Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

1. X → Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

For example:

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

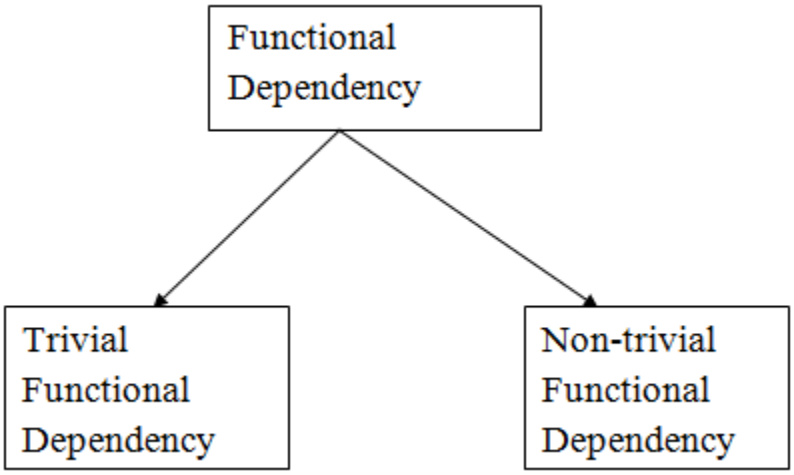
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

1. Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency



1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too.

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

1. ID → Name,
2. Name → DOB

Normalization

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfy desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

What is Normalization?

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

Why do we need Normalization?

The main reason for normalizing the relations is removing these anomalies. Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows. Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

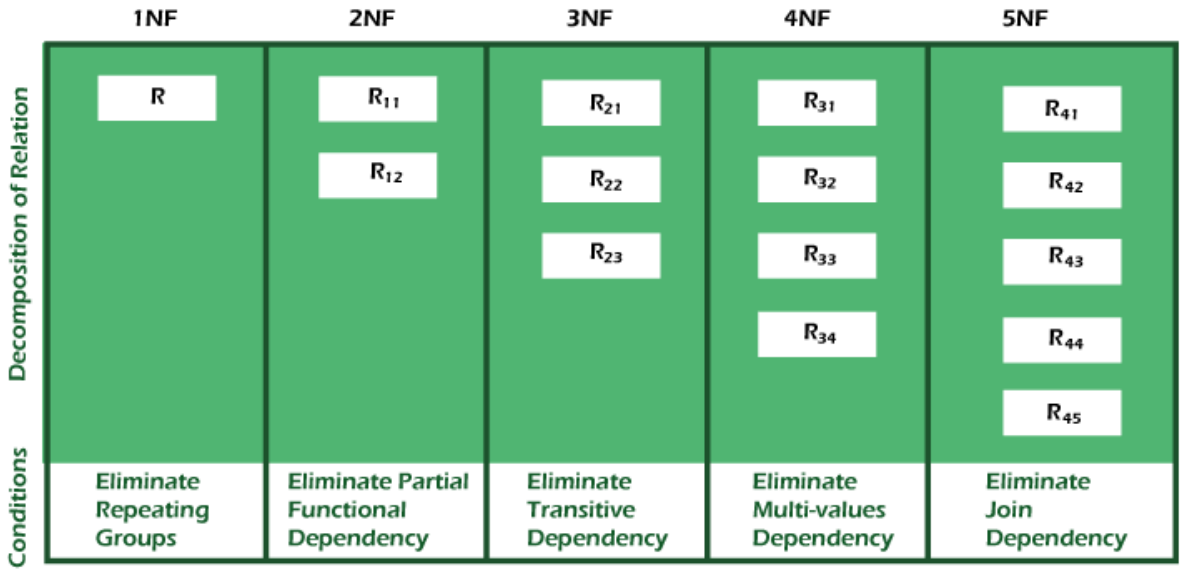
Data modification anomalies can be categorized into three types:

- **Insertion Anomaly:** Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- **Deletion Anomaly:** The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- **Updatation Anomaly:** The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

Types of Normal Forms:

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

Following are the various types of Normal forms:



Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.

