Normalization and Normal Forms

What is normalization?

- Normalization is the **process of removing redundant data** from tables **to improve data integrity, scalability and storage efficiency**.
  - data integrity (completeness, accuracy and consistency of data)
  - scalability (ability of a system to continue to function well in a growing amount of data)
  - storage efficiency (ability to store and manage data that consumes the least amount of space)

- What we do in normalization?
  - Normalization generally involves **splitting an existing table into multiple (more than one) tables**, which can be **re-joined or linked** each time a query is issued (executed).

  - Decomposition is done with the help of FDs

How many normal forms are there?

- Normal forms:
  - 1NF (First normal form)
  - 2NF (Second normal form)
  - 3NF (Third normal form)
  - BCNF (Boyce–Codd normal form)
  - 4NF (Forth normal form)
  - 5NF (Fifth normal form)

As we move from 1NF to 5NF **number of tables** and **complexity increases** but **redundancy decreases**.

Normal forms
1NF (First Normal Form)

1NF (First Normal Form)

- Conditions for 1NF

  - Each **cell of a table should contain a single value (it should not contain any composite attribute or multi-valued attributes or their combinations.)**
  - Attribute Domain does not change.
  - There is a unique name for every Attribute/Column.
  - The order in which data is stored does not matter.

- A relation R is in first normal form (1NF) if and only if it **does not contain any composite attribute or multi-valued attributes or their combinations**.

OR

- A relation R is in first normal form (1NF) if and only if **all underlying domains contain atomic values only**.

OR

- A relation R is in first normal form (1NF) if and only if **there should not be multiple values in any row vs column entry.**

1NF (First Normal Form) [Example - Composite attribute]

Customer		
CID	**Name**	**Address**
C01	Raju	Jamnagar Road, Rajkot
C02	Mitesh	Nehru Road, Jamnagar
C03	Jay	C.G Road, Ahmedabad

- In customer relation **address is composite attribute** which is further divided into sub-attributes as “Road” and “City”.
- So customer relation is not in 1NF.

- **Problem**: It is **difficult to retrieve the list of customers living in ’Jamnagar’ city** from customer table.
- The reason is that **address attribute is composite attribute** which **contains road name as well as city name in single cell**.
- It is possible that **city name word is also there in road name**.
- In our example, ’Jamnagar’ word occurs in both records, in first record it is a part of road name and in second one it is the name of city.

1NF (First Normal Form) [Example - Composite attribute]

Customer

CID	Name	Address
C01	Raju	Jamnagar Road, Rajkot
C02	Mitesh	Nehru Road, Jamnagar
C03	Jay	C.G Road, Ahmedabad

Customer

CID	Name	Road	City
C01	Raju	Jamnagar Road	Rajkot
C02	Mitesh	Nehru Road	Jamnagar
C03	Jay	C.G Road	Ahmedabad

- **Solution**: **Divide composite attributes** into **number of sub-attributes** and insert value in proper sub-attribute.

Exercise Convert below relation into 1NF (First Normal Form)

Person

PID	Full_Name	City
P01	Raju Maheshbhai Patel	Rajkot

1NF (First Normal Form) [Example - Multivalued attribute]

Student

Rno	Name	Failed in Subjects
101	Raju	DS, DBMs
102	Mitesh	DBMS, DS
103	Jay	DS, DBMS, DE
104	Jeet	DBMS, DE, DS
105	Harsh	DE, DBMS, DS
106	Neel	DE, DBMS

- In student relation **Failed in Subjects attribute is a multi-valued attribute** which can store more than one values.
- So above relation is not in 1NF.

- **Problem**: It is difficult to retrieve the **list of students failed in 'DBMS' as well as 'DS' but not in other subjects** from student table.
- The reason is that "Failed in Subjects" attribute is multi-valued attribute so it contains more than one value.

