Module - IV

Database Design: Functional Dependencies - Normal forms; First Normal form, Second Normal Form, Third Normal Form, Boyce Codd Normal Form, Multivalued Dependency & Fourth Normal Form, Join Dependencies & Fifth Normal form. Inference Rules for Functional Dependency Minimal sets of functional Dependencies, Prop: of RDBMS decomposition.

Functional Dependency

A f.d is a constraint b/w two sets of attributes from the db, denoted by,

$$X \rightarrow Y$$

Y is functionally dependent on X.

Inference Rules for Functional Dependency

(1) Reflexive Rule :- If $Y \subseteq X$, then $X \rightarrow Y$

(2) Augmentation Rule :- If $X \rightarrow Y$, then $XZ \rightarrow YZ$

(3) Transitive Rule :- If $X \rightarrow Y$ & $Y \rightarrow Z$, then $X \rightarrow Z$.

(4) Decomposition or Projective Rule : If $X \rightarrow YZ$, then $X \rightarrow Y$ & $X \rightarrow Z$.

(5) Union or Additive Rule : If $X \rightarrow Y$ & $X \rightarrow Z$, then $X \rightarrow YZ$.

(6) Pseudo Transitive Rule :- If $X \rightarrow Y$ & $WY \rightarrow Z$, then $WY \rightarrow Z$.

The 1st 3 rules are

- sound :- generate only fun.d that actually hold.
- complete :- generate all f.d that hold. They are called Armstrong's axioms, the last 3 inference rules are inferred from armstrongs axioms.

Examples

$R = (A, B, C, G, H, I)$

$F = \begin{Bmatrix} A \rightarrow B \\ A \rightarrow C \end{Bmatrix}$ find $f^n.d$ F^+ of F.

$\begin{rcases} CG \rightarrow H \\ CG \rightarrow I \end{rcases}$

$B \rightarrow H$

By transitivity from $A \to B$ & $B \to H$
we get $A \to H$.

(2) $AG \to I$
by augmentation $A \to C$ with G, to get $AG \to CG$
& then transitivity with $CG \to I$

(3) $CG \to HI$
from $CG \to H$ & $CG \to I$ (union rule).

Decomposition

When we decompose a relation schema with a set of fds F into $R_1 \ldots R_n$ we want

1) **Lossless decomposition** :- otherwise decomposition would result in information loss.

$R(A,B,C)$

decompose

$R_1(A,B)$ $\quad$ $R_2(A,C)$

recover

$R'(A,B,C)$

$R(A,B,C) = R'(A,B,C) \rightarrow$ lossless decomp.

A decomposition of R into R_1 & R_2 is lossless form iff atleast one of the following condn hold is in F^+.

$R_1 \cap R_2 \rightarrow R_1$

$R_2 \cap R_2 \rightarrow R_2$

Example.

Table having fields

studid, name, marks.

decomposed into

studid, name $\quad$ studid, marks

There is a common attribute studid & orignal information can be retrieved without any information loss.

Illustration of Lossy Decomposition

Given instances of the decomposed relns, if we are not able to reconstruct the corresponding instances of the original reln – there is informatn loss i.e. lossy decomposition.

eg:- $R = (A, B)$

$R_1 = (A)$, $R_2 = (B)$

Table 1

A	B
α	1
α	2
β	1

A
α
β

B
1
2

recovered →

$\Pi_A(r) \bowtie \Pi_B(r)$

A	B
α	1
α	2
β	1
β	2

The original reln is not retrieved after applying decomposition & retrieving data, so this comes under lossy decomposition.

(2) No redundancy :- The reln R_i shld be either Boyce Codd Normal form or 3rd Normal form.

(3) Dependency Preserving :- Let F_i be the set of dependencies F^+ that include only attributes in R_i.

Preferably the decomposition shld be dependancy preserving, that is

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$

Otherwise, checking updates for violation in functional dependencies may require computing joins, which is expensive.

Let $F' = F_1 \cup F_2 \cup \ldots F_n$ ∴ F' is a set of f.ds on schema R, but in general $F' \neq F$, It may be $F'^{+} = F^{+}$. If the latter is true then every dependency in F is logically implemented in F', & if we verify that F' is satisfied, we have verified that F is satisfied.

The conclusion is decomposition having the property $F'^{+} = F^{+}$ is a dependency-preserving decomposition.

