

UNIT-IV

RELATIONAL DATABASE DESIGN

Features of Good Database Design:

- A Good database contains consistent data and there is no chance for inconsistency.
- It also reduce the data redundancy and there is no chance for duplication of data.
- There is no wastage of space.
- Uniform of data.

NORMALIZATION:

- It is a process of organising the data in a database and also avoid the data redundancy,insertion,deletion,updatation anomaly.

ANOMOLIES IN DATABASE:

- ➔ Definition: It is a possibly of certain data getting inconsistent.
- ➔ There are 3 types of anomalies in database:
 - (1)Update Anomoly.
 - (2)Deletion Anomoly.
 - (3)Insert Anomoly.

Employee table

eid	name	address	department
1	abc	guntur	cse
1	abc	guntur	ece
2	def	vizag	mec
3	xyz	vijayawada	eee
3	xyz	vijayawada	ce

(1)Update Anomoly:

It is a data inconsistency and there is a partial update.

Example:

In above table,2 rows for employee “abc” and he belongs to 2 department.If we want to update the address of “abc”,data will become inconsistent and there is a partial update and correct address can’t be updated.

(2)Deletion Anomoly:

It means loss of data due to deletion of other data.

Example:

Suppose an institute close the department “mec”,the information of employee lost.

(3)Insert Anomoly:

- It means inability to add the database due to absence of other data.

Example:

Suppose new employee’s are joining in an institute,those who are under training and currently not assigning to any department then we couldn’t able to insert the data into table and it doesn’t allow null values.

- To overcome these anomolies we need to use “**Normalization**” forms.

Functional Dependency(FD):

- **Definition:** Association among attributes is known as “Functional Dependency” and it is represented as (\longrightarrow)
- Let us consider a relation ‘R’ and A,B are two non-empty attributes in a relation ‘R’ then FD is represented as $A \longrightarrow B$. Where ‘A’ functionally determines B. So, ‘A’ is called “**Determinant**”.
- The Lefthandside(LHS) attributes determines the values of attributes on the rightside.
- A primary key is a special case of FD.
- A Functional Dependency “ $A \longrightarrow B$ ” says that if two tuples agree on the values in the attribute ‘A’ they also agree on the values in attributes ‘B’.
 $t_1.A = t_2.A$ then
 $t_1.B = t_2.B$

here, A,B are two attributes and t_1, t_2 are tuples.

- A FD is single to single($A \longrightarrow B$),
single to many($A \longrightarrow BC$),
many to single($BC \longrightarrow A$),
many to many($AB \longrightarrow CD$).

Example:

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

Find the following FD's Satisfied are Not Satisfied ???

- (a) $A \rightarrow B$: Not satisfied..
- (b) $B \rightarrow C$: Not satisfied..
- (c) $C \rightarrow D$: Not satisfied..
- (d) $D \rightarrow A$: Not satisfied..
- (e) $A \rightarrow BC$: Not satisfied..
- (f) $AB \rightarrow C$: Satisfied..
- (g) $AC \rightarrow D$: Not satisfied..

Types Of Functional Dependencies:

There are 3 types:

(1) Trivial Functional Dependencies (FD's):

If the right hand side attributes are completely subset of left hand side attributes.

Symbolically:

if $A \rightarrow B$ then B is subset to A.

The following are also trivial FD's that are,

$A \rightarrow A$, $B \rightarrow B$.

Example: Let us consider a student relation it contains attributes student_id, student_name.

$\{student_id, student_name\} \rightarrow \{student_id\}$ is a FD's.

Also,

$s_id \rightarrow s_id$

$s_name \rightarrow s_name$

(a) Non-Trivial Functional Dependencies:

If one of the right hand side attribute is not subset of left hand side attributes.

Example:

A student relation with 3 attributes
student_id, student_name, student_address.
The following are non-trivial.

Sid \rightarrow name

sid \rightarrow address

(i) Completely Non-Trivial FD's:

If a Functional Dependency $A \rightarrow B$ holds true
where $A \cap B$ is NULL.

$A \cap B \neq \emptyset$

Here, RHS attributes are independent of LHS
attributes.

(2) TRANSITIVE FD's:

A FD is said to be transitive if $A \rightarrow B, B \rightarrow C$ then
 $A \rightarrow C$. Transitive Dependency is occurred with 3 or
more than 3 attributes and also form with 2 or more
FD's.

