Chapter 4: Relational Database Design

Functional Dependencies:

- FDs are the fundamental to the process of normalization.
- FD plays a key role in differentiating good database design from bad database design.
- FD describes the relationship between attributes in a table.
- For attributes X and Y in relation R, functional dependencies between X and Y is shown as

  X → Y

  Here, X is determinant and Y is functionally dependent on X.
- Example1: employee (eid, name, job, salary)

  Here, following FDs exists:
  - eid → name
  - eid → job
  - eid → salary

  But not like:
  - name → eid
  - name → job
  - salary → job
- Example2: emp_pro_info (eid, projectNo, hours, empName, proName, proLocation)

  Here, following FDs exists:
  - eid → empName
  - projectNo → {proName, proLocation}
  - {eid, projectNo} → hours
- Types of Functional Dependency:
  1. Full Functional Dependency
  2. Partial Functional Dependency
  3. Transitive Dependency
  4. Trivial Dependency
  5. Non-Trivial Dependency

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 occurs when it is already functional dependency and the set of attributes on the left of FD can't be reduced any farther.

Order#	Line#	Qty	Price
A01	L01	10	200
A02	L01	20	300
A02	L02	20	350
A03	L01	15	450

Here, FDs *{Order#, Line#}* → *Qty* and *{Order#, Line#}* →*Price* are full functional dependency.

II. Partial Functional Dependency

- A functional dependency X → Y is partial dependency if some attribute A from X can be removed from X and the dependency still holds.
- E.g. in functional dependency *{empid, phone}* → *name*, if we remove attribute phone from it, dependency still holds i.e. *empid* → *name*.

III. Transitive Dependency

- It occurs when there is an indirect relationship that causes a functional dependency.
- If X → Y and Y → Z then transitive dependency exists is X → Z.
- E.g. if we FDs empid → job and job → salary, we can say FD empid → salary exists.

IV. Trivial Dependency

- It occurs when functional dependency of an attribute is described on a collection of attributes that includes the original attribute.
- FD is trivial if R.H.S. is a subset of L.H.S.
- E.g. {name, SSN} → SSN

V. Non-Trivial Dependency

- It is one that is not trivial.
- E.g. {S#, P#} → {S#, Qty}

Axioms or Rules of Inference for Functional Dependency:

1. **Reflexivity Rule:**
   If Y is subset of X then X → Y holds.
2. **Augmentation Rule:**
   If X → Y holds and Z is a set of attributes then ZX → ZY holds.
3. **Transitivity Rule:**
   If X → Y holds and Y → Z holds then X → Z holds.

4. **Union Rule:**
   If X → Y holds and X → Z holds then X → YZ holds.
5. **Decomposition Rule:**
   If X → YZ holds then X → Y holds and X → Z holds.
6. **Pseudo Transitivity Rule:**
   If X → Y holds and ZY → P then XZ → P holds.
7. **Self Determination:**
   A → A

e.g. 1

R = (A, B, C, G, H, I) and F = {A → B, A → C, CG → H, CG → I, B → H}

Then we can have

- A → H (transitivity rule: A → B, B→ H)
- CG →HI (union rule: CG → H, CG → I)
- AG →I (pseudo transitivity rule: A → C, CG → I)

e.g. 2

F = {A → B, C → D, C subset of B}

Prove or Disprove A → C

Here, B → C (reflexivity rule)

So, A → C exists because of transitivity rule i.e. A → B, B → C

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 functional dependency F denoted by F^+ is a set of functional dependencies that are implied by functional dependencies in F.
- Axioms are useful to find F^+ of F.
- E.g.1 R = (A, B, C, D) and F = {A → B, A→ C, BC → D}. Find F^+.
- E.g.2 R = (A, B, C, G, H, I) and F = {A → B, A→ C, CG → H, CG → I, B → H}. Find F^+.

Closure of attribute sets

- Closure of attribute set is the set of attributes which are functionally dependent on the attribute set.
- Given a set of attributes A1………….An, the closure $\{A1……….An\}^+$ is the set of attributes B such that A1……………An → B
- E.g. if we have:
  Name → Color
  Category → Department
  {Color, Category} → Price

  Then, $\{Name\}^+$ = {Name, Color}, $\{Color\}^+$ = {Color}, $\{Name, Category\}^+$ = {Name, Category, Color, Department, Price}
- ***Algorithm to compute α^+: the closure of α under F***
  Result = α;
  While (Changes to Result) do
  For each functional dependency β → γ in F do
  Begin
  If β is subset of Result then
  Result = result U γ;
  End

- E.g.1 R = (A, B, C, G, H, I) and F = {A → B, A → C, CG → H, CG → I, B → H}, Compute $(AG)^+$.
- E.g.2 R = (A, B, C, D, E, F) and F = {A → BC, E → CF, B → E, CD → EF}, Compute $(AB)^+$.

