HW4 **TEAM-10**

Members: Harshika, Rithvika, Vishakha ,Vaishnavi

Part One

1)

Given the attribute set R={A,B,C,D,E,G,H} and the functional dependency (FD) set F={AB→C,AC→B,AD→E,B→D,BC→A,E→G}
STEPS:
<u>Identifying Functional Dependencies (FD s) that Hold over the set:</u>For a given subset of attributes, consider only the FDs from the original set where both the left-hand side (determinant) and the right-hand side (dependent) attributes are entirely contained within the subset.
<u>Minimal Cover :</u>
If any FD has multiple attributes on the RHS, decompose it into single-attribute Fds.
Eliminate redundant FDs by checking whether the FD can be derived from the remaining set.
Simplify FDs with multiple attributes on the LHS by removing unnecessary attributes while preserving equivalence.
Finding Normal form : First we find candidate keys , then proceed by checking 1NF,2NF,3NF and finally BCNF.

(a) R={A,B,C}
i) Functional Dependencies that Hold**:** {AB→C, AC→B, BC→A}

- Minimal Cover:
    - No RHS decomposition needed.(No multiple attributes)
    - No redundant FDs or attributes. And no unnecessary attributes in LHS to remove. The Final minimal cover: {AB→C, AC→B, BC→A}.
- Candidate Keys: {AB, BC, CA}
  ii) Normal Form**:** It satisfies 1NF , 2NF , 3NF .Since no partial or transitive dependencies exist. And the relation is in BCNF (satisfies the BCNF properties)**.**
  Therefore the strongest normal form **: BCNF**
  iii) It is already in BCNF.

(b) R={A,B,C,D,E}

i) Functional Dependencies that Hold**:**{AB→C, AC→B, AD→E, B→D, BC→A}

- Minimal Cover:
    - No RHS decomposition needed.(No multiple attributes)
    - No redundant FDs or attributes. And no unnecessary attributes in LHS to remove. The Final minimal cover: {AB→C, AC→B, AD→E, B→D, BC→A}.
- Candidate Keys: {AB, BC, CA}
- Normal Form: It satisfies 1NF , FD B→D introduces a partial dependency, violating 2NF. Therefore the strongest normal form : **1 NF**
- Decomposition into BCNF:
    - Split ABCDE into two relations ABCE and BD. (B → D is violating the 2NF)

- These relations, ABCE and BD are in BCNF.

(c) R={A,B,C,G}

i) Functional Dependencies that Hold**:**{AB→C, AC→B, BC→A}

- Minimal Cover:
  - No RHS decomposition needed.(No multiple attributes)
  - No redundant FDs or attributes. And no unnecessary attributes in LHS to remove. The Final minimal cover: {AB→C, AC→B, BC→A}.
- Candidate Keys: {ABG , BGC , CGA}
- Normal Form: It is in 1 NF . It violates 2 NF as all relations are partial dependencies. Therefore the strongest normal form : **1 NF**
- Decomposition into BCNF:
  - Split ABCG into two relations ABC and ABG.
  - Both the resulting relations , ABC and ABG are in BCNF.

(d) R={D,E,G,H}

i) Functional Dependencies that Hold**:** {E→G}

- Minimal Cover:
  - No RHS decomposition needed.(No multiple attributes)
  - No redundant FDs or attributes. And no unnecessary attributes in LHS to remove. The Final minimal cover: {E→G}.
- Candidate Keys: { DEH }
- Normal Form: The relation is in 1 NF . As E→ G violates the 2 NF as it is partial dependency .
  Therefore the strongest normal form **: 1 NF**
- Decomposition into BCNF:
  - Split DEGH into two relations DEH and EG.
  - Both resulting relations , DEH and EG are in BCNF.

(e) R={A,B,C,E,H}

i) Functional Dependencies that Hold**: :** {AB→C, AC→B, BC→A}

- Minimal Cover:
  - No RHS decomposition needed.(No multiple attributes)
  - No redundant FDs or attributes. And no unnecessary attributes in LHS to remove. The Final minimal cover: {AB→C, AC→B, BC→A}.
- Candidate Keys: {ABEH , BCEH, ACEH}
- Normal Form: The relation is in 1 NF . All relations are partial dependencies which violates 2NF.
  Therefore the strongest normal form : **1 NF**
- Decomposition into BCNF:
  - Split ABCEH into two relations ABEH and ABC.
  - Both resulting relations ABEH and ABC are in BCNF.

2. R = ABCDEG and F = {$AB \rightarrow C$, $AC \rightarrow B$, $AD \rightarrow E$, $B \rightarrow D$, $BC \rightarrow A$, $E \rightarrow G$}

a) {AB, BC, ABDE, EG}

Dependency preserving:

For each decomposed relation R_i, we included all functional dependencies in F where both the left-hand side and the right-hand side attributes belong to R_i. We found the closure for each of the dependencies with respect to the original dependencies and included only those whose closure was given by the original dependencies. Then, we took a union of these dependencies, F', and took a closure of the original dependencies with respect to the newly formed dependencies set F'. If the original dependencies could be formed using this new set of dependencies F', then the decomposition is dependency preserving.

Using this method we found F', the new dependencies set

$B \rightarrow D$

$AB \rightarrow D$

$AB \rightarrow E$

$AD \rightarrow E$

$BE \rightarrow D$

$AB \rightarrow DE$

$ABD \rightarrow E$

$ABE \rightarrow D$

We found that when we took the closure of AB, AB^+, with respect to this set, $AB \rightarrow C$ (or even $AC \rightarrow B$) was not preserved. Therefore, the given decomposition is not dependency preserving.