Example:

sid \rightarrow name,

name \rightarrow age,

sid \rightarrow age.

(3) MULTIVALUED FD's:

If more than one independent attributes depends
on particular attribute is known as "Multivalued
FD's".

for **example**, A, B, C are 3 attributes, the attributes B and C are independent to each other but depends on attribute A.

$$A \twoheadrightarrow B$$
$$A \twoheadrightarrow C$$

****Armstrong Axioms/Inference Rules/Properties of FD's: ****

- Armstrong axioms is a set of rules.
- It was developed by William W. Armstrong in 1974.
- These rules reduce the set of functional dependencies.

(1) Reflexivity Rule:

It is same as Trivial FD's. If A, B are two non-empty sets of attributes and B is a subset of A.

(2) Transitivity Rule:

$$A \rightarrow B$$
$$B \rightarrow C$$
$$A \rightarrow C$$

(3) Augmentation Rule:

If attribute A holds B and C is a particular attribute then AC holds BC i.e., $AC \rightarrow BC$. It means that attribute C doesn't change the basic dependencies ($A \rightarrow B$).

(4) Union Rule:

If attribute A holds B and attribute A holds C then A holds BC.

$A \rightarrow B, A \rightarrow C$
therefore, $A \rightarrow BC$.

(5) Decomposition Rule:

If attribute A holds BC then A holds C.

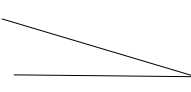
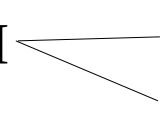
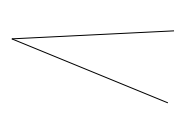
$A \rightarrow BC$
 $A \rightarrow B, A \rightarrow C$

(6) Pseudo Transitivity Rule:

If attribute A holds B and BC holds D then AC holds D.

$A \rightarrow B$
 $BC \rightarrow D$ then
 $AC \rightarrow D$.

Example:

$A \rightarrow B$		
$B \rightarrow C$		$A \rightarrow C$
$D \rightarrow E$		
$EF \rightarrow GH$		$EF \rightarrow G$ $EF \rightarrow H$
$H \rightarrow KF$		$H \rightarrow K$ $H \rightarrow F$
$K \rightarrow A$		$EF \rightarrow A$ $H \rightarrow A$

Closure set:

→ A functional dependency is said to be closure if it cover all attributes or not based on given condition.

Algorithm or procedure:

step1: Equivate an attribute or attributes for which closure need to be identified($x^+=x$)

step2: Take FD one by one and verify whether LHS is available in X, if so add RHS attributes to x, if it is not available.

Step3: Repeat step2 as many times as possible to cover attributes.

Step4: Stop procedure after no more attribute added to X and declare X as closure set of attributes.

EX: suppose a relation R with schema R(ABCDEFGHK) with set of FD's

F:A→B

B→DE

E→GH

K→H

B→K

find

a) A^+ b) B^+ c) E^+

a) A^+

step1: $A^+ = A$

step2: $A^+ = A$

$= AB(A \rightarrow B)$

$= ABDE(B \rightarrow DE)$

$= ABDEGH(E \rightarrow GH)$

$= ABDEGHK(B \rightarrow K)$

$= ABDEGHK$

therefore, A^+ is closure set

it is super key because it covered all attributes

b) B^+

step1: $B^+ = B$

step2: $B^+ = B$

$= BDE(B \rightarrow DE)$

$= BDEGH(E \rightarrow GH)$

$= BDEGHK(B \rightarrow K)$

$= BDEGHK$

B^+ is not superkey

because it does not covered all attributes(i.e, A)

c) E^+

step1: $E^+ = E$

step2: $E^+ = E$

$= EGH(E \rightarrow GH)$

$= EGH$

E^+ is not superkey

because it does not covered all attributes(i.e, A,B,C,D,K)

Ex2: consider the following FD's

$AB \rightarrow CD$

$AF \rightarrow D$

$DE \rightarrow F$

$C \rightarrow G$

$F \rightarrow E$

$G \rightarrow A$

which following is incorrect?

