Database Management System

Unit-3

Dharmendra Kumar(Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research, Prayagraj

RELATIONAL DATABASE DESIGN

1 Functional dependency

Consider a relation schema R, and let $\alpha \subseteq R$ and $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on relation schema R if, in any legal relation r(R), for all pairs of tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, then $t_1[\beta] = t_2[\beta]$ must also satisfy with in r(R).

Super key: A subset α of a relation schema R is said to be super key of R if $\alpha \rightarrow R$ holds.

Candidate key: A subset α of a relation schema R is said to be super key of R if
(1) α should be super key of R i.e. $\alpha \rightarrow R$.
(2) There should not exist any proper subset K of α such that K $\rightarrow$ R.

Example: Consider the following relation :-

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

Find out which functional dependencies are satisfied.

Solution: Observe that A$\rightarrow$C is satisfied. There are two tuples that have an A value of a_1. These tuples have the same C value namely, c_1. Similarly, the two tuples with an A value of a_2 have the same C value, c_2. There are no other pairs of distinct tuples that have the same A value. The functional dependency C $\rightarrow$ A is not satisfied, however. To see that it is not, consider the tuples $t_1 = (a_2, b_3, c_2, d_3)$ and $t_2 = (a_3, b_3, c_2, d_4)$. These two tuples have the same C values, c_2, but they have different A values, a_2 and a_3, respectively. Thus, we have found a pair of tuples t_1 and t_2 such that $t_1[C] = t_2[C]$, but $t_1[A] \neq t_2[A]$.
Some other functional dependencies which satisfied are the following:-
AB $\rightarrow$ C, D $\rightarrow$ B, BC $\rightarrow$ A, CD $\rightarrow$ A, CD $\rightarrow$ B, AD $\rightarrow$ B, AD $\rightarrow$ C.

1.1 Trivial functional dependency

A functional dependency $\alpha \rightarrow \beta$ is said to be trivial if $\beta \subseteq \alpha$.

Some trivial functional dependencies are the following:- ABC $\rightarrow$ C, CD $\rightarrow$ C, A $\rightarrow$ A.

1.2 Closure of a Set of Functional Dependencies

Consider F is a set of functional dependencies defined on relation schema R.
Closure of F is the set of all the functional dependencies which are logically implied(or derived) from F. It is denoted by F^+.

1.3 Armstrong's axioms

Following three rules are said to be Armstrong's axioms.

- **Reflexivity rule:** If α is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ holds.
- **Augmentation rule:** If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds.
- **Transitivity rule:** If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

Some additional rules are the following:-

- **Union rule:** If $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
- **Decomposition rule:** If $\alpha \rightarrow \beta\gamma$ holds then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ holds.
- **Pseudo transitivity rule:** If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\gamma\alpha \rightarrow \delta$ holds.

Example: Consider relation schema R = (A, B, C, G, H, I) and the set F of functional dependencies A $\rightarrow$ B, A $\rightarrow$ C, CG $\rightarrow$ H, CG $\rightarrow$ I, B $\rightarrow$ H. We list several members of F+ here:

- A $\rightarrow$ H. Since A $\rightarrow$ B and B $\rightarrow$ H hold, we apply the transitivity rule.
- CG $\rightarrow$ HI . Since CG $\rightarrow$ H and CG $\rightarrow$ I , the union rule implies that CG $\rightarrow$ HI.
- AG $\rightarrow$ I. Since A $\rightarrow$ C and CG $\rightarrow$ I, the pseudo transitivity rule implies that AG $\rightarrow$ I holds.

Algorithm to compute F^+ using Armstrong's axioms
In this algorithm, the input will be F and R. It is computed by following algorithm:-
Note: The left-hand and right-hand sides of a functional dependency are both subsets of R. Since a set of size n has 2^n subsets, therefore there are a total of $2 \times 2^n = 2^{n+1}$ possible functional dependencies, where n is the number of attributes in R. [h] F and R F^+ $F^+ \leftarrow F$
F^+ does not change any further for each functional dependency f in F^+ apply reflexivity and augmentation rules on f
add the resulting functional dependencies to F^+ each pair of functional dependencies f_1 and f_2 in F^+ f_1 and f_2 can be combined using transitivity rule add the resulting functional dependency to F^+ A procedure to compute F+

1.4 Closure of attribute sets

Consider relation schema R and a set of functional dependencies F. Let $\alpha \subseteq$ R.
The closure of α is the set of all the attributes of R which are logically determined by α under a set F. It is denoted by α^+.
The closure of α is computed by following algorithm:- α and F α^+ = result $result \leftarrow \alpha$ changes to result each functional dependency $\beta \rightarrow \gamma$ in F $\beta \subseteq result$ $result \leftarrow result \cup \gamma$
An algorithm to compute α^+, the closure of α under F
Example: Consider relation schema R = (A, B, C, G, H, I) and the set F of functional dependencies A → B, A → C, CG → H, CG → I, B → H. Compute the closure of {A,G}, {C,G} and {A}.
Solution:

$\{A, G\}^+$ = {A,G}
= {A,B,C,G}
= {A,B,C,G,H,I}

Therefore, $\{A, G\}^+$ = {A,B,C,G,H,I}

$\{C, G\}^+$ = {C,G}
= {C,G,H,I}

Therefore, $\{C, G\}^+$ = {C,G,H,I}

$\{A\}^+$ = {A}
= {A,B,C}
= {A,B,C,H}

Therefore, $\{A\}^+$ = {A,B,C,H}

Uses or applications of attribute closure
There are several uses of the attribute closure:

- To test if α is a superkey, we compute α^+, and check if α^+ contains all attributes of R.
- We can check if a functional dependency $\alpha \rightarrow \beta$ holds by checking if $\beta \subseteq \alpha^+$.
- It gives us an alternative way to compute F^+: For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$.

