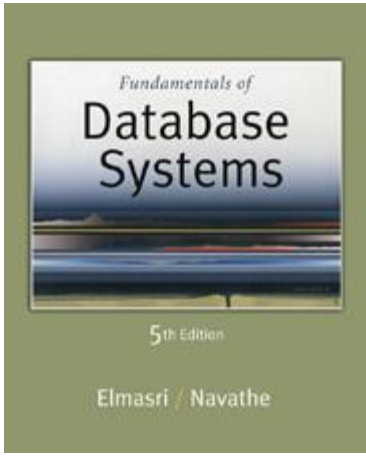
UNIT - 3

Functional Dependencies



#A Redundant Information in Tuples and Update Anomalies

- Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    1. Insertion anomalies
    2. Deletion anomalies
    3. Modification anomalies

Figure: Two relation schemas suffering from update anomalies

Figure 10.3
Two relation schemas suffering from update anomalies.
(a) EMP_DEPT and
(b) EMP_PROJ.

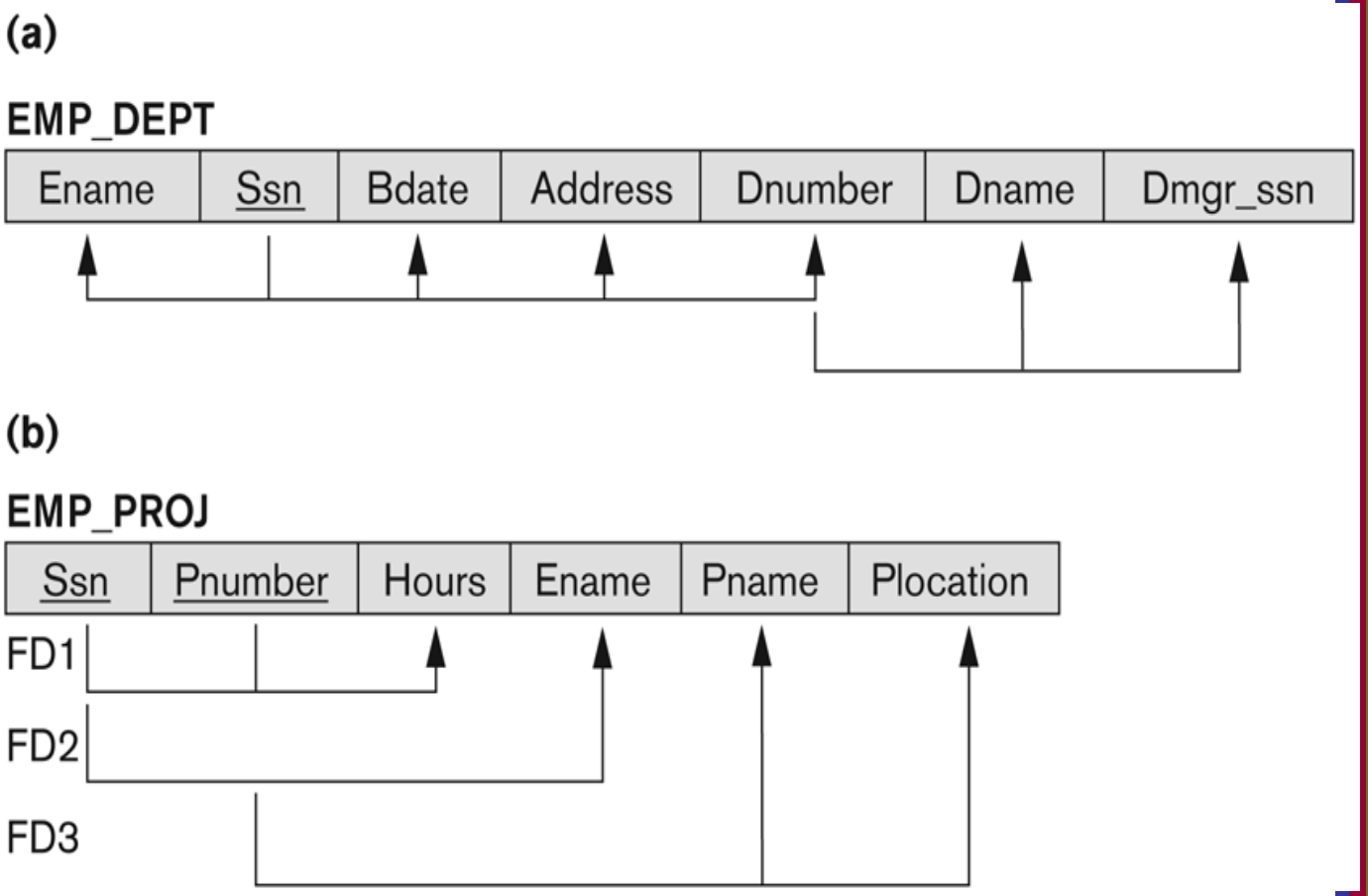


Figure: Example States for EMP_DEPT and EMP_PROJ

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston
888665555	20	Null	Borg, James E.	Reorganization	Houston

1. EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Update Anomaly:
  - Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1. So it is time consuming.

2. EXAMPLE OF AN INSERT ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it.
- Conversely
  - Cannot insert an employee unless an he/she is assigned to a project.

3. EXAMPLE OF AN DELETE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Delete Anomaly:
  - When a project is deleted, it will result in deleting all the employees who work on that project.
  - Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

#B Guideline to Redundant Information in Tuples and Update Anomalies

- GUIDELINE 1:
  - **Design a schema that does not suffer from the insertion, deletion and update anomalies.**
  - If there are any anomalies present, then note them so that applications can be made to take them into account.

B.1 Null Values in Tuples

- GUIDELINE 2:
  - Relations should be designed such that their tuples will have as few NULL values as possible
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - Attribute not applicable or invalid
  - Attribute value unknown (may exist)
  - Value known to exist, but unavailable

B.2 Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

- GUIDELINE 3:
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

#C Functional Dependencies

- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs And keys are used to define **normal forms** for relations
  - Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- **A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y.**
  - **i.e. X->Y**

#C Functional Dependencies

- **X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y**
- **For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]**
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints

- Social security number determines employee name
  - SSN -> ENAME
- Project number determines project name and location
  - PNUMBER -> {PNAME, PLOCATION}
- Employee ssn and project number determines the hours per week that the employee works on the project
  - {SSN, PNUMBER} -> HOURS

Examples of FD constraints

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every* relation instance r(R)
- If K is a key of R, then K functionally determines all attributes in R
  - (since we never have two distinct tuples with t1[K]=t2[K])

Examples of FD constraints

- Now let us consider, a relational schema "R" and let "x" and "y" be the two set of attributes, now there is a functional dependency from "x" to "y".

  i.e. **x->y** iff, **t1[x] = t2[x]** then, **t1[y] = t2[y]**

Example: In the corresponding relation:

Tuple	A	C
t1	a1	c1
t2	a1	c1
t3	a2	c2
t4	a2	c2
t5	a3	c2
t5	a3	c2

A → C holds as
t1[A] = t2[A] then t1[C] = t2[C]
t3[A] = t4[A] then t3[C] = t4[C]

but, C → A does not holds as.
t3[C] = t4[C] = t5[C] but
t3[A] = t4[A] does not equal t5[A]

Examples of FD constraints

- Consider the following table of data r(R) of the relation schema R(ABCDE) shown in Table:

1. Since the values of A are unique (a1, a2, a3, etc.), it follows from the FD definition that:

$$A \rightarrow B, \quad A \rightarrow C, \quad A \rightarrow D, \quad A \rightarrow E$$

- This can be summarized as

$$A \rightarrow BCDE.$$

- From our understanding of primary keys, A is a primary key.

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	C2	d2	e1
a3	b2	C1	d1	e1
a4	b2	C2	d2	e1
a5	b3	C3	d1	e1

Table R

Examples of FD constraints

2. Since the values of E are always the same (all e1), it follows that:

$A \rightarrow E, \quad B \rightarrow E, \quad C \rightarrow E, \quad D \rightarrow E$

Other observations:

- Combinations of BC & BD are unique, therefore **BC → ADE, BD → ACE**.
- If C values match, so do D values.
  - Therefore, **C → D**
  - However, D values don't determine C values
  - So C determine D, and D does not determine C.

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	C2	d2	e1
a3	b2	C1	d1	e1
a4	b2	C2	d2	e1
a5	b3	C3	d1	e1

Table R

Types of Functional Dependencies

1. Trivial functional dependency: 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.**
- **The following dependencies are also trivial: A->A & B->B**

For example the table : **Student (Student_Id, Student_Name)**

Then trivial FD are:

- **{Student_Id, Student_Name} -> Student_Id**
- **Student_Id -> Student_Id**
- **Student_Name -> Student_Name**

Types of Functional Dependencies

2. Non-trivial functional dependency: 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 the table: Employee(emp_id, emp_name, emp_address)

Then Non-trivial FD are:

emp_id -> emp_name

emp_id -> emp_address

Completely non trivial FD: If a Functional dependency X->Y holds true where X intersection Y is Null then this dependency is said to be completely non trivial function dependency.