Extraneous attribute

- An attribute of functional dependency is extraneous if we can remove it without changing the closure of the set of functional dependencies.
- Consider F as set of functional dependencies and functional dependency α → β in F.
- ***To test if attribute X is extraneous in α***
  - *Compute $(\{\alpha\} - X)^+$ using dependencies in F.*
  - *Check that $(\{\alpha\} - X)^+$ contains X; if it does , X is extraneous.*

- ***To test if attribute X is extraneous in β***
    - *Compute α^+ using only dependencies in F', here F' = (F - { $\alpha \rightarrow \beta$ }) U $\alpha \rightarrow (\beta - X)$*
    - *Check that α^+ contains X; if it does, X is extraneous.*
- E.g.1 F = { A → B, B → C, AC → D}
- E.g.2 F = { A → B, B → C, A → CD}

Canonical Cover of FD

- Canonical Cover of functional dependency set F, denoted by F_C is a set of dependencies such that F logically implies all dependencies in F.
- F_C must have following properties
    - *No functional dependency in F_C contains an extraneous attribute.*
    - *Each left side of functional dependency in F_C is unique i.e. there is no two dependencies $X_1 \rightarrow Y_1$ and $X_2 \rightarrow Y_2$ in F_C such that $X_1 = X_2$*
- To compute canonical cover

  F_C = F

  Repeat

      Use union rule to replace any dependencies

          $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ then $\alpha_1 \rightarrow \beta_1 \beta_2$

      Find a FD $\alpha \rightarrow \beta$ in F_C with extraneous attribute either in α or in β

      If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

  Until F_C do not change
- E.g. F = {A→BC, B→C, A→B, AB→C} find F_C.

  Solution:

      F_C = F = {A→BC, B→C, A→B, AB→C}

      For A→BC, A→B, and using union rule: A→BC

      In FD, AB→C, A is extraneous, hence: B→C

      In FD, A→BC, C is extraneous, hence: A→B

  So, Canonical Cover F_C = {B→C, A→B}

Determining Candidate key

- Compute closure of each attributes.
- Any attribute is called candidate key of any relation if attribute closure is equal to the relation.
- Prime attribute and Non-Prime attribute
    - Attributes that are part of candidate key are called prime attribute.
    - Attributes that are not part of candidate key are called non-prime attribute.
- E.g. R = (A, B, C, D), F = {AB→C, C→D, D→A}. List all candidate keys, prime and non-prime attribute.

  Solution:

  A^+ = A; A is not a candidate key

  B^+ = B; B is not a candidate key

  C^+ = CDA; C is not a candidate key

D^+ = DA; D is not a candidate key
$(AB)^+$ = ABCD; AB is candidate key
$(AC)^+$ = ACD; AC is not candidate key
$(AD)^+$ = AD; AD is not candidate key
$(BC)^+$ = BCDA; BC is candidate key
$(BD)^+$ = BDAC; BD is candidate key
$(CD)^+$ = CDA; CD is not candidate key

Hence, candidate key = {AB, BC, BD}, prime attributes = {A, B, C, D}

Decomposition

- Refers to the breaking down of one table into multiple tables.
- Desirable properties of decomposition
  - Attribute prevention
  - Dependency prevention
  - Avoid redundancy
- Types
  - Lossy Decomposition
    - Loss of information due to decomposition is called lossy decomposition.
    - It is not acceptable.
  - Lossless or Non-Loss Decomposition
    - Refers to decomposition where all information is preserved.

Normalization

- Database normalization is a technique of organizing data in the database.
- It is the process of removing data redundancy and undesirable characteristics like insertion, update and deletion anamolies.
- Mainly two purposes of normalization are
  - Eliminating redundant data
  - Ensuring data dependencies make sense
- It involves dividing table into two or more tables and defining relationship between the tables.
- Normal Forms:
  - Represents guidelines for record design.
  - Edgar F. Codd originally established three normal forms 1NF, 2NF, 3NF
  - Other forms are BCNF(Boyce-Codd Normal Form), 4NF, 5NF, DKNF(Domain Key Normal Form)
- **UnNormalized Form:**
  - It contains one or more repeating groups or each row may contain multiple set of values for some columns.
  - E.g.

CID	CNAME	CADDRESS	CPHONE
C101	Ram	Kathmandu	9841000000, 9851000000
C102	Sita	Pokhara	9803000001
C103	Gita	Patan	9841010100, 9843000102

Fig. 1 : UNF

- **First Normal Form (1NF):**
  - Values of each attribute are atomic.
  - No composite values.
  - All entries in any column must be of same kind.
  - Each column must have unique name.
  - No two rows are identical.

CID	CNAME	CADDRESS	CPHONE
C101	Ram	Kathmandu	9841000000
C101	Ram	Kathmandu	9851000000
C102	Sita	Pokhara	9803000001
C103	Gita	Patan	9841010100
C103	Gita	Patan	9843000102

Fig.1.1

Fig.1.1 is in 1NF.