1.5 Canonical Cover

Before defining canonical cover, first we are going to define some concepts related with it.
Extraneous attribute: Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.

- Attribute A is said to be extraneous in α if A $\in \alpha$, and F logically implies (F - $\{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
- Attribute A is said to be extraneous in β if A $\in \beta$, and the set of functional dependencies (F - $\{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.

Example: Consider F = { AB → C and A → C}. Find extraneous attributes in F.
Solution: Consider the functional dependency AB → C. In this dependency, right hand side contains single attribute, therefore right hand side has no extraneous attribute.
Now, consider left hand side. Now, we check A is extraneous attribute or not.

Eliminate A from FD, AB $\rightarrow$ C. We get B $\rightarrow$ C. Clearly, B $\rightarrow$ C can not be derived from F, therefore A is not extraneous attribute.
Now, we check B is extraneous attribute or not.
Eliminate B from FD, AB $\rightarrow$ C. We get A $\rightarrow$ C. Clearly, A $\rightarrow$ C is derived from F, therefore B is an extraneous attribute.
Consider the functional dependency A $\rightarrow$ C. In this dependency, left hand and right hand side contains single attribute, therefore this FD has no extraneous attribute.

Example: Consider F = {AB $\rightarrow$ CD and A $\rightarrow$ C}. Find extraneous attributes in F.
Solution: Consider the functional dependency AB $\rightarrow$ CD. In this FD, both sides may contain extraneous attributes.
Now, consider left hand side. Now, we check A is extraneous attribute or not.
Eliminate A from FD, AB $\rightarrow$ CD. We get B $\rightarrow$ CD. Clearly, B $\rightarrow$ CD can not be derived from F, therefore A is not extraneous attribute.
Now, we check B is extraneous attribute or not.
Eliminate B from FD, AB $\rightarrow$ CD. We get A $\rightarrow$ CD. Clearly, A $\rightarrow$ CD can not be derived from F, therefore B is not extraneous attribute.
Now, we check C is extraneous attribute or not.
Eliminate C from FD, AB $\rightarrow$ CD. We get AB $\rightarrow$ D. Clearly, Set F' = {AB $\rightarrow$ D, A $\rightarrow$ C} derives set F, therefore C is an extraneous attribute.
Now, we check D is extraneous attribute or not.
Eliminate D from FD, AB $\rightarrow$ CD. We get AB $\rightarrow$ C. Clearly, Set F' = {AB $\rightarrow$ C, A $\rightarrow$ C} can not derive set F, therefore D is not an extraneous attribute.
Consider the functional dependency A $\rightarrow$ C. In this dependency, left hand and right hand side contains single attribute, therefore this FD has no extraneous attribute.

Redundant functional dependency A functional dependency $\alpha \rightarrow \beta$ in F is said to be redundant if after eliminating $\alpha \rightarrow \beta$ from F, we get a set of functional dependency F' equivalent to F. That is, $F^{+} = F'^{+}$.
Example: Consider F = {A $\rightarrow$ B, B $\rightarrow$ C, and A $\rightarrow$ C}. In this set F, FD A $\rightarrow$ C} is redundant because it is derived from A $\rightarrow$ B and B $\rightarrow$ C using transitivity rule.

Canonical Cover:
Canonical cover is defined for a set F of functional dependencies.
Canonical cover of F is the minimal set of functional dependencies equivalent to F that is canonical cover is a set of functional dependencies equivalent to F which does not contain any extraneous attribute and redundant FD. It is denoted by F_c.
A canonical cover for a set of functional dependencies F can be computed by following algorithm.
[h] F F_c $F_c \leftarrow F$
F_c does not change any further Use the union rule to replace any dependencies in F_c of the form $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1\beta_2$.
Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β.
/* Note: the test for extraneous attributes is done using F_c, not F */
If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$.
Computing canonical cover

Example: Consider the following set F of functional dependencies on relation schema R = (A,B,C):
A $\rightarrow$ BC
B $\rightarrow$ C
A $\rightarrow$ B
AB $\rightarrow$ C
Compute the canonical cover for F.
Solution:

- There are two functional dependencies with the same set of attributes on the left side of the arrow:
  A $\rightarrow$ BC, A $\rightarrow$ B
  We combine these functional dependencies using union rule into A $\rightarrow$ BC.
- A is extraneous in AB $\rightarrow$ C because F logically implies (F - {AB $\rightarrow$ C}) $\cup$ {B $\rightarrow$ C}. This assertion is true because B $\rightarrow$ C is already in our set of functional dependencies.
- C is extraneous in A $\rightarrow$ BC, since A $\rightarrow$ BC is logically implied by A $\rightarrow$ B and B $\rightarrow$ C.

