Experiment - 4

Name: Shivam Agarwal **UID : 23BCS13100**
Section: 23KRG-3B **Branch: BE-CSE**
Semester : 5th **Date of Performance : 12-Sep-2025**
Subject Name: ADBMS **Subject- Code : 23CSP - 333**

1. Aim:

a). Consider a relation R having attributes as R(ABCD), functional dependencies are given below:

AB -> C
C -> D
D - > A

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

ANSWER:

Given:

- Relation R(A, B, C, D)
- Functional dependencies:
  - AB → C
  - C → D
  - D → A

Step 1: Find Candidate Keys by computing closures

- AB^+ = {A, B} + C (from AB → C) + D (from C → D) + A (from D → A)
  So, AB^+ = {A, B, C, D} → AB is a candidate key.

- BC^+ = {B, C} + D (from C → D) + A (from D → A)
  So, BC^+ = {A, B, C, D} → BC is a candidate key.

Step 2: Prime and Non-Prime Attributes

- Prime attributes (part of candidate keys): A, B, C
- Non-prime attribute: D

Summary:

- Candidate Keys: AB, BC
- Prime Attributes: A, B, C
- Non-Prime Attribute: D

b). Relation R(ABCDE) having functional dependencies as :

A->D
B->A
BC->D
AC->BE

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

Let's solve this step-by-step for relation **R(A, B, C, D, E)** with these functional dependencies:

- A → D
- B → A
- BC → D
- AC → BE

Step 1: Identify candidate keys

We need to find minimal attribute sets whose closure covers **all attributes** {A, B, C, D, E}.

Step 2: Try attribute closures

Check BC closure (BC^+):

- Start: {B, C}
- B → A, so add A → {A, B, C}
- AC → BE: since we have A and C, add B and E → but B already included, so add E → {A, B, C, E}
- A → D, so add D → {A, B, C, D, E}

BC^+ = {A, B, C, D, E} → BC is a candidate key.

Check BA closure (BA^+):

- Start: {B, A}
- B → A (already have A)
- A → D, add D → {A, B, D}
- AC → BE (need C, not in closure)
- BC → D (need C, not in closure)

No C or E yet → BA^+ = {A, B, D} (not all attributes) → Not a candidate key.

Check AC closure (AC^+):

- Start: {A, C}
- AC → BE, add B and E → {A, B, C, E}
- B → A (already have A)
- A → D, add D → {A, B, C, D, E}

AC^+ = {A, B, C, D, E} → AC is a candidate key.

Step 4: Final candidate keys:

- **AC**
- **BC**

Step 5: Prime and Non-Prime attributes

Prime attributes: those in candidate keys → A, B, C
Non-prime attributes: D, E

c) Consider a relation R having attributes as R(ABCDE), functional dependencies are given below:

B->A
A->C
BC->D
AC->BE

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

ANSWER:

Given:

Relation R(A, B, C, D, E)
Functional Dependencies (FDs):

- B → A
- A → C
- BC → D
- AC → BE

Step 1: Check closures of attribute sets

Try BC closure (BC^+):

- Start with {B, C}
- B → A → add A → {A, B, C}
- A → C → already have C
- BC → D → add D → {A, B, C, D}
- AC → BE → Since we have A and C, add B (already have) and E → {A, B, C, D, E}

BC^+ = {A, B, C, D, E}
So BC is a candidate key.

Try AC closure (AC^+):

- Start with {A, C}
- A → C (already have)
- AC → BE → add B and E → {A, B, C, E}
- B → A (already have)
- BC → D → need B and C, we have both, add D → {A, B, C, D, E}

AC^+ = {A, B, C, D, E}

So AC is a candidate key.

Step 2: Candidate keys

- BC
- AC

Step 3: Prime and Non-Prime attributes

- Prime attributes = attributes in candidate keys = {A, B, C}
- Non-prime attributes = remaining attributes = {D, E}

d).Consider a relation R having attributes as R(ABCDEF), functional dependencies are given below:

A->BCD
BC->DE
B->D
D->A

Identify the set of candidate keys possible in relation R. List all the set of prime and non prime attributes.

ANSWER:

Given:

Relation R(A, B, C, D, E, F)
Functional Dependencies (FDs):

- A → B, C, D
- BC → D, E
- B → D
- D → A

Step 1: Understand the FDs and try attribute closures

Step 2: Check closure of some attribute sets:

Try AF closure (AF^+):

- {A, F}
- From above, A^+ = {A, B, C, D, E}
- So $AF^+ = A^+ \cup \{F\}$ = {A, B, C, D, E, F}

AF is a candidate key.

- Try B F:

Start {B, F}

- B → D → add D
- D → A → add A
- A → B, C, D → add C
- BC → D, E (need C and B, both present) → add E

Now: {A, B, C, D, E, F} → all attributes!

So BF is a candidate key.

