SQL Tutorial

SQL

Agenda

Agenda

What is Normalisation ?

Normalization is a technique for designing relational database tables to minimize duplication of information and increase the logical consistency

Why do we perform Normalisation ?

Redundancy: Repetition of similar data at multiple places

	XXX
	XXX
	XXX

01

To remove redundancy in the database

Whenever we construct a table in a Relational Database, there are some attributes which repeat themselves. In order to remove those redundant values we perform normalisation

Why do we perform Normalisation ?

Redundancy: Repetition of similar data at multiple places

	XXX
	XXX
	XXX

01

To remove redundancy in the database

02

To maintain consistency & integrity

Consistency

Multiple users access a database. Some read the values while some others update these same values. Consistency is ensuring that this data is same across all it present copies in the database.

Integrity

Receiving data without any manipulation. If the data is true, then it upholds integrity.

Whenever we construct a table in a Relational Database, there are some attributes which repeat themselves. In order to remove those redundant values we perform normalisation

If I make a change at a particular place, I don't have to repeatedly perform change across various places so eventually consistency or the integrity is maintained

Why do we perform Normalisation ?

Employee_ID	Name	Role	Manager	Office_num
101	Ashok Kumar	Research Analyst	David	+91-7022374614
102	Prakhar Ranjan	Research Analyst	David	+91-7022374614
103	Kodeeswaran	Research Analyst	David	+91-7022374614
104	Ramendra Tripathi	Research Analyst	David	+91-7022374614
105	Anirudh	Research Analyst	David	+91-7022374614

In this case, **Role**, **Manager**, **Office_num** are same for all the entries we make.
Whenever a new employee entry is created, this entire information is repeated. This is Redundancy

Agenda

Anomalies

In case we do not perform normalisation, there arise certain Anomalies

Insert

Insert

Delete

Update

Employee_ID	Name	Role	Manager	Office_num
101	Ashok Kumar	Research Analyst	David	+91-7022374614
102	Prakhar Ranjan	Research Analyst	David	+91-7022374614
103	Kodeeswaran	Research Analyst	David	+91-7022374614
104	Ramendra Tripathi	Research Analyst	David	+91-7022374614
105	Anirudh	Research Analyst	David	+91-7022374614

- If we have to add a 6th employee, you will have to repeat the same data which holds the **Role**, **Manager**, **Office_num** again.
- If we have to enter the data for 1000 more employees, we have to repeat the data 1000 more times which leads to an **Insertion anomaly** because the repetition of data increases as we insert more entries to our table

Delete

Insert

Delete

Update

Employee_ID	Name	Role	Manager	Office_num
101	Ashok Kumar	Research Analyst	David	+91-7022374614
102	Prakhar Ranjan	Research Analyst	David	+91-7022374614
103	Kodeeswaran	Research Analyst	David	+91-7022374614
104	Ramendra Tripathi	Research Analyst	David	+91-7022374614
105	Anirudh	Research Analyst	David	+91-7022374614

If company wants to fire everyone and hire for new employees, we will also lose the information of **Role**, **Manager**, **Office_num**. This is **Deletion Anomaly**

Update

Insert

Delete

Update

Employee_ID	Name	Role	Manager	Office_num
101	Ashok Kumar	Research Analyst	David	+91-7022374614
102	Prakhar Ranjan	Research Analyst	David	+91-7022374614
103	Kodeeswaran	Research Analyst	David	+91-7022374614
104	Ramendra Tripathi	Research Analyst	David	+91-7022374614
105	Anirudh	Research Analyst	David	+91-7022374614

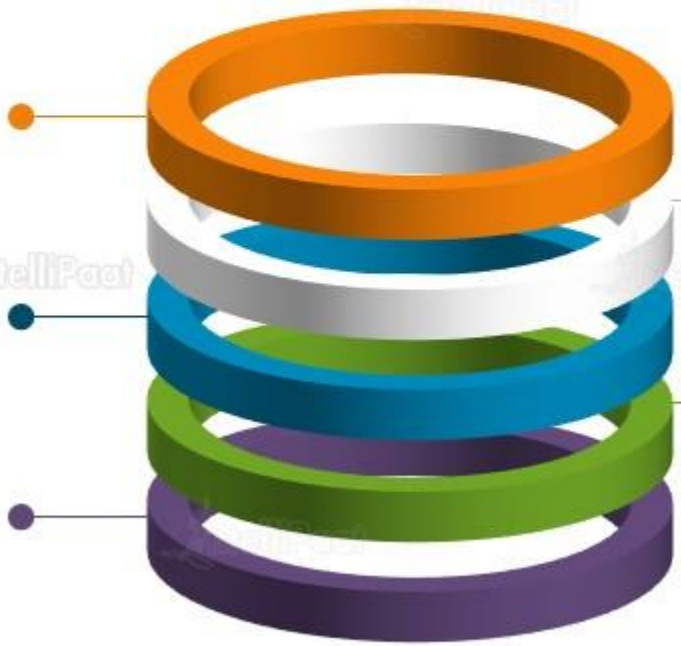
David might leave the organisation or is no longer the manager. In that case all the employee records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is **Updation anomaly**.

Agenda

Dependencies

Dependencies in Database Management System is a relation between two or more attributes (Columns)

- Functional dependency
- Partial functional dependency
- Multi-valued functional dependency



- Full functional dependency
- Transitive functional dependency

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

When we say A determines B : A $\longrightarrow$ B

- It means for instance of A, we will have unique values of B
- If we have: $a_1 \longrightarrow b_3$, in the entire table, a_1 will result to b_3
- There will not be a case where:

$$\left.\begin{array}{l} a_1 \longrightarrow b_2 \\ a_1 \longrightarrow b_3 \end{array}\right\} \times$$

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

A ⟶ B
A ⟶ C
C ⟶ D

Lets try to identify some of the functional dependencies for this table :

A ⟶ B

- Our approach is to discard this
- lets try to find out an element in A column corresponding to which two different b's are there
- lets choose a_1
- So across a_1, I will try to find out 2 different values of b

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

A ⟶ B
A ⟶ C
C ⟶ D

Lets try to identify some of the functional dependencies for this table :

A ⟶ B

- We have 2 different elements corresponding to a single element

$a_1 \longrightarrow b_1$
$a_1 \longrightarrow b_2$ } ✗

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Lets try to identify some of the functional dependencies for this table :

A ⟶ B

- We have 2 different elements corresponding to a single element

$a_1 \longrightarrow b_1$
$a_1 \longrightarrow b_2$ } ✗

A ⟶ B ✗
A ⟶ C
C ⟶ D

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Lets try to identify some of the functional dependencies for this table :

A → B ✗

A → C

C → D

A → C

$a_1 \rightarrow c_1$

$a_1 \rightarrow c_1$

$a_2 \rightarrow c_2$

$a_2 \rightarrow c_2$

$a_3 \rightarrow c_2$ ✓

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Lets try to identify some of the functional dependencies for this table :

A → B ✗

A → C ✓

C → D

$a_1 \rightarrow c_1$, $a_1 \rightarrow c_1$, $a_2 \rightarrow c_2$, $a_2 \rightarrow c_2$, $a_3 \rightarrow c_2$ ✓

A → C

$a_2 \rightarrow c_2$, $a_3 \rightarrow c_2$ ✓

We can have elements a2 & a3 deriving same element c3

$a_2 \rightarrow c_1$, $a_2 \rightarrow c_2$ ✗

But we cannot have it in reverse

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Lets try to identify some of the functional dependencies for this table :

A ⟶ B ✗

A ⟶ C ✓

C ⟶ D

C ⟶ D

$c_1 \longrightarrow d_1$

$c_1 \longrightarrow d_2$ } ✗

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Lets try to identify some of the functional dependencies for this table :

A ⟶ B ✗

A ⟶ C ✓

C ⟶ D ✗

C ⟶ D

$c_1 \longrightarrow d_1$

$c_1 \longrightarrow d_2$ } ✗

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Functional Dependency

If the information stored in a table can uniquely determine another information in the same table, then it is called **Functional Dependency**

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Functional Dependency

If the information stored in a table can uniquely determine another information in the same table, then it is called **Functional Dependency**

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Employee_Name	Employee_Id	Experience
Chintan	200	2
Tarun Singh	201	6
Sabid Ansari	202	4
Chintan	203	7
Adarsh Khare	204	3

We have a database of an Employee with name, id and Experience. How can I find the experience of Chintan ?

