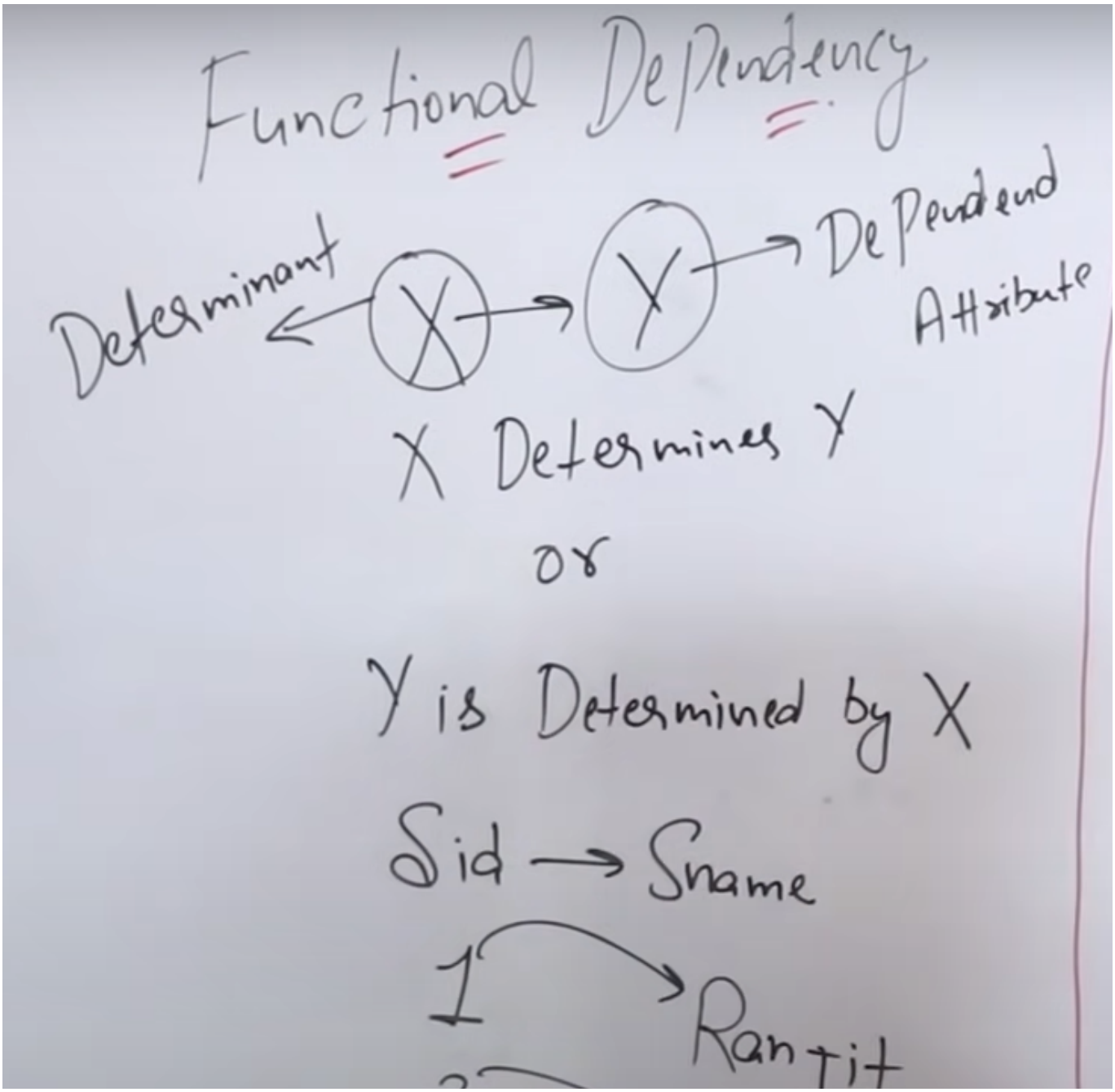
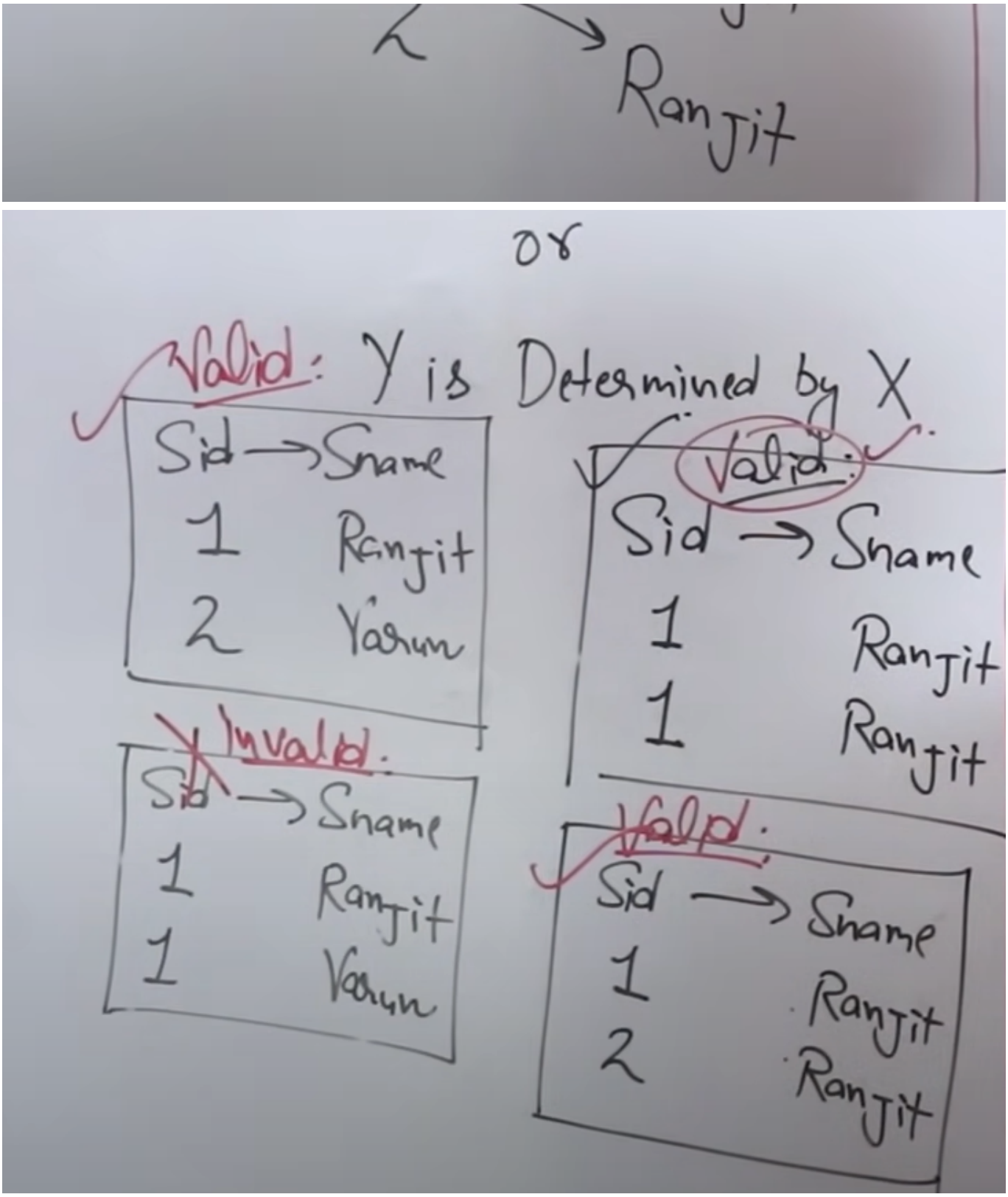
Functional Dependency

FUNCTIONAL DEPENDENCY

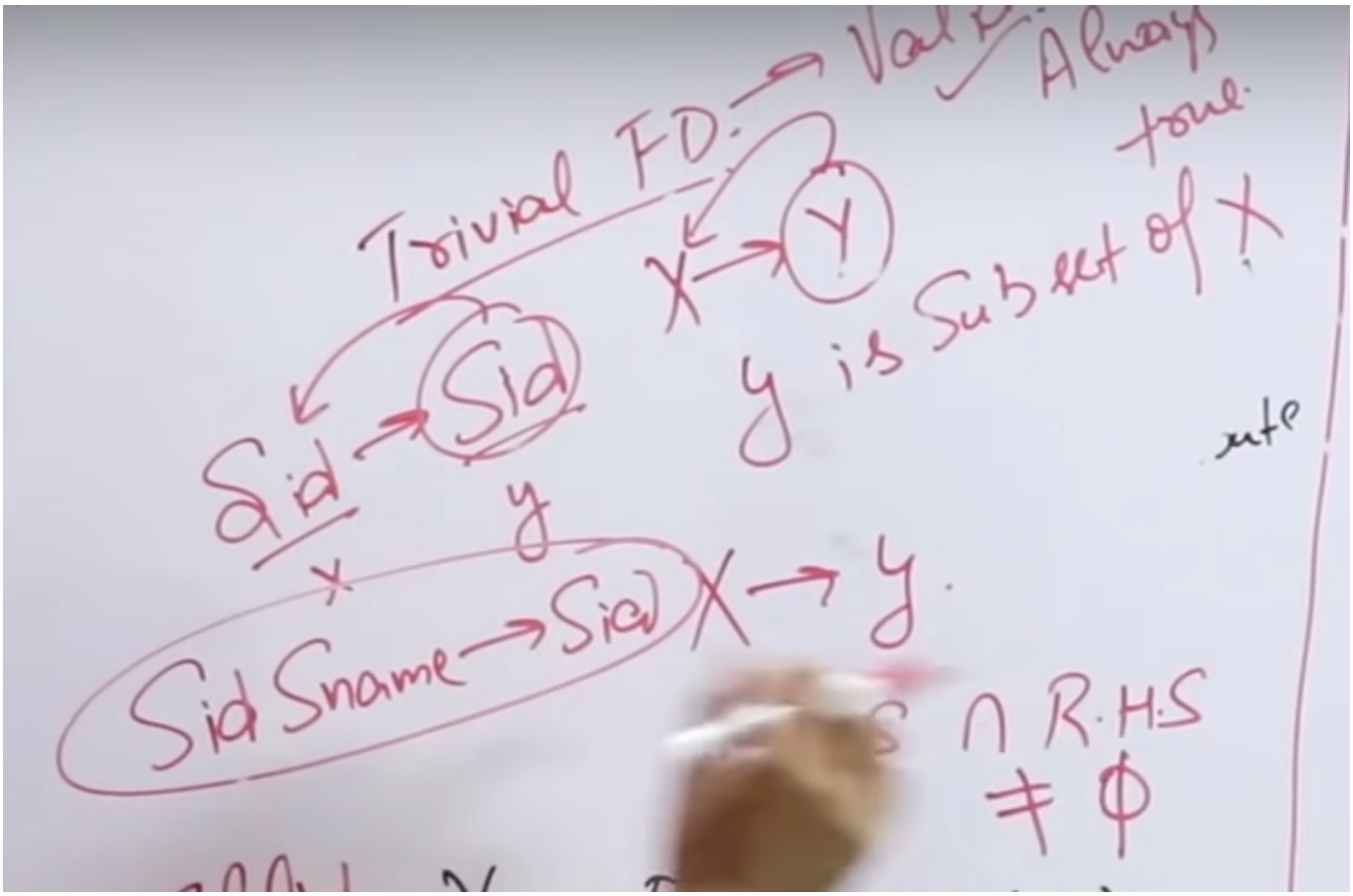
- A formal tool for analysis of relational schemas.
- In a Relation R, if 'α' ⊑ R AND 'β' ⊑ R, then attribute or a Set of attribute 'α' Functionally derives an attribute or set of attributes 'β', iff each 'α' value is associated with precisely one 'β' value.
- For all pairs of tuples t_1 and t_2 in R such that
  - If $T_1[\alpha] = T_2[\alpha]$
  - Then, $T_1[\beta] = T_2[\beta]$

X	Y	Z
1	4	2
1	4	3
2	6	3
3	2	2

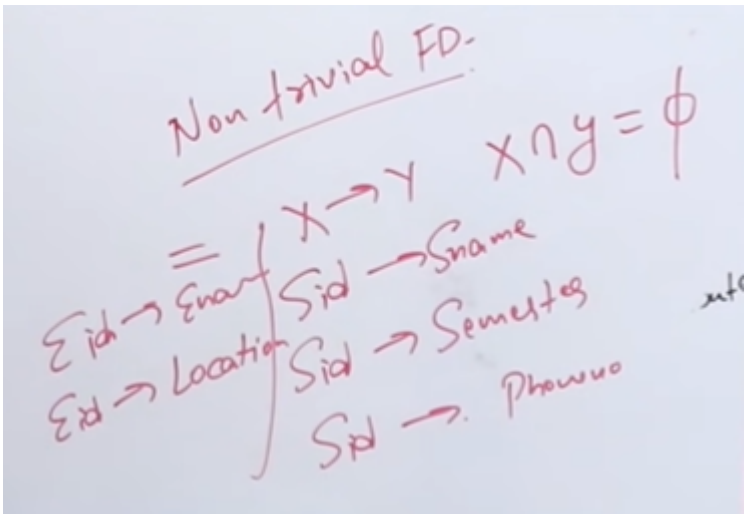



Ranjit
or
Valid: Y is Determined by X
Sid → Sname
1 Ranjit
2 Varun
Invalid.
Sid → Sname
1 Ranjit
1 Varun
Valid:
Sid → Sname
1 Ranjit
1 Ranjit
Valid:
Sid → Sname
1 Ranjit
2 Ranjit