1NF (First Normal Form) [Example - Multivalued attribute]

Student

Rno	Name	FailedinSubjects
101	Raju	DS, DBMs
102	Mitesh	DBMS, DS
103	Jay	DS, DBMS, DE
104	Jeet	DBMS, DE, DS
105	Harsh	DE, DBMS, DS
106	Neel	DE, DBMS

Student

Rno	Name
101	Raju
102	Mitesh
103	Jay
104	Jeet
105	Harsh
106	Neel

Result

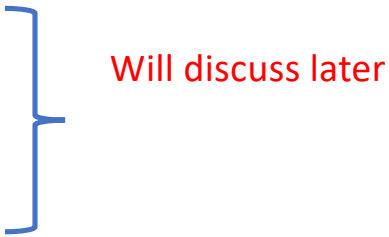
Rno	Subject
101	DS
101	DBMS
102	DBMS
102	DS
103	DS
…	…

- **Solution**: Split the table into two tables in such as way that
  - the **first table contains all attributes except multi-valued attribute** with same primary key
  - **second table contains multi-valued attribute**
  - **insert the primary key of first table in the second table as a foreign key**.

FDs and Types of FDs

- Deals with one-to-one relationship among attribute.
- FD must be defined on schema not on instance.
- FD must be non-trivial or completely non-trivial.

- Types of FDs:
  - Trivial FD
  - Non-Trivial FD
  - Completely Non-Trivial
  - Transitive FD
  - Full Functional Dependency
  - Partial FD



Types of Functional Dependency (FD)

- **Trivial FD:** RHS is subset of LHS
  - X → Y is trivial FD if **Y is a subset of X**
  - Eg. {Roll_No, Dept_Name, Semester} → Roll_No, Semester OR ABC →AB

- **Nontrivial FD:** One of RHS attribute is not subset LHS.
  - X → Y is nontrivial FD if **Y is not a subset of X**
  - Eg. {Roll_No, Dept_Name, Semester} → Stud_Name, Semester OR ABC →AD

- **Completely Non-Trivial FD**: None of RHS attribute is subset of LHS.
  - X → Y is Completely nontrivial FD if **all subset of Y is not a subset of X**
  - Eg. {Roll_No, Dept_Name, Semester} → Stud_Name, SPI OR ABC →DE

Types of Functional Dependency (FD)

- **Full Functional Dependency**
  - In a relation, the attribute B is fully functional dependent on A if **B is functionally dependent on A, but not on any proper subset of A**.

  - **Example:** {Roll_No, Semester, Department_Name} → SPI
  - We **need all three {Roll_No, Semester, Department_Name} to find SPI**.

Types of Functional Dependency (FD)

- **Partial Functional Dependency**
  - Partial Dependency occurs when a nonprime attribute is functionally dependent on part of a candidate key (prime attribute).

Example: R(A, B, C, D), Assume Key is {AB}, then AB → C, and AB → D are allowed whereas, A → C, A → D or B → C, B → D are not allowed.

Prime attribute → Non-prime attribute

Types of Functional Dependency (FD)

- **Transitive Functional Dependency**
  - In a relation, if attribute(s) **A → B and B → C, then A → C (means C is transitively depends on A** via B).

Sub_Fac		
Subject	**Faculty**	**Age**
DS	Shah	35
DBMS	Patel	32
DF	Shah	35

  - **Example:** Subject → Faculty & Faculty → Age then Subject → Age

  - Therefore as per the rule of transitive dependency: Subject → Age should hold, that makes sense because if we know the subject name we can know the faculty's age.

Normal forms
2NF (Second Normal Form)

2NF (Second Normal Form)

- Conditions for 2NF

It is **in 1NF** and each **table should not contain partial dependency**.

- A relation R is in second normal form (2NF) iff:
  1. It is in **1NF** and
  2. **Non-prime attribute is NOT *partially dependent* on the key**

