Database Design Theory

Functional Dependencies :- A functional dependency, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$

A functional Dependency is a property of the semantics or meaning of the attributes.

Normalization of Relations :- Normalization of data can be considered a process of analyzing the given relation schema based on their FDs and primary keys to achieve the desirable properties of minimizing redundancy and minimizing the insertion deletion and update anomalies.

The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

First Normal Form :- 1NF was defined to disallow multivalued attributes, composite attributes and their combinations. 1NF disallows having a set of values, a tuple of values or a combination of both as an attribute value for a single tuple

Eg:- Department

Dname	Dno	Dmgr_SSN	DLocations

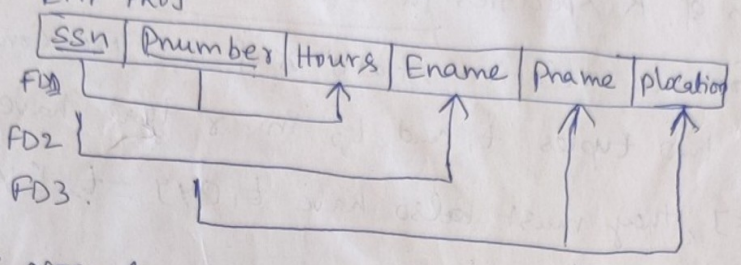
The Dept relation is not in 1NF. Remove the attribute DLocations that violates 1NF and place it in a seperate relation.

Second Normal Form :- A relation schema R is in 2NF if every non prime attribute A in R is fully functionally dependent on the primary key of R.

2NF is based on the concept of full functional dependency.

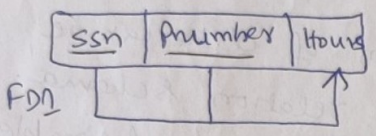
A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute A from X means that the depend does not hold any more.

Eg:- EMP-PROJ

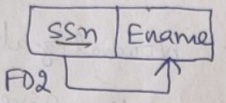


2NF Normalization

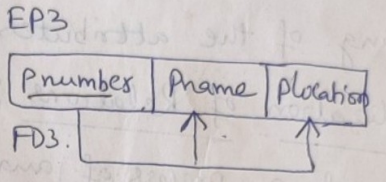
EP1



EP2



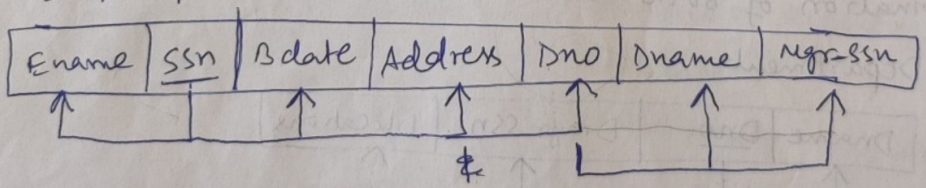
EP3



Third Normal Form :- 3NF is based on the concept of transitive dependency. A functional dependency $X \rightarrow Y$ in a relation schema R is a transitive dependency if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

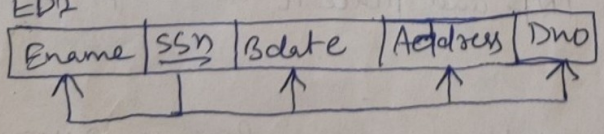
A Relation schema R is in 3NF if it satisfies 2NF and no non prime attribute of R is transitively dependent on the primary key.