Thus, canonical cover of F is
F_c = {A $\rightarrow$ B, B $\rightarrow$ C}.

Note: A canonical cover might not be unique.

Example: Consider the following set F of functional dependencies on relation schema R = (A,B,C):
A $\rightarrow$ BC
B $\rightarrow$ AC
C $\rightarrow$ AB
Compute the canonical cover for F.
Solution: If we apply the extraneity test to A BC, we find that both B and C are extraneous under F. However, it is incorrect to delete both. The algorithm for finding the canonical cover picks one of the two, and deletes it. Then,

1. If C is deleted, we get the set F' = {A $\rightarrow$ B, B $\rightarrow$ AC, and C $\rightarrow$ AB}. Now, B is not extraneous in the right hand side of A $\rightarrow$ B under F'. Continuing the algorithm, we find A and B are extraneous in the right hand side of C $\rightarrow$ AB, leading to two canonical covers
   F_c = {A $\rightarrow$ B, B $\rightarrow$ C, and C $\rightarrow$ A}, and
   F_c = {A $\rightarrow$ B, B $\rightarrow$ AC, and C $\rightarrow$ B}.
2. If B is deleted, we get the set F' = {A $\rightarrow$ C, B $\rightarrow$ AC, and C $\rightarrow$ AB}. This case is symmetrical to the previous case, leading to the canonical covers
   F_c = {A $\rightarrow$ C, C $\rightarrow$ B, and B $\rightarrow$ A}, and
   F_c = {A $\rightarrow$ C, B $\rightarrow$ C, and C $\rightarrow$ AB}.

1.6 Decomposition

Let R be a relation schema. A set of relation schemas $\{R_1, R_2, ..., R_n\}$ is a decomposition of R if

R = $R_1 \cup R_2 \cup$ .... $\cup R_n$

That is, $\{R_1, R_2, ..., R_n\}$ is a decomposition of R if, for i = 1, 2, ... , n, each R_i is a subset of R, and every attribute in R appears in at least one R_i.
Let r be a relation on schema R, and let $r_i = \Pi_{R_i}$(r) for i = 1, 2, ... , n. That is, $\{r_1, r_2, ..., r_n\}$ is the database that results from decomposing R into $\{R_1, R_2, ..., R_n\}$. It is always the case that

r $\subseteq r_1 \bowtie r_2 \bowtie \Delta\Delta\Delta \bowtie r_n$.

A decomposition $\{R_1, R_2, ..., R_n\}$ of R is a lossless-join decomposition if, for all relations r on schema R,

r = $\Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \bowtie$ .... $\bowtie \Pi_{R_n}(r)$..........(1)

If equation (1) is not satisfied, then this decomposition is said to be lossy decomposition.

1.7 Desirable Properties of Decomposition

Lossless-Join Decomposition
Let R be a relation schema, and let F be a set of functional dependencies on R. Let R_1 and R_2 form a decomposition of R. This decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies is in F+:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

In other words, if $R_1 \cap R_2$ forms a superkey of either R_1 or R_2, then the decomposition of R is a lossless-join decomposition.
Eample: Consider the following schema R and F.
R = (branch-name, branch-city, assets, customer-name, loan-number, amount)
F is the following :-
branch-name $\rightarrow$ branch-city assets
loan-number $\rightarrow$ amount branch-name
R is decomposed into following relation schemas:-
Branch-schema = (branch-name, branch-city, assets)
Loan-info-schema = (branch-name, customer-name, loan-number, amount)
Is this decomposition lossless or lossy?
Solution:
Branch-schema $\cap$ Loan-info-schema = {branch-name}
Clearly, branch-name is a super key of Branch-schema. Therefore this decomposition is lossless.

Dependency Preservation
Let F be a set of functional dependencies on a schema R, and let $R_1, R_2, ..., R_n$ be a decomposition of R. Let $F_1, F_2, ..., F_n$ is the set of dependencies corresponding to $R_1, R_2, ..., R_n$.
Let F' = $F_1 \cup F_2 \cup ... \cup F_n$
If F' = F, then the decomposition will be functionally dependency preserve.
If F' $\neq$ F , then the decomposition may or may not be functionally dependency preserve.

In this case, compute F^+ and F'^+. If $F^+ = F'^+$, then the decomposition will be functionally dependency preserve otherwise not.

Eample: Consider the following schema R and F.
R = (branch-name, branch-city, assets, customer-name, loan-number, amount)
F is the following :-
branch-name $\rightarrow$ branch-city assets
loan-number $\rightarrow$ amount branch-name
R is decomposed into following relation schemas:-
Branch-schema = (branch-name, branch-city, assets)
Loan-info-schema = (branch-name, customer-name, loan-number, amount)
Is this decomposition functionally dependency preserve?
Solution:
Let F_1 and F_2 are the set of functional dependencies corresponding to Branch-schema and Loan-info-schema respectively. Therefore
F_1 = { branch-name $\rightarrow$ branch-city assets} and
F_2 = { loan-number $\rightarrow$ amount branch-name}
Now, F' = $F_1 \cup F_2$ = {branch-name $\rightarrow$ branch-city assets, loan-number $\rightarrow$ amount branch-name}
Clearly, F' = F. Therefore this decomposition is functionally dependency preserve.

