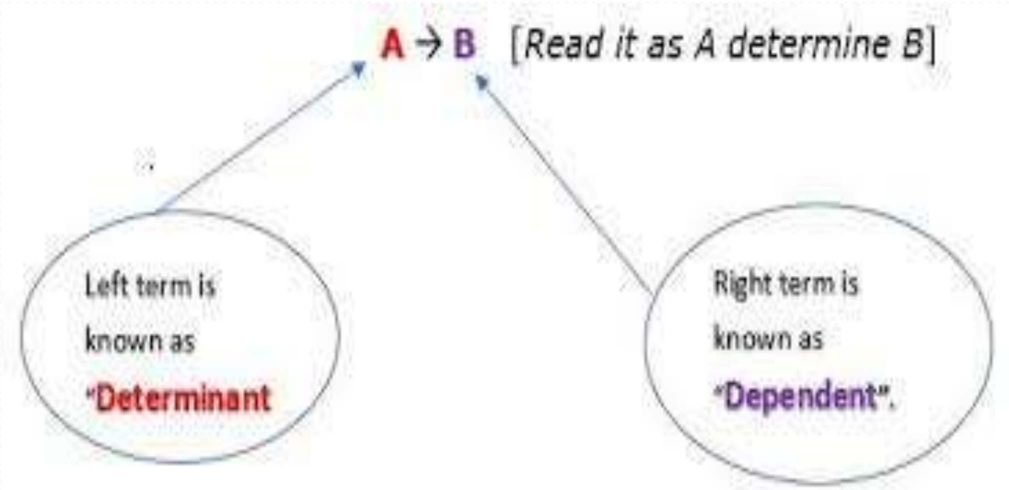
Attribute domain – Every attribute has some pre-defined value and scope which is known as attribute domain

Properties of Relations

- Values are atomic.
- Column values are of the same kind.
- Each row is unique.
- The sequence of columns is insignificant.
- The sequence of rows is insignificant.
- Each column must have a unique name.

Functional Dependency (FD)

- Functional Dependency determines the relation of one attribute to another attribute in a database management system (DBMS) system.
- Functional dependency helps you to maintain the quality of data in the database.
- A functional dependency is denoted by an arrow →.
- The functional dependency of X on Y is represented by X → Y.
- Functional Dependency plays a vital role to find the difference between good and bad database design.



For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.

Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

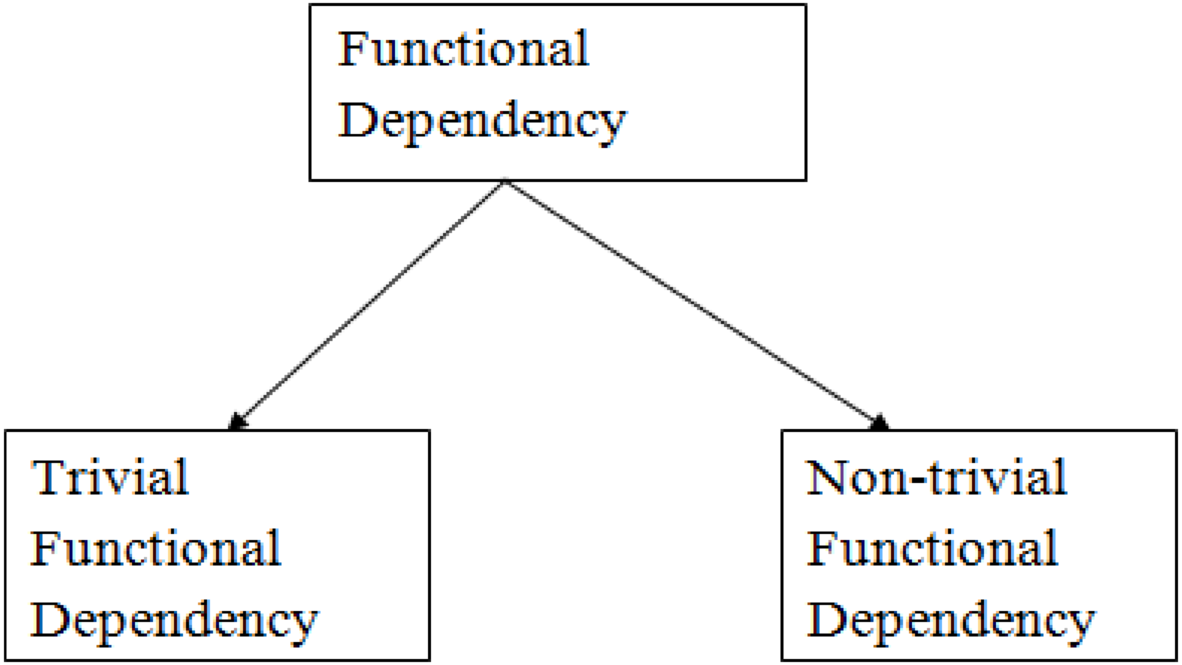
We can say that Emp_Name is functionally dependent on Emp_Id.

