Module 2 (16 hours)

FUNCTIONAL DEPENDENCY(FD) (Maier,1983)

Definition: If X and Y are attributes of a relation R, then Y is said to be functionally dependent on X (or Y is functionally determined by X or X functionally determines Y), denoted by X->Y, if each value of X is associated with exactly one value of Y so that for any two tuples t1 and t2 in relation state r of R, we have

- X->Y => (t1[X]=t2[X] => t1[Y]=t2[Y])
- L.H.S.(=X) is known as determinant
- R.H.S.(=Y) is known as determined

Examples-FUNCTIONAL DEPENDENCY(FD)

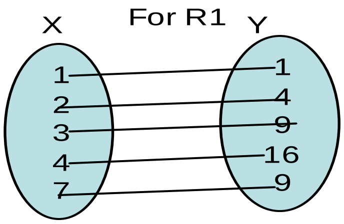
R1

X	Y
1	1
2	4
3	9
4	16
7	9

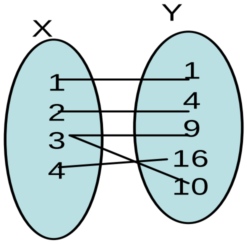
R2

X	Y
1	1
2	4
3	9
4	16
3	10

- ➢ In R1, X->Y, because it is one-one relation
- ➢ In R2, X->Y is not satisfied because it is one-many relation



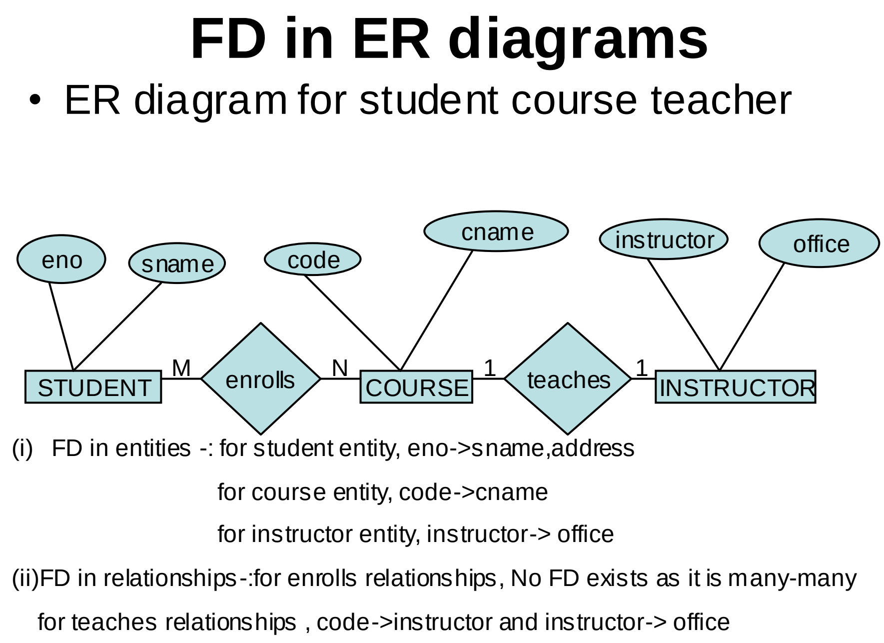
R1:One-to-One relationship between data for X and Y



R2:one-to-many relationship between data for X and Y

- X->Y (i.e. Y is functionly dependent on X) <=>There exists one-to-one and many-to-one relationships (associations) between data values of attributes X and Y
- Yis notfunctionally dependent on X if there exists one-to-many and many-to-many relationships (associations) between data values of attributes X and Y
- A functional dependency is a constraint between two sets of attributes from the database(i.e. Data inter-relationship in a relation or relationship between attributes)
- FD is a property of the semantics or meaning of the attributes.
- FD is a property of a relational schema(intension) and not a property of a particular instance of the schema(extension).
- If X->Y in R, this does not say whether or not Y->X in R
- **FD in relationships**: If X and Y be the primary keys of two different entities or tables and the two entities have relationships(1:1, M:1, 1:N, M:N), then we must have the following conclusions:
  - For one-to-one (1:1)relationships: X->Y and Y->X
  - For many-to-one (M:1)relationship(with X on many side): X->Y but not Y->X
  - For one-to- many(1:M)and many-to-many(M:N) relationship: No FD exists.

- **FD in ER diagrams**:



FD in ER diagrams

- ER diagram for student course teacher

(i) FD in entities -: for student entity, eno->sname,address

for course entity, code->cname

for instructor entity, instructor-> office

(ii)FD in relationships-:for enrolls relationships, No FD exists as it is many-many

for teaches relationships , code->instructor and instructor-> office

- **Exercise:** Below is an instance of R(A1,A2,A3,A4). Choose the FD which may hold on R. mm1.A4->A1 , 2.A2A3->A4, 3.A2A3->A1 .

A1	A2	A3	A4
1	2	3	4
1	2	3	5
6	7	8	2
2	1	3	4

Solution: 1.A4->A1 is incorrect: The 1st and 4th tuple violates it.
2. A2A3->A4 is incorrect: The1st and 2nd tuple violates it.
3. A2A3 -> A1 is correct.

REDUNDANCY AND ASSOCIATED PROBLEMS

- **Anomalies(or database anomalies):** Relations(or tables) that that have redundant data may have problems of wastage of memory space called as anomalies.
- **Classification (or types) of anomalies:**
  - Update(or modification) anomaly
  - Insertion anomaly
  - Deletion anomaly
    - **Update anomaly**: This anomaly is caused due to the data redundancy. Redundant information makes updates more difficult. An update anomaly results in data inconsistency.
    - **Insertion anomaly**: This anomaly is caused due to the inability to represent certain information.
    - **Deletion anomaly**: This anomaly is caused due to the loss of information.
- Illustration of anomalies through example: Consider the relation “STUDENT” as shown in figure below:

Eno	sname	address	code	cname	instructor	Office
05112345	**RAHUL**	**RANCHI**	**MCS-011**	**PROBLEM SOLUTION**	**NAYAN KUMAR**	**102**
05112345	**RAHUL**	**RANCHI**	**MCS-012**	**COMPUTER ORGANIZATION**	**ANURAG SHARMA**	**105**
05112345	**RAHUL**	**RANCHI**	**MCS-014**	**SAD**	**PREETI ANAND**	**103**
0511341	**APARANA**	**GURGAON**	**MCS-014**	**SAD**	**PREETI ANAND**	**103**

- **Update anomaly:** Changing the name of the instructor of MCS-014 would require that all tuples containing MCS-014 would require that all tuples containing MCS-014 enrolment information be updated. If for some reason , all tuples are not updated , we might have a database that gives two names of instructor for the subject MCS-014 which is inconsistent information. This problem is called update anomaly.
- Insertion anomaly: If one wanted to insert the code and name of a new course in the database 'STUDENT', it would not be possible until a student enrolls in that course. Information about a new student can't be inserted in the database until the student enrolls in a course. These problems are called insertion anomalies.
- Deletion anomaly: In 'STUDENT' relation, if we delete the tuple corresponding to student'05011341' enrolled for MCS-014, we will lose relevant information about the student like eno, snme and address of this student. Deletion of tuple having sname "RAHUL" and cno MCS-012" will result in loss of information that MCS-012 is named computer organization having an instructor "Anurag Sharama" , whose office number is 105. These are called deletion anomalies.