[*C and D*]

a) $(CF)^+ = \{A, C, D, E, F, G\}$

b) $(BG)^+ = \{A, B, C, D, G\}$

c) $(AF)^+ = \{A, C, D, E, F, G\}$

d) $(AB)^+ = \{A, C, D, F, G\}$

a) $(CF)^+ = \{A, C, D, E, F, G\}$

$(CF)^+ = CFE(F \rightarrow E)$

$= CFEG(C \rightarrow G)$

$= CFEGA(G \rightarrow A)$

$= CFEGAD(AF \rightarrow D)$ (correct)

$= CFEGAD$

b) $(BG)^+ = \{A, B, C, D, G\}$

$(BG)^+ = BG$

$= BGA(G \rightarrow A)$

$= BGACD(AB \rightarrow CD)$ (correct)

c) $(AF)^+ = AF$

$= AFE(F \rightarrow E)$

=AFED(AF->D) (incorrect)

d) $(AB)^+ = \{A, C, D, F, G\}$

$AB^+ = AB$

=ABCD(AB->CD)

=ABCDG(C->G) (incorrect)

Ans: c and d

Ex:3 In a schema with attributes A,B,C,D,E and set of FD's are

A->B

A->C

CD->E

B->D

E->A

Which of the following FD's not implied by the above set

a)CD->AC b)BD->CD c)BC->CD d)AC->BC

[**B**]

a)CD->AC

$(CD)^+ = CD$

=CDE (CD->E)

=CDEA (E->A)

=CDEAB (A->B)

b)BD->CD

$(BD)^+ = BD$ (it not contain CD)

c)BC->CD

$(BC)^+ = BC$

=BCD (B->D)

d)AC->BC

$$\begin{aligned}(AC)^+ &= AC \\ &= ACB \quad (A \rightarrow B)\end{aligned}$$

EX-4: Consider the relational schema R(EFGHIJKLMN) and set of FD's are

$$\begin{aligned}EF &\rightarrow G \\ F &\rightarrow IJ \\ EH &\rightarrow KL \\ K &\rightarrow M \\ L &\rightarrow N\end{aligned}$$

what is key for relation R

[**B**]

- a){E,F}
- b){E,F,H}
- c){E,F,H,K,L}
- d)E

$$\begin{aligned}\text{a)}(EF)^+ &= EF \\ &= EFG(EF \rightarrow G) \\ &= EFGIJ(F \rightarrow IJ)\end{aligned}$$

$$\begin{aligned}\text{b)}(EFH)^+ &= EFH \\ &= EFHG \\ &= EFHGIJ \\ &= EFHGIJKL \\ &= EFHGIJKLMN\end{aligned}$$

$$\begin{aligned}\text{c)}(EFHKL)^+ &= EFHKL \\ &= EFHKL MN \\ &= EFHKL MNG\end{aligned}$$

d) $E^+ = E$

ans: **b**

EX:5 R(ABCDEH) what are the candidate keys for [**d**]

A \rightarrow B

BC \rightarrow D

E \rightarrow C

D \rightarrow A

a) AE, BE

$(AE)^+ = AE$

= AEC

= AECB

= AECBD

$(BE)^+ = BE$

= BEC

b) AE, BE, DE

$(AE)^+ = AE$

= AEC

= DEACB

= AECB

= AECBD

$(BE)^+ = BE$

= BEC

$(DE)^+ = DE$

c) AEH, BEH, BCH

$(AEH)^+ = AEH$

= AEHB

= AEHBCD

$(BEH)^+ = BE$

= BEHC

= BEHCDA

$(BCH)^+ = BCH$

= BCHD

= BCHDA

d) AEH, BEH, DEH

$$\begin{aligned}
 (AEH)^+ &= AEH \\
 &= AEHB \\
 &= AEHBCD
 \end{aligned}$$

$$\begin{aligned}
 (BEH)^+ &= BEH \\
 &= BEHC \\
 &= BEHCDA
 \end{aligned}$$

$$\begin{aligned}
 (DEH)^+ &= DEH \\
 &= DEHC \\
 &= DEHCAB
 \end{aligned}$$

Equivalence of FD's:

A two sets of functional dependencies F1 and F2 are equivalent if $(F1^+ = F2^+)$