Trivial



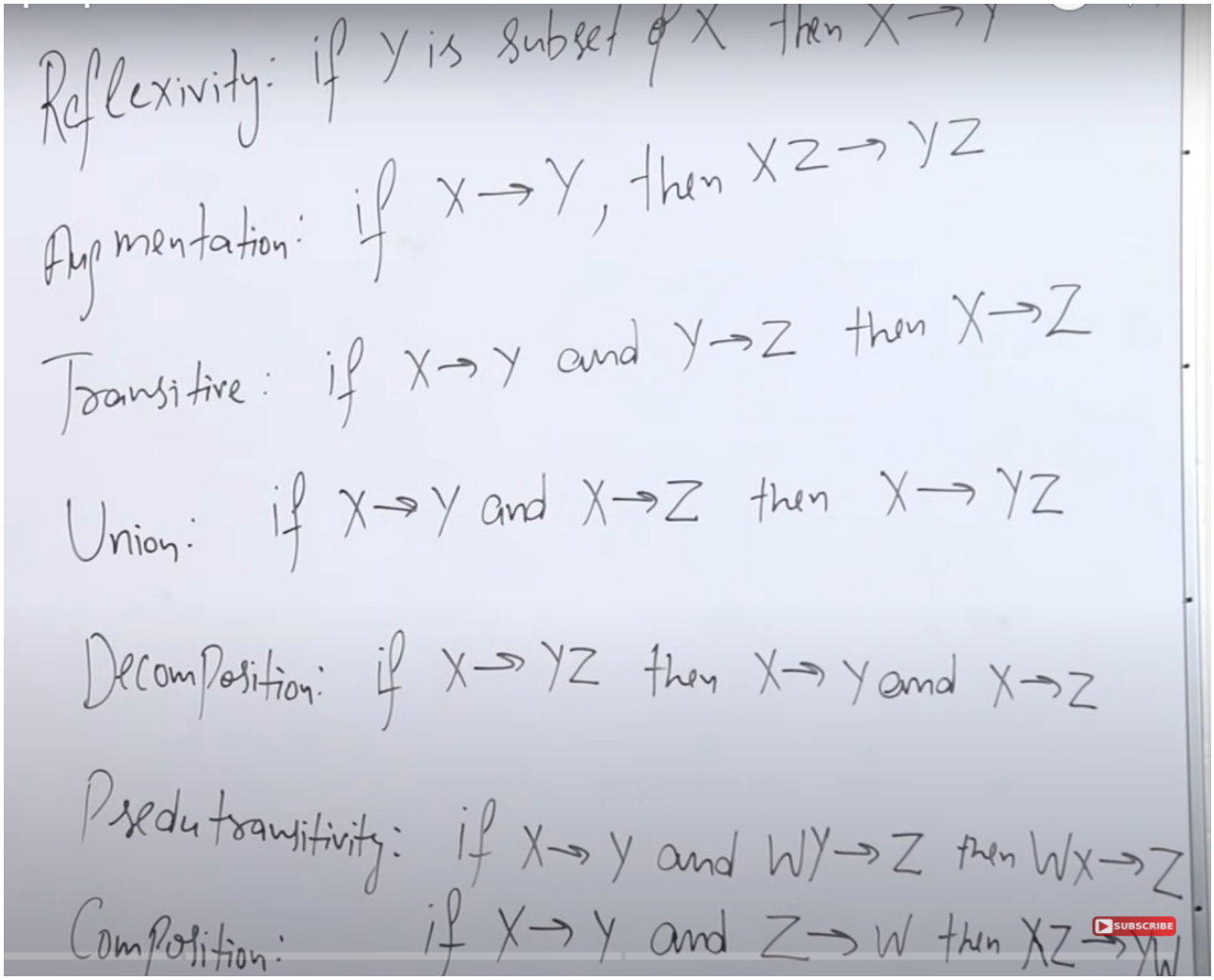
Non-Trivial

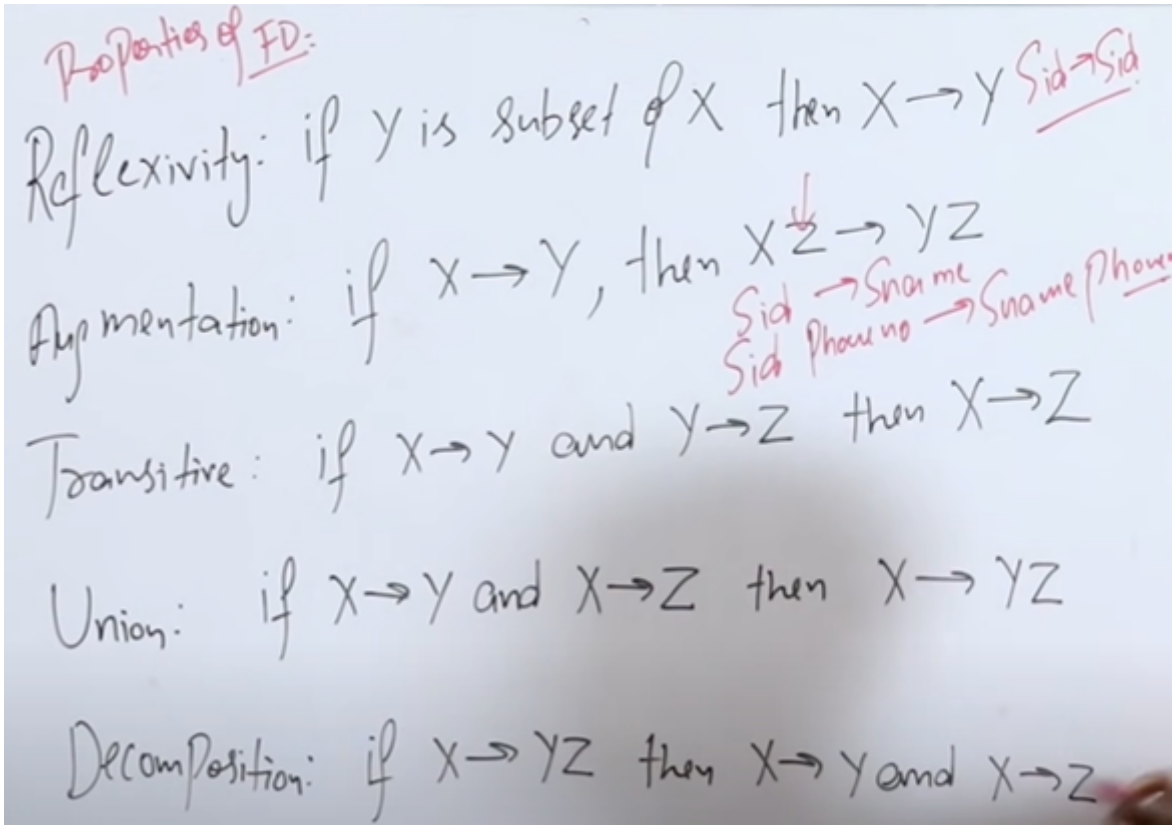


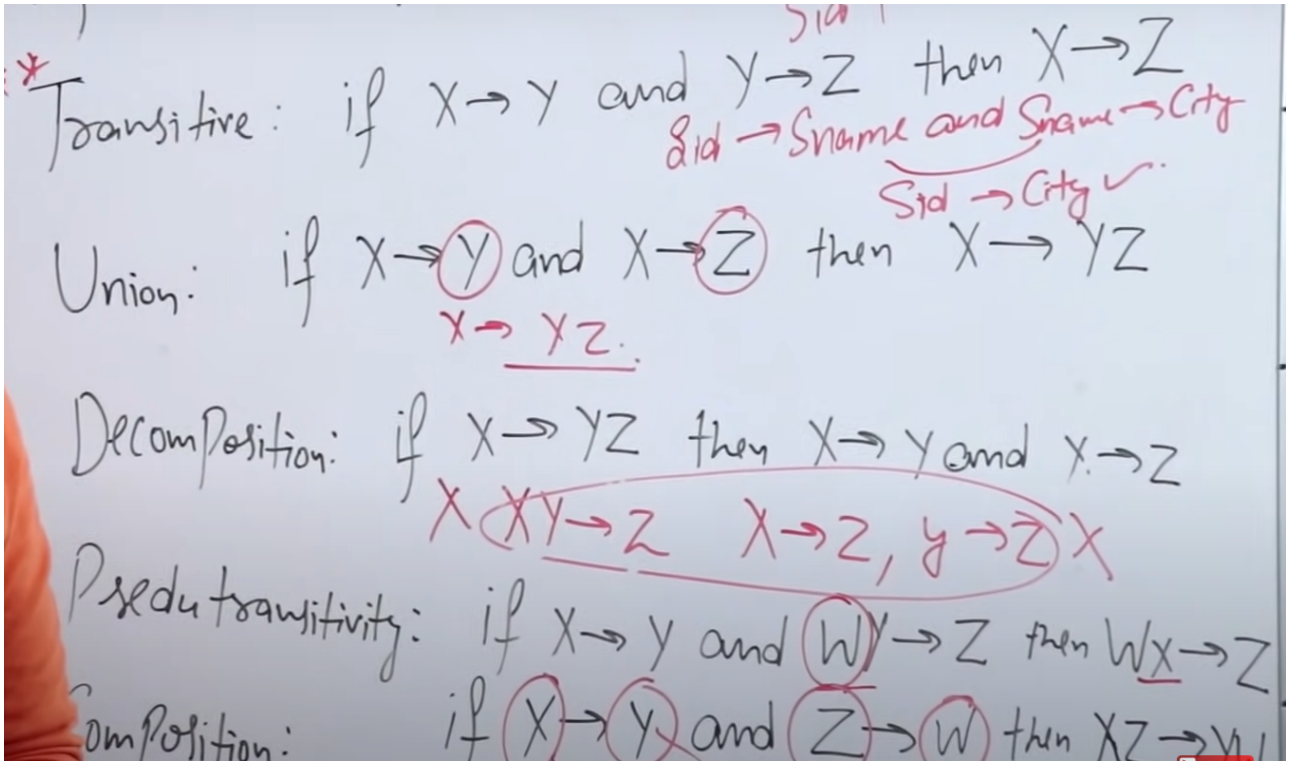
Properties of FD / Armstrong axioms

- Reflexivity: If Y is a subset of X, then X -> Y
- Augmentation: If X -> Y, then XZ -> YZ
- Transitivity: If X -> Y and Y -> Z, then X -> Z From these rules, we can derive these secondary rules-
- Union: If X -> Y and X -> Z, then X -> YZ
- Decomposition: If X -> YZ, then X -> Y and X -> Z
- Pseudo transitivity: If X -> Y and WY -> Z, then WX -> Z

