# Functional dependencies (FD's)

* FD's play a key role in differentiating good database designs from bad database designs.
* A FD is a type of constraint.

**Definition:** Consider a relation schema R, and let $\alpha \subseteq R$ and $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on schema R,

* for all pairs of tuples $t_1$ and $t_2$ in the relation r such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$.
* The FD $\alpha \rightarrow \beta$ holds on schema r(R) if, in every legal instance of r(R) it satisfies the functional dependency.
* K is a super key of r(R) if the FD $K \rightarrow R$ holds on r(R), then $t_1[K] = t_2[K]$, it is also the case that $t_1[R] = t_2[R]$.

(ex) Schema (R),

Loaninfo = (loan#, bname, cname, amount)

r(R) loaninfo

| loan# | bname | cname | amount |
|---|---|---|---|
| 11 | KECnagar | Ramu | 5000 |
| 11 | KECnagar | Ravi | 5000 |
| 20 | Salem | Sita | 12000 |
| 35 | Erode | banu | 20000 |

instance I

Set of FDs

loan# → bname

loan# → amount

There is no FD,

loan# → cname

$t_1[11] = t_2[11]$ but

$t_1[\text{Ramu}] \neq t_2[\text{Ravi}]$

F : {loan# → bname, loan# → amount}

## Use of functional dependencies:

1. To test instances of relations to see whether they satisfy a given set F of FD's.
2. To specify constraints on the set of legal relations. If we wish to constrain ourselves to relations on schema R that satisfy a set of FD's, we say that F holds on R.

## Trivial and non trivial FD:

If there is a FD $X \rightarrow Y$, each attribute value in X uniquely determines each of the value of the attribute Y, then Y is said to be functionally dependent on X and X is also called as determinant.

1. A FD $X \rightarrow Y$ is said to be trivial if $Y \subseteq X$. [ex) dname, sno → dname, xy → x
   AB → A ]
2. A FD $X \rightarrow Y$ is said to be non trivial if $Y \not\subseteq X$. [ex) loan# → bname, A → B].

## Functional dependency theory

**Closure of a set of FD's:** For a given set F of FD's on a schema, there must be some other FD's also hold on the schema.

* These FD's are logically implied by F.
* When testing for normal forms, to consider all FD's that hold on the schema.

(Ex) $R(A, B, C, G, H, I)$ and the set of FD's F is

$$F : \begin{cases} A \to B \\ A \to C \\ CG \to H \\ CG \to I \\ B \to H \end{cases}$$

The FD $A \to H$ is logically implied. (ie)
Suppose that $t_1$ and $t_2$ are tuples such that, $t_1[A] = t_2[A]$ then $t_1[B] = t_2[B]$ then $t_1[H] = t_2[H]$

Definition of closure! If f is a set of FD's on a relation schema R, $F^+$ is the closure of F, is the set of FDs such that $F \subseteq F^+$ and no FD can be derived from f by using the Armstrong axioms (or) inference axioms (rules) that are not contained in $f^+$.

Armstrong Axioms!

1. Reflexivity rule! If $\alpha$ is a set of attributes and $\beta \subseteq \alpha$, then $\alpha \to \beta$ holds.
2. Augmentation rule! If $\alpha \to \beta$ holds and $\gamma$ is a set of attributes, then $\gamma\alpha \to \gamma\beta$ holds.
3. Transitivity rule! If $\alpha \to \beta$ holds and $\beta \to \gamma$ holds, then $\alpha \to \gamma$ holds.

Armstrong's axioms are sound and complete.

Sound! Do not generate any incorrect FD's

Complete: From a given set f of FD's, they allow us to generate all $f^+$.

4. Union rule: If $\alpha \to \beta$ holds and $\alpha \to \gamma$ holds then $\alpha \to \beta\gamma$ holds.
5. Decomposition rule: If $\alpha \to \beta\gamma$ holds, then $\alpha \to \beta$ holds and $\alpha \to \gamma$ holds.
6. Pseudotransitivity rule: If $\alpha \to \beta$ holds and $\gamma\beta \to \delta$ holds, then $\alpha\gamma \to \delta$ holds.

Procedure to compute $F^+$:

```
F+ = F
repeat
    for each FD f in F+
        apply reflexivity and augmentation rules on f
        add the resulting FD to F+
    for each pair of FDs f1 and f2 in F+
        if f1 and f2 can be combined using transitivity
        add the resulting FD to F+
until F+ does not change any further.
```

Problem: find the closure $F^+$ for a schema $R = (A, B, C, G, H, I)$ and the set F of FD's are $\{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$.

$F^+ = \{A \to B, A \to C, CG \to H, CG \to I, B \to H,$
$A \to H$ (transitivity rule), $CG \to HI$ (union rule), $AG \to I$ (pseudotransitivity), $\ldots\}$

Another way of finding $AG \to I$,
use Augmentation rule $AG \to CG$
Apply transitivity rule, $AG \to CG$, $CG \to I$
then $AG \to I$

**Closure of Attribute sets:** If there is a fd $\alpha \to \beta$ and to test whether $\alpha$ is a superkey (or) not.

**Algorithm to compute $\alpha^+$**

```
result := α ;
repeat
    for each fd β → γ in f do
        begin
        if β ⊆ result then result = result ∪ γ;
        end
until (result does not change)
```

**Problem:** List the candidate keys for R and the set of FDs are $f = \{A \to B, A \to C, CG \to H, CG \to I, B \to H\}$

**Soln:** $R(A, B, C, G, H, I)$

Consider a attributes AG, find $(AG)^+$

result = AG

repeat

$A \to B$ in f do

begin

if $A \subseteq AG$ then result = AG ∪ B

∴ result = AGB

end

$A \to C$ in f do

begin

if $A \subseteq AGB$ then result = AGB ∪ C

∴ result = ABCG

end

Candidate keys are {AG}

Canonical cover: Whenever a user performs an update on the relation, the database system must ensure that the update does not violate any fd (ie) all the fd's in f are satisfied in the new database state.

* The system must rollback the update if it violates any fd in f.

* To reduce the effort spent in checking for violations by testing a simplified set of fds that has the same closure as the given set.

* Simplified set is easy to test. Any database that satisfies the simplified set of FDS will also satisfy the original set and vice versa, since the two sets have the same closure. A simplified set can be constructed by canonical cover.