OR

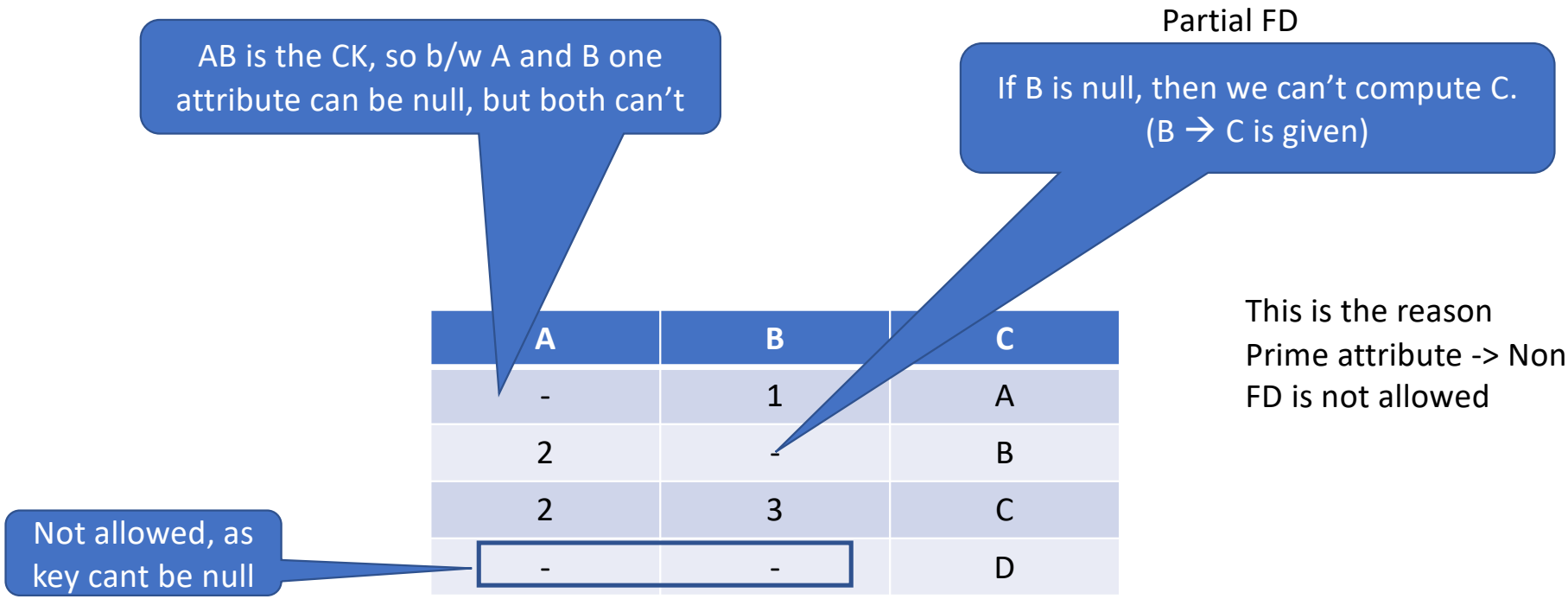
Not allowed

Prime attribute ~~→~~ non-prime attribute

- A relation R is in second normal form (2NF) iff:
  1. It is in **1NF** and
  2. **Every non-prime attribute is fully dependent on the key: fully functionally dependent**

Example: R(A, B, C, D), Assume Key is {AB}, {A, B} = prime attributes & {C, D} = non-prime attributes;
then AB → C, and AB → D are allowed
whereas, A → C, A → D or B → C, B → D are not allowed.

What is the problem with partial dependency?



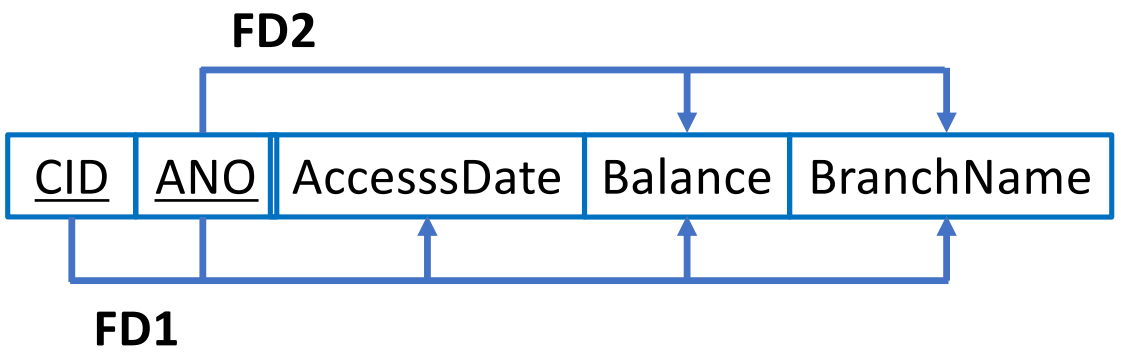
A	B	C
-	1	A
2	-	B
2	3	C
-	-	D

This is the reason
Prime attribute -> Non prime attribute
FD is not allowed

2NF (Second Normal Form) [Example]

Customer

CID	ANO	AccessDate	Balance	BranchName
C01	A01	01-01-2017	50000	Rajkot
C02	A01	01-03-2017	50000	Rajkot
C01	A02	01-05-2017	25000	Surat
C03	A02	01-07-2017	25000	Surat