- Composition: If X -> Y and Z -> W, then XZ -> YW

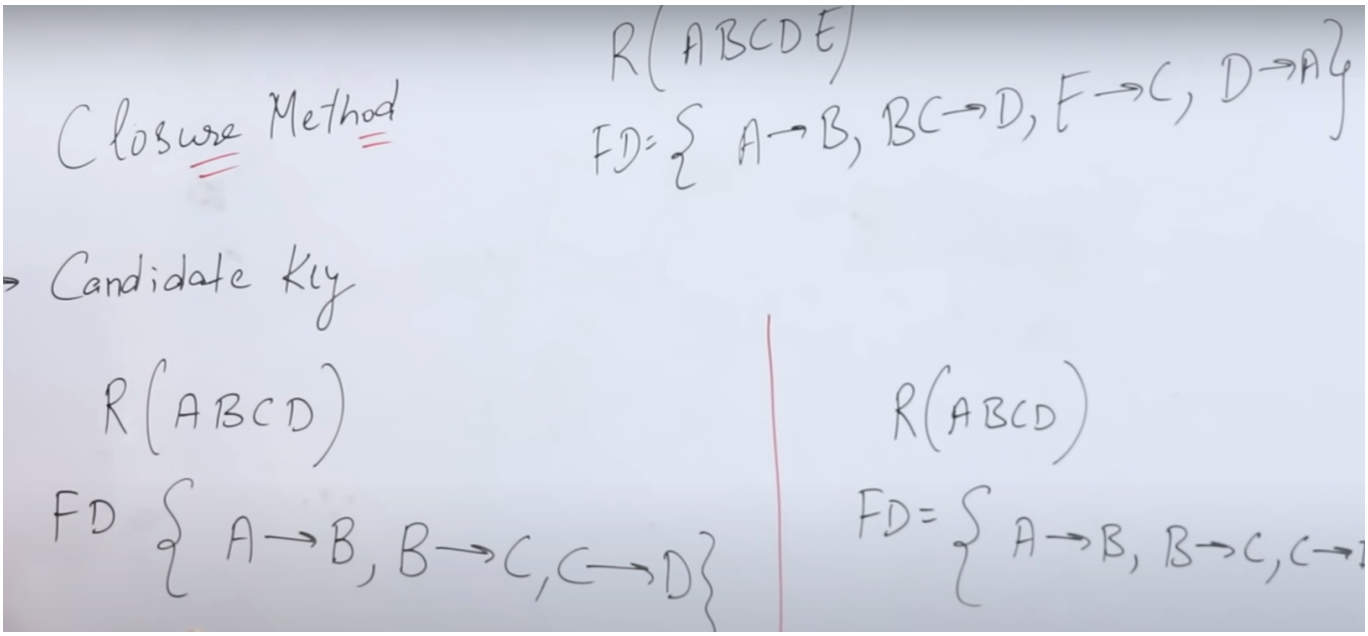




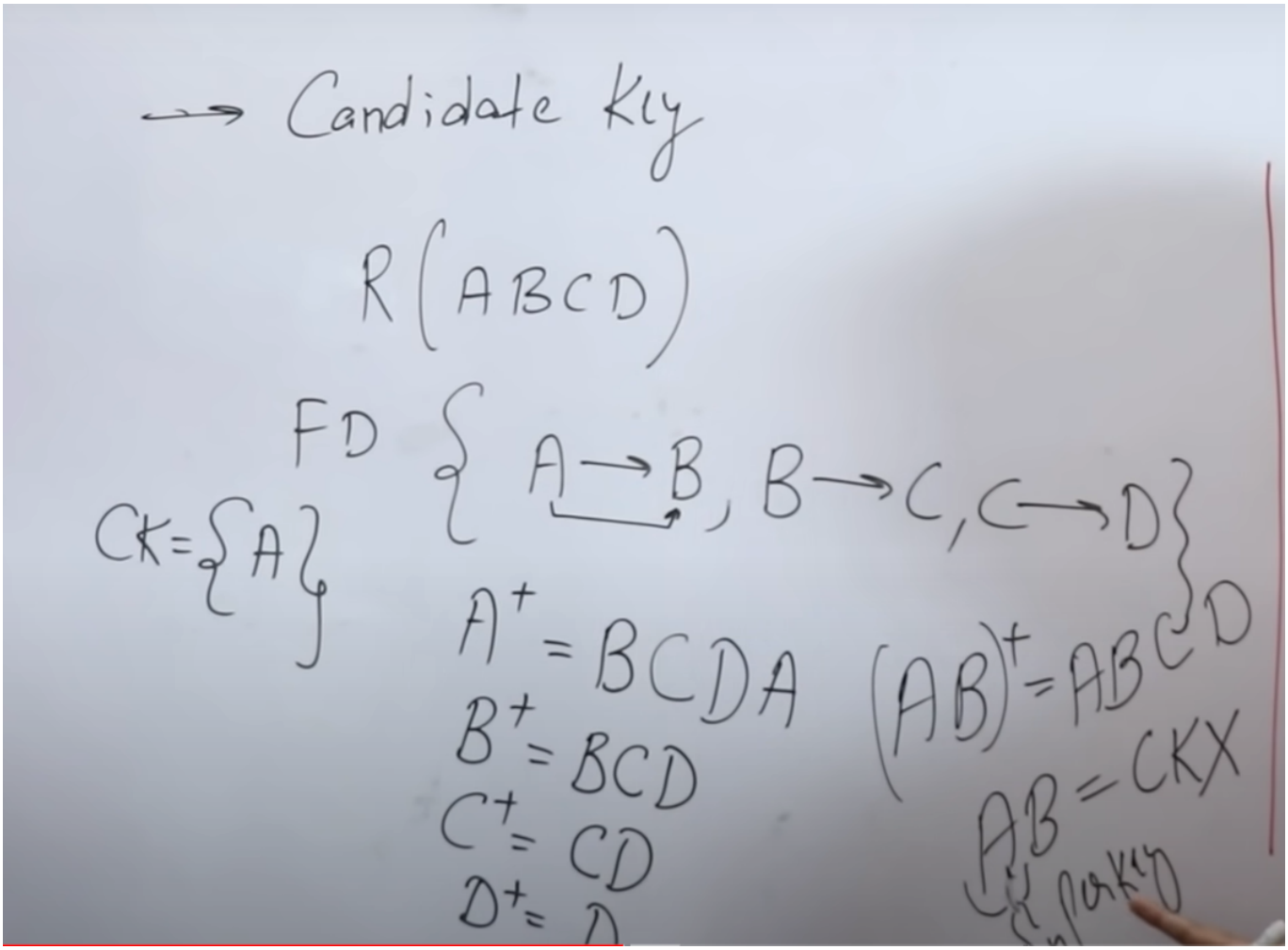


Finding Closures

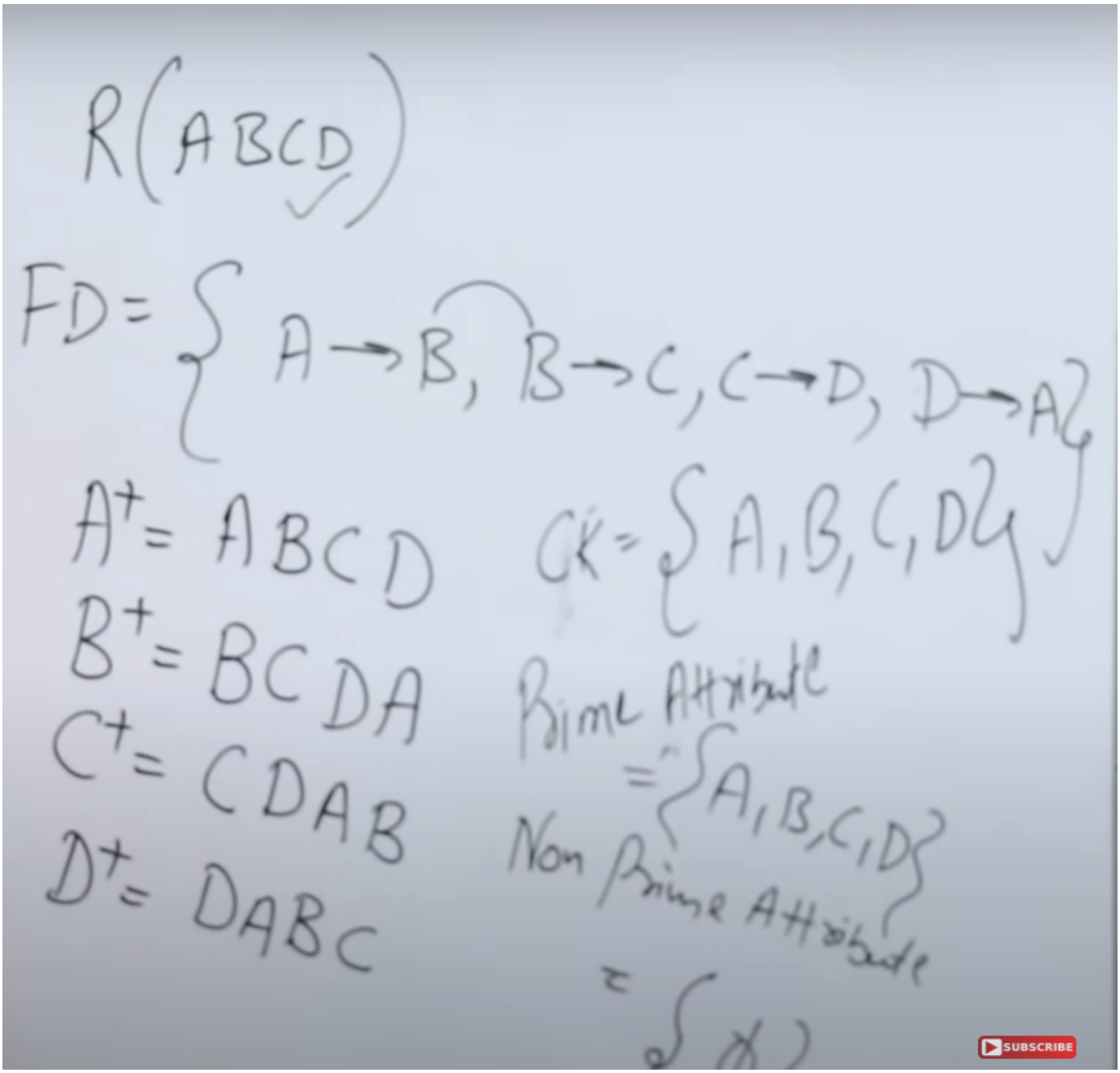
Q - Find CK(unique) So to get uniquess/ If a attribute/set of attributes uniquely identifies all attributes : Closure


Closure Method
R(ABCDE)
FD={ A→B, BC→D, E→C, D→A }
→ Candidate Key
R(ABCD)
FD { A→B, B→C, C→D }
R(ABCD)
FD={ A→B, B→C, C→

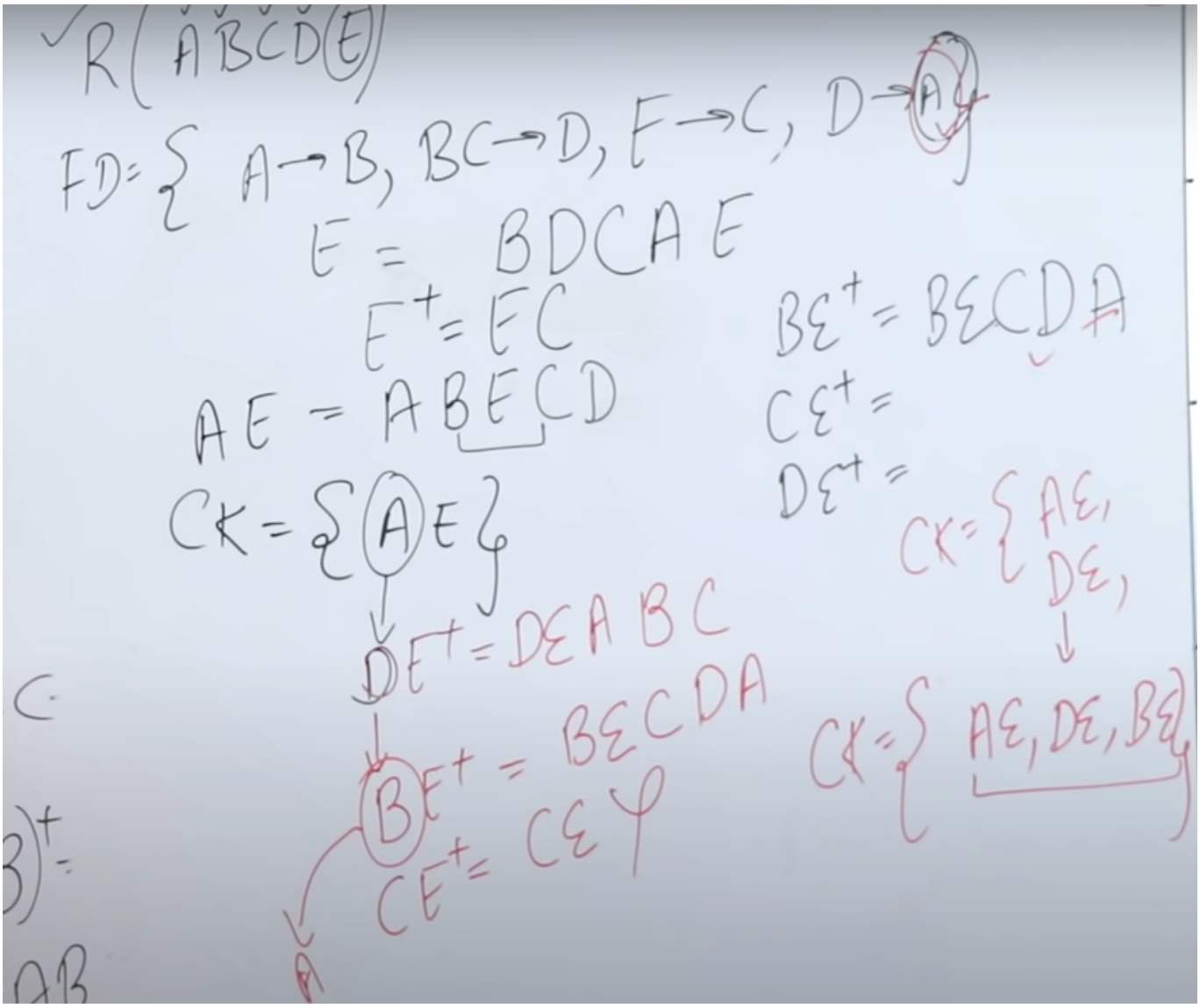
Ex 1


→ Candidate Key
R(ABCD)
FD { A→B, B→C, C→D }
CK={A}
A+ = BCDA
B+ = BCD
C+ = CD
D+ = D
(AB)+ = ABCD
AB = CKX
SK

Ex 2


R(ABCD)
FD = { A → B, B → C, C → D, D → A }
A+ = ABCD
B+ = BCDA
C+ = CDAB
D+ = DABC
CK = { A, B, C, D }
Prime Attribute = { A, B, C, D }
Non Prime Attribute = { ∅ }
SUBSCRIBE

Ex 3



GATE Q

GATE Question: Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, N} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R? (GATE-CS-2014)

A. {E, F}

B. {E, F, H}

C. {E, F, H, K, L}

D. {E}

EFH -> EFGHIJKLMN

EF -> G
F->IJ
EH->KL
K->M
L->N

{A -> B, A -> C, CD -> E, B -> D, E -> A}

Which of the following functional dependencies is NOT implied by the above set? (GATE IT 2005)

A. CD -> AC

B. BD -> CD

C. BC -> CD

D. AC -> BC

CD+ -> CDEAB

BD+ -> BD

BC+ ->BCDEA

AC+ -> ACBDE

GATE Question: Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–> D, E–>C, D–>A}. What are the candidate keys of R? [GATE 2005]

(a) AE, BE

(b) AE, BE, DE

(c) AEH, BEH, BCH

(d) AEH, BEH, DEH

EH ->

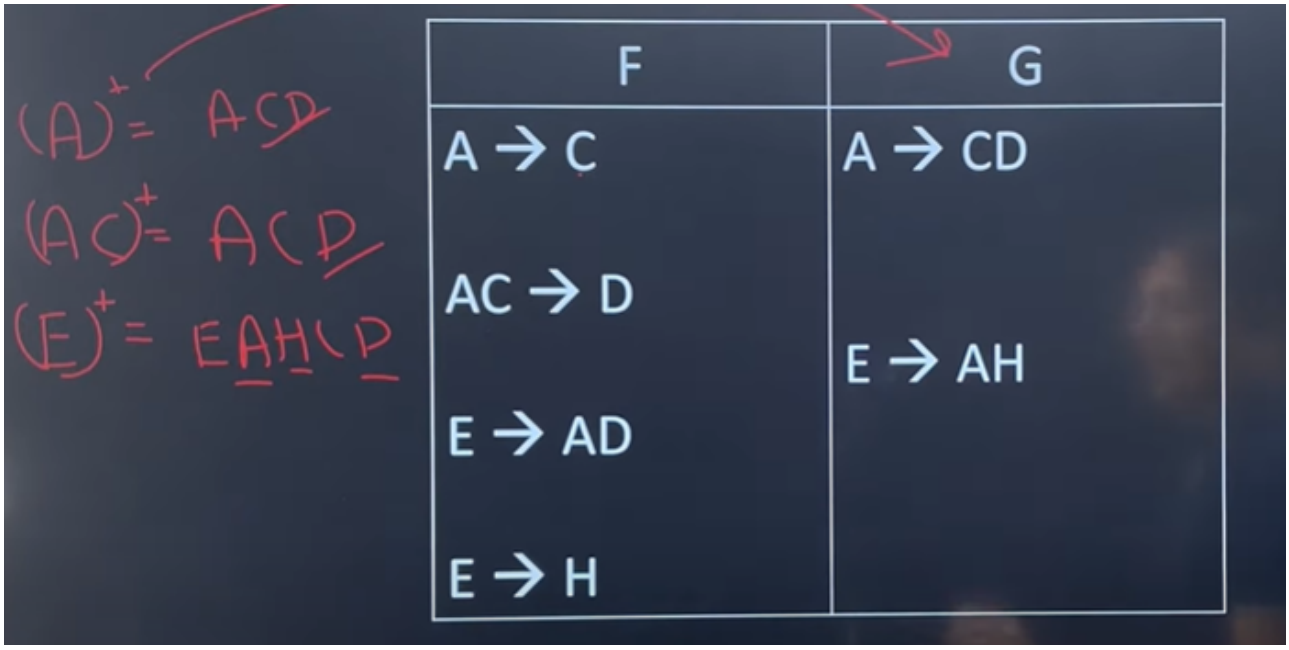
Equivalence of Two FD sets

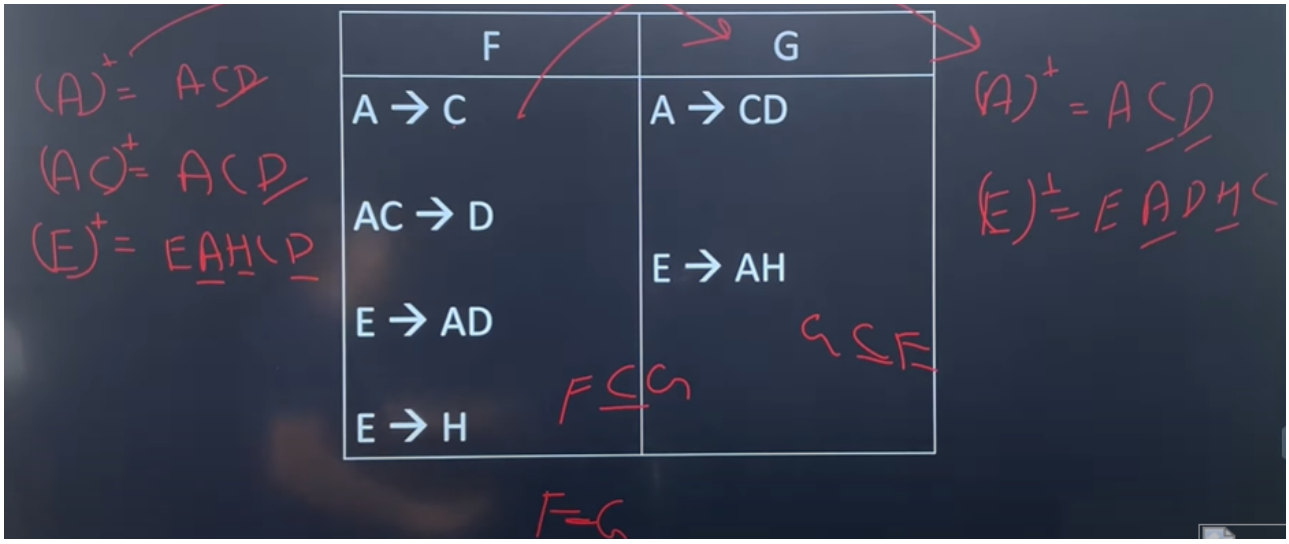
- Two FD sets F1 and F2 are equivalent if – F1+ = F2+ Or F1 ⊑ F2+ and F2 ⊑ F1+

Steps :

1. Find F's left hand side attribute's closure using G Set
2. Check F's FD with the closure you got
3. Find G's left hand side attribute's closure using F Set
4. Check G's FD with the closure you got