- Much more flexible database design.
- Enforces the concept of relational integrity.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

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

EMPLOYEE table:

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

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

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

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.

2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

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

Super key in the table above:

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

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

Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

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

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

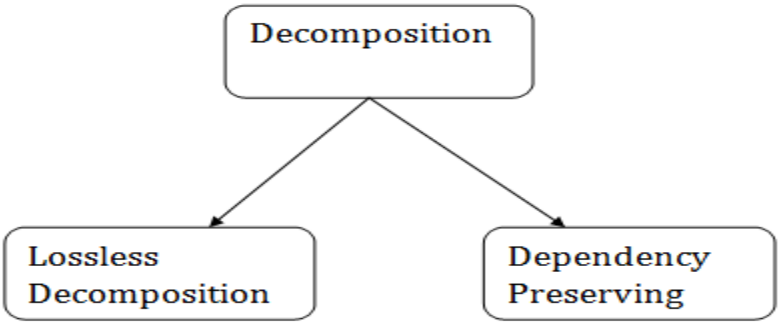
Inclusion Dependency

- Multivalued dependency and join dependency can be used to guide database design although they both are less common than functional dependencies.
- Inclusion dependencies are quite common. They typically show little influence on designing of the database.
- The inclusion dependency is a statement in which some columns of a relation are contained in other columns.
- The example of inclusion dependency is a foreign key. In one relation, the referring relation is contained in the primary key column(s) of the referenced relation.
- Suppose we have two relations R and S which was obtained by translating two entity sets such that every R entity is also an S entity.
- Inclusion dependency would be happen if projecting R on its key attributes yields a relation that is contained in the relation obtained by projecting S on its key attributes.
- In inclusion dependency, we should not split groups of attributes that participate in an inclusion dependency.
- In practice, most inclusion dependencies are key-based that is involved only keys.

Relational Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Types of Decomposition



Lossless Decomposition

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

Example:

EMPLOYEE_DEPARTMENT table:

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

The above relation is decomposed into two relations EMPLOYEE and DEPARTMENT

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY
22	Denim	28	Mumbai
33	Alina	25	Delhi
46	Stephan	30	Bangalore
52	Katherine	36	Mumbai
60	Jack	40	Noida

DEPARTMENT table

DEPT_ID	EMP_ID	DEPT_NAME
827	22	Sales
438	33	Marketing
869	46	Finance
575	52	Production
678	60	Testing

Now, when these two relations are joined on the common column "EMP_ID", then the resultant relation will look like:

Employee ⋈ Department

EMP_ID	EMP_NAME	EMP_AGE	EMP_CITY	DEPT_ID	DEPT_NAME
22	Denim	28	Mumbai	827	Sales
33	Alina	25	Delhi	438	Marketing
46	Stephan	30	Bangalore	869	Finance
52	Katherine	36	Mumbai	575	Production
60	Jack	40	Noida	678	Testing

Hence, the decomposition is Lossless join decomposition.

Dependency Preserving

- It is an important constraint of the database.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.
- If a relation R is decomposed into relation R1 and R2, then the dependencies of R either must be a part of R1 or R2 or must be derivable from the combination of functional dependencies of R1 and R2.
- For example, suppose there is a relation R (A, B, C, D) with functional dependency set (A->BC). The relational R is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of relation R1(ABC).

Multivalued Dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:

1. BIKE_MODEL → → MANUF_YEAR
2. BIKE_MODEL → → COLOR

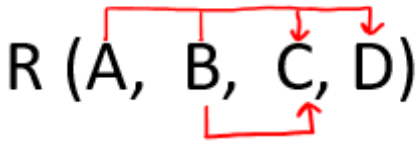
This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

Join Dependency

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over C is equal to relation R, then we can say that a join dependency (JD) exists.
- Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.
- A JD ⋈ {R1, R2,..., Rn} is said to hold over a relation R if R1, R2,....., Rn is a lossless-join decomposition.
- The *(A, B, C, D), (C, D) will be a JD of R if the join of join's attribute is equal to the relation R.
- Here, *(R1, R2, R3) is used to indicate that relation R1, R2, R3 and so on are a JD of R.