Eg:- EMP-DEPT

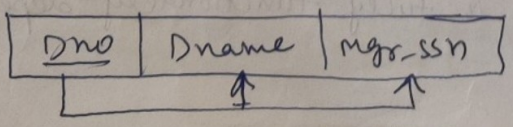


3NF Normalization

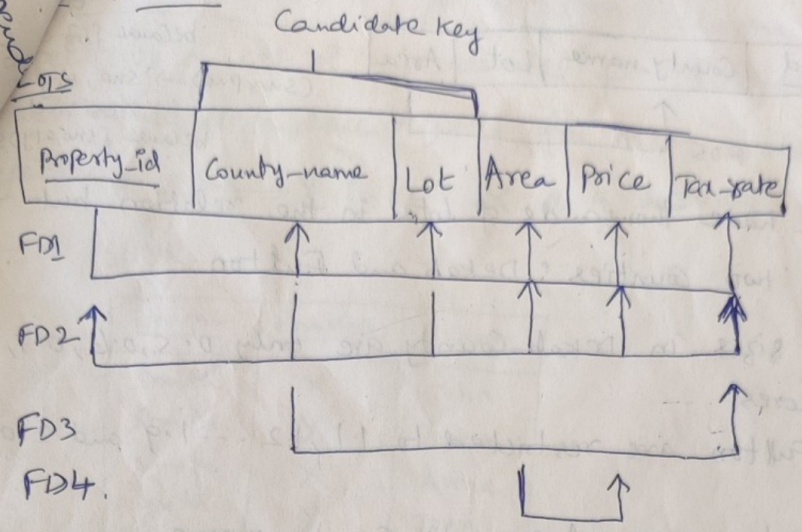
ED1



ED2



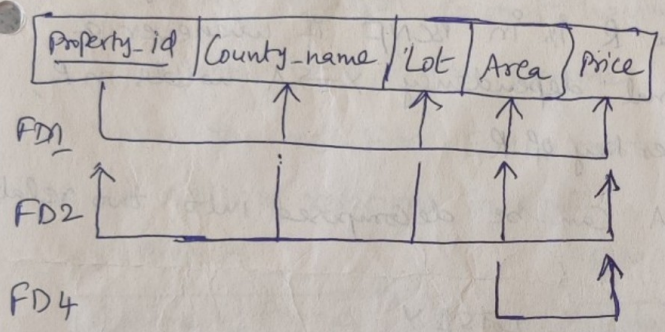
Forms :-



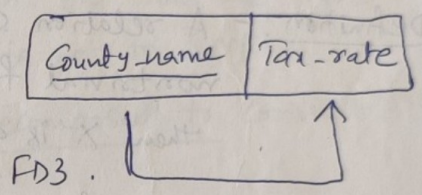
LOTS violates 2NF due to FD3.

Decompose LOTS to LOTS1 and LOTS2

LOTS1



LOTS2

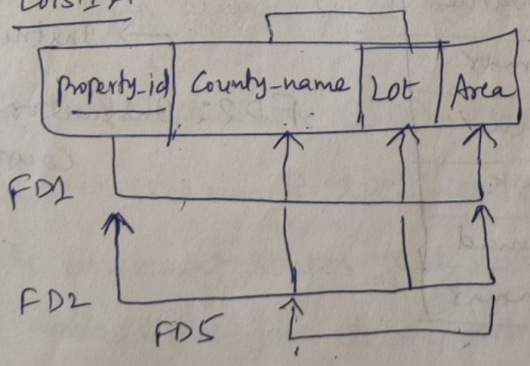


Both LOTS1 & LOT2 ~~are~~ both are in 2NF

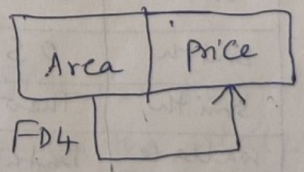
LOTS 2 is in 3NF but LOTS1 is not in 3NF.

Decompose LOTS1 to LOTS1A & LOTS1B.

LOTS1A.



LOTS1B



LOTS1A is in 3NF but not in BCNF. if we add another FD to it.

Every relation in BCNF is also in 3NF but not the reverse is true.

BCNF :- LOTS1A.

Property_id	County_name	Lot	Area

FD5

Suppose that we have thousands of lots in the relation but the lots are from only two counties : Dekab and Fulton.

Suppose that lot sizes in Dekab County are only 0.5, 0.6, 0.7, 0.8 ... 0.9 and 1.0 acres.

Lot sizes in Fulton are restricted to 1.1, 1.2 ... 1.9 and 2.0 acres.

Now the FD becomes Area → County_name.

So, LOTS1A is in 3NF but not in BCNF. FD5 satisfies 3NF because County_name is a prime attribute, but Area is not a super key of LOTS1A.

Definition :- A relation schema R is in BCNF if whenever a nontrivial functional dependency X → A holds in R, then X is a super key of R.

The above relation LOTS1A can be decomposed into two relations as follows

LOTS1AX

Property_id	Area	Lot

LOTS1Y

Area	County_name

Eg :-

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	OS	Ammar
Smith	Theory	John
Wallace	Database	Mark
Wallace	OS	Ahmad
Narayan	OS	Ammar

FD1 : {Student, Course} → Instructor

FD2 : Instructor → Course

The above relation TEACH is in 3NF but not in BCNF.