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Employee_Name	Employee_Id	Experience
Chintan	200	2
Tarun Singh	201	6
Sabid Ansari	202	4
Chintan	203	7
Adarsh Khare	204	3

There are 2 people with the same name Chintan.
Whose experience should I fetch ?

Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Employee_Name	Employee_Id	Experience
Chintan	200	2
Tarun Singh	201	6
Sabid Ansari	202	4
Chintan	203	7
Adarsh Khare	204	3

We can determine the experience of an employee using Employee_Id
It means Employee_Id is uniquely associated with Experience

Employee_Id → Experience
Employee_Id → Employee_Name
} Functional dependency

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

$BC \longrightarrow D$ ✗

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

$BC \longrightarrow D$ ✗

$C \longrightarrow D$ ✗

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

$BC \longrightarrow D$ ✗

$C \longrightarrow D$ ✗

$A \longrightarrow D$ ✗

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

$BC \longrightarrow D$ ✗

$C \longrightarrow D$ ✗

$A \longrightarrow D$ ✗

} Any proper subset of ABC cannot determine D

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Let's take a functional dependency

$X \longrightarrow Y$

Y is said to be fully functional dependent if it cannot be determined by any of the subset of X

$ABC \longrightarrow D$

D cannot be determined by any of the subset (or) proper subset of BC

$BC \longrightarrow D$ ✗
$C \longrightarrow D$ ✗
$A \longrightarrow D$ ✗

} Any proper subset of ABC cannot determine D

As only ABC determines D, we can say that D is Fully functional dependent on ABC

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Fully Functional Dependency

An attribute is **Fully Functional Dependent** on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Fully Functional Dependency

An attribute is **Fully Functional Dependent** on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

How will you determine the Grade ?

Full Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num, Course_Id ⟶ Grade ✓

Roll_num & **Course_Id** are fully functional Dependent

Full Functional Dependency

- Functional Dependency
- **Full Functional Dependency**
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num, Course_Id ⟶ Grade ✓

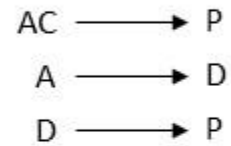
Roll_num & **Course_Id** are fully functional Dependent

Name and **Subject** are not fully functional dependent on Composite key

Composite key refers to cases where more than one column is used to specify the primary key of a table

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency



Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

AC → P

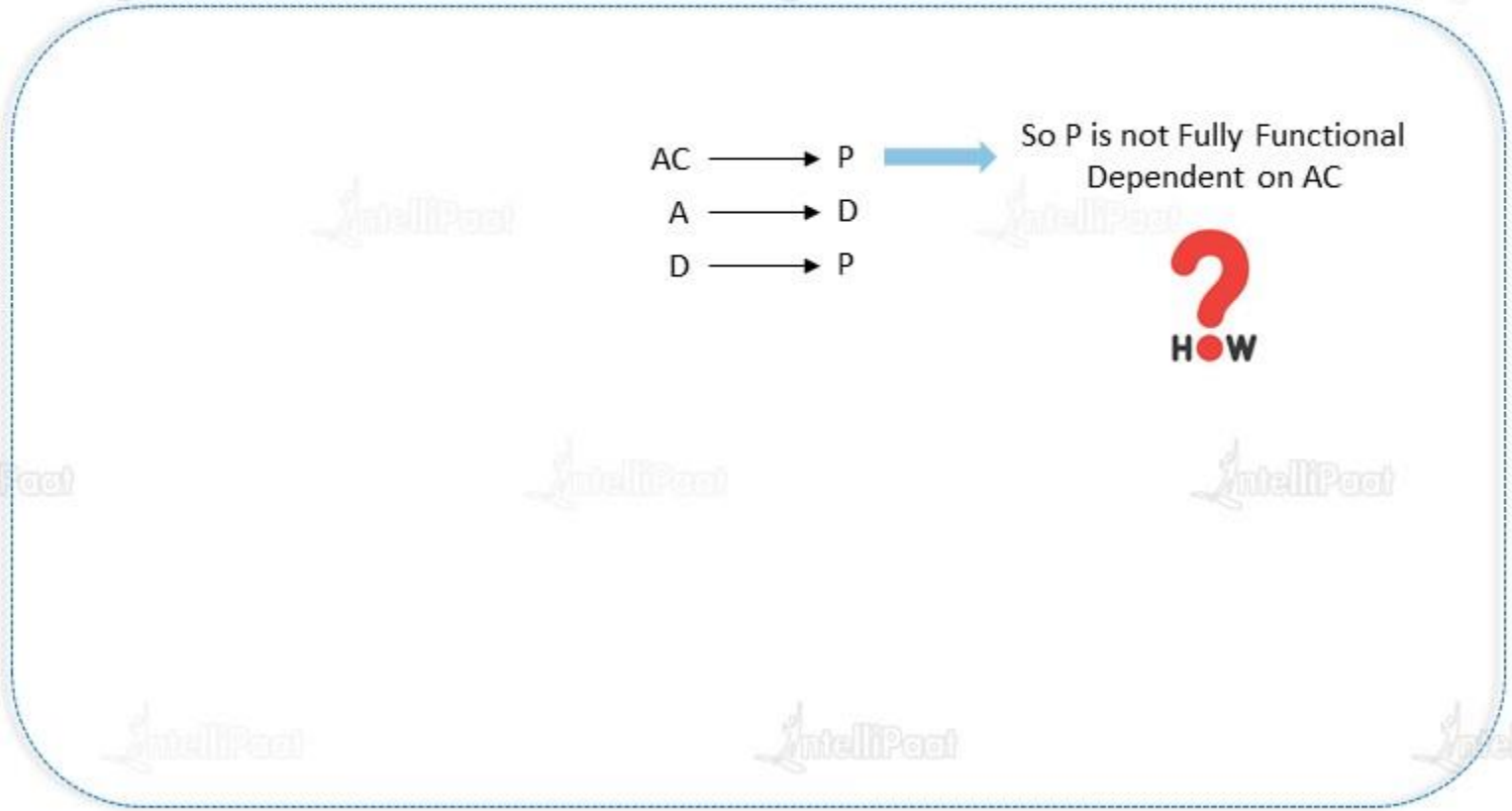
A → D

D → P

So P is not Fully Functional Dependent on AC

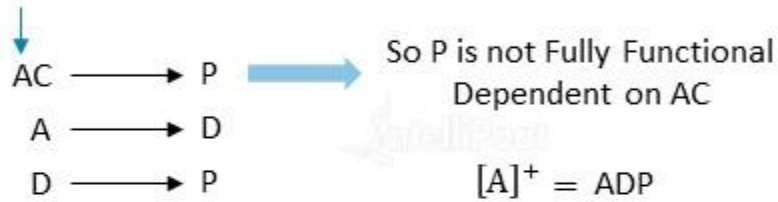
Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- **Partial Functional Dependency**
- Transitive Functional Dependency
- Multi-valued Dependency



