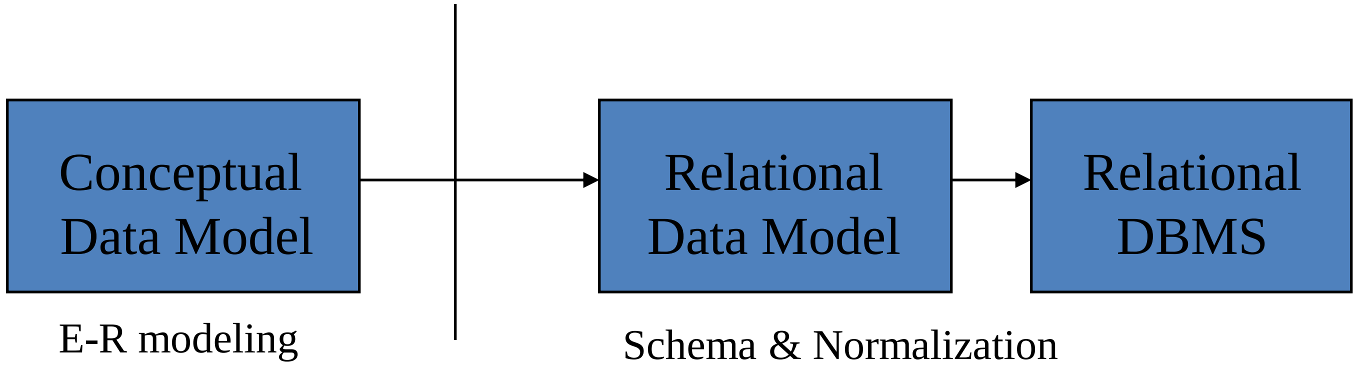
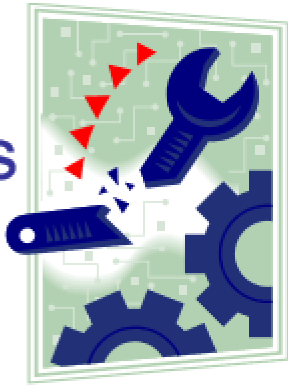

Conceptual Data Model
Relational Data Model
Relational DBMS
E-R modeling
Schema & Normalization

- **Big (and common) DB Problem:**

  In a ‘poorly designed’ DB information is stored redundantly



Consequences:

- Wastes storage
- Causes problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

Our objective:

- Design a schema that does not suffer from the insertion, deletion and update anomalies.

Special Care for NULL values

- Relations should be designed such that their tuples will have as few NULL values as possible
- Attributes that are NULL frequently could be placed in separate relations (with the primary key)

- **Reasons for nulls:**
  - Attribute not applicable or invalid
  - Attribute value unknown (may exist)
  - Value known to exist, but unavailable

We might need to decompose a schema to achieve what we want.

Decomposition of a relation schema

If R doesn't satisfy a particular normal form,
we decompose R into smaller schemas

What's a decomposition?

$R = (A_1, A_2, \ldots, A_n)$

$D = (R_1, R_2, \ldots, R_k)$ st $R_i \subseteq R$ and $R = R_1 \cup R_2 \cup \ldots \cup R_k$

(R_i's need not be disjoint)

Replacing R by $R_1, R_2, \ldots, R_k$ – process of decomposing R

Ex: gradeInfo (rollNo, studName, course, grade)

R_1: gradeInfo (<u>rollNo, course</u>, grade)

R_2: studInfo (<u>rollNo</u>, studName)

- There are two important properties of decompositions:
  a) Non-additive or losslessness of the corresponding join
  b) Preservation of the functional dependencies.

- Note that:
  - Property (a) is extremely important and *cannot* be sacrificed.
  - Property (b) is less stringent and may be sacrificed.

(i) Lossless join property

- the information in an instance r of R must be preserved in the instances $r_1, r_2, \ldots, r_k$ where $r_i = \pi_{R_i}(r)$

(ii) Dependency preserving property

- if a set F of dependencies hold on R it should be possible to enforce F by enforcing appropriate dependencies on each r_i

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "**lossless join**" property is used to guarantee meaningful results for join operations
- **GUIDELINE :**
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

One generic example of generation of spurious tuples

$R = (A, B, C); R_1 = (A, B); R_2 = (B, C)$

Original info is distorted

r:

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_3

Lossy join

r_1:

A	B
a_1	b_1
a_2	b_2
a_3	b_1

r_2:

B	C
b_1	c_1
b_2	c_2
b_1	c_3

$r_1 * r_2$:

A	B	C
a_1	b_1	c_1
a_1	b_1	c_3
a_2	b_2	c_2
a_3	b_1	c_1
a_3	b_1	c_3

Spurious tuples

One more example of generation of spurious tuples

Consider relation r(ABC) and its projections r1(AB) and r2(BC).

r

A	B	C
1	2	3
1	2	4
1	0	4
2	0	1
2	2	3

ρ

r1

A	B
1	2
1	0
2	0
2	2

r2

B	C
2	3
2	4
0	4
0	1
2	3

⊗

r'

A	B	C
1	2	3
1	2	4
1	0	4
1	**0**	**1**
...	...	...
2	**0**	**4**
2	0	1
2	2	3

Phantom records

Original info is distorted

Observation

Not all decompositions of a table can be combined using *natural join* to reproduce the original table.

Lossless joins are also called non-additive joins

Example of generation of spurious tuples

r1

A	B	C
1	2	3
4	2	3
2	2	6
3	1	4

r2

B	C	D
2	3	4
2	3	5
1	4	2

$\otimes$

$r = r1 \otimes r2$

A	B	C	D
1	2	3	4
1	2	3	5
4	2	3	4
4	2	3	5
3	1	4	2

ρ

$r_1' = \Pi_{ABC}(r)$

A	B	C
1	2	3
4	2	3
3	1	4

$r2' = \Pi_{BCD}(r)$

B	C	D
2	3	4
2	3	5
1	4	2

Consider the following two relations $r_1(ABC)$ and $r_2(BCD)$.

Compute natural join $r = r_1 \otimes r_2$

Evaluate projections
$r_1' = \Pi_{ABC}(r)$ and $r_2' = \Pi_{BCD}(r)$

Observation

Tables r_2 and r_2' are the same however tuple <2,2,6> $\in r_1$ but not present in r_1'

Testing for lossless decomposition property(1/6)

R – given schema with attributes $A_1, A_2, \ldots, A_n$
F – given set of FDs
D – $\{R_1, R_2, \ldots, R_m\}$ given decomposition of R

Is D a lossless decomposition?