* A canonical cover $f_c$ for f is a set dependencies such that f logically implies all dependencies in $f_c$ and $f_c$ logically implies all dependencies in f. $f_c$ must have two properties.

i) No fd in $f_c$ contains an extraneous attribute.

ii) Each left side of FD in $f_c$ is unique. There is no two fds $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ in $f_c$

such that $\alpha_1 = \alpha_2$

**Extraneous Attributes!** consider a set f of fds and the fd $\alpha \to \beta$ in f.

1. Attribute A is extraneous in $\alpha$ if $A \in \alpha$ and f logically implies
$(F - \{\alpha \to \beta\}) \cup \{(\alpha - A) \to \beta\}$
2. Attribute A is extraneous in $\beta$ if $A \in \beta$ and f logically implies
$(F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\}$

(ex) i) The fds $AB \to C$ and $A \to C$ in f, then B is extraneous attribute in $AB \to C$

ii) The fds $AB \to CD$ and $A \to C$ in f, then C is extraneous attribute in $AB \to CD$

**Algorithm for computing canonical cover**

$F_c = F$

repeat

Use the union rule to replace any dependencies in $f_c$ of the form $\alpha_1 \to \beta_1$ and $\alpha_1 \to \beta_2$ with $\alpha_1 \to \beta_1 \beta_2$

find a fd $\alpha \to \beta$ in $f_c$ with an extraneous attribute either in $\alpha$ or in $\beta$

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

until ($f_c$ does not change)

Problem: Find the canonical cover for the following fds on schema (A, B, C).

$f : \{A \to BC,\ B \to C,\ A \to B,\ AB \to C\}$

The canonical cover for f is computed as

Step 1:

* There are two FDs with the same set of attributes on the left side.

$A \to BC$ $\quad$ $A \to B$

Combine these FDs, we get $A \to BC$

* from other two fds, $AB \to C$, $B \to C$, we get $AB \to C$

So, the FDs in f are $\{A \to BC, AB \to C\}$

Step 2: Find an extraneous attribute in the set,

i) A is extraneous attribute in $AB \to C$,

$(F - \{AB \to C\}) \cup \{(AB - A) \to C\}$

$(A \to BC, AB \to C - \{AB \to C\}) \cup B \to C$

$\therefore (A \to BC, B \to C)$

$B \to C$ is already in the set of FD's.

ii) C is extraneous attribute in $A \to BC$

$(F - \{A \to BC\}) \cup \{A \to (BC - C)\}$

$(A \to BC, B \to C - \{A \to BC\}) \cup A \to B$

$\therefore (B \to C, A \to B)$

$A \to B$ is already in the set of FD's.

$\therefore$ The canonical cover $F_c$ is $(A \to B, B \to C)$

A canonical cover might not be unique.

$F_c : (A \to C)$

Problem: Find the list of canonical covers for the set of fds $F = \{A \to BC, B \to AC, C \to AB\}$

## Data Redundancy and Update Anomalies

* The design goal of a relational database design is to group attributes into relations to minimize data redundancy and thereby reduce the file storage space required by the implemented base relations.

Staffbranch (Staffid, Sname, position, salary, branch#, baddress)
Staff (Staffid, Sname, position, salary, branch#)
branch (branch#, baddress)

Staffbranch

| staffid | sname | position | salary | branch # | baddress |
|---|---|---|---|---|---|
| S1 | Arun | manager | 30000 | B005 | Erode |
| S2 | Ajay | OA | 5000 | B001 | karur |
| S3 | Banu | clerk | 20000 | B002 | KEC nagar |
| S4 | Charu | cashier | 25000 | B002 | KEC nagar |
| S5 | devi | clerk | 20000 | B005 | Erode |

In this table, baddress is a redundant data and it is repeated for every staff. This could e avoided and appear only once for each brand n the branch relation, and only branch# is to be repeated in staff relation.

* Data redundancy leads to problems called update anomalies. This can be classified as insertion, deletion and modification anomalies.

Insertion Anomalies: i) insert new staff at B002, the same baddress is to be entered correctly.
ii) insert new branch details that currently has

no staff, it is necessary to enter nulls into the attributes of staff such as staffid, sname, position, salary. But staffid is a primary key and it is not allowed.

Deletion Anomalies! If only one staff is working at a branch and if that staff's information is deleted then branch details are also lost.

Modification Anomalies! To change the value of the attribute bname for the attribute branch# = B002, we must update the tuples of all staff located at B002.

To avoid update anomalies, decompose the original relation.

Staff

| staffid | sname | position | salary | branch# |
|---|---|---|---|---|
| S1 | Arun | manager | 30000 | B005 |
| S2 | Ajay | OA | 5000 | B001 |
| S3 | Banu | clerk | 20000 | B002 |
| S4 | Charu | cashier | 25000 | B002 |
| S5 | devi | clerk | 20000 | B005 |

branch

| branch# | bname |
|---|---|
| B005 | Erode |
| B001 | Karur |
| B002 | Krishnagiri |

## Loss of information via a bad decomposition

employee (ID, name, Street, city, salary)

decompose as

employee1 (ID, name) employee2 (name, street, city, salary)

employee

| ID | name | street | city | salary |
|---|---|---|---|---|
| 1009 | John | main | Delhi | 75000 |
| 1020 | John | North | mumbai | 67000 |

employee1

| Id | name |
|---|---|
| 1009 | John |
| 1020 | John |

employee2

| name | street | city | salary |
|---|---|---|---|
| John | main | Delhi | 75000 |
| John | North | mumbai | 67000 |

natural join

| ID | name | street | city | salary |
|---|---|---|---|---|
| 1009 | John | main | Delhi | 75000 |
| 1009 | John | North | mumbai | 67000 |
| 1020 | John | main | Delhi | 75000 |
| 1020 | John | North | mumbai | 67000 |

loss of information

So, the important two properties associated with decomposition are,

i) Lossless join : This property ensures that any instance of the original relation can be identified by applying natural join in the smaller relations at the corresponding instance.

SQL Query can be written as,

```
Select * from ( select R1 from r) natural join
              ( select R2 from r);
```

where $r(R)$ can be decomposed as $r_1(R_1)$ and $r_2(R_2)$

* This can be written in relational algebra as,

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

* A decomposition that is not a lossless decomposition is called as lossy decomposition.

Testing of lossless join property: Use fds to show certain decompositions are lossless.

* Let R, $R_1$, $R_2$ and F as,

R = instdept (ID, name, salary, dname, building, budget)

$R_1$ = instructor (ID, name, salary, dname)

R2 = department (dname, building, budget)

F = { ID → name, salary, dname → building, budget }

* If $R_1$, $R_2$ form a lossless decomposition of R if atleast one of the following fds is in $F^+$

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

(ie) if $R_1 \cap R_2$ forms a superkey of either $R_1$ or) $R_2$, the decomposition of R is lossless decomposition.

* Use attribute closure to find superkeys.

$ID^+$ = {ID, name, salary}

$dname^+$ = { dname, building, budget }

$R_1 \cap R_2$ = { ID, name, salary, dname } ∩ { dname, building, budget }

$R_1 \cap R_2$ = dname

* dname is a superkey which retrive the entire relation of $R_2$. So, the lossless decomposition rule is satisfied.

ii) Dependency Preservation property: This property ensures that a constraint on the original relation can be maintained by simply enforcing same constraint on each of the smaller relations. 

Problems!