Example

$R = (A, B, C)$

$F = \{A \to B, B \to C\}$

Normalization

The basic objectives of normalization is to reduce redundancy, which means that information is to be stored only once. Storing information several times leads to wastage of storage space & ↑ ing the total size of the data stored. Relations are normalized so that when rltns in the db are to be altered, we do not loss informatn or introduce inconsistencies.

Functional Dependency

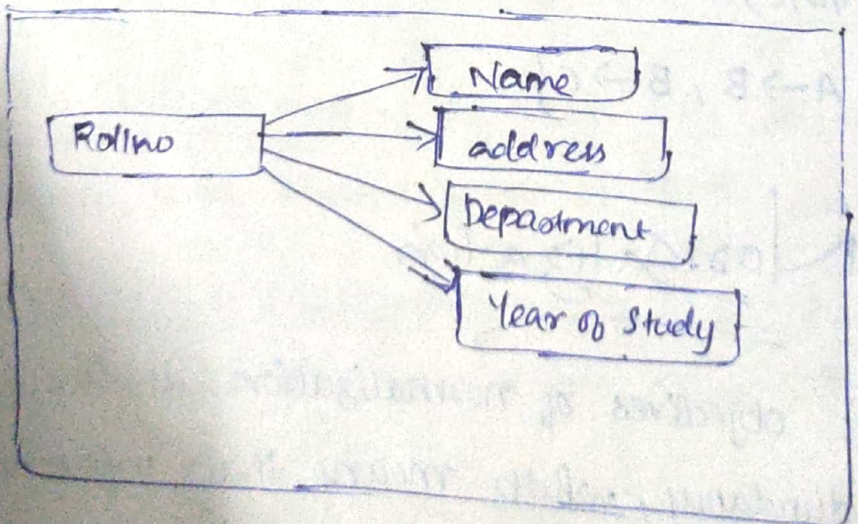
Let X & Y be two attributes of a relln. Given the value of X, if there is only one value of Y corresponding to it, then Y is said to be functionaly dependent on X. This is indicated by the notation.

$X \longrightarrow Y$

if we write,

$X, Z \longrightarrow Y$ – It means that there is only one value of Y corresponding to given values of X, Z.

Student



In this relln named student, the values of all
2 the attributes can be uniquely determined.

Thus all the attributes are f.d on the attribute Rollno.

Anomalies in a db / Pitfalls in a Relt𝑛 Db

Consider the rltn

Name	Course	Phno	Major	Prof	Grade
Jones	353	237-46843	CS	Smith	A
Smith	329	431-53876	Phy	Turner	B
Martin	456	531-64789	Chem	Clark	B
Duke	293	444-75689	Math	James	A+
Jones	379	237-46843	Histy	Lanl	B

The keys of the rltns is Name & course.
Here the attribute grade is fully f.d on the key, Name & course.

Name ← phone no

Name ← Major

Course ← Professor

Name, course ← grade

Thus, the determinants of these f.d. r not the entire key, but only part of the key of the rltn.

The anomalies forced here are.

* Update Anomalies :- A change in the ph no: of jones must be made, for consistancy in all tuples pertaining to the student Jones.

* Insertion Anomalies :- If this is the only rltn in the db showing the association b/w a faculty member & the course, the fact that a given professor is teaching a given course cannot be entered in the db unless a student is registered in the course.

* Deletion Anomalies :- If the only student registered in a given course discontinues the course the information as to which professor is offering the course will be lost, if this is the only rltn in the db showing the association b/w a faculty member & the course.

- It often leads to generation of ~~set~~ spurious ~~tables~~. tuples

⇒ First Normal Form

① The domain of the attributes must include only atomic values

② It was designed to disallow multivalued, composite attributes & their combination.

③ It disallows nested rltns.

Converting a rltn to 1NF is the 1st essential step of normalization. Each form is an improvement over the earlier form - 2NF is a improvement on 1NF, 3NF is an improvement on 2NF & so on.

Student Rltn

Rollno	Name	Depatmnt	year	phnno
1	Anju	MCA	1	234567
2	Mary	MBA	2	467892
3	clarie	MCA	1	489435
4	jist	MBA	2	498763
5	Latha	Btch	3	235865

The above rltn is in 1NF, as it does not contain any non atomic value, or composite attribute or multivalued attribute.

Second Normal form

① A rltn is said to be in 2NF, if it is already 1NF & non-key attributes are fully functionally dependent on the 1° key attribute.

② If the key has more than one attribute then no non-key attributes shld be f.d upon part of the key attribute.