2 Normalization

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. Normalization rules divides larger tables into smaller tables and links them using relationships. The purpose of Normalization is to eliminate redundant (repetitive) data and ensure data is stored logically.

2.1 Anomalies in DBMS

Anomalies are problems that can occur in poorly planned, unnormalized databases where all the data is stored in one table (a flatfile database). There are three types of anomalies that occur when the database is not normalized. These are – **insertion, update and deletion** anomaly. Let's take an example to understand this. Consider a relation Emp-Dept.

Insertion anomaly: Let us assume that a new department has been started by the organization but initially there is no employee appointed for that department, then the tuple for this department cannot be inserted into this table as the E# will have NULL, which is not allowed as E# is primary key.
This kind of a problem in the relation where some tuple cannot be inserted is known as insertion anomaly.

Deletion anomaly:
Now consider there is only one employee in some department and that employee leaves the organization, then the tuple of that employee has to be deleted from the table, but in addition to that the information about the department also will get deleted.

E#	Ename	Address	D#	Dname	Dmgr#
123456789	Akhilesh	Ghaziabad	5	Research	333445555
333445555	Ajay	Kanpur	5	Research	333445555
999887777	Shreya	Lucknow	4	Administration	987654321
987654321	Sanjay	Mirjapur	4	Administration	987654321
666884444	Om Prakash	Lucknow	5	Research	333445555
453453453	Manish	Delhi	5	Research	333445555
987987987	Ishani	Prayagraj	4	Administration	987654321
888665555	Garvita	Prayagraj	1	Headquarters	888665555

Table 1: Emp-Dept

This kind of a problem in the relation where deletion of some tuples can lead to loss of some other data not intended to be removed is known as deletion anomaly.

Modification/update anomaly:
Suppose the manager of a department has changed, this requires that the Dmgr# in all the tuples corresponding to that department must be changed to reflect the new status. If we fail to update all the tuples of the given department, then two different records of employee working in the same department might show different Dmgr# leading to inconsistency in the database.
This is known as modification/update anomaly.

2.2 Normalization

To overcome these anomalies we need to normalize the data. In the next section we will discuss different types of normal forms. These normal forms are:-

1. First normal form(1NF)
2. Second normal form(2NF)
3. Third normal form(3NF)
4. Boyce-Codd normal form(BCNF)
5. Fourth normal form(4NF)
6. Fifth normal form(5NF)

First normal form(1NF):
As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.
Consider the following table:-
This table is not in 1NF as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.
To make the table complies with 1NF we should have the data like this:
Now this table is in 1NF.

Second normal form(2NF)

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 , 9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 , 8123450987

Table 2: Employee

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Table 3: Revised Employee table

Consider a relation schema R with set of functional dependencies F. A relation schema R is said to be in second normal form(2NF) if
(1) It is in 1NF.
(2) Every non-prime attribute is fully functionally dependent on candidate key.

Prime attribute An attribute which is a part of any candidate key is said to be prime attribute. And which is not part of any candidate key is said to be non-prime attribute.

Example: Consider a relation schema R = (A,B,C,D) with only one candidate key as {A,B}. In this case, A and B are prime attributes. C and D are non-prime attributes.

Partial functional dependent
A partial dependency means if the non-key attributes depend on the part of candidate key then it is said to be partial dependency.
An attribute is fully functional dependent on a set of attributes α, if it is functionally dependent on only α and not on any of its proper subset.

Note: A relation with a single-attribute primary key is automatically in at least 2NF.

Example: Consider following functional dependencies in relation R(A, B, C, D)
AB $\rightarrow$ C
BC $\rightarrow$ D
Is this relation in 2NF?
Solution:
First find candidate keys. Here candidate key is A,B. Clearly non-prime attributes are C and D. From functional dependencies, C and D are fully functional dependent, therefore R is in 2NF.

Example: Consider a relation- R (V , W , X , Y , Z) with functional dependencies-
VW $\rightarrow$ XY

Y → V
WX → YZ
Is this relation in 2NF?
Solution:
Here candidate keys are VW, WX and WY. Therefore non-prime attributes are Z. Clearly Z is fully dependent on candidate keys. Therefore R is in 2NF.

Example: Consider relation R(A, B, C, D, E) with set of following functional dependencies
A → BC,
CD → E,
B → D,
E → A
Is this relation in 2NF?
Solution:
Here candidate keys are A, E, CD, BC. Clearly all the attributes belong into some candidate keys, therefore no non-prime attribute. Therefore, R is in 2NF.

Third normal form(3NF)
A relation schema R is in third normal form (3NF) with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is a trivial functional dependency.
- α is a super key for R.
- Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

In other words, we can define 3NF as following:-
A relation schema R with set of functional dependencies F is said to in 3NF if
(1) It is in 2NF.
(2) No non-prime attribute transitively depends on any candidate key.

Example: Consider a relation- R (V , W , X , Y , Z) with functional dependencies-
VW → XY
Y → V
WX → YZ
Is this relation in 3NF?
Solution:
Here candidate keys are VW, WX and WY. Therefore non-prime attributes are Z. Clearly Z is fully dependent on candidate keys. Therefore R is in 2NF.
Now, Z is not transitively dependent on any candidate key. Therefore, it is in 3NF.

Example: Consider relation R(A, B, C, D, E) with set of following functional dependencies
A → BC,
C → E,
B → D,
Is this relation in 3NF?