1. Suppose that we decompose the schema $r(A, B, C, D, E)$ into $r_1(A, B, C)$ $r_2(A, D, E)$. Show that this decomposition is a loseless decomposition if the following set F of fds holds,
$F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
2. List all fds for the above problem
3. List the candidate keys for relation schema $R(A, B, C, D, E)$
$F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
4. Show that the decomposition is not a lossless decomposition if the schema $r(A, B, C, D, F)$ decompose into $r_1(A, B, C)$ $r_2(C, D, E)$ and the fds are
$F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
5. Consider the following set F of fds on the relation schema $r(A, B, C, D, E, F)$
$F = \{A \rightarrow BCD, BC \rightarrow DE, B \rightarrow D, D \rightarrow A\}$
   i) Compute $B^+$
   ii) Prove that AF is a superkey
   iii) Compute atleast two canonical covers.

6) List the three design goals for relational databases.

## Three goals of database design:

To avoid i) Repetition of information
ii) Inability to represent information
iii) Loss of information

## Normalization

* Database design may contain redundant and inconsistent data. To avoid these problems some refinement is made to the database design. The refinement process is normalization.

**Definition:** It is step-by-step process of decomposing a complex relation into simple relations.

* FD's is the concept used to define all normal forms.

## Types of functional dependencies

1. full functional dependency
2. Partial functional dependency
3. Transitive dependency

port (Stud#, course#, sname, cname, marks, grade)

FDs : { Stud#, course# → marks, sname
course# → cname
marks → grade

* Use attribute closure to find the candidate key. The key is (Stud#, course#)

Prime, non prime (or) key, non key attributes:

* Attributes which are present in candida

Key are called key (or) prime attributes. The remaining attributes are called non key (or) non prime attributes.

**Full functional dependency:**

If x and y are attributes, if there is a fd $x \rightarrow y$, then y is said to be fully dependent on x.

(ex) marks $\longrightarrow$ grade

**Partial functional dependency:**

If x and y are attributes, attribute y is partially dependent on the attribute x only if it is dependent on a subset of attribute x.

(ex) Stud# course# $\rightarrow$ dname

course# $\rightarrow$ cname

So, cname is partially dependent on Stud# course#

**Transitive functional dependency:**

If x, y and z are the attributes, there are fds $x \rightarrow y$, $y \rightarrow z$ then $\rightarrow z$ is said to be transitive fd.

(ex) Stud# course# $\rightarrow$ marks

marks $\longrightarrow$ grade

$\therefore$ grade transitively depends on (Stud#, course#)

## Normal forms

**first normal form (or) 1NF :** A relation is said to be 1NF, if it eliminate multivalued attributes and all the attributes of the relation R are atomic in nature.

**Atomic :** The smallest level to which data may be broken down and remain meaningful.

(ex)

| Sno | Sname | address details | | | area of interest |
|---|---|---|---|---|---|
| | | Door# | Street | city | |
| S1 | Ramu | 35 | Thindal | Erode | OS, C, C++ |
| S2 | Ravi | 42 | Grobi | Grobi | DBMS, Networks |
| S3 | Sita | 54 | Bhavani | Erode | OS, C, DBMS |

⇓ 1NF

| Sno | Sname | door# | Street | city |
|---|---|---|---|---|
| S1 | Ramu | 35 | Thindal | Erode |
| S2 | Ravi | 42 | Grobi | Grobi |
| S3 | Sita | 54 | Bhavani | Erode |

**Second Normal form (or) 2NF :** A relation is said to be in 2NF if and only if

i) It is in the first normal form and

ii) No partial dependency exists between non key attributes and key attributes (i.e) eliminate partial dependency.

(Ex) Retail (custid, itemid, cname, acc#, itemname, unitprice, classes, qtypurchased, netprice)

FDs:
- custid → cname, acc#
- itemid → itemname, unitprice, classes
- custid, itemid → qtypurchased, netprice
- unitprice → classes

Use attribute closure to find the candidate key.

{custid, itemid} is candidate key

Key attributes are {custid, itemid}

Non key attributes are {cname, acc#, itemname, unitprice, classes, qtypurchased, netamount}

From the fds,

Fully functionally dependent on key attribute

custid, itemid → qtypurchased, netprice

partial dependency with respect to key attribute

custid → cname, acc#

itemid → itemname, unitprice, classes

No dependency with respect to key attributes

unitprice → classes

Retail

| custid | itemid | cname | acc# | itemname | unitprice | classes | qtypurchased | netprice |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

2 NF

item purchase

| custid | itemid | qtypurchased | netprice |
|---|---|---|---|

itemclass

Customer

| custid | cname | acc# |
|---|---|---|

Itemclass

| Itemid | Itemname | Unitprice | classes |
|---|---|---|---|
| | | | |

Retail Table (Already in INF)

↓ 2NF process

Itempurchase table — Itemtable class — customertable

## Third normal form:

A relation R is said to be in 3NF if and only if

i) It is in 2NF

ii) NO transitive dependency exists between non key attributes and key attributes through another non key attribute.

from the fds,

itemid → itemname, unitprice, classes

unitprice → classes

∴ itemid → classes, the attribute classes is transitively dependent on itemid.

So, remove the transitive dependency attributes from the item class table.