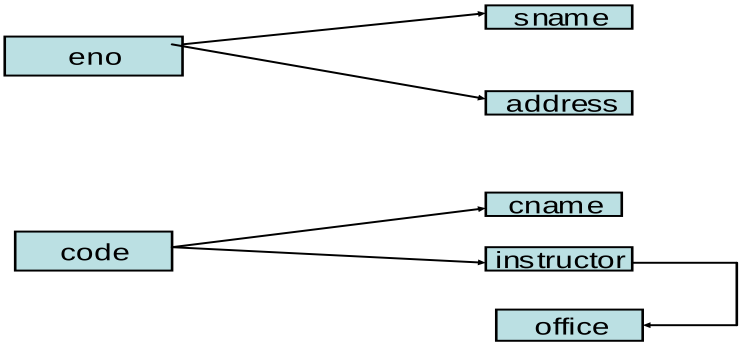
- The anomalies arise primarily because the relation "STUDENT" has information about students as well as subjects.
- Solution to the database anomalies: One solution to the anomaly problems is to decompose the relation into two or more small relations. The basis of this decomposition is determined by data inter-relationship. Data inter-relationship is determined by the concept of functional dependency. The root cause of the presence of anomalies in a relation is determined by the components of the key and non-key attributes.
- Hence {eno}->{sname,address}, {code}->{cname,instructor},{instructor}->{office}
- Now the smaller tables are:

Eno	sname	address
05112345	**RAHUL**	**RANCHI**
0511341	**APARANA**	**GURGAON**

Code	cname	instructor
MCS-011	**PROBLEM SOLUTION**	**NAYAN KUMAR**
MCS-012	**COMPUTER ORGANIZATION**	**ANURAG SHARMA**
MCS-014	**SAD**	**PREETI ANAND**

insructor	office
NAYAN KUMAR	**102**
ANURAG SHARMA	**105**
PREETI ANAND	**103**

Normalization involves decomposition of a relation into smaller relations based on the concept of functional dependencies to overcome undesirable anomalies



Pictorial representation of FDs

- **Trivial and Non-trivial FDs**
  - X->Y is trivial iff $Y \subseteq X$ otherwise X->Y is non-trivial
  - Non trivial FDs represent integrity constraints for the relation
- Main characteristics of F.D.'s that we use in normalization: Functional dependencies
  - have one-to-one relationship between attribute(s) on L.H.S. and R.H.S of the FD
  - holds for all time
  - are non-trivial
- **Closure of functional dependencies(F^+)**:

  Definition: Closure of a set F of FDs is defined as the set F^+ of all FDs that include F as well as all dependencies that can be inferred from F .If the functional dependency X->Y is inferred from the set of functional dependencies F then we say that F logically implies(|=) X->Y, denoted by F|=X->Y. i.e.$\mathbf{F^+=\{X\text{->}Y \mid F\models X\text{->}Y\}}$
- **Armstrong's axioms(or Armstrong's inference rules) (Armstrong-1974)**
  - ➢ If X, Y and Z be the subsets of the attributes of a relation R, then Armstrong's axioms are as follows**:**
    - IR1. (**Reflexive or self determination rule**): If $Y \subseteq X$, then X -> Y or X->X
    - IR2. (**Augmentation rule**): If X -> Y, then XZ -> YZ or XZ->Y

      (Notation: XZ stands for X U Z)
    - IR3. (**Transitive**): If X -> Y and Y -> Z, then X -> Z
- **Inference rules using Armstrong's axioms are as follows:**
  - IR4**.** (**Decomposition or Projective rule**) :If X -> YZ, then X -> Y and X -> Z
  - IR5. (**Union or additive rule**): If X -> Y and X -> Z, then X -> YZ
  - IR6. (**Psuedotransitivity rule**): If X -> Y and WY -> Z, then WX -> Z
  - IR7.(**Composition rule**): If X->Y, Z->W, then XZ->YW
  - IR8.(**Self accumulation rule**): If X->YZ, Z->W, then X->YZW

- **Proof of Armstrong's Axiom and their additional inference rules** :
  - ○ **Prof of IR1**.(i) If $Y \subseteq X$ and for any two tuples t1 and t2 that exist in some relation instance r of R such that t1[X]=t2[X] => t1[Y]=t2[Y] , then X->Y must hold in r of R. (Proved)

(ii) **Y is subset of X => X->Y**. If Y=X then **X ⊆X=> X->X**. (Proved).

- **Proof of IR2**.(By contradiction method):

(i)Let us assume that X->Y holds in a relation instance r of R but XZ->YZ doesn't hold. then there must exist two tuples t1 and t2 in r such that

(1)t1[X]=t2[X]
(2)t1[Y]=t2[Y]
(3)t1[XZ]=t2[XZ]
(4)t1[YZ] ≠ [YZ]

X->Y => t1[X]=t2[X] =>t1[Y]=t2[Y]
XZ->YZ does not hold good=> t1[XZ]=t2[XZ]=> t1[YZ] ≠t2 [YZ]

From (1) and (3) we deduce
(5)t1[Z]=t2[Z]
From (2) and (5) we deduce
(6)t1[YZ]=t2[YZ], contradicting (4)
So **X->Y=>XZ->YZ** (Proved)

- (ii) Let us assume that X->Y holds but XZ->Y doesn't hold . Then we must have:

X->Y => t1[X]=t2[X] =>t1[Y]=t2[Y]
XZ->Y does not hold good=> t1[XZ]=t2[XZ]=> t1[Y] ≠ t2[Y]
(1)t1[X]=t2[X]
(2)t1[Y]=t2[Y]
(3)t1[XZ]=t2[XZ]
(4)t1[Y] ≠t2[Y]
(2) and (4) gives the contradiction.
So **X->Y=>XZ->Y** (Proved)

- **Proof of IR3**: Let us assume that X->Y and Y->Z both holds in relation instance r of R. Then for any two tuples t1 and t2 in r of R we must have t1[X]=t2[X] =>t1[Y]=t2[Y] and t1[Y]=t2[Y]=>t1[Z]=t2[Z]. Now we can write:

t1[X]=t2[X]=>t1[Z]=t2[Z] => X->Z.
Hence **X->Y, Y->Z=>X->Z** (Proved)

- **Proof of IR4**: (1) X->YZ (Given)

(2)YZ->Y (because: Y subset YZ =>YZ->Y , by reflexivity rule/IR1)
(3)X->Y (because: X->YZ , YZ->Y=> X->Y, by transitivity rule/IR3)
(4) X->YZ (Given)
(5)YZ->Z (because: Z subset of YZ=> YZ->Z, by reflexivity rule/IR1)
(6)X->Z (because: X->YZ, YZ->Z=>X->Z, by transitivity rule/IR3)
From (3) and (6) it is clear that X->YZ=> X->Y, X->Z.(Proved).

