FUNCTIONAL DEPENDENCY

Functional Dependency

- **It is a relationship that exists when one attribute uniquely determines another attribute.**
- It is a set of constraints between two attributes in a relation.
- If R is a relation with attributes X and Y.
- **Functional dependency can be expressed as : X -> Y**
  - In this functional dependency, X is determinant and Y is dependent.
  - X uniquely determines Y.
  - Y is functionally dependent on X.

Functional Dependency(Contd...)

Example:

Consider the relation: student (id, name, dateofbirth)

- The functional dependencies for this relation are:

  id -> name

  id -> dateofbirth

- **If every attribute of a relation R is determined by a particular attribute (say A), then *A is a primary key* attribute for the relation.**

Functional Dependency(Contd...)

Trivial functional dependency:

•Consider a relation R (X, Y, Z). A functional dependency X ->Y is said to be the trivial functional dependency if and only if: **Y is a subset of X.**

•In other words, if R.H.S of some functional dependency is the subset of the L.H.S of the functional dependency then, it is called as trivial functional dependency.

Example: XY -> X or XY -> Y are trivial functional dependencies.

Non-trivial functional dependency:

•If there is at least one attribute in the R.H.S that is not part of L.H.S, such functional dependency is called Non-trivial functional dependency.

Example: XY -> Z is a non-trivial functional dependency.

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES: (F^+)

- **"The set of all functional dependencies logically implied by F or determined by using the given set of functional dependencies is the closure of F"**
- We denote the closure of F by **F^+**

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES: (F^+) (Contd...)

The closure F^+ can be identified by applying Armstrong's Axioms:

- if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
- if $\alpha \rightarrow \beta$, then $\gamma\ \alpha \rightarrow \gamma\ \beta$ **(augmentation)**
- if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**

We can further simplify manual computation of F^+ by using the following *additional rules*.

- If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds **(union)**
- If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds **(decomposition)**
- If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds **(pseudotransitivity)**

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES: (F^+) (Contd...)

Procedure to compute closure of Functional dependencies (F^+)

$F^+ = F$

repeat

for each functional dependency f in F^+

apply reflexivity and augmentation rules on f

add the resulting functional dependencies to F^+

for each pair of functional dependencies f1and f2 in F^+

if f1 and f2 can be combined using transitivity

then add the resulting functional dependency to F^+

until F^+ does not change any further

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES: (F^+) (Contd...)

Example : To find the closure F^+

Let R (A,B,C,G,H,I) be the relation schema and the set of functional dependencies { A → B, A→ C, CG→H, CG→I, B→ H } . Find the closure of functional dependencies.

Solution:

Given:

The set of functional dependencies:

A→ B

A→ C

CG→H

CG→I

B→ H

CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES: (F^+) (Contd...)

Members of F^+ are:

1)	A→H	By applying ***transitivity*** on $A \rightarrow B$ *and* $B \rightarrow H$, **A→ H holds.**
2)	A→BC	By applying union on A→B and A→C
3)	CG→HI	By applying ***union*** on CG→H and CG→I, **CG→ HI holds.**
4)	AG→ I	i) By applying ***augmentation*** rule on A→C, AG→CG ii) By applying ***transitivity*** on AG→ CG and **CG→I holds.**
5)	AG→H	i) By applying ***augmentation*** rule on A→C, AG→CG ii) By applying ***transitivity*** on AG→ CG and **CG→H holds.**

A→ B

A→ C //Given

CG→H

CG→I

B→ H

CLOSURE OF ATTRIBUTE SETS (α^+)

- Closure of an attribute α is a set of attributes which can be derived from it. Used to identify the key. It is denoted as α^+.
- We say that an attribute B is functionally determined by α if $\alpha \rightarrow B$
- *To test whether a set α is a super key, the set of attributes functionally determined by α has to be computed.*

Procedure to compute attribute closure (α^+):

```
result := α;
while (changes to result) do
        for each β → γ in F do
                begin
                        if β ⊆ result then result := result ∪ γ
                end
```

CLOSURE OF ATTRIBUTE SETS (α^+) Contd...