Create an $m \times n$ matrix S with columns labeled as $A_1, A_2, \ldots, A_n$
and rows labeled as $R_1, R_2, \ldots, R_m$

Initialize the matrix as follows:
set $S(i,j)$ as symbol b_{ij} for all i,j.
if A_j is in the scheme R_i, then set $S(i,j)$ as symbol a_j , for all i,j

Testing for lossless decomposition property(2/6)

After *S* is initialized, we carry out the following process on it:

repeat
  for each functional dependency $U \rightarrow V$ in *F* **do**
    for all rows in *S* which agree on *U*-attributes **do**
      make the symbols in each *V*- attribute column
        the *same* in all the rows as follows:
          if any of the rows has an "*a*" symbol for the column
            set the other rows to the same "*a*" symbol in the column
          else // if no "*a*" symbol exists in any of the rows
            choose one of the "*b*" symbols that appears
            in one of the rows for the *V*-attribute and
            set the other rows to that "b" symbol in the column
until no changes to S

At the end, if there exists a row with all "*a*" symbols then D is lossless otherwise D is a lossy decomposition

Testing for lossless decomposition property(3/6)

R = (rollNo, name, advisor, advisorDept, course, grade)

FD's = { rollNo → name; rollNo → advisor; advisor → advisorDept; rollNo, course → grade}

D : { R_1 = (rollNo, name, advisor), R_2 = (advisor, advisorDept), R_3 = (rollNo, course, grade) }

Matrix S : (Initial values)

	rollNo	name	advisor	advisor Dept	course	grade
R_1	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	b_{25}	b_{26}
R_3	a_1	b_{32}	b_{33}	b_{34}	a_5	a_6

Testing for lossless decomposition property(4/6)

R = (rollNo, name, advisor, advisorDept, course, grade)

FD's = { rollNo → name; rollNo → advisor; advisor → advisorDept;
rollNo, course → grade}

D : { R_1 = (rollNo, name, advisor), R_2 = (advisor, advisorDept),
R_3 = (rollNo, course, grade) }

Matrix S : (After enforcing rollNo → name & rollNo → advisor)

	rollNo	name	advisor	advisor Dept	course	grade
R_1	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	b_{25}	b_{26}
R_3	a_1	$\cancel{b_{32}} a_2$	$\cancel{b_{33}} a_3$	b_{34}	a_5	a_6

Testing for lossless decomposition property(5/6)

R = (rollNo, name, advisor, advisorDept, course, grade)

FD's = { rollNo → name; rollNo → advisor; advisor → advisorDept;
rollNo, course → grade}

D : { R_1 = (rollNo, name, advisor), R_2 = (advisor, advisorDept),
R_3 = (rollNo, course, grade) }

Matrix S : (After enforcing advisor → advisorDept)

	rollNo	name	advisor	advisor Dept	course	grade
R_1	a_1	a_2	a_3	$\cancel{b_{14}}a_4$	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	b_{25}	b_{26}
R_3	a_1	$\cancel{b_{32}}a_2$	$\cancel{b_{33}}a_3$	$\cancel{b_{34}}a_4$	a_5	a_6

No more changes. Third row with all *a* symbols. So a lossless join.

Testing for lossless decomposition property(6/6)

R – given schema. F – given set of FDs

The decomposition of R into R_1, R_2 is lossless wrt F if and only if either $R_1 \cap R_2 \rightarrow (R_1 - R_2)$ belongs to F^+ or

$R_1 \cap R_2 \rightarrow (R_2 - R_1)$ belongs to F^+

Eg. gradeInfo (rollNo, studName, course, grade)

with FDs = {rollNo, course → grade; studName, course → grade;

rollNo → studName; studName → rollNo}

decomposed into

grades (rollNo, course, grade) and studInfo (rollNo, studName)

is lossless because

rollNo → studName

A property of lossless joins

D_1: $(R_1, R_2, \ldots, R_K)$ lossless decomposition of R wrt F

D_2: $(R_{i1}, R_{i2}, \ldots, R_{ip})$ lossless decomposition of R_i wrt $F_i = \pi_{R_i}(F)$

Then

$D = (R_1, R_2, \ldots, R_{i-1}, R_{i1}, R_{i2}, \ldots, R_{ip}, R_{i+1}, \ldots, R_k)$ is a lossless decomposition of R wrt F

This property is useful in the algorithm for BCNF decomposition

Dependency Preserving Decompositions

Decomposition $D = (R_1, R_2, \ldots, R_k)$ of schema R *preserves* a set of dependencies F if

$$(\pi_{R_1}(F) \cup \pi_{R_2}(F) \cup \ldots \cup \pi_{R_k}(F))^+ = F^+$$

Here, $\pi_{R_i}(F) = \{ (X \rightarrow Y) \in F^+ \mid X \subseteq R_i, Y \subseteq R_i \}$
(called projection of F onto R_i)

Informally, any FD that logically follows from F must also logically follow from the union of projections of F onto R_i's
Then, D is called dependency preserving.

An example of dependency preservation

Schema R = (A, B, C)
FDs $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

Decomposition $D = (R_1 = \{A, B\}, R_2 = \{B, C\})$
$\pi_{R_1}(F) = \{A \rightarrow B, B \rightarrow A\}$
$\pi_{R_2}(F) = \{B \rightarrow C, C \rightarrow B\}$

$$(\pi_{R_1}(F) \cup \pi_{R_2}(F))^+ = \{A \rightarrow B, B \rightarrow A,\ B \rightarrow C, C \rightarrow B,\ A \rightarrow C, C \rightarrow A\} = F^+$$

Hence Dependency preserving

- Normalization theory is based on the observation that relations with certain properties are more effective in inserting, updating and deleting data than other sets of relations containing the same data

- Normalization is a multi-step process beginning with an “unnormalized” relation

Normalization

Normalization

We discuss four normal forms: first, second, third, and Boyce-Codd normal forms
1NF, 2NF, 3NF, and BCNF

Normalization is a process that "improves" a database design by generating relations that are of higher normal forms.

The *objective* of normalization:
"*to create relations where every dependency is on the key, the whole key, and nothing but the key*".