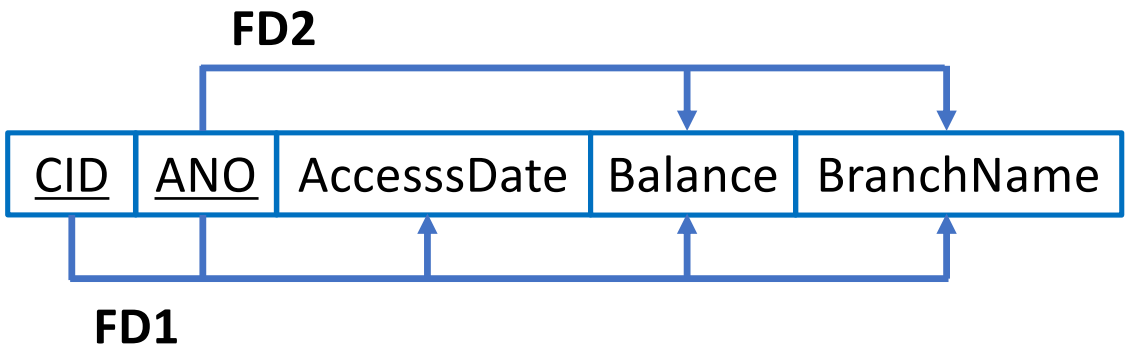


- **FD1**: {CID, ANO} → {AccesssDate, Balance, BranchName}
- **FD2**: ANO → {Balance, BranchName}
- **Balance and BranchName are partial dependent on key (CID + ANO)**. So customer relation is not in 2NF.

2NF (Second Normal Form) [Example]

Customer

CID	ANO	AccessDate	Balance	BranchName
C01	A01	01-01-2017	50000	Rajkot
C02	A01	01-03-2017	50000	Rajkot
C01	A02	01-05-2017	25000	Surat
C03	A02	01-07-2017	25000	Surat



- **Problem:** For example, in case of a joint account multiple (more than one) customers have common (one) accounts.
- If an account **'A01' is operated jointly by two customers** says **'C01' and 'C02'** then **data** values for attributes **Balance and BranchName** will be **duplicated in two different tuples** of customers 'C01' and 'C02'.

2NF (Second Normal Form) [Example]

Customer

CID	ANO	AccessDate	Balance	BranchName
C01	A01	01-01-2017	50000	Rajkot
C02	A01	01-03-2017	50000	Rajkot
C01	A02	01-05-2017	25000	Surat
C03	A02	01-07-2017	25000	Surat

Table-1

ANO	Balance	BranchName
A01	50000	Rajkot
A02	25000	Surat

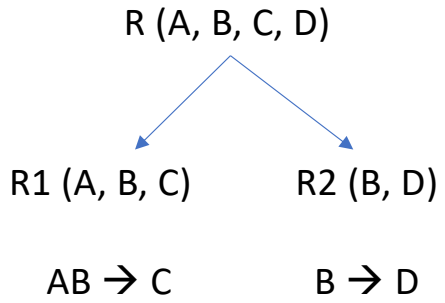
Table-2

CID	ANO	AccessDate
C01	A01	01-01-2017
C02	A01	01-03-2017
C01	A02	01-05-2017
C03	A02	01-07-2017

- **Solution:** **Decompose relation** in such a way that **resultant relations do not have any partial FD**.
  - **Remove partial dependent attributes** from the relation that violets 2NF.
  - **Place them in separate relation** along with the **prime attribute on which they are fully dependent**.
  - The **key of new relation** will be the **attribute on which it is fully dependent**.
  - **Keep other attributes same** as in that table with the **same primary key**.

