Relational Database Design(3)

Relational Database Design

- Introduction
- First Normal Form
- Pitfalls in Relational Database Design
- Functional Dependencies
- Decomposition
- Boyce-Codd Normal Form
- Third Normal Form
- Multivalued Dependencies and Fourth Normal Form
- Overall Database Design Process

Third Normal Form: Motivation

- There are some situations where
  - ***BCNF*** is not dependency preserving, and
  - ***efficient checking for FD violation on updates is important***

Third Normal Form: Motivation

Solution: define a weaker normal form, called 3NF.

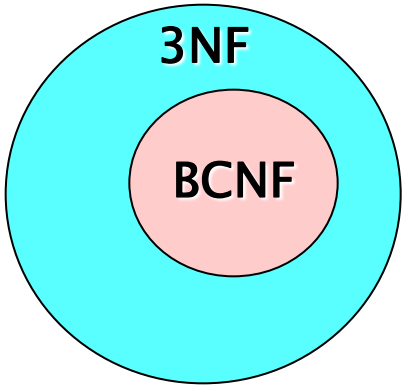
- Allows some redundancy (with resultant problems; we will see examples later)
- But FDs can be checked on individual relations without computing a join.
- ***There is always a lossless-join, dependency-preserving decomposition into 3NF.***

Algorithm

Third Normal Form (revisited)

- A relation schema *R* is in 3NF if for all:

$$X \rightarrow Y \text{ in } F^+$$



at least one of the following holds:

BCNF

1. $X \rightarrow Y$ is trivial (i.e., $Y \in X$) or
2. **X is a superkey for *R* *or***
3. If X is not a super key then,
   for each attribute *A* in ***Y***, A $\subseteq$ a candidate **key** for *R.*
   Key → X → K1 ***Reflexive, not transitive FD***

(NOTE: each attribute may be in a different candidate key)

- Third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later).

Third Normal Form (revisited)

$X \rightarrow Y$ in F^+

If X is not a super key then,

for each attribute *A* in ***Y***, A $\subseteq$ a candidate **key** for *R.*

Key → X → K1 : *X is not a key, but any key→X*

Key → K1 : *k1 in Key so it is reflexive, not transitive*

it is trivial , always true

Ex. Fname+Lname → Lname

(NOTE: each attribute may be in a different candidate key)

Enrollment

Registration number: 205

Student : Mr. John Smith ID: 41010703
Department : Computer Science

CourseNo	Course Name	Unit	Section
729101	Accounting	4	700
729111	Finance	3	711
999211	Database	3	713

Total Units: 10
Total amount :5000

Not BCNF but in 3NF

StdSSN	RegistNo	CourseNo	Section
41010703	205	729101	700
41010703	205	**729111**	711
41010703	205	999211	713
41010943	**1368**	**729111**	712
41010943	1368	999211	711
41010943	1368	729104	700
41012147	**1684**	729111	711
41012147	**1684**	999211	713

CK (StdSSN, CourseNo) → RegistNo, Section

CK (RegistNo, CourseNo) → StdSSN, Section

Not a key

StdSSN → RegistNo

RegistNo → StdSSN ⊂ CK

CK (StdSSN, CourseNo) → RegistNo, Section

RegistNo → StdSSN

Not a key

⊂ CK

(StdSSN, CourseNo)→ RegistFormNo→ StdSSN

Hence (StdSSN, CourseNo) → StdSSN

reflexive, always true

3NF

allows a transitive relationship such that

key → y → k1

a part of key

Canonical Cover

- Sets of FD may have redundant dependencies that can be inferred from the others
  - Eg: $A \rightarrow C$ is redundant in:

    $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$

$A \rightarrow C$ can be inferred from $\{A \rightarrow B, B \rightarrow C\}$

$\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$ equivalent to

$\{A \rightarrow B, \quad B \rightarrow C\}$

Canonical Cover

- Parts of a FD may be redundant

On RHS: $\{A \rightarrow B, B \rightarrow C, A \rightarrow CD\}$

can be simplified to

$\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

because

$A \rightarrow CD$ means $A \rightarrow C$ and $A \rightarrow D$

Canonical Cover

- Parts of a FD may be redundant

On LHS: $\{A \rightarrow B, B \rightarrow C, AC \rightarrow D\}$

can be simplified to

$\{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

because

$A \rightarrow C$ hence we can drop C on LHS of $AC \rightarrow D$

Ex. (ID, Name) → Telephone equi to

ID → Telephone

because ID → Name

Canonical Cover

- canonical cover of F (Fc) is a "**minimal**" set of FDs equivalent to F, having no redundant dependencies or redundant parts of dependencies