Itemclass

| itemid | itemname | unit price | classes |
|---|---|---|---|

⇓ 3NF

Item

| Item id | item name | unit price |
|---|---|---|

class

| unit price | classes |
|---|---|

At the end of 3NF process, the original table Retail is split into

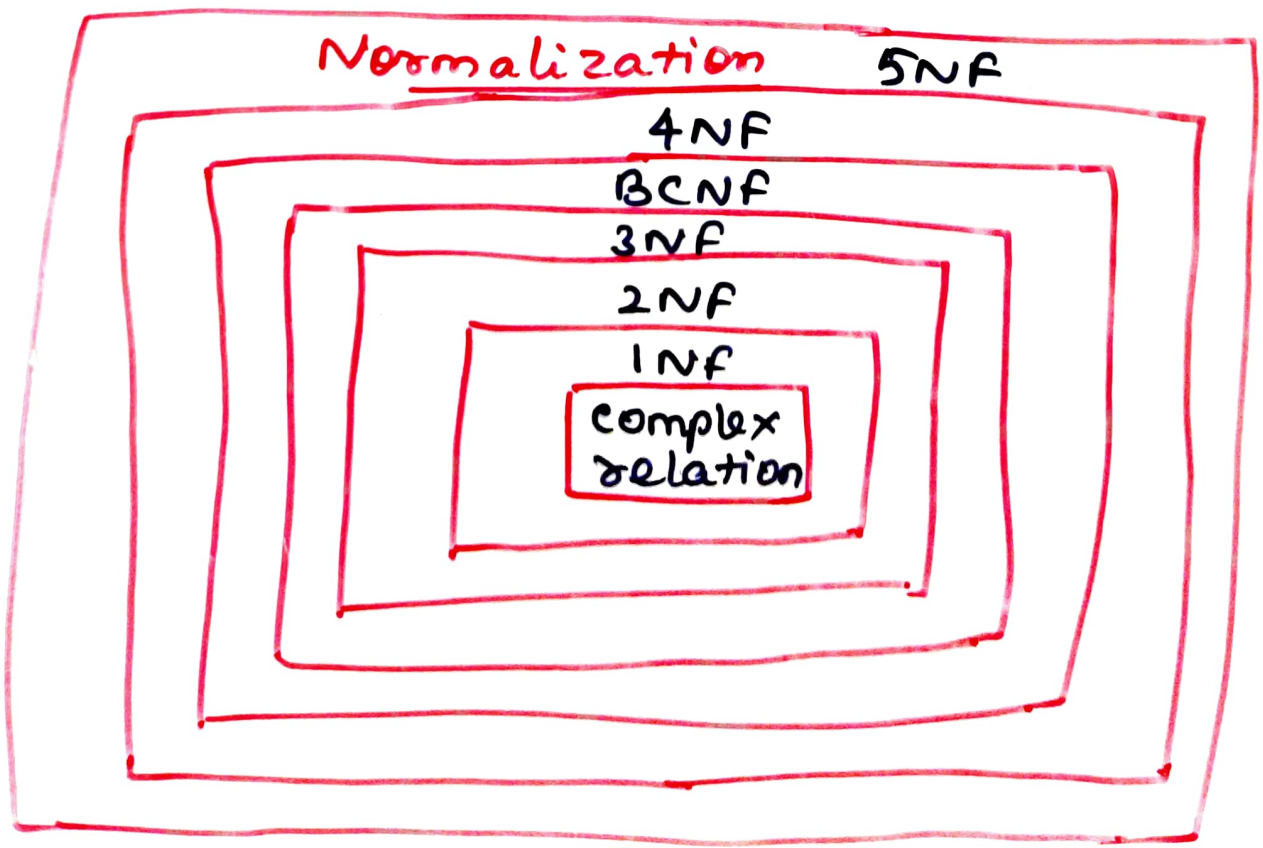
Retail (Already in 1NF)

⇓ 2NF process

itempurchase table | itemclasstable | customer table

⇓ 3NF process

itemtable | classtable



**3NF:** A relation schema R is in 3NF with respect to a set F of fds if, for all fds in $F^+$ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$ atleast one of the following holds:

* $\alpha \rightarrow \beta$ is a trivial fd ($\beta$ is a prime attribute of R)
* $\alpha$ is a superkey for R
* Each attribute A in $\beta - \alpha$ is contained in a candidate key of R.

**BCNF:** BCNF is more rigorous form of 3NF. A relational table is in BCNF if every determinant is candidate key. $\alpha \rightarrow \beta$ ($\alpha$: determinant)

* A relation schema R is in BCNF with respect to a set F of fds if, for all fds in $F^+$ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, atleast one of the following holds.
  * $\alpha \rightarrow \beta$ is a trivial fd
  * $\alpha$ is a superkey for R

**Difference between 3NF and BCNF**

* BCNF requires all non trivial dependencies of the form $\alpha \rightarrow \beta$ where $\alpha$ is a superkey.
* 3NF allows certain non trivial fds whose left side is not a superkey.

**Multivalued dependencies**

Let r(R) be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The multivalued dependency $\alpha \twoheadrightarrow \beta$ holds on R if, in any legal instance

of relation $r(R)$, for all pairs of tuples $t_1$ and $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples $t_3$ and $t_4$ in $r$ such that,

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$

$t_1[\beta] = t_3[\beta]$

$t_2[R-\beta] = t_3[R-\beta]$

$t_2[\beta] = t_4[\beta]$

$t_4[R-\beta] = t_1[R-\beta]$

Tabular representation of $\alpha \twoheadrightarrow \beta$

| | $\alpha$ | $\beta$ | $R-\beta-\alpha$ |
|---|---|---|---|
| $t_1$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $a_{j+1} \ldots a_n$ |
| $t_2$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $b_{j+1} \ldots b_n$ |
| $t_3$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $b_{j+1} \ldots b_n$ |
| $t_4$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $a_{j+1} \ldots a_n$ |

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

* $\alpha \twoheadrightarrow \beta$ is a trivial multivalued dependency
* $\alpha$ is a superkey of R.

## Other normal forms

5NF - Project-join normal form (PJNF)

6NF - Domain key normal form (DKNF)

**4NF Definition:** A table is in the 4NF if it is in BCNF and has no multivalued dependencies.A relation schema $R$ is in **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 hold:

- $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
- $\alpha$ is a superkey for schema $R$

**Multivalued Dependencies (MVDs) :**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 for tuples $t_1$ and $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples $t_3$ and $t_4$ in $r$ such that:

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

Tabular representation of $\alpha \rightarrow\rightarrow \beta$

| | $\alpha$ | $\beta$ | $R - \alpha - \beta$ |
|---|---|---|---|
| $t_1$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $a_{j+1} \ldots a_n$ |
| $t_2$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $b_{j+1} \ldots b_n$ |
| $t_3$ | $a_1 \ldots a_i$ | $a_{i+1} \ldots a_j$ | $b_{j+1} \ldots b_n$ |
| $t_4$ | $a_1 \ldots a_i$ | $b_{i+1} \ldots b_j$ | $a_{j+1} \ldots a_n$ |

The multi-valued dependency X→→Y holds in a relation R if whenever we have two tuples of R that agree (same) in all the attributes of X, then we can swap their Y components and get two new tuples that are also in R. For example the Student table is

| StudentID | AreasofInterest | Hobbies |
|---|---|---|
| 100 | OS, DBMS | Singing, Painting |
| 101 | JAVA | Reading |

Suppose a student can have more than one subject and more than one activity. StudentID can be associated with many subjects as well as many activities (multi-valued dependency).A database always contains atomic values. So, the above table is converted as

| StudentID | AreasofInterest | Hobbies |
|---|---|---|
| 100 | OS | Singing |
| 100 | DBMS | Painting |
| 100 | OS | Painting |
| 100 | DBMS | Singing |
| 101 | JAVA | Reading |

Suppose student 100 signs up for Acting. Then we would insert (100, OS, Acting).

| StudentID | AreasofInterest | Hobbies |
|---|---|---|
| 100 | OS | Singing |
| 100 | DBMS | Painting |
| 100 | OS | Painting |
| 100 | DBMS | Singing |
| 100 | OS | Acting |
| 101 | JAVA | Reading |

Suppose a query is,
List the students areaofinterest whose hobby is acting.
**Select * from student where hobbies= 'acting';**
It return the row 100 OS Acting
It does not return the areaofinterest as DBMS. So, include the row that implies the student 100 has DBMS subject with acting, so in order to keep the data consistent.
Add one more row (100, DBMS, Acting). This is an insertion anomaly.

| StudentID | AreasofInterest | Hobbies |
|---|---|---|
| 100 | OS | Singing |
| 100 | DBMS | Painting |
| 100 | OS | Painting |
| 100 | DBMS | Singing |
| 100 | OS | Acting |
| 100 | DBMS | Acting |
| 101 | JAVA | Reading |