Normalization

There is a sequence to normal forms:

1NF is considered the weakest,
2NF is stronger than 1NF,
3NF is stronger than 2NF, and
BCNF is considered the strongest

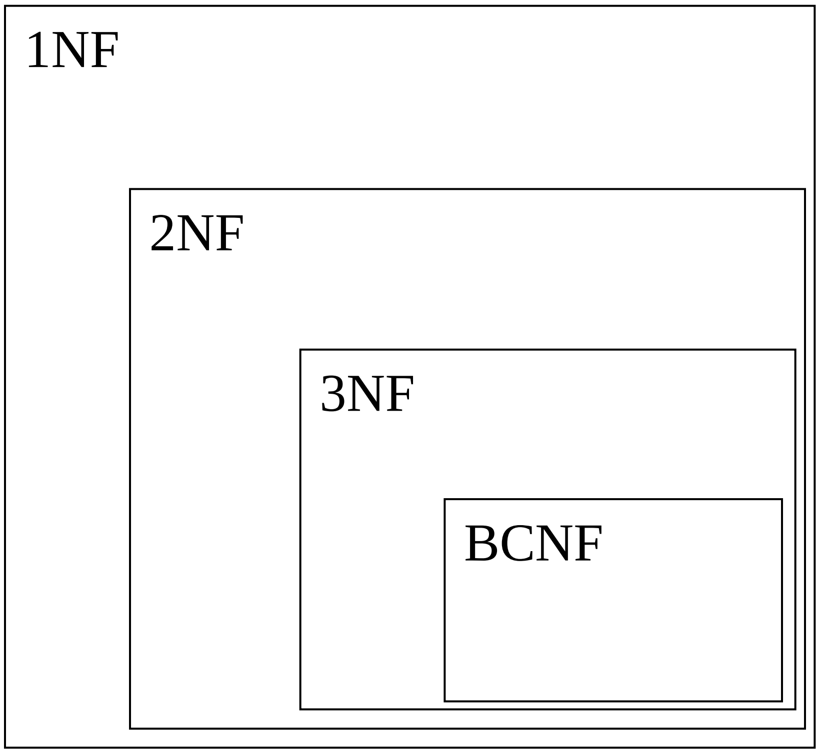
Also,

any relation that is in BCNF, is in 3NF;
any relation in 3NF is in 2NF; and
any relation in 2NF is in 1NF.

Normalization

Unnormalized Form,
1NF,
2NF,
3NF/Elementary Key normal form (EKNF),
3.5NF / BCNF,
4NF,
5NF / Project-Join normal form (PJNF),
Domain-Key normal Form,
6NF

Normalization



a relation in BCNF, is also in 3NF

a relation in 3NF is also in 2NF

a relation in 2NF is also in 1NF

Normalization

We consider a relation in BCNF to be fully normalized.

The benefit of higher normal forms is that update semantics for the affected data are simplified.

This means that applications required to maintain the database are simpler.

A design that has a lower normal form than another design has more redundancy. Uncontrolled redundancy can lead to data integrity problems.

First we revisit the concept of ***functional dependency***

Determinant

Functional Dependency

EmpNum $\rightarrow$ EmpEmail

Attribute on the LHS is known as the ***determinant***

- EmpNum is a determinant of EmpEmail

Transitive dependency

Transitive dependency

Consider attributes A, B, and C, and where

$A \rightarrow B$ and $B \rightarrow C$.

Functional dependencies are transitive, which means that we also have the functional dependency

$A \rightarrow C$

We say that C is transitively dependent on A through B.

Transitive dependency

EmpNum → DeptNum

EmpNum	EmpEmail	DeptNum	DeptNname

DeptNum → DeptName

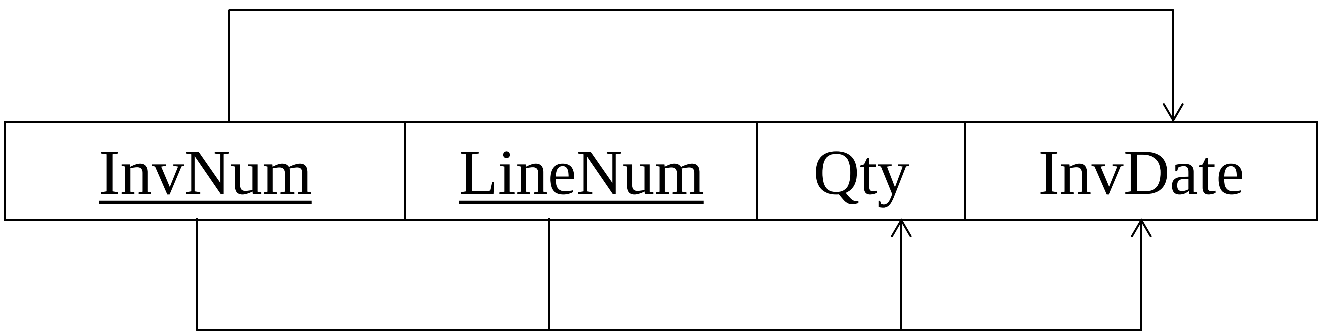
EmpNum	EmpEmail	DeptNum	DeptNname

DeptName is *transitively dependent* on EmpNum via DeptNum

EmpNum → DeptName

Partial dependency

A **partial dependency** exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

First Normal Form

First Normal Form

We say a relation is in **1NF** if all values stored in the relation are single-valued and atomic.

1NF places restrictions on the structure of relations. Values must be simple.

First Normal Form

- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic

- Considered to be part of the definition of relation

First Normal Form

The following in **not** in 1NF

<u>**EmpNum**</u>	**EmpPhone**	**EmpDegrees**
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

EmpDegrees is a multi-valued field:

employee 679 has two degrees: *BSc* and *MSc*

employee 333 has three degrees: *BA*, *BSc*, *PhD*

First Normal Form

<u>EmpNum</u>	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

To obtain 1NF relations we must, without loss of information, replace the above with two relations - see next slide

First Normal Form

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

An outer join between Employee and EmployeeDegree will produce the information we saw before

Second Normal Form

Second Normal Form

A relation is in **2NF** if it is in 1NF, and every non-key attribute is fully dependent on each candidate key. (That is, **we don't have any partial functional dependency**.)

- 2NF (and 3NF) both involve the concepts of key and non-key attributes.
- A *key attribute* is any attribute that is part of a key; any attribute that is not a key attribute, is a *non-key attribute*.
- Relations that are not in BCNF have data redundancies
- A relation in 2NF will not have any partial dependencies

Second Normal Form

Consider this **InvLine** table (in 1NF)**:**

<u>InvNum</u>	<u>LineNum</u>	ProdNum	Qty	InvDate

InvNum, LineNum $\longrightarrow$ ProdNum, Qty

There are two candidate keys.