Partial Functional Dependency

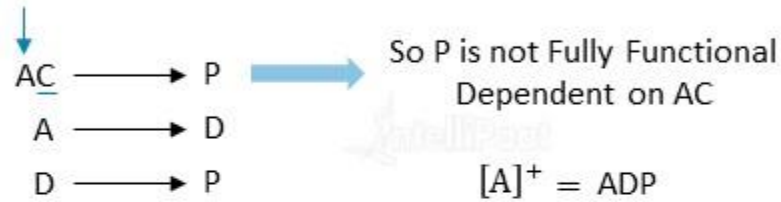
- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency



- A alone can determine P

Partial Functional Dependency

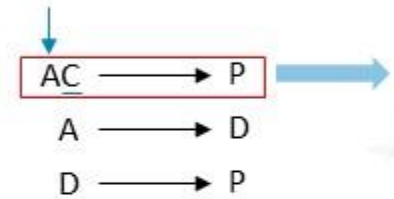
- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency



- A alone can determine P
- C is extraneous attribute.
- Even with out C, we can determine P

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency



So P is not Fully Functional Dependent on AC

$[A]^+ = ADP$

- A alone can determine P
- C is extraneous attribute.
- Even with out C, we can determine P

A → P

So P is not Fully Functional Dependent on AC
In reverse we can say that, P is Partially Functional dependent on AC because subset of AC can determine P

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Partial Functional Dependency

In a relation, there exists **Partial Functional Dependency**, when a non prime attribute is functionally dependent on part of a candidate key

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Partial Functional Dependency

In a relation, there exists **Partial Functional Dependency**, when a non prime attribute is functionally dependent on part of a candidate key

Non prime attribute: Attributes which are not a part of any candidate key

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Partial Functional Dependency

In a relation, there exists **Partial Functional Dependency**, when a non prime attribute is functionally dependent on part of a candidate key

Non prime attribute: Attributes which are not a part of any candidate key

Candidate Key ???

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Candidate Key

Candidate Key is a super key with out Redundancy

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Candidate Key

Candidate Key is a super key with out Redundancy

Super Key ???

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Superkey

Combination of attributes that specifies no two tuples of a relationship are same

if [X, Y, Z] is a super key, then for any of the tuple in the relationship, the combination of these three will not be same at a time.

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

If you check these 2 combinations, they are always unique

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

[Emp_Id]

All Emp_Id are unique

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

[Emp_Id]

[Emp_Id , Employee_Name, Manager]

When we take the combination of all the attributes from a relationship, that will be super key

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

[Emp_Id]

[Emp_Id , Employee_Name, Manager]

Note: Super Key may have redundant attributes

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

[Emp_Id]

[Emp_Id , Employee_Name, Manager]

Once we say **Employee_Id** is unique through out, then what is the point of saying **[Emp_Id , Employee_Name, Manager]** ?

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit

[Employee_Name, Manager]

[Emp_Id]

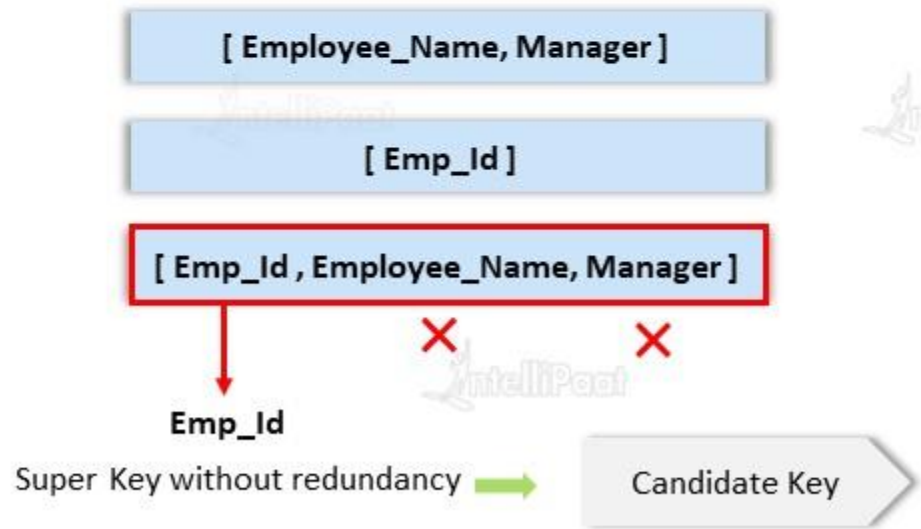
[Emp_Id , Employee_Name, Manager]

In this case, **Employee_Name** and **Manager** are redundant attributes

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- **Partial Functional Dependency**
- Transitive Functional Dependency
- Multi-valued Dependency

Emp_Id	Employee_Name	Manager
1	Tarun	Mahesh
2	Chintan	Mahesh
3	Tarun	Rohit
4	Sabid	Suresh
5	Chintan	Rohit



Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Partial Functional Dependency

In a relation, there exists **Partial Functional Dependency**, when a non prime attribute is functionally dependent on part of a candidate key

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

Roll_num, **Course_id**, **Subject** are the candidate keys as they are not present in the Right Hand Side

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

Roll_num, **Course_id**, **Subject** are the candidate keys as they are not present in the Right Hand Side

Prime Attributes : Attributes which are present in any of the candidate keys

Roll_num, Course_Id, Subject

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- **Partial Functional Dependency**
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

Roll_num, **Course_id**, **Subject** are the candidate keys as they are not present in the Right Hand Side

Prime Attributes : Attributes which are present in any of the candidate keys

Roll_num, Course_Id, Subject

Non prime attribute: Attributes which are not a part of any candidate key

Name, Grade

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- **Partial Functional Dependency**
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

In order to be to be Partial Dependent, non-prime attributes should be functionally dependent on part of a candidate key

Partial Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Roll_num	Name	Course_Id	Subject	Grade
1	Ashok	CSE301	DBMS	A
1	Ashok	CSE306	DS	C
2	Tarun	CSE301	DBMS	B
2	Tarun	CSE306	DS	A
3	Charan	CSE316	AI	C

Roll_num → Name

Roll_num, Course_id → Grade

Roll_num, Subject → Grade

In order to be to be Partial Dependent, non-prime attributes should be functionally dependent on part of a candidate key

Name is partially dependent on Roll_num.

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

There is a transitive relationship between the functional dependencies

$X \longrightarrow Y$

$Y \longrightarrow Z$

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

There is a transitive relationship between the functional dependencies

$$X \longrightarrow Y$$
$$Y \longrightarrow Z$$

If X determines Y and Y determines Z, then we can say X transitively determines Z

$$X \longrightarrow Z$$

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- **Transitive Functional Dependency**
- Multi-valued Dependency

There is a transitive relationship between the functional dependencies

$$X \longrightarrow Y$$
$$Y \longrightarrow Z$$

If X determines Y and Y determines Z, then we can say X transitively determines Z

$$X \longrightarrow Z$$

Transitive Functional Dependency

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

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

There is a transitive relationship between the functional dependencies

$$X \longrightarrow Y$$
$$Y \longrightarrow Z$$

If X determines Y and Y determines Z, then we can say X transitively determines Z

$$X \longrightarrow Z$$

Lets understand this with an example

Transitive Functional Dependency

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

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- **Transitive Functional Dependency**
- Multi-valued Dependency

Dept_id	Dept_name	Hod
1	CSE	Mr. A
2	ECE	Mr. B
3	MECH	Mr. C
4	CIVIL	Mr. D

Dep_id → Dept_name

Dept_name → Hod

Transitive Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- **Transitive Functional Dependency**
- Multi-valued Dependency