Roll	Name	Courses
1	Ram	C, JAVA
2	Sita	DBMS, JAVA
3	Hari	DBMS, MATH

Fig.2

Fig.2 is not in 1NF because value of attribute course is not atomic. Following Fig.3 is 1NF of Fig.2

Roll	Name	Courses
1	Ram	C
1	Ram	JAVA
2	Sita	DBMS
2	Sita	JAVA
3	Hari	DBMS
3	Hari	MATH

Fig.3

- **Second Normal Form (2NF):**
  - A table or relation is in 2NF if and only if it is in 1NF and all attributes dependent on full primary key.

PersonID	**ProjectID**	**PersonName**	**ProjectName**	**Phone**
1	1	Ram	Database	9841202020
2	1	Shyam	Database	9851000000
1	2	Ram	Web	9841202020
2	2	Shyam	Web	9851000000

Fig.4

Here, Fig.4 is in 1NF but not in 2NF because all attributes do not depend on full primary key {PersonID, ProjectID}. So break it down into following tables:

PersonID	**PersonName**	**Phone**
1	Ram	9841202020
2	Shyam	9851000000

Fig. 5

ProjectID	**ProjectName**
1	Database
2	Web

Fig.6

PersonID	**ProjectID**
1	1
1	2
2	1
2	2

Fig.7

Here, Fig. 5, Fig. 6 and Fig. 7 are in 2NF.

- **Third Normal Form (3NF):**
    - A table or relation is in 3NF if and only if it is in 2NF and there is no transitive dependency i.e. there should not be case that non-prime attribute is determined by another non-prime attribute.

EID	**Name**	**Job**	**City**	**ZipCode**
1	Ram	Programmer	Kathmandu	44600
2	Sita	Manager	Pokhara	33700
3	Hari	Programmer	Pokhara	33700
4	Gita	Analyst	Chitwan	44200
5	Shyam	Programmer	Kathmandu	44600

Fig. 8

Here, Fig. 8 is in 2NF but not in 3NF because non-prime attribute City is dependent on another non-prime attribute ZipCode i.e transitive dependency occurs.Now, break down it into followings:

EID	Name	Job	ZipCode
1	Ram	Programmer	44600
2	Sita	Manager	33700
3	Hari	Programmer	33700
4	Gita	Analyst	44200
5	Shyam	Programmer	44600

Fig. 9

ZipCode	City
44600	Kathmandu
33700	Pokhara
44200	Chitwan

Fig. 10

Here, Fig. 9 and Fig. 10 are in 3NF.

- **BCNF (Boyce Codd Normal Form):**
  - A table is in BCNF if it is already in 3NF and every determinant in that table is a candidate key.
  - Advance form of 3NF often referred as 3.5NF.
  - If table contain only one candidate key, 3NF and BCNF are equivalent.

Student	Course	Teacher
Ram	Database	Hari
Ram	JAVA	Gita
Sita	Database	Hari
Sita	JAVA	Gita

Fig. 11

Here, Key = {Student, Course}

Functional Dependency = {Student, Course} → Teacher ; Teacher → Course

But Teacher is not candidate key and still determines Course, Hence above table is not in BCNF.

So, above fig is break down into followings:

Teacher	**Course**
Hari	Database
Gita	Java

Fig. 12

Student	**Course**
Ram	Database
Ram	JAVA
Sita	Database
Sita	JAVA

Fig. 13

Course
Database
JAVA

Fig. 14

- **Multivalued Dependency:**
  - In relation R(A, B, C) if each A values 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 ->> B, A ->> C
  - E.g.

Model	MakeYear	Color
M1	2009	RED
M1	2010	BLUE
M2	2018	RED
M2	2019	BLUE
M3	2020	RED
M3	2021	BLUE

Fig. 15

Here, for each model, there are multiple values of MakeYear and Color. i.e.

Model ->> MakeYear, Model ->> Color

- **Fourth Normal Form (4NF):**
  - A table is in 4NF if it is already in BCNF or 3NF and if it contains no multivalued attributes.

- E.g.

Ename	Pname	Dname
Ram	X	Gita
Ram	Y	Shyam
Ram	X	Shyam
Ram	Y	Gita

Fig. 16

Here, Ename ->> Pname and Ename ->> Dname. Hence, it is not in 4NF. So, decompose it into followings:

Ename	Pname
Ram	X
Ram	Y

Fig. 17

Ename	Dname
Ram	Gita
Ram	Shyam

Fig. 18

- **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 same 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.

Question: Normalize following example.

Employee (Name, Project, Task, Office, Floor, Phone)

Name	Project	Task	Office	Floor	Phone
Ram	XYZ	T1	IT	3	1400
Ram	XYZ	T2	IT	3	1400
Ram	ABC	T1	IT	3	1400
Ram	ABC	T2	IT	3	1400
Sita	XYZ	T3	Finance	3	1442
Sita	ABC	T3	Finance	3	1442
Sita	PQR	T3	Finance	3	1442
Hari	XYZ	T2	HR	4	1558