5. If you can get all the FD with the closures, they are said to be equivalent.



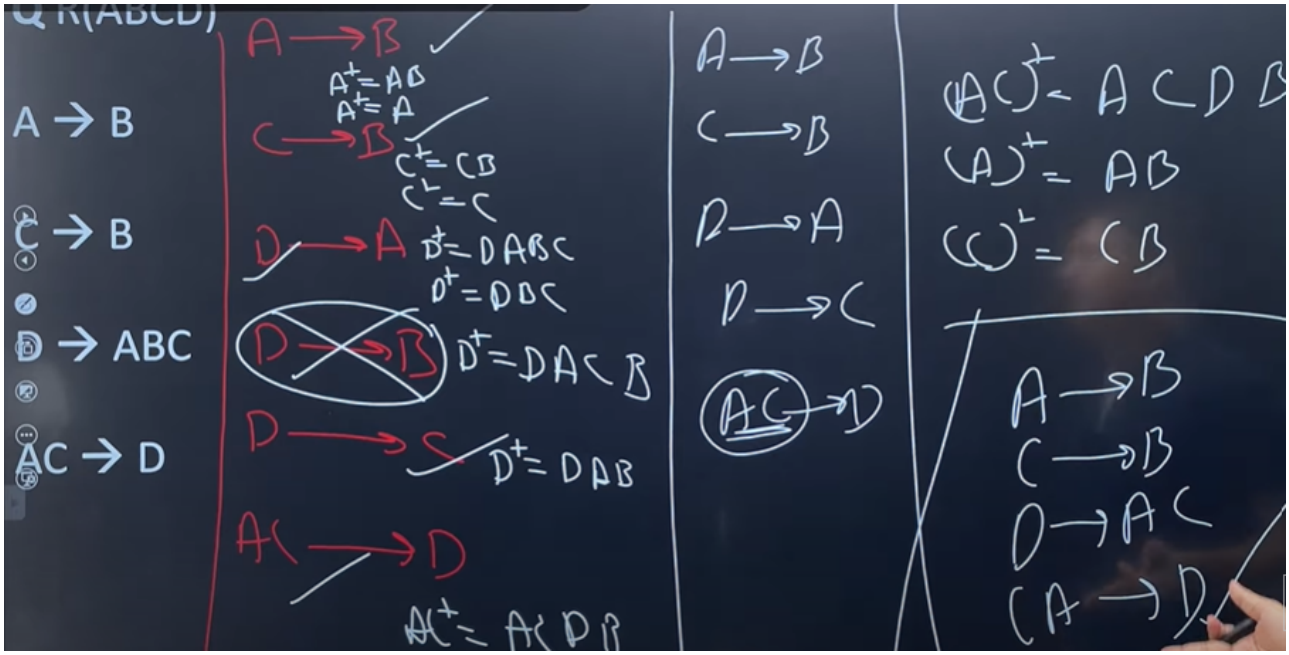


Canonical Cover

To find the MINIMAL COVER / CANONICAL COVER / IRREDUCIBLE SET

- A **canonical cover** (also known as a minimal cover) for a set of functional dependencies in a database is a minimal set of functional dependencies that is equivalent to the original set, but with redundant dependencies and extraneous attributes removed. It is used in the normalization process of database design to simplify the set of functional dependencies and to find a good set of relations.
- There may be any following type of redundancy in the set of functional dependencies: -
  - Complete production may be Redundant.
  - One or more than one attributes may be redundant on right hand side of a production.
  - One or more than one attributes may be redundant on Left hand side of a production.

- Ex 1 R(A,B,C,D) A -> B C -> B D -> ABC : BREAK AC -> D : DO NOT BREAK



Understand Keys again...

Super key

- Set of attributes using which we can identify each tuple uniquely is called Super key.
- Let X be a set of attributes in a Relation R , if X^+(Closure of X) determines all attributes of R then X is said to be Super key of R .
- There should be at least one Super key in every relation.

Candidate key

- Minimal set of attributes using which we can identify each tuple uniquely is called Candidate key. A super key is called candidate key if it's No proper subset is a super key. Also called as **MINIMAL SUPER KEY.**
- There should be at least one candidate key.

Prime attribute - Attributes that are member of at least one candidate Keys are called Prime attributes.

Primary key

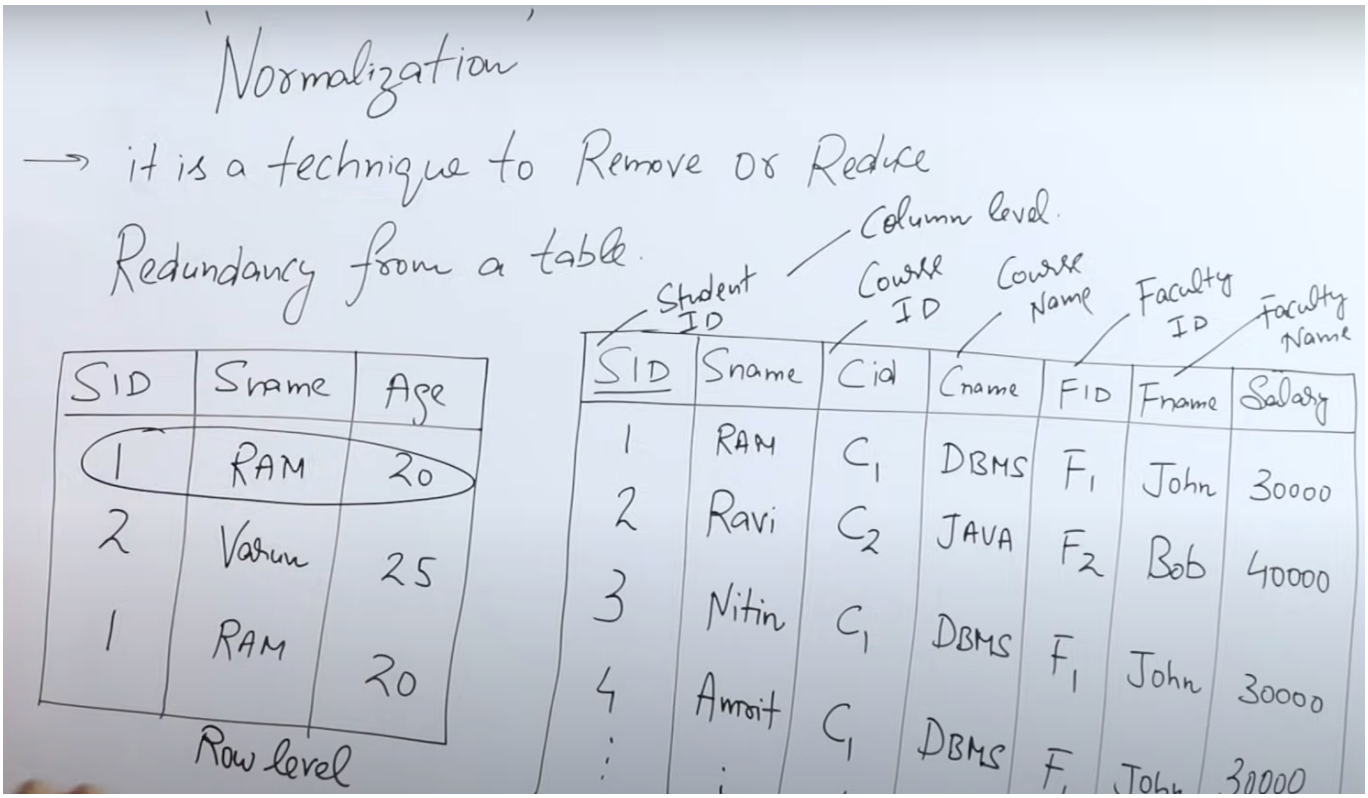
- One of the candidate keys is selected by database administrator as a Primary means to identify tuple is called primary Key. Primary Key attribute are not allowed to have Null values. Exactly one Primary Key per table in RDMS.
- Candidate key which are not chosen as primary key is alternate key.

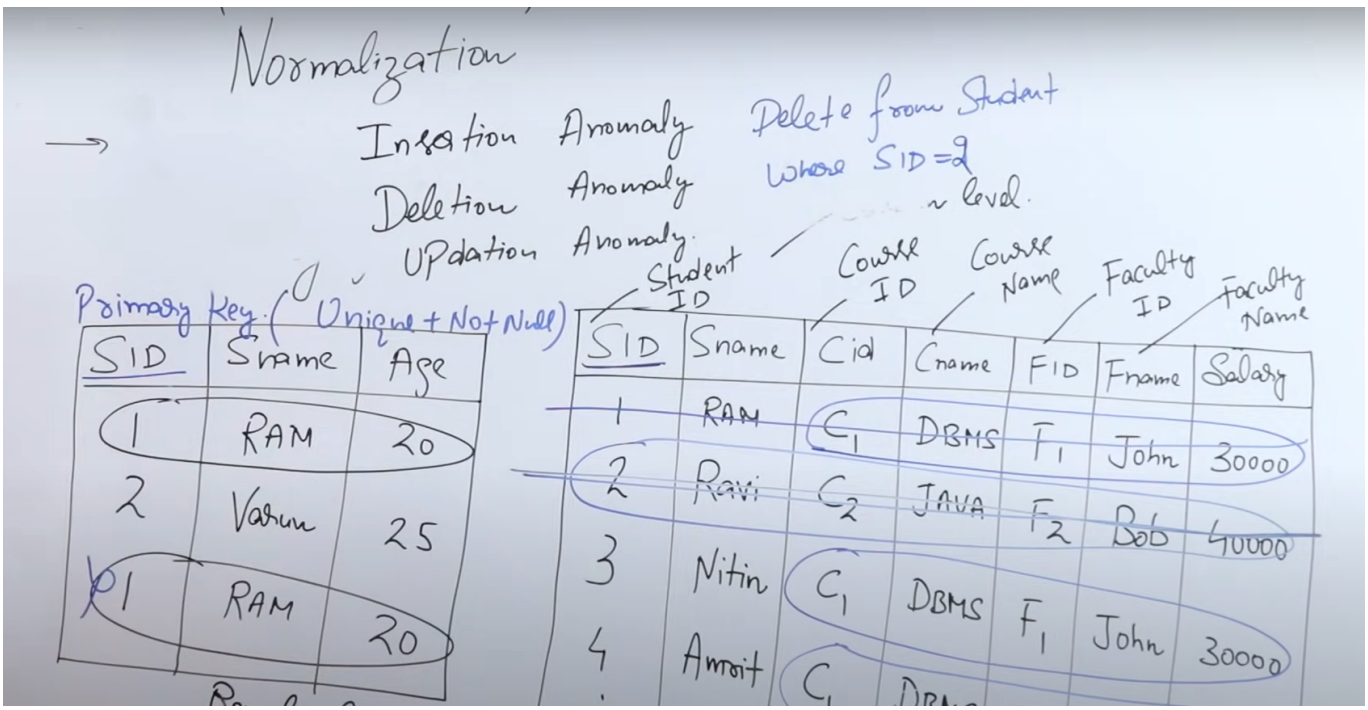
Foreign Keys

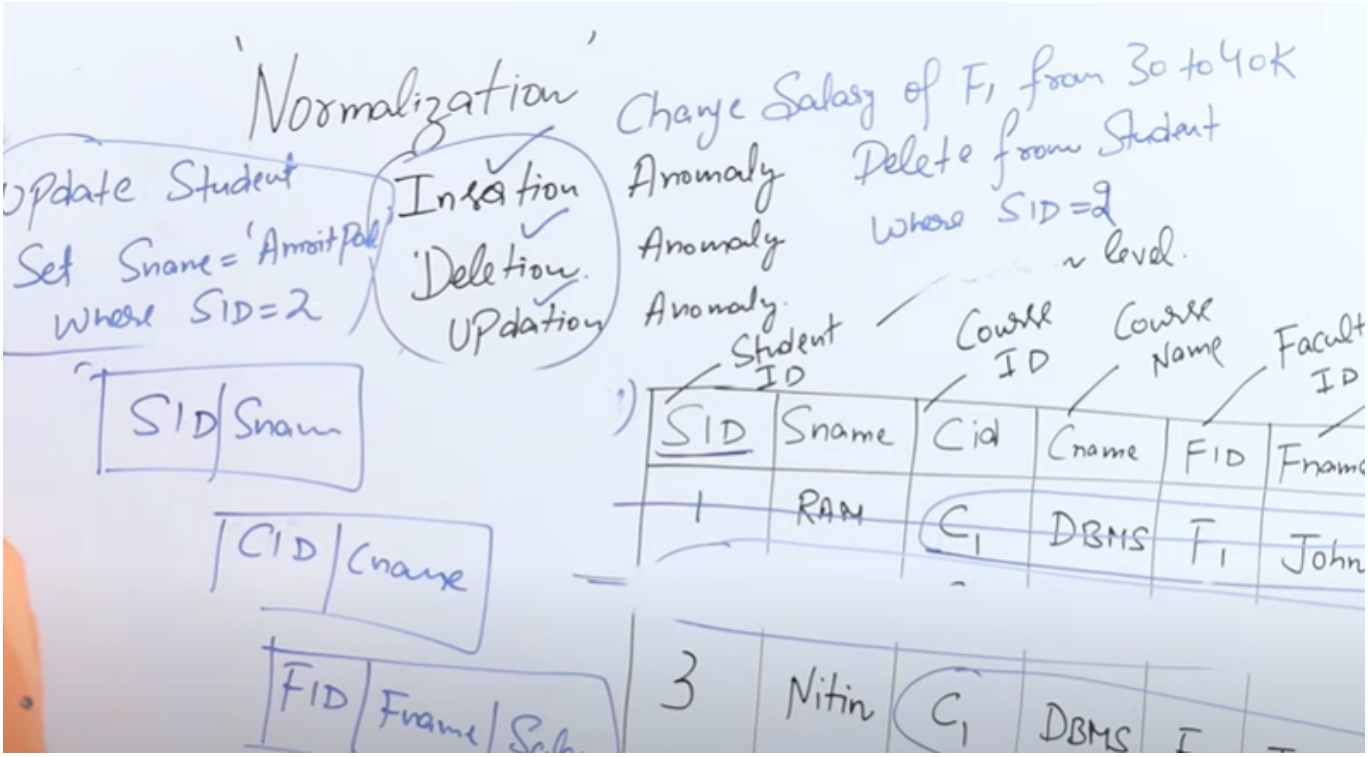
- A foreign key is a column or group of columns in a relational database table that refers the primary key of the same table or some other table to represent relationship.
- The ***concept of referential integrity*** is derived from foreign key theory.

- **Composite key** – Composite key is a key composed of more than one column sometimes it is also known as concatenated key.
- **Secondary key** – Secondary key is a key used to speed up the search and retrieval contrary to primary key, a secondary key does not necessary contain unique values.