Qty is the only non-key attribute, and it is dependent on InvNum

InvNum $\longrightarrow$ InvDate

Since there is a determinant that is not a candidate key, InvLine is **not BCNF**

InvLine is **not 2NF** since there is a partial dependency of InvDate on InvNum

InvLine is only in **1NF**

Second Normal Form

InvLine

<u>InvNum</u>	<u>LineNum</u>	ProdNum	Qty	InvDate

The above relation has redundancies: the invoice date is repeated on each invoice line.

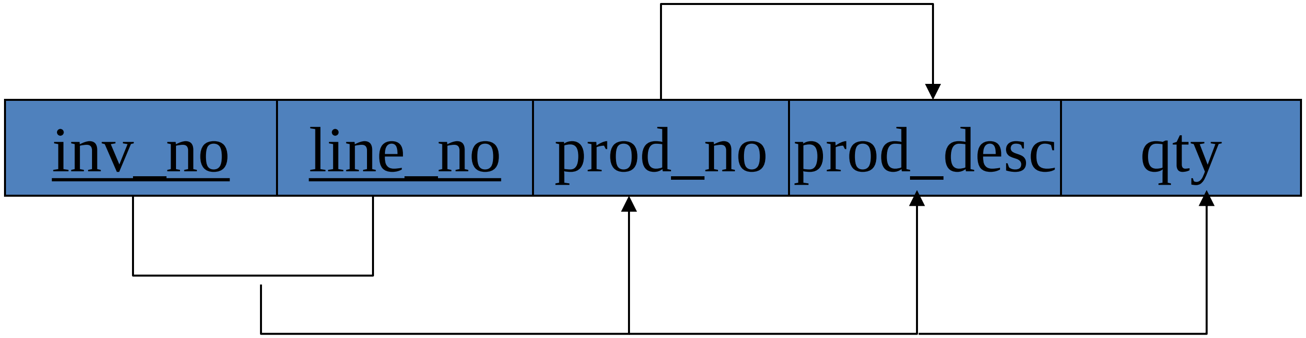
We can *improve* the database by decomposing the relation into two relations:

<u>InvNum</u>	<u>LineNum</u>	ProdNum	Qty

<u>InvNum</u>	InvDate

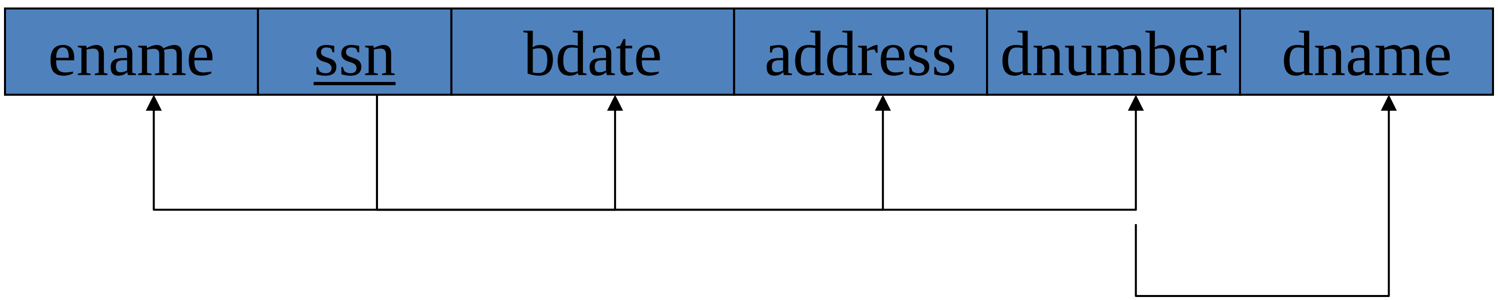
Question: What is the highest normal form for these relations? 2NF? 3NF? BCNF?

Is the following relation in 2NF?



2NF, but not in 3NF, nor in BCNF:

EmployeeDept



since dnumber is not a candidate key and we have:

dnumber $\rightarrow$ dname.

Third Normal Form

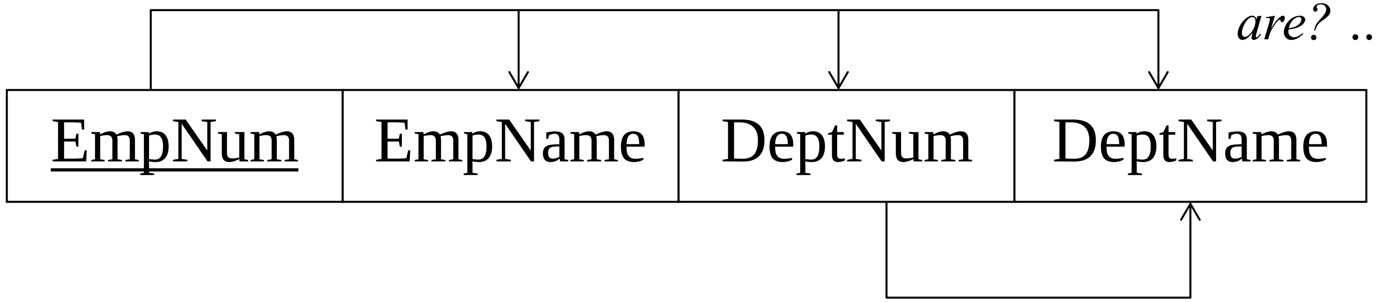
Third Normal Form

- A relation is in **3NF** if the relation is in 1NF and all determinants of *non-key* attributes are candidate keys
  That is, for any functional dependency: $X \rightarrow Y$, where Y is a non-key attribute (or a set of non-key attributes), X is a candidate key.
- This definition of 3NF differs from BCNF only in the specification of non-key attributes - 3NF is weaker than BCNF. (BCNF requires all determinants to be candidate keys.)
- A relation in 3NF will not have any transitive dependencies of non-key attribute on a candidate key through another non-key attribute.

Third Normal Form

Consider this **Employee** relation

Candidate keys are? …



EmpName, DeptNum, and DeptName are non-key attributes.