To make the relation to be in BCNF, we decompose the relation into two relations.

{Instructor, Course} and {Instructor, Student}

other alternatives are {Student, Instructor} and {Student, Course}
{Course, Instructor} and {Course, Student} → but they give spurious tu

Normal Form: – A relation schema R is in 4NF with respect to a set of dependencies F (functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency $X \rightarrow\rightarrow Y$ in F^+. X is a superkey for R.

Eg:– EMP.

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

An all key relation is always in BCNF since it has no FDs.

The EMP relation is not having FDs but has the MVD

Ename $\rightarrow\rightarrow$ Pname | Dname and is not in 4NF.

Now the relation EMP must be decomposed to convert it into a set of relations in 4NF.

EMP-Projects

Ename	Pname
Smith	X
Smith	Y

EMP-Dependents

Ename	Dname
Smith	John
Smith	Anna

Fifth Normal form: – This normal form identifies a peculiar constraint that causes a relation to be decomposed into several components so that they always yield the original relation back after a join.

A join dependency (JD) denoted by $JD(R_1, R_2 \ldots R_n)$, specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into $R_1, R_2 \ldots R_n$. Hence, for every such r we have $*(\pi_{R_1}(r), \pi_{R_2}(r) \ldots \pi_{R_n}(r)) = r$.

A join dependency $JD(R_1, R_2 \ldots R_n)$ specified on relation schema R is a trivial JD if one of the relation schemas R_i in $JD(R_1, R_2 \ldots R_n)$ is equal to R.

Fifth normal form :- A relation schema R is in 5NF (Projection-Join normal form) with respect to a set F of functional, multivalued and join dependencies if, for every non trivial join dependency JD $(R_1, R_2 \ldots R_n)$ in F^+, every R_i is a superkey of R.

Eg :-

SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	Proj X
Smith	Nut	Proj Y
Adamsky	Bolt	Proj Y
Walton	Nut	Proj Z
Adamsky	Nail	Proj X
Adamsky	Bolt	Proj X
Smith	Bolt	Proj Y

R1.

Sname	partname

R2

Sname	Proj_name

R3

Partname	Proj_name

BCNF example :-

STU-ID	STAFF-ID	CLASS-CODE	ENROLL-GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

Each class-code identifies a class uniquely

A student can take many classes

A staff member can teach many classes but each class is taught by only one staff member

{stu_id, staff_id} → class_code, enroll_grade

class_code → staff_id

stu_id	staff_id	class_code	enroll_grade

— Relation is in 3NF but not BCNF.

Decompose the relation into 2 relations

stu_id	class_code	enroll_grade

class_code	staff_id

The rules for FDs:-

The set of all dependencies that include F as well as all dependencies that can be infered from F is called the closure of F; it is denoted by F^+

For the relation Emp-Dept (Ename, SSN, Bdate, Address, Dnumber, Dname, mgr-SSN)

$F = \{SSN \rightarrow \{Ename, Bdate, Address, Dnumber\}, Dnumber \rightarrow \{Dname, mgr_SSN\}\}$

Some of the additional FDs that we can infer from F are the following:

$SSN \rightarrow \{Dname, mgr_SSN\}$

$SSN \rightarrow SSN$

$Dnumber \rightarrow Dname$

An FD $X \rightarrow Y$ infered from a set of dependencies F specified on R if $X \rightarrow Y$ holds in every legal relation state r of R.

The closure F^+ of F is the set of all functional dependencies that can be infered from F.

IR1 : If $X \supseteq Y$, then $X \rightarrow Y$ – reflexive

IR2 : $\{X \rightarrow Y\} \models XZ \rightarrow YZ$ – augmentation rule

IR3 : $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$ – transitive rule

IR4 : $\{X \rightarrow YZ\} \models X \rightarrow Y$ – decomposition or projective rule

IR5 : $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$ – union or additive rule

IR6 : $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$ – Pseudotransitive rule

A FD $X \rightarrow Y$ is trivial if $X \supseteq Y$; otherwise, it is nontrivial.

The inference rules IR1 through IR3 are sound and complete.

For each such set of attributes X, we determine the set X^+ of attributes that are functionally determined by X based on F; X^+ is called the closure of X under F.