From the table, the multivalued FDs are F: {(StudentID→→ AreasofInterest), (StudentID→→ Hobbies)}. So the Student table is decomposed as

| StudentID | AreasofInterest |
|---|---|
| 100 | OS |
| 100 | DBMS |
| 101 | JAVA |

| StudentID | Hobbies |
|---|---|
| 100 | Singing |
| 100 | Painting |
| 100 | Acting |
| 101 | Reading |

5NF:A table is in the 5NF if it is in 4NF and if for all Join dependency (JD) of (R1, R2, R3, ...,Rm) in R, every Ri is a superkey for R. (or)A table is in the 5NF if it is in 4NF and if it cannot have a lossless decomposition in to any number of smaller tables (relations).It is also known as Project-Join Normal Form (PJNF). For example the table AgentCompanyProduct is decomposed as three tables namely P1,P2 and P3.

| **AgentCompanyProduct** | | |
|---|---|---|
| **Agent** | **Company** | **Product** |
| Suneet | ABC | Nut |
| Raj | ABC | Bolt |
| Raj | ABC | Nut |
| Suneet | CDE | Bolt |
| Suneet | ABC | Bolt |

| P1 | |
|---|---|
| **Agent** | **Company** |
| Suneet | ABC |
| Raj | ABC |
| Suneet | CDE |

| P2 | |
|---|---|
| **Agent** | **Product** |
| Suneet | Nut |
| Raj | Bolt |
| Raj | Nut |
| Suneet | Bolt |

| P | |
|---|---|
| **Company** | **Product** |
| ABC | Bolt |
| ABC | Nut |
| CDE | Bolt |

Perform natural join between the above three relations then no spurious (extra) rows are added so this decomposition is called lossless decomposition. So above three tables P1, P2 and P3 are in 5 NF.

| P1⋈P2 | | |
|---|---|---|
| Agent | Company | Product |
| Suneet | ABC | Nut |
| Suneet | ABC | Bolt |
| Suneet | CDE | Nut |
| Suneet | CDE | Bolt |
| Raj | ABC | Bolt |
| Raj | ABC | Nut |

| P3 | |
|---|---|
| Company | Product |
| ABC | Nut |
| ABC | Bolt |
| CDE | Bolt |

| P1⋈P2⋈P3 | | |
|---|---|---|
| Agent | Company | Product |
| Suneet | ABC | Nut |
| Suneet | ABC | Bolt |
| Suneet | CDE | Bolt |
| Raj | ABC | Nut |
| Raj | ABC | Bolt |

6NF: It is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints. So, it is also called as Domain Key Normal Form (DKNF).

**Example: Consider** relations CAR (MAKE, vin#) and MANUFACTURE (vin#, country), Where vin# represents the vehicle identification number, 'country' represents the name of the country where it is manufactured. A general constraint may be of the following form:

If the MAKE is either 'HONDA' or 'MARUTI' then the first character of the vin# is 'B' if the country of manufacture is 'INDIA'. If the MAKE is 'FORD' or 'ACCURA', the first character of the vin# is 'A' if the country of manufacture is 'USA'.

1. Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values. F = {CH -> G, A -> BC, B -> CFH, E -> A, F -> EG} is a set of functional dependencies (FDs) so that $F^+$ is exactly the set of FDs that hold for R. How many candidate keys does the relation R have? (GATE 2013)
A.3 **B.4** C.5 D.6

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

3.Consider a relation scheme R = (A, B, C, D, E, H) on which the following functional dependencies hold: {A–>B, BC–>D, E–>C, D–>A}. What are the candidate keys of R? (GATE 2005)
A.AE, BE B.AE, BE, DE C.AEH, BEH, BCH **D.AEH, BEH, DEH**

4.From the following instance of a relation scheme R (A, B, C), we can conclude that :(GATE 2002)

| A | B | C |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 0 |
| 2 | 3 | 2 |
| 2 | 3 | 2 |

A.A functionally determines B and B function-ally determines C

**B.A functionally determines B and B does not functionally determine C**

**C.B does not functionally determine C**

D.A does not functionally determine B and B does not functionally determine C

5.Given the following relation instance. (GATE 2000)

| X | Y | Z |
|---|---|---|
| 1 | 4 | 2 |
| 1 | 5 | 3 |
| 1 | 6 | 3 |
| 3 | 2 | 2 |

Which of the following functional dependencies are satisfied by the instance? (GATE CS 2000)

A.XY -> Z and Z -> Y

**B.YZ -> X and Y -> Z**

C.YZ -> X and X -> Z

D.XZ -> Y and Y -> X

6. Which of the following is the trivial functional dependency in F+ is closure of F? (GATE 2015)

A.{P,R}→{S,T}

B.{P,R}→{R,T}

**C.{P,S}→{S}**

D.{P,S,U}→{Q}

7. Let R = ( A, B, C, D, E, F ) be a relation scheme with the following dependencies: {C→F, E→A, EC→D, A→B}. Which of the following is a key of R?(GATE 99)

A.CD **B.EC** C.AE D.AC

8. Which of the following FD can't be implied from FD set: {A->B, A->BC, C->D} ?(GATE CS 2018)

(A) A->C **(B) B->D** (C) BC->D (D) All

9.Let x, y, z, a, b, c be the attributes of an entity set E. If {x}, {x,y}, {a,b}, {a,b,c}, {x,y,z} are superkeys then which of the following are the candidate keys? (ISRO CS 2014)

A.{x,y} and {a,b} **B.{x} and {a,b}**

C.{x,y,z} and {a,b,c} D.{z} and {c}

10.Let R = ABCDE is a relational scheme with functional dependency set F = {A → B, B → C, AC → D}. The attribute closures of A and E are (UGC NET 2014)

A.ABCD, Φ **B.ABCD, E** C.Φ, Φ

D.ABC, E

11.Find the functional dependencies that stand valid on the part of the relation shown below:

| p | q | r |
|---|---|---|
| a | 1 | x |
| a | 2 | x |
| b | 3 | x |
| b | 4 | x |

a) $p \rightarrow q, p \rightarrow r$

b) $q \rightarrow p, p \rightarrow r$

c) $q \rightarrow p, r \rightarrow p, p \rightarrow r$

d) $p \rightarrow q, p \rightarrow r, r \rightarrow p$

12. Relation R=(a, b, c, d, e, g) having the functional dependencies F=(a→b, bg→e, c→d, d→g). Find the candidate key.

a) ag b) abc **c) ac** d) abd

13. In the following relation R suppose the following functional dependency holds: F={m→n, np→g, o→p, q→p, n→q}. The closure of $(n)^+$ is:

a) (n, m, q, p, g) **b) (n, q, p, g)** c) (n, o, p, q)

14. The following functional dependencies are given :{AB → CD, AF → D, DE → F, C → G, F → E, G → A} Which one of the following options is false? a) $CF^+ = \{ACDEFG\}$

b) $BG^+ = \{ABCDG\}$ c) $AF^+ = \{ACDEFG\}$

**d) $AB^+ = \{ABCDFG\}$**

GATE, NPTEL, Foreign University Problems ①

**1. Check the functional dependencies of a relation are satisfied or not?**

| A | B | C | D |
|---|---|---|---|
| 1 | 1 | 2 | 3 |
| 1 | 2 | 2 | 3 |
| 1 | 3 | 2 | 3 |
| 2 | 4 | 5 | 6 |
| 5 | 6 | 7 | 8 |

Consider a relation R (A, B, C, D) with the following instance;
Which of the following functional dependencies are satisfied by this relation? How?
(a) A → B (b) **A → CD** (c) **AB → CD**
**(d) C → D** (e) **B → A** (f) **BD → AC**
(g) AD → BC (h) D → B (i) **D → C** (j) **C → A**

Functional dependency – For a given Left Hand Side (LHS) value of an attribute (or set of attributes) of a functional dependency, there should be **at most** one Right Hand Side (RHS) value of an attribute (or set of attributes). Then we would say that the functional dependency holds on that relation.

