Module-04

① Explain an informal design guidelines for relational schema design. (July/aug 2022, July/aug 2021-18CS53, July-aug-2021 17CS53, Jan/Feb-2021-17CS53, Aug/sept 2020-17CS53, Feb/Mar 2022-15CS53) → 6m/8M.

(Or)

Explain the informal design guidelines used as measure to determine the Quality of relation schema design. (July/aug 2022-17CS53) → 8M.

→ There are four informal measures for relational schema and those are -

1) Semantics of the attributes
2) Reducing the redundant value in tuple
3) Reducing NULL values in tuple
4) Disallowing ~~them~~ generation of spurious tuples.

1) Semantics of the attributes

The term sematics of the attributes refers to meaning of the attributes. It tells how to interpret the values stored

If the semantic is easier then the relational schema would be better

Guideline 1 :- 'Design the relation schema so that it is easy to explain the meaning. Do not combined attributes from multiple entity types into single relation'.

Note that the relational schema should always correspond to one entity type or one

Eg,

EmpNo	Empname	Deptno	Deptname	Project name

2) Reducing redundant values in tuple.

Redundancy is a condition created in database in which same piece of data is held at two different places.

Problems caused by redundancy :-

i) Redundant storage :- some information is stored repeatedly.

ii) Update anomalies :- if one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

iii) Insertion anamalies :- Due to insertion of new record repeated information get added to the relation schema.

iv) Deletion anamalies :- Due to deletion of particular record some other important information associated with the deleted record get deleted and we may lose some other important information from the schema.

Eg:- Following example illustrates the above discussed anamalies or redundancy problems consider following schema in which all possible information about employee is stored.

EmpID	Ename	Salary	Dept ID	Dept name	Dept loc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai

Redundancy!!

i) Redundant storage :- Note that the information about deptID, deptname and deptloc is repeated.

ii) Update anamalies :- In above table if we change Deptloc of pune to chennai, then it will result inconsistancy as for deptID 101 the deptloc is pune. or otherwise, we need to update multiple Copies of deptloc from pune to chennai. Hence this is an update anamaly.

iii) Insertion anamalies :- For above table if we want to add new tuple say (5, EEE, 50000) for deptID 101 then it will cause repeated information of (101, XYZ, pune) will occur.

iv) Deletion anamalies :- For above table if we delete a record for empID 4, then auto--matically information about the deptID 102, deptname PQR and deptloc mumbai will get deleted and may not be aware about deptID 102, This causes detion anamaly.

Guideline 2 :-

* Design relation schemas so that no insertion, deletion or modification anamalies are present in the relations.
* If any anamalies are present, note then clearly and make sure that the application programs that update the database will operate Correctly.

3) Reducing NULL Values in tuple.

Following the reasons why NULL Values appear,

i) attribute does not apply to the tuple
ii) The attribute Values for this tuple is unknown
iii) The Values is Known but not recorded

Guideline 3:-

* As far as possible, avoid placing attributes in the base relation whose values frequently be NULL.
* If nulls are unavoidable they should be applied only to exceptional cases and not to ~~P~~ majority of tuples.

4) Disallowing generation of spurious tuples.

Splitting of tables based on non primary key values results in spurious tuples or incorrect information.

Bad designs for a relational database may result in erroneous results for certain JOIN operations.

Guideline 4:-

* Design relation schemas so that they can be joined with equality conditions on primary key/foreign keys, which guarantees no spurious tuples to be generated.
* Do not have relations that contain matching attributes other than PK / FK combinations.

Eg, Consider poorly designed relation student_info.

RollNo	Subject_Code	Name	Subjectname	Location
1	S101	AAA	Maths	Pune
1	S102	AAA	English	Mumbai
2	S101	BBB	Maths	Pune
2	S102	BBB	English	Mumbai
3	S101	CCC	Maths	Pune.

The poorly designed table will split on the basis of Location which is a non-primary Key attribute. the resultant relation would be -:

Student_Loc

Name	Location
AAA	Pune
AAA	Mumbai
BBB	Pune
BBB	Mumbai
CCC	Pune

Student_subject

Roll_No	Subject_Code	Subject name	Location
1	S101	Maths	Pune
1	S102	English	mumbai
2	S101	Maths	Pune
2	S102	English	mumbai
3	S101	Maths	Pune

Now if student_loc and student_subject tables are joined on the basis of Location then we will not get the Original table.

---

(2) Consider two sets of functional dependency $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$ AND $G = \{A \rightarrow CD, E \rightarrow AH\}$. Find the canonical form ?
(July/aug 2022-18CS53) → 6M

→ Suppose we minimize LHS first, then go through each production rule one by one considering LHS.

Step 1) :- Suppose we minimize LHS, then go through each production rule one by one considering LHS.

$A \rightarrow C$, Keep it as it is

$AC \rightarrow D$, Here $A \rightarrow C$ and $A \rightarrow D$, so we remove $A \rightarrow C$, hence $A \rightarrow D$ is kept by eliminating C from LHS.

$E \to AD$, Keep it as it is as E is a single attribute at LHS.

$E \to H$, Keep it as it is.

Step 2):- Now we will minimize RHS.

$A \to C$, Keep it as it is

$A \to D$, Keep it as it is

$E \to AD$. That means $E \to A$ and $E \to D$.

as $A \to D$ is also present in the FD, so we get $E \to A$ and $A \to D$. Thus $E \to D$ is transitive. Hence neglect it. So we keep $E \to A$ only.

$E \to H$, Keep it as it is.

Step 3):- From step 1 and 2, we get minimal Cover of FD as

$A \to C$

$A \to D$

$E \to A$

$E \to H$

Hence the Canonical form is

$A \to CD$

$E \to AH$.

③ What is the need for normalization ? explain 1NF, 2NF and 3NF with examples (July/Aug 2022, Jan/Feb 2021, 1(8M) 53 Aug/Sept 2020, July/Aug 2022-17CS53, July/Aug 2021-17CS53) -8M

⇒ • Normalization is the process of reorganizing data in a database so that it meets two basic requirements:

1) There is no redundancy of data (all data is stored in only one place).

2) Data dependencies are logical (all related data items are stored together).

* Need for normalization :

1) It eliminates redundant data.
2) It reduces chances of data error.
3) The normalization is important because it allows database to take up less disk space.
4) It also help in increasing the performance.
5) It improves the data integrity and consistency.