- **Proof of IR5**: (1)X->Y (Given)

(2)X->Z (Given)
(3) X->XY (because: X->Y =>XX->XY=>X->XY, by augmentation rule applied to (1) by X/IR2 and XX=X)
(4)XY->YZ (because: X->Z =>XY->YZ, by augmentation rule applied to (2) by Y/ IR2)
(5)X->YZ (because: X->XY, XY->YZ => X->YZ, by transitivity rule applied to (3) and (4)/IR3)
So **X->Y, X->Z=> X->YZ**. (Proved)

- **Proof of IR6**: (1) X->Y (Given)

(2) WY->Z(Given)
(3)WX->WY (because: X->Y=> WX->WY, by augmentation rule/IR2)
(4)WX->Z(because: WX->WY, WY->Z=>WX->Z, by transitivity/IR3)
So **X-> Y, WY->Z => WX->Z** (Proved)

- **Proof of IR7**: (1) X->Y (Given)

(2) XZ->YZ (By augmentation rule/IR2 applied to (1) by Z)
(3)Z->W (Given)
(4)YZ->YW (By augmentation rule/IR2 applied to (3) by Y)

(5)XZ->YW (By transitivity rule/IR2 applied to (2) and (4))
So **X->Y, Z->W => XZ->YW** (Proved)

- **Proof of IR8**: Given X-> YZ => X->Y, X->Z (By decomposing rule/IR4)
  X->Z, Z->W=> X->W (By transitivity rule/IR3)
  X->YZ, X->W => X-> YZW (By union rule/IR5)
  X->YZ, Z->W => X->YZW (Proved)

- **Armstrong's Axioms(IR1,IR2, and IR3) are sound and complete**
- **Armstrong's Axioms are sound**
  Given a set of functional dependencies F specified on a relation schema R, any dependency that we can infer from F by using Armstrong's axiom(IR1 through IR3) holds in every relation r of R that satisfies the dependencies in F.
- **Armstrong's Axioms are complete**
  Using inference rules IR1 through IR3 (Armstrong's Axiom) repeatedly to infer dependencies until no more dependencies can be inferred results in the complete set of all possible dependencies that can be inferred from F.

  Or

  The set of dependencies can be determined from F by using only inference rules (IR1,IR2 and IR3)of Armstrong.
- **Solved Problems on Closure of functional dependencies(F^+):**
  Q.1. If R=(A,B,C,D) and F={A->B, A->C,BC->D} then prove that F|=A->D.
  Solution: {A->B,A->C}|=A->BC (By union rule-IR5)
  {A->BC, BC->D}|=A->D (By transitivity rule-IR3)
  => F|=A->D =>A->D is in F^+. (Proved).

  Q.2.If F={W->X, X->Y, W->XY} then find the closure of functional dependencies(F^+).
  Solution: F^+ = { W->X, X->Y, W->XY, W->W, X->X, Y->Y, W->Y}
  because W->W, X->X, Y->Y are obtained by reflexivity rule(self determination rule-IR1) and W->X, X->Y=> W->Y by transitivity rule(IR3). Then
  F^+ = F U { Inference rules obtained from F}
  ={W->X, X->Y, W->XY} U { X->X, Y->Y,W->W, W->Y}
  ={W->W, X->X, Y->Y, W->Y, W->X, X->Y, W-> XY}.(Answer).
- **Closure of attribute X under set of FD F:**
  Definition: Closure of a set of attribute(s) X with respect to the set of functional dependencies F is the set X^+ of all attributes{ A1,A2,…,Am} that are functionally determined by X based on F such that X->Ai (Ai belongs to X^+) can be calculated by repeatedly applying Armstrong's axiom(IR1,IR2,and IR3).
- Lemma: F|=X->Y <=> Y⊆ X
- ***Attribute Closure Algorithm(T*o compute the closure of attribute X (X^+) under fd F)**

```
$X^+$:= X ;// according to reflexivity
repeat
{
        old$X^+$ = $X^+$;
        for (each fd $Y \rightarrow Z \in F$ )
        {
              if ($Y \subseteq X^+$)
                  { $X^+ = X^+ \cup Z$ ;//according to   augmentation & transitivity
                  }
        }
} until ( $X^+$= old$X^+$)
```

- **Time complexity of *Attribute Closure Algorithm***
  - If a and f are the number of attributes and functional dependencies present in the set F and each FD in F involves only one attribute on R.H.S., then the inner

for loop will be executed at most ‘f’ times, one for each FD in F, and each each such execution can take the time proportional to ‘a’ to check if one set is contained in another set.

  - The order of execution of the for loop is O(af)
  - The while loop can be repeated at most ‘f’ times and so the time complexity of the attribute closure algorithm is given by O(af^2)

- **Membership algorithm(Testing if an FD is in a closure)**

  Input: A set of functional dependencies F and the functional dependency X->Y

  Output: Is X->Y$\in$ F^+ or not

  Compute X^+ using closure algorithm

  If Y $\subseteq$ X^+ then X->Y$\in$ F^+ :=true else X->Y$\in$ F^+ :=false

- **Tricks for finding the keys from a set of functional dependencies:**
  - If an attribute never appears on the R.H.S. of any FD, it must be part of the key.
  - If an attribute never appears on the L.H.S. of any FD, but appears on the R.H.S. of any FD, it must not be the part of any key.
  - If the closure of any attribute(s) X in a relation R is equal to the set of all the attributes in the relation (X->R i.e $X^+ = R$), then X is super key of the relation.
  - The super key X is said to be the candidate key of the relation R if the closures of all the proper subsets of X is not equal to the set of all the attributes in the relation .

    (i.e. Candidate key is a super key whose all proper subsets are not unique)

    ($A \subseteq X \Rightarrow A^+ \neq R$ but $A^+ \subseteq R$, where A is the proper subset of attribute X).
  - ***To find candidate keys using Closure of attributes***

    Question1: We are given a relation R(A,B,C,D,E) and the functional dependencies (FDs) amongst its attributes F = {A -> B, AC -> D, B -> E} and asked to find the candidate key(s) of R.