1. Given a relation R(A, B, C, D) and Functional Dependency set FD = { AB → CD, B → C }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes AB is not determined by any of the given FD, hence AB will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have W compulsory attribute.

Let us calculate the closure of AB

AB + = ABCD (from the method we studied earlier)

Since the closure of AB contains all the attributes of R, hence **AB is Candidate Key**

From the definition of Candidate Key(**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have AB as an integral part, and we have proved that AB is Candidate Key, Therefore, any superset of AB will be Super Key but not Candidate key.

Hence there will be only **one candidate key AB**

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key

Since R has 4 attributes: - A, B, C, D, and Candidate Key is AB, Therefore, prime attributes (part of candidate key) are A and B while a non-prime attribute are C and D

a) FD: **AB → CD** satisfies the definition of 2NF, that non-prime attribute(C and D) are fully dependent on candidate key AB

b) FD: **B → C** does not satisfy the definition of 2NF, as a non-prime attribute(C) is partially dependent on candidate key AB(i.e. key should not be broken at any cost)

As FD B → C, the above table R(A, B, C, D) is not in 2NF

Convert the table R(A, B, C, D) in 2NF:

Since **FD: B → C,** our table was not in 2NF, let's decompose the table

R1(B, C)

Since the key is AB, and from FD AB → CD, we can create R2(A, B, C, D) but this will again have a problem of partial dependency B → C, hence R2(A, B, D).

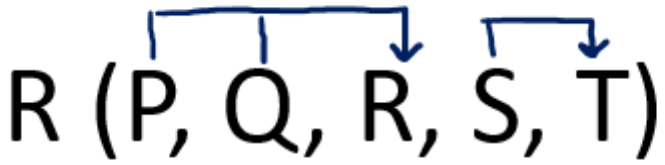
Finally, the decomposed table which is in 2NF

a) R1(B, C)

b) R2(A, B, D)

2. Given a relation R(P, Q, R, S, T) and Functional Dependency set FD = { PQ → R, S → T }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes PQS is not determined by any of the given FD, hence PQS will be the integral part of the Candidate key, i.e., no matter what will be the candidate key, and how many will be the candidate key, but all will have PQS compulsory attribute.

Let us calculate the closure of PQS

PQS + = PQSRT (from the method we studied earlier)

Since the closure of PQS contains all the attributes of R, hence **PQS is Candidate Key**

From the definition of Candidate Key (**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have PQS as an integral part, and we have proved that PQS is Candidate Key. Therefore, any superset of PQS will be Super Key but not Candidate key.

Hence there will be only **one candidate key PQS**

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key.

Since R has 5 attributes: - P, Q, R, S, T and Candidate Key is PQS, Therefore, prime attributes (part of candidate key) are P, Q, and S while a non-prime attribute is R and T

a) FD: **PQ** → **R** does not satisfy the definition of 2NF, that non-prime attribute(R) is partially dependent on part of candidate key PQS.

b) FD: **S** → **T** does not satisfy the definition of 2NF, as a non-prime attribute(T) is partially dependent on candidate key PQS (i.e., key should not be broken at any cost).

Hence, FD PQ → R and S → T, the above table R(P, Q, R, S, T) is not in 2NF

Convert the table R(P, Q, R, S, T) in 2NF:

Since due to FD: PQ → R and S → T, our table was not in 2NF, let's decompose the table

R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)

R2(S, T) (Now in table R2 FD: S → T is Full F D, hence R2 is in 2NF)

And create one table for the key, since the key is PQS.