① First Normal Form (1NF):

The table is said to be in 1NF if it follows following rules :

i) It should only have single (atomic) valued attributes/ columns.
ii) values stored in a column should be of the same domain.
iii) All the columns in a table should have unique names.
iv) And the order in which data is stored does not matter.

consider following student table

Student

sid	Sname	phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. we can make it in 1NF. The conversion is as follows-

sid	Sname	phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

3 Second Normal Form (2NF):

- In the 2NF, relational must be in 1NF.
- In the Secondas normal form, all non-key attributes are fully functional dependent on the primary key.

Example:

Let's assume, a school can store the data of teachers and the subjects. they teach. In a school, a teacher can teach more than one subject.

TEACHER table:

TEACHER_ID	SUBJECT	TEACHER_AGE
25	chemistry	30
25	Biology.	30
47	English	35
83	Math.	38
83	Computer	38

- In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID. which is a proper subset of a candidate key. That's why it violates the rule of 2NF.

- To convert the given table into 2NF, we decomes it into two tables.

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	chemistry.
25	Biology.
47	English
83	Math
83	Computer

3 Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X→Y

1. X is a super key

2. Y is a ~~primary~~ prime attribute, i.e, each element of Y is part of some candidate key.

Ex:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Bangalore
333	Stephan	02228	US	Mysore.
444	Lan	60007	US	Davanagere
555	Katharine	06389	UK	Mandya.
666	John	462007	MP	Mangalore.

- Super key in the above table is:

~~Candidate keys~~ $\boxed{\text{\{EMP\_ID\}, \{EMP\_ID, EMP\_NAME\}, \{EMP\_ID, EMP\_NAME, EMP\_ZIP\} --- So on.}}$

- candidate key : $\boxed{\text{\{EMP\_ID\}}}$

Non-prime attributes:

- In the given table, all attributes except EMP_ID are ~~EMP_CITY~~ non-prime attribute.

- Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The Non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on Super key (EMP_ID). It violates the rule of third normal form.

- Thats why we need to move the EMP_CITY & EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephen	02228
444	Lan.	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Banglare
02228	US	Mysore
60007	US	Davanagere
06389	UK	Mandya
462007	MP	Mangalore.

(4) Consider the relation

EMP_PROJ = {SSN, PNumber, Hours, EName, PName, PLocation}

Assume {SSN, PNumber} as primary key.

The dependencies are :

{SSN, PNumber} → {Hours}

SSN → {EName}

PNumber → {PName, PLocation}

Normalize the above relation into 2NF. and explain second normal form and need for normalization (10 Marks).

⇒ consider the given dependencies

(1) {ssn, Pnumber} ⟶ Hours

(2) ssn ⟶ Ename

(3) Pnumber → {PName, Plocation}

The dependencies 2 and 3 represents the partial dependency.

Hence we convert the relation into second normal form by splitting the given Emp-proj into three relations

Emp = {<u>ssn</u>, Ename}

Proj = {<u>Pnumber</u>, Pname, Plocation}

works = {<u>ssn, Pnumber</u>, Hours}

* (For second normal form and need for normalization refer ~~Befo~~ previous question).

5. Consider the following relations for published books.

Book (Book_title, Author_name, Book_type, List_price, Author_Application, publisher).

Suppose the following dependencies ~~exist~~ exists

Book_title → publisher, Book_type

Book_type → List_price

Author_name → Author_Application.

a) what normal form is the relation in? explain your answer

b) Apply normalization until you cannot decompose the relations further, state the reasons further, state the reasons behind each decomposition. (8M)

Ans:

a) Given the Relation:

Book (Book_title, Author_name, Book_type, Listprice, Author-application, publisher).

FD:

1) Book_title → publisher, Book_type

2) Book_type → List_price

3) Author_name → Author_Application.

- The key for this relation is:

>>> Book_title, Author_name

- This relation is is 1NF and not in 2NF as no attributes are FFD on the key. It is also not in 3NF.

ⓑ i) 2NF decomposition:

- Book0 (Book_title, Author_name)
- Book1 (Book_title, publisher, Book_type, Listprice)
- Book2 (Authorname, AuthorApplication).

ii) 3NF decomposition

- Book0 (Book_title, Author_name)
- Book1-1 (Book_title, publisher, Book_type)
- Book1-2 (Book_type, Listprice)
- Book2 (Authorname, Author_Application)

This decomposition eliminates the transitive dependency of Listprice.

⑥ Explain Second and Third Normal forms with example? How BCNF is stronger than 3NF? (8M)

⇒ (For the Explanation of second & Third Normal form. refer previous question).

BCNF is a stronger form of Normalization than 3NF because it eliminates the Second condition for 3NF, which allowed the right side of the FD to be a prime attribute.
Thus, every left side of an FD in a table must be a Super key. Every table that is BCNF is also 3NF, 2NF & 1NF

(or)

A Relation R is in 3NF if and only if every dependency A→B staisfied by R meets at least ONE of the following criteria:

1) A→B is trival (i.e. B is subset of A)
2) A is a super key
3) B is a subset of a candiate key. (BCNF doesn't permit the third of these options)

∴ BCNF is said to be stronger than 3NF because 3NF permits some dependencies which BCNF does not,

7. What do you understand by attributes closure ? Give an example. (July/August 2022 - 18CS53, Jan/Feb - 2017 - 17CS53) (4M)

⇒ closure of an attribute can be defined as a set of attributes that can be functionally determined from it.

(or)

closure of an attribute x is the set of all attributes that are functional dependencies on x with respect to F. It is denoted by x^+ which means what x can determine

Algorithm:

Let's see the algorithm to compute x^+

- Step 1 - $x^+ = x$
- Step 2 - repeat until x^+ does not change
  - For each FD $Y \to Z$ in F
    - If $Y \subseteq x^+$ then $x^+ = x^+ \cup Z$

Example:

R(A, B, C, D) AND F: $A \to B, B \to C, C \to D, A \to E$.

Find the closure of F.

Sol:

$A^+ = \{A, B, C, D, E\}$

$B^+ = \{B, C, D\}$

$C^+ = \{C, D\}$

$F^+ = \{A \to A, A \to B, A \to C, A \to D, A \to E, B \to B, B \to C,$
$B \to D, C \to C, C \to D\}$

(8) What is functional dependency ? Explain the inference rules for functional dependency with proof — 8m

(Or)

State and prove the inference rules for functional dependencies (July August 2021 – 18CS531) — 6m.

Functional Dependencies:

A functional dependency $A \rightarrow B$ in a relation holds if two tuples having same value of attribute A also have the same value of attribute B. it is denoted by $A \rightarrow B$ where A is called determinant and B is called dependent.

for example :- Consider Student table as follows:

Roll	Name	City
1	A A A	Mumbai
2	B B B	Pune
3	C C C	Gandhinagar

Here

Roll → Name hold

But

Name → City does not hold.