Normalization


Normalization
→ it is a technique to Remove or Reduce Redundancy from a table.
Column level.
Student ID
Course ID
Course Name
Faculty ID
Faculty Name
SID Sname Age
1 RAM 20
2 Varun 25
1 RAM 20
Row level
SID Sname Cid Cname FID Fname Salary
1 RAM C1 DBMS F1 John 30000
2 Ravi C2 JAVA F2 Bob 40000
3 Nitin C1 DBMS F1 John 30000
4 Amrit C1 DBMS F1 John 30000

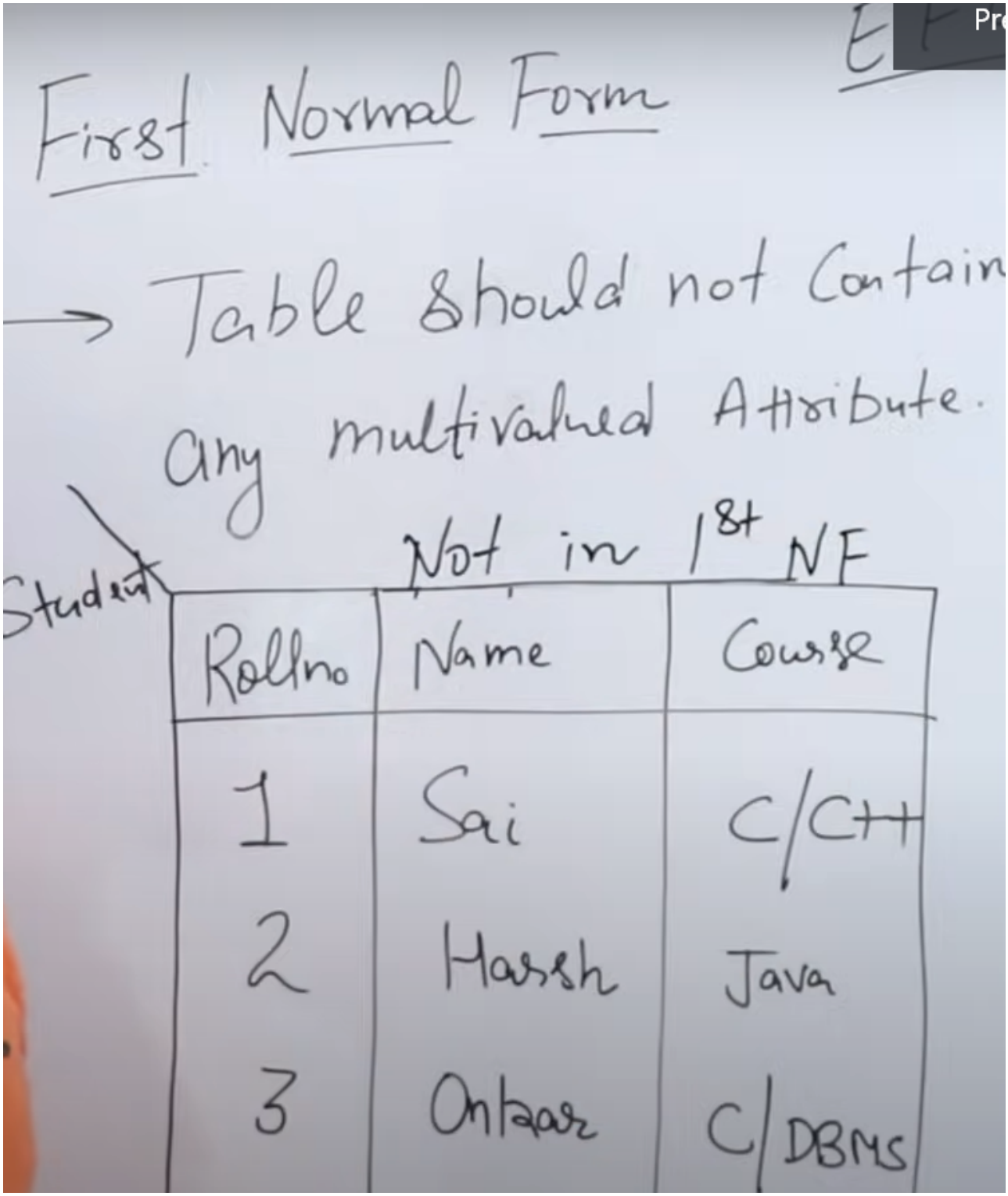

Normalization
→
Insertion Anomaly
Deletion Anomaly
Updation Anomaly
Delete from Student where SID=2
Primary Key (Unique + Not Null)
Student ID
Course ID
Course Name
Faculty ID
Faculty Name
SID Sname Age
1 RAM 20
2 Varun 25
1 RAM 20
SID Sname Cid Cname FID Fname Salary
1 RAM C1 DBMS F1 John 30000
2 Ravi C2 JAVA F2 Bob 40000
3 Nitin C1 DBMS F1 John 30000
4 Amrit C1

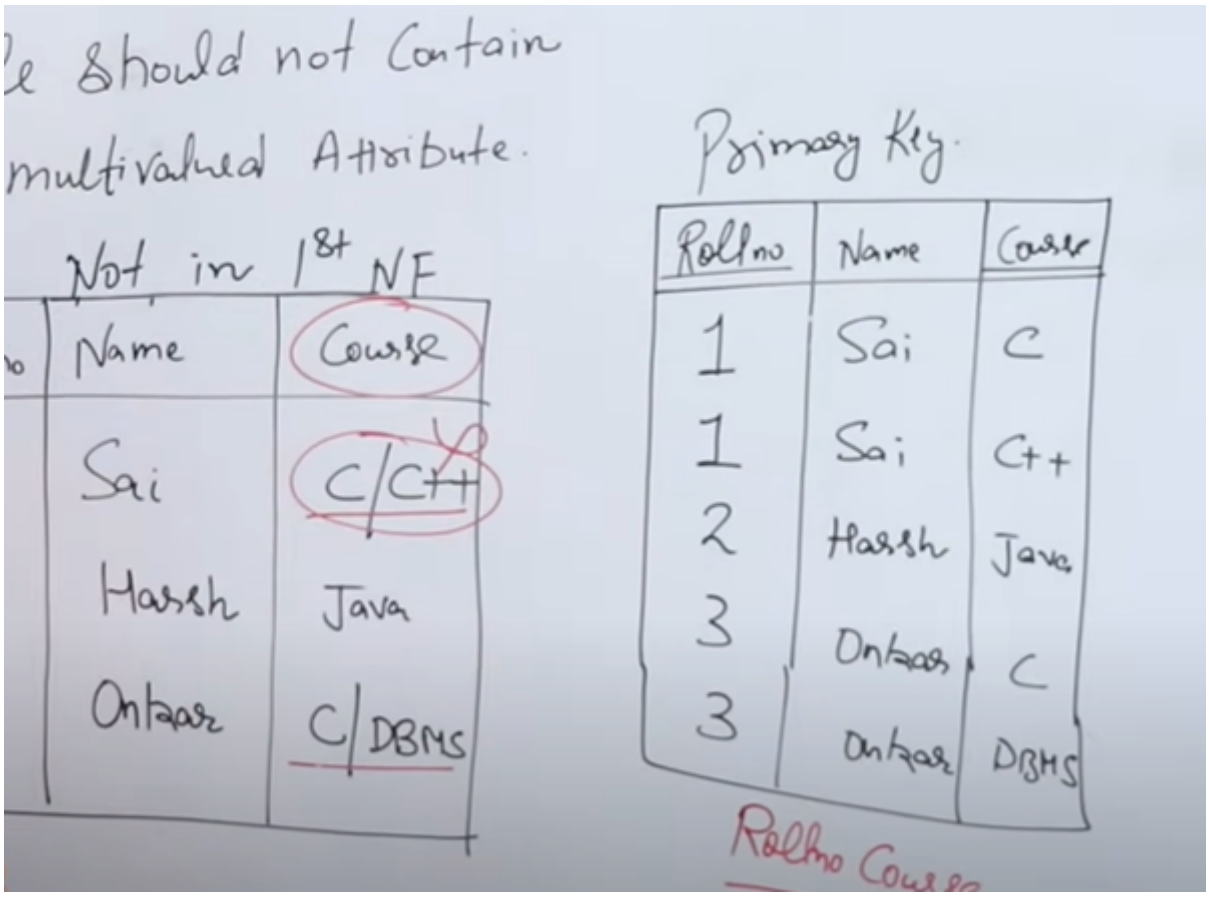


- `Atomic Values`: Each cell in a table contains indivisible, atomic values. Means a Relation should not contain any multivalued or composite attributes.
- `Unique Columns`: Each column must have a distinct name to identify the data it contains.
- `Primary Key`: A table in 1NF should have a primary key that uniquely identifies each record.
- `Eliminating Duplicates`: Duplicate rows are removed to prevent data redundancy.
- `Prime attribute`: A attribute is said to be prime if it is part of any of the candidate key
- `Non-Prime attribute`: A attribute is said to be non-prime if it is not part of any of the candidate key
- `Partial Dependency`: When a non – prime attribute is dependent only on a part (Proper subset) of candidate key then it is called partial dependency. (PRIME > NON-PRIME)
- `Full Dependency`: When a non – prime attribute is dependent on the entire candidate key then it is called Full dependency
- `Transitive Dependency`: A functional dependency from non-Prime attribute to non-Prime attribute is called transitive E.g.- R(A, B, C, D) with A as a candidate key A->B B->C [transitive dependency] C->D [transitive dependency
- `Multivalued Dependency`
- Denoted by, A ->-> B, Means, for every value of A, there may exist more than one value of B.
- E.g. S_name -> -> Club_name S_Name Club_name Kamesh Dance Kamesh Guitar

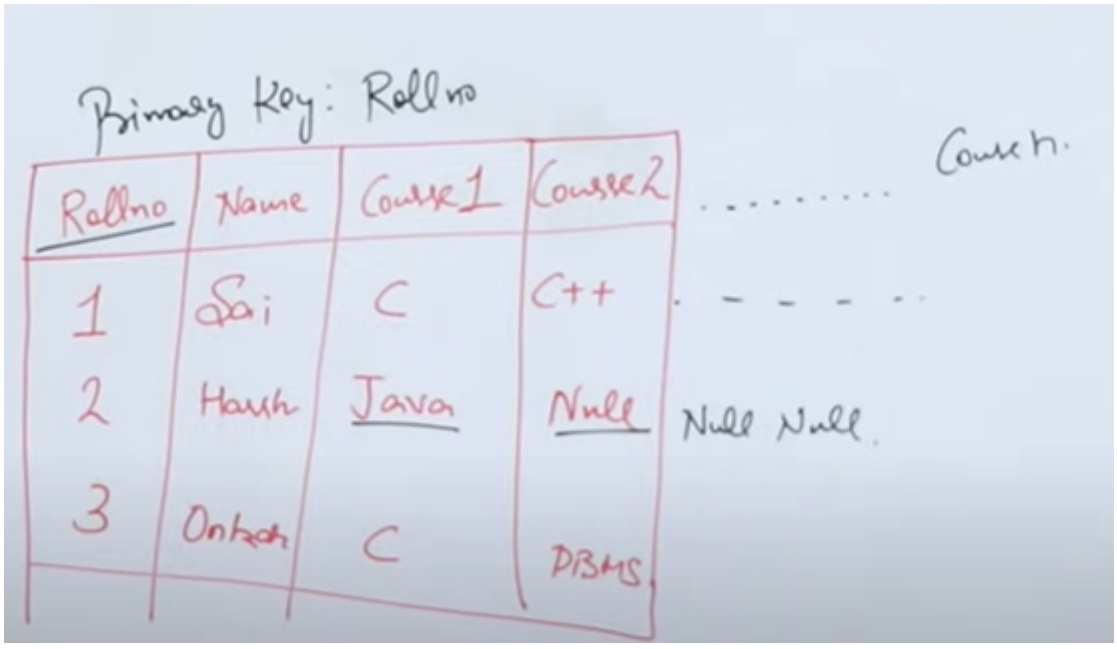
1NF

- Should not have multivalued attributes

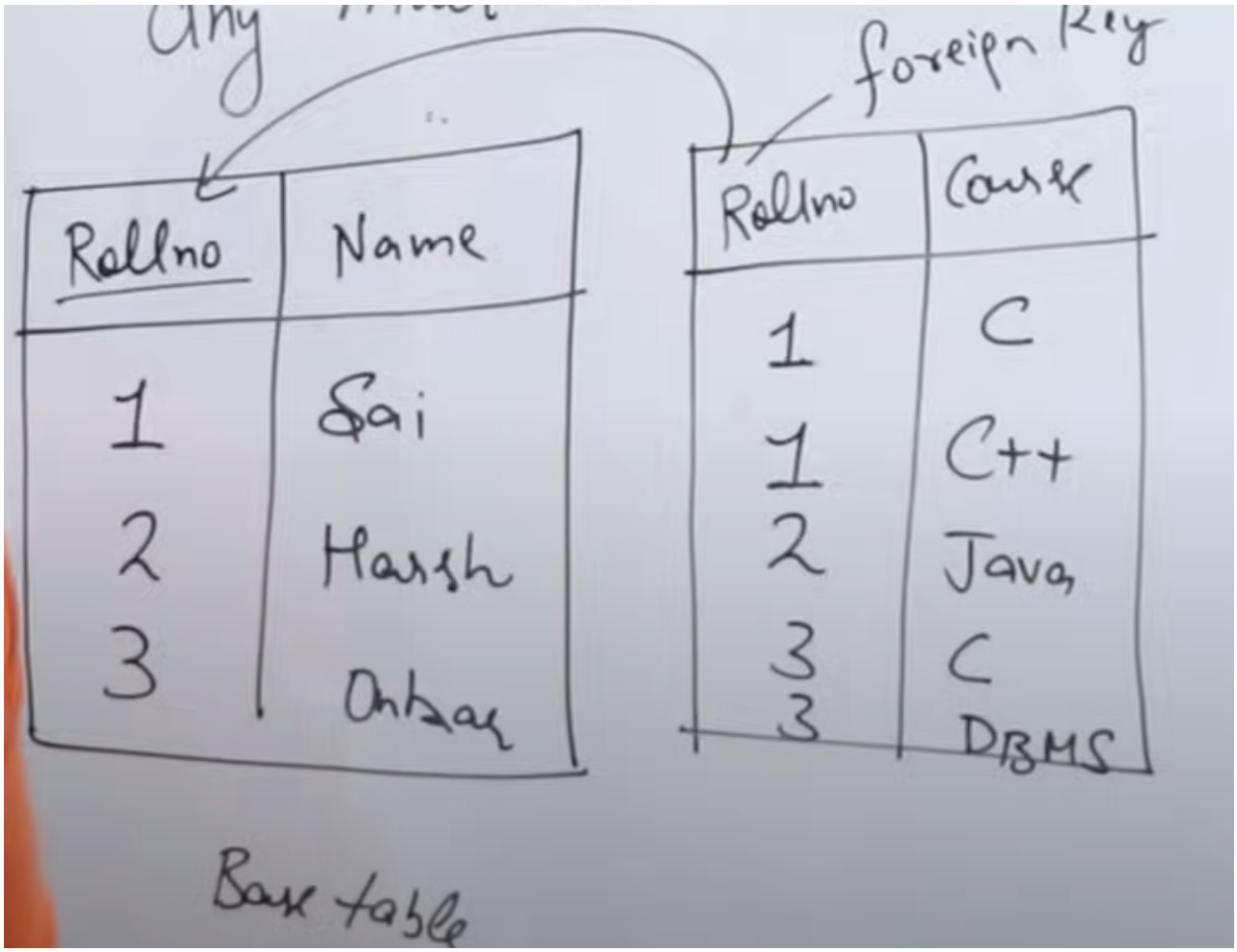


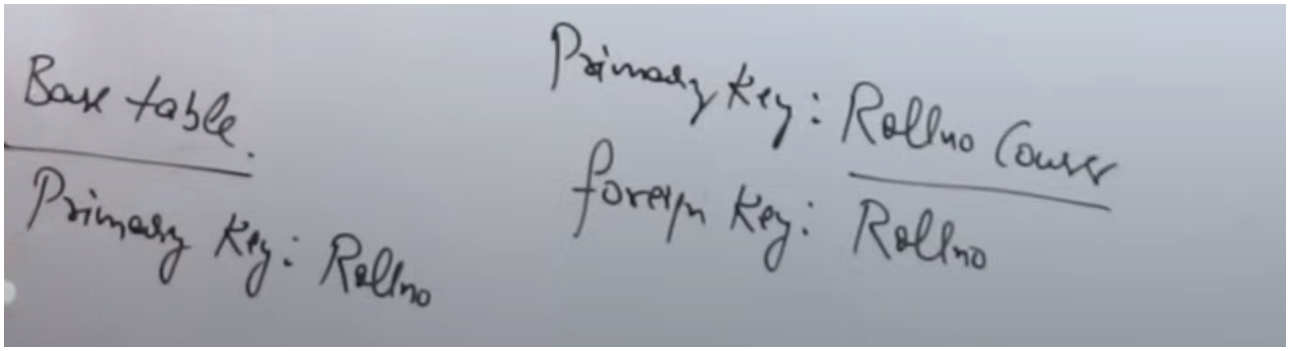

e should not Contain
multivalued Attribute.
Not in 1st NF
Name
Course
Sai
C/C++
Harsh
Java
Onkar
C/DBMS
Primary Key.
Rollno
Name
Course
1
Sai
C
1
Sai
C++
2
Harsh
Java
3
Onkar
C
3
Onkar
DBMS