Lossless Join:

To find if the decomposition is lossless or not we considered consider an following instance of R with the following tuples:

{(a1, b, c1, d1, e1, g1), (a2, b, c2, d2, e2, g2)}

Because of the functional dependencies $BC \rightarrow A$ and $AB \rightarrow C$, $a1 \neq a2$ if and only if $c1 \neq c2$.

We performed the join $AB \bowtie BC$, it contained 4 tuples: {(a1, b, c1), (a1, b, c2), (a2, b, c1), (a2, b, c2)}.

We then performed the join $ABDE \bowtie EG$ and got 2 tuples, then we did a full join of AB, BC, ABDE and EG. We observed that it contained 4 tuples, which does not match the number of tuples in the original relation. Therefore, the decomposition here is a lossy decomposition.

b) {ABC, ACDE, ADG}

Using the same method as mentioned above, we found F′, the new dependencies set

AB → C

BC → A

AC → B

AD → G

AC → D

AC → E

AC → DE

AD → E

ACD → E

ACE → D

the decomposition is <u>not dependency preserving</u> because, B → D is not covered and there is no way to derive it from any closure

LOSLESS JOIN:

The decomposition is lossless-join if, for every pair of subrelations Ri and Rj their intersection Ri ∩Rj:

1. Is a superkey for at least one of Ri or Rj, or
2. Functionally determines all the attributes in Ri or Rj

Relation R=ABCDEG with FD set:

F={AB→C,AC→B,AD→E,B→D,BC→A,E→G}.

Decomposition:

{ABC,ACDE,ADG}.

1. Join of ABC and ACDE:
   - Intersection: ABC∩ACDE=AC.
   - Check if AC is a superkey for ABCDEG:
     - AC→B (from AC→B).
     - AC+B→A (from BC→A).
     - AC→B→D (from B→D).
     - AD→E and E→G.
     - Thus, AC→ABCDEG, making AC a superkey.

   Conclusion: Join of ABC and ACDE is lossless.

2. Join of Intermediate Relation (ABC⋈ACDE) and ADG:

- Intersection: $(ABC \bowtie ACDE) \cap ADG = AD$.
- Check if AD is a superkey for ADG:
  - $AD \rightarrow E$ (from $AD \rightarrow E$).
  - $E \rightarrow G$, so $AD \rightarrow ADG$.

Conclusion: Join of Intermediate Relation and ADG is lossless.

Since both steps in the decomposition maintain the lossless-join property, the decomposition {ABC,ACDE,ADG} is lossless-join.

PART 2

TABLE: PURCHASES

Customer_ID	Order_ID	Product_ID	Cust Name	Product Name	Phn Nos	Day	Discount

a) Conversion to 1NF

To convert the given "PURCHASES" table to 1NF: "Phn Nos" attribute in the given table is multi-valued. To ensure that the table has atomic (indivisible) values and each column contains only one value for each row, separate "Phn Nos" with multiple values into new rows. This change makes sure that the table is in 1NF.

TABLE1:

Composite Primary Keys = { Customer_ID , Order_ID , Product_ID }

Customer_ID	Order_ID	Product_ID	Cust Name	Product Name	Phn Nos	Day	Discount

b) Conversion to 2NF

To convert the table "TABLE1" to 2NF: We need to ensure that there are no partial dependencies. Create separate tables to eliminate partial dependencies if there are any. Each table should have attributes that depend fully on the primary key (Composite key). In TABLE1, there are three partial dependencies.

Cust Name and Phn Nos attributes depends only on Customer_Id, but the composite primary key has Order_ID and Product_ID. So, create a new table with Customer_ID as Primary Key. Name this as TABLE1A.

Day and Discount attributes depends only on Order_ID, but the composite primary key has Customer_ID and Product_ID. So, create a new table with Order_ID as Primary Key. Name this as TABLE1B.

Product Name attribute depends only on Product_ID, but the composite primary key has Customer_ID and Order_ID. So, create a new table with Product_ID as Primary Key. Name this as TABLE1C.

After removing the partial dependencies from TABLE1, we will have the table with attributes Customer_ID, Order_ID, Product_ID. The composite primary key has Customer_ID , Order_ID , Product_ID. Name this table as TABLE1D.

TABLE1A:

Primary Key: Customer_ID

Customer_ID	Cust Name	Phn Nos

TABLE1B:

Primary Key: Order_ID

Order_ID	Day	Discount

TABLE1C:

Primary Key: Product_ID

Product_ID	Product Name

TABLE1D:

Composite Primary Key: { Customer_ID , Order_ID , Product_ID }

Foreign Keys: { Customer_ID , Order_ID , Product_ID }

Customer_ID	Order_ID	Product_ID

c) Conversion to 3NF

To satisfy 3NF conditions for the above tables, we need to remove transitive dependencies if we have any in any of the tables. A transitive dependency occurs when a non-prime attribute depends on another non-prime attribute, which in turn depends on the primary key. In TABLE1B, we have a transitive dependency. Non-prime attribute Discount depends on another non-prime attribute Day, Day depends on primary key Order_ID. To remove this transitive dependency, we create a new table with Day and Discount attributes where Day is a primary key (with table name TABLE1E). We remove the Discount attribute from TABLE1B.

TABLE1A:

Primary Key: Customer_ID

Customer_ID	Cust Name	Phn Nos

TABLE1B:

Primary Key: Order_ID

Foreign Key: Day

Order_ID	Day

TABLE1C:

Primary Key: Product_ID

Product_ID	Product Name

TABLE1D:

Composite Primary Key: { Customer_ID , Order_ID , Product_ID }

Foreign Keys: { Customer_ID , Order_ID , Product_ID }

Customer_ID	Order_ID	Product_ID

TABLE1E:

Primary Key: Day

Day	Discount

Above relations are in 3NF form. There are no violations in 3NF.