2. Identify the functional dependencies of a relation that are satisfied and not satisfied? Also, identify the key.

**Student (Regno, Name, DOB, Phone, Gender, Course_ID, Course_Name, Instructor_ID, Instructor_Name, Instructor_Office)**

| Regno | Name | DOB | Phone | Gender | C ID | C Name | Ins_ L | Ins_ Name | Ins_ Office |
|---|---|---|---|---|---|---|---|---|---|
| **14M01** | **Kumar** | **12.1.96** | **12345** | **M** | **C1** | **DBMS** | **I1** | **Kesav** | **G123** |
| **14M05** | **Mary** | **10.6.95** | **12367** | **F** | **C1** | **DBMS** | **I1** | **Kesav** | **G123** |
| **14M07** | **Ram** | **10.5.96** | **12898** | **M** | **C1** | **DBMS** | **I2** | **Ragav** | **G127** |
| **14M01** | **Kumar** | **12.1.96** | **12345** | **M** | **C3** | **DS** | **I5** | **Mani** | **G125** |
| **14B01** | **Revathi** | **10.12.95** | **23456** | **F** | **C3** | **DS** | **I5** | **Mani** | **G125** |
| **14M09** | **Steve** | **23.10.95** | **34567** | **M** | **C4** | **OS** | **I5** | **Mani** | **G125** |
| **14B04** | **Sita** | **20.7.96** | **23454** | **F** | **C4** | **OS** | **I5** | **Mani** | **G125** |
| **14B03** | **Ramya** | **20.7.96** | **23456** | **F** | **C4** | **CS** | **I5** | **Mani** | **G125** |

***The functional dependencies* that are satisfied *on Student:***

(a) Regno → Name (b)Regno → DOB (c) Regno → Phone
(d) Regno → Gender (e) CID → CName

(f) Regno → Name, DOB, Phone, Gender (g)CName → CID
(h)Ins_ID → Ins_Name, Ins_Office

(i) Regno, CID → Name, DOB, Phone, Gender, CName

(j) Regno, CID, Ins_ID → Name, DOB, Phone, Gender, CName, Ins_Name, Ins_Office

**The functional dependencies that are not satisfied on Student:**

Phone → Regno [As per the given instance of Student, for a given phone value 23456, there are two students with register numbers 14B01, and 14B03. Hence, this FD does not satisfy].

CID → Ins_ID [CID C1 is taught by two instructors I1 and I2. Hence, this FD does not satisfy].

DOB → Regno, Name, Phone, Gender. [This FD does not hold. Two students have same DOB. Hence, this FD does not satisfy].

**3. Write down your own trivial and non trivial FDs for the following problem. From that FDs, identify the functional dependencies of a relation which are satisfied or not satisfied? Find the key for Customer.**

The following relation schema is used to store information about the customers of a car service centre.

**Customer (Customer_ID, CName, CAddr, Phone, Customer_Visit_Date, Service_Advisor_ID, SAName, SAPhone, No_of_Cars_Serviced, Service_Date)**

A customer is identified with a unique Customer_ID, he can have only one phone number, and he may have visited the service center many times. There are more service advisors and each advisor services many cars on a given service date Any service advisors can be reached at only one phone number.

From the given information, we can derive the following set of **non trivial functional dependencies**;

**Customer_ID → CName, CAddr ,Phone** [as customer id is unique, every id is related to exactly one name, one address, and one phone]

**Service_Advisor_ID → SAName, SAPhone** [Service advisors name and phone can be uniquely determined by service advisor id]

**Service_Advisor_ID,Service_Date → No_of_Car_Servicesd** [if we know the service date and the advisor id, we can uniquely determine the number of cars serviced on a data by an advisor]

**The FDs that do not hold**

**Customer_ID → Customer_Visit_Date** – [This FD does not hold because a customer can visit many times. Hence, customer id cannot uniquely determine customer visit date].

**Customer_ID → Service_Advisor_ID** – [This FD does not hold because a might have been attended by different advisors during each of his visit].

**Service_Advisor_ID → Service_Date,No_of_Cars_Serviced** – [This FD does not hold because a service advisor services every day with number of cars. Hence, we cannot uniquely determine the number of cars serviced or the service dates].

**Service_Date → No_of_Cars_Serviced** – [This FD does not hold because on a given service date other service advisors may have serviced many cars as well].

4. Find all keys for R. Consider a relation R with attributes ABCDEFGH and functional dependencies S as follows:

S = {A → CD, ACF → G, AD → BEF, BCG → D, CF → AH, CH → G, D → B, H → DEG} ④

. We may start checking all the left hand side attributes of any/all of the given set of functional dependencies. If we find the closure of an attribute and that attribute is the candidate key then any superset cannot be the candidate key. For example, if A is a candidate key, then AB is not a candidate key but a super key.

| LHS | Result | Decision |
|---|---|---|
| $A^+$ | ACDBEFGH | Result includes all the attributes of relation R. Hence, **A is one candidate key.** |
| $ACF^+$ | ACDBEFGH | ACF is a super key but not candidate key. |
| $AD^+$ | ACDBEFGH | AD is a super key but not candidate key. |
| $BCG^+$ | =BCGD from BCG → D | Result does not include all the attributes of relation R. Hence, (BCG) cannot be a candidate key. |
| $CF^+$ | ACDBEFGH | Result includes all the attributes of relation R. Hence, **(CF) is one candidate key.** |
| $D^+$ | = DB from D → B | Result does not include all R. Hence, D cannot be a key. |
| $H^+$ | = HDEG from H → DEG<br>= HDEGB from D → B | Result does not include all R. Hence, H cannot be a key. |

From the above table,

**A, ACF, AD, CF are Super keys.**

**A and CF are the candidate keys.**[ Any superset cannot be the candidate key]

**A** is the Primary key **[minimal attribute of candidate key]**

**5.Identify the anomalies that are present in the given table**

**Treatment(*docID, doctorName,docaddr,patientID, diagnosis)***

| ***docID*** | ***doctorName*** | address | ***patientID*** | ***diagnosis*** |
|---|---|---|---|---|
| D001 | Mohan | Erode | PAT123 | Fever |
| D004 | Raja | Salem | PAT121 | heart |
| D002 | Vijay | Madurai | PAT110 | Allergy |
| D001 | Mohan | Erode | PAT114 | Fever |
| D003 | Jenifer | Chennai | PAT112 | Eye |
| D002 | Vijay | Madurai | PAT121 | Cold |

The functional dependencies are **(doctorID → doctorName)**, **((doctorID,patientID) → diagnosis)**, (doctorname → address)

The combination **(doctorID, patientID)** is the **primary key** for TREATMENT.