Fc is equivalent to F

3NF Decomposition Algorithm

Find a canonical cover (Fc) for *F;*

Rc ={ }
for each functional dependency **X→*Y*** in *Fc* **do**
{ create a schema XY if *XY* ⊄ any R_i
Rc = Rc U {XY} **}**

if none of the schemas in Rc
contains **a candidate key** for *R*
then Rc = Rc U {CK}
where CK is any candidate key for R

return *Rc*

3NF Decomposition Algorithm (Cont.)

- **Above algorithm ensures:**
  - each relation schema R_i is in 3NF
  - decomposition is dependency preserving and lossless-join

Example

- Relation schema:

  (A, B, C, D)

- The functional dependencies for this relation schema are:

$$C \rightarrow A, D$$
$$A, B \rightarrow C$$

Not in 3NF

Has transitive FD

- The key is: **{*A* ,*B*}**

Applying 3NF to *Banker-info-schema*

$Fc = \{ C \rightarrow (A,D) , (A,B) \rightarrow C \}$

- In the **for** loop, we add :

  R1 = (*A*, *C*, *D*)
  R2 = (*A*,*B*, *C*)

- Since *R2* contains a candidate key for *R1*, we are done with the decomposition process.
- Return Rc = {R1,R2}

3NF Decomposition Algorithm

1. Find keys,and check if the relation is already in3NF
2. Decompose the right hand sides of the dependencies

   $X \rightarrow YZ$ ==> $X \rightarrow Y, X \rightarrow Z$
3. Remove redundant attributes on the left hand sides

   $XY \rightarrow Y$ ==> $X \rightarrow Y$
4. *Remove redundant functional dependencies*
5. Combine dependencies with same left hand sides
6. Create a relation for each functional dependency
7. Remove relations contained in other relations
8. If no relation contains a key of the original relation, add a relation whose attributes form such a key

1.Finding the Keys

R= { A,B,C,D,E,F,G }

(A $\rightarrow$ BC), (B $\rightarrow$ C), (AB $\rightarrow$ D),

(EF $\rightarrow$ G), (G $\rightarrow$ F)

L = attributes appearing only on left sides = AE

R = attributes appearing only on right sides = CD

N = attributes not appearing at all = { }

Every key must include L and N

Every key must be disjoint from R

Hence, {A,E} $\subseteq$ key $\subseteq$ { A,E,B,F,G }

1.Finding the Keys (cont'd)

$\{ A,E \} \subseteq \textbf{key} \subseteq \{ A,E,B,F,G \}$ *appearing on left sides*

$(A \rightarrow BC), (B \rightarrow C), (AB \rightarrow D),$

$(EF \rightarrow G), (G \rightarrow F)$

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

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

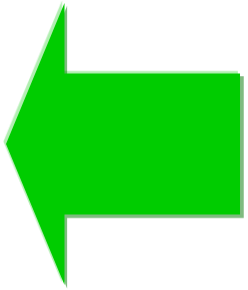
$\{ A,E,G \}^{+} = A,E,B,C,D,G,F = R$

Hence, keys are {A,E,F } and {A,E,G }

2. Decomposing right hand sides

- Decompose the right hand sides of the functional dependencies, e.g.,
  - X→YZ becomes X→Y and X→Z.
- This gives us:

A –> B
A –> C
B –> C
AB –> D
EF –> G
G –> F



(A→BC),
(B→C),
(AB→D),
(EF→G),
(G→F)

3.Remove redundant attributes

(A → BC), (B → C), (AB → D), (EF → G), (G → F)

- **Remove redundant attributes from the left sides**
- Check (**AB –> D**) for redundancy on left:

  *Since **B+** = C; A is needed to obtain D*

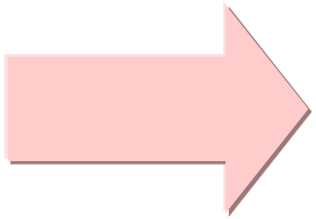
  *Since **A+** = ABCD; A suffices for D*