Try DF:

Start {D, F}

- D → A → add A
- A → B, C, D → add B, C
- BC → D, E → add E

So {A, B, C, D, E, F} → all attributes

So DF is the candidate key.

Candidate keys:

- AF
- BF
- DF

Prime and Non-prime attributes:

- Prime attributes: Attributes part of any candidate key = {A, B, D, F}
- Non-prime attributes: {C, E}

E).

•Designing a student database involves certain dependencies which are listed below:

X ->Y

WZ ->X

WZ ->Y

Y ->W

Y ->X

Y ->Z

The task here is to remove all the redundant FDs for efficient working of the student database management system.

ANSWER:

Given FDs:

1. X → Y
2. WZ → X
3. WZ → Y
4. Y → W
5. Y → X
6. Y → Z

Goal:

Remove redundant FDs while preserving the dependency equivalence.

Step 1: Check if any FD can be derived from others

FD 3: WZ → Y

- We have WZ → X (FD 2)

- And X → Y (FD 1)

From these two, can we get WZ → Y?

Yes:
WZ → X → Y

So FD 3 (WZ → Y) is redundant because it can be derived from WZ → X and X → Y.

Step 2: Check if FD 1 (X → Y) is redundant

Is X → Y derivable from others?

Try from Y → X (FD 5), but that goes the other way (Y to X).

No FD shows Y determines X → Y, so no.

So FD 1 is not redundant.

Step 3: Check if FD 5 (Y → X) is redundant

Is Y → X derivable from others?

- Y → W (FD 4)
- WZ → X (FD 2)
- But without Z, from Y we can't get X directly.

No direct derivation, so FD 5 is not redundant.

Step 4: Check if FD 6 (Y → Z) is redundant

No FD shows anything to get Z from other attributes except Y → Z.

So FD 6 is not redundant.

Step 5: Check if FD 4 (Y → W) is redundant

No other FD shows how to get W from others, so FD 4 is not redundant.

Step 6: Check if FD 2 (WZ → X) is redundant

- Y → X (FD 5)
- Y → Z (FD 6)
- If WZ → X was derivable from Y → X and others, then WZ would need to determine Y.

Is WZ → Y? We removed that FD because it is redundant.

Can WZ determine Y? Let's check closure of WZ without FD 3:

- $WZ^+ = \{W, Z\}$
- No FD to get Y directly.

So WZ → X is not redundant.

Final set of non-redundant FDs:

- X → Y
- WZ → X
- Y → W
- Y → X
- Y → Z

Debix Pvt Ltd needs to maintain a database having dependent attributes ABCDEF. These attributes are functionally dependent on each other for which functionally dependency set F given as:

{A -> BC

D -> E

BC -> D

A -> D} Consider a universal relation R1(A, B, C, D, E, F) with functional dependency set F, also all attributes are simple and take atomic values only. Find the highest normal form along with the candidate keys with prime and non-prime attributes.

Given:

Relation **R1(A, B, C, D, E, F)**
Functional dependencies (FDs):

- A → B, C
- D → E
- BC → D
- A → D

Goal:

1. Find **candidate keys**
2. Identify **prime and non-prime attributes**
3. Determine the **highest normal form** of R1

Step 1: Understand the functional dependencies

FD set FF:

- A → B, C
- A → D (given separately, but A → B, C means A → B and A → C, so A → D means A → B, C, D)
- D → E
- BC → D

Let's rewrite to be clear:

- A → B
- A → C
- A → D
- D → E
- BC → D

Step 2: Find candidate keys

The candidate key is a minimal set of attributes that functionally determine **all attributes** A,B,C,D,E,FA, B, C, D, E, F.

Try closure of A:

- Start with {A}
- A → B, C, D → add B, C, D → {A, B, C, D}
- D → E → add E → {A, B, C, D, E}

So A alone is **not a candidate** key.

Try closure of A and F:

- Start with {A, F}
- From above, A → B, C, D, and D → E
- So A, F^+ = {A, B, C, D, E, F}

All attributes covered → **AF** is a candidate key.

Try closure of BC:

- Start with {B, C}
- BC → D → add D → {B, C, D}
- D → E → add E → {B, C, D, E}

So BC is **not a candidate** key.

Try closure of BC and F:

- $\{B, C, F\}^+$
- BC → D → add D → {B, C, D, F}
- D → E → add E → {B, C, D, E, F}

Try closure of A and B:

- $\{A, B\}^+$
- A → B, C, D → add C, D → {A, B, C, D}
- D → E → add E → {A, B, C, D, E}

Try closure of A, F:

- Already done → candidate key.

Try closure of A, F:

- Candidate key = **AF**

Step 3: Prime and non-prime attributes

- **Prime attributes**: Attributes that are part of any candidate key → A, F
- **Non-prime attributes**: B, C, D, E

Step 4: Check normal forms

1NF:

Given all attributes are atomic, so **R1 is in 1NF**.