**Anomalies:**

***Insertion anomaly: The inability to store certain attributes' values without the other attributes. If we are not able to insert a record into a relational table because of the absence of values for other attributes is called insertion anomaly.***

For Treatment, we cannot insert the doctor information like doctorID and doctorName without any patient. That is, we need at least one patient to include the doctor information into the table. For example, if one more doctor Dr.Neeraj is appointed, we need to allocate at least one patient to insert Dr.Neeraj's information. This inability is *insertion anomaly*.

***Deletion anomaly: Deletion of values of certain attributes from any records (rows) will lead to lose of other attributes' values of the same records. That means, we lose the complete information about an entity if we delete few values. This is called as deletion anomaly.***

Deleting patients' diagnosis could delete the name of their doctor. For example, Dr.Jenifer has only one patient PAT112 registered for him. If we delete the patient PAT112, we need to delete Dr.Jenifer's details as well. This is the *deletion anomaly* present in the TREATMENT table.

***Modification anomaly: Modification of certain values in a table may lead to change the values in more than one record if that particular value has duplicates. If we miss any one occurrence of that data, that leads to inconsistency of the table. This is called as modification anomaly.***

A doctor may have more than one patient, so an update anomaly may result if a doctor's address is changed for a given doctorID for only one patient. For example, in our table TREATMENT Dr.Vijay has two patients. Suppose that we need to change the doctor address from Madurai to Kovai. This change has to be done on two records. What if we change with Third record and not changing with Sixth record? If we do such thing, we are not able to find the exact address of the doctor of doctor ID D002. This stage is called as modification anomaly.

**6. Find all candidate keys. Suppose that you are given a relation R = (A,B,C,D,E) with the following functional dependencies: {CE → D,D → B,C → A}.**

CE → ABCDE. Hence, **CE is the only candidate key** for the given relation R.

**1. Decompose the following table upto 3NF. The table User_Personal as given below;** (7)

| *UserID* | U_email | Fname | Lname | City | State | Zip |
|---|---|---|---|---|---|---|
| *MA12* | Mani@ymail.com | Manish | Jain | Bilaspur | Chatisgarh | 45899 |
| *PO45* | Pooja.g@gmail.com | Pooja | Magg | Kacch | Gujrat | 83221 |
| *LA33* | Lavle98@jj.com | Lavleen | Dhalla | Raipur | Chatisgarh | 85357 |
| *CH99* | Cheki9j@ih.com | Chimal | Bedi | Trichy | Tamil nadu | 63201 |
| *DA74* | Danu58@g.com | Dany | James | Trichy | Tamil nadu | 64501 |

The given FDs are F = {{ UserID → U_email, Fname, Lname,City,State,Zip},{Zip → City, State}}.

. **Is this table in First Normal Form?**

**Yes.** All the attributes contain only atomic values.

. **Is this table in Second Normal Form?**

To verify this Normal form, first to identify a key. Use Attribute closure to find key.

- As UserID attribute can uniquely determine all the other attributes So, UserID as the Primary key for User_Personal table.

The next step is to check for the 2NF properties;

***Property 1 – The table should be in 1NF. Property 2 – There should not be any partial key dependencies.***

- **Our table is in 1NF, hence property 1 is holding.**
- Primary key of the table is UserID and UserID is single simple attribute. **As the key is not composite, there is no chance for partial key dependency to hold.** Hence property 2 is also holding.

User_Personal table is in 2NF.

**. Is User_Personal in 3NF?**

To verify this we need to check the 3NF properties;

***Property 1 – Table should be in 2NF.***

***Property 2 – There should not be any Transitive Dependencies in the table.***

**Table User_Personal is in 2NF, hence property 1 is satisfied.**

User_Personal table holds the following Transitive dependency;

UserID → Zip, Zip → City ,State

**Hence, property 2 is not satisfied and the table is not in 3NF.**

Decompose User_Personal. For this, we can use the functional dependencies Zip → City,State and UserID→ U_email, Fname, Lname ,City, State ,Zip.

- As a result, the following tables (primary keys are underlined) are formed.

User_Personal (UserID, U_email, Fname, Lname, Zip)

City (Zip, City, State) -

**Table - User_Personal**

| *UserID* | U_email | Fname | Lname | Zip |
|---|---|---|---|---|
| *MA12* | Mani@ymail.com | MANISH | JAIN | 458991 |
| *PO45* | Pooja.g@gmail.co | POOJA | MAGG | 832212 |
| *LA33* | Lavle98@jj.com | LAVLEEN | DHALLA | 853578 |
| *CH99* | Cheki9j@ih.com | CHIMAL | BEDI | 632011 |
| *DA74* | Danu58@g.com | DANY | JAMES | 645018 |

**Table – City**

| Zip | City | State |
|---|---|---|
| 458991 | BILASPUR | CHATISGARH |
| 832212 | KACCH | GUJRAT |
| 853578 | RAIPUR | CHATISGARH |
| 632011 | TRICHY | TAMIL NADU |
| 645018 | TRICHY | TAMIL NADU |

**2. Suppose I have a table that contains a single primary key field (it doesn't have a concatenated primary key). Is it possible for this table to contain a 2NF violation?**

No, it is not possible to have 2NF violation, if a table that contains a single primary key field.

**3.A BCNF violation contains a stray dependency. Does the arrow which shows it: A) Come from a primary key field or from a non-primary key field? B) Go into a primary key field or a non-primary key field?**

**A) Come from a non-primary key field** B)

**Go into a primary key field**

4. Consider the following dependencies and the BOOK table in a relational database design. Determine the normal form of the given relation.F:{ ISBN → Title, ISBN → Publisher, Publisher → Address } (10)

A.First Normal Form **B.Second Normal Form**
C.Third Normal Form D.BCNF

5. The best normal form of relation scheme R(A, B, C, D) along with the set of functional dependencies
F = {AB → C, AB → D, C → A, D → B} is

A.Boyce-Codd Normal form **B.Third Normal form**
C.Second Normal form D.First Normal form

6. For a database relation R(a,b,c,d) where the domains of a, b, c and d include only atomic values, only the following functional dependencies and those that can be inferred from them hold F:{a → c, b →d}.The relation is in

**A.First normal form but not in second normal form**
B.Second normal form but not in third normal form
C.Third normal form D.None of the above