    Solution1: Find out the closure of each determinant (L.H.S.)of F . The determinants are {A}, {AC}, and{B}

    To see if determinate A is a super key using attribute closure algorithm

X+ := {A}	X+ is {A}
oldX+:= {A}	
for A -> B, A $\subseteq$ X+ so B is added to X+	X+ is now {A,B}
for AC -> D, AC is not a subset of X+	X+ is still {A,B}
for B -> E, B $\subseteq$ X+ so E is added to X+	X+ is now {A,B,E}
oldX+ != X+ **repeat**	
oldX+:= {A,B,E}	
for A -> B, A $\subseteq$ X+, B is already in X+	X+ is still {A,B,E}
for AC -> D, AC is not a subset of X+	X+ is still {A,B,E}
for B -> E, B $\subseteq$ X+, E is already in X+	X+ is still {A,B,E}
X+ = oldX+ **stop**	

The closure of {A} over F is {A,B,E} which is a proper subset of R. Therefore, {A} is *not* a super key of R and *cannot* be a candidate key for R.

To see if determinate AC is a superkey using attribute closure algorithm

```
X+ := AC                                      X+ is {A,C}

oldX+:= {A,C}
for A  -> B,
  A  is a subset of X+
  so B is added to X+                         X+ is now {A,C,B}
for AC -> D,
  AC is a subset of X+
  so D is added to X+                         X+ is now {A,C,B,D}
for B  -> E,
  B  is a subset of X+
  so E is added to X+                         X+ is now {A,C,B,D,E}
oldX+ != X+ repeat

oldX+:= {A,C,B,D,E}
for A  -> B,
  A  is a subset of X+, B is already in X+    X+ is still {A,C,B,D,E}
for AC -> D,
  AC is not a subset of X+                    X+ is still {A,C,B,D,E}
for B  -> E,
  B  is a subset of X+, E is already in X+    X+ is still {A,C,B,D,E}
X+ = oldX+ stop
```

The closure of {A,C} over F is {A,B,C,D,E}=R which is not a proper subset of R. {A,C} is a super key for R and may be a candidate key for R.

To see if determinate B is a superkey using attribute closure algorithm

```
X+ := B                         X+ is {B}

oldX+:= {B}
for A -> B,
  A is a not a subset of X+     X+ is still {B}
for AC -> D,
  AC is not a subset of X+      X+ is still {B}
for B -> E,
  B is a subset of X+
  so E is added to X+           X+ is now {B,E}
oldX+ != X+ repeat

oldX+:= {B,E}
for A -> B,
  A is a subset of X+           X+ is still {B,E}
for AC -> D,
  AC is not a subset of X+      X+ is still {B,E}
for B -> E,
  B is a subset of X+,
  E is already in X+            X+ is still {B,E}
X+ = oldX+ stop
```

The closure of {B} over F is {B,E} which is a proper subset of R. Therefore, {B} is *not* a superkey of R and *cannot* be a candidate key for R.

Using attribute closure algorithm , I have found one superkey {A,C} for R.

A candidate key is a superkey such that no proper subset is a superkey within the relation. So, I need to check the proper subsets of the superkey. The proper subsets of {A,C} are

- the null set
- {A}
- {C}

Examining this possible superkeys, I note that:

- A candidate key, which may be chosen to be a primary key, cannot be null.
- I have already discovered above that {A} is not a superkey for R.

So I need to consider only {C} as a possible superkey.

Using attribute closure algorithm to see if C is a superkey

`X+ := C`	`X+ is {C}`
`oldX+:= {C}`	
`for A -> B,`	
`A is a not a subset of X+`	`X+ is still {C}`
`for AC -> D,`	
`AC is not a subset of X+`	`X+ is still {C}`
`for B -> E,`	
`B is not a subset of X+`	`X+ is still {C}`
`oldX+ = X+` **stop**	

The closure of {C} over F is {C} which is a proper subset of R. Therefore, {C} is *not* a superkey of R and *cannot* be a candidate key for R.

I have discovered that no proper subsets of {A,C} are superkeys of R, so {A,C} is a candidate key for R.

I know that any attribute which is in **none** of the determinants in F *cannot* be a candidate key.

Are there any other candidate keys for R?

I know that any attribute which is in **none** of the determinants in F *cannot* be a candidate key.

So {D}, {E} and {DE} can't be the candidate keys. If we would like to see an exhaustive application of attribute closure algorithm ,we can find all the subsets of R and apply the algorithm to each subset. I have found only one candidate key {A,C} which I must select as the primary key for R. Therefore, the relational schema for R is R (A, C, [pk] B, D, E) .

Alternative method: We have : F = {A -> B, AC -> D, B -> E}. Since the attributes A and C never appears on the R.H.S of any FD in F, both A and C must be part of the key. Both attributes D and E never appears on the L.H.S. of any FD in F, but appears on the RHS of any FD, both D and E must not be the part of the key. As A , C are the part of the keys, both A and C can't be the keys but their combination {AC} may be the key. So we find the closure of the attribute {AC}as follows:

AC^+ = AC (as AC->AC, by reflexivity rule)

=ABC (as A->B, by augmentation rule)

=ABCD (as AC->D, by augmentation rule)

=ABCDE (as B->E, by augmentation rule)

=R

AC^+ =ABCDE=R => {AC} is a super key and can be a candidate key if all the proper subsets are not super keys (i.e. closure of all proper subsets of {AC} are not equal to R=ABCDE. All the proper subsets of {AC} are null set, {A}, {C}. Null set can't be a key because super key/candidate key can't be null. Now the closure of {A} is given below:

A^+ = A (as A->A)

=AB (as A->B)

=ABE (as B->E) ≠ R => {A} is not unique => {A} is not a super key.

C^+ = C (as C-> C) ≠ R => {C} is not unique => {C} is not a super key.

So $\{AC\}^+$ = R , $\{A\}^+$ ≠ R, $\{C\}^+$ ≠ R i.e.{AC} is a super key such that all of its proper subsets are not super keys => {AC} is a candidate key.(Answer).

Question2: Let R(ABCDEFGH) satisfy the following functional dependencies: {A->B, CH->A, B->E, BD->C, EG->H, DE->F}.Which of the following FD is also guaranteed to be satisfied by R?

1. BFG --> AE
2. ACG --> DH
3. CEG --> AB

Hint: Compute the closure of the LHS of each FD that you get as a choice. If the RHS of the candidate FD is contained in the closure, then the candidate follows from the given FDs, otherwise not.