Roll_no	Name	GPA
1	Prabin	3
2	Suman	3
3	Suman	3.5
4	Tilak	2.5

A → B
Roll_no → Name
Roll_no →GPA
1 → Prabin
2 → Suman
3 → Suman

GPA → NAME
3 → Prabin
3 → Suman

Types of Functional dependency



Trivial Functional dependency:

$A \rightarrow B$ has trivial functional dependency if B is a subset of A.

$A \cap B \neq \phi$

Consider a table with two columns Employee_Id and Employee_Name.

{Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}.

Non-trivial functional dependency

$A \rightarrow B$ has a non-trivial functional dependency if B is not a subset of A.

When $A \cap B = \phi$, then $A \rightarrow B$ is called as complete non-trivial.

ID → Name,

Name → DOB

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.
- The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR_1)

In the reflexive rule, if Y is a subset of X, then X determines Y.

If X ⊇ Y then X → Y

SID → SID

2. Augmentation Rule (IR_2)

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

If X → Y then XZ → YZ

SID → NAME then SID PHONE → NAME,PHONE

3. Transitive Rule (IR_3)

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If X → Y and Y → Z then X → Z

SID → NAME and NAME → CITY then SID → CITY

4. Union Rule (IR_4)

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If X $\rightarrow$ Y and X $\rightarrow$ Z then X $\rightarrow$ YZ

5. Decomposition Rule (IR_5)

Decomposition rule is also known as project rule. It is the reverse of union rule.

This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If X $\rightarrow$ YZ then X $\rightarrow$ Y and X $\rightarrow$ Z

6. Pseudo transitive Rule (IR_6)

In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If X $\rightarrow$ Y and Z $\rightarrow$ W then XZ $\rightarrow$ W

Normalization of Database

- Database Normalization is a technique of organizing the data in the database.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies.
- It is a multi-step process that puts data into tabular form, removing duplicated data from the relation tables.
- **Normalization is used for mainly two purposes:**
    - Eliminating redundant(useless) data.
    - Ensuring data dependencies make sense i.e data is logically stored.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are :

- Insertion anomaly
- Update anomaly
- Deletion anomaly

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Sabin	Pulchowk	D001
101	Sabin	Pulchowk	D002
123	Mohan	New Road	D890
166	Rabin	Kalimati	D900
166	Rabin	Kalimati	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly: In the above table we have two rows for employee Sabin as he belongs to two departments of the company. If we want to update the address of Sabin then we have to update the same in two rows or the data will become inconsistent. If somehow, the correct address gets updated in one department but not in other then as per the database, Sabin would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn’t allow nulls.

Delete anomaly: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Mohan since he is assigned only to this department.