Types of Functional Dependencies

3. Fully functional dependency:

In a relation ***R(A,B,C)*** if (A,B) is a primary key then

(A,B)->C holds and neither ***A->C*** nor ***B->C*** holds then **C** is fully functionally dependent on ***(A,B).***

E.g.: {SSN, PNUMBER} -> HOURS

Say you are making a database of all storage devices like CD, DVD etc. ***PRODUCT(Item, Make, Rate, Discount)***

Then {Item, Make} -> {Rate, Discount}

> The set of attributes ***(Rate, Discount)*** are fully dependent on the attributes ***(Item, Make)***. This means we need to get the information of both Item and Make to get values of Rate and Discount.

Types of Functional Dependencies

4. Partial dependency:

- In a relation ***R(A,B,C,D)*** if the key is ***(A,B)*** and

  (A,B)->(C,D) holds, also ***A->C*** holds, ***C*** is said to be partially dependent on ***(A,B)***

- ***For example: PRODUCT2(Item, Make, Rate, Capacity)***.
- In this case, capacity of the disk only depends on the item (CD->700MB, Blu Ray Disk->25GB etc).

Then {Item, Make} -> {Rate, Capacity}

Also Item-> Capacity

#D Inference Rules for FDs

- Given a set of FDs F, we can **infer** additional FDs that hold whenever the FDs in F hold
- **Armstrong's inference rules or Primary Rules:**
  - **IR1. (Reflexive/trivial) If Y *subset-of* X, then X -> Y**
  - **IR2. (Augmentation) If X -> Y, then XZ -> YZ**
    - **(Notation: XZ stands for X U Z)**
  - **IR3. (Transitive) If X -> Y and Y -> Z, then X -> Z**
- **IR1, IR2, IR3 form a sound and complete set of inference rules**
  - These are rules hold and all other rules that hold can be deduced from these

Inference Rules for FDs

- **Some Additional/Secondary inference rules that are useful:**
  - **Union:** If X -> Y and X -> Z, then X -> YZ
  - **Decomposition:** If X -> YZ, then X -> Y and X -> Z
  - **Psuedotransitivity:** If X -> Y and WY -> Z, then WX -> Z
- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Closure of Attribute for FDs

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F.
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X.
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Steps to Find the Attribute Closure

- Given FD set of a Relation R, The attribute closure set S be the set of Attributes A in FD:

1. Add A to S.
2. Recursively add attributes which can be functionally determined from attributes of the set S until done.

Example to Find the Attribute Closure

Given: R(E-ID, E-NAME, E-CITY, E-STATE)

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

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

Similarly: (E-NAME)+ = {E-NAME}
(E-CITY)+ = {E-CITY, E_STATE}

Find the attribute closures of given FDs: R(ABCDE) = {AB->C, B->D, C->E, D->A}

- $(B)^+ = \{B,D,A,C,E\}$
- $(C,D)+ = \{C,D,E,A\}$
- $(B,C)+ = \{B,C,D,E,A\}$

Finding Candidate Keys and Super Keys of a Relation using FD set

- The **set of attributes** whose attribute closure is set of all attributes of relation is called super key of relation. For Example, the EMPLOYEE relation shown in Table 1 has following 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.

Finding Candidate Keys and Super Keys of a Relation using FD set

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

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 keys: $(E\text{-}ID)^+$, $(E\text{-}ID, E\text{-}NAME)^+$, $(E\text{-}ID, E\text{-}CITY)^+$, $(E\text{-}ID, E\text{-}STATE)^+$, $(E\text{-}ID, E\text{-}CITY, E\text{-}STATE)^+$

Extracting keys from Closure of attributes:

STUDENT					
STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE

- (STUD_NO, STUD_NAME)+ = {STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE}
- (STUD_NO)+ = {STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY, STUD_AGE}
- (STUD_NO, STUD_NAME) will be super key but not candidate key because its subset (STUD_NO)+ is equal to all attributes of the relation. So, STUD_NO will be a candidate key.