1 2


Primary Key: Rollno
Rollno
Name
Course1
Course2
1
Sai
C
C++
2
Harsh
Java
Null
3
Onkar
C
DBMS
Null Null

3





2NF

- Relation R is in 2NF if :-
- R should be in 1 NF.
- R should not contain any Partial dependency. (that is every non-prime attribute should be fully dependent upon candidate key)

Q R(A, B, C) B→C

A	B	C
a	1	X
b	2	Y
a	3	Z
C	3	Z
D	3	Z
E	3	Z

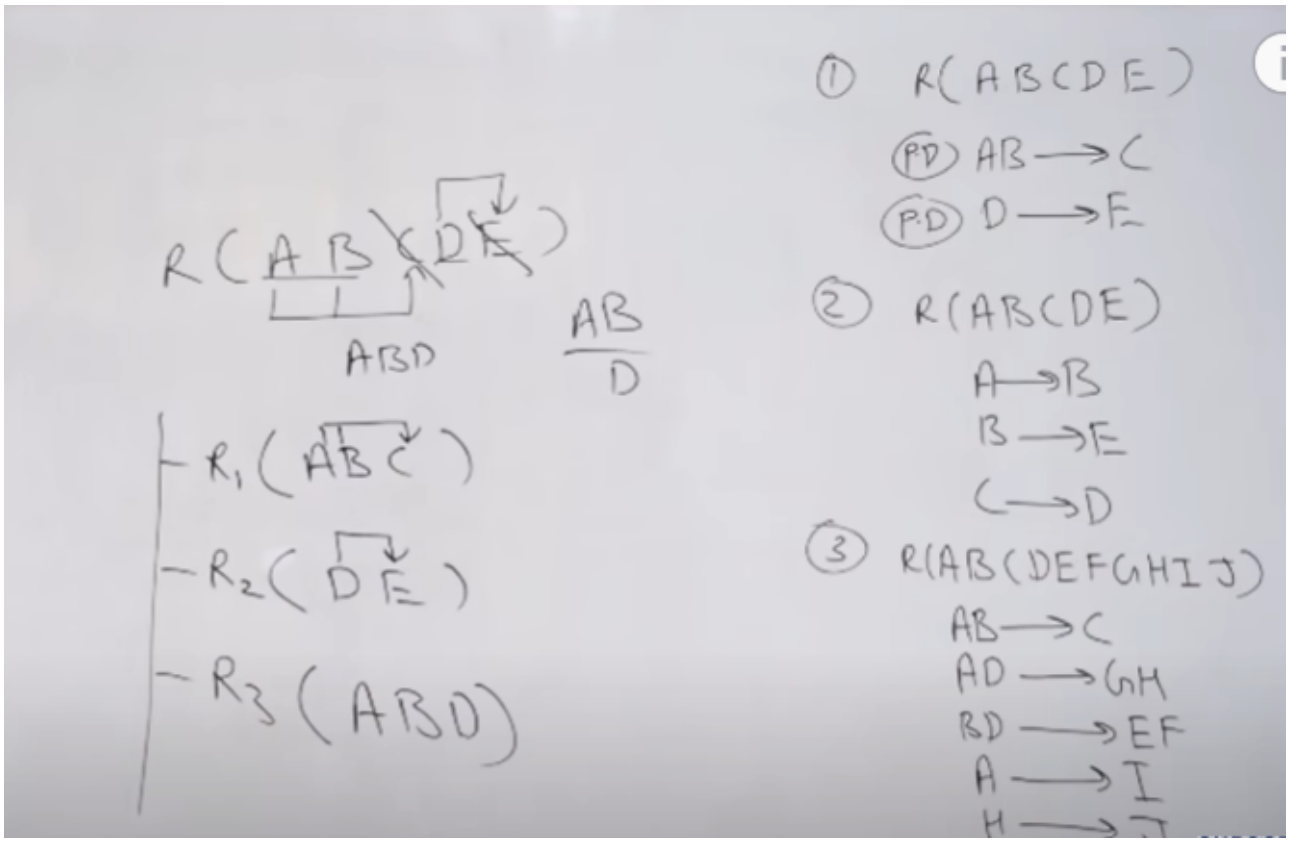
A	B
A	1
B	2
A	3
C	3
D	3
E	3

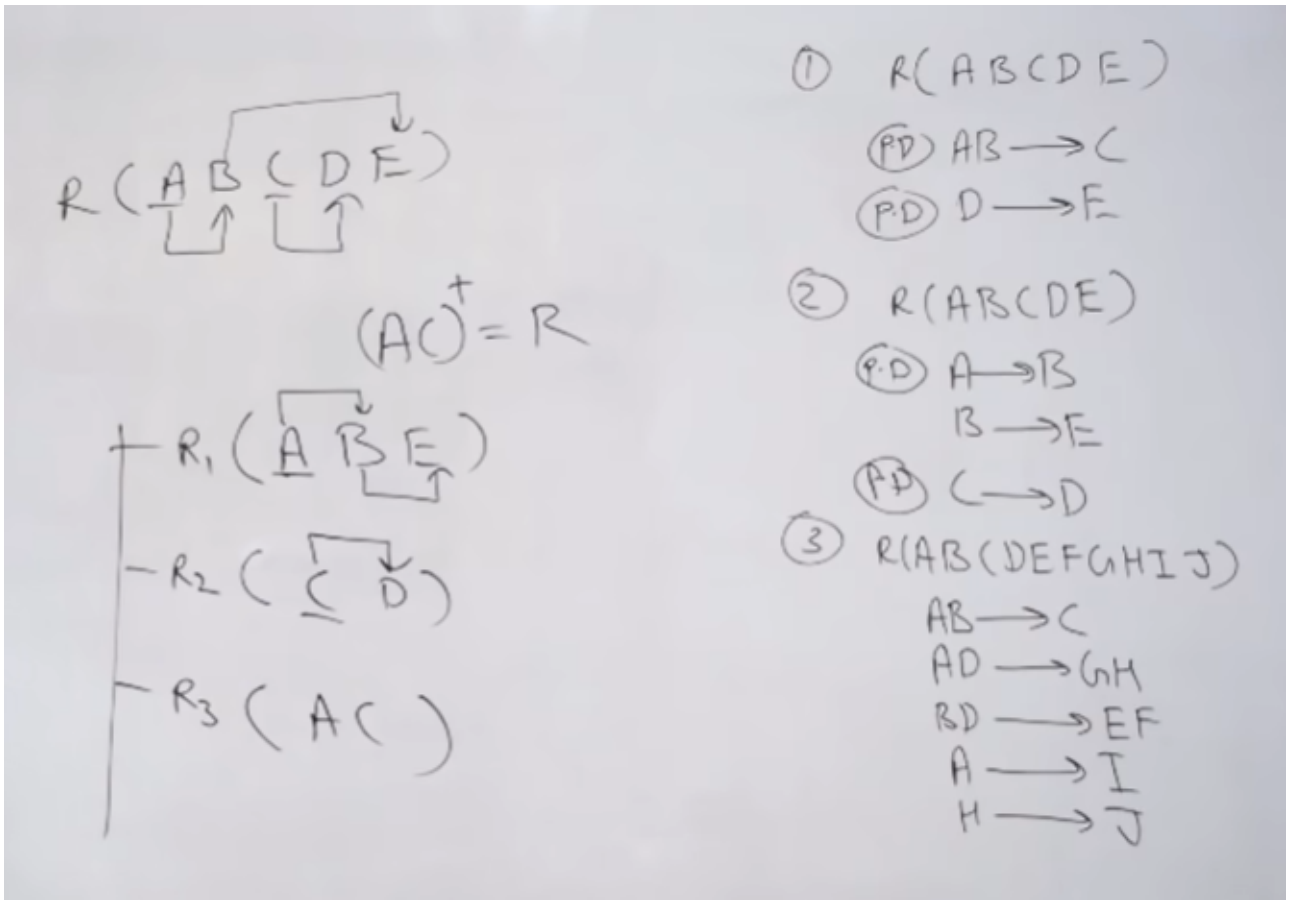
B	C
1	X
2	Y
3	z

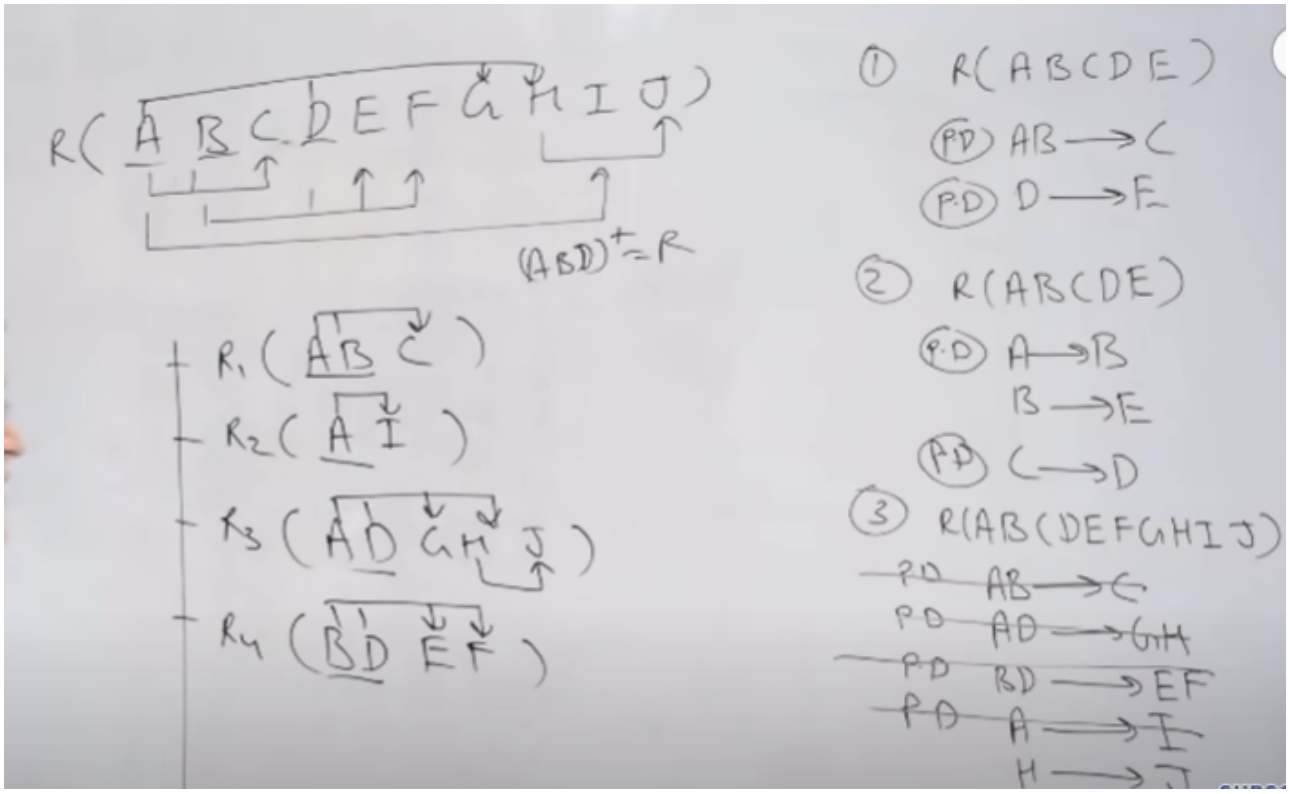
Decompose to 2NF

1. Consider the universal relational schema R (A, B, C, D, E, F, G, H, I, J) and a set of following functional dependencies. F = {AB -> C, A -> DE, B -> F, F -> GH, D -> IJ} determine the keys for R ? Decompose R into 2nd normal form.

- Answer : Its in 1NF
- To decompose in 2NF : Form 3 tables : AB, ADEIJ, BFGH



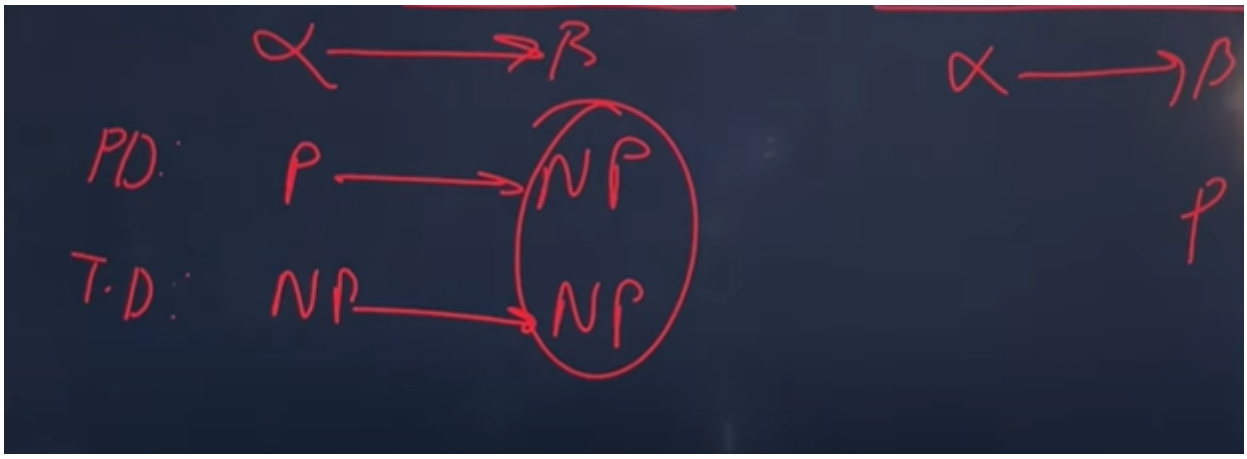




3NF

- Let R be the relational schema, it is said to be in 3 NF
- R should be in 2NF
- It must not contain any transitive dependency

- A relational schema R is said to be 3 NF if every functional dependency in R from α-->β, either α is super key or β is the prime attribute



- NPA->NPA NOT ALLOWED

R(A, B, C)