- In above table, student roll number is unique hence each student's name & city can be uniquely identified using his roll number.

- But using name we cannot uniquely identify his/her city because there can be same names of the students. Similarly using city name we can not identify the student uniquely. As in the same city may belong to multiple students

<u>Inference Rule (IR)</u>

- Armstrong's axioms are the basic inference rule, used to conclude functional dependencies on a relational database.

- It provides a set of rules for a simple reasoning technique in functional dependencies

- An inference rule is an assertion that can apply a user on a set of functional dependencies to derive other FD (functional dependencies).

Inference rules are divided into major two parts:

1. Axioms or primary rules
2. Additional rules or secondary rules.

1. Axioms or primary rules
   (i) Reflexive rule R (IR1)
   (ii) Augmentation Rule (IR2)
   (iii) Transitive Rule (IR3)

2. Additional rules or secondary rules
   (i) Union Rule (IR4)
   (ii) Decomposition Rule (IR5)
   (iii) Pseudo transitive Rule (IR6)

The functional dependency has 6 types of inference rules:

(1) Reflexive Rule (IR 1)

If X is a set of attributes and Y is the subset of X, then X functionally determines Y. In the reflexive rule, if Y is a subset of X, then 'X' determines Y.

1. If $X \supseteq Y$ then $X \rightarrow Y$.

<u>Eg</u>: $X = \{a, b, c, d, e\}$
$Y = \{a, b, c\}$

(2) Augmentation Rule (IR 2)

The augmentation is also called a partial dependency. If X determines Y, then XZ determines YZ for any 'Z'

If X determines Y & Z is any attribute set, then XZ determines YZ. It is also called a partial dependency

1. If $X \rightarrow Y$ then $XZ \rightarrow YZ$

Eg: for R{A B C D}, if $A \rightarrow B$ then $AC \rightarrow BC$

(3) Transitive Rule (IR4)

In the transitive rule, if X determines Y and Y determines Z, then X must also determine Z. If X determines Y and Y determines Z, then X also determines Z.

→ If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

(4) Union Rule (IR4)

This rule also known as the additive rule. If X determines Y and X determines Z, then X also determines both Y and Z.

Union rule says, if X determines Y and X determines Z ~~and~~, then X must also determine Y & Z.

→ if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$.

Proof: (1) $X \rightarrow Y$ (given).
(2) $X \rightarrow Z$ (given)
(3) $X \rightarrow XY$ (using IR2 on 1 by augmentation with X, where $XX = X$).
(4) $XY \rightarrow YZ$ (using IR2 on 2 by augmentation with Y)
(5) $X \rightarrow YZ$ (using IR3 on 3 and 4)

(5) **Decomposition Rule** (IR5)
This rule is the reverse of the union rule and is also known as the project rule. If X determines Y and Z together, then X determines Y and Z separately.
Q. → If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Proof: (1) $X \rightarrow YZ$ (given)
(2) $YZ \rightarrow Y$ (using IR1 rule)
(3) $X \rightarrow Y$ (using IR3 on 1 and 2)

(6) **Pseudo transitive Rule** (IR6)
In the pseudo transitive rule, if X determines Y and YZ determines W, then XZ determines W.
→ If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

Proof:

(1) $X \to Y$ (given)

(2) $WY \to Z$ (given)

(3) $WX \to WY$ (using ER2 on 1 by augmenting with W).

(4) $WX \to Z$ (using ER3 on 3 and 2)

9) Define 4NF. when it is violated? why it is useful? (6M) (2 time)

Defe: (OR)

Define Multivalued dependency. explain 4NF with example. (3 times) (8M, 10M).

⇒ ★ 4NF (Fourth normal form).

4NF is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.

It builds on the first three normal forms (1NF, 2NF and 3NF) and the Boyce-codd Normal Form (BCNF).

• ~~To~~ (OR)

For a table to satisfy the fourth normal form, it should satisfy the following conditions:

1) It should be in the Boyce-codd Normal form (BCNF)
2) And, the table should not have any multivalued dependency.

★ 4NF violations:

• A table violates 4NF when it contains two / more independent multivalued dependencies.
• Note that BCNF is not violated because there are no Functional dependencies in this table. In fact the primary key for this table is made up of all three attributes.

★ How 4NF is useful

- The Benefit of conforming to BCNF or 4NF is very similar to the benefit of conforming to 2NF or 3NF
- It eliminates some harmful redundancy
- prevents certain cases where database contradicts itself.
- It's a rare case where a table can be in 3NF but fail to be in BCNF | 4NF.

★ Explanation for 4NF

consider following student relation which is not in 4NF as it contains multivalued dependency

student table

sid	course	skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French.

Now to convert the above table to 4NF. we must decompose the table into following two tables.

Student_course Table.

key : (Sid, course)

Sid	course
1	C
1	C++
2	Java

Student_skill Table

key : (sid, skill).

Sid	Skill
1	English
1	German
2	English
2	French

Thus the tables are now in 4NF.

- Definition for Multivalued Dependency

⇒ Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on third attribute

⇒ A Multivalued dependency consists of at least two attributes that are dependent on third. attribute that's why it always requires at least 3 attributes.

Ex:- consider Bike manufacturer company which produces two colour (white & Black) of each model every year

Bike_model	Manf_year	color
M2001	2008	Black
M2001	2008	white
M3001	2013	Black
M3001	2013	white

* Here columns color & manuf_year are dependent on Bike_model & independent of each other

In this case, 2 columns can be called as multivalued dependent on Bike_model

Representation :

Bike_model →→ Manuf_year

Bike_model →→ color

i.e, "Bike_model" multidetermined Manuf_year" &

" Bike_model multidetermined color ".

(10) Consider two sets of functional dependency $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$ And $G = \{A \rightarrow CD, E \rightarrow AH\}$. Are they equivalent?
- (July/August 2022 - 18CS53) - 6 m

→ First find the functional dependencies of both F and G and then check for $F \supseteq G$ and $G \subseteq F$

Step 1: For F

Using G functional dependencies -

$(A)^+ = \{A, C, D\}$ → As $A \rightarrow CD$ is in G

$(AC)^+ = \{A, C, D\}$ → As $A \rightarrow CD$ is in G

$(E)^+ = \{A, C, D, E, H\}$ → As $E \rightarrow AH$, $A \rightarrow CD$ is in G.

Using F functional dependencies -

$(A)^+ = \{A, C, D\}$ → As $A \rightarrow C$ and $AC \rightarrow D$ is in F

$(AC)^+ = \{A, C, D\}$ → As $AC \rightarrow D$ is in F

$(E)^+ = \{A, C, D, E, H\}$ → As $E \rightarrow AD$, $E \rightarrow H$ and $A \rightarrow C$ is in F.