Solution2:FDs: {A->B, CH->A, B->E, BD->C, EG->H, DE->F}

1. BFG --> AE ???.**Incorrect:** BFG+ = BFGEH, which includes E, but not A.

2. ACG --> DH ???.**Incorrect:** ACG+ = ACGBE, which includes neither D nor H.

3. CEG --> AB ???.**Correct:** CEG+ = CEGHAB, which contains AB.

Question 3: Which of the following could be a key for R(A,B,C,D,E,F,G) with functional dependencies {AB->C, CD->E, EF->G, FG->E, DE->C, and BC->A}.1. BDF,2. ACDF, 3. ABDFG,4. BDFG

Solution 3:F={AB->C, CD->E, EF->G, FG->E, DE->C, and BC->A}

1. BDF ???.No. BDF+ = BDF

2. ACDF ???.No. ACDF+ = ACDFEG (The closure does not include B)

3. ABDFG ???.No. This choice is a super key, but it has proper subsets that are also keys. (e.g. BDFG+ = BDFGECA)

4. BDFG ???.BDFG+ = ABCDEFG

- Check if any subset of BDFG is a key:
  - Since B, D, F never appear on the RHS of the FDs, they must form part of the key.
  - BDF+ = BDF ← Not key.
  - So, BDFG is the minimal key, hence the candidate key.

Question4: Consider R = {A, B, C, D, E, F, G, H} with a set of FDs F = {CD→A, EC→H, GHB→AB, C→D, EG→A, H→B, BE→CD, EC→B}. Find all the candidate keys of R **.**

Solution 4: F = {CD→A, EC→H, GHB→AB, C→D, EG→A, H→B, BE→CD, EC→B}

- First, we notice that:
  - EFG never appear on RHS of any FD. So, EFG must be part of ANY key of R
  - A never appears on LHS of any FD, but appears on RHS of some FD. So, A is not part of ANY key of R
  - We now see if EFG is itself a key…
  - EFG+ = EFGA ≠ R; So, EFG alone is not key
  - Checking by adding single attribute with **EFG** (except **A**):
  - BEFG+ = ABCDEFGH = R; it's a key [BE→CD, EG→A, EC→H]
  - CEFG+ = ABCDEFGH = R; it's a key [EG→A, EC→H, H→B, BE→CD]
  - DEFG+ = ADEFG ≠ R; it's not a key [EG→A]
  - EFGH+ = ABCDEFGH = R; it's a key [EG→A, H→B, BE→CD]
  - If we add any further attribute(s), they will form the superkey. Therefore, we can stop here searching for candidate key(s).
  - Therefore, candidate keys are: {BEFG, CEFG, EFGH}

Exercise 5: Consider R = {A, B, C, D, E, F, G} with a set of FDs F = {ABC→DE, AB→D, DE→ABCF, E→C}. Find all the candidate keys of R .

Solution 5: F = {ABC→DE, AB→D, DE→ABCF, E→C}

- First, we notice that:
  - G never appears on RHS of any FD. So, G must be part of ANY key of R.
  - F never appears on LHS of any FD, but appears on RHS of some FD. So, F is not part of ANY key of R
  - G+ = G ≠ R So, G alone is not a key!
  - Now we try to find keys by adding more attributes (except F) to G
  - Add LHS of FDs that have only one attribute (E in E→C):
  - GE+ = GEC ≠ R
  - Add LHS of FDs that have two attributes (AB in AB→D and DE in DE→ABCF):
  - GAB+ = GABD
  - GDE+ = ABCDEFG = R; [DE→ABCF] It's a key!
  - Add LHS of FDs that have three attributes (ABC in ABC→DE), but not taking super set of GDE:
  - GABC+ = ABCDEFG = R; [ABC→DE, DE→ABCF] It's a key!
  - GABE+ = ABCDEFG = R; [AB→D, DE→ABCF] It's a key!
  - If we add any further attribute(s), they will form the superkey. Therefore, we can stop here.
  - The candidate key(s) are {GDE, GABC, GABE}

Exercise 6: Consider R = {A, B, C, D, E} with a set of FDs F = {AB→DE, C→E, D→C, E→A}

And we wish to project those FDs onto relation S={A, B, C}.<u>Give the FDs that hold in S</u>

- **Hint: We need to compute the closure of all the subsets of {A, B, C}, except the empty set and ABC. Then, we ignore the FDs that are trivial and those that have D or E on the RHS**

Solution6: R = {A, B, C, D, E}F = {AB→DE, C→E, D→C, E→A}S={A, B, C}

- A+ = A
- B+ = B
- C+ = CEA [C→E, E→A]
- AB+ = ABDEC [AB→DE, D→C]
- AC+ = ACE [C→E]
- BC+ = BCEAD [C→E, E→A, AB→DE]
- We ignore D and E.
- So, the FDs that hold in S are:
- {C→A, AB→C, BC→A}
- (Note: BC→A can be ignored because it follows logically from C→A)

Exercise7: If R=(A, B, C,D,E), F={B->CD, D->E, B->A, E->C} then check for the followings?

1. Is B->E in F^+?, 2. Is D a key for R?, 3. Is AD a key for R?, 4. Is AD a candidate key for R?

5.Is ADE a candidate key for R?

Solution 7.1. B^+=ABCDE=R (B->CD, D->E, B->A) =>B->ABCDE=> B->E $\in F^+$
2.D^+=DEC(D->E, E->C) $\neq$ R=> D is not a key for R.
3.AD^+ =ABCDE=R (AD->B, B->CD, D->E) => AD is a key for R.
4. AD^+ =ABCDE=R, A^+= A$\neq$ R and D^+= DEC(D->E, E->C) $\neq$ R
=>AD is a candidate key for R.
5.{AD} is a key => {ADE} is a super key but not a candidate key.

Exercise8: If R(ABCDE) be a relation with a set of functional dependencies F(A->BC, CD->E, B->D, E->A) then find all candidate keys of R.

Answer8: Here all the attributes of R are present in both L.H.S. and R.H.S of F. Hence we can't compute all the candidate keys easily using attribute closure algorithm.

A^+ = A(as A->A)=ABC(as A->BC)=ABCD(as B->D)=ABCDE(as CD->E)=R
⇨ {A} is the candidate key.
CD^+ = CD(as CD->CD) =CDE(as CD->E) = CDEA (as E->A)=CDEABC(as A->BC)=R ,
C^+=C $\neq$ R and D^+ = D$\neq$ R => {CD} is a candidate key.
B^+ = B(as B->B) = BD(as B->D) $\neq$ R => {B} is not a candidate key.
E^+ = E(as E->E) = EA(as E->A) = EABC (as A->BC) = EABCD(as B->D)=R
⇨ {E} is a candidate key.
⇨ E->ABCDE
B->D => BC-> CD(by augmentation rule) .
CD-> E , E-> ABCDE => CD-> ABCDE (by transitivity rule)
Again BC->CD, CD->ABCDE => BC->ABCDE (by transitivity rule)=> BC^+=ABCDE=R
⇨ {BC} is a candidate key as B^+ = B(as B->B) = BD(as B->D) $\neq$ R and C^+=C $\neq$ R .
So {A}, {BC}, {CD} and {E} are the candidate keys of R.