Solution:
Here candidate key is A. Therefore, non-prime attributes are B, C, D, E. Since candidate key contains single attribute, therefore R is in 2NF.
Now, Clearly attributes D and E are transitively dependent on candidate key A, therefore R is not in 3NF.

Example: The relation schema Student_Performance (name, courseNo, rollNo, grade) has the following FDs:
name,courseNo → grade
rollNo,courseNo → grade
name → rollNo
rollNo → name
Is this relation in 3NF?
Solution:
Here candidate keys are {name, courseNo} and {rollNo, courseNo}. Therefore non-prime attributes are grade. Clearly, grade is fully functional dependent on candidate keys, therefore this relation schema is in 2NF.
Now, clearly grade is not transitively dependent on candidate keys, therefore this relation schema is in 3NF.

Boyce-codd normal form (BCNF)

A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is a trivial functional dependency.
- α is a super key for R.

Example: The relation schema Student_Performance (name, courseNo, rollNo, grade) has the following FDs:
name,courseNo → grade
rollNo,courseNo → grade
name → rollNo
rollNo → name
Is this relation in BCNF?
Solution:
Consider FD, name,courseNo → grade. Clearly, {name, courseNo} is a super key, therefore this satisfy 2nd criteria of BCNF.
Consider FD, rollNo,courseNo → grade. Clearly, {rollNo, courseNo} is a super key, therefore this satisfy 2nd criteria of BCNF.
Consider FD, name → rollNo. Clearly, this functional dependency not satisfy any condition of BCNF. Therefore, this is not in BCNF.

Example: Consider the following relation schemas and their respective functional dependencies:
Customer = (customer-name, customer-street, customer-city)
customer-name → customer-street, customer-city
Branch = (branch-name, assets, branch-city)

branch-name $\rightarrow$ assets, branch-city
Loan = (branch-name, customer-name, loan-number, amount)
loan-number $\rightarrow$ amount, branch-name

Clearly, Customer and Branch schema are in BCNF, because left side of functional dependency **customer-name $\rightarrow$ customer-street, customer-city** and **branch-name $\rightarrow$ assets, branch-city** is super key.
But Loan schema is not in BCNF, because left side of functional dependency **loan-number $\rightarrow$ amount, branch-name** is not super key and this functional dependency is also not trivial.

2.3 Decomposition into Normal form

Decomposition into 2NF
Example:
Let R = { A, B, C, D} and F = { AB $\rightarrow$ C, B $\rightarrow$ D }.
Is this relation schema in 2NF? If not then decompose it into 2NF.
Solution:
Here, Primary key = {A,B}.
Clearly, this is not in 2NF because partial dependency holds.
Now, we decompose R into R_1 and R_2 as the following:-
R_1 = (A, B, C), F_1 = {AB $\rightarrow$ C}
R_2 = (B, D), F_2 = {B $\rightarrow$ D}
Now, R_1 and R_2 are in 2NF.

Example:
Student-course-info(Name, Course, Grade, Phone-no, Major, Course-dept)
F = { Name $\rightarrow$ Phone-no Major, Course $\rightarrow$ Course-dept, Name Course $\rightarrow$ Grade }
Is this relation schema in 2NF? If not then decompose it into 2NF.
Solution:
Here, Primary key = {Name,Course}.
Clearly, this is not in 2NF because partial dependency holds.
Now, we decompose R into R_1, R_2 and R_3 as the following:-
R_1 = (Name, Phone-no, Major), F_1 = {Name $\rightarrow$ Phone-no Major}
R_2 = (Course, Course-dept), F_2 = {Course $\rightarrow$ Course-dept}
R_3 = (Name, Course, Grade), F_3 = {Name Course $\rightarrow$ Grade}
Now, R_1, R_2 and R_3 are in 3NF.

Decomposition into 3NF
Example:
Let R = { A, B, C, D} and F = { A $\rightarrow$ B, A $\rightarrow$ C, B $\rightarrow$ D }.
Is this relation schema in 3NF? If not then decompose it into 3NF.
Solution:
Here, Primary key = {A}.
Clearly, this is not in 3NF because transitivity dependency holds.
Now, we decompose R into R_1 and R_2 as the following:-
R_1 = (A, B, C), F_1 = {A $\rightarrow$ B, A $\rightarrow$ C}
R_2 = (B, D), F_2 = {B $\rightarrow$ D}

Now, R_1 and R_2 are in 3NF.

Example:
Let Banker-info = { branch-name, customer-name, banker-name, office-number} and F = { banker-name → branch-name office-number, customer-name branch-name → banker-name}.
Is this relation schema in 3NF? If not then decompose it into 3NF.
Solution:
Here, Primary key = {customer-name, branch-name}.
Clearly, this is not in 3NF because transitivity dependency holds.
Now, we decompose relation Banker-info into R_1 and R_2 as the following:-
R_1 = (banker-name, branch-name, office-number), F_1 = {banker-name → branch-name office-number}
R_2 = (customer-name, branch-name, banker-name), F_2 = {customer-name branch-name → banker-name}
Now, R_1 and R_2 are in 3NF.