Step 2: For G

Using F functional dependencies -

$(A)^+ = \{A, C, D\}$ → As $A \rightarrow C$ and $AC \rightarrow D$ is in F

$(E)^+ = \{A, C, D, E, H\}$ → As $E \rightarrow AD$, $E \rightarrow H$ is in F

Using G functional dependencies -

$(A)^+ = \{A, C, D\} \rightarrow$ As $A \rightarrow CD$ is in G

$(E)^+ = \{A, C, D, E, H\} \rightarrow$ As $E \rightarrow AH$ and $A \rightarrow CD$ is in G

Step 3: From both these steps

$G \subseteq F$ and $F \subseteq G$

Hence F and G are equivalent.

---

(11) What is functional dependency? Write an algorithm to find a minimal cover for a set of functional dependencies.

(July / August 2021 - 18CS53, Feb / Mar - 2022 - 17CS53, July / August 2021 - 17CS53, Aug / Sept 2020 - 17CS53, Feb / Mar 2022 - 15CS53) - 6 m, 8 m

→ A functional dependency $A \rightarrow B$ in a relation holds if two tuples having same value of attribute, A also have the same value for attribute B.

It is denoted by $A \rightarrow B$ where,

'A' is called determinant

'B' is called dependent

Algorithm for computing canonical cover for set of functional dependencies F.

$F_c = F$

repeat

Use the union rule to replace any dependencies in F_c of the form

$\alpha_1 \to \beta_1$ and $\alpha_1 \to \beta_2$ and $\alpha_1 \to \beta_1\beta_2$

Find a functional dependency $\alpha \to \beta$ in F_c with an extraneous attribute either in α or in β.

/* The test for extraneous attributes is done using F_c, not F */

If an extraneous attribute is found, delete it from $\alpha \to \beta$ in F_c.

until (F_c does not change)

(12) What is functional dependency? Write an algorithm to find minimal cover for set of functional dependencies. Construct minimal cover m for set of functional dependencies which are: E: {B → A, D → A, AB → D}

(Jan / Feb 2021 – 18 CS53) – 10 m

→ (For definition and algorithm refer Q. No: 11)

Given,

E: {B → A, D → A, AB → D}

Step 1: Split the FD such that R.H.S contain single attribute.

But here, as already RHS contains only atomic attribute so it remain as it is.

$B \rightarrow A$

$D \rightarrow A$

$AB \rightarrow D$

Step 2: Find the redundant entries and delete them. This can be done as follows-

* For $B \rightarrow A$: we first find $(B)^+$ by assuming that we delete $B \rightarrow A$ temporarily. We get $(B)^+ = B$. Thus from B it is not possible to obtain A by deleting $B \rightarrow A$. This means we cannot delete $B \rightarrow A$.

* For $D \rightarrow A$: we find $(D)^+$ by assuming that we delete $D \rightarrow A$ temporarily. We get $(D)^+ = D$. Thus from D it is not possible to obtain A by deleting $D \rightarrow A$. This means we cannot delete $D \rightarrow A$

* For $AB \rightarrow D$: we find $(AB)^+$ by assuming that we delete $AB \rightarrow D$ temporarily. We get $(AB)^+ = AB$. Thus from AB it is not possible to obtain D by deleting $AB \rightarrow D$. This means we cannot delete $AB \rightarrow D$.

Step 3: Now we have to simplify L.H.S

Consider $AB \rightarrow D$. Here we can split A and B. For that we find closure set of A and B.

$(A)^+ = A$

$(B)^+ = BAD$

Thus A can be obtained from both A and B. That also means we need not ~~have~~ to have AB on LHS.

Instead only B can be allowed and A can be eliminated. Thus we get,

$B \rightarrow D$

To summarise we get now,

$B \rightarrow A$

$D \rightarrow A$

$B \rightarrow D$

Step 4: The simplified LHS and RHS can be combined together to form.

$B \rightarrow AD$

$D \rightarrow A$

The minimal cover is $\{B \rightarrow AD, D \rightarrow A\}$

13 Find canonical cover of F. The FD = $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$ (July / August 2021 - 17CS53).

→ Given,

$F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$

Step 1: Split the FD such that R.H.S contain single attribute. Hence we get,

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C$

Step 2: Find the redundant entries and delete them. This can be done as follows -

* For $A \rightarrow B$: we find $(A)^+$ by assuming that we delete $A \rightarrow B$ temporarily. We get $(A)^+ = ABC$. Thus from A it is possible to obtain B by deleting $A \rightarrow B$. Thus,

A → B is redundant. We can delete it.

* For A → C: We find $(A)^+$ by assuming that we delete A → C temporarily. We get $(A)^+ = ABC$. Thus from A it is possible to obtain C by deleting A → C. Therefore A → C is redundant. We can delete it.

* For B → C: We find $(B)^+$ by assuming that we delete B → C temporarily. We get $(B)^+ = B$. Thus from B it is not possible to obtain C by deleting B → C. Therefore we cannot delete B → C.

* For A → B: We find $(A)^+$ by assuming that we delete A → B temporarily. We get $(A)^+ = A$. Thus from A it is not possible to obtain B by deleting A → B. This means we cannot delete A → B.

* For AB → C: We find $(AB)^+$ by assuming that we delete AB → C temporarily. We get $(AB)^+ = ABC$. Thus from AB it is possible to obtain C by deleting AB → C. Therefore AB → C is redundant. We can delete it.

To summarize we get now

B → C

A → B

Thus R.H.S gets simplified.

Step 3: L.H.S is already in simplified form.

Step 4: The simplified LHS and RHS can be combined together to form

A → B

B → C. The minimal cover is {A → B, B → C}

(14) Find the minimal cover of F: The given set of FDs be $F: \{A \rightarrow BCDE,\ CD \rightarrow E\}$
(July/August - 2021 - 17CS53)

$\rightarrow F: \{A \rightarrow BCDE,\ CD \rightarrow E$

Step 1: We will make right hand sides atomic.

$A \rightarrow B$
$A \rightarrow C$
$A \rightarrow D$
$A \rightarrow E$
$CD \rightarrow E$

Step 2: Now we will remove redundant FDs using RHS.

* For $A \rightarrow B$. Now compute $(A)^+$ without considering the $A \rightarrow B$. We get $(A)^+ = ACDE$. That means it is not possible to obtain B without $A \rightarrow B$ from A. Thus we cannot delete $A \rightarrow B$.