R3(P, Q, S)

Finally, the decomposed tables which is in 2NF are:

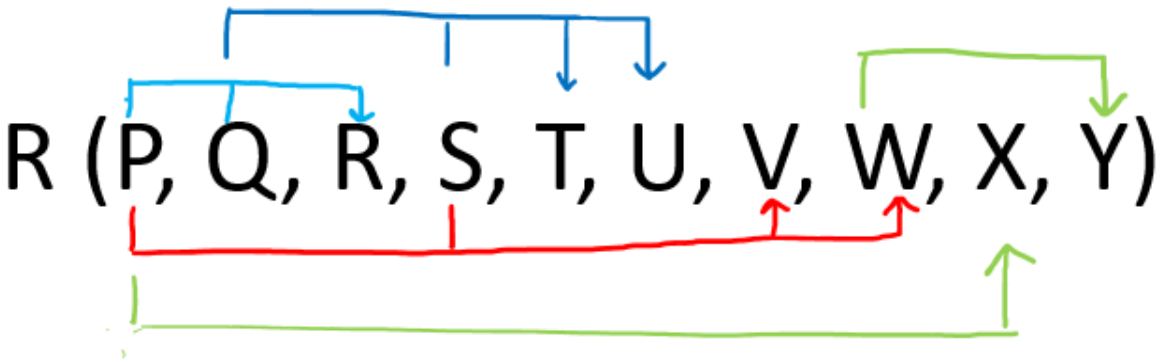
a) R1(P, Q, R)

b) R2(S, T)

c) R3(P, Q, S)

3. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes PQS is not determined by any of the given FD, hence PQS will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have PQS compulsory attribute.

Let us calculate the closure of PQS

PQS + = P Q S R T U V W X Y (from the closure method we studied earlier)

Since the closure of PQS contains all the attributes of R, hence **PQS is Candidate Key**

From the definition of Candidate Key(**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have PQS as an integral part, and we have proved that PQS is Candidate Key, Therefore, any superset of PQS will be Super Key but not a Candidate key.

Hence there will be only **one candidate key PQS**

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key

Since R has 10 attributes: - P, Q, R, S, T, U, V, W, X, Y, and Candidate Key is PQS calculated using FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }. Therefore, prime attribute(part of candidate key) are P, Q, and S while non-prime attribute are R, T, U, V, W, X and Y

a. FD: **PQ → R** does not satisfy the definition of 2NF, that non-prime attribute(R) is partially dependent on part of candidate key PQS

b. FD: **PS → VW** does not satisfy the definition of 2NF, that non-prime attribute(VW) is partially dependent on part of candidate key PQS
c. FD: **QS → TU** does not satisfy the definition of 2NF, that non-prime attribute(TU) is partially dependent on part of candidate key PQS
d. FD: **P → X** does not satisfy the definition of 2NF, that non-prime attribute(X) are partially dependent on part of candidate key PQS
e. FD: **W → Y** does **not violate** the definition of 2NF, as the non-prime attribute(Y) is dependent on the non-prime attribute(W), which is not related to the definition of 2NF.

Hence because of FD: PQ → R, PS → VW, QS → TU, P → X the above table R(P, Q, R, S, T, U, V, W, X, Y) is not in 2NF

Convert the table R(P, Q, R, S, T, U, V, W, X, Y) in 2NF:

Since due to FD: PQ → R, PS → VW, QS → TU, P → X our table was not in 2NF, let's decompose the table

R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)

R2(P, S, V, W) (Now in table R2 FD: PS → VW is Full F D, hence R2 is in 2NF)

R3(Q, S, T, U) (Now in table R3 FD: QS → TU is Full F D, hence R3 is in 2NF)

R4(P, X) (Now in table R4 FD : P → X is Full F D, hence R4 is in 2NF)

R5(W, Y) (Now in table R5 FD: W → Y is Full F D, hence R2 is in 2NF)

And create one table for the key, since the key is PQS.