Student Rltn

Studid	Name	Course Id	Coursename	Grade
101	Johsn	MS1250	Btech	A+
102	Johsn	MS1415	MBA	A
103	Thennen	MS1331	Hot Mgt	B+
104	Chris	MS1455	MCA	B

pk

Primary key – studid, courseid

studid ⟵ Name ⟶ Partial dependencies

courseid ⟵ coursename ⟶ "

studid, courseid ⟵ Grade ⟶ full "

Create or replace ... sqrt. no / no number(3)

To convert this relation to 2NF.

Student

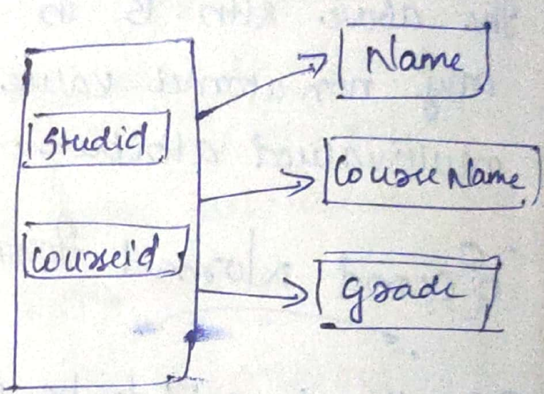
Studid	Name
101	Johns
102	Lennon
103	Chris

Course

CourseId	Coursename
MS1250	Btech
MS1415	MBA
MS1331	Hotl Mng
MS1455	MCA

Student-Grade

Studid	Courseid	grade
101	MS1250	A+
101 ~~102~~	MS1415	A
103 102	MS1331	B+
103 ~~104~~	MS1455	B



Third Normal Form

A table is in 3NF iff

(1) It is already in 2NF &

(2) No two non-prime attributes should be f.d on each other ie all the non-prime attributes should be non-transitively dependent on each other in the relation.

Rollno	Name	Dprtmnt	year	hostel (duplicate)
1784	Raman	Phy	1	ganga
1896	Maya	Chem	1	ganga
1487	Singh	Maths	2	kaveri
1693	Rajan	CS	2	"
1847	krishnan	CS	3	krishna

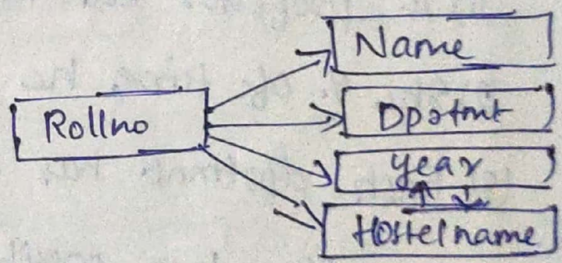
observe that given the year of student, his hostel is known & vice versa.

Here, dependency is b/w the attributes yr & hostel names, which are both non key attributes. It leads to unnecessary duplication of data.

To convert this into 3NF,

Rollno	Name	Dpstmnt	year
1784	Raman	Phy	1
1896	Maya	Chem	1
1487	Singh	Maths	2
1693	Rayan	CS	2
1847	krishnan	CS	3

year	Hostelname
1	ganga
2	kaveri
3	krishna
4	godavari



⊛ Boyce-Codd Normal Form (BCNF)

It arises when

(1) The table is already in 3NF.

(2) A rltn has more than one posible key

(3) Further that the composite keys have a common attribute.

4) If an attribute of a composite key is dependent on the attribute of the other composite key, a normalization, BCNF is required.

Example - Professor

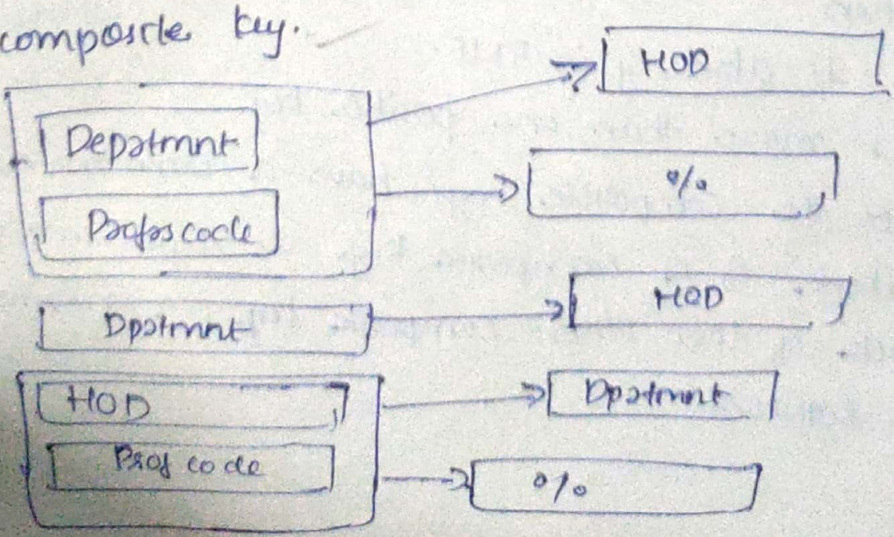
Professor	Department	HOD	Percent
P_1	Phy	Rao	50
P_1	Maths	Krishnan	30
P_2	Chem	Ghosh	75
P_2	Phy	Rao duplicated	95
P_3	Maths	Krishnan	25