* For $A \rightarrow C$. Now compute $(A)^+$ without considering the $A \rightarrow C$. We get $(A)^+ = ABDE$. That means it is not possible to obtain C without $A \rightarrow C$ from A. Thus we cannot delete $A \rightarrow C$.

* For $A \rightarrow D$. Now compute $(A)^+$ without considering the $A \rightarrow D$. We get $(A)^+ = ABCE$. That means it is not possible to obtain D without $A \rightarrow D$ from A. Thus we cannot delete $A \rightarrow D$.

* For $A \rightarrow E$. Now compute $(A)^+$ without considering the

$A \to E$. We get $(A)^+ = ABCDE$. That means it is possible to obtain E without $A \to E$ from A. Thus $A \to E$ is redundant. We can delete it.

* For $CD \to E$: Now compute $(CD)^+$ without considering the $CD \to E$. We get $(CD)^+ = CD$. That means it is not possible to obtain E without $CD \to E$ from CD. Thus we cannot delete $CD \to E$

To summarise we get now

$A \to B$

$A \to C$

$A \to D$

$CD \to E$

Step 3 : Now we will simplify LHS

Consider $CD \to E$. Here we can split C and D. For that we find closure of C and D

$(C)^+ = C$

$(D)^+ = D$

So we cannot remove $CD \to E$.

Step 4: The simplified LHS and RHS can be combined together to form.

$A \to BCD$

$CD \to E$

The minimal cover is $\{A \to BCD, CD \to E\}$.

---

15 A set of functional dependencies for the relation r$\{A, B, C, D, E, F\}$ is $AB \to C$, $C \to A$, $BC \to$

D, $ACD \to B$, $BE \to C$, $EC \to FA$, $CF \to BD$, $D \to E$.
Find the minimal cover for this set of functional dependencies.

→ Step 1: we will make right hand sides atomic

$AB \to C$

$C \to A$

$BC \to D$

$ACD \to B$

$BE \to C$

$EC \to F$

$EC \to A$

$CF \to B$

$CF \to D$

$D \to E$

Step 2: Now we will remove redundant FDs using RHS.

* For $AB \to C$

$(AB)^+ = AB$ without considering $AB \to C$.
∴ We cannot obtain C without $AB \to C$ from AB.
Thus we cannot delete $AB \to C$

* For $C \to A$: $(C)^+ = C$ without considering $C \to A$ ∴ we cann-out obtain A without $C \to A$ from C. Thus we canno-t delete $C \to A$.

* For $BC \to D$: $(BC)^+ = BCA$ without considering $BC \to D$
∴ From BC we cannot obtain D without $BC \to D$.
Thus we cannot delete $BC \to D$.

* For $ACD \to B$: $(ACD)^+ = ACDEFBD$ without considering $ACD \to B$ ∴ From ACD we can obtain B without ACD

$\rightarrow B$. Thus it is redundant. We can delete it.

* For $BE \rightarrow C$: $(BE)^+ = BE$ without considering $BE \rightarrow C$
$\therefore$ We cannot obtain C without $BE \rightarrow C$ from BE.
We cannot delete $BE \rightarrow C$.

* For $EC \rightarrow F$: $(EC)^+ = ECA$ without considering $EC \rightarrow F$
$\therefore$ From EC we cannot obtain F without $EC \rightarrow F$. We cannot delete $EC \rightarrow F$.

* For $EC \rightarrow A$: $(EC)^+ = ECAFBD$ without considering $EC \rightarrow F$ $\therefore$ From EC we can obtain A without $EC \rightarrow A$. We can delete it as it is redundant.

* For $CF \rightarrow B$: $(CF)^+ = CFDEA$ without considering $CF \rightarrow B$.
$\therefore$ From CF we cannot obtain B without $CF \rightarrow B$. We cannot delete $CF \rightarrow B$.

* For $CF \rightarrow D$: $(CF)^+ = CFBDEA$ without considering $CF \rightarrow D$
$\therefore$ From CF we can obtain D without $CF \rightarrow D$. We can delete it as it is redundant.

* For $D \rightarrow E$: $(D)^+ = DE$ without considering $D \rightarrow E$
$\therefore$ From D we cannot obtain E without $D \rightarrow E$. We cannot delete $D \rightarrow E$.

To summarise we get

$AB \rightarrow C$

$C \rightarrow A$

$BC \rightarrow D$

$BE \rightarrow C$

$EC \rightarrow F$

$CF \rightarrow B$, $D \rightarrow E$

Step 3: Now we will simplify LHS.

For $AB \to C$

$(A)^+ = A$

$(B)^+ = B$

So we cannot remove $AB \to C$

For $BC \to D$

$(B)^+ = B$

$(C)^+ = CA$

So we cannot remove $BC \to D$

For $BE \to C$

$(B)^+ = B$

$(E)^+ = E$

So we cannot remove $BE \to C$

For $EC \to F$

$(E)^+ = E$

$(C)^+ = CA$

So we cannot remove $EC \to F$

For $CF \to B$

$(C)^+ = CA$

$(F)^+ = F$

So we cannot remove $CF \to B$

Step 4: The simplified LHS and RHS can be combined together to form

$AB \to C$ $\quad$ $BE \to C$

$C \to A$ $\quad$ $EC \to F$

$BC \to D$ $\quad$ $CF \to B$ $\quad$ $D \to E$

∴ The minimal cover is {AB → C, C → A, BC → D, BE → C, EC → F, CF → B, D → E}

(16) Find the minimal cover of FD E: {AB → D, B → C, AE → B, A → D, D → EF}

→ ~~Step 1~~: We will make right hand sides atomic

~~AB → D~~
B → C
AE → B
A → D
D → E
D → F

~~Step 2~~: Now we will remove redundant FDs using RHS

• For AB → D. Now compute $(AB)^+$ without considering the ~~AB → D~~ i.e {G − (AB → D)} we get $(AB)^+$ = {ABCDEF}. That means we can remove AB → D as it is redundant entry.

~~Hence~~ grammar is

B → C
~~AE~~ → B
A → D
D → E
D → F

• For B → C compute $(B)^+$ by considering {G − (B → C)}
$(B)^+$ = {ACEBDF}. As C is not present in this set, that means B → C is not redundant. So we can not remove it.

~~Hence~~ grammar is.

B → C
AE → B
A → D
D → E
D → F.