Exercise9: F={ A->BC, CD->E, E->C, D->AEH, ABH->BD, DH->BC}. Test whether F|=BCD->H or not?

Answer9: BCD^+ = BCDE (as CD->E)
=BCDEAH (as D->AEH)
=ABCDEH =R =>H $\subseteq$ BCD^+ (=ABCDEH) => F|=BCD->H=> BCD->H$\in F^+$

Exercise 10:If R(ABCDEH) and F={A->BC, CD->E, E->C, D->AEH, ABH->BD, DH->BC}, then find the candidate keys of R.

Answer10: A^+ = ABC (as A->BC)≠R

CD^+ = CDE (as CD->E)

=CDAEH (as D->AEH)

=ABCDEH (as DH->BC)=R and C^+≠R, D^+=DAEH=DABCEH=ABCDEH=R

{CD} is the super key and {D} is the candidate key

Again A->BC and D->AEH => AD->ABCEH => AD^+ = ABCDEH=R

Since A^+ = ABC (as A->BC)≠R and D^+=DAEH=DABCEH=ABCDEH=R

=>{AD} is the super key

Now E->C, D->AEH => ED->CEH => ED^+=EDCAEH=ABCDEH=R and E^+=EC≠R, D^+=R =>{ ED} is a super key.

$(ABH)^+$=ABCDEH=R, $(DH)^+$=ABCDEH =R =>{ABH} and {DH} are super keys.

A Key Finding Algorithm:

```
1 .K := R                    (*K is initialized as a super key*)
2. for each A in K  do
3.{ if  (K − A )+F : = R   then  K := K− A }
```

(F is the set of functional dependencies and*$(K - A)^+_F$ *is the closure of (K-A) with respect to F*)*

Example1: If R(A,B,C) and F={A->B,B->C}. Find the key of R

Solution: K=ABC

$(K-A)^+$ = $(BC)^+$ =BC (as B->C)

$(K-A)^+$ does not contain all attributes of R(A,B,C)

i.e.BC!=ABC =>K:=ABC

$(K-B)^+$=$(ABC-B)^+$=$(AC)^+$=ABC (as A->B and B->C)

$(K-B)^+$ contains all the attributes of R =>K:=K-B=AC

$(K-C)^+$=$(AB)^+$ =ABC(as A->B and B->C)

$(K-C)^+$ contains all the attributes of R =>K:=K-C=AC-C=A

So the key of relation R(A,B,C) ,with the given functional dependencies, is the key K={A} .

- *Find a key for each of the following questions:*
    - *R = {A, B, C }, F = {A →B, B →C, C →A }*
    - *R = {D, E, F }, F = {D →E, E →D, D →F }*
    - *R = {G, H, I }, F = {G →H, G →I }*
    - *R = {C, E, J }, F = {CE →J }*
    - *R = {C, E, G }, F = {}*
    - *R = {I, L }, F = {I →L }*
    - *R(A,B,C,D,E,F,G,H,I,J) ,F={AB->C,A->DE,B->F,F->GH,D->IJ}.*

Inferring Additional Keys: If *X = {A1,..., Ai,..., Ak } be a relation schema (R, F) key, where* $X \subseteq R$*, and* $W \rightarrow Z \in F$ *(*$Z \not\subseteq W$ *,* $Z \subseteq X$ *and* $W \not\subseteq X$ *) then Y = (X- Z)* $\cup$ *W is also a relation schema (R, F) key,*

e.g. R = {A, B, C, D }, F = {AB →C, C →D, D →B }. X = AB is a key of (R, F) because $(AB)^+$ *= ABCD=R. since D →B* $\in$*F, Y = AD is another key of (R, F) because* $(AC)^+$ *= ABCD=R. since C →D* $\in$*F, Z = AC is a key of (R, F), as well.*

Covers:

Definition: F covers G if every FD in G can be inferred from F (i.e., if $G^+ \subseteq F^+$)

Equivalence of sets of functional dependencies:

Definition: F and G are equivalent if F covers G and G covers F.(i.e. $F^+ = G^+$).

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.

Example: Show that F={ A->C, AC->D, E->AD, E->H} and G={ A->CD, E->AH} are equivalent or not? **Solution**: In F, we have A->C => AA->AC (by augmentation rule)=> A->AC (as AA=A).
A->AC, AC->D => A->D (by transitivity rule). Again A->C, A->D => **A->CD** (by union rule).
In F, we have E->AD => E->A and E->D (be decomposition rule) .
Now E->A, E->H =>**E->AH** (by union rule) . Thus G={ A->CD, E->AH} is inferred from F.
In G, we have A->CD => **A-> C** and A->D (by decomposition rule).
Now, we have A->D => **AC->D** (by augmentation rule). Again in G, E->AH => E->A, **E->H.**
We have E-> A, A->D=> E->D (by transitivity rule) . Again E->A,E->D => **E->AD** (by union rule).Thus F={ A->C, AC->D, E->AD, E->H} is inferred from G. Hence F and G are equivalent.

Non redundant cover:

Definition: If we have a set of FDs F, then we say that it is nonredundant if no proper subset F’ of F is equivalent to F i.e.no F’ exists such that $F'^{+} = F^{+}$.

Non redundant cover algorithm:

Input: A set of FDs F.
Output: A non redundant cover of F.

```
  G:=F; /* Initialize G to F*/
 for each FD  X->Y in G do
           if  each FD X->Y in G do if X-> Y in G do
                    if X->Y  belongs to {F-(X->Y)}+ then F:= { F-(X-Y)};
  G:=F /* G is the non redundant cover of F*/
```

Example: If F={ A->BC, CD->E, E->C, D->AEH, ABH-> BD, DH->BC}. Find the redundant and non redundant cover of F.

Solution: We have D-> AEH => D->E (by decomposition rule)
Now D->E => CD-> E (by augmentation rule)
So **CD->E is derived from D->AEH**.
Again D->AEH(given)=> D->A (by decomposition rule)
Now D->A, A->BC(given) => D->BC (by transitivity rule)
So D->BC=> DH->BC (by augmentation rule) .

Hence **DH->BC is derived from D->AEH and A->BC**.
Redundant FDs ={ CD->E, DH->BC}
Non redundant FDs F:={ F-(CD->E, DH->BC)}={ A->BC, E->C, D->AEH, ABH-> BD}

Extraneous attribute:

- An attribute A of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies.
- If F is a set of functional dependencies and the functional dependency X->Y in F .
  - Attribute A is extraneous in X if A∈ X , and F|=(F-{X->Y})U{(X-A)->Y}
  - Attribute A is extraneous in Y if A∈Y, and the set of functional dependencies (F-{X->Y})U{X->(Y-A)} |=F

Canonical cover F_C : Canonical cover F_C is a set of dependencies such that F logically implies all dependencies in F_C and F_C logically implies all dependencies in F. (F|= F_C and F_C |= F).

