CHAPTER – 10

FUNCTIONAL DEPENDENCY

Prepared By: Prof. Kinjal Acharya
Assistant Professor
DDU, Nadiad

INTRODUCTION

- A functional dependency is an **association between two attributes of the same relational database table.**
- An attribute in a relational model is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent.
- One of the attributes is called the **determinant** and the other attribute is called the **determined**.
- For each value of the determinant there is associated one and only one value of the determined.
- Basically, a functional dependency (FD) is a **many-to-one relationship** from one set of attributes to another within a given relvar. In the case of the shipments relvar SP, for example, there is a functional dependency from the set of attributes {S#, P#} to the set of attributes {QTY}.
- What this means in that within any relation that happens to be legal value for relvar SP:

  a) For any given value for the pair of attributes S# and P#, there is just one corresponding value of attribute QTY , but

  b) Many distinct values of the pair of attributes S# and P# can have the same corresponding value for attribute QTY (in general).
- If A is the determinant and B is the determined then we say that *A functionally determines B and graphically represent this as* $A \rightarrow B$.

Example

The following table illustrates A ⟹ B:

A	B
1	1
2	4
3	9
4	16
2	4
3	9

Since for each value of A there is associated one and only one value of B. Hence A→B.

Also, {S#, P#} → QTY

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Example

The following table illustrates that A does not functionally determine B:

A	B
1	1
2	4
3	9
4	16
3	10

BASIC DEFINITIONS

- **FUNCTIONAL DEPENDENCY:** An attribute in a relational model is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent.
- Let r be a relation, and let X and Y be arbitrary subsets of the set of attributes of r. Then we say that Y is functionally dependent on X – in symbols, X → Y (read "X functionally determines Y", or simply "X arrow Y") – if and only if each X value in r has associated with it precisely one Y value in r. In other words, whenever two tuples of r agree on their X value, they also agree on their Y value.
- **For example,** We make use of a revised version of the shipments relvar, one that includes, in addition to the usual attributes S#, p# and QTY, an attribute CITY, representing the city for the relevant supplier. We will refer to this relvar as SCP. A possible value for relvar SCP is shown in figure:

The relation shown in figure satisfies FD { S# } → { CITY } Because every tuple of that relation with a given S# value also has the same CITY value.
Indeed, it also satisfies several more FDs, the following among them :

{ S#, P# } → { QTY }
{ S#, P# } → { CITY }
{ S#, P# } → { CITY, QTY }
{ S#, P# } → { S# }
{ S#, P# } → { S#, P#, CITY, QTY }
{ S# } → { QTY }

S#	CITY	P#	QTY
S1	London	P1	100
S1	London	P2	100
S2	Paris	P1	200
S2	Paris	P2	200
S3	Paris	P2	300
S4	London	P2	400
S4	London	P4	400
S4	London	P5	400

BASIC DEFINITIONS

- The left and right hand sides of an FD are sometimes called the **DETERMINANT** and the **DEPENDENT**, respectively.
- As the definition indicates, the determinant and dependent are both sets of attributes.
- When such a set contains just one attribute, i.e., when it is a singleton set – we often drop the set braces and write just S# → CITY.
- In the case of SCP, for example the FD S# → CITY holds for all possible values of SCP, because, at any given time, a given suppliers has precisely one corresponding city, and so any two tuples appearing in SCP at the same time with the same supplier number must necessarily have the same city as well.
- Indeed, the fact that this FD holds "for the time" (i.e., for all possible values of SCP) is an integrity constraint on relvar SCP.

- **TRIVIAL FUNCTIONAL DEPENDENCIES:** The dependency of an attribute on a set of attributes is known as **trivial functional dependency** if the set of attributes includes that attribute. **Symbolically**: A ->B is trivial functional dependency if B is a subset of A. A->A and B->B are also trivial FDs.
- **For example**: Consider a table with two columns Student_id and Student_Name.
  {Student_Id, Student_Name} -> Student_Id is a trivial functional dependency as Student_Id is a subset of
  {Student_Id, Student_Name}.

- **NON TRIVIAL FUNCTIONAL DEPENDENCIES:** If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called **non trivial Functional dependency**.
- **For example**: An employee table with three attributes: emp_id, emp_name, emp_address.
  The following functional dependencies are non-trivial:
  emp_id -> emp_name (emp_name is not a subset of emp_id) and emp_id -> emp_address (emp_address is not a subset of emp_id)

CLOSURE OF A SET OF DEPENDENCIES

- Some FDs might imply others.
- As a simple example, the FD { S# , P# } → { CITY , QTY }.
- Implies both of the following :
- { S# , P# } → CITY and { S# , P# } → QTY
- The set of all FDs that are implied by a given set S of FDs is called closure of S, written as S+.
- Let A, B, and C be arbitrary subsets of the set of attributes of the given relvar R, following Inference Rules (Armstrong axioms) can be applied on them to find its closure of set of dependencies:

1. Reflexivity: If B is a subset of A, then A → B.
2. Augmentation: If A → B, then AC □ BC.
3. Transitivity: If A → B and B → C, then A → C.
4. Self-determination: A → A.
5. Decomposition: If A → BC , then A → B and A → C.
6. Union: If A → B and A → C, then A → BC.
7. Composition: If A → B and C → D, then AC → BD.
8. General Unification Theorem: If A → B and C → D, then A U (C – B) → BD (where 'U' is union and '-' is set deference).

GENERAL UNIFICATION THEOREM

If A $\rightarrow$ B and C $\rightarrow$ D, then A U (C – B)$\rightarrow$ BD

Proof:-

1. A $\rightarrow$ B (given)
2. C $\rightarrow$ D (given)
3. A $\rightarrow$ B ∩ C (joint dependence, 1)
4. C – B $\rightarrow$ C – B (self-determination)
5. A U (C – B) $\rightarrow$ (B ∩ C) U (C – B) (composition, 3, 4)
6. A U (C – B) $\rightarrow$ C (simplifying 5)
7. A U (C – B) $\rightarrow$ D (transitivity, 6,2)
8. A U (C – B) $\rightarrow$ B U D (composition, 1, 7)

CLOSURE OF A SET OF DEPENDENCIES: EXAMPLE

- Suppose we are given relvar R with attributes A, B, C, D, E, F and the FDs:

  A → BC

  B → E

  CD → EF

- Take A as employee number, B as department number, C as manager's employee number, D as project number for a project directed by that manager (unique within manager), E as department name, and F as percentage of time allocated by the specified manager to the specified project.

- We now show that the FD AD → F holds for R and is thus a member of the closure of the given set:

  1. A → BC (given)
  2. A → C (1, decomposition)
  3. AD → CD (2, augmentation)
  4. CD → EF (given)
  5. AD → EF (3 and 4, transitivity)
  6. AD → F (5, decomposition)

- Hence in step 6, it is proved that AD → F holds true for given set of FDs.

CLOSURE OF A SET OF ATTRIBUTES

- For a given a relvar, a set Z of attributes of R, and a set S of FDs that hold for R, we can determine the set of all attributes of R that are functionally dependent on Z – the so called closure Z+ of Z under S.

- **Algorithm to compute the closure Z+ of Z under S:**

```
CLOSURE[Z,S] := Z;
Do “forever”
    For each FD X-->Y in S
        Do
            If X ≤ CLOSURE[Z,S] then
                CLOSURE[Z,S]  := CLOSURE[Z,S] U Y;
        End;
    If CLOSURE[Z,S] did not change on this iteration then leave loop;
End;
```

CLOSURE OF A SET OF ATTRIBUTES: EXAMPLE

Suppose we are given relvar R with attributes A, B, C, D, E, F, and FDs

A → BC

E → CF

B → E

CD → EF

- We now compute the **closure {A, B}+** of the set of attributes {A, B} under this set of FDs:

1. We initialize the result CLOSURE[Z,S] to {A,B}.
2. On the first iteration (for the FD A → BC), we find that the left-hand side is indeed a subset of CLOSURE[Z, S] as computed so far, so we add attributes (B and) C to the result. CLOSURE [Z, S] is now the set { A, B, C}.
3. On the second iteration (for the FD E → CF), we find that the left-hand side is not a subset of the result as computed so far, which thus remains unchanged.
4. On the third iteration (for the FD B → E), we add E to CLOSURE [Z, S], which now has the value {A,B,C,D}.
5. On the fourth iteration (for the FD CD → EF), CLOSURE [Z,S] remains unchanged.
6. Now we go round the inner loop four times again. On the first iteration, the result does not change; on the second, it expands to { A,B,C,E,F}; on the third and fourth, it does not change.
7. CLOSURE [Z,S] with { A,B }+ = {A,B,C,E,F}.

CLOSURE OF A SET OF ATTRIBUTES: EXAMPLE

1) Relation R with attributes (ABCDE) is given. The FD's are as follows: - {AB->C, B->D, C->E, D->A}

- **Compute B^+, $(C,D)^+$, $(B,C)^+$**

B^+	(C,D)+	(B,C)+
1. B 2. BD (B→D) 3. BDA (D→A) 4. BDAC (AB→C) 5. BDACE (C→E)	1. CD 2. CDE (C→E) 3. CDEA (D→A)	1. BC 2. BCD (B→D) 3. BCDE (C→E) 4. BCDEA (D→A)

2) Relation EMP with attributes (E-ID,E-Name,E-City,E-state) is given with FD's {E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE}. Compute $(\text{E-ID})^+$, $(\text{E-NAME})^+$, $(\text{E-CITY})^+$

E_ID^+	$E\text{-}NAME^+$	$E\text{-}CITY^+$
1. E-ID 2. E-ID,E-NAME (E-ID→E-NAME) 3. E-ID,E-NAME,E_CITY (E-ID→E_CITY) 4. E-ID,E-NAME, E_CITY , E-STATE(E_CITY→E-STATE)	1 E_NAME	1. E_CITY 2. E_CITY,E_STATE ((E_CITY→E-STATE)

CLOSURE OF A SET OF ATTRIBUTES: EXERSICE

- Relation R with attributes (ABCDEFG) is given with FD's A→B, BC→DE and AEG→G. Compute AC^{+}.
- Relation R with attributes (ABCDE) is given with FD's A→BC, CD→E , B→D and E→A. Compute B^{+}.
- Relation R with attributes (ABCDEF) is given with FD's AB→C, BC→AD, D→E and CF→B. Compute AB^{+}.
- Relation R with attributes (ABCDEFGH) is given with FD's A→BC, CD→E, E→C, D→AEH, ABH→BD and DH→BC. Compute BCD^{+}.
- Relation R with attributes (ABCDEF) is given with FD's A→BC, E→CF, B→E and CD→EF. Compute AB^{+}

SUPER_KEY AND CANDIDATE KEY:

- For Example, the EMPLOYEE relation has given FD set:

{E-ID->E-NAME, E-ID->E-CITY, E-ID->E-STATE, E-CITY->E-STATE}

- Let us calculate attribute closure of different set of attributes:

(E-ID)+ = {E-ID, E-NAME,E-CITY,E-STATE}, (E-ID,E-NAME)+ = {E-ID, E-NAME,E-CITY,E-STATE},

(E-ID,E-CITY)+ = {E-ID, E-NAME,E-CITY,E-STATE}, (E-ID,E-STATE)+ = {E-ID, E-NAME,E-CITY,E-STATE},

(E-ID,E-CITY,E-STATE)+ = {E-ID, E-NAME,E-CITY,E-STATE}, (E-NAME)+ = {E-NAME},

(E-CITY)+ = {E-CITY,E-STATE}

- **Super Key** is **set of attributes** of a relation which can be used to identify a tuple uniquely.
- If Z^+ consists of all the attributes of R, Z is said to be the super key of R and a candidate key is an irreducible super key.
- As (E-ID)+, (E-ID, E-NAME)+, (E-ID, E-CITY)+, (E-ID, E-STATE)+, (E-ID, E-CITY, E-STATE)+ give set of all attributes of relation EMPLOYEE. So all of these are super keys of relation.
- The **minimal set of attributes** whose attribute closure is set of all attributes of relation is called **candidate key** of relation.
- As shown above, (E-ID)+ is set of all attributes of relation and it is minimal. So E-ID will be candidate key.
- On the other hand (E-ID, E-NAME)+ also is set of all attributes but it is not minimal because its subset (E-ID)+ is equal to set of all attributes.
- So (E-ID, E-NAME) is not a candidate key.

IRREDUCIBLE SETS OF DEPEDENCIES

A set S of FDs is irreducible if and only if it satisfies the following three properties:

1. The right-hand side (the dependent) of every FD in S involves just one attribute (i.e., it is a singleton set).
2. The left-hand side (the determinant) of every FD in S is irreducible in turn-meaning that no attribute can be discarded from the determinant without changing the closure S+ (i.e., without converting S into some set not equivalent to S). We will say that such an FD is left-irreducible.
3. No FD in S can be discarded from S without changing the closure S+ (i.e., without converting S into some set not equivalent to S).

- For example, consider the familiar parts relvar P. The following FDs (among others) hold for that relvar :

  P# → PNAME

  P# → COLOR

  P# → WEIGHT

  P# → CITY

- The right-hand side is a single attribute in each case, the left-hand side is obviously irreducible in turn, and none of the FDs can be discarded without changing the closure (i.e., without losing some information).
- So, this set of FDs is irreducible.

IRREDUCIBLE SETS OF DEPEDENCIES: EXAMPLE

Suppose a relvar R with attributes A, B, C, D and FDs:

A → BC

B → C

AB → C

AC → D

Compute an irreducible set of FDs that is equivalent to the given set.

1. The first step is to rewrite the FDs such that each has a singleton right-hand side :

a. A → B

b. A → C

c. B → C

d. AB → C

f. AC → D

IRREDUCIBLE SETS OF DEPEDENCIES: EXAMPLE

2. Next, attribute C can be eliminated from the left-hand side of the FD AC → D, because
    a. A →C, so A → AC by augmentation,
    b. AC → D (Given)
    c. A → D by transitivity; thus the C on the left – hand side of AC → D is redundant.

3. Next, the FD AB → C can be eliminated, because again we have
    a. A →C, so
    b. AB → CB by augmentation, so
    c. AB → C and AB →B by decomposition.

So, AB→C is equivalent to A→C and so it can be eliminated.

4. Finally, the FD A → C is implied by the FDs A → B and B → C, so it can also be eliminated.

We are left with :

A → B
B → C
A → D

This set is irreducible.

IRREDUCIBLE SETS OF DEPEDENCIES: EXCERSICE

1: Is following set of FDs irreducible?

P# → { PNAME, COLOR }

P# → WEIGHT

P# → CITY

2: Is following set of FDs irreducible?

{ P#, PNAME } → COLOR

P# → PNAME

P# → WEIGHT

P# → CITY

3: Is following set of FDs irreducible?

P# → P#

P# → PNAME

P# → COLOR

P# → WEIGHT

P# → CITY