Dept_id	Dept_name	Hod
1	CSE	Mr. A
2	ECE	Mr. B
3	MECH	Mr. C
4	CIVIL	Mr. D

Dep_id ⟶ Dept_name

Dept_name ⟶ Hod

Dept_id ⟶ Dept_name

Dep_id determines **Dept_name** and **Dept_name** determines **Hod,** so we can say **Dep_id** transitively determines **Hod**

Multi-valued Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Multi-valued Functional Dependency

Multi-valued Functional Dependency occurs when there are more than one independent multivalued attributes in a table

Multi-valued Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Multi-valued Functional Dependency

Multi-valued Functional Dependency occurs when there are more than one independent multivalued attributes in a table

Multi-valued Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Consider a Car manufacture company, which produces two colors (Blue and Red) in each model every year

Car_model	Manufacture_year	Colour
Hyundai Creta	2017`	Blue
Hyundai Creta	2017	Red
Kia Seltos	2018	Blue
Kia Seltos	2018	Red
Volkswagen T-Roc	2019	Blue
Volkswagen T-Roc	2019	Red

Multi-valued Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Consider a Car manufacture company, which produces two colors (Blue and Red) in each model every year

Car_model	Manufacture_year	Colour
Hyundai Creta	2017`	Blue
Hyundai Creta	2017	Red
Kia Seltos	2018	Blue
Kia Seltos	2018	Red
Volkswagen T-Roc	2019	Blue
Volkswagen T-Roc	2019	Red

Manufacture_year and **Colour** are independent of each other and dependent on **Car_model**

Multi-valued Functional Dependency

- Functional Dependency
- Full Functional Dependency
- Partial Functional Dependency
- Transitive Functional Dependency
- Multi-valued Dependency

Consider a Car manufacture company, which produces two colors (Blue and Red) in each model every year

Car_model	Manufacture_year	Colour
Hyundai Creta	2017`	Blue
Hyundai Creta	2017	Red
Kia Seltos	2018	Blue
Kia Seltos	2018	Red
Volkswagen T-Roc	2019	Blue
Volkswagen T-Roc	2019	Red

Manufacture_year and **Colour** are independent of each other and dependent on **Car_model**

These two columns are said to be Multi-valued functional dependent on Car_model. These dependencies can be represented like this:

Car_model ⟶⟶ Manufacture_year

Agenda

Decomposition

Decomposition

The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

Decomposition

Decomposition

The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

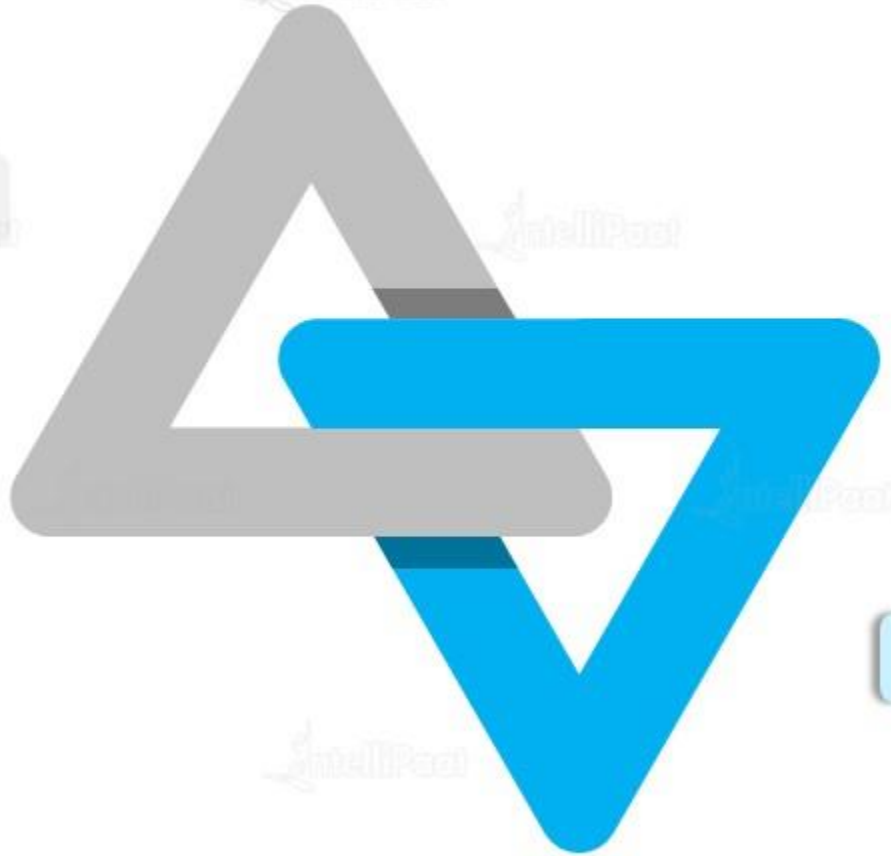
Let R be a Relation Schema

A set of relation schemas(R_1, R_2, R_3..... R_n) is a decomposition of R if

$$R = R_1 U R_2 U R_3 \cdot \cdots U R_n$$

Types of Decomposition

Lossless Join Decomposition



Dependency Preserving Decomposition

Lossless Join Decomposition

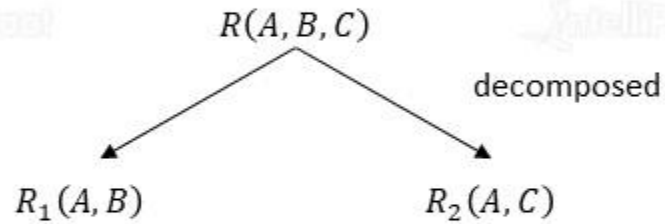
- Lossless Join Decomposition
- Dependency Preserving Decomposition

Decomposition $\{R_1, R_2, R_3 \ldots\ldots R_n\}$ of a relation R is called **Lossless decomposition** for R if the natural join of $R_1, R_2, R_3 \ldots\ldots R_n$ produces exactly the relation R

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

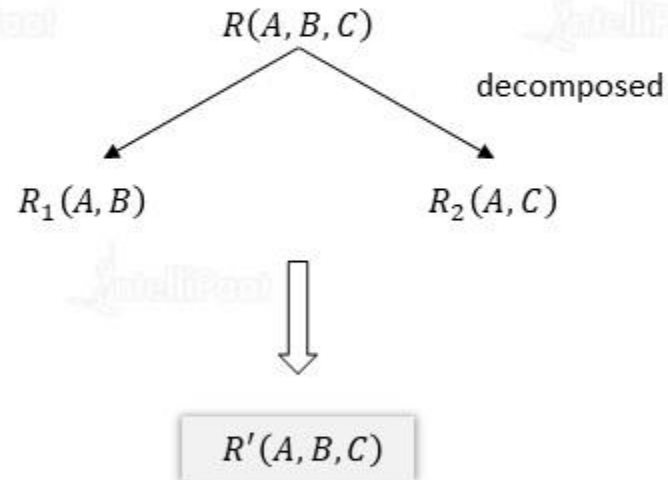
Decomposition $\{R_1, R_2, R_3 \ldots\ldots R_n\}$ of a relation R is called **Lossless decomposition** for R if the natural join of $R_1, R_2, R_3 \ldots\ldots R_n$ produces exactly the relation R



Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

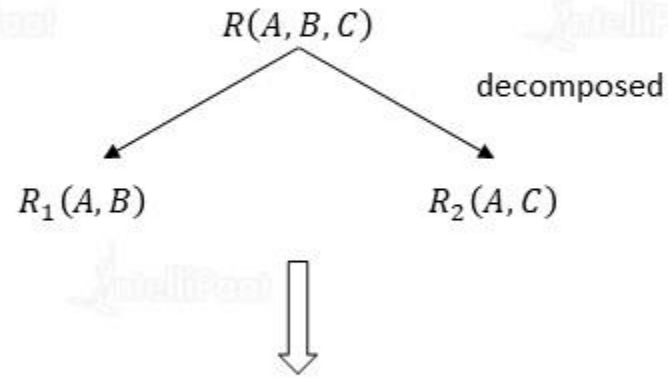
Decomposition $\{R_1, R_2, R_3 \ldots\ldots R_n\}$ of a relation R is called **Lossless decomposition** for R if the natural join of $R_1, R_2, R_3 \ldots\ldots R_n$ produces exactly the relation R



Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

Decomposition $\{R_1, R_2, R_3 \ldots\ldots R_n\}$ of a relation R is called **Lossless decomposition** for R if the natural join of $R_1, R_2, R_3 \ldots\ldots R_n$ produces exactly the relation R



$R'(A, B, C)$

After natural join of above relations, you will get exact relation

$R = R'$

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

How to ensure Lossless Join Decomposition

Lossless Join Decomposition

Lossless Join Decomposition

Dependency Preserving Decomposition

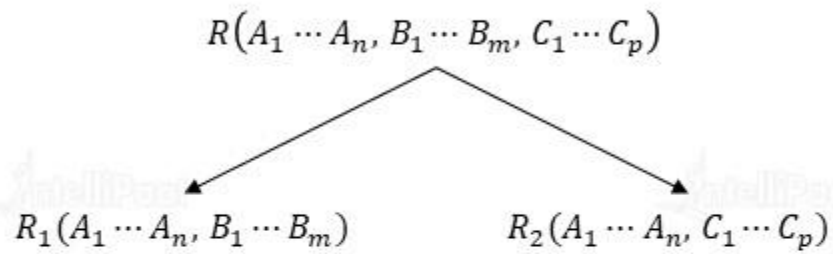
$$R(A_1 \cdots A_n, B_1 \cdots B_m, C_1 \cdots C_p)$$

These are the attributes that are present in the relation

Lossless Join Decomposition

Lossless Join Decomposition

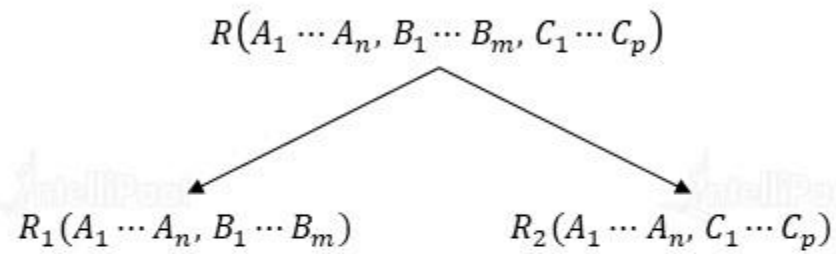
Dependency Preserving Decomposition



Lossless Join Decomposition

Lossless Join Decomposition

Dependency Preserving Decomposition



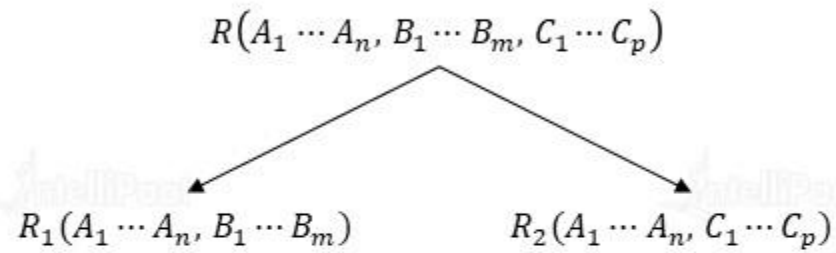
$A_1 \cdots A_n \longrightarrow B_1 \cdots B_m$

(or)

$A_1 \cdots A_n \longrightarrow C_1 \cdots C_p$

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition



$A_1 \cdots A_n \longrightarrow B_1 \cdots B_m$

(or)

$A_1 \cdots A_n \longrightarrow C_1 \cdots C_p$

When Functionally dependency exists between the attributes then decomposition is lossless

Lossless Join Decomposition

When a relation R is decomposed into two relations R_1 & R_2 then it will be lossless join if:

Lossless Join Decomposition

Dependency Preserving Decomposition

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

When a relation R is decomposed into two relations R_1 & R_2 then it will be lossless join if:

01

Union of Attributes of R_1 and R_2 must be equal to attribute of R. Each attribute of R must be either in R_1 or in R_2

$$\boldsymbol{Att(R_1) \cup Att(R_2) = Att(R)}$$

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

When a relation R is decomposed into two relations R_1 & R_2 then it will be lossless join if:

01

Union of Attributes of R_1 and R_2 must be equal to attribute of R. Each attribute of R must be either in R_1 or in R_2

$$Att(R_1) \cup Att(R_2) = Att(R)$$

02

Intersection of Attributes of R_1 and R_2 must not be NULL.

$$Att(R_1) \cap Att(R_2) \neq \phi$$

Lossless Join Decomposition

- Lossless Join Decomposition
- Dependency Preserving Decomposition

When a relation R is decomposed into two relations R_1 & R_2 then it will be lossless join if:

01

Union of Attributes of R_1 and R_2 must be equal to attribute of R. Each attribute of R must be either in R_1 or in R_2

$$Att(R_1) \cup Att(R_2) = Att(R)$$

02

Intersection of Attributes of R_1 and R_2 must not be NULL.

$$Att(R_1) \cap Att(R_2) \neq \phi$$

03

Common attribute must be a key for at least one relation (R_1 or R_2)

$$Att(R_1) \cap Att(R_2) \longrightarrow Att(R_1)$$

OR

$$Att(R_1) \cap Att(R_2) \longrightarrow Att(R_2)$$

Dependency Preserving Decomposition

Lossless Join Decomposition

Dependency Preserving Decomposition

If we decompose a relation R into relations R_1 and R_2, All dependencies of R either must be a part of R_1 or R_2 or must be derivable from combination of Functional Dependencies of R_1 and R_2.

Dependency Preserving Decomposition

Lossless Join Decomposition

Dependency Preserving Decomposition

If we decompose a relation R into relations R_1 and R_2, All dependencies of R either must be a part of R_1 or R_2 or must be derivable from combination of Functional Dependencies of R_1 and R_2.

Dependency Preserving Decomposition

Lossless Join Decomposition

Dependency Preserving Decomposition

If we decompose a relation R into relations R_1 and R_2, All dependencies of R either must be a part of R_1 or R_2 or must be derivable from combination of Functional Dependencies of R_1 and R_2.