R6(P, Q, S)

Finally, the decomposed tables which is in 2NF are:

R1(P, Q, R)

R2(P, S, V, W)

R3(Q, S, T, U)

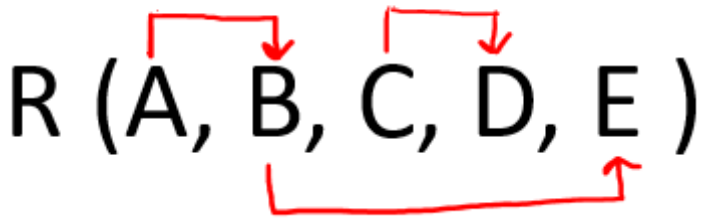
R4(P, X)

R5(W, Y)

R6(P, Q, S)

4. Given a relation R(A, B, C, D, E) and Functional Dependency set FD = { A → B, B → E, C → D}, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes AC is not determined by any of the given FD, hence AC will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have W compulsory attribute.

Let us calculate the closure of AC

AC + = ACBED(from the closure method we studied earlier)

Since the closure of AC contains all the attributes of R, hence **AC is Candidate Key**

From the definition of Candidate Key(**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have AC as an integral part, and we have proved that AC is Candidate Key, Therefore, any superset of AC will be Super Key but not Candidate key.

Hence there will be only **one candidate key AC**

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key

Since R has 5 attributes: - A, B, C, D, E and Candidate Key is AC, Therefore, prime attribute (part of candidate key) are A and C while the non-prime attribute are B D and E

a. FD: **A** → **B** does not satisfy the definition of 2NF, as a non-prime attribute(B) is partially dependent on candidate key AC (i.e., key should not be broken at any cost).

b. FD: **B** → **E** does **not violate** the definition of 2NF, as a non-prime attribute(E) is dependent on the non-prime attribute(B), which is not related to the definition of 2NF.

c. FD: **C → D** does not satisfy the definition of 2NF, as a non-prime attribute(D) is partially dependent on candidate key AC (i.e., key should not be broken at any cost)

Hence because of FD A → B and C → D, the above table R(A, B, C, D, E) is not in 2NF

Convert the table R(A, B, C, D, E) in 2NF:

Since due to FD: A →B and C → D our table was not in 2NF, let's decompose the table

R1(A, B, E) (from FD: A → B and B → E and both are violating 2 NF definition)

R2(C, D) (Now in table R2 FD: C → D is Full F D, hence R2 is in 2NF)

And create one table for candidate key AC

R3 (A, C)

Finally, the decomposed tables which are in 2NF:

a. **R1(A, B, E)**
b. **R2(C, D)**
c. **R3(A, C)**

Procedure: To verify that given relational schema R is in 2NF or NOT, If NOT then Convert it to 2NF:

STEP 1: Calculate the Candidate Key of given R by using an arrow diagram on R.

STEP 2: Verify each FD with **Definition of 2NF** (No non-prime attribute should be partially dependent on Candidate Key)

STEP 3: Make a set of FD which do not satisfy 2NF, i.e. all those FD which are partial.

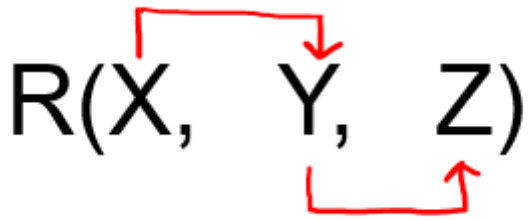
STEP 4: Convert the table R in 2NF by decomposing R such that each decomposition based on FD should satisfy the definition of 2NF:

STEP 5: Once the decomposition based on FD is completed, create a separate table of attributes in the Candidate key.

STEP 6: All the decomposed R obtained from STEP 4 and STEP 5 forms the required decomposition where each decomposition is in 2NF.

Question 1: Given a relation R(X, Y, Z) and Functional Dependency set FD = { X → Y and Y → Z }, determine whether the given R is in 3NF? If not convert it into 3 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attribute X is not determined by any of the given FD, hence X will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have X compulsory attribute.

Let us calculate the closure of X

X + = XYZ (from the closure method we studied earlier)

Since the closure of X contains all the attributes of R, hence **X is Candidate Key**

From the definition of Candidate Key (**Candidate Key is a Super Key whose no proper subset is a Super key**)

Since all key will have X as an integral part, and we have proved that X is Candidate Key, Therefore, any superset of X will be Super Key but not the Candidate key.