EX: Find equivalence of FD's

$$\begin{aligned}
 \mathbf{F1:} & A \rightarrow C \\
 & AC \rightarrow D \\
 & E \rightarrow AD \\
 & E \rightarrow H
 \end{aligned}$$

$$\begin{aligned}
 A^+ &= A \\
 &= AC \\
 &= ACD \\
 (AC)^+ &= ACD \\
 E^+ &= EADCH
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{F2:} & \\
 & A \rightarrow CD \\
 & E \rightarrow AH
 \end{aligned}$$

$$\begin{aligned}
 A^+ &= ACD \\
 E^+ &= EAH \\
 &= EAHCD
 \end{aligned}$$

∴ F1=F2

EX-2: R(ABCDEH)

$$\begin{aligned}
 \mathbf{F1:} & \\
 & A \rightarrow C \\
 & AC \rightarrow D \\
 & B \rightarrow D \\
 & E \rightarrow AD
 \end{aligned}$$

$$\begin{aligned}
 \mathbf{F2:} & \\
 & A \rightarrow CD \\
 & E \rightarrow AH \\
 & A^+ = A \\
 & \quad = ACD
 \end{aligned}$$

$$\begin{aligned}
A^+ &= A \\
&= AC \\
&= ACD \\
(AC)^+ &= ACD \\
B^+ &= BD \\
E^+ &= E \\
&= EAD \\
&= EADC
\end{aligned}$$

$$\begin{aligned}
E^+ &= EAH \\
&= EAHCD
\end{aligned}$$

∴ **F1! = F2**

Irreducible set/canonical cover/Minimal cover of FD's:

It is set of FD's with a simplified version of FD's with a simplified version of FD's that is closure of particular attributes, if it cover all attributes that can be redundant otherwise it is irreducible or canonical.

Procedure:

Step1: If we have multiple attributes on RHS, if necessary FD use decomposition rule.

Step2: Removal of redundant FD that means to check whether Fd is redundant or not, first hode that Fd and find closure of left in that Fd. If closure contains all attributes then Fd is redundant and remove that Fd from the set.

Step3: Look for Fd having more than one attribute on the LHS remove an attribute from the FD and take the closure of remaining attribute other FD. If removed attribute exists

in closure then remove it from FD. Repeat this process for remaining attributes.

Step4: apply union rule

Ex: R(ABCD)

A \rightarrow B

C \rightarrow B

D \rightarrow ABC

AC \rightarrow D

Ans:

Step1: using decomposition rule

A \rightarrow B

C \rightarrow B

D \rightarrow A

D \rightarrow B

D \rightarrow C

AC \rightarrow D

step2: Finding redundant FD and remove it

1. A \rightarrow B hide 1, closure of A $\Rightarrow A^+ = A$

2. C \rightarrow B hide 2, closure of C $\Rightarrow C^+ = C$

3. D \rightarrow A hide 3, closure of D $\Rightarrow D^+ = DBC$

4. D \rightarrow B hide 4, closure of D $\Rightarrow D^+ = DACB$

5. D \rightarrow C hide 5, closure of D $\Rightarrow D^+ = DAB$

6. AC \rightarrow D hide 6, closure of AC $\Rightarrow AC^+ = ACB$

remove 4 as it covered all attributes

step3:

AC \rightarrow D

Remove A, closure C = CB

remove C, closure $A=AB$

as A and C doesnot exist we can't remove that FD

step4: By applying union, the resultant minimal set is

$A \rightarrow B$

$C \rightarrow B$

$D \rightarrow AC$

$AC \rightarrow D$

ASSIGNMENT:

1) R(WXYZ)

$X \rightarrow W$

$WZ \rightarrow XY$

$Y \rightarrow WXZ$

2) R(ABCDE)

$A \rightarrow B$

$AB \rightarrow C$

$D \rightarrow AC$

$D \rightarrow E$

3) R(ABCDE)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

4) R(ABC)

$A \rightarrow BC$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C$

Finding Candidate Keys:

If any attribute that cannot appear on the righthand side (RHS) from the given schema that is called “candidate key”.

Ex-1: R(ABCDE)

A \rightarrow C

E \rightarrow D

B \rightarrow C

$(ABE)^+ = ABCDE = R$,

\therefore ABE is a candidate key because it covered all attributes.