• For AE → B, we will compute $(AE)^+$ under {G − (AE → B)}
$(AE)^+$ = {AEDF} as B is not present in $(AE)^+$. So we cannot remove AE → B from Grammar. Hence grammar will be

$B \rightarrow C$
$AE \rightarrow B$
$A \rightarrow D$
$D \rightarrow E$
$D \rightarrow F$

- For $D \rightarrow E$, compute $(D)^+$ under $\{G - (D \rightarrow E)\}$ $(D)^+ = \{DE\}$. As E is not present in $(D)^+$, we cannot remove $D \rightarrow E$
- For $D \rightarrow E$, compute $(D)^+$ under $\{G - (D \rightarrow F)\}$ $(D)^+ = \{DE\}$ as 'F' is not present in $(D)^+$, we remove $D \rightarrow F$. Finally the grammar is.

$B \rightarrow C$
$AE \rightarrow B$
$A \rightarrow D$
$D \rightarrow E$
$D \rightarrow F$.

Step 3: Remove redundant entries based on RHS.
$B \rightarrow C$, $A \rightarrow D$, $D \rightarrow E$ and $D \rightarrow F$ as LHS is atomic. Now we consider $AE \rightarrow B$
For A: compute E^+ with respect $\{G - (AE \rightarrow B) \cup (E \rightarrow B)\}$
E^+ using $\{B \rightarrow C, E \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\} = EBC$
E^+ doesn't contain A, so A not redundant in $AE \rightarrow B$
For E: compute A^+ with respect to $\{G - (AE \rightarrow B) \cup (A \rightarrow B)\}$ A^+ using $(B \rightarrow C, A \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F)$ $= ABDEFC$. A^+ contains E, so E is redundant in $AE \rightarrow B$. Hence we consider, $AE \rightarrow B$ as $A \rightarrow B$

Finally minimal closure is $\{B \rightarrow C, A \rightarrow B,$
$A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

(17) Consider two sets of functional dependency $F = \{A \to C, AC \to D, E \to AD, E \to H\}$ and $G = \{A \to CD, E \to AH\}$. Are they equivalent?

$\Rightarrow$ We will find the functional dependencies of both F and G and then check for F and G $\supset$ FD2 and G $\supset$ F

<u>Step 1</u> :- For F

Using G functional dependencies -

$(A)^+ = \{A, C, D\}$ ← As $A \to CD$ is in G

$(AC^+) = \{A, C, D\}$ ← As $A \to CD$ is in G

$(E)^+ = \{A, C, D, E, H\}$ ← As $E \to AH$, $A \to CD$ is in G

Using F functional dependencies

$(A)^+ = \{A, C, D\}$ ← As $A \to C$ and $AC \to D$ is in F

$(AC)^+ = \{A, C, D\}$ ← $AC \to D$ is in F

$(E)^+ = \{A, C, D, E, H\}$ ← As $E \to AD$, $E \to H$ & $A \to C$ is in F

<u>Step 2</u> :- For G

Using F Functional dependencies

$(A)^+ = \{A, C, D\}$ ← As $A \to C$ and $AC \to D$ is in F

$(E)^+ = \{A, C, D, E, H\}$ ← As $E \to AD$, $E \to H$ is in F

Using G Functional dependencies

$(A)^+ = \{A, C, D\}$ ← As $A \to CD$ is in G

$(E)^+ = \{A, C, D, E, H\}$ ← As $E \to AH$ and $A \to CD$ is in G

Step 3 :- From both these steps

$G \subseteq F$ and $F \subseteq G$

Hence, F and G are equivalent.

---

(18) Given below are two sets of FD's for a relation R(A, B, C, D, E). Are they equivalent.

i) A->B, AB->C, D->AC, D->E   ii) A->BC, D->AE

⇒ We will assume these relations as F and G. That means,

F : A->B, AB->C, D->AC, D->E

G : A->BC, D->AE

Now, we will find F^+ and G^+ as follows

Step 1 :- For F

Using G Functional dependency

$(A)^+ = \{A, B, C\}$

$(AB)^+ = \{A, B, C\}$

$(D)^+ = \{D, A, C, E, B\}$

Using F Functional dependency

$(A)^+ = \{A, B, C\}$

$(AB)^+ = \{A, B, C\}$

$(D)^+ = \{D, A, C, E, B\}$

Step 2 :- For G

Using F functional dependencies

$(A)^+ = \{A, B, C\}$

$(D)^+ = \{D, A, C, E, B\}$

Using G functional dependencies

$(A)^+ = \{A, B, C\}$

$(D)^+ = \{D, A, C, E, B\}$

Step 3 :- From both these steps

$G \subseteq F$ and $F \subseteq G$

Hence, F and G are equivalent

(19) Consider the relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies $F = \{\{A, B\} \to C, A \to \{D, E\},$ $B \to F, F \to \{G, H\}, D \to \{I, J\}\}$

(1) What is the key for R? Demonstrate it using the Inference rules.

(2) Decompose R into 2NF, them 3NF relations

⇒ Let, $A \to DE$ (given)

$A \to D, A \to E$ (decomposition rule)

$D \to IJ, A \to IJ$

Using union rule we get

$A \to DEIJ$

As, $A \to A$

We get, $A \to ADEIJ$

Using augmentation rule we compute AB

$AB \to ABDEIJ$

But, $AB \to C$ (given)

$AB \to ABCDEIJ$

$B \to F$ (given) ∴ $B \to GH$ (transitivity)

$F \to GH$

$AB \to AGH$ is also true

$AB \to AF$ ∵ $B \to F$ (given)

Similarly

Thus now using union rule

$$AB \longrightarrow ABCDEFGHIJ$$

$\therefore$ AB is a Key

The table can be converted to <u>2NF</u> as

$R_1 = (A, B, C)$

$R_2 = (A, D, E, I, J)$

$R_3 = (B, F, G, H)$

The above 2NF relations can be converted to <u>3NF</u> as follows

$R_1 = (A, B, C)$

$R_2 = (A, D, E)$

$R_3 = (D, I, J)$

$R_4 = (B, E)$

$R_5 = (E, G, H)$

(20) Using the minimal cover algorithm, find the minimal cover for the following FD's:

$F = \{AB \rightarrow C, A \rightarrow D, BD \rightarrow C, D \rightarrow BG, AE \rightarrow F\}$

$\Rightarrow$ We will make right hand side atomic

$AB \rightarrow C$

$A \rightarrow D$

$BD \rightarrow C$

$D \rightarrow B$

$D \rightarrow G$

$AE \rightarrow F$

Step 2: Now we will remove redundant FD's using RHS

- For $AB \rightarrow C$ we compute $(AB)^+$ without considering the rule $AB \rightarrow C$ i.e, $(G-(AB \rightarrow C))$. We get $(AB)^+ = \{ABDBGC\}$ i.e, C is present by other way also in the set. Hence we can remove $AB \rightarrow C$ as it is redundant entry.
- For $A \rightarrow D$, we compute $(A)^+$ using $(G-(A \rightarrow D))$. We get $(A)^+ = \{A\}$. We can not get D. So it is not a redundant entry and we can not remove it.
- For $BD \rightarrow C$, we compute $(BD)^+$ using $(G-(BD \rightarrow C))$. We get $(BD)^+ = \{BDG\}$. This is also not a redundant entry and we can not remove it.