Example : To find the attribute set closure α^+

Given the set F of functional dependencies

$R = (A, B, C, G, H, I)$

$F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$

Find the closure of the following attributes: A, B, C, AB, BC, BG, AG . Identify which of the above can act a key for the given relation R.

Closure of attributes

Closure of attribute A

$(A^+) \rightarrow \{ABCH\}$ *(Because $A \rightarrow A$, $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow H$)*

Closure of attribute B

$(B^+) \rightarrow \{BH\}$ *(Because $B \rightarrow B$, $B \rightarrow H$)*

Closure of attribute C

$(C^+) \rightarrow \{C\}$ *(Because $C \rightarrow C$)*

Closure of attribute AB

$(AB^+) \rightarrow \{ABCH\}$ *(Because $AB \rightarrow AB$, $A \rightarrow C$, $B \rightarrow H$)*

Closure of attribute BC

$(BC^+) \rightarrow \{BCH\}$ *(Because $BC \rightarrow BC$, $B \rightarrow H$)*

Closure of attribute BG

$(BG^+) \rightarrow \{BGH\}$ *(Because $BG \rightarrow BG$, $B \rightarrow H$)*

Closure of attribute AG

$(AG^+) \rightarrow \{AGBCHI\}$ *(Because $AG \rightarrow AG$, $A \rightarrow B$, $A \rightarrow C$, $CG \rightarrow H$, $CG \rightarrow I$)*

Since, Attribute AG determines all attributes of the Relation, it can act as a Key attribute for the Relation R.

Finding candidate keys for the relation R from the set of functional dependencies

Example 1:

Consider a relational schema R=(A,B,C,G,H,I) and a set of functional dependencies F = {A → B, A → C, CG → H, CG → I , B → H }. Find all possible candidate key and super keys of R.

Solution:

Given:

The set of functional dependencies:

A→ B
A→ C
CG→H
CG→I
B→ H

Identifying the attributes in candidate key:

- In the given set of functional dependencies, the attributes that is not present in the dependent side (R.H.S of the functional dependency), will be a part of candidate key.
- In the given set of functional dependencies, the **attributes A and G** are not present in the R.H.S of the functional dependency.
- Therefore, *attributes AG will be in the candidate key.*

Finding candidate keys for the relation R from the set of functional dependencies(Contd...)

Test for Candidate Key

Find AG^+

$AG^+ \rightarrow AGBCHI$ (Attributes AG determines all the attributes of relation R, so **attributes AG will be in the candidate key**)

Therefore, The candidate key for relation R is : AG

The super keys are : AG, ABG,ACG,AHG,AIG,ABCG,ABHG,ABIG,ACGH,ACGI etc.,

Given:

The set of functional dependencies:

$A \rightarrow B$

$A \rightarrow C$

$CG \rightarrow H$

$CG \rightarrow I$

$B \rightarrow H$

Finding candidate keys for the relation R from the set of functional dependencies (Contd...)

Example 2: (Gate 2013)

Consider a relation R(A,B,C,D,E,F,G,H) and a set of functional dependencies given as

F = {
CH→G
A→BC
B→CFH
E→A
F→EG
}

Find the number of candidate keys.

Solution:

Given:

The set of functional dependencies

F= {
CH→G
A→BC
B→CFH
E→A
F→EG }

Test for Candidate Key

- In the given set of functional dependencies, the attribute D is not present in the R.H.S of the functional dependency.
- Therefore, the **attribute D will be in the candidate key**.

Finding candidate keys for the relation R from the set of functional dependencies(Contd...)

Find D^+

$D^+ \rightarrow D$ (Attribute D alone cannot determine all the attributes of the relation. So, D is not a candidate key)

We will check other possible combination with attribute D.

$AD^+ \rightarrow ADBCFHEG$	**(all attributes of R are identified with AD. So, AD is a candidate key)**
$BD^+ \rightarrow BDCFHEGA$	**(all attributes of R are identified with BD. So, BD is a candidate key)**
$CD^+ \rightarrow CD$	(all attributes of R are not identified by CD. So, CD is not a candidate key)
$DE^+ \rightarrow DEABCFHG$	**(all attributes of R are identified with DE. So, DE is a candidate key)**
$DF^+ \rightarrow DFEGABCH$	**(all attributes of R are identified with DF. So, DF is a candidate key)**
$DG^+ \rightarrow DG$	(all attributes of R are not identified by DG. So, DG is not a candidate key)
$DH^+ \rightarrow DH$	(all attributes of R are not identified by DH. So, DH is not a candidate key)

Therefore, the number of candidate keys = 04 (They are AD,BD,DE,DF)

Given:

The set of functional dependencies

F= {
CH→G
A→BC
B→CFH
E→A
F→EG }

CANONICAL COVER/
MINIMAL COVER/
MINIMAL SET OF FUNCTIONAL DEPENDENCIES/
IRREDUCIBLE SET OF FUNCTIONAL DEPENDENCIES

Definition of Canonical Cover:

•"A canonical cover of a set of functional dependencies F is a **simplified set of functional dependencies that has the same closure as the original set F**."

•The canonical cover F_c of a set of functional dependencies F such that ALL the following properties are satisfied :

- F logically implies all dependencies in F_c.
- F_c logically implies all dependencies in F.
- No functional dependency in F_c contains an extraneous attribute.
- Each left side of a functional dependency in F_c is unique.

Definition of Extraneous attributes:

•An attribute of an FD is said to be extraneous if we can remove it without changing the closure of the set of FD.

Example: Given a relational Schema R(A, B, C) and set of Function Dependency FD = { A → BC, B → C, A->B, AB → C }. Find the canonical cover.

Solution:

Step 1: Apply Decomposition rule on the given set of FDs.

A -> B

A -> C

B -> C

A -> B

AB -> C

The set of FD in step 1 : { A -> B,

A -> C,

B -> C,

AB ->C }

Step 2: Find closure of the left side of each of the given FD (in Step 1) by including that FD and excluding that FD, if closure in both cases are same then that FD is redundant and we remove that FD from the given set, otherwise if both the closures are different then we do not exclude that FD.

FD : A -> B

Including this FD, A+ = {ABC}

Excluding this FD, A+ = {AC}

Both closures are not same, so this FD cannot be removed.

FD : A -> C

Including this FD, A+ = {ABC}

Excluding this FD, A+ = {ABC}

Both closures are same, so this FD is redundant and can be removed.

Therefore, the attribute C is called as EXTRANEOUS ATTRIBUTE

Revised set of F is : { A -> B

B-> C

AB->C}

FD : B -> C

Including this FD, B+ = {BC}

Excluding this FD, B+ = {B}

Both closures are not same, so this FD cannot be removed.

FD : AB -> C

Including this FD, AB+ = {ABC}

Excluding this FD, AB+ = {ABC}

Both closures are same, so this FD is redundant and can be removed.

Revised set of F is : { A -> B

B-> C }

After this step, F does not change anymore. Hence the required canonical cover is,

The Canonical Cover F_C : { A -> B

B-> C }

LOSSLESS DECOMPOSITION

- Let R1 and R2 form a decomposition of a relation R.

Lossless Decomposition:

- The decomposition is a lossless decomposition if **there is no loss of information by replacing original relation r (R)** with two relation schemas r1(R1) and r2(R2).
- This can be expressed as

$$\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) = r$$

- If we project r onto R1 and R2, and compute the natural join of the projection results, we get back exactly r.

Checking Lossless Decomposition:

If **R1 ∩ R2 forms a superkey** of either R1 or R2, the decomposition of R is a lossless decomposition.

Examples:

1) **Consider the employee schema employee (ID, name, street, city, salary) is decomposed into two relations**

   Employee1 (id,name)

   Employee2 (name,street, city,salary)

Identify whether the decomposition is lossy or lossless.

Solution:

Testing Lossless Decomposition

- **The decomposition is lossy** since the join result has lost information about which employee id's correspond to which addresses and salaries, in the case where two or more employees have the same name.
- Here, the ***intersection of two schemas Employee1 (id,name) ∩Employee2 (name,street, city,salary) is name.***
- Name attribute cannot be a superkey of the either employee1 or employee2 because two or more persons can have the same name
- ***Therefore, the decomposition is lossy.***

2) Consider the schema Ins_ dept (ID, name, salary, dept_name, building, budget) decomposed into:

instructor (ID, name, dept_name, salary)

department (dept_name, building, budget)

Identify whether the decomposition is lossy or lossless.