Ex-2: R(ABCDE)

A \rightarrow BE

C \rightarrow BE

B \rightarrow D

$(AC)^+ = ACBED = R$,

\therefore AC is a candidate key because it covered all attributes.

Ex-3: R(ABCD)

AB \rightarrow C

BC \rightarrow D

CD \rightarrow A

Ans: $(AB)^+ = ABCD$

$$(BC)^+ = BCDA$$

$$(CD)^+ = CDA, \text{ CD not covered all attributes.}$$

∴ AB and BC are candidate keys..

Types Of Decomposition:

- **Decomposition:** The process of divide a single relation into two or more subrelations is called “Decomposition”.
- No information is lost from original relation during decomposition.
- When a subrelation are joined back the same relation is obtained i.e., Decomposition
- There are two types:
 - (1) Loss less join decomposition
 - (2) Dependency preserving decomposition.

(1) Loss Less Join Decomposition:

Step-1: Union of R_1, R_2, \dots, R_n must be equal to R . Each attribute of R must be in R_1, R_2, \dots, R_n i.e.,

$$\text{Attributes}(R_1) \cup \text{Attributes}(R_2) \cup \dots = \text{Attributes}(R)$$

Step-2: Intersection of attributes of R_1, R_2, \dots, R_n must not be NULL i.e.,

$$\text{Attributes}(R_1) \cap \text{Attributes}(R_2) \cap \dots \text{not NULL.}$$

Step-3: Common attributes must be a key for atleast one of the relation of R_1 (or) R_2 (or).... R_n .

$\text{Attr}(R_1) \cap \text{Attr}(R_2) \cap \dots \cap \text{Attr}(R_n) \rightarrow \text{Attr}(R_1)$

$\text{Attr}(R_1) \cap \text{Attr}(R_2) \cap \dots \cap \text{Attr}(R_n) \rightarrow \text{Attr}(R_1)$

Ex-1: A relation R(ABCD) is decomposed into R1(ABC) and R2(AD) with FD set is {A→BC }.

Sol:

step-1: $ABC \cup AD = ABCD$

step-2: $ABC \cap AD = A$

step-3: A is key for R1(ABC) because A→BC.

Ex-2: A relation R(ABCD) is decomposed into R1(AB), R2(AD) and R3(BD) with FD set is {A→BC , B→C, C→D, D→B}.

Sol:

step-1: $AB \cup BC \cup BD = ABCD$

step-2: $AB \cap BC \cap BD = B$

step-3: B is key for R2(BC) because B→C.

(2)Dependency Preserving Decomposition:

If we decompose a relation R into R_1, R_2, \dots, R_n then the dependencies of R must be part of R_1 (or) R_2 (or) R_n (or) must be derived from combination of FD's R_1 and R_2 , R_2 and R_3 ,.....

Ex-1: $R(ABCD)$ and FD is $A \rightarrow BC$ that can be decomposed into $R_1(ABC)$ and $R_2(AD)$.

Sol: Here Functional Dependency $A \rightarrow BC$ is part of R_1 . So it is Dependency Preserving.

Ex-2: Consider a schema $R(ABCD)$ and FD's are $A \rightarrow B$, $C \rightarrow D$ then decomposition of R into $R_1(AB)$ and $R_2(CD)$. [C]

- (A) Dependency preserving and Loss less join
- (B) Loss less join but not dependency preserving
- (C) Dependency preserving but not loss less join
- (D) Neither nor

Ex-3: $R(ABCD)$ that can be decomposed into $R_1(AB), R_2(BC), R_3(BD)$ and set of FD's are $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}$ is [B]

- (A) Dependency preserving and Loss less join
- (B) Loss less join but not dependency preserving
- (C) Dependency preserving but not loss less join
- (D) Neither nor

Key and Non-Key Attributes:

Key attributes are part of primary key. Non-Key attributes are not part of primary key.

Partial and Transitive Dependency:

All Non-key attributes in a table should be totally depend on part of primary key is called as “partial dependency(PD)”.

Ex-1: R(ABCD)
AB- \rightarrow C
B- \rightarrow D

Sol: Key: (AB)⁺ = ABCD
AB- \rightarrow C (not PD)
B- \rightarrow D (PD because D is a non-key and B is a part of primary key AB).