Decomposition into BCNF
If R is not in BCNF, we can decompose R into a collection of BCNF schemas $R_1, R_2, ..., R_n$ by the algorithm. The decomposition that the algorithm generates is not only in BCNF, but is also a lossless-join decomposition.

```
R and F   result $result \leftarrow R$
done = false
compute $F^+$
not done
    (there is a schema $R_i$ in result that is not in BCNF) let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that holds on $R_i$ such that $\alpha \rightarrow R_i$ is not in $F^+$, and $\alpha \cap \beta = \phi$
$result \leftarrow (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$
done = true  BCNF decomposition algorithm
```

Example:
Consider the following schema:-
Lending-schema = (branch-name, branch-city, assets, customer-name, loan-number, amount)
F = { branch-name → assets branch-city, loan-number → amount branch-name}
Is this relation schema in BCNF? If not then decompose it into BCNF.
Solution:
Here, Primary key = {customer-name, loan-number}.
Clearly, this is not in BCNF.
Now, we decompose relation Lending-schema into R_1 and R_2 as the following:-
R_1 = (customer-name, loan-number, amount, branch-name)
F_1 = { loan-number → amount branch-name}
R_2 = (branch-name, assets, branch-city)
F_2 = { branch-name → assets branch-city}
Clearly, R_1 is not in BCNF. Therefore, we again decompose R_1.
Now, we decompose relation R_1 into R_3 and R_4 as the following:-
R_3 = (customer-name, loan-number)
$F_3 = \phi$
R_4 = (loan-number, amount, branch-name)

F_4 = { loan-number $\rightarrow$ amount branch-name}
Now, the final relation schema are R_2, R_3 and R_4. All these are in BCNF.

2.4 Algorithm to check if a decomposition is lossless

Input: A relation schema R($A_1, A_2, ..., A_n$) and a decomposition D = $\{R_1, R_2, ..., R_m\}$ and a set F of functional dependencies.

1. Create a matrix S with one row i for each relation R_i in D, and one column j for each attribute a_j in R.
2. Set S(i,j) = a_j , if relation schema R_i contains attribute A_j otherwise set S(i,j) = b_{ij}.
3. . *change* $\leftarrow$ *true*
   (change) each functional dependency $\alpha \rightarrow \beta$ in F (row i and j exists such that the same symbol appears in each column corresponding to attribute in α) one of the symbols in the β column is a_r Make other symbol to be a_r the symbols are b_{pk} and b_{qk} Make both of them b_{pk} change = false
4. If there exists a row in which all the symbols are a's, then the decomposition has the lossless. Otherwise decomposition has lossy.

Example:
Consider the following schema:-
R=(SSN, Ename, Pnumber, Pname, Plocation, Hours)
F= { SSN $\rightarrow$ Ename, Pnumber $\rightarrow$ Pname Plocation, SSN pnumber $\rightarrow$ Hours }
R_1 = (SSN, Ename)
R_2 = (Pnumber, Pname, Plocation)
R_3 = (SSN, Pnumber, Hours)
Find out decomposition of R in R_1, R_2, R_3 is lossless or lossy.
Solution:
First construct matrix. It will be order of 3×6.
The initialization table will be the following:-

After the first iteration , the table will be the following:- After the second iteration, the table will not changed. Therefore, the above table is the final table.
Since in this table, row 3 contains only a's symbol, therefore this decomposition is lossless.

Example:
Consider the following schema:-
R=(A, B, C, D, E)
F= { AB $\rightarrow$ CD, A $\rightarrow$ E, C $\rightarrow$ D }
R_1 = (A, B, C)
R_2 = (B, C, D)
R_3 = (C, D, E)
Is the decomposition of R in R_1, R_2, R_3 is lossless or lossy?
Solution:
First construct matrix. It will be order of 3×5.
The initialization table will be the following:- After the first iteration , the table will be the following:- After the second iteration, the table will not changed. Therefore, the

above table is the final table.
Since in this table, no row contains only a's symbol, therefore this decomposition is lossy.

3 Multivalued dependency

Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The multivalued dependency $\alpha \rightarrow\rightarrow \beta$ holds on R if in any legal relation r(R), for all pairs of t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exists two tuples t_3 and t_4 in r such that

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$
$t_3[\beta] = t_1[\beta]$
$t_4[\beta] = t_1[\beta]$
$t_3[R-\beta] = t_2[R-\beta]$
$t_4[R-\beta] = t_1[R-\beta]$

3.1 Trivial multivalued dependency

A multivalued dependency $\alpha \rightarrow\rightarrow \beta$ is said to be trivial if $\beta \subseteq \alpha$ or $\beta \cup \alpha = R$.

Note: If $\alpha \rightarrow \beta$, then $\alpha \rightarrow\rightarrow \beta$. That is, every functional dependency is also a multivalued dependency.

Example: Consider the following relation schema.
In this table, multivalued dependency
customer-name $\rightarrow\rightarrow$ customer-street customer-city
holds.