The rltn is in 3NF, but the attributes dept & HOD r dupeicated.

- It is assumed that,

(1) A professor can work in more than one dept.
(2) The % of time he spends in each dept is given
(3) Each dptmnt has only one HOD.

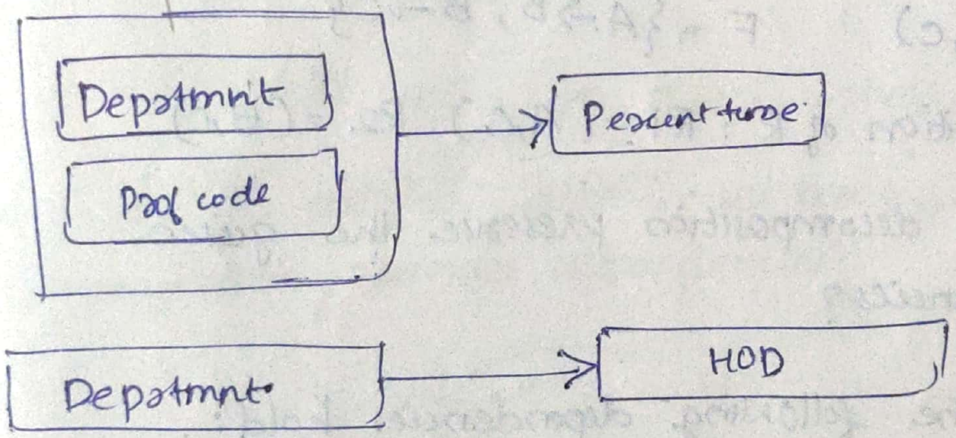
The two possible composite keys are HOD, Professor code & Profesr code & Dpt. ~~other~~
Observe that HOD & depatmnt are not non-key attributes, they are part of the composite key.



To convert to BCNF,

Profcode	Department	% time
P_1	Phy	50
P_1	Maths	30
P_2	Chem	75
P_2	Phy	75
P_3	Maths	25

Dept	HOD
Phy	Rao
Maths	Krishnan
Chem	Ghosh



Thus, a table is in BCNF iff it is already in 3NF & every determinant shld be candidate key.

1NF → Values shld be non-redundant & atomic

2NF → non-prime attributes shld be fully fundamentally dependent.

3NF → intransitively dependent

4NF → No multivalued dependency

5NF → No join dependency

6NF → every determinant shld be a candidate key.

Create or replace function sqrt.no / no number (3)

* Dependency Preservation

Mod-IV

Getting lossless decomposition is necessary. But ofcourse, we also want to keep dependencies, since losing a dependency means, that the corresponding constraint can be check only through natural join of the appropriate resultant rltn in the decomposition. This would be very expensive, so, our aim is to get a lossless dependency preserving decomposition.

Example:

$R = (A, B, C)$ $\quad F = \{A \to B,\ B \to C\}$

Decomposition of R : $R1 = (A, C)$ $\quad R2 = (B, C)$

? Does this decomposition preserve the given dependencies?

Soln:- In R1 the following dependencies hold:

$$F1' = \{A \to A,\ C \to C,\ A \to C,\ AC \to AC\}$$

In R2, " " $\quad F2' = \{B \to B,\ C \to C,\ B \to C,\ BC \to BC\}$

The set of nontrivial dependencies hold on R1 & R2:

$$F' := \{B \to C,\ A \to C\}$$

$A \to B$ can not be derived from F', so this decomposition is NOT dependency preserving.

Example

$R = (A, B, C)$

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

Decomposition of R : $R1 = (A, B)$

$R2 = (B, C)$

? Does this decomposition preserve the given dep?