Solution:

Testing Lossless Decomposition

- Here, the intersection of two schemas

  instructor (ID, name, dept_name, salary) ∩ *department (dept_name, building, budget) is dept_name.*
- Dept_name attribute is a superkey in employee2.
- *Therefore, the decomposition is lossless.*

NORMALIZATION

Normalization

- Normalization is a systematic approach of **decomposing tables to eliminate data redundancy(repetition)** and undesirable characteristics like
  - Insertion Anamolies.
  - Update Anamolies.
  - Deletion Anamolies.
- If a table is not properly normalized and have data redundancy then it will cause wastage of disk space.
- Insertion, Updation and Deletion Anomalies become very frequent.

Problems with Redundancy:

- **Redundancy:** Redundancy is a condition in which same piece of data is stored repeatedly in many places.

Insertion, Deletion and Updation Anamolies

Consider the relation student.

ID	NAME	BRANCH	HODNAME	PHONE
1	Aron	CSE	XYZ	23456789
2	Abi	CSE	XYZ	23456789
3	kavi	CSE	XYZ	23456789

Insertion Anomaly:

•If new student information is added, the Branch information is also added to the relation repeatedly. (i.e, branch, hodname and phone is repeated for all records)

Deletion Anomaly:

•If we start deleting records of student, the branch information is also deleted simultaneously.

•When we delete the last student information from the student relation, unintentionally we have deleted all branch information along with student data.

Updation Anomaly:

If we need to update the HODNAME, the updation must be made to all the rows of student relation.

During modification even if a single row is missed out, this causes data inconsistency.

Insertion, Deletion and Updation Anamolies(Contd...)

Solution to resolve the redundancy problem

•Using normalization, this redundancy can be removed by breaking the student relation into student_info and branch_info.

Student_info

ID	NAME	BRANCH
1	Aron	CSE
2	Abi	CSE
3	Kavi	CSE

Branch_info

BRANCH	HODNAME	PHONE
CSE	XYZ	23456789

•By doing so, the branch information need not be repeated for every student record.

•Thus, insertion/deletion/updation anomalies are overcome.

VARIOUS NORMAL FORMS

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce Codd Normal form (BCNF)
- Fourth normal form (4NF)
- Fifth normal form (5NF)

Note:

•**Prime Attribute** - An Attribute that is a part of the candidate key

•**Non-Prime Attribute** - An Attribute that is not a part of candidate key

•A primary key is also a candidate key, a candidate key is also a super key

First Normal Norm (1NF)

- **A relation schema R is in first normal form (1NF) if the domain of all attributes of R are atomic.**
- If a table is not in 1st normal form, it is considered as poor database design.

Rules for a table in 1NF:

- Each column should contain atomic values (i.e. column should not contain multiple values).
- A column should contain values that are of the same domain.
- Each column should have unique name.
- Order of storing the data doesn't matter.

First Normal Norm (1NF) (Cont…)

Example:

Consider a table 'student'.

Id	Name	Subjectname
101	Arun	OS,DBMS
102	Karthi	JAVA
103	Shri	C, C++

- The 'student' table is not in 1NF. This is because, the domain of the students with ID 101 and 103 are not atomic (contains multiple subjects, marks and teachername).

Converting to 1NF:

Id	Name	Subject
101	Arun	OS
101	Arun	DBMS
102	Karthi	JAVA
103	Shri	C
103	Shri	C++

- Therefore, the student table is in 1NF.

Second Normal Form(2NF)

- Second Normal Form **applies to relations with composite keys**, that is, relations with a primary key composed of two or more attributes.
- A relation R is said to be in 2NF if :
  - **It is in first normal form**
  - **No partial functional dependencies exist.** (i.e., All non-prime attributes are fully functional dependent on the primary key of R)

Second Normal Form(2NF) (Cont...)