  *Hence, we can replace **AB–>D** with(A –>D)*

- Check (**EF -> D**) for redundancy on left:
  - Both E and F are needed to get D, so the rule cannot be replaced

The Simplified Rules

1. A –> B
2. A –> C
3. B –> C
4. AB –> D
5. EF –> G
6. G –> F



1. A -> B
2. A -> C
3. B -> C
4. A -> D
5. EF -> G
6. G -> F

4.Remove the Redundant Rules

- (A –> B), (A –> C), (B –> C), (A –> D), (EF –> G), (G –> F)
- Is (A –> B) redundant?
  - Compute A+ without using this rule :
    - A+ = ACD
    - Since B not in ACD, this rule is not redundant
- Is (A –> C) redundant
  - Compute A+ without using this rule:
    - A+ = ABCD
    - Redundant, since C is in A+
- Remove the Redundant Rules (A –> C).

4.Remove the Redundant Rules(Cont.)

~~(A –> B), (A –> C),~~ (B –> C), (A –> D),(EF –> G), (G –> F)

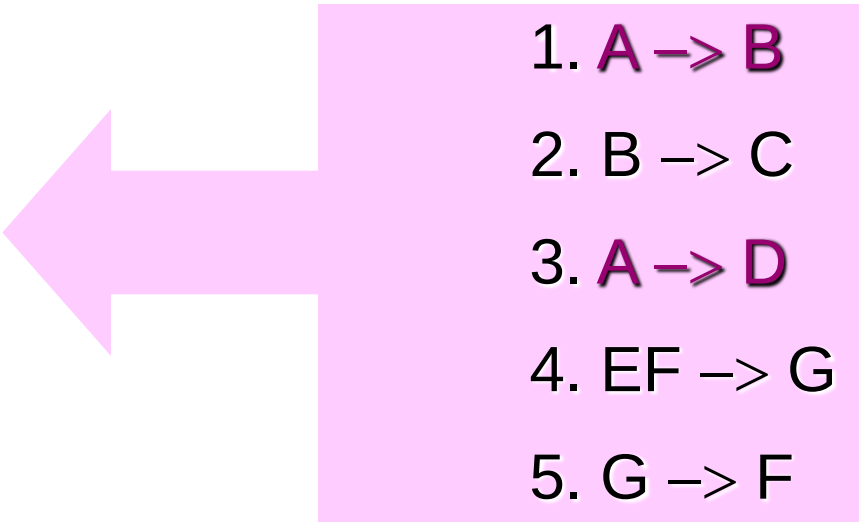
- Is (B -> C) redundant? Compute B+ :
  - B+ = B
  - Hence, not redundant
- Similarly, rules (A –> D),(EF –> G), and(G –> F) are not redundant
  - Resulting rules, after renumbering:

1. A -> B
2. B -> C
3. A -> D
4. EF -> G
5. G -> F

5.Combining Functional Dependencies

- Combine all dependencies with the same left side

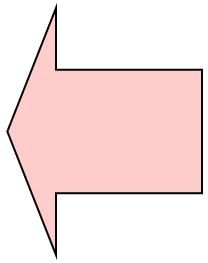
- This gives:
  1. A –> BD
  2. B –> C
  3. EF –> G
  4. G –> F



6.Creating Relations

- Create a relation for each dependency
- This gives:

1. ABD
2. BC
3. EFG
4. GF

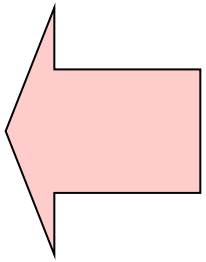


1. A –> BD
2. B –> C
3. EF –> G
4. G –> F

7.Remove Redundant Relations

- Remove relations contained in other relations
- This gives:

1. ABD
2. BC
3. EFG



1. ABD
2. BC
3. EFG
4. GF

subset

8. Assuring Storage of a Global Key

- If no relation contains a key of the original relation, then add a relation whose attributes form such a key
- None of the relations contains **AEF or AEG (keys)**
  - Hence, one of these has to be added
- Two choices for the final decomposition into 3NF:

1. ABD, BC, EFG, AEF

2. ABD, BC, EFG, AEG

Why was the Global Key Needed?