Soln

In R1 the foll: dep holds :

$F1 = \{A \rightarrow B, A \rightarrow A, B \rightarrow B, AB \rightarrow AB\}$

$F2 = \{B \rightarrow B, C \rightarrow C, B \rightarrow C, BC \rightarrow BC\}$

$F' = F' \cup F2' = \{A \rightarrow B, B \rightarrow C,$ trivial dependencies$\}$

In F' all the original dependencies occur, so this decomposition preserves dependencies.

1/19

Module-IV

* Functional Dependancies

Let x & y be the two attributes of a rltn. Given the value of x, if there is only one value for y corresponding to it, then y is said to be functionally dependent on x. Indicated by notation $X \to Y$. Y is functi depend upon x.

* Inference Rules for Functional Dependencies / Amgstrong's axioms

Ar'ax r used to conclude func'd on a rltnl db. It a type of assertions. It can apply to a set of FD to derive other FD.

1. Reflexive Rule: If $y \subseteq x$, then $x \to y$ name, adrs → $x = \{a, b, c, d\}$ $y = \{a, b, c\}$

2. Augmentation Rule: if $x \to y$, then $xz \to yz$ to combine (adding upon) ↳ R(A, BCD), if $A \to B$ then $AC \to BC$

3. Transitive Rule: if $X \to Y$ & $Y \to Z$, then $X \to Z$

   Ar

4. Decomposition or Projective Rule: if $x \to yz$, then $x \to y$ & $x \to z$ spliting / dividing

5. Union or Additive Rule: if $x \to y$ & $x \to z$, then $x \to yz$. joining

6. Pseduo Transitive Rule: if $x \to y$ & $wy \to z$, then $wx \to z$

   combo of trans & augmentn rule

Sound:- that generate only fd that actually hold.

Complete:- genrt all fd that hold. They r Ar. ax.

eg. R = (A, B, C, G, H, I)

F = { $A \to B$
$A \to C$
$CG \to H$
$CG \to I$
$B \to H$ }

9/19

Module-IV

* Functional Dependancies

Let x & y be the two attributes of a rltn. Given the value of x, if there is only one value for y corresponding to it, then y is said to be functionally dependent on x. Indicated by notation $X \to Y$. Y is functionally dependent upon X.

* Inference Rules for Functional Dependencies / Armstrong's axioms

Ar'ax r used to conclude fun'l d° on a rltnl db. It r type of assertion. It can apply to a set of FD to derive other FD.

1. Reflexive Rule: If $y \subseteq x$, then $x \to y$ name, addr → address; $x = \{a,b,c,d\}$ $y = \{a,b,c\}$

2. Augmentation Rule: if $x \to y$, then $xz \to yz$ ↳ $R(A,BCD)$, if $A \to B$ then $AC \to BC$

3. Transitive Rule: if $X \to Y$ & $Y \to Z$, then $X \to Z$

4. Decomposition or Projective Rule: if $x \to yz$, then $x \to y$ & $x \to z$

5. Union or Additive Rule: if $x \to Y$ & $x \to Z$, then $x \to YZ$.

6. Pseduo Transitive Rule: if $X \to Y$ & $WY \to Z$, then $WX \to Z$

Sound :- that generate only fd that actually hold.
Complete :- generate all fd that hold.

eg. $R = (A, B, C, G, H, I)$

$F = \{A \to B$
$A \to C$
$CG \to H$
$CG \to I$
$B \to H\}$

find functional dependencies (F^+) of Fby

① → $A \to B$ $A \to C$

Union :- $A \to B, C$

② → $CG \to H$ $CG \to I$

Union: $CG \to H, I$

③ → $A \to G$

$AG \to CG$ → Augment :- G

$CG \to H$

tran ⇒ $AG \to H$:- Pseudo transitivity

④ $A \to B$ (given)

$B \to H$ (given)

$A \to H$ (transitivity rule)

$A \to C$ = $AG \to CG$ Augment

$CG \to H$ given

→ $AG \to H$ Transitive

• Minimal sets of functional dependencies / Minimal canonial cover

A canonical cover of F (f) a minimal set of functional dependencies equivalent to F, having no redundant dependencies / redundant parts of dependencies.

eg. {$A \to B$, $B \to C$}, He informed from that $A \to C$.

A set of functional dependencies F to be minimal, it should satisfy the following condition -

- every dependency in F has a single attribute for its RHS.
- We cannot remove a dependencies from F & have a set of dependencies that is equivalent to F.
- We can't replace any dependency $x \to A$ in F with a dependency $Y \to A$ where $y \subseteq x$ & still have a set of dependencies that is equivalent to F.