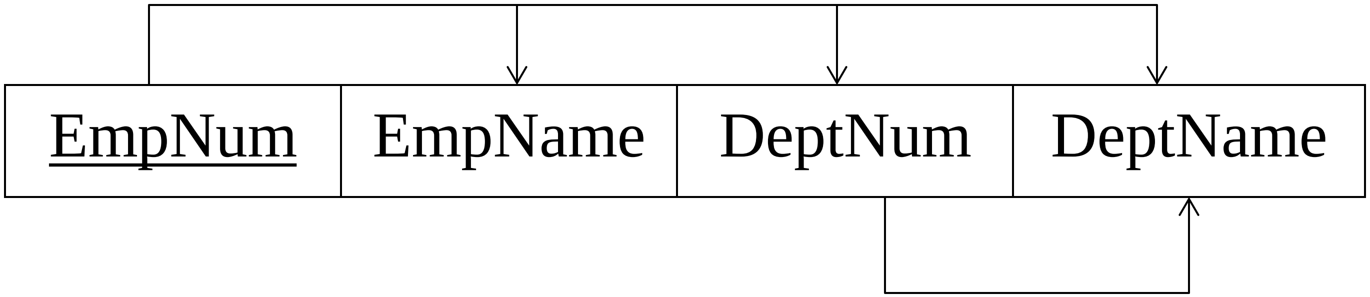
DeptNum determines DeptName, a non-key attribute, and DeptNum is not a candidate key.

Is the relation in 3NF? … no

Is the relation in BCNF? … no

Is the relation in 2NF? … yes

Third Normal Form



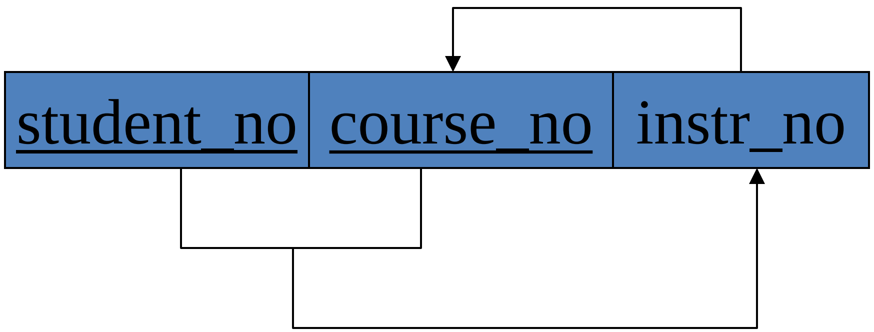
We correct the situation by decomposing the original relation into two 3NF relations. Note the decomposition is *lossless*.

EmpNum	EmpName	DeptNum

DeptNum	DeptName

Verify these two relations are in 3NF.

In 3NF, but not in BCNF:

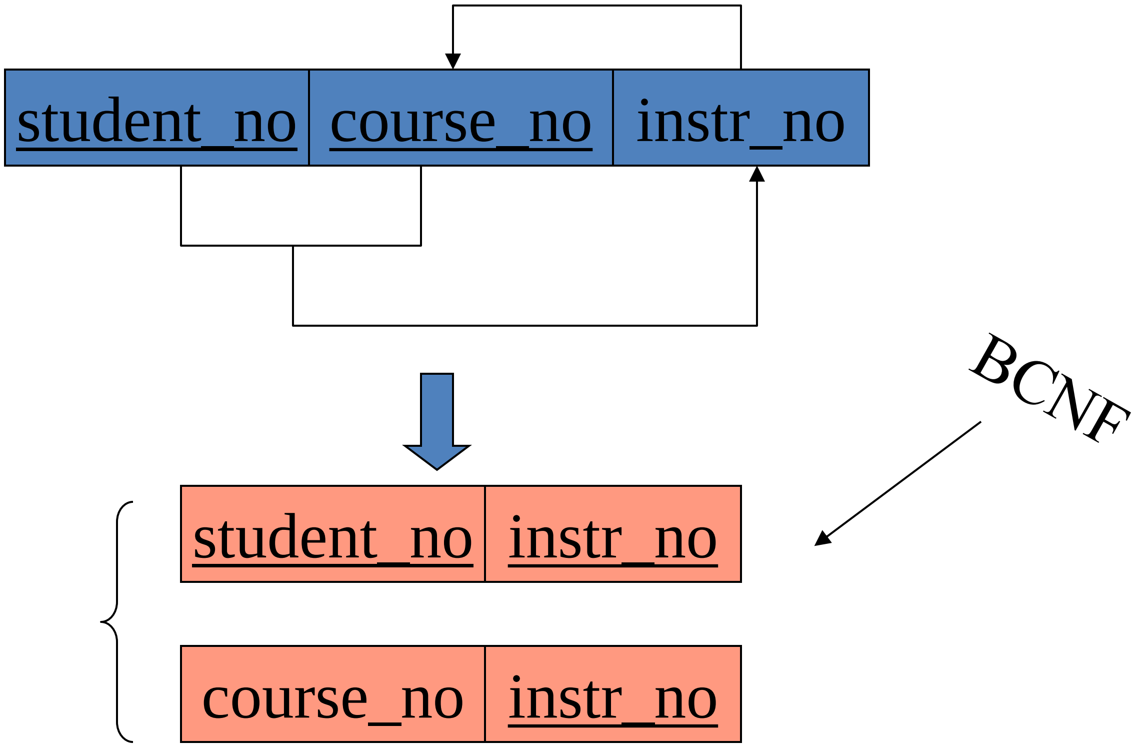


Instructor teaches one course only.

Student takes a course and has one instructor.

{student_no, course_no} → instr_no
instr_no → course_no

since we have instr_no → course-no, but instr_no is not a Candidate key.



{student_no, instr_no} → student_no
{student_no, instr_no} → instr_no
instr_no → course_no

Boyce-Codd Normal Form

- But, even if reln is in 3NF, some problems could arise.
  - e.g., Reserves SBDC, $S \rightarrow C$, $C \rightarrow S$ is in 3NF, but for each reservation of sailor S, same (S, C) pair is stored.
- Thus, 3NF is indeed a compromise relative to BCNF.
- Examples: Assign (Flight, Day, Pilot, Gate)

  $FD \rightarrow PG$; $F \rightarrow G$. How to fix it?

  $R(A,B,C,D)$ $F=\{A \rightarrow BCD; C \rightarrow D\}$

Boyce-Codd Normal Form

Boyce-Codd Normal Form

BCNF is defined very simply:

a relation is in BCNF if it is in 3NF and if every determinant is a candidate key.

- **BCNF ensures zero redundancy**
- **BCNF ensures lossless property but does not ensure dependency preservation.**

Decomposition of a Relation Scheme

- Suppose that relation R contains attributes *A1 ... An*. A ***decomposition*** of R consists of replacing R by two or more relations such that:
  - Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
  - Every attribute of R appears as an attribute of one of the new relations.
- Intuitively, decomposing R means we will store instances of the relation schemes produced by the decomposition, instead of instances of R.
- E.g., decompose SNLRWH into SNLRH and RW.

Example Decomposition (1)

- Real estate property on various counties in VA.
  - Lot (pid, county-name, lot#, size, tax-rate, price)
  - FD: pid $\longrightarrow$ Lot,

    c-name, lot# $\longrightarrow$ Lot

    c-name $\longrightarrow$ tax-rate

    size $\longrightarrow$ price

Is it in BCNF? 3NF?

Example Decomposition (2)

- Real estate property on various counties in VA.
  - Lot (pid, county-name, lot#, size, tax-rate, price)
  - FD: pid $\rightarrow$ Lot,

    c-name, lot# $\rightarrow$ Lot

    c-name $\rightarrow$ tax-rate

    size $\rightarrow$ price
- Partial dependency: c-name, lot# $\rightarrow$ c-name $\rightarrow$ tax-rate

Lot1(pid, c-name, lot#, size, price)

Lot2(c-name, tax-rate)

BCNF? 3NF?

Example Decomposition (3)

- Real estate property on various counties in VA.
  - Lot (pid, county-name, lot#, size, tax-rate, price)
  - FD: pid $\rightarrow$ Lot,

    c-name, lot# $\rightarrow$ Lot

    c-name $\rightarrow$ tax-rate

    size $\rightarrow$ price
- Transitive dependency: pid $\rightarrow$ size $\rightarrow$ price

Lot11(pid, c-name, lot#, size)

Lot12(size, price)

Lot2(c-name, tax-rate)

BCNF? 3NF?

Example Decomposition

- Decompositions should be used only when needed.
  - SNLRWH has FDs $S \rightarrow$ SNLRWH and $R \rightarrow W$
  - Second FD causes violation of 3NF; W values repeatedly associated with R values. We decompose SNLRWH into SNLRH and RW
- The information to be stored consists of SNLRWH tuples. If we just store the projections of these tuples onto SNLRH and RW, are there any potential problems that we should be aware of?

Problems with Decompositions

- There are three potential problems to consider:
  - Some queries become more expensive.
    - e.g., How much did sailor Joe earn? (salary = W*H)
  - Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
    - Fortunately, not in the SNLRWH example.
  - Checking some dependencies may require joining the instances of the decomposed relations.
    - Fortunately, not in the SNLRWH example.
- *Tradeoff*: Must consider these issues vs. redundancy.

Lossless Join Decompositions

- Decomposition of R into X and Y is *lossless-join* w.r.t. a set of FDs F if, for every instance *r* that satisfies F:
  - $\pi_X(r) \bowtie \pi_Y(r) = r$
- It is always true that $r \subseteq \pi_X(r) \bowtie \pi_Y(r)$
  - In general, the other direction does not hold! If it does, the decomposition is lossless-join.
- Definition extended to decomposition into 3 or more relations in a straightforward way.
- *It is essential that all decompositions used to deal with redundancy be lossless!*

More on Lossless Join

- The decomposition of R into X and Y is lossless-join wrt F if and only if the closure of F contains:
  - $X \cap Y \to X$, or
  - $X \cap Y \to Y$
- In particular, the decomposition of R into UV and R - V is lossless-join if $U \to V$ holds over R.

A	B	C
1	2	3
4	5	6
7	2	8

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

Dependency Preserving Decomposition

- Consider CSJDPQV, C is key, JP $\rightarrow$ C and SD $\rightarrow$ P.
  - BCNF decomposition: CSJDQV and SDP
  - Problem: Checking JP $\rightarrow$ C requires a join!
- Dependency preserving decomposition (Intuitive):
  - If R is decomposed into X, Y and Z, and we enforce the FDs that hold on X, on Y and on Z, then all FDs that were given to hold on R must also hold. *(Avoids Problem (3).)*
- *Projection of set of FDs F*: If R is decomposed into X, ... projection of F onto X (denoted F_X) is the set of FDs U $\rightarrow$ V in F^+ (*closure of F*) such that U, V are in X.

Dependency Preserving Decompositions (Contd.)

- Decomposition of R into X and Y is ***dependency preserving*** if $(F_X \text{ union } F_Y)^+ = F^+$
  - i.e., if we consider only dependencies in the closure F^+ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F^+.
- Important to consider F^+, not F, in this definition:
  - ABC, $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow A$, decomposed into AB and BC.
  - Is this dependency preserving? Is $C \rightarrow A$ preserved?????
- Dependency preserving does not imply lossless join:
  - ABC, $A \rightarrow B$, decomposed into AB and BC.

Decomposition into BCNF

- Consider relation R with FDs F. If X $\rightarrow$ Y violates BCNF, decompose R into R - Y and XY.
  - Repeated application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.
  - e.g., CSJDPQV, key C, JP $\rightarrow$ C, SD $\rightarrow$ P, J $\rightarrow$ S
  - To deal with SD $\rightarrow$ P, decompose into SDP, CSJDQV.
  - To deal with J $\rightarrow$ S, decompose CSJDQV into JS and CJDQV
- In general, several dependencies may cause violation of BCNF. The order in which we ``deal with'' them could lead to very different sets of relations!

BCNF and Dependency Preservation

- In general, there may not be a dependency preserving decomposition into BCNF.
  - e.g., CSZ, CS $\rightarrow$ Z, Z $\rightarrow$ C
  - Can't decompose while preserving 1st FD; not in BCNF.
- Similarly, decomposition of CSJDQV into SDP, JS and CJDQV is not dependency preserving (w.r.t. the FDs JP $\rightarrow$ C, SD $\rightarrow$ P and J $\rightarrow$ S).
  - However, it is a lossless join decomposition.
  - In this case, adding JPC to the collection of relations gives us a dependency preserving decomposition.
    - JPC tuples stored only for checking FD! (*Redundancy!*)

Decomposition into 3NF

- Obviously, the algorithm for lossless join decomp into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier).
- To ensure dependency preservation, one idea:
  - If $X \rightarrow Y$ is not preserved, add relation XY.
  - Problem is that XY may violate 3NF! e.g., consider the addition of CJP to \`preserve' $JP \rightarrow C$. What if we also have $J \rightarrow C$?
- Refinement: Instead of the given set of FDs F, use a *minimal cover for F*.

The following functional dependencies hold for the relations R(ABC) and S(BDE) $B \rightarrow A$, $A \rightarrow C$. The relation R contains 200 tuples and the relation S contains 100 tuples. What is the maximum number of tuples possible in the natural join $R \bowtie S$?

a) 100 b) 200 c) 300 d) 2000

Question solve on Normalization

Let R(A, B, C, D, E, P, G) be a relational schema in which the following functional dependencies are known to hold: $AB \rightarrow CD$, $DE \rightarrow P$, $C \rightarrow E$, $P \rightarrow C$ and $B \rightarrow G$. The relational schema R is

A) in BCNF

B) in 3NF, but not in BCNF

C) in 2NF, but not in 3NF

D) not in 2NF

Answer:

Calculating $(AB)^+$, we get

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

So AB is a candidate key.