Assume

SSN	Name	Project number	hours
101	A	P1	5
101	A	P2	3
102	B	P1	6

Example: Consider the relation **employee_project**

ssn	name	projectnumber	hours

Candidate Key/Primary Key: {{ssn, projectnumber}}

The FDs that exists in the above relation are:

ssn -> name ----- FD 1

ssn, projectnumber -> hours -----FD 2

- Since an employee can work in multiple projects, the hours he worked can be determined based on both ssn and projectnumber. i.e. both ssn, projectnumber together form the primary key of this table.
- So, Partial Fnctional Dependency exists in FD1 as name of the employee can be identifed by the attribute ssn alone. Thus the table is **not in 2NF.**

Converting to 2NF:

Decompose the employee_project relation into

employee

ssn	name

FD: ssn -> name

project

ssn	projectnumber	hours

FD: ssn,projectnumber -> hours

Now, the FD of both the relations do not have any Partial Functional Dependency.

Hence it is in 2NF.

Third Normal Form(3NF)

A relation schema *R* is in **3NF** if

- It is in 2NF.

- no nonprime attribute of *R* is transitively dependent on the primary key.

Transitive Dependency:

- When a non-prime attribute is determined by other non-prime attributes rather than being determined by the prime attributes or primary key, then it is called transitive dependency.

Example for Transitive Dependency,

Consider the relation R

A	B	C

A - primary key.
Prime attributes - A
Non-prime attributes - B,C

Suppose if, A->B

B->C

Transitive dependency exists in the relation as the non prime attribute C is determined by non-prime attribute B.

Third Normal Form(3NF)

Assume

ssn	ename	bdate	addr	dno	dname	Dhod_ssn
201	Abi	10-01-1988	Chennai	D1	CSE	101
202	Arun	21-09- 1980	Pune	D2	ECE	102
203	John	18-06-1987	Madurai	D2	ECE	102
204	Balaji	17-05-1988	Tirupati	D1	CSE	101

Example:

Consider the relation **employee_department** **Candidate Key : {ssn}**

ssn	ename	bdate	addr	dno	dname	dhod_ssn

The FDs for this relation are:

ssn -> ename,bdate,addr, **dno** ----- FD1

dno ->dname,dhodssn ----- FD2

Here, the non-prime attributes dname, dhodssn can be determined by another non-prime attribute dno. Thus, Transitive dependency exists, the **relation is not in 3NF.**

Converting to 3NF:

Decompose the employee_department relation into

employee

ssn	ename	bdate	addr	dno

FD: ssn -> ename, bdate, addr, dno

department

dno	dname	dhod_ssn

FD: dno ->dname, dhod_ssn

Now, the FD of both the relations do not have any transitive Functional Dependency.

Hence, it is in 3NF

BOYCE CODD NORMAL FORMF(3.5NF)

A relation schema *R* is in **BCNF** if

- It is in 3NF.

- For every functional dependency X-> Y, X should be the superkey of R

Example: Consider the following relation

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java

Primary Key:

Student_id + subject

Let us assume,

1. A student can enroll for more than one subject.
2. For each subject, each student of that subject is taught only by one teacher.
3. One Professor teaches only one subject
4. Multiple Professors can handle a single subject.

BOYCE CODD NORMAL FORM(Contd...)

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java

Let us assume,
1. A student can enroll for more than one subject.
2. One Professor teaches only one subject
3. Multiple Professors can handle a single subject.

The functional dependencies for the **student_mentor table** are table are:

student_id, subject -> professor ---FD1

professor -> subject ---FD2

PrimaryKey : Student_id + subject

The given table is in the following normal forms

- 1NF as all columns have atomic values
- 2NF since part of the primar key does not determine any other attribute
- 3NF since, a non-prime attribute is not determined by a non-prime attribute.

Checking for student_mentor table to be in BCNF:

- In student_id, subject -> professor, student_id+subject is one of the superkey
- In Professor ->Subject (FD2), Professor is not a super key.

Hence the table is not in BCNF

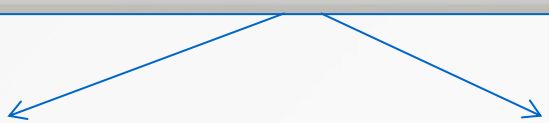
BOYCE CODD NORMAL FORM(Contd...)

Converting to BCNF

Alternative 1:

Decompose the student_mentor relation into

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java



Student_id	Professor
101	P.Java
101	P.Cpp
102	P.Java2
103	P.Chash
104	P.Java

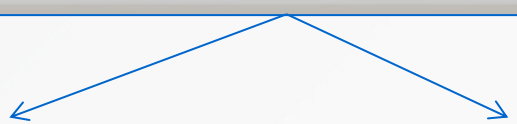
Professor	Subject
P.Java	Java
P.Cpp	C++
P.Java2	Java
P.Chash	C#

BOYCE CODD NORMAL FORM(Contd...)

Alternative 2:

Decompose the student_mentor relation into

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java



Student_id	P_id
101	P01
101	P03
102	P02
103	P04
104	P01

P_id	Professor	Subject
P01	P.Java	Java
P03	P.Cpp	C++
P02	P.Java2	Java
P04	P.Chash	C#

Fourth Normal Form(4NF)

A relation R is in 4NF if

- It is in BCNF
- **No multivalued Functional Dependencies exists**

Multivalued Functional Dependency(represented as A->>B)

A Functional dependency A->B is called multivalued functional dependency if ALL these conditions are true:

1. For a single value of A, more than one value of B exists.

2, Table should have atleast 3 columns

3. For a table with columns A,B,C, if A and B have multivalued dependency them B and C should be independent of each other.

Fourth Normal Form(4NF) (Contd...)

Example: Consider the student table

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

It is noted in this table that one student can enroll in more than one subject and has more than one hobby. The functional Dependencies of this table are:

s_id -->> course

s_id -->>hobby

there is no relationship between course and hobby. they are independent.

This table has multivalued dependency and hence not in 4NF.

Fourth Normal Form(4NF) (Contd...)

Converting the table to 4NF

To convert this table to satisfy 4NF, the table is decomposed as follows:

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

CourseOpted TABLE

s_id	course
1	Science
1	Maths
2	C#
2	Php

Hobbies TABLE

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

The Decomposed tables does not have any multivalued dependency and hence it is in 4NF.

FIFTH NORMAL FORM (5NF)

- A relation is in 5NF if
  - It is in 4NF.
  - It cannot be further non-loss decomposed.
- Applying join operation on the decomposed tables, must result in the original state of the table without any loss of records or extra records.

Example:

Consider the course_staff relation

Course	Staff	Book
DBMS	A	X
DBMS	B	Y
CA	C	Z

The functional dependencies are:

Course ↠ staff (FD1)

Course ↠ book (FD2)

The table has non-trivial multivalued dependency, it is not satisfy 4NF.

FIFTH NORMAL FORM (5NF) (Contd...)

Therefore, the course_staff relation is decomposed into R1 & R2 as follows :

R1

course	staff
DBMS	A
DBMS	B
CA	C

The functional dependencies are:

Course → staff

Course U staff = R1 (trivial multivalued dependency)

The tables are in 4NF.

R2

Course	book
DBMS	X
DBMS	Y
CA	Z

The functional dependencies are:

Course→ book

Course U book = R2 (trivial multivalued dependency)

To check whether it is in 5NF (To check whether join dependency exists):

R1 join R2

Course	Staff	book
DBMS	A	X
DBMS	A	Y
DBMS	B	X
DBMS	B	Y
CA	C	Z

When the decomposed tables R1 & R2 are joined, it results in additive tuples. **(i.e Join Dependency exists)**

R1 join R2 ≠ course_staff

So, it is not in 5NF.

FIFTH NORMAL FORM (5NF) (Contd...)

To avoid the problem:

The relation can be decomposed into three relations as follows:

R1

Course	Staff
DBMS	A
DBMS	B
CA	C

R2

Course	book
DBMS	X
DBMS	Y
CA	Z

R3

staff	book
A	X
B	Y
C	Z

(R1 join R2) join R3

Course	Staff	book
DBMS	A	X
DBMS	B	Y
CA	C	Z

(R1 join R2) join R3 = course_staff.

Thus, the table is in 5NF.