Normalization Rule

Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF
5. Fourth Normal Form

First Normal Form (1NF)

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

roll_no	name	subject
101	Ram	DBMS, C
103	Shyam	Java
102	Sita	C, C++

As per the 1st Normal form each column must contain atomic value.

Above table is not in first normal form

How to solve this Problem?

roll_no	name	subject
101	Ram	DBMS
101	Ram	C
103	Shyam	Java
102	Sita	C
102	Sita	C++

By doing so, although a few values are getting repeated but values for the subject column are now atomic for each record/row. Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

- It should be in the First Normal form.
- And, all the non prime key should be fully functional depend on candidate key. Or there should be no partial dependency in the relation.
- Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.

Customer_id	Store_id	location
1	1	Pulchowk
1	3	Kalimati
2	1	Pulchowk
3	2	New Road
4	3	Kalimati

Customer_id	Store_id	location
1	1	Pulchowk
1	3	Kalimati
2	1	Pulchowk
3	2	New Road
4	3	Kalimati

Candidate key : customer_id , store_id
Prime attributes: customer_id , store_id
Non prime attribute: Location

Customer_id	Store_id
1	1
1	3
2	1
3	2
4	3

Store_id	location
1	Pulchowk
2	New Road
3	Kalimati

3rd Normal Form Definition

A database is in third normal form if it satisfies the following conditions:

- It is in second normal form
- There is no transitive functional dependency

By transitive functional dependency, we mean we have the following relationships in the table: A is functionally dependent on B, and B is functionally dependent on C. In this case, C is transitively dependent on A via B.

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

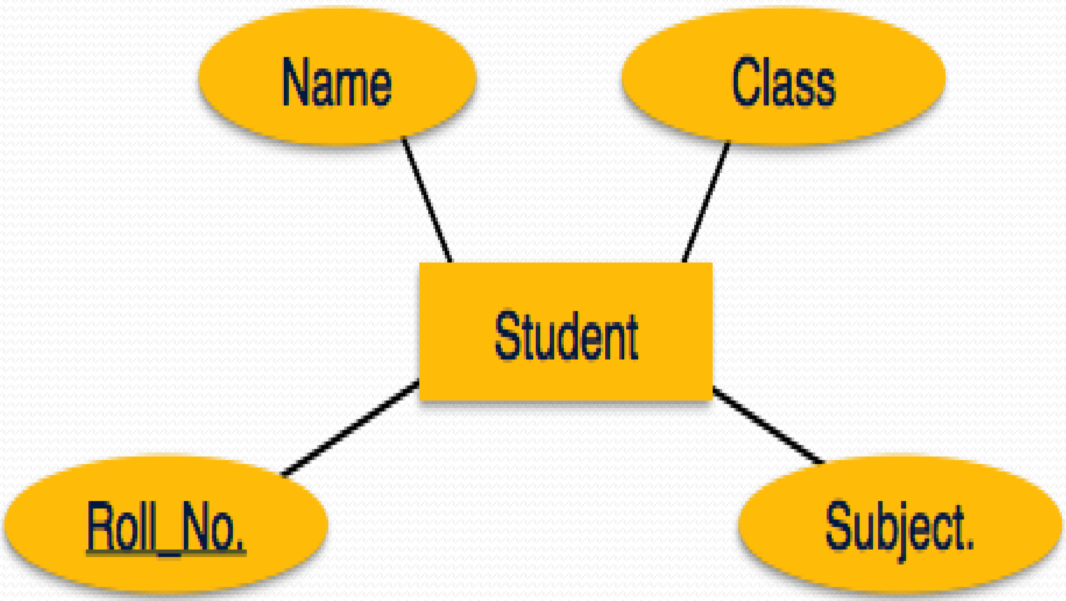
EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Convert ER diagram to relational tables