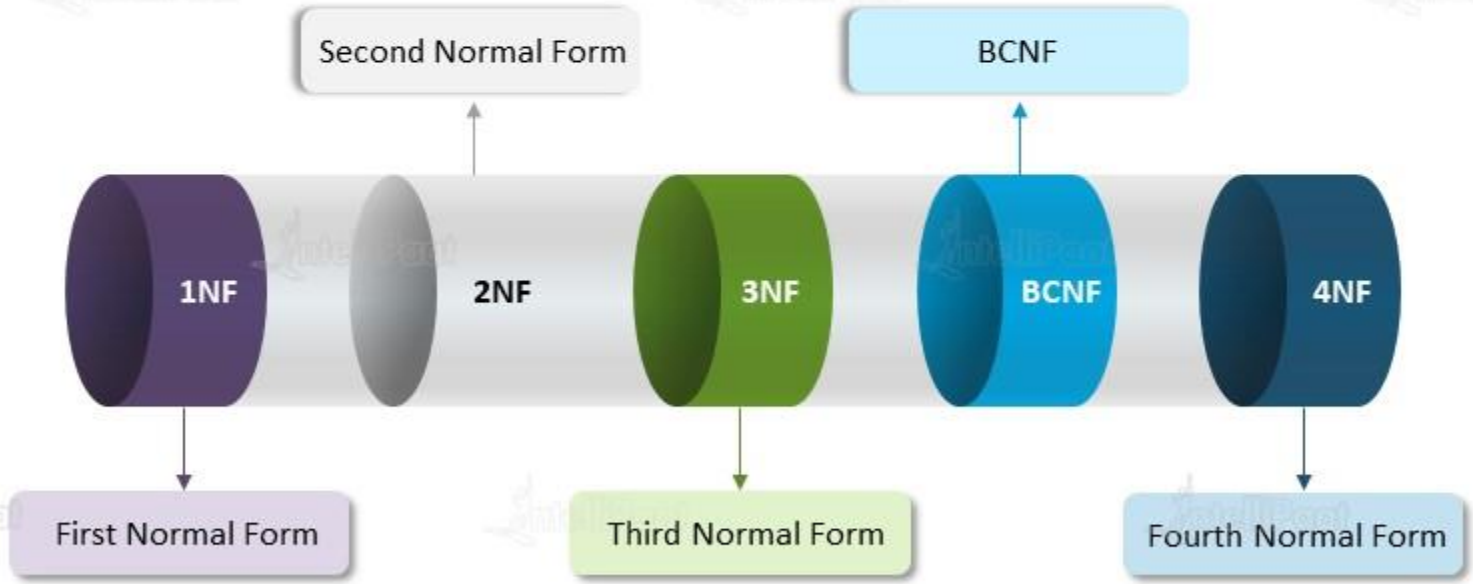
Example

A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R_1(ABC) and R_2(AD) which is dependency preserving because FD A->BC is a part of R_1(ABC).

Agenda

Normal Forms

Normalization rules are divided into the following normal forms:



First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

How to achieve 1st Normal Form?

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

There are 4 basic rules that a table should follow to be in 1st Normal Form:

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

There are 4 basic rules that a table should follow to be in 1st Normal Form:

Rule-1

Each column of your table should be single valued which means they should not contain multiple values.

Col 1	Col 2
P	A,B
Q	C,D
R	E

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

There are 4 basic rules that a table should follow to be in 1st Normal Form:

Rule-2

In each column the values stored must be of the same kind or type.

Col 1	Col 2
P	1
Q	2
3	E

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

There are 4 basic rules that a table should follow to be in 1st Normal Form:

Rule-3

All the columns in a table should have unique names.

Name	ID	ID
P	1	4
Q	2	5
R	3	6

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

There are 4 basic rules that a table should follow to be in 1st Normal Form:

Rule-4

Order doesn't matter

Id	Name	Grade
5	Ashok	O
4	Tarun	A
3	Sai	B

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

1st Normal Form is the basic requirement and Every Table in your Database should at least follow the 1st Normal form

First Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Emp_id	Name	Phone	State	Country
1	Tarun Singh	9760856822, 8360826099	Uttarakhand	India
2	Tarun Singh	9493903163	Uttarpradesh	India
3	Charan	8360331850	Telangana	India

Conversion to First Normal Form

Emp_id	Name	Phone	State	Country
1	Tarun Singh	9760856822	Uttarakhand	India
1	Tarun Singh	8360826099	Uttarakhand	India
2	Tarun Singh	9493903163	Uttarpradesh	India
3	Charan	8360331850	Telangana	India

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the Second Normal Form, it must satisfy two conditions:

- ✓ The table should be in the First Normal Form
- ✓ There should be no Partial Dependency

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the Second Normal Form, it must satisfy two conditions:

- ✓ The table should be in the First Normal Form
- ✓ There should be no Partial Dependency

In a relation, there exists **Partial Functional Dependency**, when a non prime attribute is functionally dependent on part of a candidate key

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the Second Normal Form, it must satisfy two conditions:

- ✓ The table should be in the First Normal Form
- ✓ There should be no Partial Dependency

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Lets say a school wants to store the data of teachers and the subjects they teach.

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Subject	Age
1	DS	26
1	OS	26
2	DBMS	26
3	OS	30
3	AI	30

Employee_Id → Age

Employee_Id, Subject → Age

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Subject	Age
1	DS	26
1	OS	26
2	DBMS	26
3	OS	30
3	AI	30

Employee_Id → Age

Employee_Id, Subject → Age

Candidate Keys: Employee_Id, Subject

Non prime attribute: Age

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Subject	Age
1	DS	26
1	OS	26
2	DBMS	26
3	OS	30
3	AI	30

Employee_Id → Age

Employee_Id, Subject → Age

Candidate Keys: Employee_Id, Subject

Non prime attribute: Age

- Table is in 1 NF because each attribute has atomic values.
- It is not in 2NF because non prime attribute Age is dependent on Employee_id alone which is a proper subset of candidate key.

Second Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To make the table complies with 2NF we can break it in two tables like this:

Employee Age Table

Employee_Id	Age
1	26
1	26
2	26
3	30
3	30

Employee Subject: Table

Employee_Id	Subject
1	DS
1	OS
2	DBMS
3	OS
3	AI

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the third Normal Form, it must satisfy two conditions:

- ✓ It should be in the Second Normal form.
- ✓ And it should not have Transitive Dependency.

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the third Normal Form, it must satisfy two conditions:

- ✓ It should be in the Second Normal form.
- ✓ And it should not have Transitive Dependency.

If A->B and B->C are two FDs then A->C is called **Transitive dependency**

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the third Normal Form, it must satisfy two conditions:

- ✓ It should be in the Second Normal form.
- ✓ And it should not have Transitive Dependency.

If A->B and B->C are two FDs then A->C is called **Transitive dependency**

For each functional dependency A-> B at least one of the following conditions hold:

- A is a super key of table
- B is a prime attribute of table

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To be in the third Normal Form, it must satisfy two conditions:

- ✓ It should be in the Second Normal form.
- ✓ And it should not have Transitive Dependency.

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Lets say a school wants to store the complete address of each student

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Roll	Name	Pin	State	City	District
1	Rahul	535558	AP	Bobbili	Vizianagaram
2	Swetha	243505	UP	Shahi	Bareilly
3	Asmitha	600023	TN	Ayanavaram	Chennai
4	Tarun	212206	UP	Karari	Allahabad

Super keys: {Roll}, {Roll, Name}, {Roll, Name, Pin}.....so on
Candidate Keys: {Roll}
Non-prime attributes: All attributes except Roll are non-prime as they are not part of any candidate keys.

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Roll	Name	Pin	State	City	District
1	Rahul	535558	AP	Bobbili	Vizianagaram
2	Swetha	243505	UP	Shahi	Bareilly
3	Asmitha	600023	TN	Ayanavaram	Chennai
4	Tarun	212206	UP	Karari	Allahabad

Super keys: {Roll}, {Roll, Name}, {Roll, Name, Pin}.....so on
Candidate Keys: {Roll}
Non-prime attributes: All attributes except Roll are non-prime as they are not part of any candidate keys.