- Consider R(A,B,X,Y), with (A –> X) and (B –> Y)
- The global key is AB
- Hence, the decomposition AX, BY is not lossless join
  - AX ∩ BY = { }

- But AX, BY, AB is in 3NF and lossless join+ **dependency preserving**
  - AX ∩ AB = A : A->AX,
  - AB ∩ BY = B : B->BY

AB, the key, joins all tables together

Comparison of BCNF and 3NF

- It is always possible to decompose a relation into relations in 3NF and
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into relations in BCNF and
  - the decomposition is lossless
  - *it may not be possible to preserve dependencies.*

3NF (Cont.)

- **Example**
  - ***R* = (*A*, *B*, *C*)**
    ***F* = {*AB* → *C*, *C* → *B*}**
  - Two candidate Beys: *AB* and *AC*
  - ***R* is in 3NF**

    AB → *C* — *AB* is a superkey
    C → *B* — *B* is contained in a candidate key
    - **BCNF decomposition has (AC) and (CB)**
      - Testing for *AB* → *C* requires a join
- There is some redundancy in this schema
- Equivalent to example in book:

  Banker-schema = (branch-name, customer-name, banker-name)

  banker-name → branch name

  branch name ,customer-name → banker-name

Design Goals

- **Goal for a relational database design is:**
  - **BCNF.**
  - **Lossless join.**
  - **Dependency preservation.**
- If we cannot achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy due to use of 3NF
- Interestingly, **SQL does not provide a direct way of specifying functional dependencies other than superkeys**.

  Can specify FDs using assertions, but they are expensive to test
- Even if we had a dependency preserving decomposition, using SQL **we would not be able to efficiently test a functional dependency whose left hand side is not a key.**

Multivalued Dependencies

- Consider a database

  classes(course, teacher, book)

- ***Teachers and books assigned in each course***

  1 course - many teachers (course->>teacher)

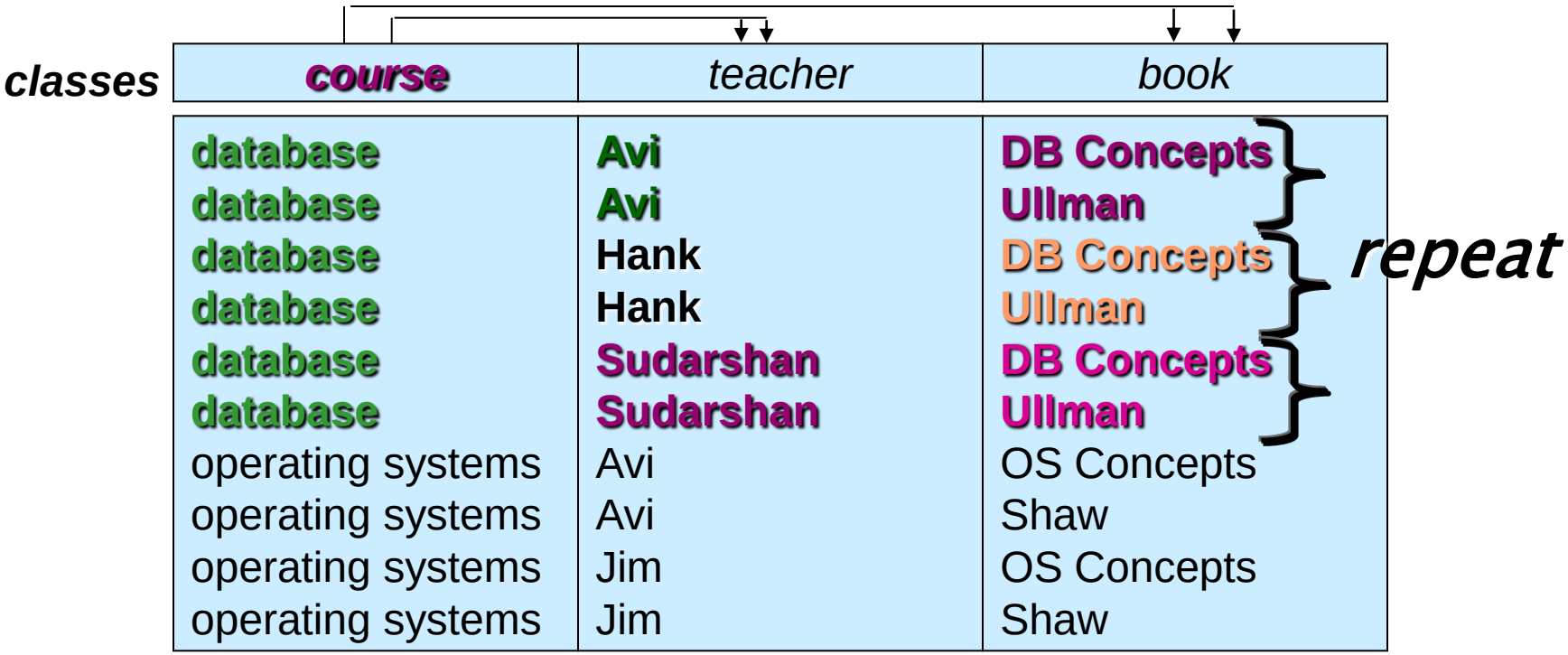
  1 course - many books (course->>book)

  No relation between teachers and books

- **no non-trivial functional dependencies** and therefore the relation is in **BCNF**

Uniquely determine

Multivalued Dependencies (Cont.)

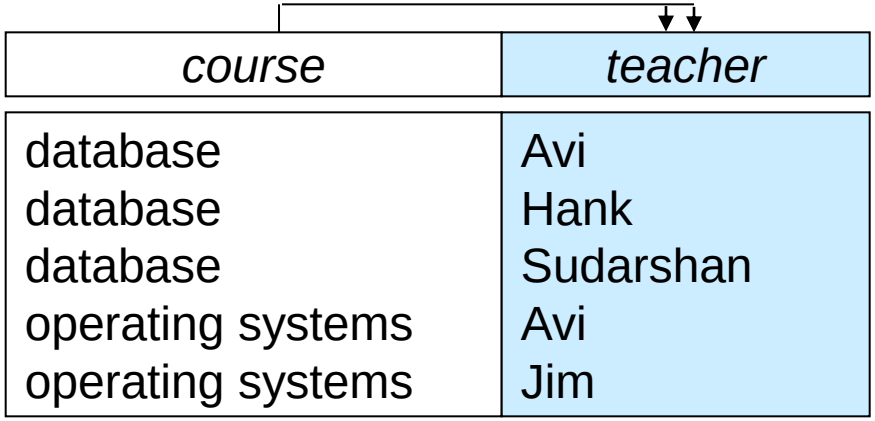


- BCNF but there are redundant data.
- Insertion anomalies – i.e., if Sara is a new teacher that can teach database, two tuples need to be inserted

  (database, Sara, DB Concepts)
  (database, Sara, Ullman)

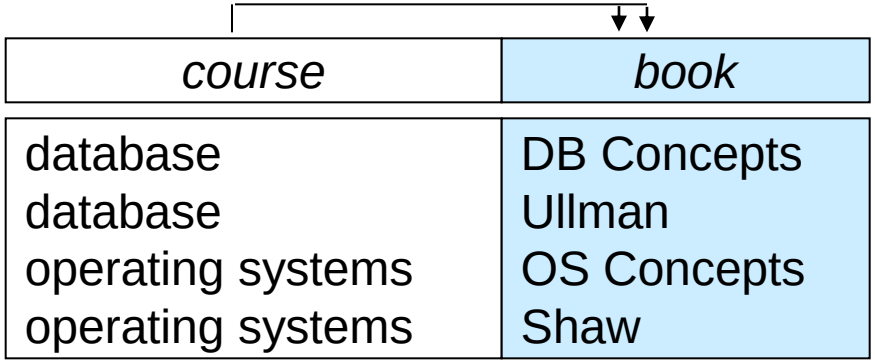
Multivalued Dependencies (Cont.)

- Therefore, it is better to decompose *classes* into:



course	*teacher*
database	Avi
database	Hank
database	Sudarshan
operating systems	Avi
operating systems	Jim

teaches



course	*book*
database	DB Concepts
database	Ullman
operating systems	OS Concepts
operating systems	Shaw

text

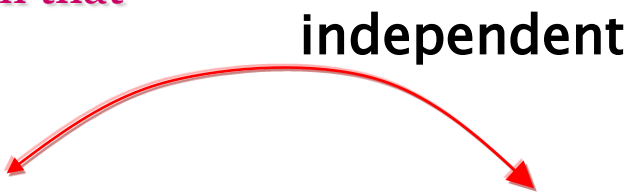
We shall see that these two relations are in **Fourth Normal Form (4NF)**

Multivalued Dependencies

- The concept of multivalued dependencies was developed to provide a basis for decomposition of relations like the classes scheme.
- Definition
  - ***X ->> Y|Z*** is said to hold for *R(X, Y, Z)* if *t1* and *t2* are given by

    t1 = [X, Y1, Z1], and
    t2 = [X, Y2, Z2]
  - then there must be tuples *t3* and *t4* such that

    t3 = [X, Y1, Z2], and
    t4 = [X, Y2, Z1]



course	teacher	book
database	Avi	DB Concepts
database	Sara	Ullman
database	Avi	Ullman
database	Sara	DB Concepts

MVD

- Dependency between attributes (for example, A, B, and C) in a relation, such that for each value of A there is a set of values for B and a set of values for C. However, set of values for B and C are independent of each other.

Fourth Normal Form (4NF)

- **MVD between attributes A, B, and C in a relation using the following notation:**

  A ->> B

  A ->> C

Fourth Normal Form (4NF)

- **Defined as a relation that is in BCNF and contains no nontrivial MVDs.**

No MVDs

Fourth Normal Form

- A relation schema *R* is in 4NF with respect to a set *D* if for all MVD in D^+ of the form $X \rightarrow\rightarrow Y$,

  where $X \subseteq R$ and $Y \subseteq R$, ***at least one of the following hold***:

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

- **If a relation is in 4NF it is also in BCNF.**

  for a relation with at least three attributes having multi-value dependency

no MVDs or $Y \cup X = R$

4NF Decomposition Algorithm

Similar to BCNF algorithm

result: = {*R*};
done := false;
compute D^+;
Let D_i denote the restriction of D^+ to R_i

while (**not** *done*)
  if (there is a schema $\mathbf{R}_i$ in *result* that is not in 4NF) **then**
    begin
      let $Y \rightarrow\rightarrow X$ be a nontrivial multivalued dependency that holds
        on R_i such that $Y \rightarrow\rightarrow R_i$ is not in D_i, and $Y \cap X = \phi$;
      result := (*result* - R_i) $\cup$ (R_i - X) $\cup$ (Y, X); *{Split into 2 schemas}*
    end
  else *done*:= true;

Note: each R_i is in 4NF, and decomposition is lossless-join

Example

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

  $F = \{ A \twoheadrightarrow B$

  $B \twoheadrightarrow HI$

  $CG \twoheadrightarrow H \}$

- R is not in 4NF since $A \twoheadrightarrow B$ and A is not a super key for R
- Decomposition

  a) $R_1 = (A, B)$ (R_1 is in 4NF; X U Y = R_1)

  b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF; CG→→H)

  c) $R_3 = (C, G, H)$ (R_3 is in 4NF)

  d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF)

- Since $A \twoheadrightarrow B$ and $B \twoheadrightarrow HI$, hence $A \twoheadrightarrow HI$, $A \twoheadrightarrow I$

  e) $R_5 = (A, I)$ (R_5 is in 4NF)

  f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

ริเลชันอาจารย์สอน-อาจารย์ที่ปรึกษา

TeacherID	TcourseNo	SID
Q1011	999211	41010943
Q1011	999211	41013780
Q1035	729101	41012451
Q1	729104	41012451
	999211	41012451
	729111	41010703
	729111	41013327
	729103	41012147

No relationship between course teaching and students

TeacherID->> TcourseNo
TeacherID->> SID

X->>Y and $X \cup Y = R$

ริเลชันรหัสอาจารย์-รหัสชุดวิชา

TeacherID	TcourseNo
Q1011	999211
Q1035	729101
Q1035	729104
Q1035	999211
Q1059	729111
Q1061	729103

ริเลชันรหัสอาจารย์-รหัสนักศึกษา

TeacherID	SID
Q1011	41010943
Q1011	41013780
Q1035	41012451
Q1059	41010703
Q1059	41013327
Q1061	41012147

Fifth Normal Form

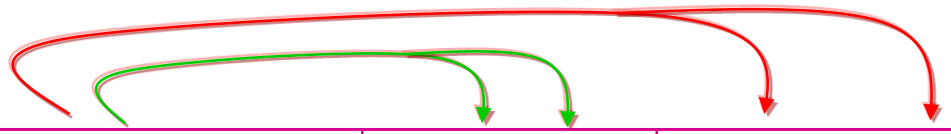
- The aim of is to have relations that cannot be decomposed further.
- A relation *R* satisfies ***join dependency*** ***(R_1, R_2, ..., R_n)*** *if and only if R is equal to the join of* $R_1, R_2, ..., R_n$

  where $R_i \subseteq R$.

- *A relation R is in* ***5NF*** *(or project-join normal form, PJNF) if for all join dependencies, at least one of the following holds.*
  - ***(R_1, R_2, ..., R_n)*** *is a trivial join-dependency (that is, one of* R_i *is R)* ***or***
  - *Every* R_i *is a candidate key for R.*

schema contains only key , can not be decomposed further

5NF



dept	*subject*	*student*
Comp. Sc.	*CP1000*	*John Smith*
Mathematics	*MA1000*	*John Smith*
Comp. Sc.	*CP2000*	*Arun Kumar*
Comp. Sc.	*CP3000*	*Reena Rani*
Physics	*PH1000*	*Raymond Chew*
Chemistry	*CH2000*	*Albert Garcia*

not 5NF

no MVDs since subject and student are dependent;

5NF

(dept, subject)
(dept, student)
(subject, student)

5NF

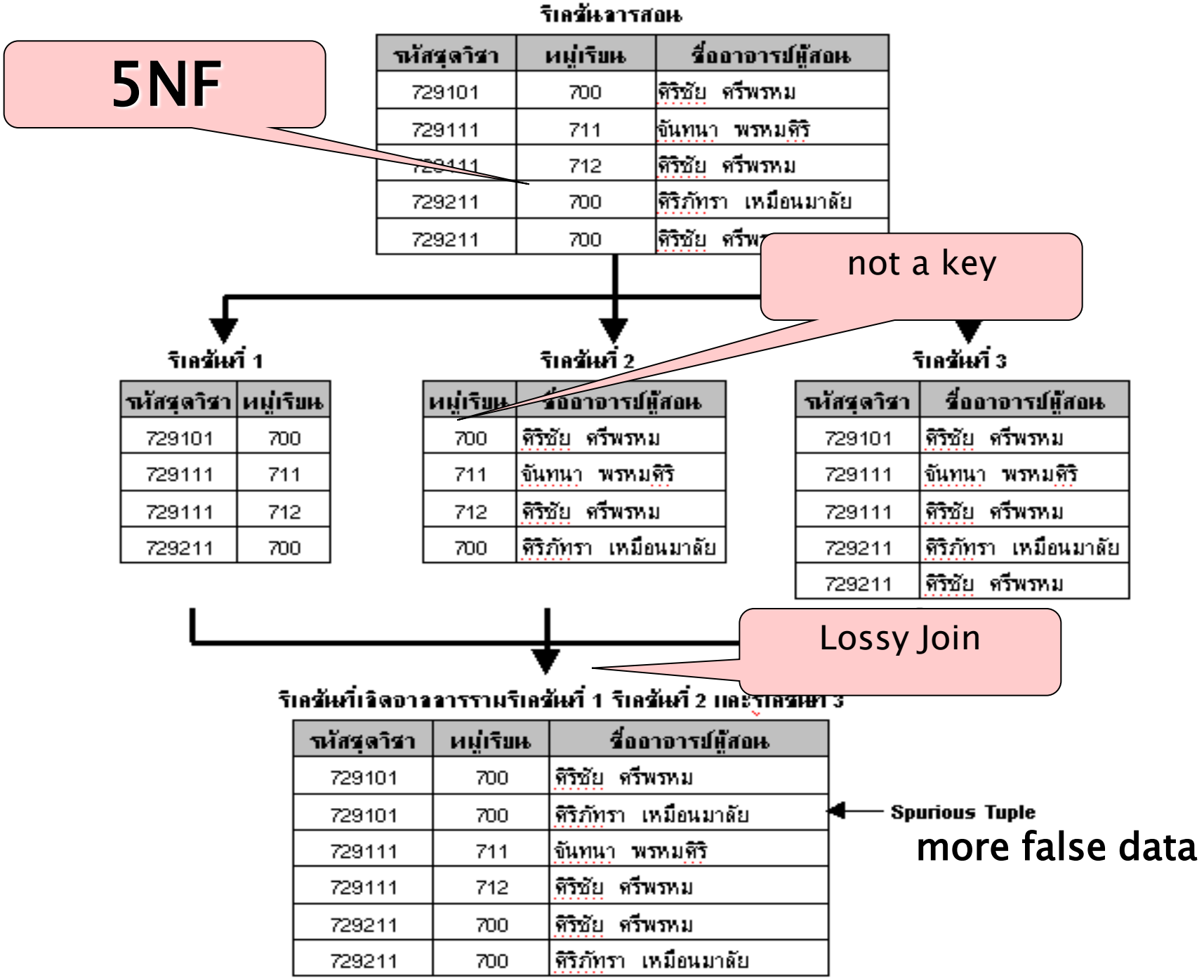
Joining these 3 tables gets the original table

Dept	Subject
Comp. Sc.	CP1000
Mathematics	MA1000
Comp. Sc.	CP2000
Comp. Sc.	CP3000
Physics	PH1000
Chemistry	CH2000

Dept	Student
Comp. Sc.	John Smith
Mathematics	John Smith
Comp. Sc.	Arun Kumar
Comp. Sc.	Reena Rani
Physics	Raymond Chew
Chemistry	Albert Garcia

(dept, subject)
(dept, student)
(subject, student)

Subject	Student
CP1000	John Smith
MA1000	John Smith
CP2000	Arun Kumar
CP3000	Reena Rani
PH1000	Raymond Chew
CH2000	Albert Garcia



รีเลชันอาจารสอน

รหัสชุดวิชา	หมู่เรียน	ชื่ออาจารย์ผู้สอน
729101	700	ศิริชัย ศรีพรหม
729111	711	จันทนา พรหมศิริ
729111	712	ศิริชัย ศรีพรหม
729211	700	ศิริภัทรา เหมือนมาลัย
729211	700	ศิริชัย ศรีพ

รีเลชันที่ 1

รหัสชุดวิชา	หมู่เรียน
729101	700
729111	711
729111	712
729211	700

รีเลชันที่ 2

หมู่เรียน	ชื่ออาจารย์ผู้สอน
700	ศิริชัย ศรีพรหม
711	จันทนา พรหมศิริ
712	ศิริชัย ศรีพรหม
700	ศิริภัทรา เหมือนมาลัย

รีเลชันที่ 3

รหัสชุดวิชา	ชื่ออาจารย์ผู้สอน
729101	ศิริชัย ศรีพรหม
729111	จันทนา พรหมศิริ
729111	ศิริชัย ศรีพรหม
729211	ศิริภัทรา เหมือนมาลัย
729211	ศิริชัย ศรีพรหม

รีเลชันที่เกิดจากการรวมรีเลชันที่ 1 รีเลชันที่ 2 และรีเลชันที่ 3

รหัสชุดวิชา	หมู่เรียน	ชื่ออาจารย์ผู้สอน
729101	700	ศิริชัย ศรีพรหม
729101	700	ศิริภัทรา เหมือนมาลัย
729111	711	จันทนา พรหมศิริ
729111	712	ศิริชัย ศรีพรหม
729211	700	ศิริชัย ศรีพรหม
729211	700	ศิริภัทรา เหมือนมาลัย

Other Design Issues

- Examples of bad database design, to be avoided:

  Instead of *earnings*(*company-id, year, amount*), use

  - *earnings-2000, earnings-2001, earnings-2002*, etc., all on the schema (*company-id, earnings*).
    - Above are in BCNF, but make querying across years difficult and needs new table each year
  - *company-year*(*company-id, earnings-2000, earnings-2001, earnings-2002*)
    - Also in BCNF, but also makes querying across years difficult and requires new attribute each year.
    - Is an example of a **crosstab**, where values for one attribute become column names
    - Used in spreadsheets, and in data analysis tools

Summarize

Ordinary Table ทำถึงแค่ 3NF , BCNF ก็ได้

Associated Table ทำ 4NF , 5NF

Question ?