But $B \rightarrow G$ is a partial dependency.

So R is not in 2NF.

Question solve on Normalization

Consider the following functional dependencies in a database

$DOB \rightarrow Age$

$Age \rightarrow Eligibility$

$Name \rightarrow Rollno$

$Rollno \rightarrow Name$

$Courseno \rightarrow Coursename$

$Courseno \rightarrow Instructor$

$(Rollno, Courseno) \rightarrow Grade$

The relation (Rollno, Name, DOB, Age) is in

A) 2NF but not in 3NF

B) 3NF but not in BCNF

C) BCNF

D) None of the above

Answer:

For the given relation, following FDs are applicable

DOB → Age
Name → Rollno
Rollno → Name

Candidate keys for the above are
(DOB, Name) and (DOB, Rollno)

But there is a partial dependency here
as DOB → Age

So it is only in 1NF.

So answer is option (D).

GATE-1998

1. Which normal form is considered adequate for normal relational database design?

(a) 2NF (b) 5NF (c) 4NF (d) 3NF

GATE-1998

1. Which normal form is considered adequate for normal relational database design?

(a) 2NF (b) 5NF (c) 4NF (d) 3NF

Ans: option (d)

Explanation:

A relational database table is often described as "normalized" if it is in the Third Normal Form because most of the 3NF tables are free of insertion, update, and deletion anomalies.

GATE-2001

2. Consider a schema R(A, B, C, D) and functional dependencies A -> B and C -> D. Then the decomposition of R into R1 (A, B) and R2(C, D) is

(a) dependency preserving and lossless join

(b) lossless join but not dependency preserving

(c) dependency preserving but not lossless join

(d) not dependency preserving and not lossless join

Ans: option (c)

Explanation:

While decomposing a relational table we must verify the following properties:

i) Dependency Preserving Property: A decomposition is said to be dependency preserving if $F^+=(F1 \cup F2 \cup .. Fn)^+$, Where F^+=total functional dependencies(FDs) on universal relation R, F1 = set of FDs of R1, and F2 = set of FDs of R2.

For the above question R1 preserves A->B and R2 preserves C->D.

Since the FDs of universal relation R is preserved by R1 and R2, the decomposition is dependency preserving.

ii) Lossless-Join Property:

The decomposition is a lossless-join decomposition of R if at least one of the following functional dependencies are in F+:-

a) R1 ∩ R2 -> R1

b) R1 ∩ R2 -> R2

It ensures that the attributes involved in the natural join ( ) are a candidate key for at least one of the two relations.In the above question schema R is decomposed into R1 (A, B) and R2(C, D), and R1 ∩ R2 is empty. So, the decomposition is not lossless.

GATE-2002

3. Relation R with an associated set of functional dependencies, F, is decomposed into BCNF. The redundancy (arising out of functional dependencies) in the resulting set of relations is

(a) Zero

(b) More than zero but less than that of an equivalent 3NF decomposition

(c) Proportional to the size of F+

(d) Indeterminate

Ans: option (b)

Explanation:

Redundancy in BCNF is low when compared to 3NF.

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

- $\alpha \rightarrow \beta$ is a trivial functional dependency (i.e. $\beta \subseteq \alpha$).
- α is a superkey for schema R.

GATE-2005

4. Which one of the following statements about normal forms is FALSE?

(a) BCNF is stricter than 3NF

(b) Lossless, dependency-preserving decomposition into 3NF is always possible

(c) Lossless, dependency-preserving decomposition into BCNF is always possible

(d) Any relation with two attributes is in BCNF

Ans: option (c)

Explanation:

Achieving Lossless and dependency-preserving decomposition property into BCNF is difficult.

GATE-2005 (IT)

5. A table has fields F1, F2, F3, F4, and F5, with the following functional dependencies:

```
F1->F3

F2->F4

(F1,F2)->F5
```

in terms of normalization, this table is in

(a) 1NF (b) 2NF (c) 3NF (d) None of these

Ans: option (a)

Explanation:

Since the primary key is not given we have to derive the primary key of the table. Using the closure set of attributes we get the primary key as (F1,F2). From functional dependencies, "F1->F3, F2->F4", we can see that there is partial functional dependency therefore it is not in 2NF. Hence the table is in 1NF.

GATE-2012

6. Which of the following is TRUE?

(a) Every relation in 2NF is also in BCNF

(b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R

(c) Every relation in BCNF is also in 3NF

(d) No relation can be in both BCNF and 3NF

GATE-2012

6. Which of the following is TRUE?

(a) Every relation in 2NF is also in BCNF

(b) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R

(c) Every relation in BCNF is also in 3NF

(d) No relation can be in both BCNF and 3NF

Ans: option (c)

GATE-2003

7. Consider the following functional dependencies in a database.

```
Date_of_Birth->Age                Age->Eligibility
Name->Roll_number                 Roll_number->Name
Course_number->Course_name        Course_number->Instructor
(Roll_number, Course_number)->Grade
```

The relation (Roll_number, Name, Date_of_birth, Age) is

(a) in second normal form but not in third normal form

(b) in third normal form but not in BCNF

(c) in BCNF

(d) in none of the above

Ans: option (d)

Explanation:

For the given relation only some of the above FDs are applicable. The applicable FDs are given below:

`Date_of_Birth->Age`

`Name->Roll_number`

`Roll_number->Name`

Finding the closure set of attributes we get the candidate keys:

(`Roll_number,Date_of_Birth`), and (`Name,Date_of_Birth`) .

On selecting any one of the candidate key we can see that the FD `Date of Birth->Age` is a partial dependency. Hence the relation is in 1NF.