FDs on EMP_PROJ are $F = \{ ssn \rightarrow Ename,$
$Pnumber \rightarrow \{Pname, Plocation\}$
$\{ssn, pnumber\} \rightarrow Hours \}$

The closure sets are

$$\{ssn\}^+ = \{ssn, Ename\}$$

$$\{pnumber\}^+ = \{pnumber, Pname, Plocation\}$$

$$\{ssn, Pnumber\}^+ = \{ssn, pnumber, Ename, pname, Plocation, Hours\}$$

<u>Equivalence of sets of FDs</u>:—

A set of FDs F is said to cover another set of FDs E if every FD in E is also in F^+; that is, if every dependency in E can be inferred from F; alternatively, we can say that E <u>is covered by F</u>.

Two sets of FDs E and F are equivalent if $E^+ = F^+$. Therefore, equivalence means that every FD in E can be inferred from F, and every FD in F can be inferred from E; that is, E is equivalent to F if both the conditions - E covers F and F covers E - hold.

<u>Minimal sets of FDs</u>:— A minimal cover of a set of FDs E is a set of functional dependencies F that satisfies the property that every dependency in E is in the closure F^+ of F.

A set of FDs F to be <u>minimal</u> if it satisfies the following conditions:

1. Every dependency in F has a single attribute for its right-hand side.
2. we cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y is a proper subset of X, and still have a set of dependencies that is equivalent to F.
3. we cannot remove any dependency from F and still have a set of dependencies that is equivalent to F.

set of dependencies are being a set of dependencies in a standard or Cannonical form and with no redundancies.

$E: \{B \to A, D \to A, AB \to D\}$
$BB \to AB$, $AB \to D$.
$B \to D$.
$\{B \to A, D \to A, B \to D\}$
$E: \{B \to A, D \to A\}$

Relation Decomposition :—

$R = \{A_1, A_2 \ldots A_n\}$.

$D = \{R_1, R_2 \ldots R_m\}$ where D is the decomposition of R

$$\bigcup_{i=1}^{m} R_i = R.$$

Each attribute in R will appear in at least one relation schema R_i in the decomposition so that no attributes are lost. This is called the attribute preservation condition of a decomposition.

Dependency preservation property of decomposition! —

Def: Given a set of dependencies F on R, the projection of F on R_i, denoted by $\pi_{R_i}(F)$ where R_i is a subset of R, is the set of dependencies $X \to Y$ in F^+ such that the attributes in $X \cup Y$ are all contained in R.

A decomposition $D = \{R_1, R_2 \ldots R_m\}$ of R is dependency preserving w.r.t F if the union of the projections of F on each R_i in D is equivalent to F; that is,

$$((\pi_{R_i}(F)) \cup \ldots \cup (\pi_{R_m}(F)))^+ = F^+$$

It is always possible to find a dependency-preserving decomposition D with respect to F such that each relation R_i in D is in 3NF.

Nonadditive (lossless) Join property of a Decomposition! —

Def: Formally, a decomposition $D = \{R_1, R_2 \ldots R_m\}$ of R has the lossless join property w.r.t the set of dependencies F on R if for every relation state r of R that satisfies F, the following holds, where * is the NATURAL JOIN of all the relations in D: $*(\pi_{R_1}(r) \ldots \pi_{R_m}(r)) = r$

This property ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to the relations resulting from the decomposition.

The word loss in lossless refers to loss of information, not to loss of tuples.

Eg:-

$R = \{SSN, Ename, Pnumber, Pname, Plocation, Hours\}$

$R_1 = EMP_LOCS = \{Ename, Plocation\}$

$R_2 = EMP_PROJ1 = \{SSN, Pnumber, Hours, Pname, Plocation\}$

$F = \{SSN \rightarrow Ename;\ Pnumber \rightarrow \{Pname, Plocation\};\ \{SSN, Pnumber\} \rightarrow Hours\}$

	SSN	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(No changes to matrix after applying FDs)

EMP

SSN	Ename

PROJECT

Pnumber	Pname	Plocation

WORKS_ON

SSN	Pnumber	Hours

$R_1 = EMP = \{SSN, Ename\}$

$R_2 = PROJ = \{Pnumber, Pname, Plocation\}$

$R_3 = WORKS_ON = \{SSN, Pnumber, Hours\}$

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

(Original matrix S at start of algorithm)

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

(Matrix S after applying the first two FDs; last row is all 'a' symbols so we stop)