$x = \{A \to B, B \to C, A \to CD\}$ Redundant

$y = \{A \to B, B \to C, A \to D\}$ minimal

or

$x = \{A \to B, B \to C, A \to C\}$ Remove, exp transitivity

↳ Redundant

$y = \{A \to B, B \to C\}$ ↳ minimal

$y \subseteq x$

* Normalization

Reduce redundancies & inconsistencies (data correctness)

* First Normal form:

The domains of the attribute must include only atomic values. It was design to disallow multivalued, single values composite attributes & their combination. (nested attributes) eg:- Student

Rollno	Name	Dept	Year	phonenumber
1	Anju	MCA	1	234567
2	Merry	MBA	2	467892
3	clairi	MCA	1	489435
4	Jisit	MBA	2	498763
5	-	Btech	3	-

Second Normal Form

A rltn is said to be (fn) 2NF, if it is already in 1NF & the non key attributes are fully functionally dependent upon the primary key attribute.

eg:- Student Relation

Studid	Name	courseid	coursename	Grade
101	Johnson	MS1250	Mtech	A+
101	Johnson	MS1415	MBA	A
102	Lennin	MS1331	Hotelman	B+
103	chris	MS1455	MCA	B

Primary key - studid, courseid

(Name get from studid) Studid ← Name } Partial dependencies
courseid ← coursename } part of the key

studid, courseid ← grade } fully dependencies
(all)
(depend on 2 primary key)

~~Sto~~

- Solution

stud

studid	Name
101	Johnson
102	Lennis
103	chris

course

courseid	coursename	~~Grade~~
MS1250	Mtech	~~A+~~
MS1415	MBA	~~A~~
MS1331	Hotel Mng	~~B+~~
MS1455	MCA	~~B~~

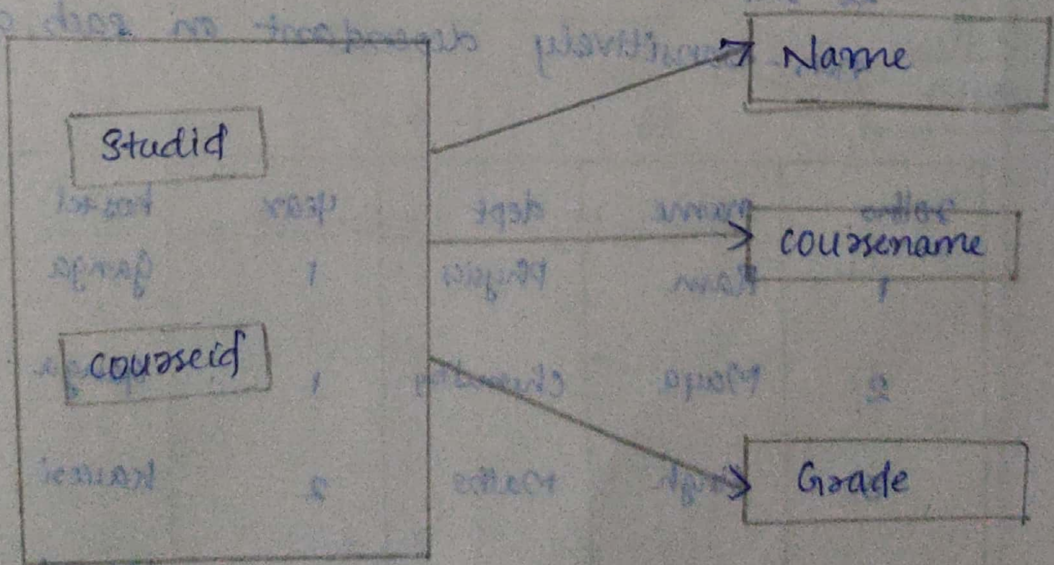
studid → name

courseid ← coursename ~~→ grade~~

studid	courseid	grade
101	MS1250	A+
101	MS1415	A
102	MS1331	B+
103	MS1455	B

studGrade

studid, courseid → grade



book

foreign key bookid references tablename <author>

bookid number (3)
bookname varchar (4)

foreign key bookid references author

- **Third Normal Form**

A table is in 3NF, iff :

(i) It is already in 2NF

(ii) No two non-prime attributes should be functionally dependent on each other.
ie all the non-prime (non-key) attributes should be non-transitively dependent on each other

rollno	name	dept	year	hostel
1	Ram	Physics	1	Ganga
2	Maya	Chemistry	1	Ganga
3	Singh	Maths	2	Kaveri
4	Rajan	CS	2	Kaveri
5	Krishnan	CS	3	Krishna

given yr of study of student his/her hostel can be known ... depending on assigned the yr, hostel, which is not non-prime/non-key attribute

Unnecessary dependency of data (...)