- For $D \rightarrow B$, let us compute $(D)^+$ using $(G - (D \rightarrow B))$. We get $(D)^+ = \{DG\}$. This again indicates that we can not get B without the rule $D \rightarrow B$. Hence it is not a redundant entry and we can not remove it.
- Similarly, we can conclude for $D \rightarrow G$ and $AE \rightarrow F$ as not redundant entries.
- Finally, the grammer will be

$A \rightarrow D$

$BD \rightarrow C$

$D \rightarrow B$

$D \rightarrow G$

$AE \rightarrow F$

Step 2: Now we will remove redundant entries based on LHS.

The $A \rightarrow D$, $D \rightarrow B$, $D \rightarrow G$ remain as it is in minimal cover as LHS is atomic.

Now consider $BD \rightarrow C$, we can replace this by $B \rightarrow C$ and eliminate D, as $D \rightarrow B$ is present.

Similarly consider $AE \rightarrow F$. But as we cannot replace it either by $A \rightarrow F$ or $E \rightarrow F$. So it is not redundant.

The minimal cover is $\{A \rightarrow D, D \rightarrow B, D \rightarrow G, B \rightarrow C, AE \rightarrow F\}$

(21) Which normal form is based on the concept of transitive functional dependency? Explain the same with an example. (July/August 2021 - 18CS53)

→ The Third Normal Form (3NF) is based on the concept of transitive functional dependency.

Third Normal Form:

A table is said to be in the third normal form ~~fro~~ where,

(i) It is in the second normal form (i.e., it does not have partial functional dependency.

(ii) It doesn't have transitive dependency.

X → Y

atleast one of the following conditions hold:

(i) X is a super key of table

(ii) Y is a prime attribute of table.

For example: Consider following table Student_details as follows -

sid	sname	zipcode	cityname	state
1	AAA	111111	Pune	Maharashtra
2	BBB	222222	Surat	Gujarat
3	CCC	333333	Chennai	TamilNadu
4	DDD	444444	Jaipur	Rajastan
5	EEE	555555	Mumbai	Maharashtra

Here, Super keys: {sid}, {sid, sname}, {sid, sname, zipcode}, {sid, zipcode, cityname}.... and so on.

* Candidate keys: {sid}

* Non-prime attributes: {sname, zipcode, cityname, state}

The dependencies can be denoted as

sid → sname

sid → zipcode

zipcode → cityname

cityname → state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows:

Student

sid	sname	zipcode
1	AAA	111111
2	BBB	222222
3	CCC	333333
4	DDD	444444
5	EEE	555555

Zip

zipcode	cityname	state
111111	Pune	Maharashtra
222222	Surat	Gujarat
333333	Chennai	Tamilnadu
444444	Jaipur	Rajastan
555555	Mumbai	Maharastra

(22) Normalize the below relation upto 3NF
(VTU Jan-20 Mark-10)

Module	Dept	lectures	Text
M_1	D_1	L_1	T_1
M_1	D_1	L_1	T_2
M_2	D_1	L_1	T_1
M_2	D_1	L_1	T_3
M_3	D_1	L_1	T_4
M_4	D_2	L_1	T_1
M_4	D_2	L_1	T_5
M_5	D_2	L_1	T_6

Solution:- The given relation is already in 1st normal form. But it has Insert, Delete and update anomalies because-

1) Insert anomalies:- We can not add a module (M) with no texts (T)

2) Delete anomalies: If we remove M_3, we remove L_2 as well.

3) update anomalies: To change lecturer for M_1, we have to chge two rows.

Hence we will Convert it to Second normal form.

Step 1:- We can define the functional dependency FD as

{Module, Text} -> {Lecturer, Dept}

But

{Module} -> {Lecturer, Dept}

That means lecturer and dept are partially dependent on the primary key. Hence for conversion of first normal form to second normal form we will decompose the given table into two tables as

Table 2a

Module	Dept	Lecturer
M_1	D_1	L_1
M_2	D_1	L_1
M_3	D_1	L_2
M_4	D_2	L_3
M_5	D_2	L_4

Table 2b

Module	Text
M_1	T_1
M_1	T_2
M_2	T_1
M_2	T_3

M_3	T_4
M_4	T_1
M_4	T_5
M_5	T_6

The relation is now in second normal form

Step 3:- The table 2a has Insert, Delete and update anomalies Because-

1) INSERT anomalies :- We can't add lecturers who teach no modules.

2) UPDATE anomalies:- To change the department for L_1 we must alter two rows.

3) DELETE anomalies:- If we delete M_3 we delete L_2 as well.

Hence, to eliminate these anomalies, we decompose table 2a into two tables and convert it to third normal form.

Step 4:- Hence we get

Table 3a

Lecturer	Dept
L_1	D_1
L_2	D_1
L_3	D_2
L_4	D_2

Table 3b:-

Module	Lecturer
M_1	L_1
M_2	L_1
M_3	L_2
M_4	L_3
M_5	L_4

Step 5:- Thus now the complete relation is decomposed into three tables and it is in third normal form. It is summarized as below.

Table 3a

Lecturer	Dept
L_1	D_1
L_2	D_1
L_3	D_2
L_4	D_2

Table 3b

Module	Lecturer
M_1	L_1
M_2	L_1
M_3	L_2
M_4	L_3
M_5	L_4

Table 2b

Module	Text
M_1	T_1
M_1	T_2
M_2	T_1
M_2	T_3
M_3	T_4
M_4	T_1
M_4	T_5
M_5	T_6

Customer_ID	Product
1111	Soap
2222	toothbrush
3333	toothpaste

Product	Price
Soap	10
toothbrush	20
toothpaste	30

Hence the above mentioned key constraints are satisfied. Thus the relational schema is is domain key normal form.

23) Why is the domain key normal form (DKNF) known as ultimate normal form?