Hence there will be only **one candidate key X**

Definition of 3NF: A relational schema R is said to be in 3NF, First, it should be in 2NF and, no non-prime attribute should be transitively dependent on the Key of the table.

If X → Y and Y → Z exist then X → Z also exists which is a transitive dependency, and it should not hold.

Since R has 3 attributes: - X, Y, Z, and Candidate Key is X, Therefore, prime attribute (part of candidate key) is X while a non-prime attribute are Y and Z

Given FD are X → Y and Y → Z

So, we can write X → Z (which is a transitive dependency)

In above **FD X → Z**, a non-prime attribute(Z) is transitively depending on the key of the table(X) hence as per the definition of 3NF it is not in 3 NF, **because no non-prime attribute should be transitively dependent on the key of the table.**

Now check the above table is in 2 NF.

a. FD: X → Y is in 2NF (as Key is not breaking and its Fully functional dependent)
b. FD: Y → Z is also in 2NF(as it does not violate the definition of 2NF)

Hence above table R(X, Y, Z) is in 2NF but not in 3NF.

We can also prove the same from Definition 2: First, it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then

a. Either X is Super Key
b. Or Y is a prime attribute.

Since we have just proved that above table R is in 2 NF. Let's check it for 3NF using definition 2.

a. FD: X → Y is in 3NF (as X is a super Key)
b. FD: Y → Z is not in 3NF (as neither Y is Key nor Z is a prime attribute)

Hence because of Y → Z using definition 2 of 3NF, we can say that above table R is not in 3NF.

Convert the table R(X, Y, Z) into 3NF:

Since due to FD: Y → Z, our table was not in 3NF, let's decompose the table

FD: Y → Z was creating issue, hence one table R1(Y, Z)

Create one Table for key X, R2(X, Y), since X → Y

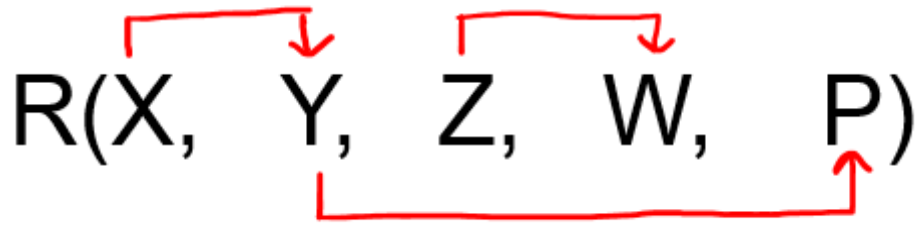
Hence decomposed tables which are in 3NF are:

R1(X, Y)

R2(Y, Z)

Question 2: Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes XZ is not determined by any of the given FD, hence XZ will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have XZ compulsory attribute.

Let us calculate the closure of XZ

XZ + = XZYPW (from the closure method that we studied earlier)

Since the closure of XZ contains all the attributes of R, hence **XZ is Candidate Key**

From the definition of Candidate Key (**Candidate Key is a Super Key whose no proper subset is a Super key**).

Since all key will have XZ as an integral part, and we have proved that XZ is Candidate Key, Therefore, any superset of XZ will be Super Key but not the Candidate key.

Hence there will be only **one candidate key XZ**

Definition of 3NF: First it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then

a. Either X is Super Key
b. Or Y is a prime attribute.

Since R has 5 attributes: - X, Y, Z, W, P and Candidate Key is XZ, Therefore, prime attribute (part of candidate key) are X and Z while a non-prime attribute are Y, W, and P

Given FD are X → Y, Y → P, and Z → W and Super Key / Candidate Key is XZ

a. FD: **X** → **Y** does not satisfy the definition of 3NF, that neither X is Super Key nor Y is a prime attribute.

b. FD: **Y** → **P** does not satisfy the definition of 3NF, that neither Y is Super Key nor P is a prime attribute.

c. FD: **Z** → **W** satisfies the definition of 3NF, that neither Z is Super Key nor W is a prime attribute.