7. Consider the relation R (ABCDE): FD = { A → B, B → C, C → D, D → E} Find out the highest normal form.

A.1 NF **B.2 NF** C.3 NF D.BCNF

8. Relations produced from E - R Model will always be in

A.1 NF B.2 NF **C.3 NF** D.4 NF

9. Identify the correct statement about the decompositions demonstrated below: (where p, q, r are column names)

i)

| p | q | r |
|---|---|---|
| a | 1 | x |
| b | 2 | y |

→

| p | q |
|---|---|
| a | 1 |
| b | 2 |

| q | r |
|---|---|
| 1 | x |
| 2 | y |

ii)

| p | q | r |
|---|---|---|
| a | 1 | x |
| b | 1 | y |

→

| p | q |
|---|---|
| a | 1 |
| b | 1 |

| q | r |
|---|---|
| 1 | x |
| 1 | y |

**a) i) is lossless, ii) is lossy**
b) ii) is lossless, i) is lossy
c) i) and ii) both are lossless
d) i) and ii) both are lossy

10. Which highest normal form is guaranteed by the following relation?

| dept_ID | dept_Name | salary |
|---|---|---|
| D1 | Sales | 50000 |
| D2 | Software | 40000 |
| D3 | Finance | 37000 |
| D4 | Marketing | 45000 |

a) 1NF **b) 2NF** c) 3NF
d) BCNF

11. R(a, b, c) have F=(ab→c, c→b). The table is not in which normal form:
a) 1NF b) 2NF c) 3NF **d) BCNF**

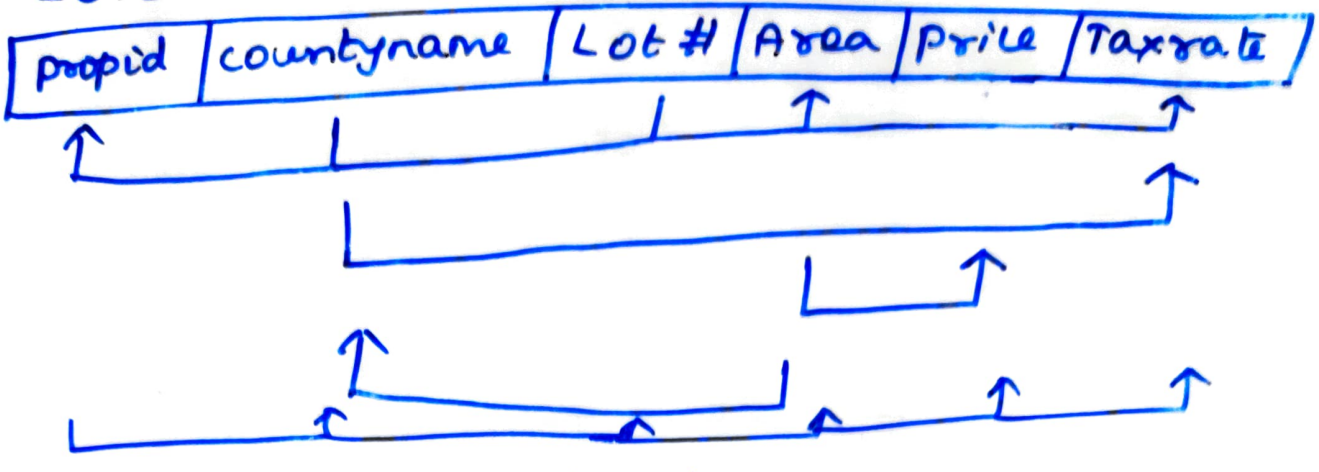
12. The following relation guarantees which highest normal form?

| sid | sname | course | teacher |
|---|---|---|---|
| S1 | RAM | C | AR |
| S2 | MADHU | DBMS | PPD |
| S1 | RAM | DBMS | PB |
| S2 | MADHU | C | SM |

**a) 1NF** b) 2NF c) 3NF
d) BCNF

1) Split the tables using normal forms upto BCNF

LOTS




Step1: find the candidate key

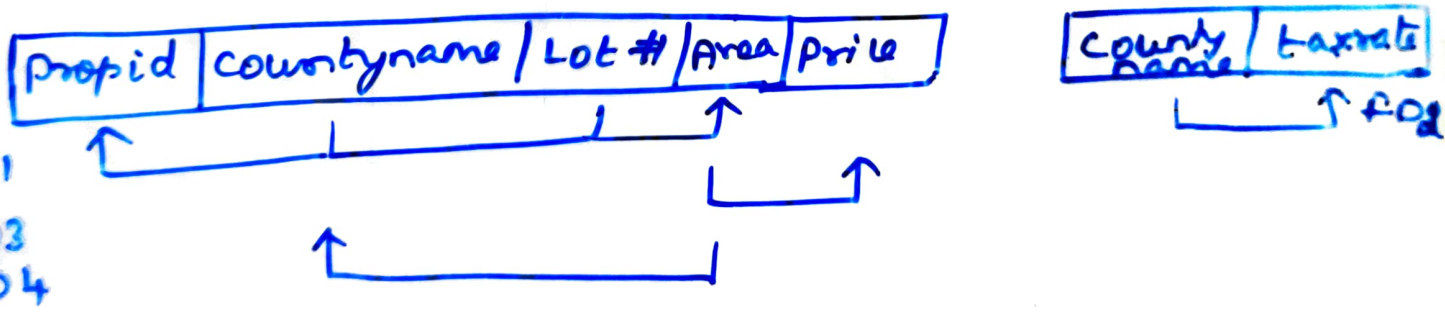
i) {countyname, Lot #}$^+$ = {countyname, Lot #, propid, Area, price, taxrate}

ii) {propid}$^+$

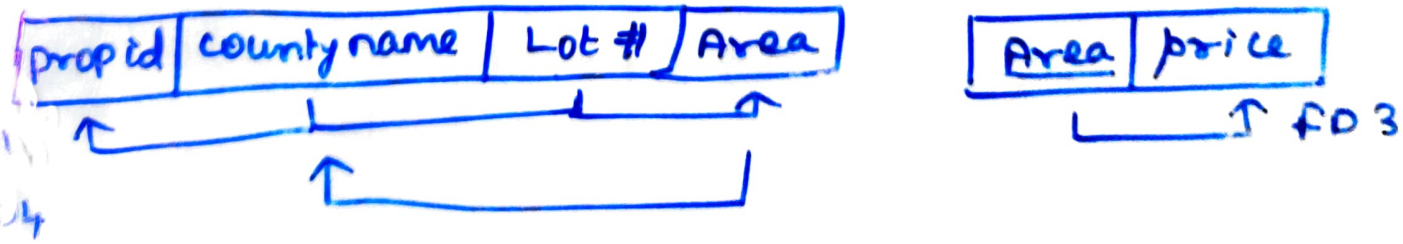
Step2: check the given table is in 1NF

Step3: check the partial FD after 1NF and eliminate partial FD.

countyname ⟶ taxrate




Step4: check for Transitive FD after 2NF




Step5: