Lecture 7

Functional Dependencies

Dr. Vandana Kushwaha

Department of Computer Science

Institute of Science, BHU, Varanasi

Introduction

- A **Functional dependency** is a **constraint** between **two sets** of **attributes** from the **database.**
- Consider the whole database as being described by a single **universal relation schema *R* = {*A1, A2*, ... , *An*}** has ***n*** attributes ***A1, A2, ...,An*.**
- A **functional dependency**, denoted by ***X* → *Y*,** between **two sets** of attributes ***X*** and ***Y*** that are **subsets of *R*** specifies a ***constraint*** on the possible **tuples** that can form a **relation state *r*** of ***R*.**
- **Constraint** : *For all pairs of **tuples t1** and **t2** in **r** that have **t1[X] = t2[X],** they must also have **t1[Y] = t2[Y].***
- There is a **functional dependency from *X* to *Y*,** or that ***Y*** is **functionally dependent** on ***X*.**
- The abbreviation for **functional dependency** is **FD** or **f.d.**
- The set of attributes ***X*** is called the **left-hand side** of the **FD,** and ***Y*** is called the **right-hand side**.

Functional Dependency

- Suppose **R** is a relation, **X,Y** are attributes over **R**, **t1,t2** are tuples in the relation **R:**

	X	Y
t1	a1	b1
t2	a2	b2

- The **functional dependency X→Y** holds in **R** if and only if
  - **(t1.X=t2.X) ═► (t1.Y=t2.Y)**
- In above **relation R** there is **no such tuple** for which the condition **t1.X=t2.X** is **true** thus the **functional dependency X→Y** holds in **R.**

Exercise 1

- Suppose you have the following table:

A	B
a1	b1
a2	b1

  – **A→B** is **true** (B is Functionally Dependent on A) because there is no such pair of tuples for which **(t1.A=t2.A) .**

Exercise 2

- Suppose you have the following table:

A	B
a1	b1
a1	b1

  – **A→B** is **true** (B is Functionally Dependent on A) because there is a pair of tuples for which **(t1.A=t2.A)** ═▶ **(t1.B=t2.B)**

Exercise 3

- Suppose you have the following table:

A	B
a1	b1
a1	b2

  – **A→B** is **False** (B is Not FD on A) because there is a pair of tuples for which **(t1.A = t2.A)** but **(t1.B ≠ t2.B)**

Exercise 4

- Suppose you have the following table:

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

- List all **functional dependencies** satisfied by the relation of above Figure.
- **A→B** is **True because** there are pair of tuples for which:
  - **(t1.A=t2.A) ═► (t1.B=t2.B)** and **(t3.A=t4.A) ═► (t3.B=t4.B)**
- **C→B** is **True because** there are pair of tuples for which:
  - **(t1.C=t3.C) ═► (t1.B=t3.B)** and
  - for tuples **t2** and **t4** no such pair exist for which the values of **t2.C** and **t4.C** match.
- A dependency that logically implied by **A→B** and **C→B** : **AC → B.**

Example 5

- Let us consider the **relation *r***, to see which **functional dependencies** are satisfied.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

- Observe that ***A* → *C*** is satisfied.
- There are **two tuples** that have an ***A* value** of ***a*1.**
- These tuples have the **same *C* value**—namely, ***c*1.**
- Similarly, the **two tuples** with an ***A* value** of ***a*2** have the **same *C* value, *c*2.**
- There are no other pairs of distinct tuples that have the same *A* value.
- The **functional dependency *C* → *A*** is **not satisfied**, however.
- As there is a pair of tuples ***t*1** and ***t*2** such that ***t*1[*C*] = *t*2[*C*], but *t*1[*A*] != *t*2[*A*].**

Example 6

- The **functional dependency** ***AB* → *D*.**
- Observe that there is **no pair of distinct tuples *t1*** and ***t2*** such that ***t1*[*AB*] = *t2*[*AB*].**
- Therefore, if ***t1*[*AB*] = *t2*[*AB*],** it must be that ***t1* = *t2*** and, thus, ***t1*[*D*] = *t2*[*D*].**
- So, ***r*** satisfies ***AB* → *D*.**

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	*b3*	c_2	d_3
a_3	b_3	c_2	d_4

Functional Dependency

- Consider the relation schema **EMP_PROJ**; the following **functional dependencies** should hold:

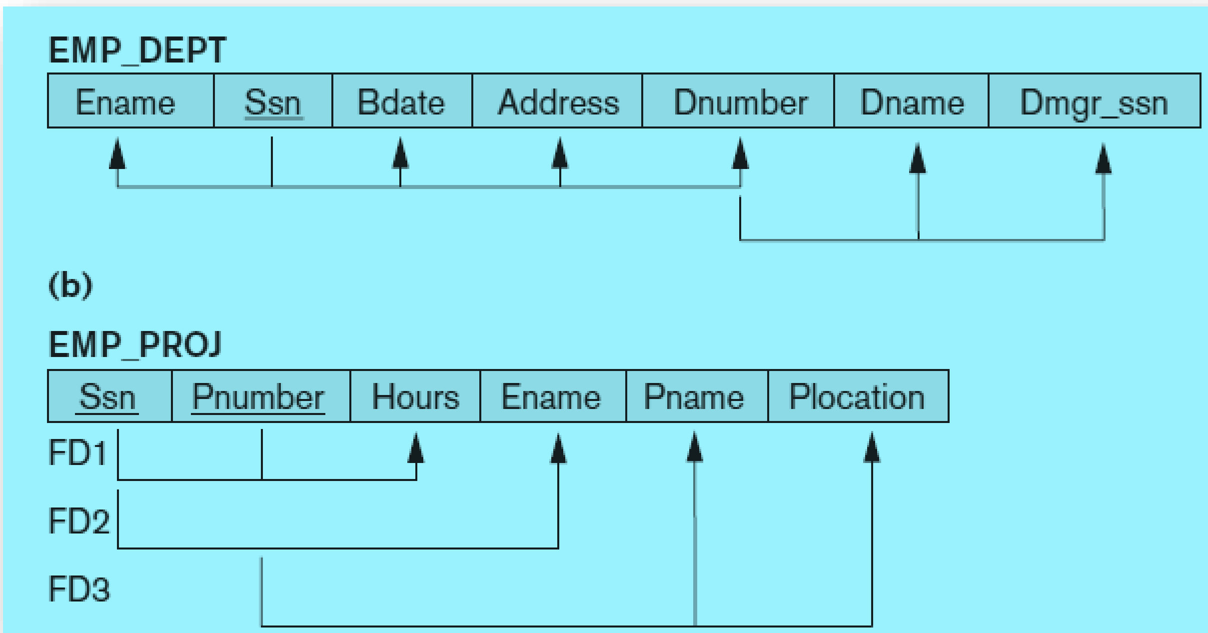
  a. Ssn → Ename

  b. Pnumber → {Pname, Plocation}

  c. {Ssn, Pnumber} → Hours

- These **functional dependencies** specify that:
  - **(a)** the value of an employee's Social Security number (**Ssn**) **uniquely determines** the **employee name** (Ename), alternatively, we say that **Ename** is **functionally determined by** (or functionally dependent on) **Ssn.**
  - **(b)** the value of a project's number (**Pnumber) uniquely determines** the **project name (Pname)** and **location (Plocation),** and
  - **(c)** a combination of **Ssn and Pnumber** values **uniquely determines** the **number of hours** the employee currently works on the project per week (Hours).

Diagrammatic Notation for FDs



Trivial Functional Dependency

- Some **functional dependencies** are said to be **trivial** because they are **satisfied by all relations.**
- For example, $A \rightarrow A$ is satisfied by **all relations** involving **attribute** A**.**
- We see that, for all tuples $t1$ and $t2$ such that $t1[A] = t2[A]$, it is the case that $t1[A] = t2[A]$.
- Similarly, $AB \rightarrow A$ is satisfied by **all relations** involving **attribute** A**.**
- In general, a **functional dependency** of the form $\alpha \rightarrow \beta$ is **trivial** if $\beta \subseteq \alpha$.

Inference Rules for FD

- The notation **F |=X → Y** to denote that the **functional dependency X→Y** is **inferred from** the **set of functional dependencies F.**
- The following **six rules IR1** through **IR6** are well-known **inference rules** for functional dependencies:

1. **IR1 (reflexive rule)1**: If $X \supseteq Y$, then $X \rightarrow Y$.
2. **IR2 (augmentation rule)2**: $\{X \rightarrow Y\} \mid= XZ \rightarrow YZ$.
3. **IR3 (transitive rule):** $\{X \rightarrow Y, Y \rightarrow Z\} \mid= X \rightarrow Z$.
4. **IR4 (decomposition, or projective, rule):** $\{X \rightarrow YZ\} \mid= X \rightarrow Y$.
5. **IR5 (union, or additive, rule):** $\{X \rightarrow Y, X \rightarrow Z\} \mid= X \rightarrow YZ$.
6. **IR6 (pseudotransitive rule):** $\{X \rightarrow Y, WY \rightarrow Z\} \mid= WX \rightarrow Z$.

- **Inference rules IR1** through **IR3** are known as **Armstrong's inference rules**.

Inferred Functional Dependency

- Let ***F denote*** the **set** of **functional dependencies** that are specified on **relation schema *R*.**
- Typically, the schema designer specifies the **functional dependencies** that are ***semantically obvious*.**
- However, numerous **other functional dependencies hold** in ***all*** legal relation instances among sets of attributes that can be **derived** from and satisfy the dependencies in ***F*.**
- Those **other dependencies** can **be *inferred* or *deduced*** from the **FDs in *F*.**
- **Example:** Dept_no → Mgr_ssn and Mgr_ssn→Mgr_phone **infer** Dept_no → Mgr_phone
- This is an **inferred FD** and need *not* be explicitly stated in addition to the two given FDs.

Closure of FD

- Formally, the **set of all dependencies** that **include *F*** as well as **all dependencies that can be inferred from *F*** is called the **closure of *F*;** it is denoted by F^+.
- **Example:** Consider the set of FDs **F:**
- *F* = {Ssn → {Ename, Bdate, Address, Dnumber}, Dnumber → {Dname, Dmgr_ssn} }
- Some of the **additional functional dependencies** that we can ***infer* from *F*** are the following:
  - Ssn → {Dname, Dmgr_ssn}
  - Ssn → Ssn
  - Dnumber → Dname
- Thus **F^+ = {**Ssn → {Ename, Bdate, Address, Dnumber}, Dnumber → {Dname, Dmgr_ssn}, Ssn → {Dname, Dmgr_ssn}, Ssn → Ssn, Dnumber → Dname **}**

Closure of FD

- **Example:**
- Let **R=**(A,B,C,G,H,I) is a relational schema with set of functional dependencies **F=**(A->B, A->C, CG->H, CG->I, B->H)
- Then members of $\mathbf{F^+}$ are:
  - **A->H,** since A->B & B->H hold **(transitivity rule)**
  - **CG->HI**, since CG->H & CG-> I **(union rule)**
  - **AG->I,** since A->C, CG->I (**pseudotransitivity rule)**
- Thus $\mathbf{F^+}$={A->B, A->C, CG->H, CG->I, B->H, A->H, CG->HI, AG->I}

Closure of attribute under a FD

- **Algorithm:** Determining $\boldsymbol{X^+}$, the **Closure of *X*** under *F*
- **Input:** A **set *F*** of **FDs** on a **relation schema R**, and a **set of attributes *X*,** which is a **subset of R.**

```
X+ := X;
repeat
      oldX+ := X+;
      for each functional dependency Y → Z in F do
            if X+ ⊇ Y then X+ := X+ ∪ Z;
until (X+ = oldX+);
```

Closure of Attribute under a FD

- Consider a **set *F*** of **functional dependencies** that should hold on **EMP_PROJ:**
- ***F*** = **{**Ssn → Ename, Pnumber → {Pname, Plocation}, {Ssn, Pnumber} → Hours**}**
  - **{Ssn}** $^+$ = Ssn
  - **{Ssn}** $^+$ = {Ssn, Ename}
  - **{Pnumber}** $^+$ = {Pnumber, Pname, Plocation}
  - **{Ssn, Pnumber}** $^+$ = {Ssn, Pnumber, Ename, Pname, Plocation, Hours}

$X^+ := X$;
repeat
    old$X^+ := X^+$;
    for each functional dependency $Y \rightarrow Z$ in F do
        if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
until ($X^+ = \text{old}X^+$);

Applications of Attribute Closure

- There are **several uses** of the **Attribute closure algorithm:**
  - To test if α is a **superkey**, we **compute $\alpha+$**, and **check if $\alpha+$ contains all attributes of R.**
  - We can check if a **functional dependency $\alpha \rightarrow \beta$ holds** (or, in other words, is in F^+), by checking if $\beta \subseteq \alpha+$.
    - That is, we **compute α^+** by using **attribute closure**, and then **check if it contains β.**

Example

- Consider the schema ***R* = (*A*, *B*, *C*, *G*, *H*, *I*)** and the **set *F*** of functional dependencies:
- ***F***=*{A → B, A → C, CG → H, CG → I, B → H}* and
- $\mathbf{F^+}$=(A->B, A->C, CG->H, CG->I, B->H, A->H, CG->HI, AG->I)
- Check **AG** is super key or not?
- **Solution:**
- $\mathbf{AG^+}$= AG
- Consider the FD A->B , here A ⊆ AG^+ thus AG^+ = AG ∪ B=AGB
- Consider the FD A->C, here A ⊆ AG^+ thus AG^+ = AGB ∪ C=AGBC
- Consider the FD CG->H, here CG ⊆ AG^+ thus AG^+ = AGBC ∪ H= AGBCH
- Consider the FD CG->I, here CG ⊆ AG^+ thus AG^+ = AGBCH ∪ I=AGBCHI
- Consider the FD B->H, here B ⊆ AG^+ thus $\mathbf{AG^+}$ = AGBCHI ∪ H=**AGBCHI**(old AG^+)
- As $\mathbf{AG^+}$ contains **all the attribute of R** thus **AG** is **a super key of R.**

Cover of Functional Dependencies

- A **set** of **functional dependencies** ***F*** is said to **cover** another **set** of **functional dependencies** ***E*** if **every FD in *E*** is also **in *F*⁺ i.e. E ⊆ *F*⁺**
- That is, if **every dependency in *E*** can be **inferred from *F***; alternatively, we can say that ***E* is covered by *F*.**
- We can **determine** whether ***F* covers *E*** by calculating ***X*⁺** *with respect to* ***F*** for each **FD *X→Y in E***, and then checking whether this ***X*⁺ includes** the **attributes in *Y*.**
- If this is the case **for *every* FD in *E***, then ***F* covers *E*.**

Equivalence of Sets of Functional Dependencies

- **Two sets** of **functional dependencies** ***E*** and ***F*** are **equivalent** if $E^+ = F^+$.
- Therefore, **equivalence** means that **every FD** in ***E*** can be **inferred from *F***, and **every FD in *F*** can be **inferred from *E*;**
- That is, ***E* is equivalent to *F*** if both the conditions**- *E* covers *F*** *and* ***F* covers *E*-** hold.
- **Example:** the following two sets of **FDs** are **equivalent:**
  - $\boldsymbol{F} = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
  - $\boldsymbol{G} = \{A \rightarrow CD, E \rightarrow AH\}$

  As **F covers G** and **G covers F** both are **true.**

Equivalence of Sets of Functional Dependencies

- **Exercise:** Show that the following two sets of FDs are **equivalent:**
  - $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
  - $G = \{A \rightarrow CD, E \rightarrow AH\}$.
- **Determining whether F covers G**
  - $A^+ = \{ A \ C \ D \}$ // closure of left side of $A \rightarrow CD$ w.r.t. set **F**
  - Here $CD \subseteq A^+$
  - $E^+ = \{ A \ C \ D \ E \ H \}$ // closure of left side of $E \rightarrow AH$ w.r.t. set **F**
  - Here $AH \subseteq E^+$
- **Thus F covers G.**

Equivalence of Sets of Functional Dependencies

- $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
- $G = \{A \rightarrow CD, E \rightarrow AH\}$.

- **Determining whether G covers F**
  - $A^+ = \{ A \ C \ D \}$ // closure of left side of $A \rightarrow C$ w.r.t. set **G**
  - Here $C \subseteq A^+$
  - $(AC)^+ = \{ A \ C \ D \}$ // closure of left side of $AC \rightarrow D$ w.r.t. set **G**
  - Here $D \subseteq (AC)^+$
  - $E^+ = \{ E \ A \ C \ D \ H \}$ // closure of left side of $E \rightarrow AD$ w.r.t. set **G**
  - Here $AD \subseteq E^+$
  - $E^+ = \{ E \ A \ C \ D \ H \}$ // closure of left side of $E \rightarrow H$ w.r.t. set **G**
  - Here $H \subseteq E^+$
  - **Thus G covers F.**
  - **It means both F and G are equivalent.**

Extraneous attributes

- An **attribute** of a **functional dependency** is said to be **extraneous** if we can **remove it** without **changing the closure** of the **set** of **functional dependencies**.
- **Formal definition of extraneous attributes:**
- Consider a **set *F* of functional dependencies** and the functional dependency $\boldsymbol{\alpha \rightarrow \beta}$ in ***F*.**
- ***Case 1* (extraneous attribute A is on *LHS of* $\boldsymbol{\alpha \rightarrow \beta}$ *)***:
  - To find if an **attribute A** in **$\boldsymbol{\alpha}$** is **extraneous** or not.
  - **Step 1**: Find $(\{\alpha\} - A)^+$ using the dependencies of F.
  - **Step 2**: If $(\{\alpha\} - A)^+$ contains all the attributes of **$\boldsymbol{\beta}$**, then **A is extraneous**.

Extraneous attributes

- ***Case 2* (extraneous attribute A is on *RHS of* $\alpha \rightarrow \beta$ *)*:**
  - To find if an **attribute A** in **β** is **extraneous** or not.
  - **Step 1**: Find **α^+** using the dependencies in **F'** where

    F' = (F – {$\alpha \rightarrow \beta$}) U { $\alpha \rightarrow (\beta - A)$ }.
  - **Step 2**: If **α^+ contains A**, then **A is extraneous**.

Example 1

- **Question:** Given a relational schema **R(P,Q,R)** and F = {P→Q, PQ→R}. Is Q extraneous in **PQ→R?**
- ***Step 1:*** *Find ({α} – A)⁺ using the dependencies of F.*
  - Here, α is PQ. So find $(PQ – Q)^+$, ie., P^+ (closure of P).
  - P^+= PQ since P→Q
  - P^+= PQR since PQ→R
  - Hence, the closure of P is PQR.
- ***Step 2:*** *If $(\{\alpha\} – A)^+$ contains all the attributes of β, then A is extraneous.*
  - $(PQ – Q)^+$ contains R. Hence, **Q is extraneous in PQ→R**.

Example 2

- **Question:** Given a relational schema R(P,Q,R) and F = {P→QR, Q→R}. Is R extraneous in P→QR?
- **Step 1:** Find α^+ using the dependencies in F' where

$$F' = (F - \{\alpha \rightarrow \beta\}) \cup \{ \alpha \rightarrow (\beta - A) \}.$$

- F' = ({P→QR, Q→R} – {P→QR}) U {P→(QR-R)}
- F' = ({Q→ R} U {P→Q}) = {Q→R, P→Q}
- Find **$(P)^+$** closure of P using the **F'** .
- P^+= PQ since P→Q
- P^+= PQR since Q→R
- **Step 2:** If α^+ contains A, then A is extraneous.
- **Since P^+** contains **R,** hence, **R** is **extraneous** in **P→QR.**

Finding Key Attribute

- **Algorithm :** Finding a **Key *K*** for ***R*** Given a set ***F*** of Functional Dependencies
- **Input:** A **relation *R*** and a set of functional dependencies ***F*** on the attributes of ***R*.**

Step 1. Set *K* := *R*.

Step 2. For each attribute ***A*** in ***K*** {compute $(K - A)^+$ with respect to ***F*;**

– if $(K - A)^+$ contains all the attributes in **R**, then set **K := K – {A} };**

Note: This algorithm **determines only one key** out of the possible **candidate keys** for **R;** the key returned depends on the order in which attributes are removed from R in step 2.

Finding Key Attribute

- Determine all **essential attributes** of the **given relation**.
- **Essential attributes** are those **attributes** which are **not present on RHS** of **any functional dependency.**
- **Essential attributes** are always a **part of** every **Candidate key.**
- Now, following **two cases** are possible-
- **Case-1:**
- If all **essential attributes together** can **determine** all remaining **non-essential attributes**, then-
  - The **combination of essential attributes** is the **candidate key.**
  - And it is the **only possible candidate key.**

Finding Key Attribute

- **Case-2:**
- If **all essential attributes together can not determine** all remaining **non-essential attributes**, then-
  - The set of **essential attributes** and **some non-essential attributes** will be the **candidate key(s).**
  - In this case, **multiple candidate keys** are possible.
  - To find the **candidate keys,** we check different **combinations** of **essential and non-essential attributes.**

Example 1

- **Ex1.** Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies- F={ C → F, E → A, EC → D, A → B } Find the candidate key(s). How many possible super keys are there?
- **Solution:**
- **Essential attributes** of the relation are- C and E.
- So, attributes **C and E** will definitely be a part of **every candidate key.**
- $\{ CE \}^+$

  = { C , E }

  = { C , E , F } (Using C → F)

  = { A , C , E , F } (Using E → A)

  = { A , C , D , E , F } (Using EC → D)

  = { A , B , C , D , E , F } (Using A → B)

  Thus **CE** is the **only possible candidate key** of the relation.

Example 2

- As **number of possible super keys** is $\mathbf{2^{(N - (size\ of\ candidate\ key))}}$
- Here **N=6**, size of candidate key=**2**
- Thus **possible super keys** is $2^{(6-2)}$= **16 super keys.**

Ex2. Let R = (A, B, C, D, E) be a relation scheme with the following dependencies-

AB → C, C → D, B → E, Determine the total number of candidate keys and super keys.

Solution:

- **Essential attributes** of the relation are- **A and B.**
- So, attributes A and B will definitely be a part of every **candidate key.**
- $\{ AB \}^+$

  = { A,B,C,D,E }

  Thus **AB** is the **only possible candidate key** of the relation.

  Possible super keys is $2^{(5-2)}$= **8 super keys.**

Example 3

- **Ex3.** Consider the relation scheme **R**(E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies **F**= EF→ G, F→ I J, EH → KL , K → M , L → N What is the **key** for **R**?
- **Solution:**
- **Essential attributes** of the relation are- E, F and H.
- So, attributes E, F and H will definitely be a part of every **candidate key.**
- **{ EFH }**$^+$ = { EFGHIJKLMN }
- So, **EFH** is the only possible **candidate key of the relation**.
- **Possible super keys** is **2**$^{(10-3)}$= **128 super keys.**

Example 4

- **Ex4.** Consider the relation scheme **R**(A, B, C, D, E, H) and the set of functional dependencies **F**= A → B, BC → D, E → C, D → A . What are the **candidate keys** of **R**?
- **Solution:**
- **Essential attributes** of the **relation R** are- **E** and **H.**
- So, attributes **E** and **H** will definitely be a part of every **candidate key.**
- **{ EH }$^+$** = { EHC } thus **EH alone is not the candidate key.**
- **{ AEH }$^+$** = {ABCDEH} , thus **AEH** is a **candidate key.**
- **{ BEH }$^+$** = {ABCDEH} , thus **BEH** is a **candidate key.**
- **{ CEH }$^+$** = {CEH} , thus **CEH** is **not a candidate key.**
- **{ DEH }$^+$** = {ABCDEH} , thus **DEH** is a **candidate key.**

Minimal Sets of Functional Dependencies

- A **Minimal cover** of a set of functional dependencies ***E*** is a set of functional dependencies ***F*** that satisfies the property that every dependency in ***E*** is in the **closure** F^+ of *F* i.e. $\mathbf{E \subseteq F^+}$
- In addition, **this property is lost** if **any dependency** from the **set *F* is removed;**
- ***F*** must have **no redundancies** in it, and the **dependencies in *F*** are in a **standard form**(that is, they have **only one attribute on the right-hand side**).
- **Minimal cover** is also known as **Canonical cover.**

Significance of Canonical Cover

- Suppose that we have a set of **functional dependencies *F*** on a **relation schema.**
- Whenever a user performs an **update** on the **relation,** the **database system** must **ensure** that the **update does not violate** any **functional dependencies**,
- That is, **all the functional dependencies in *F*** are **satisfied** in the **new database state.**
- The system must **roll back** the **update** if it **violates any functional dependencies** in the **set *F*.**
- We can **reduce the effort spent** in **checking for violations by testing** a **simplified set of functional dependencies** that has the same **closure** as the given set.