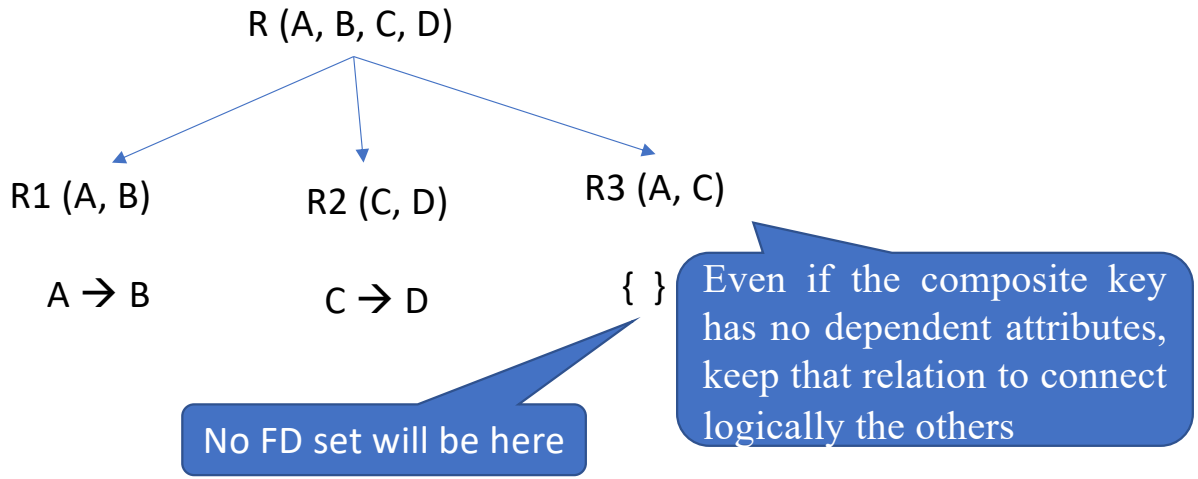
Q. 1 Given that R(A, B, C, D) & FD {AB → C, B → D}. Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {AB}.
B → D is a partial dependency.



Q. 2 Given that R(A, B, C, D) & FD {A → B, C → D}. Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {AC}.
A → B is a partial dependency.
C → D is a partial dependency.



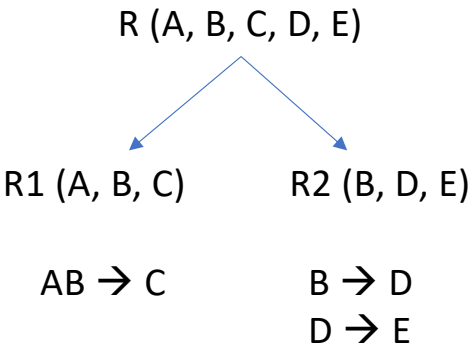
Q. 3 Given that R(A, B, C, D, E) & FD {AB → C, B → D, D → E}. Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {AB}.

B → D is a partial dependency.

B^+ = {B, D, E}

Take closure of LHS part of Partial Dependency, and keep all of them in separate relation.



Q. 4 Given that R(A, B, C, D, E) & FD {A → B, B → E, C → D}. Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {AC}.

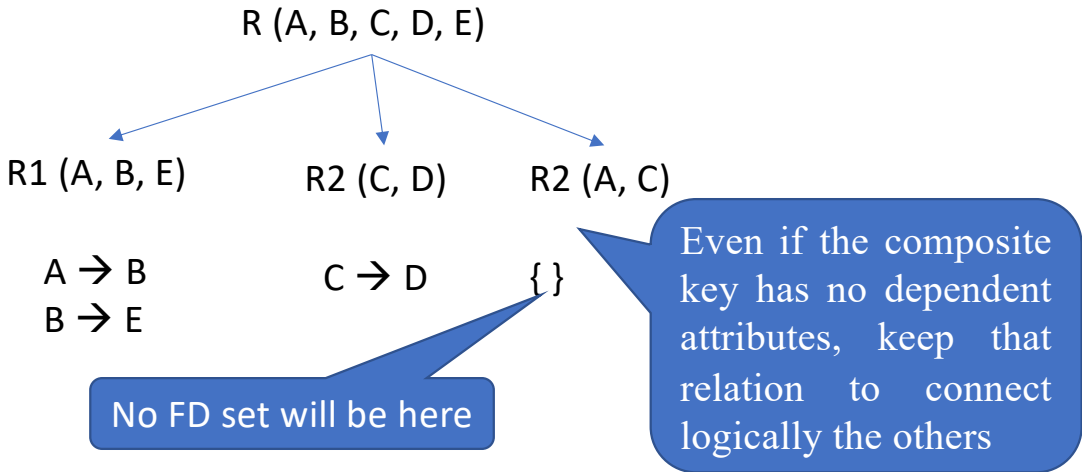
A → B is a partial dependency.

C → D is a partial dependency.