3.2 Axioms for Multivalued dependency

(1) Replication rule:
$X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$
(2) Reflexivity rule:
$X \rightarrow\rightarrow X$
(3) Augmentation rule:
$X \rightarrow\rightarrow Y \Rightarrow XZ \rightarrow\rightarrow Y$
(4) Union rule:
$X \rightarrow\rightarrow Y$ and $X \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow YZ$
(5) Complementation rule:
$X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow (R - X - Y)$
(6) Transitivity rule:
$X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Z - Y)$
(7) Intersection rule:
$X \rightarrow\rightarrow Y$ and $X \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Y \cap Z)$
(8) Difference rule:
$X \rightarrow\rightarrow Y$ and $X \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Y - Z)\ and\ X \rightarrow\rightarrow (Z - Y)$
(9) Pseudo transitivity rule:

$X \rightarrow\rightarrow Y$ and $XY \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Z - Y)$

(10) Coalescence rule:
Given that $W \subseteq Y$ and $Y \cap Z = \phi$, and if $X \rightarrow\rightarrow Y$ and $Z \rightarrow W$ then $X \rightarrow W$.

3.3 Closure under Multivalued dependency

Let D be the set of functional and multivalued dependencies.
The closure of D is the set of all functional and multivalued dependencies that are logically implied by D. It is denoted by D^+.

Example: Consider R = (A, B, C, G, H, I)
and F $=\{A \rightarrow\rightarrow B,\ B \rightarrow\rightarrow HI,\ CG \rightarrow H\}$
Find some members of D^+.
Solution: Some members of D^+ are the following:-

- $A \rightarrow\rightarrow CGHI$, By complementation rule in $A \rightarrow\rightarrow B$
- $A \rightarrow\rightarrow HI$, By transitivity rule in $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$
- $B \rightarrow H$, By coalescence rule in $B \rightarrow\rightarrow HI$ and $CG \rightarrow H$
- $A \rightarrow\rightarrow CG$, By difference rule in $A \rightarrow\rightarrow CGHI$ and $A \rightarrow\rightarrow HI$.

4 Fourth Normal Form(4NF)

A relation schema R is in fourth normal form(4NF) with respect to a set D of functional and multivalued dependencies if, for all multivalued dependencies in D^+ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow\rightarrow \beta$ is a trivial multivalued dependency.
- α is a super key for schema R.

Note: Every 4NF schema is in BCNF.

4.1 Decomposition in to 4NF

Following algorithm is used to decompose schema R into 4NF. Relation schema R and set D $R_1, R_2, ..., R_m$ $result \leftarrow R$
done=false
Compute D^+
not done
(there is a schema R_i in result that is not in 4NF w.r.t. D_i) let $\alpha \rightarrow\rightarrow \beta$ be a nontrivial multivalued dependency that holds on R_i such that $\alpha \rightarrow R_i$ is not in D_i, and $\alpha \cap \beta = \phi$
$result \leftarrow (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$
done = true BCNF decomposition algorithm **Example:** Consider the following relation

	SSN	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

	SSN	Ename	Pnumber	Pname	Plocation	Hours
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	a_2	a_3	a_4	a_5	a_6

	A	B	C	D	E
R_1	a_1	a_2	a_3	b_{14}	b_{15}
R_2	b_{21}	a_2	a_3	a_4	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	a_5

	A	B	C	D	E
R_1	a_1	a_2	a_3	a_4	b_{15}
R_2	b_{21}	a_2	a_3	a_4	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	a_5

loan-number	customer-name	customer-street	customer-city
L-23	Smith	North	Rye
L-23	Smith	Main	Manchester
L-93	Curry	Lake	Horseneck

loan-number	customer-name	customer-street	customer-city
L-23	Smith	North	Rye
L-23	Smith	Main	Manchester
L-93	Curry	Lake	Horseneck

schema. Find out this table is in 4NF or not. If not, then decompose it into 4NF.
Solution: In this table, multivalued dependency

customer-name $\rightarrow\rightarrow$ customer-street customer-city

holds.
Clearly, neither this multivalued dependency is trivial nor customer-name is super key. Therefore, this table is not in 4NF.
By using above algorithm, this table is decomposed as
R_1= (customer-name, loan-number)
R_2= (customer-name, customer-street, customer-city)
Now, R_1 and R_2 are in 4NF.

5 Join dependency

Given a relation schema R. let $R_1, R_2, ..., R_n$ are the projections of R. A relation r(R) satisfies the join dependency *$(R_1, R_2, ..., R_n)$, iff the join of the projection of r on R_i, $1 \leq i \leq n$, is equal to r.

r = $\Pi_{R_1} \bowtie \Pi_{R_2} \bowtie \Pi_{R_3} \bowtie .... \bowtie \Pi_{R_n}$

5.1 Trivial join dependency

A join dependency is trivial if one of the projections of R is R itself.

5.2 Project join normal form(PJNF) or 5NF

Consider a relation schema R and D is the set of functional, multivalued and join dependencies.
The relation R is in Project join normal form with respect to D if for every join dependency *$(R_1, R_2, ..., R_n)$, either of the following holds:-
(i) The join dependency is trivial.
(ii) Every R_i is a super key of R.

5.3 Exercise

1. Consider R = (A, B, C, D, E, F, G) and
   F= $\{A \rightarrow B,\ BC \rightarrow F,\ BD \rightarrow EG,\ AD \rightarrow C,\ D \rightarrow F,\ BEG \rightarrow FA\}$
   Calculate the following:-
   (a) $(A)^+$
   (b) $(ACEG)^+$
   (c) $(BD)^+$

2. Consider R = (A, B, C, D, E) and
   F= $\{A \rightarrow B,\ BC \rightarrow E,\ ED \rightarrow A\}$
   (a) List all the candidate keys for R.
   (b) Is R in third normal form?
   (c) Is R in BCNF?