Properties of F_C:

- No functional dependency in F_C contains an extraneous attribute.
- Each left side of a functional dependency in F_C is unique.(i.e. there are no functional dependency in X1->Y1 and X2->Y2 in F_C such that X1=X2).

Algorithm of Canonical cover for functional dependencies (FDs)F:

```
 repeat
    Use the union rule to replace any fds in F of the form X1->Y1 and X1-> Y2 with X1->Y1Y2
     Find a functional dependency X->Y with an extraneous attribute either  in X  or Y
     If an extraneous attribute is found, delete it from X->Y
  Until  F doesn’t change.
```

Problem: If R(A,B,C), F={ A->BC, B->C, A->B, AB->C} . Compute the canonical cover for F.

- ➢ Solution:
- ➢ There are two functional dependencies with the same set of atributs on the left side of the arrow A : A->BC,A->B and we combine these two FDs into A->BCB => A-> BC (as BB=B).
  A->BC => A->B, A->C (By decomposition rule)
- ➢ A is extraneous in AB->C since B->C =>AB->C (By augmentation rule).
- ➢ C is extraneous in A->BC since A->B, B->C => A->C (By transitivity rule)
  and A->B, A ->C => A->BC (By union rule)

- Thus our canonical cover is F_C ={ A->B, B->C}.

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 and still have a set of dependencies that is equivalent to F.

Minimal cover of a set of FDs: A minimal cover of a set of functional dependencies E is a minimal set of dependencies(in the standard canonical form and without redundancy) that is equivalent to E.

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets

Algorithm for finding a minimal cover F for a set of functional dependencies E

1. Set F:=E.
2. Replace each functional dependency X->{A1,A2,…An} in F by n functional dependencies X->A1, X->A2,…,X->An (i.e. each FD in F is simple)
3. for each functional dependency X->A in F
   for each attribute B that is an element of X
   if{{F-{X->A}}U{(X-{B})->A} is equivalent to F, then replace X-> A with (X-{B})->A in F.
4. For each remaining functional dependency X->A in F if {F-{X->A}} is equivalent to F, then remove X->A from F.

Problem: Find the minimal cover of the following functional dependencies:

F={ A->BC, CD->E, E->C, D->AEH, ABH-> BD, DH-> BC}

Solution: Convert all FDs into simple FDs by using decomposition rule(IR4):

A->BC=> A->B, A->C
D->AEH=>D->A, D->E, D->H
ABH->BD =>ABH->B, ABH->D
DH->BC=>DH->B, DH->C

So F={ A->B, A->C, CD->E, E->C, D->A, D->E, D->H, ABH->B, ABH->D, DH->B, DH->C}

But we have already proved that CD->E and DH->BC are redundant FDs. So remove them from F. Hence F={ A->B, A->C, E->C, D->A, D->E, D->H, ABH->B, ABH->D}.

Consider ABH->B and ABH->D.

A->B=>AH->B (By IR2: Augmentation rule)
=>ABH->BB(
=>ABH->B

=>ABH->B and it is redundant & can be derived from A->B and hence remove it from F.

. AH->D =>ABH->D (By augmentation rule) and it is redundant and can be derived from AH->D and hence remove it from F. So the minimal cover of F is given by

F= { A->B, A->C, E->C, D->A, D->E, D->H, AH->D}.

Functional dependencies and Keys:

Definition: If R(A1A2A3…An) be a relation schema, F be the set of functional dependencies, then a key 'K' of R is a subset of R such that K->A1A2A3…An $\in F^+$ and for any Y$\subseteq$ A1A2A3…An doesn't belongs to F^+.

Full functional dependency(FFD):

Definition : If X and Y are attributes of a relation R then Y is fully functionally dependent on X if Y
is functionally dependent on X but not on any proper subset of X.

or

A FD X -> Y is FFD where removal of any attribute from X means the FD does not hold any more

i.e. Y is FFD on X if X->Y but Z ($\subseteq$ X) doesn't determine Y.

Example: 1) SUPPLIER

SNO	SNAME	STATUS	CITY
s_1	Suneet	20	Quadian
s_2	Ankit	10	Amritsar
s_3	Amit	30	Amritsar

(SNO,STATUS)->CITY but SNO->CITY , STATUS->CITY

{CITY} is FD on (SNO,STATUS) but {CITY} is not FFD on {SNO,STATUS}

because {CITY} is FD on {SNO} and {CITY}.

Example:2) SHIPMENT

SNO	PNO	QTY
S1	P1	270
S1	P2	300
S1	P3	700
S2	P1	270
S2	P2	700
S3	P2	300

{QTY} is FFD on (SNO,PNO) because

(i) (SNO,PNO)->QTY
(ii) (a) SNO doesn't determine QTY and (b)PNO doesn't determine QTY

Prime and nonprime attribute:

- A **Prime attribute** must be a member of *some candidate key/primary key.*
- A **Nonprime attribute** is not a prime attribute— that is, it is not a member of any candidate key.
- Example: R(ABCDEH), F={ A->BC, CD->E, E->C, AH->D}

=> $(AH)^+$ = AHD (as AH->D)
=ABCHD (as A->BC)
=ABCHDE(as CD->E)
=ABCDEH=R
=>AH->ABCDEH
=>{AH} is the candidate key of R
=> A and H are prime attributes whereas B,C,D,E and H are nonprime attributes.

Partial dependency:

Definition: If R be a relation schema with the functional dependencies F defined on the attributes of R and K as a candidate key, if X is a proper subset of K ($X \subseteq K$) and if F|=->A then A is said to be partially dependent on K.

i.e. Attribute Y is partially functionally dependent on X1X2(=K=key) if X1->Y or X2-> Y.

i.e. If Any attribute A is functionally dependent on part of the key(K1) (i.e.K1->A), then A is partially functionally dependent on the key.

i.e. A FD X -> Y is partial dependency where removal of any attribute from X means the FD holds any more.

Example: R=(A,B,C,D), F={AB->C, B->D}
(AB)+ = ABC (as AB->C)

= ABCD(as B->D)=R

AB is the key of R.

But given B->D and B ⊆ AB(=key)

D is partially functionally dependent on the key AB.

Multi valued dependency(MVD):

Definition: If X, Y and Z are three attributes of a relation R such that for each value of X there is a set of values for Y (X->->Yi.e.X multi determines Y) and set of values for Z(X->->Zi.e. Y multi determines Z) then we say that X multi determines Y and Z denoted by X->->Y|Z or Y and Z are multi valued dependency on X, where the set of values for Y and Z are independent of each other.

