Chapter 5 :- Database constraint and Relational Database Design.

Functional Dependencies.

- They are the fundamental to the process of Normalization.
- Here, functional dependency plays a key role in differentiating good database design from bad database design.
- It describes the relationship between attributes in a table.
- For attributes X and Y in Relation R, functional dependency between X and Y is shown as :-

$$X \rightarrow Y$$

Here, X is determinant
Y is the functional dependent on X.

eg:-

eid	ename	age	salary
101	Ram	25	15000
102	Shyam	28	20000
103	Sita	26	20000

Here, eid → ename ; eid uniquely determines ename.

ename → eid ; is not always true.

eid → age ; True
eid → salary ; True

age → salary ; False

age → eid ; False

employee (eid, projectNO, Hours, ename, pname, Plocation)

Here, eid → ename
projectNO → {pname, Plocation}

{eid, projectNO} → Hours.

Types of functional dependencies

i) Full functional dependency

- X → y is a full functional dependency if the removal of any attribute A from X removes the dependency.
- A full functional dependency occours when it is already functional dependency and the set of attributes on the left side can't be reduced any further.
- example.

orderNo	LineNo	qty	price
A001	L001	10	200
A002	L001	20	400
A002	L002	30	800
A004	L001	15	300

Here, {orderNo, LineNo} → qty

{orderNo, LineNo} → price are the full functional dependencies.

ii) partial functional dependency.

- A functional dependency X→Y is partial dependency if some attributes A∈X can be removed from X and the dependency still holds.

eg:-

{eid, phone} → name

Here,

remove phone, then {eid} → name still holds true, so it is partial functional dependency.

iii) Transitive Dependency

- It occours where there is an indirect relationship that causes a functional dependency.

- If X→Y and Y→Z, then X→Z is a transitive dependency.

iv) Trivial dependency

- It occurs when the functional dependency of an attribute is described on collection of attributes that includes the original attribute.
- Functional dependency is trivial if RH.S is a subset of L.H.S

eg:-

{name, ssN} → ssN

v) Non-trivial dependency

- It is the one that is not trivial

eg:-

{s#, p#} ⇒ {s#, qty}

Axioms or Rules of inference for Functional Dependencies.

- It provides a simple technique for reasoning about functional dependies.

a) Reflexivity Rule

- If y is a subset of x then x → y holds.

b) Augmentation Rule

- If x → y holds and z is a set of attribute then zx → zy holds.

c) Transitivity Rule

If x → y and y → z holds then x → z holds.

d) union Rule

- If x → y holds and x → z holds then x → yz holds.

e) Decomposition Rule

- If X → yz holds then x → y and x → z holds.

f) pseudo Transitivity Rule

- If x → y holds and zy → p holds then xz → p holds.

g) self determination

A → A

eg:- R = (A, B, C, G, H, I)

set of functional dependencies F { A → B, A → C, CG → H, CG → I, B → H }

Then, we have.

i) A → B, A → C then A → BC {union Rule}

ii) A → B, B → H then, A → H {Transitivity Rule}

iii) CG → H, CG → I then, CG → HI {union Rule}

iv) AG → I then, {Pseudo Transitivity Rule}

eg:- If $F = \{ A \to B, C \to X, BX \to Z \}$

Prove or disprove $AC \to Z$

soln

$C \to X, BX \to Z \Rightarrow BC \to Z$ ∴ {using, pseudo-transitivity Rule}

$A \to B, CB \to Z \Rightarrow AC \to Z$

Closure of a set of functional dependencies.

- The set of functional dependencies that is logically implied by F is called closure of F and is written as F^+.
- The closure of F denoted by F^+ is the set of all Functional dependencies entailed by F.
- $F^+ = \{ x \to y \mid F \models X \to y \}$ (pie)
- If F is a set of Functional dependencies, we can compute F^+ directly from formal defination of functional dependency.
- If F were large, this process be lengthy and difficult. so, Armstrong axioms useful for fast and accurate result.

eg:- $R = \{A, B, C, D\}$

$F = \{ A \to B, A \to C, BC \to D \}$

$F^+ = ?$

soln

$A \to B, A \to C$ then $A \to BC$ {union Rule}

$A \to B, BC \to D$ then, $AC \to D$ {pseudo Transitive Rule}

$A \to C, BC \to D$ then, $AB \to D$ {pseudo Transitive Rule}

eg:- $R = \{ A, B, C, D, E, F \}$

$F = \{ A \to BC, B \to E, CD \to EF \}$

List F^+ and prove $AD \to F$

soln

$A \to BC$ then $A \to B, A \to C$ {Decomposition Rule}

$CD \to EF$ then, $CD \to E, CD \to F$ {Decomposition Rule}

$AD \to E, AD \to F$ proved //

Closure of attribute set

- Closure of attribute set is the set of attributes which are functionally dependent on the attribute set.
- Given a set of attributes $A_1 \ldots A_n$ the closure of $\{A_1 \ldots A_n\}^+$ is the set of attributes B such that $A_1 \ldots A_n \to B$

eg:-
name → color
category → department
color, category → price

Now, $\{name\}^+ = \{name, color\}$

$\{color\}^+ = \{color\}$

$\{name, category\}^+ = \{name, category, color, department, price\}$

An Algorithm to compute α^+, the closure of α under F.

result = α
while (changes to result) do
for each functional dependency $\beta \to \gamma$ in F do
begin
if $\beta \subseteq$ result then
result = result $\cup \gamma$
end

eg:- $R = \{A, B, C, D, E, H, I\}$

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

Compute $(AG)^+$

Soln Let,

result = $\{AG\}$

1st iteration

For $A \to B$

$A \subseteq \{AG\}$; True so,
$\therefore$ result = $\{ABG\}$

For $A \to C$

$A \subseteq \{ABG\}$; True so,
$\therefore$ result = $\{ABCG\}$

For $CG \to H$

$CG \subseteq \{ABCG\}$; True
$\therefore$ result = $\{ABCGH\}$

For $CG \to I$

$CG \subseteq \{ABCGH\}$; True
$\therefore$ result = $\{ABCGHI\}$

For $B \to H$

$B \subseteq \{ABCGHI\}$; True so,

$\therefore$ result = $\{ABCGHI\}$

2nd iterations :-

We get the final result
= $\{ABCGHI\}$

The result doesn't change further

$\therefore (AG)^+ = \{ABCGHI\}$

eg:- $R = \{A, B, C, D, E, F\}$

$F = \{A \to BC, E \to CF, B \to E, CD \to EF\}$

Compute $(AB)^+$

Soln

1st iteration

Let, result = AB then,

For $A \to BC$

$A \subseteq \{AB\}$; true so,

$\therefore$ result = $\{ABC\}$

For $E \to CF$

$E \subseteq \{ABC\}$; False so,

$\therefore$ result = $\{ABC\}$

For $B \to E$

$B \subseteq \{A, BC\}$; True so,

$\therefore$ result = $\{ABCE\}$

For $CD \to EF$

$CD \subseteq \{ABCE\}$; False so,

$\therefore$ result = $\{ABCE\}$

2nd iteration.

For $A \to BC$

$A \subseteq \{ABCE\}$ True so,

$\therefore$ result = $\{ABCE\}$

For $E \to CF$

$E \subseteq \{ABCE\}$ True

$\therefore$ result = $\{ABCEF\}$

For $B \to E$

$B \subseteq \{ABCEF\}$ True

$\therefore$ result = $\{ABCEF\}$

For $CD \to EF$

$CD \subseteq \{ABCEF\}$ False

$\therefore$ result = $\{ABCEF\}$

3rd iteration

(A - 5) Then we get final result

$= \{ABCEF\}$ The result doesn't change further $(AB)^+ = \{ABCEF\}$.

Extraneous Attribute :-

- An attribute of functional dependency is extraneous if we can remove it without changing the closure of the set of functional dependencies.

Formal defination:
consider F as set of functional dependency and Function dependency $\alpha \to \beta$ in F. Then.

- attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - (\alpha \to \beta)) \cup ((\alpha - A) \to \beta)$
- attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - (\alpha \to \beta)) \cup (\alpha \to (\beta - A))$ logically implies F.

a) To test if attribute A is extraneous in α,

i) compute $(\alpha - A)^+$ using dependencies in F,
ii) check that $(\alpha - A)^+$ contains A; if it does, A is extraneous.

b) To test if attribute A is extraneous in β.

i) compute α^+ using only dependencies in F'.
Here, $F' = (F - (\alpha \to \beta)) \cup ((\alpha \to (\beta - A))$

ii) check that $(\alpha)^+$ contains A, if it does, A is

extraneous.

eg:- $F = \{A \to B, B \to C, AC \to D\}$.

for $AC \to D$

$\alpha = AC$

Let, take an attribute A then,

$= (AC - A)^+$

$= C^+ = \{C\}$

$A \notin C^+ = \{C\}$

∴ A is not extraneous.

$(AC - C)^+$

$= A^+ = \{ABCD\}$

$C \in A^+ = \{ABCD\}$

∴ C is extraneous.

$\therefore F = \{A \to B, B \to C, A \to D\}$

eg:- $F = \{A \to B, B \to C, A \to CD\}$

Soln

$A \to CD$

Let, $\beta = CD$ $\alpha = A$

Let, take an attribute C than,

$\alpha^+ = A^+$

$F' = \left(\{ A \to B, B \to C, A \to CD \} - (A \to CD) \right) \cup (A \to (CD - D))$

$F' = (\{ A \to B, B \to C, A \to C \})$

$\therefore A^+ = \{ABC\}$

$\therefore C \in A^+ = \{ABC\}$

$\therefore$ 'C' is extraneous.

For attribute 'D' we get,

$D \notin A^+ = \{ABC\}$

$\therefore$ D is not extraneous.

$\therefore F = \{ A \to B, B \to C, A \to D \}$

Determining Candidate Key.

1) Compute closure of each attributes.
2) Any attributes is called candidate key of any relation if attribute closure is equal to the relation.

- Attribute that an part of candidate key are called prime attribute.

- attribute that are not part of candidate key are called non-prime attribute.

eg:- $R = (A, B, C, D)$

$F = \{ AB \to C, C \to D, D \to A \}$

List all candidate keys, prime and non-prime attributes.

soln

For A

$A^+ = \{ A \}$; A is not candidate key.

For B

$B^+ = \{ B \}$; The result doesn't match so, it is not candidate key.

For C

$C^+ = \{ CDA \}$; C is not candidate key.

For D

$D^+ = \{DA\}$; D is not candidate key.

For AB

$(AB)^+ = \{ ABCD \}$; AB is candidate key.

For AC

$(AC)^+ = \{ACD\}$; AC is not candidate key.

For (AD)

$(AD)^+ = \{AD\}$; not a candidate key.

For BC

$(BC)^+ = \{BCDA\}$; BC is not (yes) candidate key.

For BD

$(BD)^+ = \{BDAC\}$; BD is a candidate key.

For CD

$(CD)^+ = \{CDA\}$; CD is not a candidate key.

∴ candidate key = $\{AB, BC, BD\}$

prime attribute = $\{A, B, C, D\}$.

Decomposition

- It refers to the breaking down of one table into Multiple tables.

→ Desirable properties of decomposition

a) Attribute prevention
b) Dependency prevention
c) Lack of redundancy
d) Lossy Decomposition
e) Non-loss or lossless Decomposition.

a) Attribute prevention

- preserving all the attributes of relation being decomposed.

b) Dependency prevention.

- Let F be the dependencies on a relation R which is decomposed in $R_1, R_2 \ldots R_n$.
- We can partition the dependencies given by F such that $F_1, F_2, \ldots F_n$ are dependencies that only involve attributes from relation $R_1, R_2 \ldots R_n$ respectively.
- If the union of dependencies imply all the dependencies in F, then we say decomposition has preserved dependencies otherwise not.
- If the decomposition doesnot preserve the dependencies F, then the decomposed relation may contain relation that do not satisfy F.

c) Lack of Redundancy

- Redundancy should be avoided as much as possible.

d) lossy decomposition

- Loss of information due to decomposition is called lossy decomposition.

e) Non-loss decomposition

- It refers to decomposition where all information is preserved

eg:-

ABC

A	B	C
a_1	100	c_1
a_2	200	c_2
a_3	300	c_3
a_4	200	c_4

$\Rightarrow$

AB

A	B
a_1	100
a_2	200
a_3	300
a_4	200

BC

B	C
100	c_1
200	c_2
300	c_3
200	c_4

$\Downarrow$

AUBUC

A	B	C
a_1	100	c_1
a_2	200	c_2
a_2	200	c_4
a_3	300	c_3
a_4	200	c_2
a_4	200	c_4

Normalization

- Database normalization is a technique of organizing data in the database.
- It is the process of removing redundancy and undesirable and characteristics like insertion, update and deletion anamolies.
- Mainly two purposes of normalization are:-

1) Eliminating redundant data.
2) Ensuring data dependencies make sense.

- It involves dividing table into two or more tables and defining relationship between the tables.

Normal forms

- It represents the guidelines for record design.
- E.F codd originally established three normal forms. 1NF, 2NF, 3NF.
- others Forms are BCNF(Boyce codd Normal Form) 4NF, 5NF, DKNF(Domain Key Normalization)

unNormalized Form

- It contains one or more repeating groups or each row may contain multiple set of values for some columns.
- Multiple values in a single row are also called non-atomic values.

eg:-

cid	Name	Address	phone
101	Ram	Ktm	9823048543
			9861977612
102	Sita	patan	982323404
103	Hari	Dharan	9803540511
			9863944091

Fig:- unNormalized Form (UNF)

First Normal Form (1NF)

- A relation is in 1NF if and only if all columns are atomic i.e. no repeating values in columns.
- Now, Fig 1 will become.

Cid	Name	address	phone
101	Ram	Ktm	9823048543
101	Ram	Ktm	9861977612
102	Sita	Patan	982323904
103	Hari	Dharan	9803540511
103	Hari	Dharan	9863444041

- In 1NF there should be no repeating columns.

eg:-

id Name Phone Child1 Chaild2 Child3

↓

id Name phone child.

- Values of each attribute is atomic.
- No composite values.
- All entries in any columns must be of same kind.
- each column must have unique name.
- No two rows are identical.

Second Normal Form (2NF)

- A relation is in 2NF if,
  - it is in 1NF
  - all attributes depend on full primary key.

eg:-

personID	projectID	Name	projectName	Phone
1	1	Ram	Database	9863444041
2	1	Sita	Database	9803540511
1	2	Ram	web	9823622295
2	2	Sita	web	9808580414

Fig:-2

In Fig:-2 Given relation is in 1NF but not in 2NF because all attributes are not fully dependent on primary key (personID, projectID).

Now decompose above table as follows.

personID	Name	Phone
1	Ram	9863444041
2	Sita	9803540511

table 1

projectID	projectName
1	Database
2	web

table 2

id	personID	projectID
1	1	1
2	2	1
3	1	2
4	2	2

table 3

Third Normal Form (3NF)

- A relation is in 3NF if
  - it is is in 2NF.
  - There is no transitive functional dependency i.e. There should not be case that non prime attribute is determined by another non prime attribute.

eg:-

St_id	St_name	DoB	Zip	city
1	Ram	2055/5/5	01	Kathmandu
2	Sita	2055/2/1	01	Kathmandu
3	Gita	2040/3/1	02	patan
4	Hari	2045/06/06	03	Dharan

Fig:- 3

In above fig:-3

St_id is a primary key. all other attributes are dependent on St_id, so it is in 2NF but, city, non-prime attribute is dependent on Zip, another non prime attribute. so it is not in 3NF.

Now, decompose above table as below:-

St_id	St_name	DoB	Zip
1	Ram	2055/5/5	01
2	Sita	2055/2/1	01
3	Gita	2040/3/1	02
4	Hari	2045/06/06	03

table 1

Zip	city
01	Kathmandu
02	patan
03	Dharan

table 2

BCNF

A relation is in BCNF if,
- it is in 3NF
- every determinant in that table is candidate key.

- Advance Form of 3NF also referred as 3.5NF.
- If table contain only one candidate key, 3NF and BCNF are equivalent.

eg:-

student	course	Teacher
Ram	DB	Shyam
Ram	e.com	sita
Gita	DB	Shyam
Gita	e.com	sita

Fig:- 4

Here
Key = {student, course}
F.D i) {student, course} → Teacher
Teacher → course

in ii) Teacher is not a candidate key but determines course, above figure 4 is not in BCNF.

Now, Decompose it as follows

Teacher	course
Shyam	DB
Sita	e.com

table 1

student	course
Ram	DB
Ram	e.com
Gita	DB
Gita	e.com

table 2

Multivalued Dependency

- In a Relation R(A,B,C) if each A value has associated with it a set of B values and a set of C values such that B and C values are independent of each other, then the relation is said to have Multivalued dependency.

$A \twoheadrightarrow B,\ A \twoheadrightarrow C$

eg:-

Model	Year	color
M1	2007	Red
M1	2008	Blue
M2	2012	Red
M2	2016	Blue
M3	2018	Red
M3	2019	Blue

Here Model $\twoheadrightarrow$ year and Model $\twoheadrightarrow$ color.

#4NF

* A relation R is in 4NF if
  - it is is in 3NF or BCNF
  - R contains no Multivalued attributes.

eg:-

ename	pname	dname
Ram	x	John
Ram	y	sita
Ram	x	sita
Ram	y	John

Here ename $\twoheadrightarrow$ pname ename $\twoheadrightarrow$ dname.

∴ to convert it into 4NF, decompose into

ename	pname
Ram	x
Ram	y

table 1

ename	dname
Ram	John
Ram	sita

table 2

DeNormalization.

- It is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data.
- In relational database, denormalization is an approach to speed up read performance in which the administrator selectively adds back specific instances of redundant data after the data structure has been normalized.
- It is needed when multiple joins in some query can have negative impact on performance.
- denormalization should take place after a satisfactory level of normalization has taken and that any required constraints and/or rules have been created to deal with the inherent anamolies in design.

eg:- Database Normalization Example.

Employee(Name, project, task, office, floor, phone)

Name	project	task	office	floor	phone
Ram	xyz	T1	400	3	1400
Ram	ABC	T1	400	3	1400
Ram	ABC	T2	400	3	1400
Sita	xyz	T3	442	3	1442
Sita	ABC	T3	442	3	1442
Sita	pqR	T3	442	3	1442
Hari	xyz	T2	558	4	1558

Table 1

- Is Table1 in unnormalized form? ⇒ No
- Is it in 1NF? ⇒ yes

Ans:-

- Is it in 2NF? No, because with name only we can figure out office, floor and phone. i.e. all prime attributes are not dependent on fully primary key.

∴ Split it into two relations

a) Employee-project-task (name, project, task)

b) Employee-office-phone (name, office, floor, phone)

So, Tables are:-

Name	project	task
Ram	xyz	T1
Ram	ABC	T1
Ram	ABC	T2
Sita	xyz	T3
Sita	ABC	T3
Sita	pqR	T3
Hari	xyz	T2

Table a

Name	office	floor	phone
Ram	400	3	1400
Sita	442	3	1442
Hari	558	4	1558

Table b

Is table a in 3NF? yes

Is table b in 3NF? No, office can determine phone i.e. non prime attribute can determine another non prime attribute.

∴ Split it in two tables.

c) Employee-office (name, office floor)

d) Employee-phone (office, phone)

Name	office	floor
Ram	400	3
Sita	442	3
Hari	558	4

Table c

office	phone
400	1400
442	1442
558	1558

Table d.

- Are all tables table (b), (c), (d) in BCNF? Yes.
- Are all tables in 4NF?
  only table (c) and (d) are in 4NF
  Table a, b are not in 4NF because of Multivalued Attribute.

∴ Split into two Relations

e) employee_project (name, project)
f) employee_task (name, task)

Name	project
Ram	XYZ
sita	XYZ
Hari	XYZ
Ram	ABC
sita	PQR
Sita	ABC

Table e)

Name	task
Ram	T1
Ram	T2
Sita	T3
Hari	T2

Table f)

Refrential integrity :-

- It refers to the accuracy and consistency of data within a relationship.