Minimal/Canonical Cover

- **Formal definition of Minimal cover:**
- A **Minimal/Canonical cover *Fc*** for ***F*** is a **set of dependencies** such that ***F* logically implies** all **dependencies in *Fc*,** and ***Fc* logically implies** all **dependencies in *F*.**
- Furthermore, ***Fc*** must have the following **properties:**
  - No **functional dependency** in ***Fc*** contains an **extraneous attribute**.
  - Each **right side** of a **functional dependency** in ***Fc*** must have a **single attribute.**
  - Each **left side** of a **functional dependency** in ***Fc*** is **unique.**
    - That is, there are **no two dependencies $\alpha 1 \rightarrow \beta 1$** and **$\alpha 2 \rightarrow \beta 2$** in ***Fc*** such that **$\alpha 1 = \alpha 2$.**

Algorithm to compute Canonical Cover

- ***Fc* = *F***
- **repeat**
  - Use the **union rule** to **replace** any dependencies in ***Fc*** of the form
    - $\boldsymbol{\alpha 1 \rightarrow \beta 1}$ and $\boldsymbol{\alpha 1 \rightarrow \beta 2}$ with $\boldsymbol{\alpha 1 \rightarrow \beta 1\ \beta 2}$**.**
  - Find a **functional dependency** $\boldsymbol{\alpha \rightarrow \beta}$ in ***Fc*** with an **extraneous attribute** either in $\boldsymbol{\alpha}$ or in $\boldsymbol{\beta}$**.**
  - /* Note: the test for extraneous attributes is done **using *Fc***, not *F* */
  - If an **extraneous attribute** is found, **delete** it from $\boldsymbol{\alpha \rightarrow \beta}$**.**
- **until** ***Fc*** does not change.

Example

Question: Consider the following set ***F*** of functional dependencies on schema **R(*A,B,C*)**:

A →BC, B → C, A →B, AB → C*,* Compute the **canonical cover *F^c^*.**

Solution:

- ***F^c^*** *= {A →BC, B → C, A →B, AB → C}*
- Applying union rule on *A →BC and A →B we get A →BC thus*
- *F^c= {A →BC, B → C, AB → C}*
- **Is B extraneous in *AB → C*?**
- ***A^+^****= ABC thus* ***B is* extraneous** in *AB → C*.
- Thus after removing **B**, *AB → C becomes* ***A→ C.***
- **Is A extraneous in *AB → C*?**
- ***B^+^****= BC ≠ ABC thus* ***A*** *is* ***not* extraneous** in *AB → C*.
- Thus ***F^c^****= {A →BC, B → C, A → C}*

Example

- **Is B extraneous in *A →BC*?**
- We have to find **F'** = (F – {α → β}) U { α → (β – A)}
- F' = *B → C, A → C*
- ***A⁺*** *= AC does not contains B thus* ***B is not*** **extraneous**.
- **Is C extraneous in *A →BC*?**
- We have to find F' = (F – {α → β}) U { α → (β – A)}
- **F'** = *B → C, A → B*
- ***A⁺*** *= ABC contains C thus* ***C is* extraneous,** remove **C** from ***A →BC*** *which becomes* ***A→B***
- Thus ***F****c= {A →B, B → C, A → C} after removing the* ***redundant FD*** *A → C.*
- *Thus* ***Fc= {A →B, B → C}*** *is the* ***minimal cover.***
- ***Note:*** A **canonical cover** might **not be unique.**

Exercise

- Consider the set of functional dependencies ***F*** = *{A → BC, B → AC,* and *C → AB}* find all possible **canonical covers.**