#E Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$
- Definition (**Covers**):
  - F **covers** G if every FD in G can be inferred from F
    - (i.e., if G^+ *subset-of* F^+)
- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

#F Minimal Sets of FDs

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its RHS.
  2. We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
  3. We cannot replace any dependency **X -> A** in F with a dependency **Y -> A**, where Y is proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.

Minimal Sets of FDs

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set

Finding Minimal cover of F.Ds:

- Just as we applied inference rules to expand on a set *F* of FDs to arrive at *F*+, its closure, it is possible to think in the opposite direction to see if we could shrink or reduce the set *F* to its *minimal form* so that the minimal set is still equivalent to the original set *F.*
- **Definition:** An attribute in a functional dependency is considered **extraneous attribute** if we can remove it without changing the closure of the set of dependencies. Formally, given F, the set of functional dependencies and a functional dependency $X \rightarrow A$ in F, attribute Y is extraneous in X if Y *is a subset of* X, and F logically implies $(F - (X \rightarrow A) \cup \{ (X - Y) \rightarrow A \})$

Algorithm for Minimal Set of FDs

- **Algorithm 15.2. Finding a Minimal Cover F for a Set of Functional Dependencies E**
  - **Input: A set of functional dependencies E.**

1. Se tF:=E.
2. Replace each functional dependency $X \rightarrow \{A1, A2, ..., An\}$ in F by the n functional dependencies $X \rightarrow A1$, $X \rightarrow A2$, ..., $X \rightarrow An$.
3. For each functional dependency $X \rightarrow A$ in F
   for each attribute B that is an element of X
   if $\{ \{F - \{X \rightarrow A\}\} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.

   (* The above constitutes a removal of the extraneous attribute B from X *)
4. For each remaining functional dependency $X \rightarrow A$ in F if $\{F - \{X \rightarrow A\}\}$ is equivalent to F, then remove $X \rightarrow A$ from F.

   (* The above constitutes a removal of the redundant dependency $X \rightarrow A$ from F *)

(i) Computing Minimal Set of FDs

We illustrate algorithm 15.2 with the following: ***R (A,B,D)***

Let the given set of FDs be E : {$B \rightarrow A$, $D \rightarrow A$, $AB \rightarrow D$}.We have to find the minimum cover of E.

- All above dependencies are in canonical form; so we have completed step 1 of Algorithm 10.2 and can proceed to step 2. In step 2 we need to determine if $AB \rightarrow D$ has any redundant attribute on the left-hand side; that is, can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?
- Since B $\rightarrow$ A, by augmenting with B on both sides (IR2), we have $BB \rightarrow AB$, or $B \rightarrow AB$ (i). However, $AB \rightarrow D$ as given (ii).
- Hence by the transitive rule (IR3), we get from (i) and (ii), $B \rightarrow D$. Hence $AB \rightarrow D$ may be replaced by $B \rightarrow D$.
- We now have a set equivalent to original E , say E' : {$B \rightarrow A$, $D \rightarrow A$, $B \rightarrow D$}.

No further reduction is possible in step 2 since all FDs have a single attribute on the left-hand side.

- In step 3 we look for a redundant FD in E′. By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence $B \rightarrow A$ is redundant in E' and can be eliminated.
- Hence the minimum cover of E is {$B \rightarrow D$, $D \rightarrow A$}.

(ii) Example for Minimal set of FD

R (A,B,C)

$$F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$$

Resultant Minimal Set:

$$F = \{B \rightarrow C, A \rightarrow B\}$$

(iii) Example for Minimal set of FD

R (A,B,C,D,E)

F={

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

}

Resultant Minimal Set:

F={

$A \rightarrow BC$

$D \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

}

OR

F={

$A \rightarrow BC$

$C \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

}

(iv) Example for Minimal set of FD

R (A,B,C,D,E)

F = {

$A \rightarrow B$

$AB \rightarrow C$

$D \rightarrow AC$

$D \rightarrow E$

}

Resultant Minimal Set:

F = {

$A \rightarrow BC$

$D \rightarrow AE$

}

(v) Example for Minimal set of FD

Resultant Minimal Set:

$R(W,X,Y,Z)$

$FD=\{$
$X \rightarrow W$
$WZ \rightarrow XY$
$Y \rightarrow WXZ \}$

$R(W,X,Y,Z)$

$FD=\{$
$X \rightarrow W$
$W \rightarrow Y$
$Y \rightarrow XZ \}$