Convert the table R(X, Y, Z, W, P) into 3NF:

Since all the FD = { X → Y, Y → P, and Z → W} were not in 3NF, let us convert R in 3NF

R1(X, Y) {Using FD X → Y}

R2(Y, P) {Using FD Y → P}

R3(Z, W) {Using FD Z → W}

And create one table for Candidate Key XZ

R4(X, Z) { Using Candidate Key XZ }

All the decomposed tables R1, R2, R3, and R4 are in 2NF(as there is no partial dependency) as well as in 3NF.

Hence decomposed tables are:

R1(X, Y), R2(Y, P), R3(Z, W), and **R4(X, Z)**

Question 3: Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, P → ST, Q → U, U → VW, and S → XY}, determine whether the given R is in 3NF? If not convert it into 3 NF.

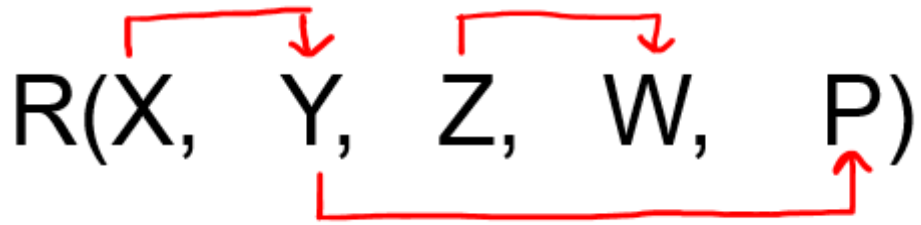
Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key

R1(X, Y)

R2(Y, Z)

Question 2: Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key.



From above arrow diagram on R, we can see that an attributes XZ is not determined by any of the given FD, hence XZ will be the integral part of the Candidate key, i.e. no matter what will be the candidate key, and how many will be the candidate key, but all will have XZ compulsory attribute.

Let us calculate the closure of XZ

XZ + = XZYPW (from the closure method that we studied earlier)

Since the closure of XZ contains all the attributes of R, hence **XZ is Candidate Key**

From the definition of Candidate Key (**Candidate Key is a Super Key whose no proper subset is a Super key**).

Since all key will have XZ as an integral part, and we have proved that XZ is Candidate Key, Therefore, any superset of XZ will be Super Key but not the Candidate key.

Hence there will be only **one candidate key XZ**

Definition of 3NF: First it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then

a. Either X is Super Key
b. Or Y is a prime attribute.

Since R has 5 attributes: - X, Y, Z, W, P and Candidate Key is XZ, Therefore, prime attribute (part of candidate key) are X and Z while a non-prime attribute are Y, W, and P

Given FD are X → Y, Y → P, and Z → W and Super Key / Candidate Key is XZ

a. FD: **X** → **Y** does not satisfy the definition of 3NF, that neither X is Super Key nor Y is a prime attribute.

b. FD: **Y** → **P** does not satisfy the definition of 3NF, that neither Y is Super Key nor P is a prime attribute.

c. FD: **Z** → **W** satisfies the definition of 3NF, that neither Z is Super Key nor W is a prime attribute.

Convert the table R(X, Y, Z, W, P) into 3NF:

Since all the FD = { X → Y, Y → P, and Z → W} were not in 3NF, let us convert R in 3NF

R1(X, Y) {Using FD X → Y}

R2(Y, P) {Using FD Y → P}

R3(Z, W) {Using FD Z → W}

And create one table for Candidate Key XZ

R4(X, Z) { Using Candidate Key XZ }

All the decomposed tables R1, R2, R3, and R4 are in 2NF(as there is no partial dependency) as well as in 3NF.

Hence decomposed tables are:

R1(X, Y), R2(Y, P), R3(Z, W), and **R4(X, Z)**

Question 3: Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, P → ST, Q → U, U → VW, and S → XY}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Solution: Let us construct an arrow diagram on R using FD to calculate the candidate key