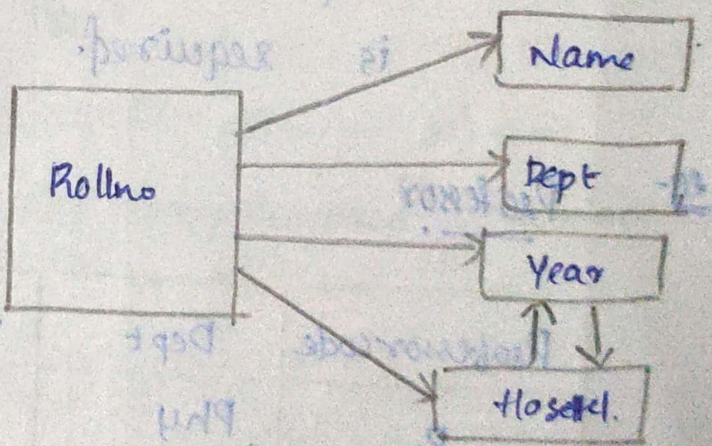
Scanned with CamScanner

(i) Student

rollno	name	dept	year
1	Ram	Phy	~~ganga~~ 1
2	Maya	Chem	~~Ganga~~ 1
3	Singh	~~2~~ Maths	~~Kaveri~~ 2
4	Rajan	~~2~~ CS	~~kaveri~~ 2
5	Krishnan	~~3~~ CS	~~krishna~~ 3

(ii) Stud-hostel

year	hostel
1	Ganga
2	kaveri
3	krishna.



Boyce - Codd Normal Form (BCNFF)

BCNF arises when:

non-key→ nonkey attr. & non-key

(a) the table is already in 3NF. non prime attrib, no dependency

(b) A rltn has more than one possible key. Profcode & Dept OR Procode & Hod

(c) furthermore, the composite keys have a common attribute. more than 1 attrib as key. Pcode

(d) If an attribute of a composite key is dependent on the attribute of the other composite key, a normalization BCNF is required. dependency Dept & HOD

eg- Professor

Professorcode	Dept	Head of Dept	Percent
P_1	Phy	Rao	50
P_1	Maths	Krishnan	30
P_2	Chem	Ghosh	60
P_2	Phy	Rao	25
P_3	Maths	Krishnan	30

one dept has only 1 HOD for each dept.

keys - composite key: pcode & Dept OR Pcode & HOD

Redundancy

dependency - HOD & Dept

non-prime - part if is part of composite key

Scanned with CamScanner

Profcode	Dept	Percent
P_1	Phy	50
P_1	Maths	30
P_2	Chem	60
P_2	Phy	25
P_3	Maths	30

Dept	~~Research~~ HOD
Phy	Rao
Maths	Krishnan
chem	Ghosh.

Fourth Normal Form

A rltn is in ~~fourth~~ 4NF iff :

(i) It is already in BCNF

(ii) It contains non-trivial multivalued dependancies.

⇒ multivalued attribute

multivalued dependencies (MVD)

A ↠ B MVD represents a dependencies between attributes

A ↠ C eg: A, B & C in a rltn S, for each value of A, there is a set of value for B & a set of values for C.

B & C r independant

However the set of values of B & C are independant of each other.

eg:- ModelNo Manufacturing year colour

ModelNo	Manufacturing year	colour
M101	2010	Red
M101	2010	Black
M102	2013	Red
M102	2014	Black
M103	2014	Red

model / manufactur
model / color

ModelNo →→ Manufacturing yr

ModelNo →→ colour

Manufacturing yr & colour r independent to each other.

eg of 4NF

vendor - supply - Projects rltn.

Supplier

Vendorcode	Itemcode	Projectcode
V_1	I_1	P_1
V_1	I_2	P_1
V_2	I_2	P_2
V_2	I_3	P_2
V_3	I_1	P_3

tabl: Vendorsupply

Vendorcode	Itemcode
V_1	I_1
V_1	I_2
V_2	I_2
V_2	I_3
V_3	I_1

table-2
VendorProject

Vendorcode	ProjectNo
V_1	P_1
V_2	P_2
V_3	P_3

$A \twoheadrightarrow B$

Studname, address $\twoheadrightarrow$ address

A MVD can be classified as trivial & non-trivial.
An MVD $A \twoheadrightarrow B$ in a relation is defined as being trivial, if B is a subset of A.