(Jan/Feb 2021) -4m

The basic idea behind the DKNF is to specify the normal form that takes into account all the possible dependences & constraints

* DKNF to is a normal for used in database normalization which requires that the database contain no constraints other than Domain constraint and key constraint

i. Domain constraint: which specifies the possible value of some attribute.

ii. Key constraint: which specifies keys of some relation

Eg:- for domain constraint:

Consider Sales Table

Customer_id	Product	Price
1111	Soap	10
2222	Tooth brush	20
3333	Tooth paste	30

Suppose we impose the constraint as customer-id must be an integer value which is >1100. this is a domain constraint which can be Easily applied

* this constraint is has to apply as product is not a key. Hence to satisfy this constraint we must divide the SALES table into two tables as follows. In one table customer-id is key and in another product is a key.

24) Explain types of Anomalies / Update Anomalies with Examples. (Jan / Feb 2021) - 6M.

Example :-

Following example illustrate the anomalies

EmpID	EName	Salary	DeptID	DeptName	Deptloc
1	A	10000	101	XYZ	Pune
2	B	20000	101	XYZ	Pune
3	C	30000	101	XYZ	pune
4	D	40000	102	PQR	Mumbai

Redundancy.

1. Redundant storage : The information about DeptID, DeptName and DeptLoc is repeated.

2. Update Anomalies :-

In above table if we change DeptLoc of pune to chennai, then it will result inconsistency as for DeptID 101 the DeptLoc is Pune, OR otherwise we ~~not~~ need to update multiple copies of Deptloc from Pune to chennai. Hence this is an update anomaly.

3. Insertion Anomalies :

If we want to add new tuple say (5, E, 5000) for DeptID 101 then it will cause repeated information of (101, XYZ, Pune) will occur.

4. Deletion Anomalies :-

If we delete a record for EmpID 4, then

authomatically information about the DeptID 102, DeptName PQR and Deptloc Mumbai will get deleted and one may no be aware about DeptID 102. This Cause deletion anomaly.

(25) Explain Armstrong inference rules

(Jan | Feb 2021) (July | August 2022) - 5M | 4M.

The closure set of functional dependancy can be Computed using basic three rules which are also called Armstrong's Axioms.

These are as follows.

i) Reflexivity : if $X \supseteq Y$ then $X \longrightarrow Y$.

ii) Augmentation : if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z.

iii) Transitivity : if $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of FD are as follows.

- Union : if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- Decomposition : if $X \rightarrow YZ$ then $X \rightarrow Y$ & $X \rightarrow Z$.

(26) Consider the Schema $R = ABCD$, subjected to FDs $F = (A \rightarrow B, B \rightarrow C)$, and the non-binary parition $D1 = \{ACD, AB, BC\}$. State whether $D1$ is a lossless decomposition ? (Jan / Feb 2021) - 10M

→ Step 1: $R_1 = \{A, C, D\}$ $R_2 = \{A, B\}$ $R_3 = \{B, C\}$

$R_1 \cup R_2 \cup R_3 = \{A, B, C, D\}$ which is nothing but R.

Therefore the first condition for checking lossless is satisfied.

Step 2: Consider $R_1 \cap R_2 = \{A, C, D\} \cap \{A, B\} = \{A\}$

$R_2 \cap R_3 = \{A, B\} \cap \{B, C\} = \{B\}$.

Hence, second condition of intersection not being ϕ gets satisfied.

Step 3: Now, consider $R_1 \{A, C, D\}$, $R_2 = \{A, B\}$, $R_3 = \{B, C\}$

We already found that $R_1 \cap R_2 = \{A\}$

$A^+ = \{A, B, C\} \in$ attributes of R_2

Consider $R_2 \cap R_3$

$B^+ = \{B, C\} \in$ attribute of R_2

Thus the third condition gets satisfied.

This shows that the given decomposition is a lossless join.

97) Consider $R = \{A, B, C, D, E\}$ which is decomposed into $R_1 = \{A, B, C\}$, $R_2 = \{C, D, E\}$ with an $FD = \{A \to BC, CD \to E, B \to D, E \to A\}$ show that the above decomposition of schema R is not lossless join decomposition.

(July/August 2022) – 10m

→ Step 1: $R_1 \cup R_2$

Here the first condition for checking lossless is satisfied as $R_1 \cup R_2 = (A, B, C) \cup (C, D, E) = \{A, B, C, D, E\}$ which is nothing but R.

Step 2: Consider $R_1 \cap R_2 = (A, B, C) \cap (C, D, E) = \{C\}$
Hence, second condition of intersection not being ϕ gets satisfied.

Step 3: Now, consider $R_1(A, B, C)$ and $R_2(C, D, E)$ we already find that $R_1 \cap R_2 = \{C\}$
$C^+ = C$ which is neither complete set of attributes of R_2 or R_3

Hence it is not lossless join decomposition. Or in other words we can say it is lossy decomposition.

(28) Consider the relational schema $R = \{A, B, C, D, E\}$. Suppose the following dependencies hold:

$\{E \rightarrow A,\ CD \rightarrow E,\ A \rightarrow BC,\ B \rightarrow D\}$

State whether the following decomposition of R are lossless join decomposition or not. Justify.

a. $\{(A, B, C),\ (A, D, E)\}$

b. $\{(A, B, C),\ (C, D, E)\}$ (Feb/Mar-2022-17CS53)

→ (A) Step 1: $R_1 \cup R_2 =$

$R_1 = \{A, B, C\} \cup R_2\ \{A, D, E\} = \{A, B, C, D, E\}$ which is nothing but R. Therefore the first condition for checking lossless is satisfied.

Step 2: Consider $R_1 \cap R_2 = \{A, B, C\} \cap \{A, D, E\}$ $= \{A\}$. Hence, second condition of intersection not being ϕ gets satisfied.

Step 3: Now, consider $R_1\ \{A, B, C\}$ and $R_2\ \{A, D, E\}$ we already found that $R_1 \cap R_2 = \{A\}$

$A^+ = \{A, B, C, D, E\} \in$ attributes of R_1 & R_2. Thus the third condition gets satisfied. This shows that the given decomposition is a lossless join.

(B) Step 1: $R_1 = \{A, B, C\}$ $R_2 = \{C, D, E\}$

$R_1 \cup R_2 = \{A, B, C, D, E\}$ which is nothing but R. Therefore the first condition for checking lossless is satisfied.

Step 2: Consider $R_1 \cap R_2 = \{A, B, C\} \cap \{C, D, E\}$ $= \{C\}$. Hence, second condition of intersection not being ϕ gets satisfied.

Step 3: Now, consider $R_1\{A, B, C\}$ and $R_2\{C, D, E\}$

we already found that $R_1 \cap R_2 = \{C\}$

$C^+ = C$ which is neither complete set of attributes of R_2 or R_3.

Hence it is not lossless join decomposition or in other words we can say it is lossy decomposition.