3. Consider R = (A, B, C, D, E, F) and
   F= $\{AB \rightarrow C,\ C \rightarrow B,\ ABD \rightarrow E,\ AD \rightarrow C,\ F \rightarrow A\}$

The decomposition of R is
D={$R_1(B,C)$, $R_2(A,C)$, $R_3(A,B,D,E)$, $R_4(A,B,D,F)$}
Check whether the decomposition is lossless or lossy.

4. Consider R = (V, W, X, Y, Z) and
F= $\{Z \rightarrow V,\ W \rightarrow Y,\ XY \rightarrow Z,\ V \rightarrow WX\}$
Sate whether the following decomposition of schema R is lossless join decomposition. Justify your answer.
(i) R_1 = (V,W,X) and R_2 = (V,Y,Z)
(ii) R_1 = (V,W,X) and R_2 = (X,Y,Z)

5. Consider R = (A, B, C, D, E, F, G, H, I, J) and
F= $\{AB \rightarrow C,\ A \rightarrow DE,\ B \rightarrow F,\ F \rightarrow GH,\ D \rightarrow IJ\}$
Is R in 2NF? If not, then decompose it into 2NF.

6. Consider R = (A, B, C, D, E) and
F= $\{A \rightarrow BC,\ CD \rightarrow E,\ B \rightarrow D,\ E \rightarrow A\}$
(i) List all the candidate keys for R.
(ii) Compute the canonical cover.

7. Consider R = (A, B, C, D, E) and
F= $\{A \rightarrow\rightarrow BC,\ B \rightarrow\rightarrow CD,\ E \rightarrow\rightarrow AD\}$
Is R in 4NF? If not, then decompose it into 4NF.

8. Consider R = (A, B, C, D, E, F, G, H) and
F= $\{AB \rightarrow C,\ BC \rightarrow D,\ E \rightarrow F,\ G \rightarrow F,\ H \rightarrow A,\ FG \rightarrow H\ \}$
Is the decomposition of R into $R_1(A,B,C,D)$, $R_2(A,B,C,E,F)$, $R_3(A,D,F,G,H)$ lossless? Is it dependency preserving?

9. Define partial functional dependency. Consider the following two sets of functional dependencies F = A →C, AC →D, E →AD, E →H and G = A →CD, E →AH. Check whether or not they are equivalent.

10. Define Minimal Cover. Suppose a relation R (A,B,C) has FD set F = A → B, B →C, A → C, AB → B, AB → C, AC → B convert this FD set into minimal cover.

11. Write the difference between 3NF and BCNF. Find normal form of relation R(A,B,C,D,E) having FD set F= A→B,BC→E,ED→A.

6 AKTU Examination Questions

1. Explain normalization. What is normal form?

2. Describe Multivalued dependency.

3. What are the different types of anomalies associated with database?

4. Why do we normalize database?

5. Define partial functional dependency. Consider the following two sets of functional dependencies F = A →C, AC →D, E →AD, E →H and G = A →CD, E →AH. Check whether or not they are equivalent.

6. Define Minimal Cover. Suppose a relation R (A,B,C) has FD set F = A → B, B →C, A → C, AB → B, AB → C, AC → B convert this FD set into minimal cover.

7. Write the difference between 3NF and BCNF. Find normal form of relation R(A,B,C,D,E) having FD set F= A→B,BC→E,ED→A.

8. Define 2 NF.

9. Write difference between BCNF Vs 3 NF.

10. Short Notes of the Following
(i)MVD or JD
(ii) Normalization with advantages

11. Explain 1NF, 2NF, 3NF and BCNF with suitable example.

12. Consider the universal relation schema R = (A, B, C, D, E, F, G, H, I, J) and F= $\{AB \to C, A \to DE, B \to F, F \to GH, D \to IJ\}$
Determine the keys for R? Decompose R into 2NF.

13. Distinguish between functional dependency and multivalued dependency.

14. Define functional dependency. What do you mean by lossless decomposition ? Explain with suitable example how function dependencies can be used to show that decomposition is lossless.

7 Gate questions

1. From the following instance of a relation schema R(A,B,C), we can conclude that

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

(a) A functionally determines B and B functionally determines C
(b) A functionally determines B and B does not functionally determines C
(c) B does not functionally determines C
(d) A does not functionally determines B and B does not functionally determines C

2. Consider the following functional dependencies in a database.
Date_of_birth → Age, Age → Eligibility
Name → Roll_number, Roll_number → Name
Course_number → Course_name, Course_number → Instructor
(Roll_number, Course_number) → grade

The relation (Roll_number, Name, Date_of_birth, Age) is in
(a) Second normal form but not in third normal form
(b) Third normal form but not in BCNF
(c) BCNF
(d) None of the above

3. The relation schema student performance (name, courseNo, rollNo, grade) has the following functional dependencies:-
(name, courseNo) $\rightarrow$ grade
(rollNo, courseNo) $\rightarrow$ grade
name $\rightarrow$ rollNo
rollNo $\rightarrow$ name

The highest normal form of this relation schema is
(a) 2NF (b) 3NF (c) BCNF (d) 4NF