• Fifth Normal Form

A relation is said to be 5NF, iff
1. It is already in 4NF
2. There should not be any join dependencies

a table is subject to a j.dep if it can always be recreated by joining multiple tables each having a subset of the attributes T.

eg:- table vendor supply
table 1

Vendorcode	Itemcode
V_1	I_1
V_1	I_2
V_2	I_2
V_2	I_3
V_3	I_1

Projects relation
table :2

Vendorcode	Projectcode
V_1	P_1
V_2	P_2
V_3	P_3

table:3.

Itemcode	Projectcode
I_1	P_1
I_2	P_1
I_2	P_2
I_3	P_2
I_1	P_3

Drawback / Problems

- Pitfalls in a rltnl database / Anomalies in db.

Stud_grade

Name	Course	Phnno	Major	Professor	Grade
Jones	353	23746843	CS	Smith	A
Smith	329	43153876	Physics	Turner	B
Martin	456	531-64789	Chemistry	Dark	B
Puke	293	444-75689	Maths	James	A+
~~Jo~~nes	379	237-4683	History	Lamb	A

- Update Anomalies
- Insertion Anomalies
- Deletion Anomalies

• Name ← Phn no
• Name ← Major
• Course ← Professor
• Name course ← grade

The anomalies freed here are

• Update Anomalies :- A change in the phn no: of johns must be need for consistancy, in all the tuples pretaining to the student john

• Insertion Anomalies: If this is the only rltn in the db showing the association between the faculty member & the course, the fact that a given professor is teaching a given course cannot be entered to the db unless a student is registered in the course.

• Deletion Anomalies: If the only student registered in the given course discontinues the course the informtn has to which professor is offering the course will be losed, if this is the only rltn in the db showing the Association b/w a faculty member & the course.

→ Properties of Relational Decomposition :- lossless & lossy

↓ Sm information loss after retrieving data from the decomposition

- Lossless join Decomposition :- (no inf loss)

$R(A,B,C)$ → $R_1(A,B)$, $R_2(A,C)$

eg:- studid, name, marks → studid, name ; studid, mark

Recover

$R'(A,B,C)$

$R(A,B,C) = R'(A,B,C)$

- Lossy Decomposition :-

$R = (A,B)$ $\quad R_1 = (A)$ $\quad R_2 = (B)$

A	B
α	1
α	2
β	1

R_1

A
α
β

R_2

B
1
2

$R_1 \bowtie R_2$

A	B
α	1
α	2
β	1
β	2

• **Propertise 1**

After decomposing & trying to retreive orginal data, if the orignal infomtn is losed it comes under lossy decomposition.

If we able to retreive or reconstruct the original data that we had before applying decomposition even after decomposing & joining the table it comes under lossless decomposition.

The point here is that when we applying decomposition in a rltnl db compulsory it shld be lossless decompstn.

• **Propertise 2**

÷ No Redundancy

· The rltn shld be either in BCNF or in 3NF.

· We want to keep dependencies since losing a dependency means the corresponding constraint can be check only through natural join of the appropriate resultant relation in the decomposition. This could be very expensive, so our aim is to get a lossless dependency preserving decomposition.

Example: $R = (A, B, C)$, $F = (A \to B , B \to C)$

Decomposition of R: $R_1 = (A, C)$, $R_2 = (B, C)$

Solution: In R_1 the following dependencies hold : $F' = (A \to A , C \to C, A \to C, AC \to AC)$

augmt => AC &
reflex get AC→AC

In R_2 the following dependencies hold : $F^2 = (B \to B, C \to C, B \to C, BC \to BC)$

not derived the fn dep [illegible] could preserve [illegible]

The set of dependencies hold on $R_1 \cup R_2$: $\{B \to C\}$, $A \to B$ cannot be derived so decomposition is not dependency preserving.

<u>Example-1</u>:

$R = (A, B, C)$ $F = (A \to B, B \to C)$

Decomposition of R: $R_1 = (A, B)$, $R_2 = (B, C)$

Does the decomposition preserve the given dependencies?

ans:- In R_1 the following dependencies hold : $F^1 (A \to A, B \to B, B \to A)$ $AB \to AB$

In R_2 the following dependencies hold : $F^2 (B \to B, C \to C; C \to B)$ $BC \to BC)$

The set of dependencies hold on $R_1 \cup R_2 = \{A \to B, B \to C\}$

The decomposition preserves the functional dependencies given in the original rtn.