A → B

B → C

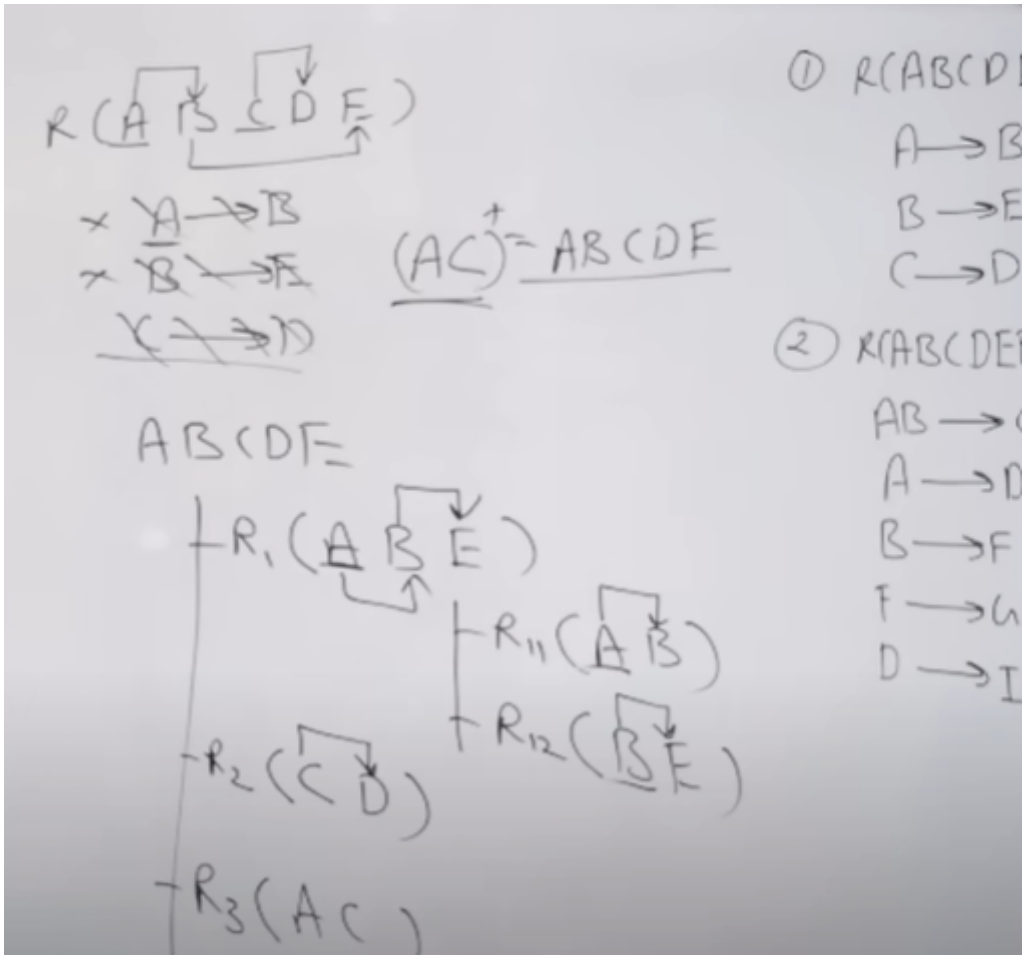
A	B	C
A	1	P
B	2	Q
C	2	Q
D	2	Q
E	3	R
F	3	R
G	4	S

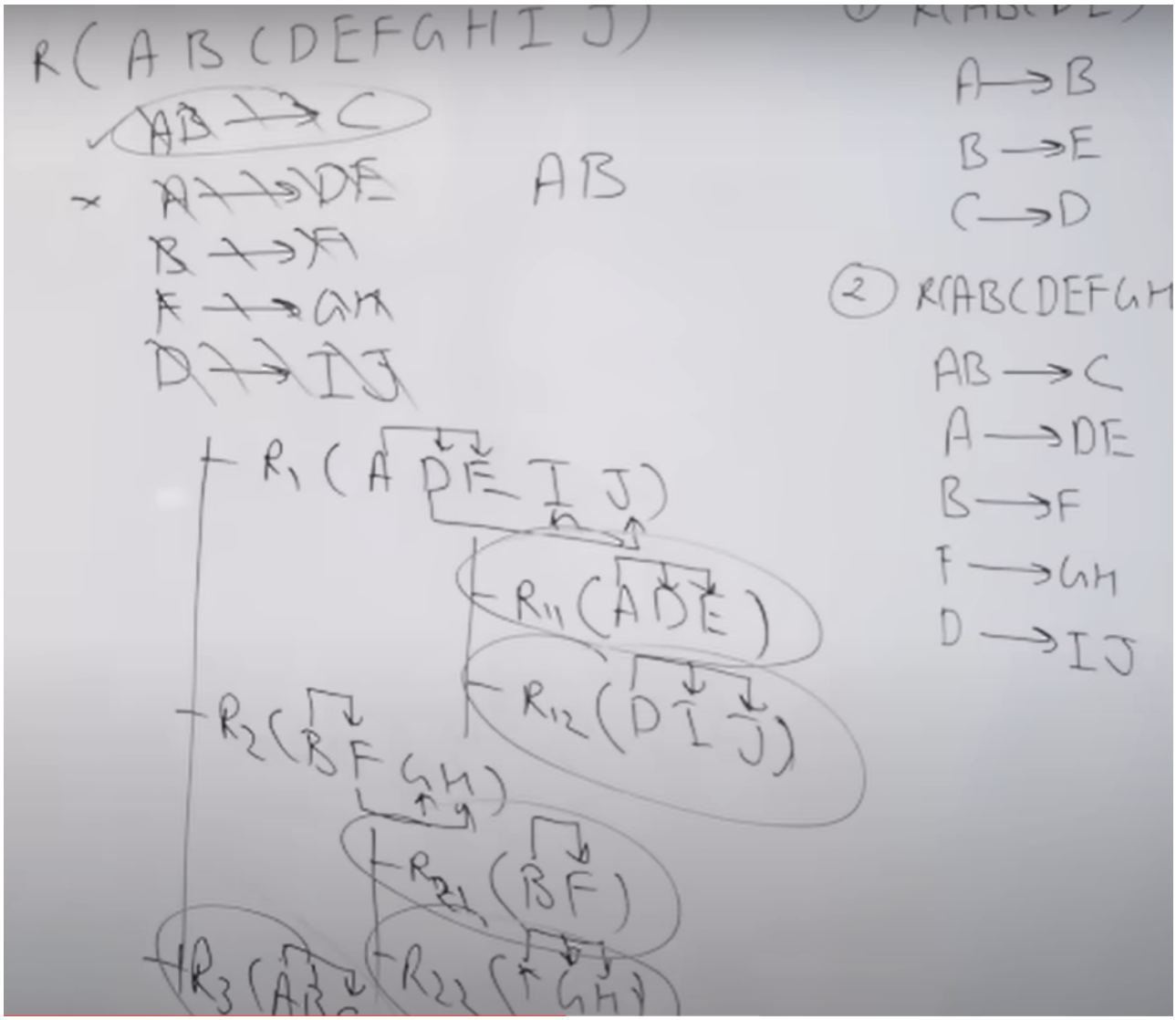
A	B
A	1
B	2
C	2
D	2
E	3
F	3
G	4

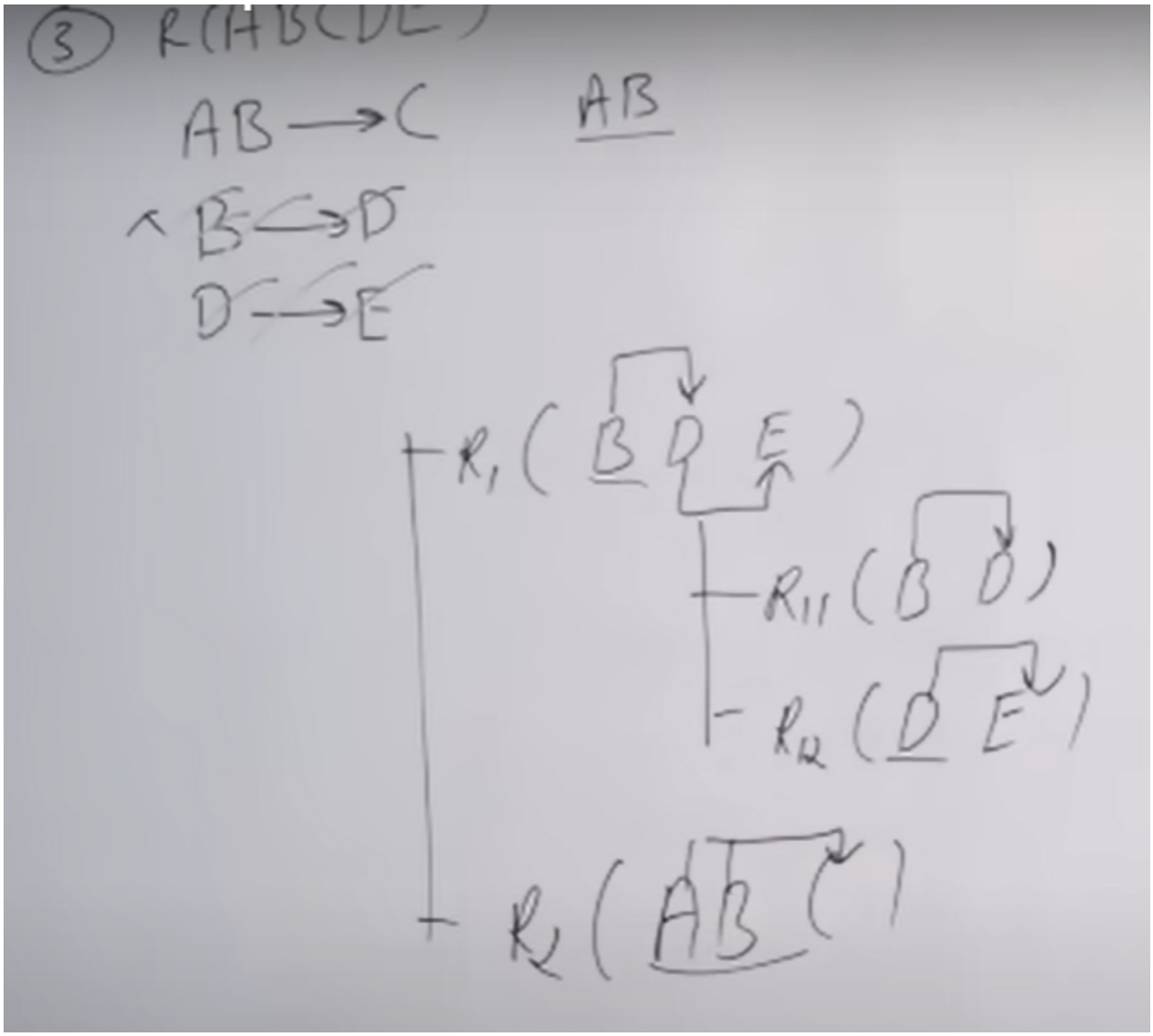
B	C
1	P
2	Q
3	R
4	S

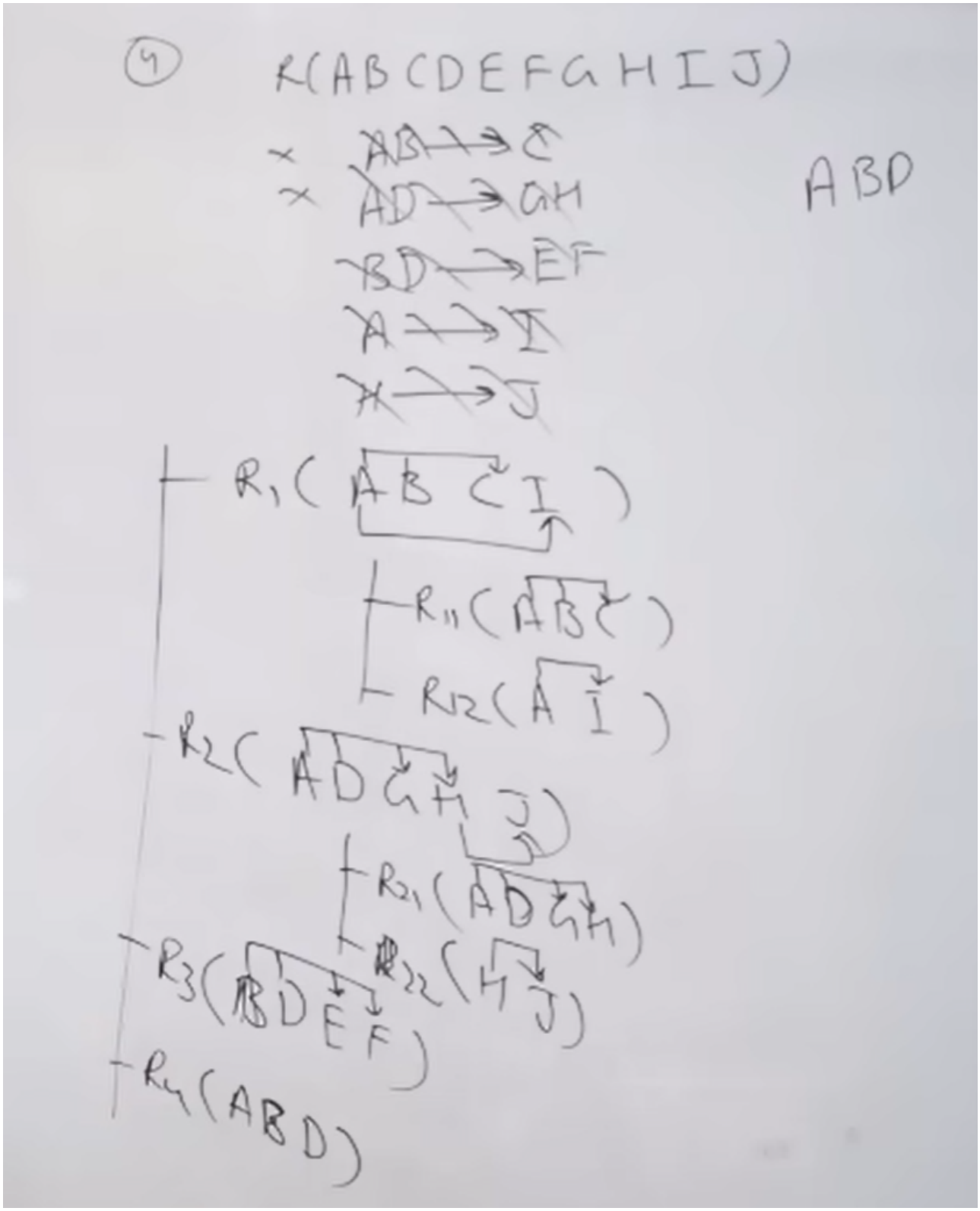
Decompose to 2NF

- a - super key or b - PA




R(A B C D E F G H I J)
AB → C
A → DE
B → F
F → GH
D → IJ
AB
R1(A D E I J)
R11(A D E)
R12(D I J)
R2(B F G H)
R21(B F)
R3(A B
R22(
A → B
B → E
C → D
(2) R(ABCDEFGH
AB → C
A → DE
B → F
F → GH
D → IJ