Or

- A multi valued dependency X->>Y specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: if two tuples t1 and t2 exist in r such that t1[X]=t2[X], then two tuples t3 and t4 should also exist in r with the following properties:
    - t3[X]=t4[X]=t1[X]=t2[X]
    - t3[Y]=t1[Y] and t4[Y]=t2[Y]
    - t3[Z]=t2[Z] and t4[Z]=t1[Z]

Trivial and nontrivial MVD:

Definition: A MVD X->-> Y in R is called a trivial MVD if (a) Y⊆ X or (b) XUY=R.

A MVD X->-> Y in R is called a nontrivial MVD if neither(a)nor (b) is satisfied.

- A trivial MVD doesn't specify a constraint on a relation while a non trivial MVD does specify a constraint.
- There are two kinds of trivial MVDs:
  (a) X->-> Null set , where null set contains empty set of attributes.
  (b)X->->A-X, where A comprises all the attributes in a relation.
- Relations containing nontrivial MVDs tend to be all key relations i.e their key is all their attributes taken together.

Theory of MVD (or inference rules or FD and MVD): If X,Y,Z and W are subsets of relation schema R={A1,A2,…,An}, then we have the followings rules for FD and MVD.

1) IR1-Reflexivity for FDs: Y⊆ X =>X->Y.
2) IR2-Augmentation for FDs: X->Y => XZ->YZ or XZ->Y
3) IR3-Transitivity for FDs: X->Y, Y->Z =>X->Z
4) IR4-Complementation for MVDs: X->->Y=>X->->(R-XUY)
5) IR5-Augmentaion for MVD:X->Y, Z⊆W =>WX->->YZ
6) IR6-Transitivity for MVDs: X->->Y, Y->->Z =>X->->(Z-Y)
7) IR7-Replication rule for FD to MVD: X->Y=>X->->Y
8) IR8-Coalescence rule for FDs and MVDs: If X->->Y and there exist W with the properties that
   (a)W∩Y= Null set,(b) W->Z, and (c)Y⊆ Z, then X->Z.
9) IR9-Reflexivity for MVDs: Y⊆ X =>X ->->Y or X->->X.
10) IR10-Intersection for MVDs: X->->Y, X->->Z=>X->->Y∩Z
11) IR11-Pseudo-transitivity for MVDs: X->->Y, YW->->Z=>XW->->(Z-WY)
12) IR12-Mixed(Pseudo) transitivity for MVDs: X->->Y, XY->->Z=>X->->(Z-Y)
13) IR13-Union rule for MVDs:X->->Y,X->->Z=>X->->YZ
14) IR14-Difference rule for MVDs: X->->Y, X->->Z =>X->->(Y-Z), X->->Z-Y
15) IR15-Decomposition/Projectivity rule for MVDs: X->->Y, X->->Z=>X->->(Y∩ Z),
    X->->(Y-Z)and X->->(Z - Y).
    - If D denotes a set of FDs and MVDs , then the closure D^+ of D is the set of all FDs and MVDs logically implied by D.
    - The list of inference rules for FDs and MVDs are sound and complete.
    - Sound rules do not generate any dependencies that are are not logically implied by D.
    - Complete rules allow us to generate all dependencies in D^+.

Question on MVD: If R=(A,B,C,G,H,I), D={A->->B, B->->HI, CG->->H}. Find the several members of closure of D(D^+).

Answer: (i)A->->B=>A->->R-B-A=CGHI (By complementation rule of MVD)
=> A->->CGHI.
(ii) A->->B, B->->HI =>A->->HI-B(By transitivity rule of MVD)
=>A->->HI. (HI-B=HI)
(iii) B->->HI , H⊆HI and CG->H and CG ∩HI =null set => B->H
(By Coalescence rule of MVD)
(iv)A->->CGHI, A->>HI =>CGHI-HI=CG i.e. A->CG (By difference rule of MVD)

Universal relation schema(R): The universal relational schema R={A1,A2,…,An} includes all the attributes of the database and every attribute name is unique.

Decomposition: Decomposition is the process of splitting a relation into its projections that will not be disjoint. The decomposition (D) of a universal relation schema R={A1,A2,A3,..An) is its replacement by a set of relation schema s D={ R1, R2,…,Rn} such that Ri ⊆ R(1≤ i≤m) and R1UR2UR3U…=R(**attribute preservation** condition of decomposition).

- Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.
- When required , the DBA decides to decompose an initial set of relation schemes.

Desirable properties of decomposition:

- Attribute preservation
- Lossless-join decomposition
- Dependency preservation
- Lack of redundancy

Types of decomposition:

- Lossy (or Lossy join) decomposition
- Lossless(or non additive or non loss) join decomposition

Lossy(or Lossy join) decomposition:

- It is a property of decomposition , which ensures that spurious(extra) tuples are generated when relations are reunited through a natural join operation.
- The decomposition R(A,B,C) into R1 and R2 is lossy when the natural join of R1 and R2 does not yield the same relation as in R but yield spurious(extra) tuple(s).
- **Example**: In table1 i.e. R(ABC), A->B, C->B. R(ABC) is decomposed to R1(AB) and R2(BC) and then R3(ABC) is the natural join of R1(AB) and R2(BC) and in R3(ABC) neither B->A nor B->C is true. Also R3(ABC) ≠ R(ABC) because R3 contains some spurious(extra) tuples and so the decomposition of R(ABC) into R1(AB) and R2(BC) is a lossy decomposition .

R: ABC		
A	B	C
A1	B1	C1
A3	B1	C1
A2	B2	C3
A4	B2	C4

R1:AB

A	B
A1	B1
A3	B1
A2	B2
A4	B2

R2:BC	
B	C
B1	C1
B1	C2
B2	C3
B2	C4

R3:ABC		
A	B	C
A1	B1	C1
A1	B1	C2
A3	B1	C1
A3	B1	C2
A2	B2	C4
A2	B2	C3
A4	B2	C3
A4	B2	C4

Lossless (or non additive or non loss) join decomposition:

- The word loss in lossless refers to loss of information.
- It is a property of decomposition which ensures that no spurious tuples are generated when relations are reunited through a natural join operation.
- The decomposition of R(X,Y,Z) into R1(X,Y) and R2(X,Z) is lossless if for attributes X, common to both R1 and R2, either X->Y or X->Z.
- A decomposition D={R1,R2,R3,…,Rm} of a relation R has lossless(non additive) join property with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds: natural join of the projections of all the relations in D from R =R.
- Example: Relation R(ABC) decomposed into R1(AB) and R2(BC) is lossless because the natural join of R1 and R2 = R3=R

R: ABC		
A	B	C
A1	B1	C1
A2	B2	C2
A3	B2	C1
A4	B1	C2

R1:AB	
A	B
A1	B1
A2	B2
A3	B2
A4	B1

R2:BC	
B	C
B1	C1
B2	C2
B3	C1
B4	C2

R3:ABC		
A	B	C
A1	B1	C1
A2	B2	C2
A3	B2	C1
A4	B1	C2