Ex-2: R(ABCDEFGH IJ)
AB- \rightarrow C (not PD)
A- \rightarrow DE (**PD**)
B- \rightarrow F (**PD**)
F- \rightarrow GH (not PD)
D- \rightarrow IJ (not PD)

Sol: **Key:** (AB)⁺ = ABCDEFGH IJ
A- \rightarrow DE, B- \rightarrow F are partial dependencies.

Transitive Dependency:

If there is a relationship between Non-key attributes(both LHS & RHS) is known as “Transitive Dependency(TD)”.

Ex-1: R(ABCD)

AB→C

C→D

Sol: **Key:**(AB)+=ABCD

AB→C (not TD)

C→D (TD because C,D are Non-keys)

Ex-2: R(ABCDEFGH IJ)

AB→C (not TD)

A→DE (not TD)

B→F (not TD)

F→GH (**TD**)

D→IJ (**TD because Both LHS & RHS are non-keys**)

TYPES OF NORMAL FORMS:

(1)1NF(First-Normal Form)

(2)2NF(Second-Normal Form)

(3)3NF(Third-Normal Form)

(4)BCNF(Boyce- codd Normal Form)

(1)1NF(First-Normal Form):

A table is said to be in first normal form then it satisfies following conditions:

(i) Column name should be unique.

(ii) Rows must be Unique.

(iii) Each column suppose to have same domain value.

(iv) No restriction on No.of rows.

(v) Intersection of rows and columns must be a single value.

Ex:

Student table

sid	name	course
1	a	DBMS,OS
2	b	JAVA,DAA
3	c	IDA,DBMS
4	d	DS

Sol:

The above given table not in first normal form because different courses in a course column.

1NF student table

sid	name	course
1	a	DBMS
1	a	OS
2	b	JAVA
2	b	DAA
3	c	IDA
3	c	DBMS
4	d	DS

Now, the table is in a First Normal Form.

(2)2NF(Second-Normal Form):

A table is said to be in 2NF if it is already in 1NF and also free from PD.

Note:

If there is a PD in a table, remove the partial dependent attributes from the original table and place it in a separate table along with the copy of determinant.

Example: R(ABCDEFGH IJ)

AB \rightarrow C

A \rightarrow DE (PD)

B \rightarrow F (PD)

F \rightarrow GH

D \rightarrow IJ

Sol: **Key:AB**

R₁: ADEIJ

R₂: BFGH

R₃: AB

Example-2: R(ABCDEFGH IJ)

AB \rightarrow C (PD)

BD \rightarrow EF (PD)

AD \rightarrow GH (PD)

A \rightarrow I (PD)

H \rightarrow J

Sol: **Key: ABD**

R₁: ABCI R11: ABC
 R12: AI

R₂: BDEF

R₃: ADGHJ

R₄: ABD

ABC
AI
BDEF
ADGHJ
ABD

(3)3NF(Second-Normal Form):

A table is said to be in 3NF if it is already in 1NF and also free from Transitive Dependency.

Note:

If there is a Transitive Dependency in a original table, remove those transitive dependency attributes from 2NF table and place it in a seperate table along with copy of determinant.

Example: R(ABCDEFGHIJ)

AB -> C

A -> DE (PD)

$B \rightarrow F$
 $F \rightarrow GH$
 $D \rightarrow IJ$

Sol: **Key: AB**

$R_1: ADEIJ$ \swarrow ADE
 \searrow DIJ

$R_2: BFGH$ \swarrow BF
 \searrow FGH

$R_3: AB$

Now the table in 3NF,

ADE
DIJ
BF
FGH
AB

(4)BCNF(Boyce-Codd Normal Form):

A table is said to be in BCNF if it is already in 3NF and all determinants are keys.


Example: $R(ABCDEFGHIJ)$

$AB \rightarrow C$
 $A \rightarrow DE$ (PD)
 $B \rightarrow F$ (PD)
 $F \rightarrow GH$ (TD)

$D \rightarrow IJ$ (TD)

Sol: **Key:** AB

$R_1: ADEIJ$ 

$R_2: BFGH$ 

$R_3: AB$

Now the table in 3NF,

ADE
DIJ
BF
FGH
AB

From the attributes, AB, A, B, F, D are keys.

Note: Every LHS attributes are Keys.