GATE-2004

8. The relation schema Student_Performance (name, courseNo, rollNo, grade) has the following FDs:

name,courseNo->grade

rollNo,courseNo->grade

name->rollNo

rollNo->name

The highest normal form of this relation scheme is

(a) 2NF (b) 3NF (c) BCNF (d)4NF

GATE-2004

8. The relation schema Student_Performance (name, courseNo, rollNo, grade) has the following FDs:

```
name,courseNo->grade

rollNo,courseNo->grade

name->rollNo

rollNo->name
```

The highest normal form of this relation scheme is

(a) 2NF (b) 3NF (c) BCNF (d)4NF

Ans: option (b)

GATE-2004 (IT)

9. The relation EMPDT1 is defined with attributes empcode(unique), name, street, city, state, and pincode. For any pincode,there is only one city and state. Also, for any given street, city and state, there is just one pincode. In normalization terms EMPDT1 is a relation in

(a) 1NF only

(b) 2NF and hence also in 1NF

(c) 3NF and hence also in 2NF and 1NF

(d) BCNF and hence also in 3NF, 2NF and 1NF

GATE-2004 (IT)

9. The relation EMPDT1 is defined with attributes empcode(unique), name, street, city, state, and pincode. For any pincode,there is only one city and state. Also, for any given street, city and state, there is just one pincode. In normalization terms EMPDT1 is a relation in

(a) 1NF only

(b) 2NF and hence also in 1NF

(c) 3NF and hence also in 2NF and 1NF

(d) BCNF and hence also in 3NF, 2NF and 1NF

Ans: option (b)

GATE-2007

10. Which one of the following statements if FALSE?

(a) Any relation with two attributes is in BCNF

(b) A relation in which every key has only one attribute is in 2NF

(c) A prime attribute can be transitively dependent on a key in a 3 NF relation.

(d) A prime attribute can be transitively dependent on a key in a BCNF relation.

GATE-2007

10. Which one of the following statements if FALSE?

(a) Any relation with two attributes is in BCNF

(b) A relation in which every key has only one attribute is in 2NF

(c) A prime attribute can be transitively dependent on a key in a 3 NF relation.

(d) A prime attribute can be transitively dependent on a key in a BCNF relation.

Ans: option (d)

GATE-2008

11. Consider the following relational schemes for a library database:

Book (Title, Author, Catalog_no, Publisher, Year, Price)

Collection (Title, Author, Catalog_no)

With the following functional dependencies:

```
I. Title Author -> Catalog_no

II. Catalog_no -> Title Author Publisher Year

III. Publisher Title Year -> Price
```

Assume {Author, Title} is the key for both schemes. Which of the following statements is true?

(a) Both Book and Collection are in BCNF

(b) Both Book and Collection are in 3NF only

(c) Book is in 2NF and Collection is in 3NF

(d) Both Book and Collection are in 2NF only

GATE-2008

11. Consider the following relational schemes for a library database:

Book (Title, Author, Catalog_no, Publisher, Year, Price)

Collection (Title, Author, Catalog_no)

With the following functional dependencies:

```
I. Title Author -> Catalog_no

II. Catalog_no -> Title Author Publisher Year

III. Publisher Title Year -> Price
```

Assume {Author, Title} is the key for both schemes. Which of the following statements is true?

(a) Both Book and Collection are in BCNF

(b) Both Book and Collection are in 3NF only

(c) Book is in 2NF and Collection is in 3NF

(d) Both Book and Collection are in 2NF only

Ans: option (c)

Multi-valued Dependencies (MVDs)

studCourseEmail(<u>rollNo,courseNo,emailAddr</u>)

a student enrolls for several courses and has several email addresses

rollNo →→ courseNo (read as rollNo *multi-determines* courseNo)

If (CS05B007, CS370, shyam@gmail.com)

(CS05B007, CS376, shyam@yahoo.com) appear in the data then

(CS05B007, CS376, shyam@gmail.com)
(CS05B007, CS370, shyam@yahoo.com)

should also appear for, otherwise, it implies that having gmail address has something to with doing course CS370 !!

By symmetry, rollNo →→ emailAddr

More about MVDs

Consider studCourseGrade(<u>rollNo,courseNo</u>,grade)

Note that rollNo $\rightarrow\rightarrow$ courseNo *does not* hold here even though courseNo is a multi-valued attribute of student

If (CS05B007, CS370, A)

(CS05B007, CS376, B) appear in the data then

(CS05B007, CS376, A)

(CS05B007, CS370, B) will not appear !!

Attribute 'grade' depends on (rollNo,courseNo)

MVD's arise when two unrelated multi-valued attributes of an entity are sought to be represented together.

More about MVDs

Consider

studCourseAdvisor(rollNo,courseNo,advisor)

Note that rollNo $\rightarrow\rightarrow$ courseNo *holds* here

If (CS05B007, CS370, Dr Ravi)
(CS05B007, CS376, Dr Ravi)

appear in the data then swapping courseNo values gives rise to existing tuples only.

But, since rollNo $\rightarrow$ advisor and (rollNo, courseNo) is the key, this gets caught in checking for 2NF itself.

Alternative definition of MVDs

Consider R(X,Y,Z)

Suppose that $X \rightarrow\rightarrow Y$ and by symmetry $X \rightarrow\rightarrow Z$

Then, decomposition D = (XY, XZ) should be lossless

That is, for any instance r on R, $r = \pi_{XY}(r) * \pi_{XZ}(r)$

MVDs and 4NF

An MVD $X \rightarrow\rightarrow Y$ on scheme R is called *trivial* if either $Y \subseteq X$ or $R = X \cup Y$. Otherwise, it is called *nontrivial.*

4NF: A relation R is in 4NF if it is in BCNF and for every nontrivial MVD $X \rightarrow\rightarrow A$, X must be a superkey of R.

studCourseEmail(rollNo,courseNo,emailAddr)

is not in 4NF as

rollNo $\rightarrow\rightarrow$ courseNo and

rollNo $\rightarrow\rightarrow$ emailAddr

are both nontrivial and rollNo is not a superkey for the relation

4th Normal Form

- Any relation is in Fourth Normal Form if it is BCNF *and* any multivalued dependencies are trivial
- Eliminate non-trivial multivalued dependencies by projecting into simpler tables

Forth Normal Form (4NF) (Cont.)

- Example of a table not in 4NF:

Student	Major	Hobby
Sok	IT	Football
Sok	IT	Volleyball
Sao	IT	Football
Sao	Med	Football
Chan	IT	NULL
Puth	NULL	Football
Tith	NULL	NULL

- Key: {Student, Major, Hobby}
- MVD: Student →→ Major, Hobby

Solution: Decouple to each table contains MVD. Finally, connect each to a third table contains ***Student***.

Student
Sok
Sao
Chan
Puth
Tith

Student	Major
Sok	IT
Sao	IT
Sao	Med
Chan	IT
Puth	NULL
Tith	NULL

Student	Hobby
Sok	Football
Sok	Volleyball
Sao	Football
Chan	NULL
Puth	Football
Tith	NULL

5th Normal Form

- A relation is in 5NF if every join dependency in the relation is implied by the keys of the relation
- *Implies that relations that have been decomposed in previous NF can be recombined via natural joins to recreate the original relation.*