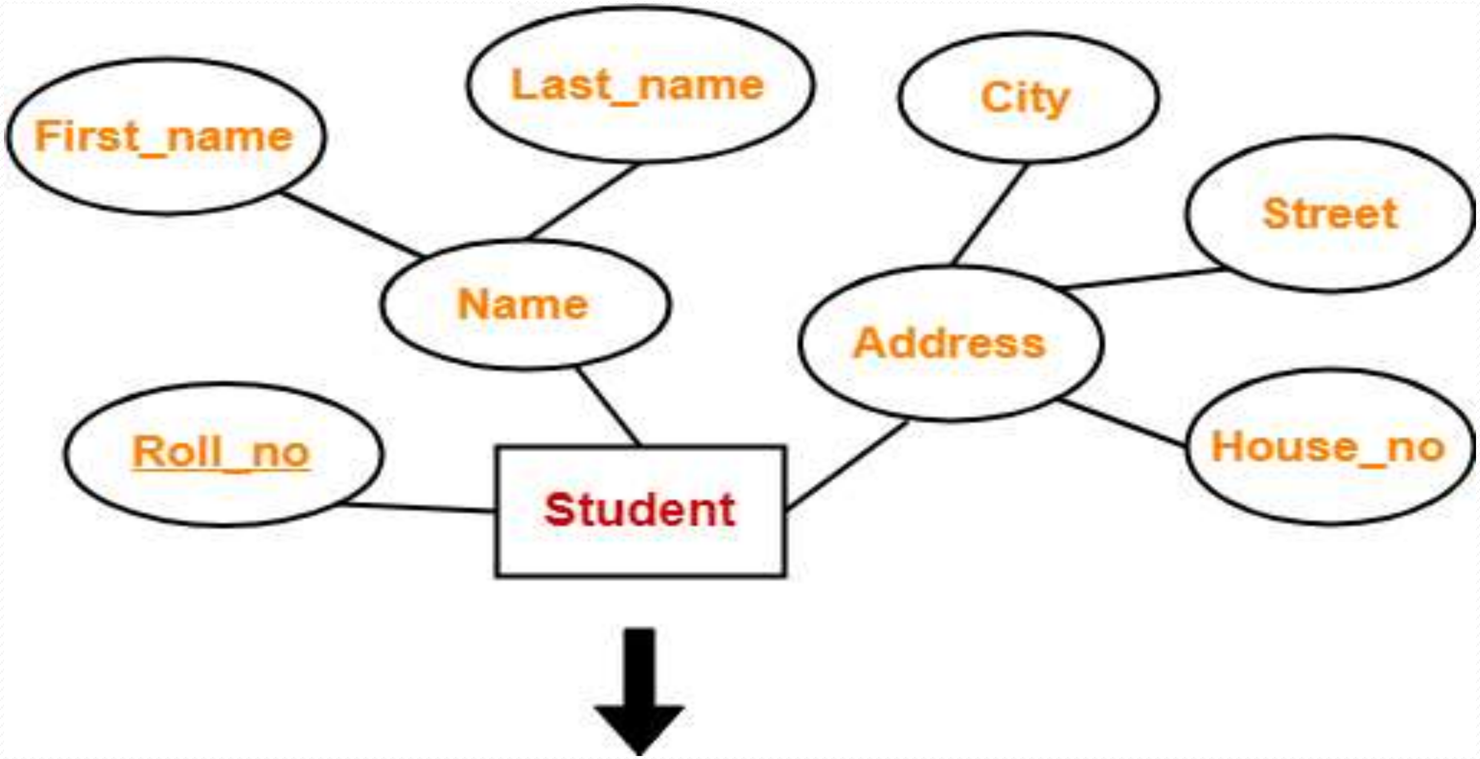
- Entity becomes a table.
- All single-valued attribute becomes a column for the table.
- A key attribute of the entity type represented by the primary key.
- The multivalued attribute is represented by a separate table.
- Any composite attributes are merged into same table as different columns.
- Derived attributes are not considered in the table.

Strong Entity Set With Only Simple Attributes



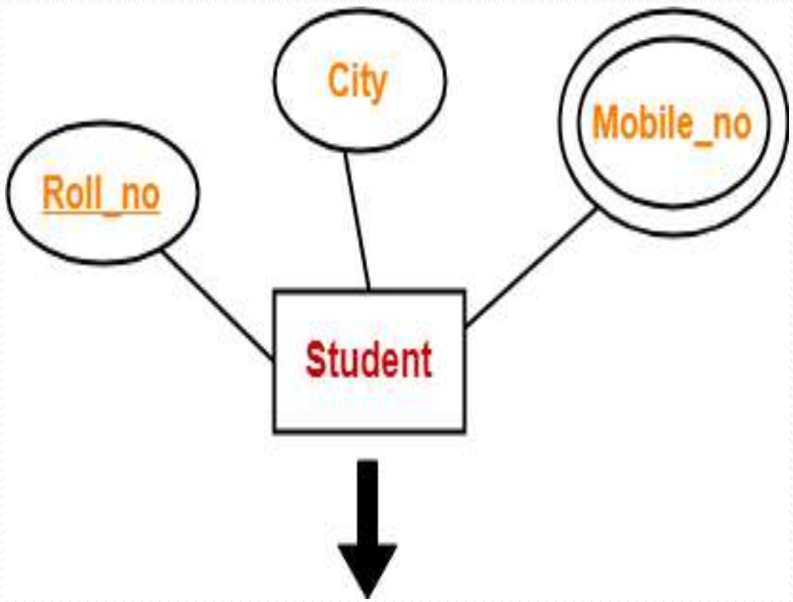
Schema : Student (<u>Roll_no</u> , Name , class,subject)

Strong Entity Set With Composite Attributes



Schema : Student (<u>Roll_no</u> , First_name , Last_name , House_no , Street , City)

Strong Entity Set With Multi Valued Attributes



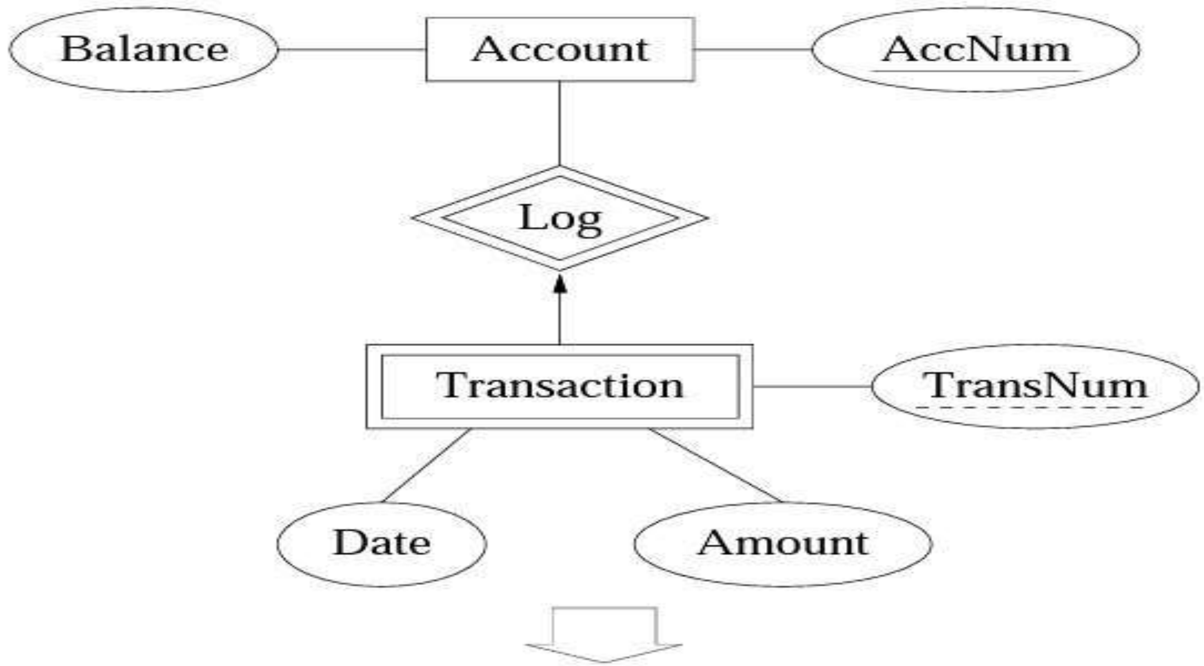
Roll_no	City

Roll_no	Mobile_no

Weak Entity

- For each weak entity create a separate table with the same name.
- Include all attributes.
- Include the Primary key of a strong entity as foreign key is the weak entity.
- Declare the combination of foreign key and decimator attribute as Primary key from the weak entity.

Weak Entity



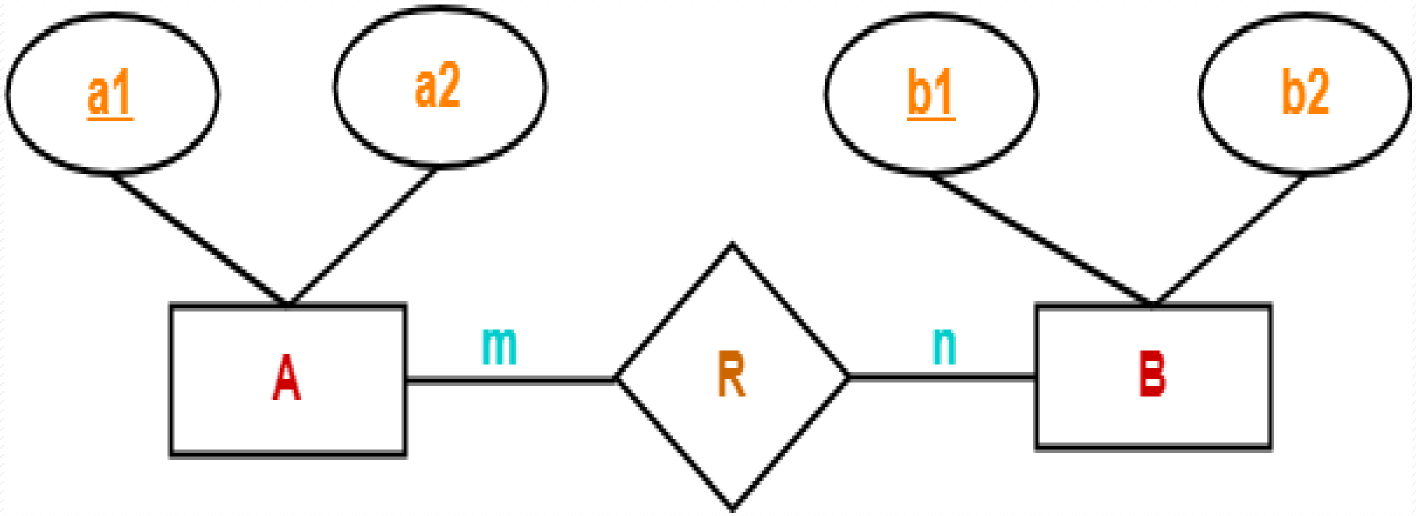
Account

AccNum	Balance

Transaction

TransNum	AccNum	Date	Amount

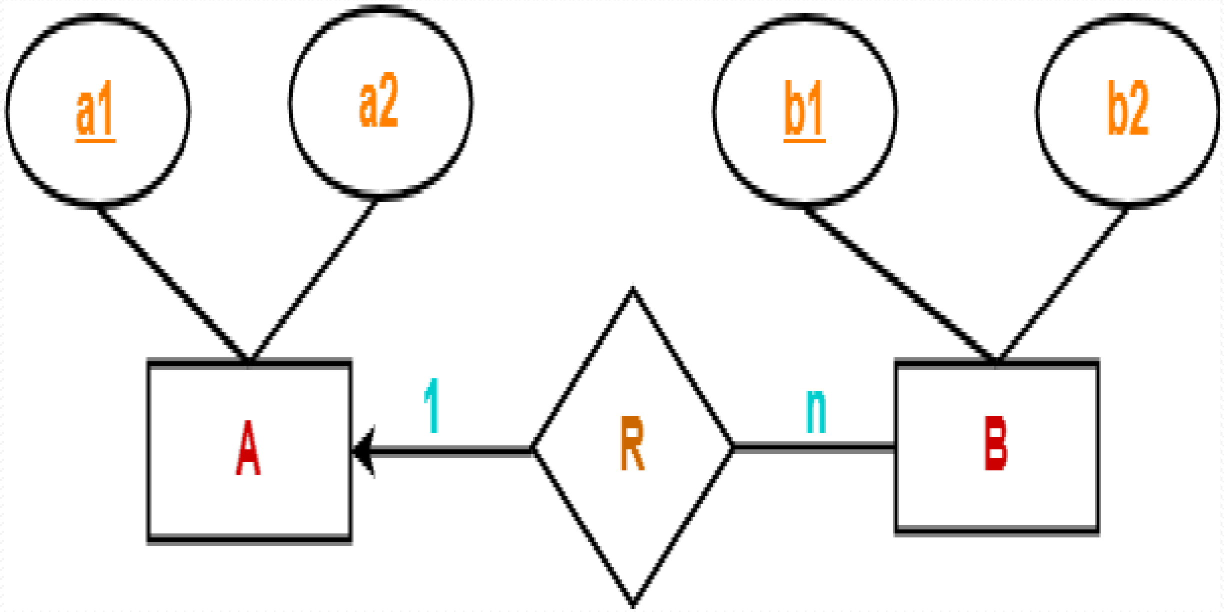
Binary Relationship With Cardinality Ratio m:n



Three tables will be required-

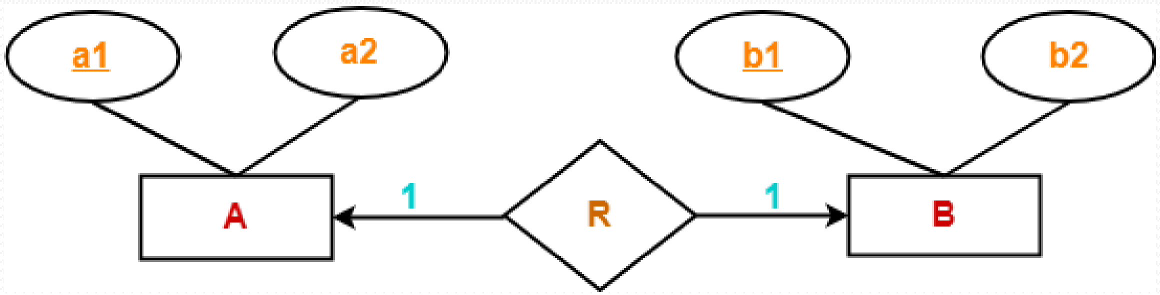
- A (a1 , a2)
- R (a1 , b1)
- B (b1 , b2)

Binary Relationship With Cardinality Ratio 1:n



- Here, two tables will be required-
- A (<u>a1</u> , a2)
- BR (a1 , <u>b1</u> , b2)

Binary Relationship With Cardinality Ratio 1:1



Way-01:

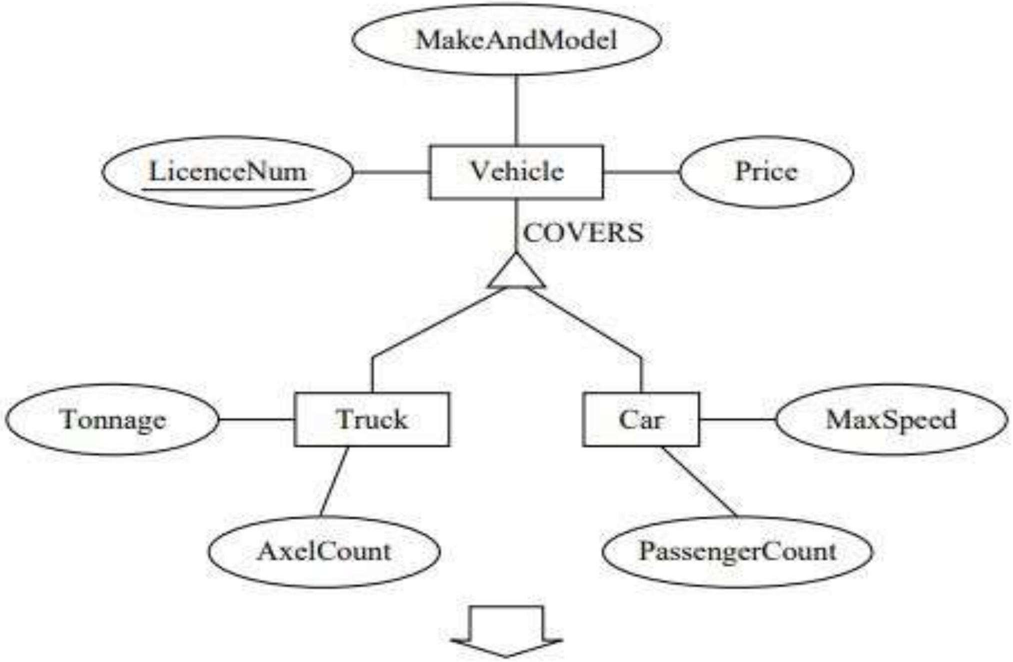
- AR (<u>a1</u> , a2 , b1)
- B (<u>b1</u> , b2)

Way-02:

- A (<u>a1</u> , a2)
- BR (a1 , <u>b1</u> , b2

Representing Generalization (Approach #1)

Example:



Truck

<u>LicenceNum</u>	MakeAndModel	Price	Tonnage	AxelCount

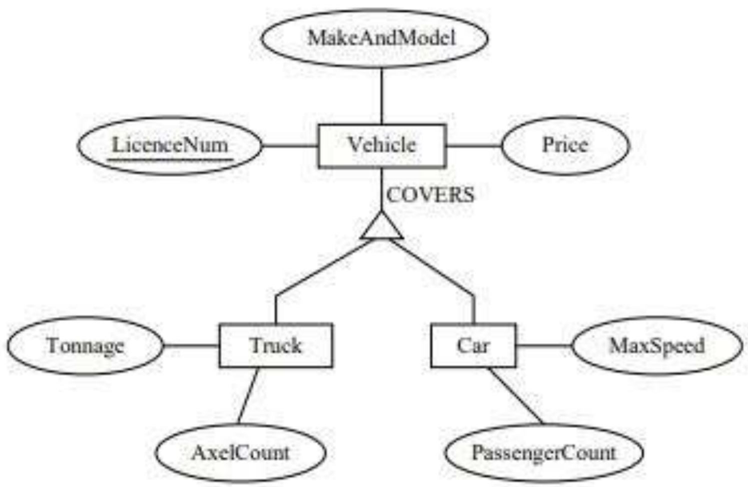
Car

<u>LicenceNum</u>	MakeAndModel	Price	MaxSpeed	PassengerCount

Representing Generalization (Approach #2)

Treat generalization the same as specialization.

Example:



Vehicle

LicenceNum	MakeAndModel	Price

Truck

LicenceNum	Tonnage	AxelCount

Car

LicenceNum	MaxSpeed	PassengerCount