Take closure of LHS part of Partial Dependency

A^+ = {A, B, E}

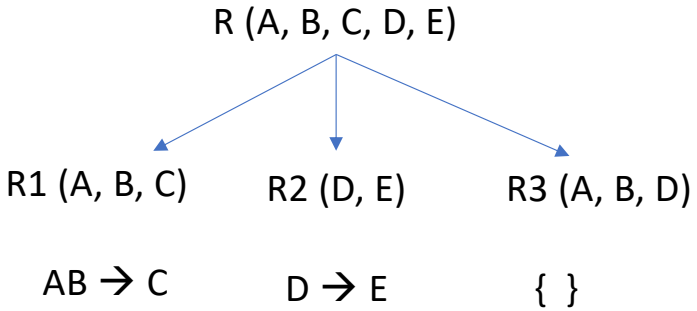
C^+ = {C, D}



Q. 5 Given that R(A, B, C, D, E) & FD {AB → C, D → E}. Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {ABD}.
AB → C is a partial dependency.
D → E is a partial dependency.

AB^+ = {A, B, C}
D^+ = {D, E}



Q. R(A, B, C, D, E, F)

A → FC

C → D

B → E; Check whether it is in 2 NF or not, if not, then convert it into 2NF.

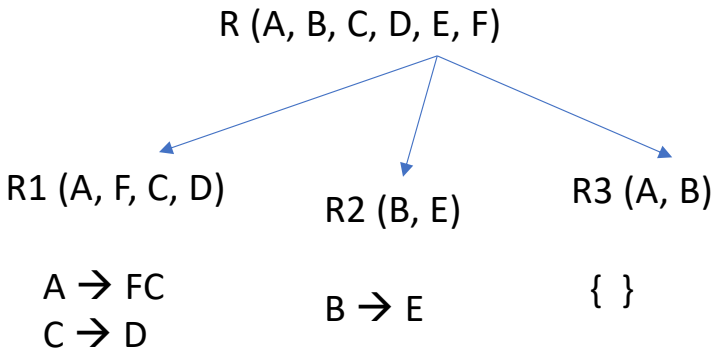
Key = {AB}.

A → FC is a partial dependency.

B → E is a partial dependency.

A^+ = {A, F, C, D}

B^+ = {B, E}



Q. R(A, B, C, D, E, F, G, H, I, J)

AB → C

AD → GH

BD → EF

A → I

H → J

Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {ABD}.

AB → C is a partial dependency.

AD → GH is a partial dependency.

BD → EF is a partial dependency.

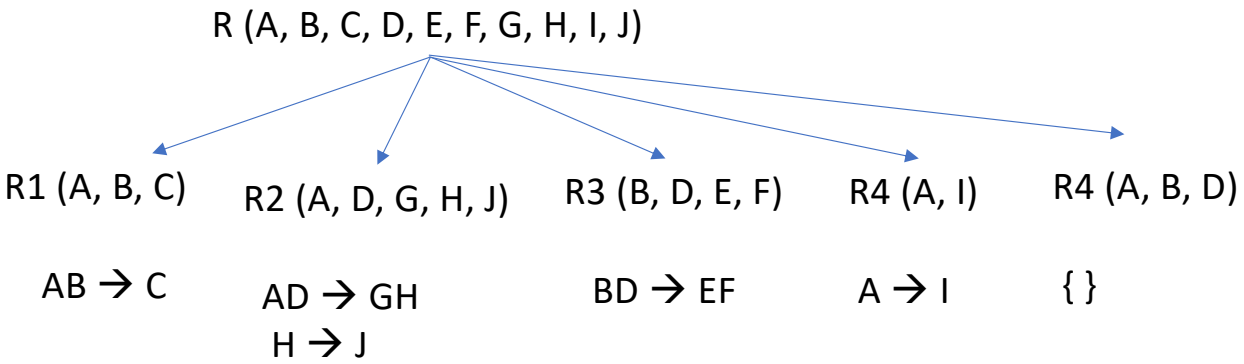
A → I is a partial dependency.

AB^+ = {A, B, C}

AD^+ = {A, D, G, H, J}

BD^+ = {B, D, E, F}

A^+ = {A, I}



Q. R(A, B, C, D, E, F, G, H, I, J)

AB → C

A → DE

B → F

F → GH

D → IJ

Check whether it is in 2 NF or not, if not, then convert it into 2NF.

Key = {AB}.

A → DE is a partial dependency.

B → F is a partial dependency.

A^+ = {A, D, E, I, J}

B^+ = {B, F, G, H}