3
AB → C
AB
B → D
D → E
R1 (B D E)
R11 (B D)
R12 (D E)
R2 (AB C)



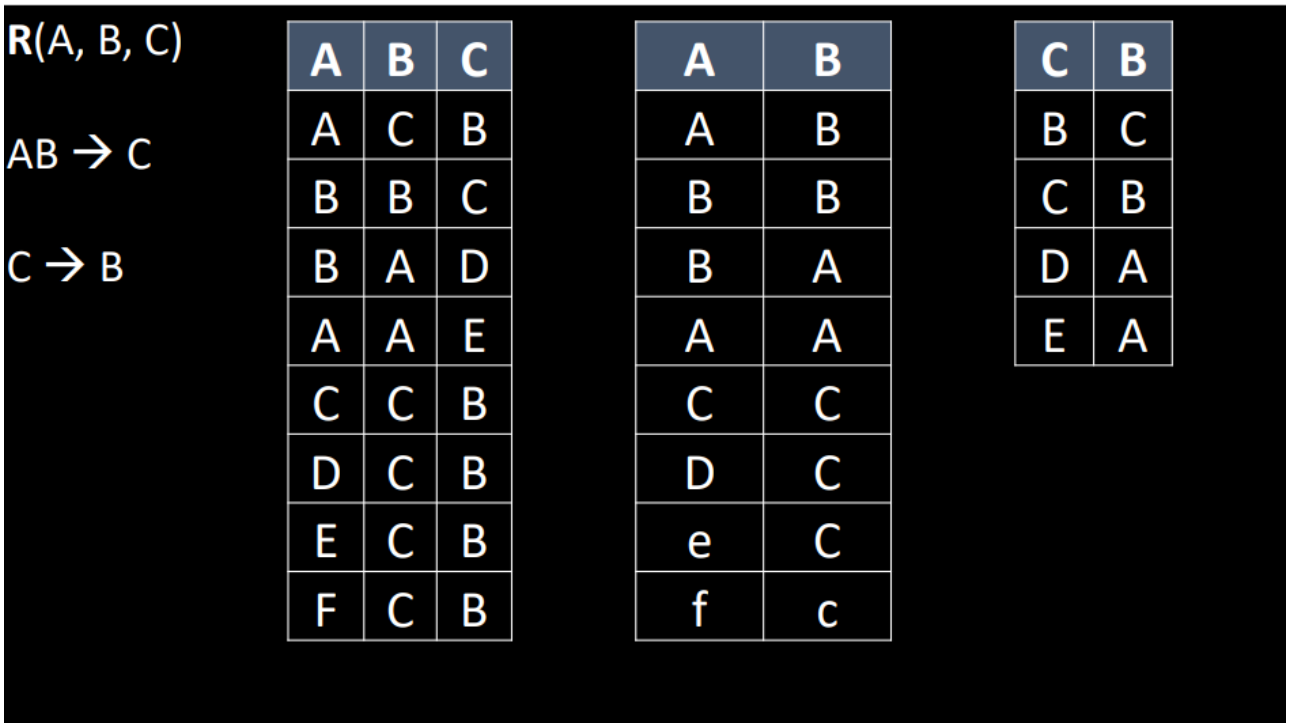
BCNF

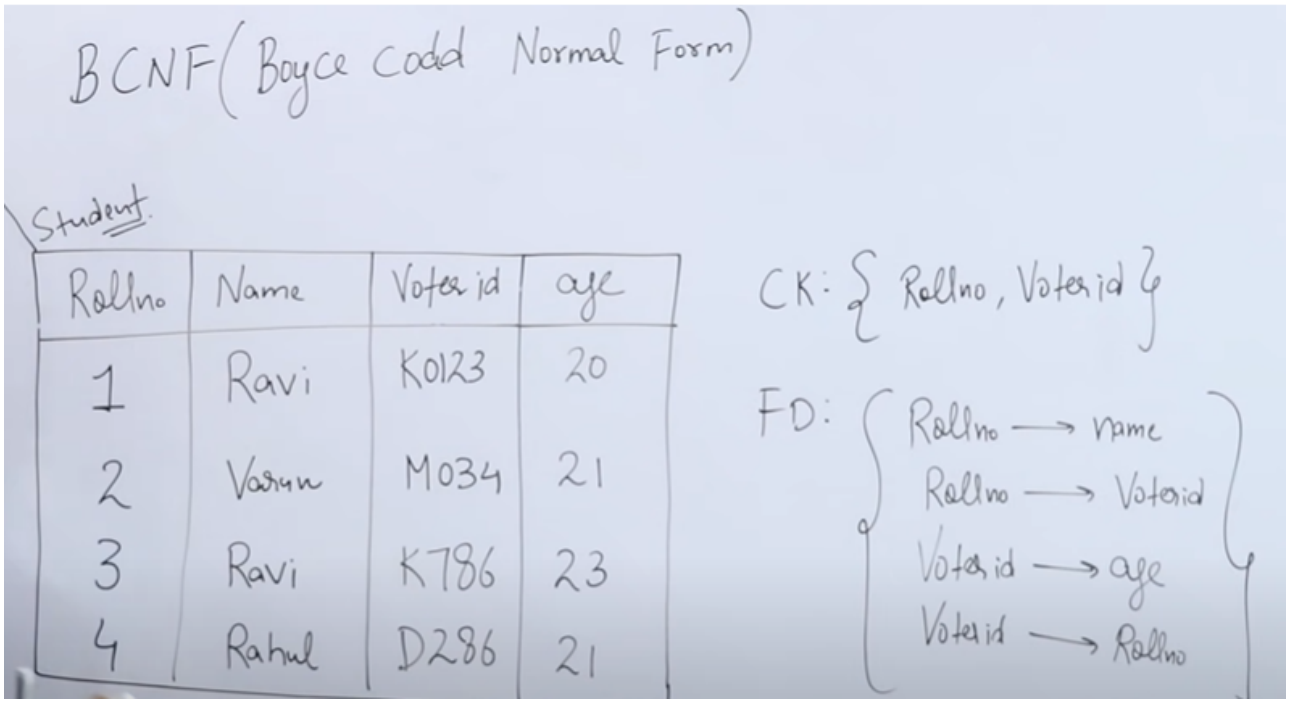
- A relational schema R is said to be BCNF if every functional dependency in R from
- α -> β
- α must be a super key or CK

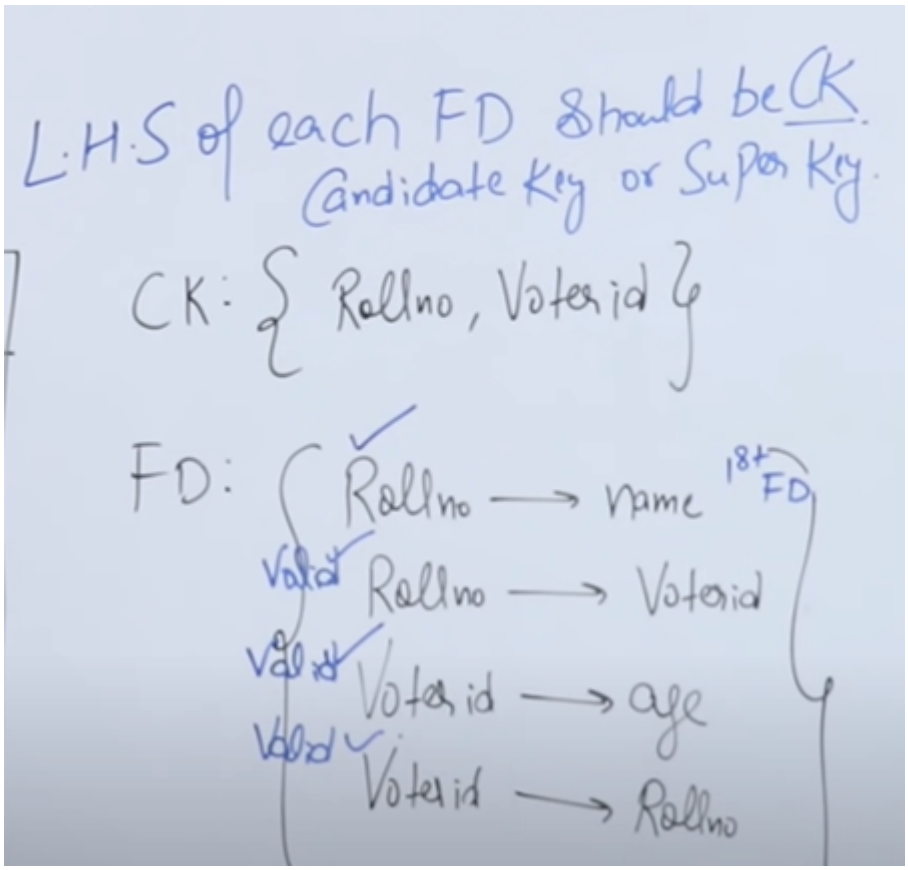
Important Points

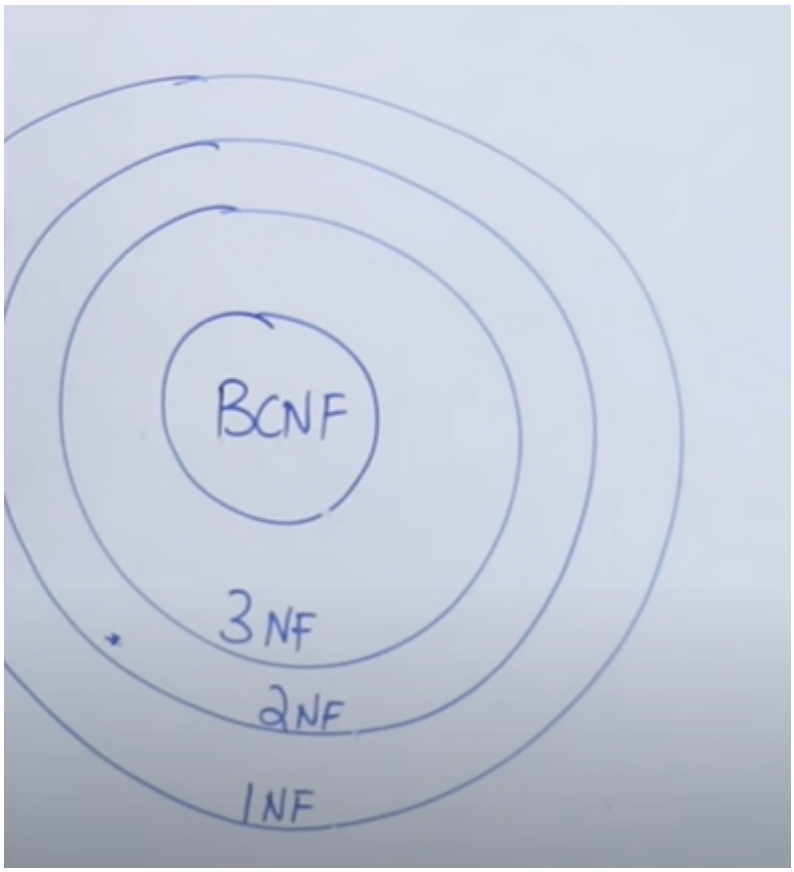
- A Relation with `two attributes is always in BCNF`.

- A Relation schema R consist of `only prime attributes then R is always in 3NF`, but `may or may not be in BCNF`.
- Boyce-Codd Normal Form guarantees a good decomposition, without anomalies



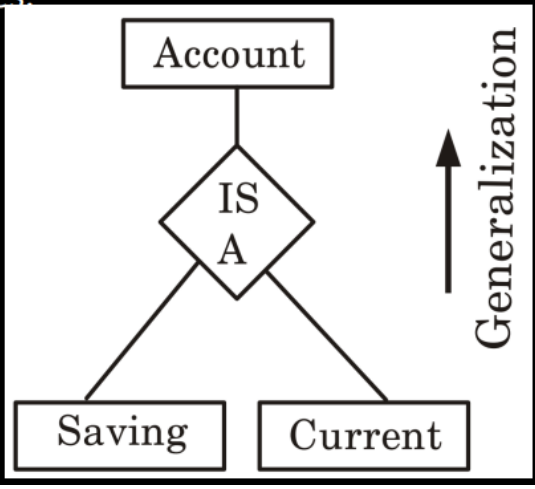



L.H.S of each FD should be CK.
Candidate Key or Super Key.
CK: { Rollno, Voterid }
FD:
Rollno → name
Rollno → Voterid
Voterid → age
Voterid → Rollno
18+ FD


BCNF
3NF
2NF
1NF

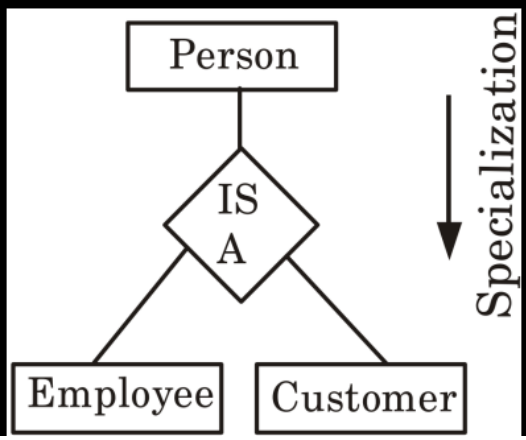
Generalization

- Involves merging two lower-level entities to create a higher-level entity.
- A bottom-up approach that builds complexity from simpler components.
- Highlights similarities among lower-level entity sets while hiding differences.
- Leads to a simplified, structured data representation, aiding in database design and querying processes.



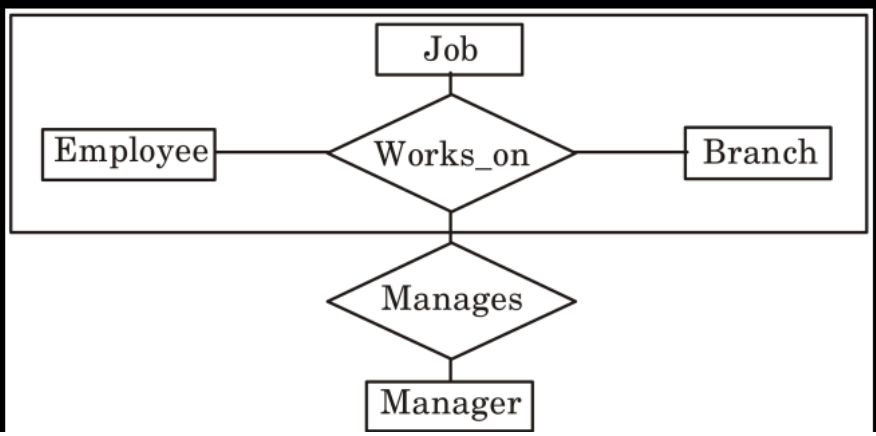
Specialization

- A process where a higher-level entity is broken down into more specific, lower-level entities.
- This top-down approach delineates complexity into simpler components.
- Acts as the converse of the generalization process, focusing on differentiating properties rather than similarities.



Aggregation

- A concept wherein relationships are abstracted to form higher-level entities, enabling a more organized representation of complex relationships.



Advantages of BCNF

1. No anomalies : ✓ (Due to no redundancy)
2. Lossless Join : ✓
3. Dependency Preservation : ✗

Summary

1. 1NF : DEFAULT(Unless told otherwise)
2. 2NF : Partial Dependency (0, 100)
3. 3NF : Transitive Dependency (NPA -> NPA not be there)
4. BCNF : left should be candidate/super key