- State, City & District are dependent on Pin.
- Pin is dependent on Roll that makes non-prime attributes (State , City & District) transitively dependent on super key (Roll).
- This violates the rule of 3NF

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Roll	Name	Pin	State	City	District
1	Rahul	535558	AP	Bobbili	Vizianagaram
2	Swetha	243505	UP	Shahi	Bareilly
3	Asmitha	600023	TN	Ayanavaram	Chennai
4	Tarun	212206	UP	Karari	Allahabad

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

Third Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Student Table

Roll	Name	Pin
1	Rahul	535558
2	Swetha	243505
3	Asmitha	600023
4	Tarun	212206

Student Pin code Table

Pin	State	City	District
535558	AP	Bobbili	Vizianagaram
243505	UP	Shahi	Bareilly
600023	TN	Ayanavaram	Chennai
212206	UP	Karari	Allahabad

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

- ✓ Boyce and Codd Normal Form is a higher version of the Third Normal form
- ✓ A Table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

- ✓ Boyce and Codd Normal Form is a higher version of the Third Normal form
- ✓ A Table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Let's assume there is a company where employees work in more than one department and company want to store the details of its employees

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Country	Dep	Dep_Id	No of Emp
1	Australia	Marketing	121	106
1	Australia	Testing	121	321
2	India	SEO	123	343
2	India	Development	123	213

Employee_Id → Country
Dep → {Dep_Id, No of Emp}

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Country	Dep	Dep_Id	No of Emp
1	Australia	Marketing	121	106
1	Australia	Testing	121	321
2	India	SEO	123	343
2	India	Development	123	213

Employee_Id → Country
Dep → {Dep_Id, No of Emp}

Candidate key: {Employee_Id , Dep}

The table is not in BCNF because neither **Dep** nor **Employee_Id** alone are keys.

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Country	Dep	Dep_Id	No of Emp
1	Australia	Marketing	121	106
1	Australia	Testing	121	321
2	India	SEO	123	343
2	India	Development	123	213

To convert the given table into BCNF, we decompose it into three tables:

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Country
1	Australia
1	Australia
2	India
2	India

Employee Country table

Employee_Id	Dep_Id
1	121
1	121
2	123
2	123

Employee Department table

Dep	Dep_Id	No of Emp
Marketing	121	106
Testing	121	321
SEO	123	343
Development	123	213

Employee Department Mapping table

BCNF

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Country
1	Australia
1	Australia
2	India
2	India

Employee Country table

Employee_Id	Dep_Id
1	121
1	121
2	123
2	123

Employee Department table

Dep	Dep_Id	No of Emp
Marketing	121	106
Testing	121	321
SEO	123	343
Development	123	213

Employee Department Mapping table

Candidate keys

Employee Country table: Employee_Id
Employee Department table : Dep
Employee Department Mapping table: {Employee_Id, Dep}

Now, this is in BCNF ✓

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To satisfy the Fourth Normal Form, it should satisfy the following two conditions:

- It should be in the Boyce-Codd Normal Form
- Table should not have any Multi-valued Dependency

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

To satisfy the Fourth Normal Form, it should satisfy the following two conditions:

- It should be in the Boyce-Codd Normal Form
- Table should not have any Multi-valued Dependency

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Certification	Hobby
24	Java	Music
24	Python	Dance
26	SQL	Story Writing
26	C#	Books

Employee with Employee_Id 24 is certified with Java and Python, and has two hobbies: Music and Dance.

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

Employee_Id	Certification	Hobby
24	Java	Music
24	Python	Dance
26	SQL	Story Writing
26	C#	Books

The two records for Employee with Employee_Id 24 will give rise to two more records, as shown below, because for one Employee, two hobbies exists, hence along with both the certification, these hobbies should be specified.

Employee_Id	Certification	Hobby
24	Java	Music
24	Python	Dance
24	Java	Dance
24	Python	Music

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

There is no relationship between the columns: Certification and Hobby. They are independent of each other. So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

Fourth Normal Form

- First Normal Form
- Second Normal Form
- Third Normal Form
- BCNF
- Fourth Normal Form

In order to satisfy the 4th normal form, we can decompose the table into 2 tables.

Employee_Id	Certification
24	Java
24	Python
26	SQL
26	C#

Certification Table

Employee_Id	Hobby
24	Music
24	Dance
26	Story Writing
26	Books

Hobbies Table

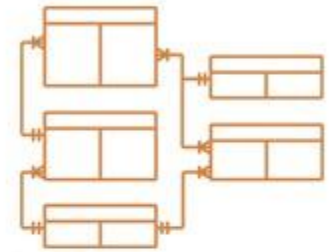
Agenda

ER Model

01

Entity Relationship Model

Entity relationship model defines the conceptual view of database. It works around real world entity and association among them.



Entities have attributes

Example: people have names and addresses

02

Entity

Entity is an object that exists and is distinguishable from other objects.

Example: Specific person, company, event, plant

03

Entity Set

Entity set is a set of entities of the same type that share the same properties.

Example: Set of all persons, companies, trees, holidays

ER Model

04

Attributes

Entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Example: instructor = (Id, name, street, city, salary), course= (course_id, title, credits)

Domain

Set of permitted values for each attribute

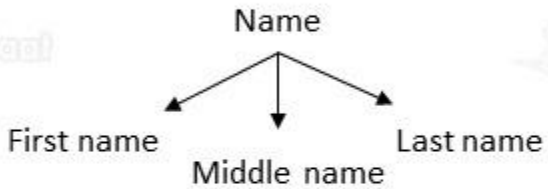
Attribute types

- **Simple** and **Composite attributes.**
- **Single-valued** and **Multivalued attributes**
- **Derived attributes**

Types of Attributes

Simple Attribute

Attribute that consist of a single atomic value.
Example: a student's phone number is an atomic value of 10 digits



Composite Attribute

Attribute value not atomic.
Example: example, a student's complete name may have first_name, middle_name and last_name.

Single Valued Attribute

Attribute that hold multiple values
Example 1: City **Example 2**: Customer id

Multi Valued Attribute

Attribute that hold multiple values
Example: A customer can have multiple phone numbers, email id's

Types of Attributes

Derived Attribute	An attribute that's value is derived from a stored attribute. **Example**: age, and it's value is derived from the stored attribute Date of Birth

.

Entity Sets

Instructor_Id	Name
1	A
2	B
3	C
4	D
5	E

Instructor

Student_Id	Name
11	P
12	Q
13	R
14	S
15	T
16	U

Student

Relationship Sets

Relationship is an association among several entities

Instructor_Id	Name
1	A
2	B
3	C
4	D
5	E

Instructor

Student_Id	Name
11	P
12	Q
13	R
14	S
15	T
16	U

Student

Degree of Relationship

The number of participating entities in a relationship defines the degree of the relationship.

Degree of Relationship

- ✓ Binary = degree 2
- ✓ Ternary = degree 3
- ✓ n-ary = degree

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

01 One-to-One

02 One-to-Many

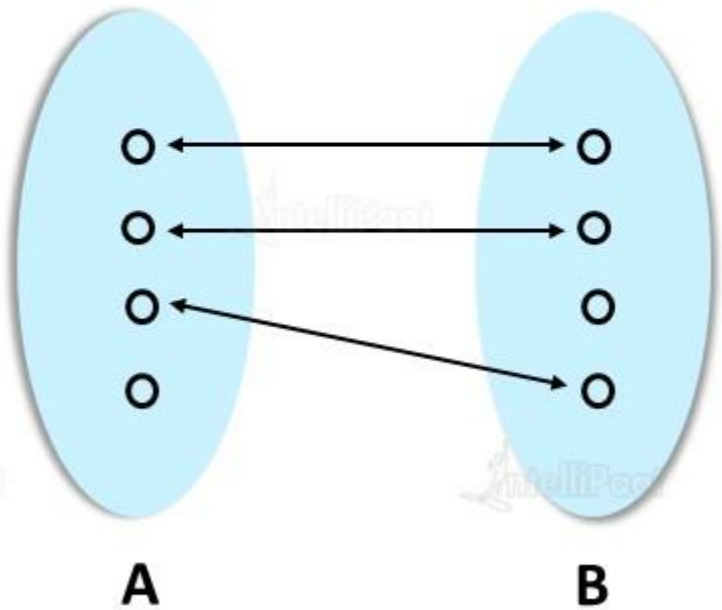
03 Many-to-one

04 Many-to-many

One-to-One

- One-to-One
- One-to-Many
- Many-to-one
- Many-to-many

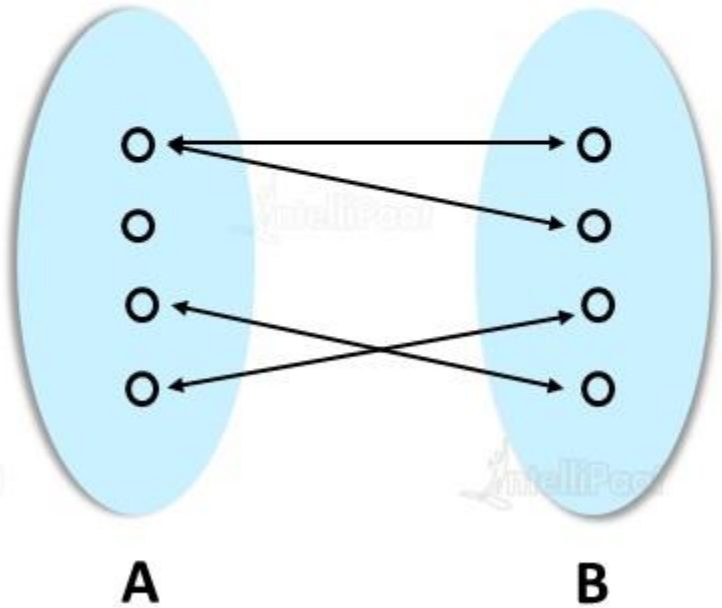
One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



One-to-Many

- One-to-One
- One-to-Many
- Many-to-one
- Many-to-many

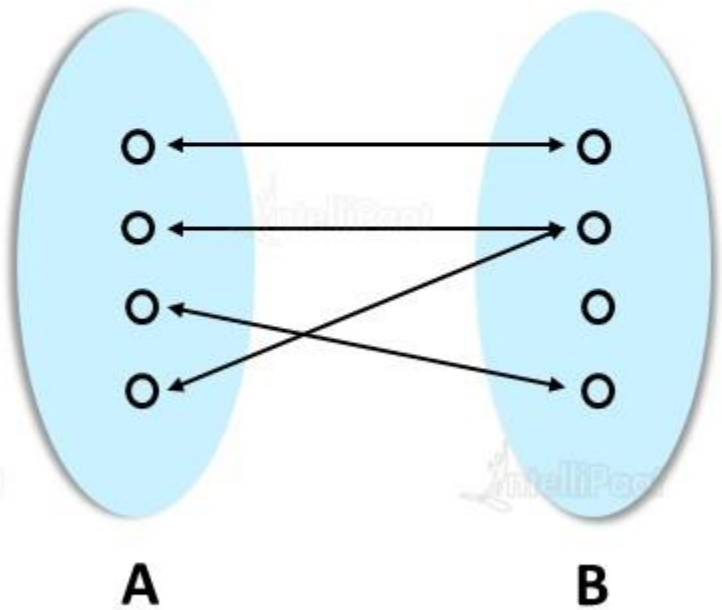
One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



Many-to-One

- One-to-One
- One-to-Many
- Many-to-one
- Many-to-many

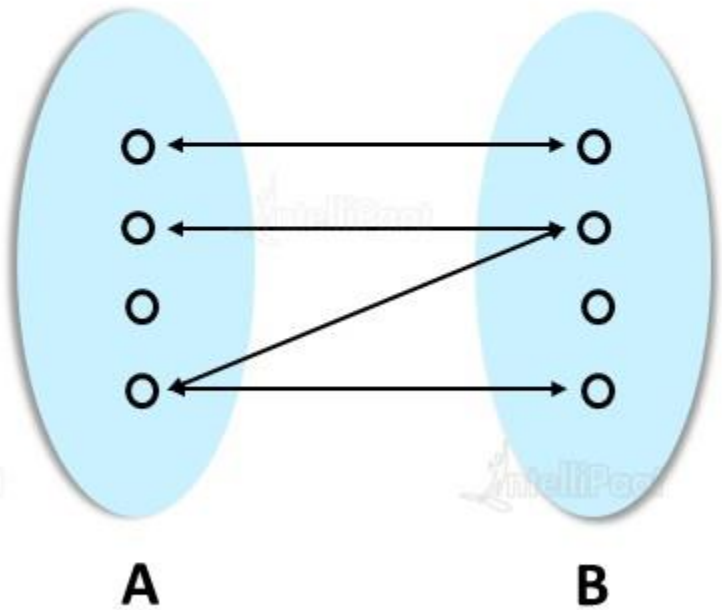
More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



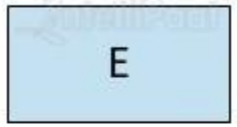
Many-to-One

- One-to-One
- One-to-Many
- Many-to-one
- Many-to-many

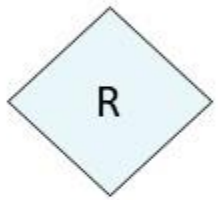
One entity from A can be associated with more than one entity from B and vice versa.



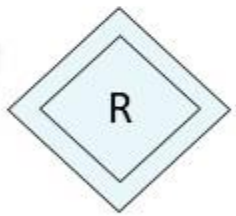
Symbols used in E-R Notation



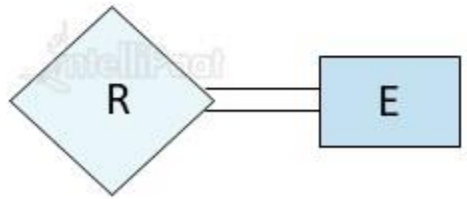
Entity Set



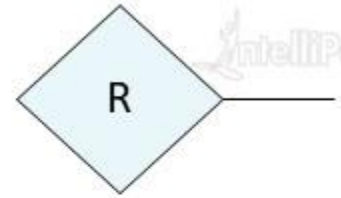
Relationship Set



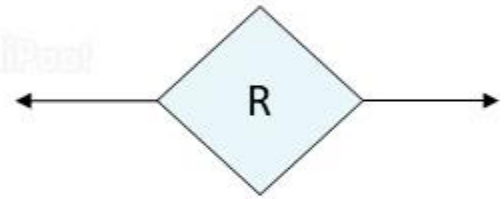
Identifying Relationship set for weak Entity set



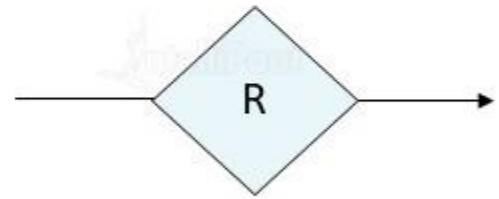
Total Participation of Entity set in relation



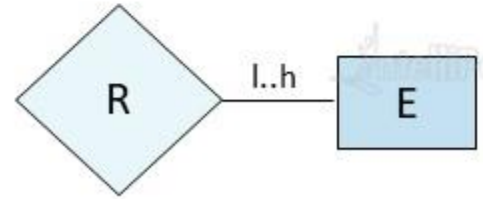
Many-to-Many Relationship



One-to-One Relationship



Many-to-One Relationship



Cardinality Limits

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor