Ceng 302
Database Managment Systems

Relational Database Design and Normalization-2

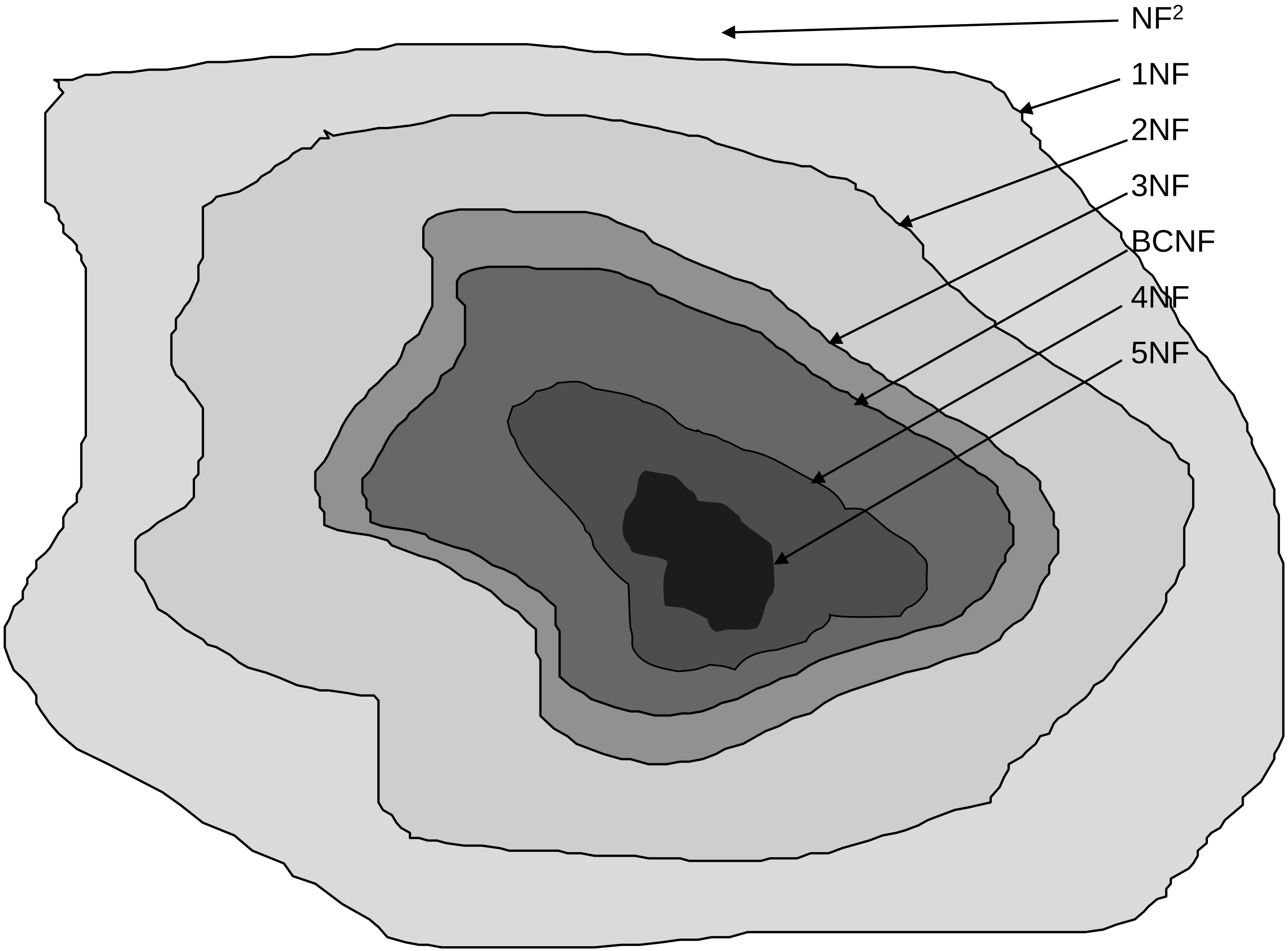
Prof. Dr. Adnan YAZICI

Department of Computer Engineering,

Middle East Technical University

(**Fall 2021**)

Overview of Normal Forms



First Normal Form

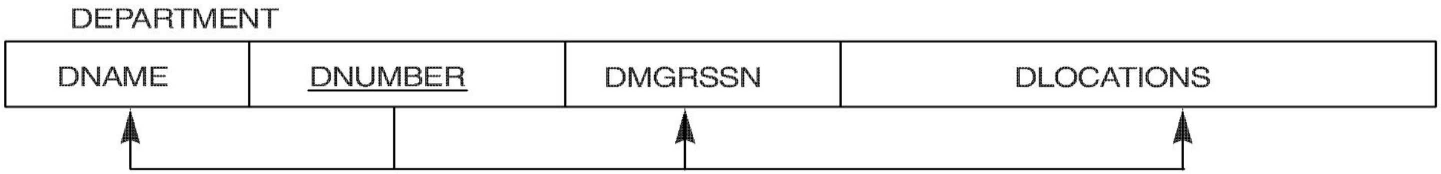
- Disallows composite attributes, multivalued attributes, and **nested relations**; attributes whose values *for an individual tuple* are non-atomic

- Considered to be part of the definition of relation

Normalization into 1NF

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

(a)



(b)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(a)

EMP_PROJ

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b)

EMP_PROJ

SSN	ENAME	PNUMBER	HOURS
123456789	Smith,John B.	1	32.5
		2	7.5
666884444	Narayan,Ramesh K.	3	40.0
453453453	English,Joyce A.	1	20.0
		2	20.0
333445555	Wong,Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya,Alicia J.	30	30.0
		10	10.0
987987987	Jabbar,Ahmad V.	10	35.0
		30	5.0
987654321	Wallace,Jennifer S.	30	20.0
		20	15.0
888665555	Borg,James E.	20	null

(c)

EMP_PROJ1

SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

The Schema (Intention) of the *Department* Relation

Department

Dept No	Dept Name	Instructors		Students				Courses		
		Name	SSN	SNo	Name	Contact-Person		CNo	Pre-Requisites	Categ
						Name	TelNum			
71	Computer Science	Dr. C.Bee	9078913	10364	Ann Adams	Clark Adams	0.365.3871733	71120	{36151}	[Math,General]
		Prof. Z. Cruise	7897988			Betty Adams	0.365.3532233	71341	{71320,36250}	[Math, AI]
		Assoc.Prof.S.Dark	8834587	10366	Tommas Astor	Martin Astor	0.212..7654489	71348	{36152,36255}	[C-Arch,EE]
						Mary Astor	0.212..5567765			
		⋮			⋮	⋮			⋮	
		Prof.F. Twins	8765890							
		Dr. A. White	8781765	10366	Albert Walker	Jack Walker	0.312..4437795	71663	{ dne }	[IS]
36	Mathematics	Prof. H. Arikan	6578911	12155	Altar Kunter	Leyla Kunter	0.216..5552233	36145	{ dne }	[Math,General]
		Assoc.Prof.D.Bach	9123476			Mert Kunter	0.216..5552233	36330	{71120}	[Progr,NumAn]
		⋮			⋮	⋮			⋮	
		Dr.Q. McGill	9345761	13553	Melih Zobu	Murat Zobu	0.344..2213456			
⋮	⋮	⋮			⋮	⋮			⋮	

Second Normal Form

- Uses the concepts of **FDs**, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Example: {Ssn → Ename,

Pnumber → {Pname, Plocation}, {Ssn,Pnumber} → Hours}

- {SSN, PNUMBER} -> HOURS is a **full FD** since neither SSN -> HOURS nor PNUMBER -> HOURS hold.
- {SSN, PNUMBER} -> ENAME is ***not*** **a full FD** (it is called a ***partial dependency***) since SSN -> ENAME also holds

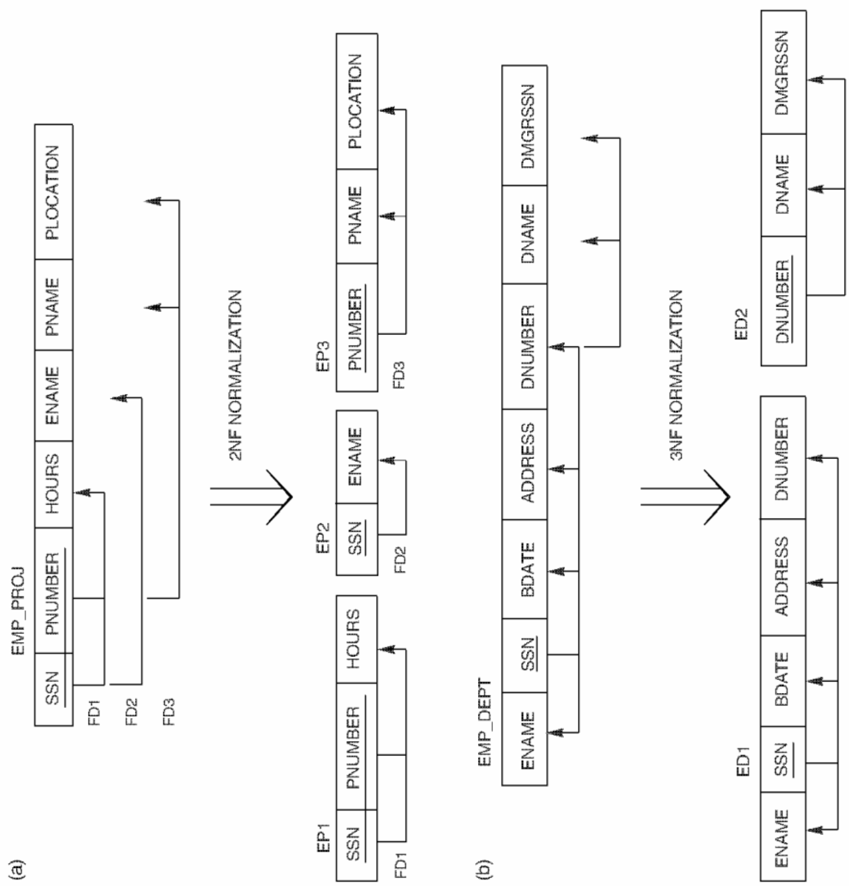
Second Normal Form

- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on the primary key
- A more general definition: A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R
- R can be decomposed into 2NF relations via the process of 2NF normalization

Third Normal Form

- A relation schema R is in **third normal form** (**3NF**) if it is **in 2NF** *and* no non-prime attribute A in R is **transitively dependent** on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization

Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



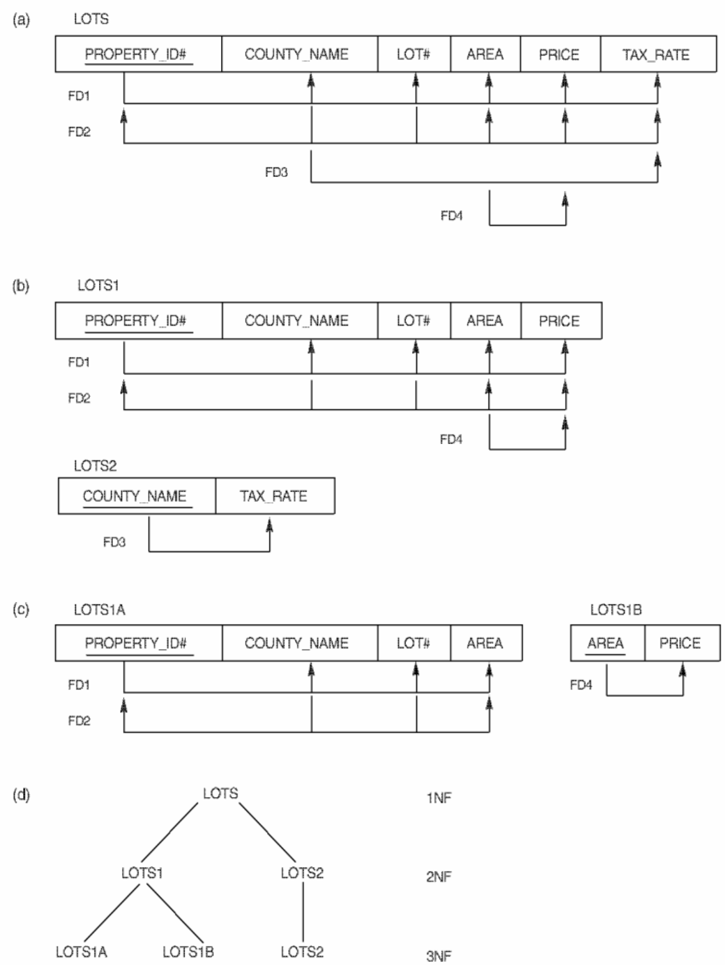


Figure 14.11 Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies fd1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a **superkey** of R (i.e. prime attributes should be dependent on the key as well.)

- The goal is to have each relation in BCNF (or 3NF)

Relation TEACH that is in 3NF but not in BCNF

Figure 14.13 A relation TEACH that is in 3NF but not in BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:
  - *fd1: { student, course} -> instructor*
  - *fd2: instructor -> course*
- {student, course} is a candidate key for this relation. So this relation is in 3NF <u>but not in</u> BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations.

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH

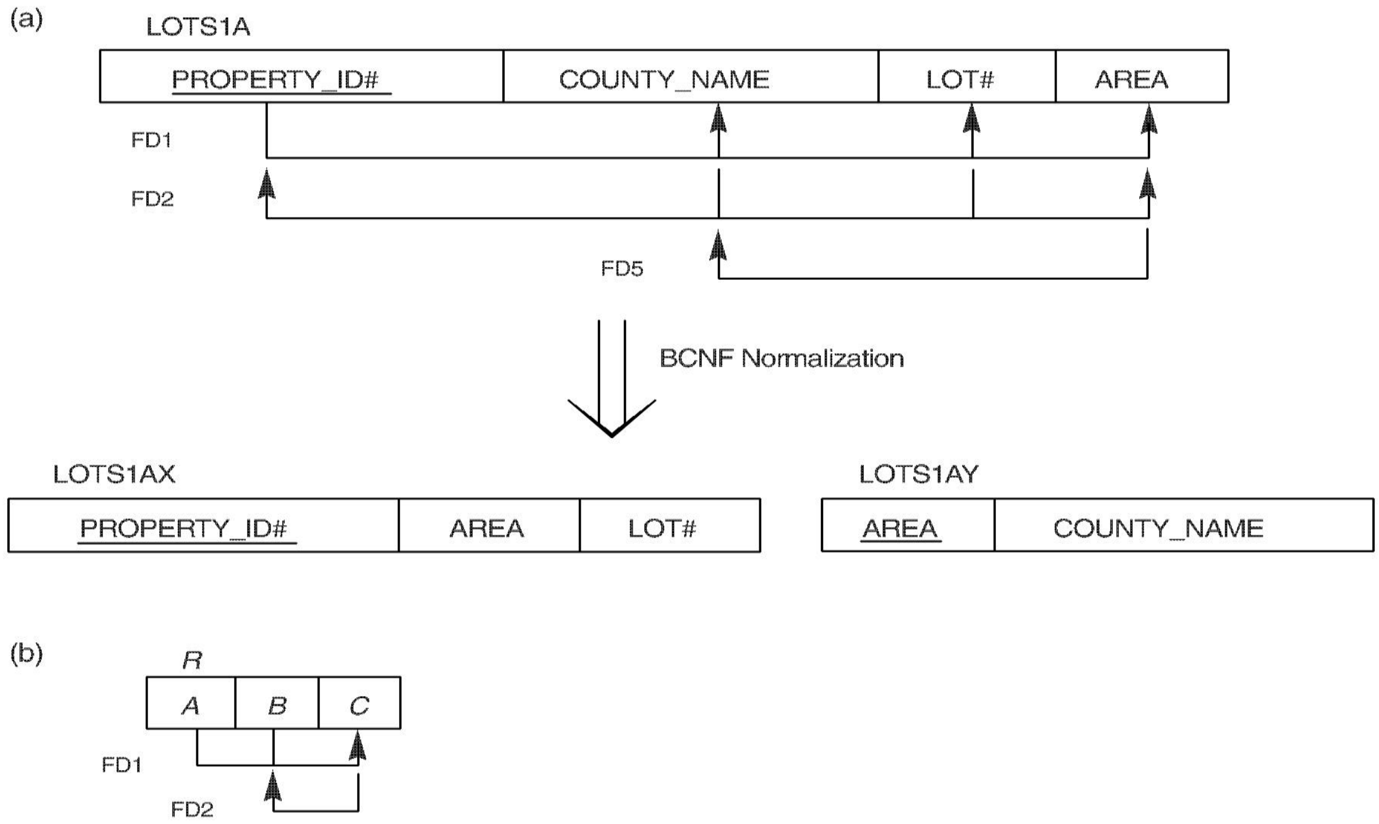
  1. {<u>student, instructor</u>} and {<u>student, course</u>}
  2. {course, <u>instructor</u> } and {<u>course, student</u>}
  3. {<u>instructor</u>, course } and {<u>instructor, student</u>}

- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we <u>cannot</u> sacrifice the non-additivity property after decomposition.
- Out of the above three, only the **3rd decomposition** will not generate spurious tuples after join.(and hence has the non-additivity (no spurious tuples) property).

fd1: { student, course} -> instructor
fd2: instructor -> course

Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation R in 3NF but not in BCNF.



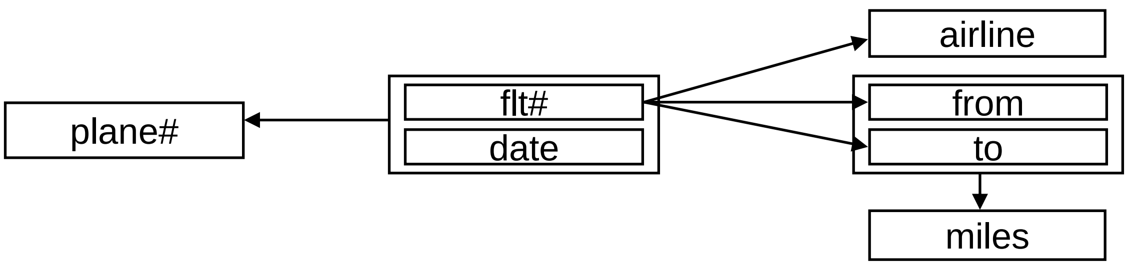
Normal Forms – all definitions

- NF^2: non-first normal form
- 1NF: R is in 1NF. iff all domain values are atomic2
- 2NF: R is in 2. NF. iff R is in 1NF and every nonkey attribute is fully dependent on the key
- 3NF: R is in 3NF iff R is 2NF and every nonkey attribute is non-transitively dependent on the key
- BCNF: R is in BCNF iff every determinant is a superkey (or candidate key)
- Determinant: an attribute on which some other attribute(s) is fully functionally dependent.

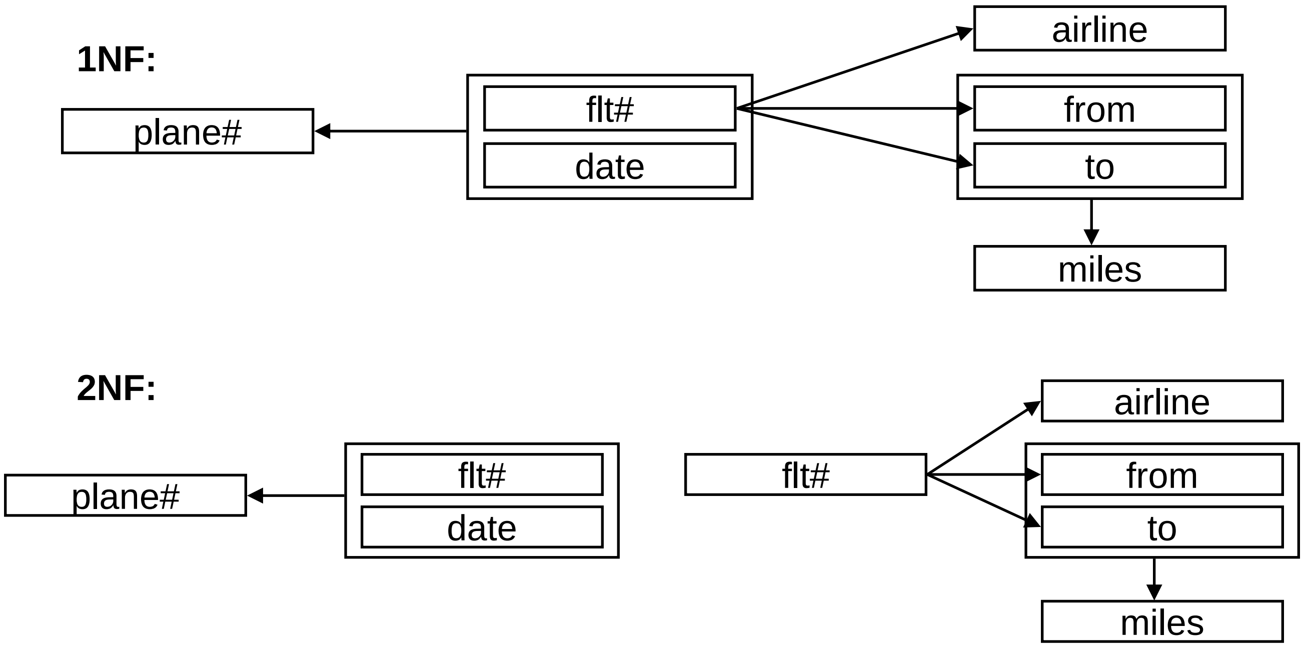
Example of Normalization

FLT-INSTANCE

flt#	date	plane#	airline	from	to	miles

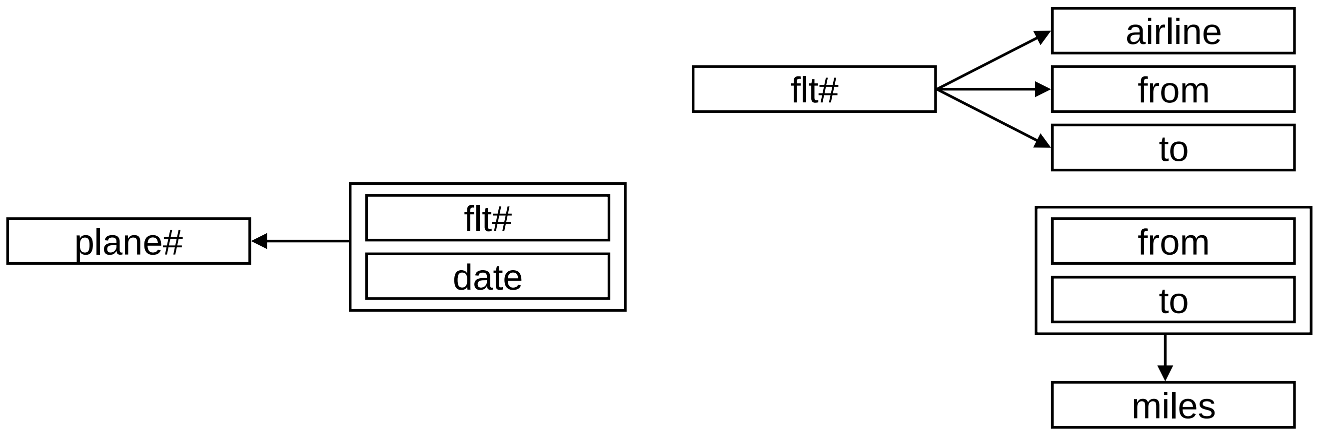


Example of Normalization

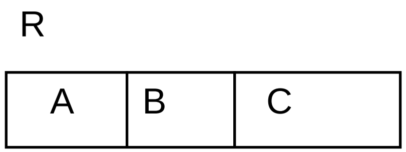


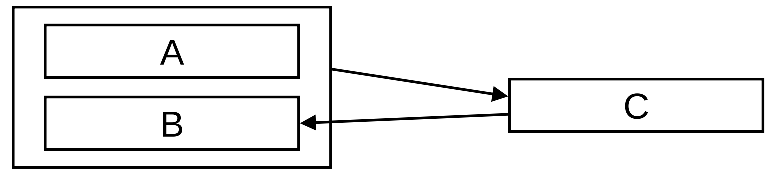
Example of Normalization

3NF & BCNF:



3NF that is not BCNF





Determinants: {A,B} and {C}

Candidate keys: {A,B} and {A,C}

A decomposition:

Lossless, but not dependency preserving ($AB \rightarrow C$ is not preserved)!

Normalization Theory

Example: Q = {S,B,D}, sailor S can reserve a boat B for at most one day D, (SB→D), and on any given day D at most one boat B can be reserved, (D→B).

That is, F = {SB→D, D→B}. So, D→B violates BCNF.

Algorithm: Relational decomposition into BCNF with nonadditive (lossless) join property

1. Decomposes a universal relational schema $R = \{A_1,\ldots,A_n\}$ into a decomposion $D = \{R_1,\ldots,R_m\}$
2. Set D = {R}.
3. While there is a relation schema Q in D that is not in BCNF do

```
{
        Choose a relation schema Q in D that is not in BCNF;
        Find a FD X→Y in Q that violates BCNF
        Replace Q in D by two relation schemas: (X ∪ Y) and (Q-Y);
};
```

Example: Q is decomposed into two relations,

$Q_1 = \{S,D\}$ and $Q_2 = \{D,B\}$ and

SB→D cannot be preserved.

Normalization Theory

How to guarantee lossless-joins

$$\boxed{R_1 \bowtie R_2 = R}$$

- Decompose relation, R, with functional dependencies, F, into relations, R_1 and R_2, with attributes, A_1 and A_2, and associate FDs, F_1 and F_2.

- The decomposition is **lossless iff**:
  - $R_1 \cap R_2 \rightarrow R_1 - R_2$ is in F^+, or
  - $R_1 \cap R_2 \rightarrow R_2 - R_1$ is in F^+

Normalization Theory

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

This relation can be decomposed in two different ways

1. $R_1 = (A, B), \quad R_2 = (B, C)$
   - Lossless-join decomposition:

     $R_1 \cap R_2 \rightarrow R_1 - R_2$ in F^+?

     $R_1 \cap R_2 = \{B\}$ and $R_1 - R_2 = \{A\}$

     $B \rightarrow$ A in not in F^+.

     But $R_1 \cap R_2 \rightarrow R_2 - R_1$ in F^+?

     $R_1 \cap R_2 = \{B\}$ and $R_2 - R_1 = \{C\}$

     Yes, it is lossless, since $B \rightarrow$ C in in F^+.
   - It is also dependency preserving.
2. *Example: R = ABC, and F={A→B, C→B},*

   R_1= (AB), R_2= (CB),

   Since AB $\cap$ CB = B and neither B→A nor B→C exists.

   Therefore, it is not a lossless join decomposition and is not dependency preserving.

Normalization Theory

How to guarantee lossless-join decomposition?

(**a)Applying the algorithm to test the decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS.**

R={SSN, ENAME, PNUMBER,PNAME, PLOCATION, HOURS}

R_1=EMP_LOCS={ENAME, PLOCATION}
R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME; PNUMBER→{PNAME, PLOCATION}; {SSN,PNUMBER} →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 functional dependencies, so they are not all ”a” . Therefore, it is not lossless join)

Normalization Theory

How to guarantee lossless-join decomposition

Applying the algorithm to the decomposition:

R={SSN, ENAME, PNUMBER,PNAME, PLOCATION, HOURS}

R_1=EMP={SSN, ENAME}
R_2 =PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→ENAME; PNUMBER→{PNAME, PLOCATION}; {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_2	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

Normalization Theory

How to guarantee lossless-join decomposition

Applying the algorithm to the decomposition:

R={SSN, ENAME, PNUMBER,PNAME, PLOCATION, HOURS}

R_1=EMP={SSN, ENAME}
R_2 =PROJ={PNUMBER, PNAME, PLOCATION}
R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→ENAME; PNUMBER→{PNAME, PLOCATION}; {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_2	a_1	$(b_{32} \rightarrow a_2)$	a_3	$(b_{34} \rightarrow a_4)$	$(b_{35} \rightarrow a_5)$	a_6

(Matrix S after the first two functional dependencies. The last row is all “a” symbols, so we stop. Therefore, it is a lossless join decomposition)

Normalization Theory

How to guarantee preservation of FDs

- Decompose relation, R, with functional dependencies, F, into relations, $R_1,\ldots, R_k$, with associated functional dependencies, $F_1,\ldots, F_k$.
- The decomposition is dependency preserving iff:

$$\boxed{F^+ = (F_1 \cup \ldots \cup F_k)^+}$$

Normalization Theory

How to guarantee preservation of FDs

$R=\{SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS\}$

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

$R_2=PROJ=\{PNUMBER, PNAME, PLOCATION\}$

$R_3=WORKS\_ON=\{SSN, PNUMBER, HOURS\}$

$F=\{SSN \rightarrow ENAME;\ PNUMBER \rightarrow \{PNAME, PLOCATION\};$
$\{SSN,PNUMBER\} \rightarrow HOURS\}$

$F_1 = \{SSN \rightarrow ENAME\}$

$F_2 = \{PNUMBER \rightarrow \{PNAME, PLOCATION\}\}$

$F_3 = \{\{SSN,PNUMBER\} \rightarrow HOURS\}$

The decomposition is dependency preserving since

$$F^+ = (F_1 \cup \ldots \cup F_k)^+$$

$F^+ = (SSN \rightarrow ENAME \cup PNUMBER \rightarrow \{PNAME, PLOCATION\} \cup \{SSN,PNUMBER\} \rightarrow HOURS)^+$

Normalization Theory

Algorithm: Relational synthesis into 3NF with dependency preserving and nonadditive (lossless) join property

1. Find a **minimal cover** G for F.
2. For each left-hand-side X of a FD that appears in G, create a relation schema in D with attributes $\{X \cup \{A_1\} \cup, \ldots, \cup \{A_k\}\}$, where $X \rightarrow A_1, \ldots, X \rightarrow A_k$ are the only dependencies in G with X as left-hand-side (X is the key of this relation).
3. If none of the relation schemas in D contains a key of R, then **create one more relation schema** in D that contains attributes that **form a key of R**.

Normalization Theory

Example: R = ABC, and F={A→B, C→B},

- Key is AC.
- When we use standart process of repeated decomposition, we obtain the following:

  R_1=AB, R_2=CB,

  Since AB $\cap$ CB → R_1 - R_2 → B and B→A or B→C is not in F+.

  So, it is not lossless.
- Notice that the key (AC) is not included in any of the relation.
- Therefore, we create R_3=AC along with R_1=AB and R_2=CB.
- Now, the decomposition is dependency preserving and lossless-join of R.
- We obtain this result through a process of synthesis rather than through a process of repeated decomposition.

Normalization Theory

Algorithm: Relational decomposition into BCNF with nonadditive (lossless) join property

Example: Contracts (contractid, supplierid, projid, deptid, partid, qty, value).

That is, R = CSJDPQV, and F={C→SJDPQV, JP→C, SD→P, J→S}

- Candidate keys = C, JP, SDJ
- C→P is implied by C→S, C→D, and SD→P, so we can delete C→P.

  C→S is implied by C→J and J→S; so we can delete C→S.

  Therefore,

  F_{min} = {C→D C→J, C→Q, C→V, JP→C, SD→P, J→S}.
- Since we have the constraint that each project deals with a single supplier: because of J→S, R= CSJDQV is not in BCNF.
- We decompose R as: R_1= CJDQV, R_2= SDP, R_3= JS,
- However, this decomposition is not dependency preserving, since JP→C cannot be enforced without a join.
- One way to deal with this is to add R_4=CJP, which amounts to storing some information redundantly to make the dependency enforcement cheaper.
- We obtain this result through a process of synthesis rather than through a process of repeated decomposition.

Multivalued Dependencies: A New Form of Redundancy

- Multivalued dependencies (MVD's) express a condition among tuples of a relation that exists when the relation is trying to represent more than one many-many relationship.

- Then certain attributes become independent of one another, and their values must appear in all combinations.

Multivalued Dependencies: A New Form of Redundancy

EMP

Ename	Proj-name	Dep-name
Smith	{Y,Z}	{Anna, John}
Suzan	{X,Z}	{Ali, Aigerim}

- This relation represents two independent 1:N relationships, one between *employees* and *projects* and the other between *employees* and their *dependents*.

Multivalued Dependencies (MVDs)

- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$.
- The multivalued dependency

$$\alpha \rightarrow\rightarrow \beta$$

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

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

Multivalued Dependencies (MVDs)

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

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

- Note that since the behavior of **β** and **R-α-β** are identical it follows that

$\alpha \rightarrow\rightarrow \beta$ **if α** $\rightarrow\rightarrow$ R- α- β

Multivalued Dependencies (MVDs)

Example-1: ***course* $\rightarrow\rightarrow$ *teacher*,**
course* $\rightarrow\rightarrow$ *book

Or ***course* $\rightarrow\rightarrow$ *book / teacher***

- The above formal definition is supposed to formalize the notion that given a particular value of *course* it has associated with it a set of values of *teacher* and a set of values of *book*, and these two sets are in some sense independent of each other.
- Note:
  - If $Y \rightarrow Z$ then $Y \rightarrow\rightarrow Z$
  - Indeed, we have (in above notation) $Z_1 = Z_2$.

Example-2:

***ename*$\rightarrow\rightarrow$ *project*,**
***ename*$\rightarrow\rightarrow$ *dependents*,**
ename*$\rightarrow\rightarrow$ *project/dependents

Armstrong's inference rules for MVDs

Rules of the computation:

- Reflexivity for FDs: if $Y \subseteq X$, then $X \rightarrow Y$
- Augmentation for FDs: if $X \rightarrow Y$, then $WX \rightarrow WY$
- Transitivity for FDs: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- Complementation rule for MVDs): if $X \rightarrow\rightarrow Y$, then $X \rightarrow\rightarrow (R-(X \cup Y)$.
- Augmentation rule for MVDs: if $X \rightarrow\rightarrow Y$ and $W \subseteq Z$, then $WX \rightarrow\rightarrow ZY$.
- Transitive rule for MVDs: if $X \rightarrow\rightarrow Y$ and $Y \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow (Z-Y)$.
- Intersection rule for MVDs: if $X \rightarrow\rightarrow Y$ and $X \rightarrow\rightarrow Z$, then $X \rightarrow\rightarrow Y \cap Z$.
- Replication rule for FDs and MVDs: If $X \rightarrow Y$, then $X \rightarrow\rightarrow Y$.

- sound (generate only dependencies that actually hold) and
- complete (generate all dependencies that hold).

Use of Multivalued Dependencies

- We use multivalued dependencies in two ways:
  1. To test relations to determine whether they are legal under a given set of functional and multivalued dependencies.
  2. To specify constraints on the set of legal relations. We shall thus concern ourselves *only* with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation *r* fails to satisfy a given MVD, we can construct a relation r' that does satisfy the MVD by adding tuples to *r*.

4NF Definition

- A relation R is in 4NF if whenever $X \rightarrow\rightarrow Y$ is a nontrivial MVD, then X is a superkey.
  - "Nontrivial" means that:
    1. Y is not a subset of X, and
    2. X and Y are not, together, all the attributes.
  - Note that the definition of "superkey" still depends on FD's only.

4NF (Fourth Normal Form)

Example: **ename →→ project/dependents**

ename	project	dependents
Ali	X	Ayşe
Ali	Y	Ayşe
Ali	Y	Hasan
Ali	X	Hasan
Emin	W	Sami
Emin	X	Sami
Emin	Y	Sami
Emin	Z	Sami
Emin	W	Selim
Emin	X	Selim
Emin	Y	Selim
Emin	Z	Selim
Emin	W	Aysun
Emin	X	Aysun
Emin	Y	Aysun
Emin	Z	Aysun

4NF normalization →

ename →→ project

ename	project
Ali	X
Ali	Y
Emin	W
Emin	X
Emin	Y
Emin	Z

ename	dependents
Ali	Ayşe
Ali	Hasan
Emin	Sami
Emin	Selim
Emin	Aysun

ename →→ dependents

4NF

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

$$F = \{ A \rightarrow\rightarrow B,\ B \rightarrow\rightarrow HI,\ CG \rightarrow\rightarrow H \}$$

- R is not in 4NF since $A \rightarrow\rightarrow B$ is not trivial, and A is not a superkey for R
- Decomposition

a) $R_1 = (A,B)$ (R_1 is in 4NF)

b) $R_2 = (A,C,G,H,I)$ (R_2 is not in 4NF, since $CG \rightarrow\rightarrow H$)

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

d) $R_4 = (A,C,G,I)$ (R_4 is not in 4NF, by $A \rightarrow\rightarrow HI\text{-}B$, *so* $A \rightarrow\rightarrow HI$, *and then* $A \rightarrow\rightarrow H$ *and* $A \rightarrow\rightarrow I$)

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

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

BCNF Versus 4NF

- Remember that every FD $X \rightarrow Y$ is also an MVD, $X \rightarrow\rightarrow Y$.
- Thus, if R is in 4NF, it is certainly in BCNF.
  - Because any BCNF violation is a 4NF violation.
- But R could be in BCNF and not in 4NF, because MVD's are "invisible" to BCNF.

- Note: If a relation schema is in BCNF and at least one of its keys consists of a single attribute, then it is also in 4NF.

4NF

Example: Drinkers(<u>name</u>, addr, <u>phones</u>, <u>beersLiked</u>)

FD: name → addr

MVD's: name →→ phones

name →→ beersLiked

- Key is {name, phones, beersLiked}.
- Therefore, all dependencies violate 4NF.

Conclusion

Designing a database schema

- Usually many designs possible
- Some are (much) better than others!
- How do we choose?

❖ Very nice theory for relational database design

- Normal forms – "good" relations
- Design by decomposition
- Usually intuitive and works well
- Some shortcomings
  - Dependency enforcement ✓
  - Query workload ✓
  - Over-decomposition