Unit 7

Logical Database Design

Contents

7.1 Introduction

- Logical Database Design vs. Physical Database Design
- Problem of Normalization
- Normal Forms

Logical Database Design

- Logical database design vs. Physical database design
- Logical Database Design
  - **Normalization**
  - **Semantic Modeling,** eg. E-R model
- Problem of Normalization
  - Given some body of data to be represented in a database, how to decide the suitable logical structure they should have?
    - what relations should exist?
    - what attributes should they have?

Problem of Normalization

<e.g.>

S1, Smith, 20, London, P1, Nut, Red, 12, London, 300
S1, Smith, 20, London, P2, Bolt, Green, 17, Paris, 200
.
.
S4, Clark, 20, London, P5, Cam, Blue, 12, Paris, 400

⇩ **Normalization**

S

S#	SNAME	STATUS	CITY
s1	.	.	London
.	.	.	.

P

P#	...	...	...
.	.	.	.
.	.	.	.

SP

S#	P#	QTY
.	.	.
.	.	.

or

S'

S#	SNAME	STATUS
S1	Smith	.
S2	.	.
.	.	.

P

P#	...	...	...
.	.	.	.
.	.	.	.

SP'

S#	CITY	P#	QTY
S1	London	P1	300
S1	London	P2	200
.	.	.	.

Redundancy ⇨ ***Update Anomalies!*** (異常)

Normal Forms

- A relation is said to be in a particular **normal form** if it satisfies a certain set of constraints.

<e.g.> **1NF**: A relation is in **First Normal Form** (1NF) iff it contains only atomic values.

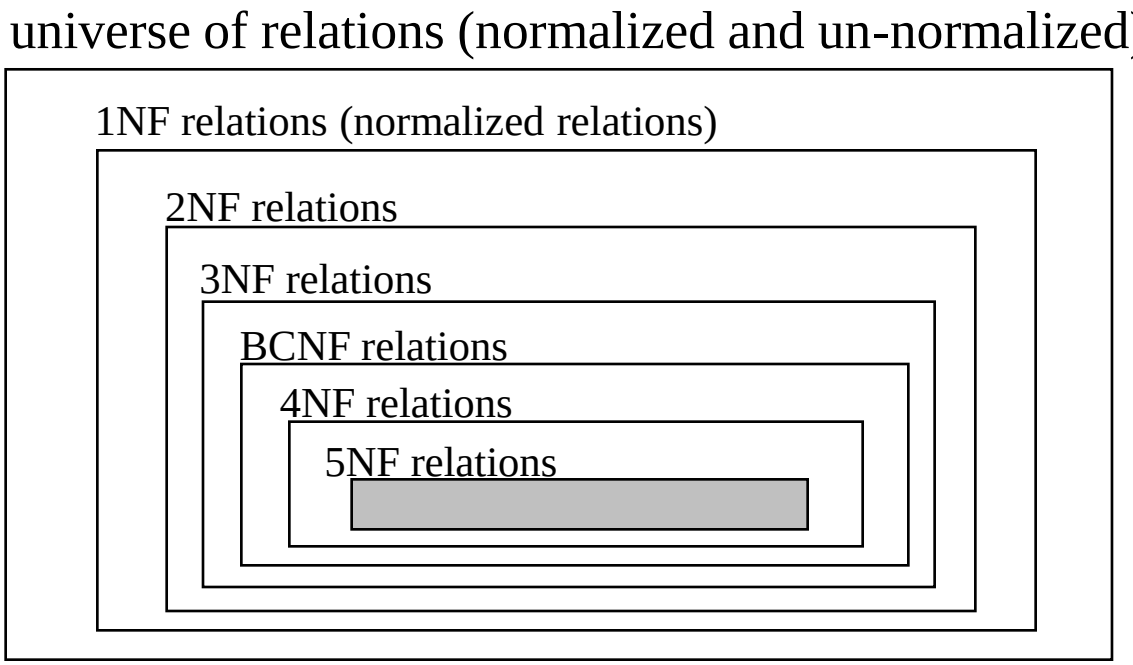


Fig. 7.1: Normal Forms

7.2 Functional Dependency

- Functional Dependency (FD)
- Fully Functional Dependency (FFD)

Functional Dependency

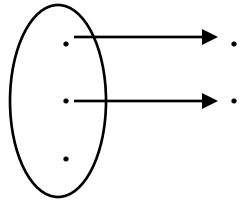
- Functional Dependency
  - Def: Given a relation ***R***, ***R.Y*** is functionally dependent on ***R.X*** *iff* each ***X***-value has associated with it precisely one ***Y***-value (at any time).
  - Note: ***X, Y*** may be the composite attributes.

R

	X		Y

- Notation:

R.X ⟶ R.Y



read as "R.X functionally determines R.Y"

Functional Dependency (cont.)

<e.g.1>

S

S.S# ⟶ S.SNAME

S.S# ⟶ S.STATUS

S.S# ⟶ S.CITY

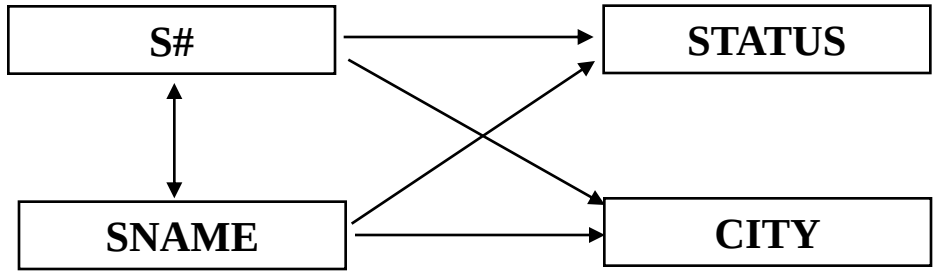
S.STATUS ↛ S.CITY

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

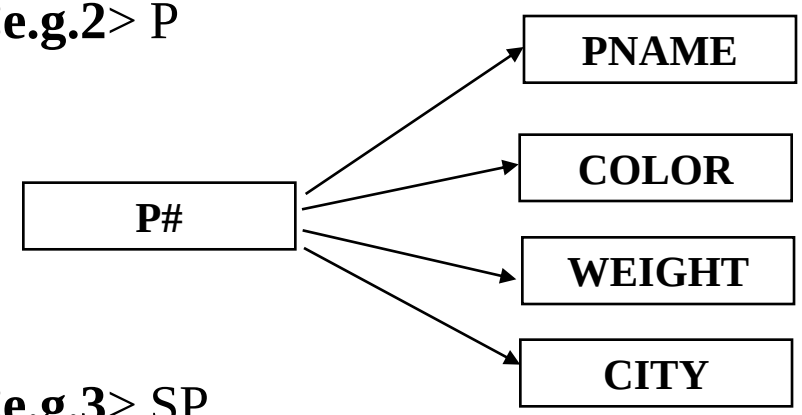
Note: Assume STATUS is some factor of Supplier and no any relationship with CITY.

FD Diagram:



Functional Dependency (cont.)

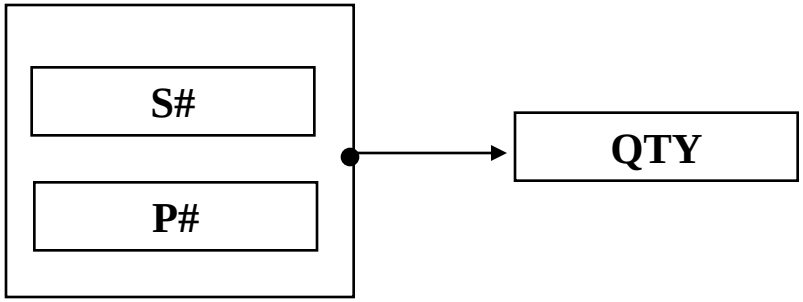
<**e.g.2**> P



P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

<**e.g.3**> SP



SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- If ***X*** is a candidate key of ***R***, then all attributes ***Y*** of ***R*** are functionally dependent on ***X***. (i.e. X ⟶ Y)

Fully Functional Dependency (FFD)

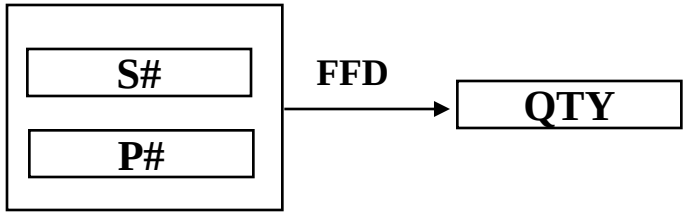
- Def: ***Y*** is <u>fully functionally dependent</u> on ***X*** ***iff***
  - (1) *Y* is FD on *X*
  - (2) *Y* is not FD on any proper subset of *X*.

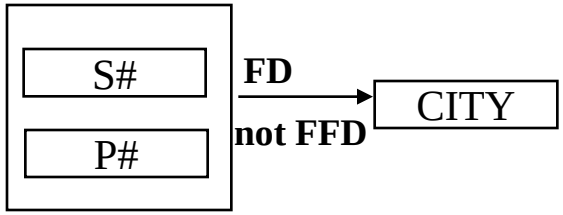
S#		$city_1$	$city_2$
S1		London	Taipei

<e.g> **SP'** (S#, CITY, P#, QTY)

SP'

S#	CITY	P#	QTY
S1	London	P1	300
S1	London	P2	200
…	….	…	…





S#	CITY
…	…..

Fully Functional Dependency (cont.)

<Note> 1. Normally, we take FD to mean FFD.

2. FD is a semantic notion.

<e.g> S# ⟶ CITY (Ref P.10-9)

Means: each supplier is located in precisely one city.

3. FD is a special kind of integrity constraint.

```
CREATE INTEGRITY RULE SCFD
    CHECK FORALL SX FORALL SY
        (IF SX.S# = SY.S# THEN SX.CITY = SY.CITY);
```

4. FDs considered here applied within a single relation.

<e.g> SP.S# ⟶ S.S# is not considered!

7.3 First, Second, and Third Normal Forms (1NF, 2NF, 3NF)

Normal Forms: 1NF

- Def: A relation is in 1NF ***iff*** all underlying simple domains contain atomic values only.

fact

S#	STATUS	CITY	(P#, QTY)
S1	20	London	{(P1, 300), (P2, 200), ..., (P6, 100)}
S2	10	Paris	{(P1, 300), (P2, 400)}
S3	10	Paris	{(P2, 200)}
S4	20	London	{(P2, 200), (P4, 300), (P5, 400)}

Suppose 1. CITY is the main office of the supplier.
2. STATUS is some factor of CITY

FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Key:(S#,P#),
Normalized 1NF

1NF Problem: Update Anomalies!

<1> **Update**

If suppler S1 moves from London to Paris, then 6 tuples must be updated!

<2> **Insertion**

Cannot insert a supplier information if it doesn't supply any part, because that will cause a null key value.

FIRST

S#	STATUS	CITY	P#	QTY
.	.	.	.	.
S3	20	Paris	P2	300
.	.	.	.	.
.	.	.	.	.
S5	**30**	**Athens**	**NULL**	**NULL**

<3> **Deletion**

Delete the information that "S3 supplies P2", then the fact "S3 is located in Paris" is also deleted.

FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	**P2**	**200**
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Key:(S#,P#),
Normalized 1NF

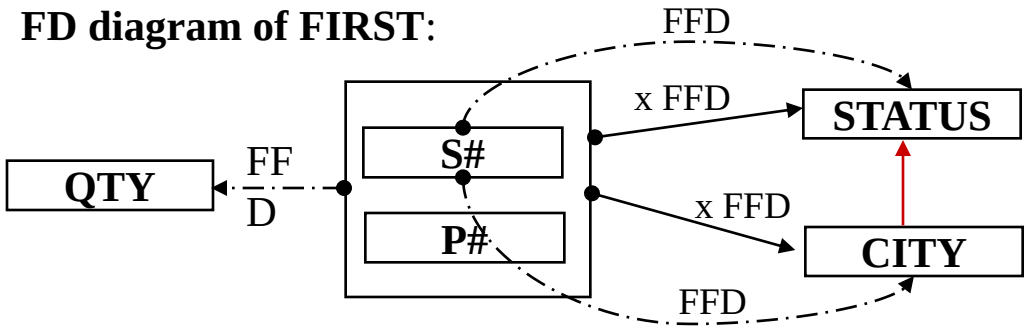
1NF Problem: Update Anomalies! (cont.)

<e.g.> Suppose 1. CITY is the main office of the supplier.

2. STATUS is some factor of CITY (ref.p.7-9)

Primary key of **FIRST**: (S#, P#)

FD diagram of FIRST:



FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	**P2**	**200**
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Key:(S#,P#),
Normalized 1NF

FD:

1. S# → STATUS
2. S# → CITY
3. CITY → STATUS
4. (S#, P#) → QTY

primary key (S#, P#) $\xrightarrow{\text{FFD}}$ STATUS

primary key (S#, P#) $\xrightarrow{\text{FFD}}$ CITY

Normal Form: 2NF

- Def: A relation R is in 2NF iff

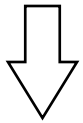
  (1) R is in 1NF (i.e. atomic)

  (2) Non-key attributes are FFD on primary key. (e.g. QTY, STATUS, CITY in FIRST)

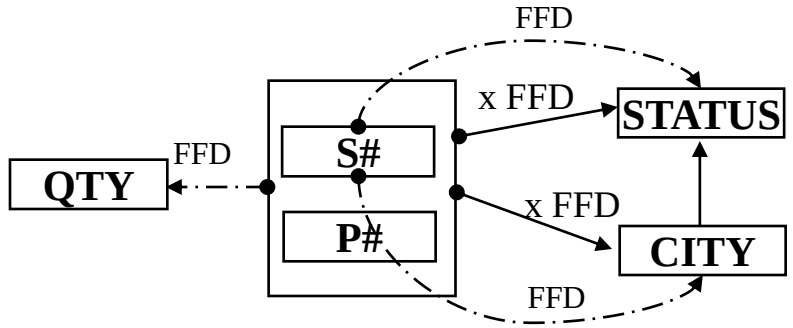
<e.g.> FIRST is in 1NF, but not in 2NF

(S#, P#) $\xrightarrow{\text{FFD}}$ STATUS, and

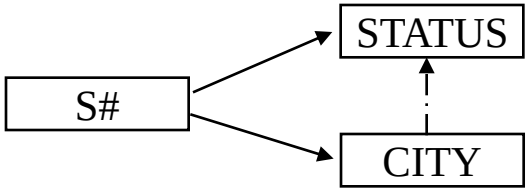
(S#, P#) $\xrightarrow{\text{FFD}}$ CITY



Decompose FIRST into:



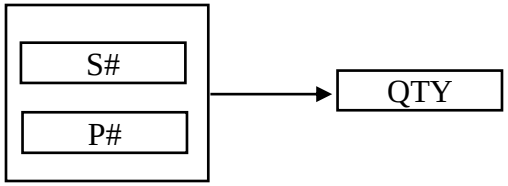
<1> SECOND (S#, STATUS, CITY):
primary key: S#



FD:

1. S# → STATUS
2. S# → CITY
3. CITY → STATUS

<2> SP (S#, P#, QTY):
Primary key: (S#, p#)

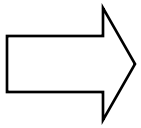


FD: 4. (S#, P#) → QTY

Normal Form: 2NF (cont.)

FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400



SECOND (in 2NF)

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	**10**	**Paris**
S4	20	London
S5	**30**	**Athens**

SP (in 2NF)

S#	P#	QTY)
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S2	P1	300
S2	P2	400
S3	**P2**	**200**
S4	P4	300
S4	P5	400

<1> **Update:** S1 moves from **London** to **Paris**

<2> **Insertion:** **(S5 30 Athens)**

<3> **Deletion**

Delete **"S3 supplies P2 200"**, then the fact **"S3 is located in Paris"** is also deleted.

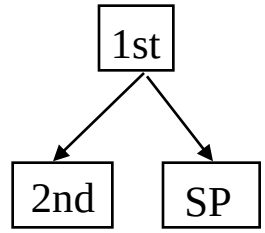
Normal Form: 2NF (cont.)

- A relation in 1NF can always be reduced to an equivalent collection of 2NF relations.
- The reduction process from 1NF to 2NF is *non-loss* decomposition.

FIRST(S#, STATUS, SCITY, P#, QTY)

projections ↓ ↑ natural joins

SECOND(S#, STATUS, CITY), **SP**(S#,P#,QTY)



- The collection of 2NF relations may contain "more" information than the equivalent 1NF relation.

<e.g.> (S5, 30, Athens)

Problem: Update Anomalies in SECOND!

- **Update Anomalies** in **SECOND**

  <1> UPDATE: if the status of London is changed from 20 to 60, then two tuples must be updated

  <2> DELETE: delete supplier S5, then the fact "the status of Athens is 30" is also deleted!

  <3>INSERT: cannot insert the fact "the status of Rome is 50"!

SECOND (in 2NF)

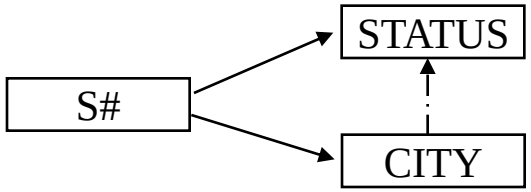
S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

- **Why:**

  S.S# ⟶ S.STATUS

  S.S# ⟶ S. CITY

  S.CITY ⟶ S.STATUS cause a transitive dependency



FD:

1. S# ⟶ STATUS
2. S# ⟶ CITY
3. CITY ⟶ STATUS

Normal Forms: 3NF

- Def : A relation R is in 3NF iff

  (1) R is in 2NF

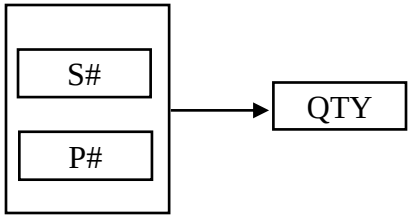
  (2) Every non-key attribute is non-transitively dependent on the primary key.

  e.g. STATUS is transitively on S#

  (i.e., non-key attributes are mutually independent)

<e.g.> **SP** is in 3NF, but **SECOND** is not!

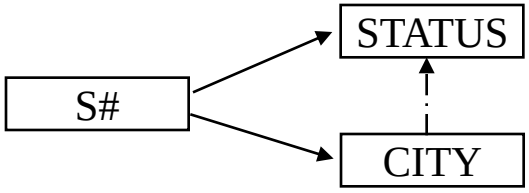
SP FD diagram



SECOND (not 3NF)

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

SECOND FD diagram



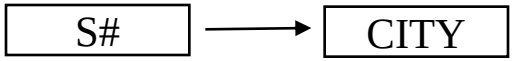
Normal Forms: 3NF (cont.)

- Decompose **SECOND** into:

<1> **SC**(S#, CITY)

primary key : S#

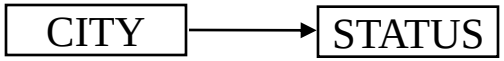
FD diagram:



<2> **CS**(CITY, STATUS):

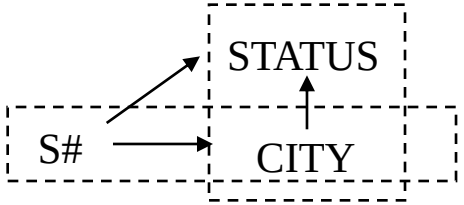
primary key: CITY

FD diagram:



SECOND

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens



SC (in 3NF)

S#	CITY
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

CS (in 3NF)

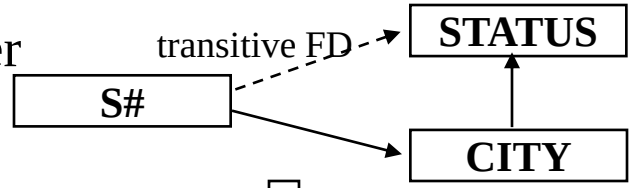
CITY	STATUS
Athens	30
London	20
Paris	10
Rome	50

Normal Forms: 3NF (cont.)

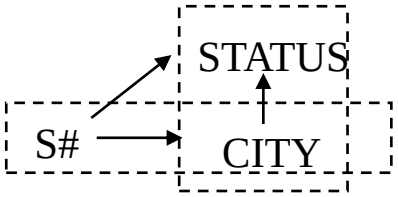
- Note:
  (1) Any 2NF diagram can always be reduced to a collection of 3NF relations.
  (2) The reduction process from 2NF to 3NF is *non-loss* decomposition.
  (3) The collection of 3NF relations may contain "*more information*" than the equivalent 2NF relation.

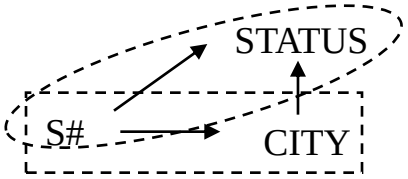
Good and Bad Decomposition

- Consider



Suppose 1. CITY is the main office of the supplier.
2. STATUS is some factor of CITY (ref. p.7-9)





①Decomposition **A:**

SC: S# → CITY

CS: CITY → STATUS

Good !

(Rome, 50) can be inserted.

② Decomposition **B**:

SC: S# → CITY

CS: S# ⇢ STATUS

Bad !

(Rome, 50) can not be inserted unless there is a supplier located at Rome.

③ Decomposition **C**:

S# -> status

city -> status

'Good' or 'Bad' is dependent on item's semantic meaning!!

Good and Bad Decomposition (cont.)

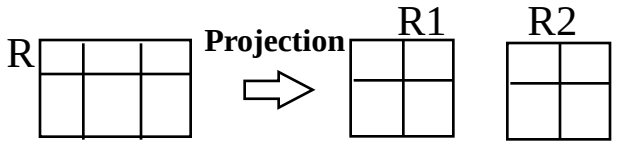
- Independent Projection: (by Rissanen '77, ref[10.6])

  Def: Projections R1 and R2 of R are **independent** *iff*

  (1) Any FD in R can be reduced from those in R1 and R2.

  (2) The common attribute of R1 and R2 forms a candidate key for at least one of R1 and R2.

<e.g.>



- Decomposition **A**:

  SECOND(S#, STATUS, CITY)

  (R)

  ⟶ { SC (S#, CITY) (R1)
  CS (CITY, STATUS) (R2)

  (1) FD in SECOND (R):

  S# ⟶ CITY

  CITY ⟶ STATUS

  S# - - - - ▸ STATUS

  FD in SC and CS

  R1: S# ⟶ CITY

  R2: CITY ⟶ STATUS

  ⇨ S# - - - - ▸ STATUS

  (2) Common attribute of SC and CS is CITY, which is the primary key of CS.

  ∴ SC and CS are independent → Decomposition **A** is good!

Good and Bad Decomposition (cont.)

Decomposition B:

SECOND(S#, STATUS, CITY) $\longrightarrow$ { SC (S#, CITY)
SS (S#, STATUS)

(1) FD in SC and SS:

S# $\longrightarrow$ CITY
S# $\longrightarrow$ STATUS

$\not\Rightarrow$ CITY $\dashrightarrow$ STATUS

$\therefore$ SC and SS are dependent $\rightarrow$ decomposition B is bad!
though common attr. S# is primary key of both SC, SS.

Decomposition C:

SECOND(S#, STATUS, CITY) $\longrightarrow$ { SS (S#, STATUS)
CS (CITY, STATUS)

(1) FD in SS and CS

S# $\rightarrow$ STATUS
CITY $\rightarrow$ STATUS

$\not\Rightarrow$ S# $\dashrightarrow$ CITY

(2) Common attribute STATUS is not only a candidate key of either SS or CS.

$\therefore$ Decomposition C is not only a bad decomposition, but also an invalid decomposition, since it is not non-loss.

Atomic Relation

- Def: Atomic Relation -- A relation that cannot be decomposed into independent components.

  <e.g.> SP (S#, P#, QTY) is an atomic relation, while

  S (S#, SNAME, STATUS, CITY) is not.

7.4 Boyce/Codd Normal Form (BCNF)

Problems of 3NF:

Do not deal with the cases:

<1> A relation has multiple candidate keys,
<2> Those candidate keys were composite,
<3> The candidate keys are overlapped.

Example

- S: student
- J: subject
- T: teacher

SJT(S, J, T)

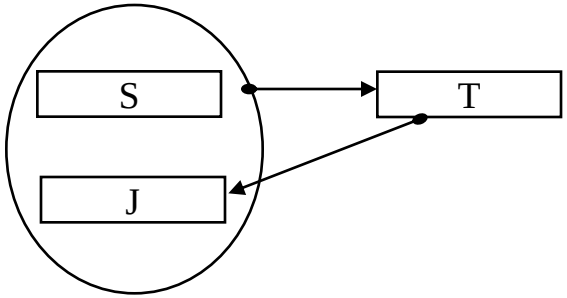
S	J	T
Smith	Math.	Prof. White
Smith	Physics	Prof. Green
Jones	Math.	Prof. White
Jones	Physics	Prof. Brown

- Meaning of a tuple: student S is taught subject J by teacher T.
- Suppose
  - For each subject, each student of that subject is taught by only one teacher.

    i.e. (S, J) $\rightarrow$ T
  - Each teacher teaches only one subject.

    i.e. T $\rightarrow$ J
- Candidate keys

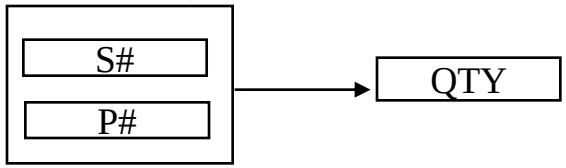
  (S, J) and (S, T)
- FD diagram



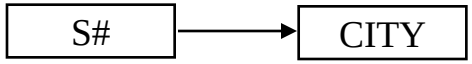
BCNF

- Def: A relation R is in BCNF iff every determinant is a candidate key.
  - A ⟶ B: A determines B, and A is a **<u>determinant</u>**.
  - <e.g.1> [only one candidate key]

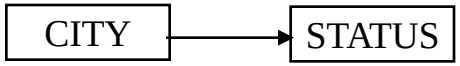
SP (S#, P#, QTY): in 3NF & BCNF



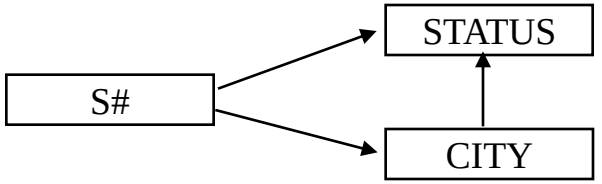
SC (S#, CITY): in 3NF & BCNF

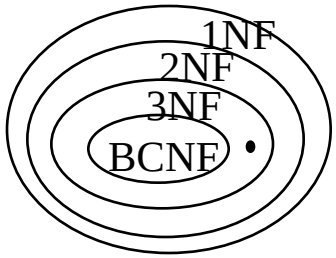


CS (CITY, STATUS): in 3NF & BCNF



SECOND(S#, STATUS, CITY): ***not*** in 3NF & ***not*** in BCNF



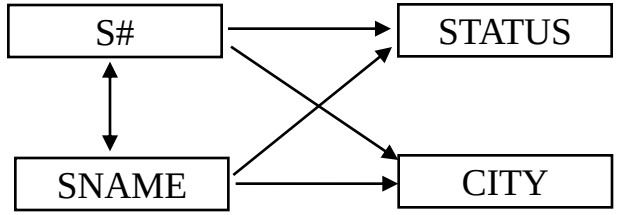


- in 3NF, ***not*** in BCNF e.g.3, e.g.4 (P.7-33)

BCNF (cont.)

- <e.g.2> [two disjoint (nonoverlapping) candidate keys]

  S(S#, SNAME, STATUS, CITY)



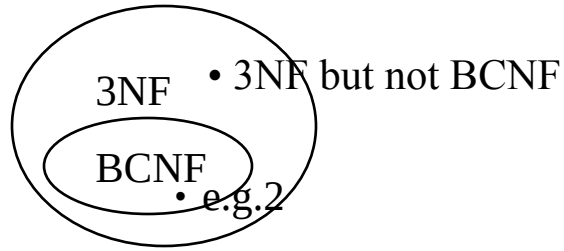
Assume :

(1) CITY, STATUS are independent

(2) SNAME is a candidate key

⇨ S#, SNAME (determinants) are candidate keys.

∴ S is in BCNF (also in 3NF).



BCNF (cont.)

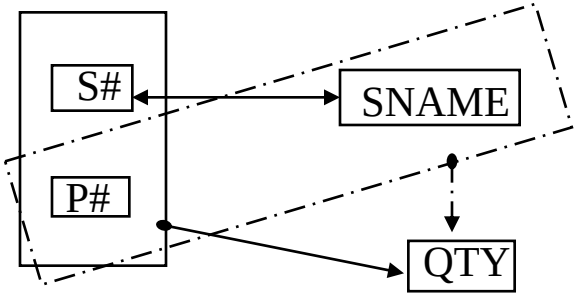
<e.g.3> [overlapping candidate keys -1]

SSP (S#, SNAME, P#, QTY)

key in SSP: (S#, P#), (SNAME, P#)

FD in SSP

1. S# ⟶ SNAME
2. SNAME ⟶ S#
3. {S#, P#} ⟶ QTY
4. {SNAME, P#} ⟶ QTY



SSP

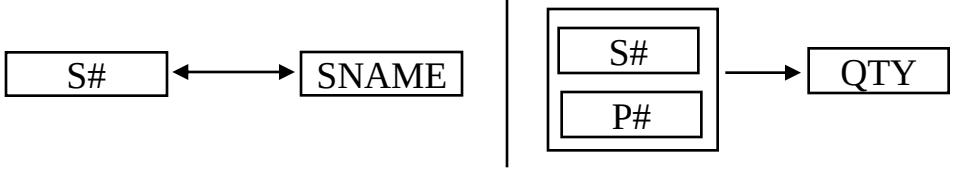
S#	SName	P#	QTY

- in 3NF∵ nonkey attribute is FFD on primary key and mutually independent. e.g. QTY only
- not in BCNF ∵ S# is a determinant but not a candidate key. S# ⟶ SNAME

Decompose:

SS (S#, SNAME): in BCNF

SP (S#, P#, QTY): in BCNF



BCNF (cont.)

<e.g.4> [overlapping candidate keys-2]

SJT(S, J, T)

- S: student
- J: subject
- T: teacher

S	J	T
Smith	Math.	Prof. White
Smith	Physics	Prof. Green
Jones	Math.	Prof. White
Jones	Physics	Prof. Brown

- meaning of a tuple: student S is taught subject J by teacher T.
- Suppose
  - For each subject, each student of that subject is taught by only one teacher.

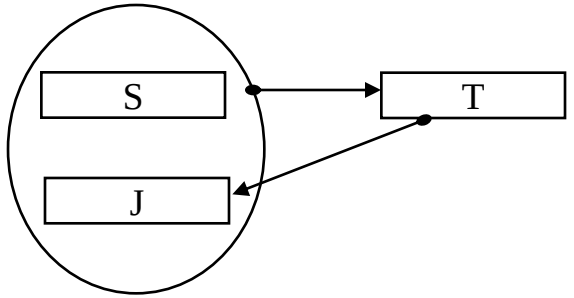
    i.e. (S, J) $\rightarrow$ T
  - Each teacher teaches only one subject.

    i.e. T $\rightarrow$ J

– Candidate keys

(S, J) and (S, T)

– FD diagram



BCNF (cont.)

- In 3NF, no nonkey attribute.
- not in BCNF, $T \rightarrow J$ but T is not a candidate key
- **update anomalies occur**!

e.g. (delete "Jones is studying Physics" ⟶ the fact "Brown teaches Physics" is also deleted!)

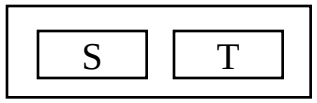
Decompose 1:

ST (S, T)

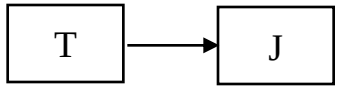
S	T
Smith	Prof. White
Smith	Prof. Green
Jones	Prof. White
Jones	Prof. Brown

TJ(T, J)

T	J
Prof. White	Math.
Prof. Green	Physics
Prof. Brown	physics



in BCNF



in BCNF

Decompose 2:

S	J

T	J

- Is this decomposition Good or Bad?
- In Rissanen's sense, ST(S, T) and TJ(T, J) are not independent!

the FD: (S,T)⟶T cannot be deduced from FD: T⟶J

∴The two objectives:

<1> decomposing a relation into BCNF, and

<2> decomposing it into independent components may be in conflict!

S	J	T
Smith	Math.	Prof. White
Smith	Physics	Prof. Green
Jones	Math.	Prof. White
Jones	Physics	Prof. Brown

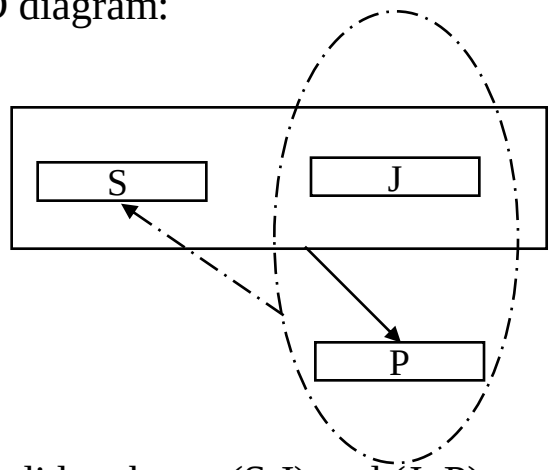
BCNF (cont.)

<e.g.5> [overlapping candidate keys-3]

EXAM(S, J, P); S: student, J: subject, P: position.

- meaning of a tuple: student S was examined in subject J and achieved position P in the class.
- suppose no two students obtained the same position in the same subject.

  i.e. (S, J) → P and (J, P) → S
- FD diagram:



EXAM

S	J	P
A	DBMS	5
B	DBMS	8
A	Network	1

- candidate keys: (S,J) and (J, P), overlap key: J.
- in BCNF !

Why Normal Form?

- Avoid update anomalies
- Consider the SSP(S#, SNAME, P#, QTY)

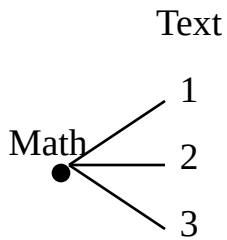
  Common sense will tell us SS(S#, SNAME) & SP(S#, P#, QTY) is a better design.
- The concepts of FD, 1NF, 2NF, 3NF and BCNF to formalize common sense.
- → Mechanization is possible!
  - i.e., we can write a program to do the work of normalization for us!

7.5 Fourth Normal Form (4NF)

Un-Normalized Relation

CTX

COURSE	TEACHER	TEXT
Physics	{Prof. Green, Prof. Brown}	{Basic Mechanics, Principle of Optics}
Math.	{Prof. Green}	{Basic Mechanics, Vector Analysis, Trigonometry}



- meaning of a record: the specified course can be taught by any of the specified teachers and uses all of the specified texts as references.
- Assume:
  - For a given course, there exists any number of teachers and any number of texts.
  - Teachers and texts are independent.
  - A given teacher or a given text can be associated with any number of courses.

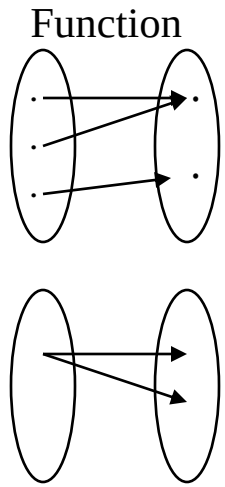
Un-normalized Relation (cont.)

- Note: No FD exists in this relation!

Normalized

C T X

COURSE	TEACHER	TEXT
Physics(c)	Prof. Green(t1)	Basic Mechanics (x1)
physics(c)	Prof. Green(t1)	Principle of Optics (x2)
physics(c)	Prof. Brown(t2)	Basic Mechanics (x1)
physics(c)	prof. Brown(t2)	Principles of Optics(x2)
Math	prof. Green	Basic Mechanics
Math	prof. Green	Vector Analysis
Math	prof. Green	Trigonometry



Un-normalized Relation (cont.)

- Meaning of a tuple: course C can be taught by teacher T and uses text X as a reference.
  - primary key: (COURSE, TEACHER, TEXT)

COURSE	TEACHER	TEXT
Physics(c)	Prof. Green(t1)	Basic Mechanics (x1)
physics(c)	Prof. Green(t1)	Principle of Optics (x2)
physics(c)	Prof. Brown(t2)	Basic Mechanics (x1)
physics(c)	prof. Brown(t2)	Principles of Optics(x2)
Math	prof. Green	Basic Mechanics
Math	prof. Green	Vector Analysis
Math	prof. Green	Trigonometry

- Check:
  - in 1NF (simple domain contains atomic value only)
  - in 2NF (Nonkey attributes are FFD on primary key,

    no key attributes)
  - in 3NF (Nonkey attributes are mutually independent.)
  - in BCNF (Every determinant is a candidate key)

Un-normalized Relation (cont.)

- Problem: a good deal of redundancy!
  - property:
    if (c, t1, x1), (c, t2, x2) both appear
    then (c, t1, x2) , (c, t2, x1) both appear also!
  - reason: No FD, but has MVD!

COURSE	TEACHER	TEXT
Physics(c)	Prof. Green (t1)	Basic Mechanics (x1)
physics(c)	Prof. Green (t1)	Principle of Optics (x2)
physics(c)	Prof. Brown (t2)	Basic Mechanics (x1)
physics(c)	prof. Brown (t2)	Principles of Optics (x2)
Math	prof. Green	Basic Mechanics
Math	prof. Green	Vector Analysis
Math	prof. Green	Trigonometry

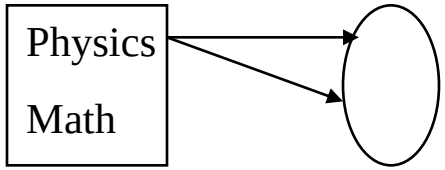
intuitively decomposed ⇩

CT:

COURSE	TEACHER
Physics	Prof. Green
Physics	Prof. Brown
Math	Prof. Green

CX:

COURSE	TEXT
Physics	Basic Mechanics
Physics	Principles of Optics
Math	Basic Mechanics
Math	Vector Analysis
Math	Trigonometry



Not FD!

- the decomposition cannot be made on the basis of FD.

MVD (Multi-Valued Dependencies)

- Def: Given R(A, B, C), the multivalued dependence (MVD)
  R.A $\rightarrow\!\!\!\!\rightarrow$ R.B holds in R iff the set of B-values matching a given (A-value, C-value) pair is R, depend only on A-value, and is independent of C-value.

  <e.g> COURSE $\rightarrow\!\!\!\!\rightarrow$ TEACHER, COURSE $\rightarrow\!\!\!\!\rightarrow$ TEXT

  { *physics* , { *Green* / *Brown* } , *Basic Mechanics* }

  A **B** **C**

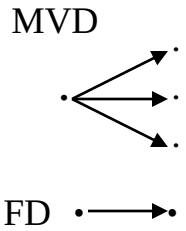
- Thm: Given R(A, B, C), the MVDR.A $\rightarrow\!\!\!\!\rightarrow$ R.B holds iff the MVD R.A $\rightarrow\!\!\!\!\rightarrow$ R.C also holds.
  - Notation: R.A $\rightarrow\!\!\!\!\rightarrow$ R.B | R.C

    <e.g.> COURSE $\rightarrow\!\!\!\!\rightarrow$ TEACHER | TEXT

  <Note> 1. FD is a special case of MVD

  $\rightarrow$ all FD's are also MVD's

  2. MVDs (which are not also FD's) can exist only if the relation R has at least 3 attributes.



Norma Forms: 4NF

- Problem of CTX: involves MVD's that are not also FD's.
- Def: A relation R is in **4NF**

  iff whenever there exists an MVD in R, say A $\twoheadrightarrow$ R,

  then all attributes of R are also FD on A.

  i.e. R is in 4NF iff (i) R is in BCNF, (ii) all MVD's in R are in fact FD's.

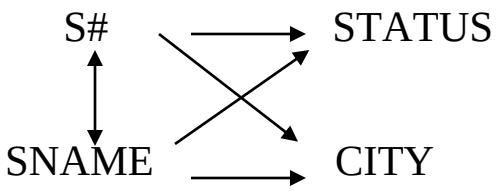
  i.e. R is in 4NF iff (i) R is in BCNF, (ii) no MVD's in R.

<e.g.1> CTX (COURSE, TEACHER, TEXT)

COURSE $\twoheadrightarrow$ TEACHER

COURSE $\twoheadrightarrow$ TEXT $\quad \therefore$ *not in 4NF*

<e.g.2> S (S#, SNAME, STATUS, CITY)



no MVD which is not FD in 4NF

Norma Forms: 4NF (cont.)

- Thm: Relation R(A, B, C) can be <u>no loss</u> decomposed into R1(A, B) and R2(A, C) iff A $\twoheadrightarrow$ B | C holds in R.

  <e.g> CTX (COURSE, TEACHER, TEXT)

  COURSE $\twoheadrightarrow$ TEACHER | TEXT

  ⇩

  CT (COURSE, TEACHER)

  CX (COURSE, TEXT)

  no MVD $\dot{\rightarrow}$ in 4NF

  no MVD $\rightarrow$ in 4NF

7.6 Fifth Normal Form (5NF)

A Surprise

- There exist relations that cannot be nonloss-decomposed into two projections, but can be decomposed into three or more.
- Def: n-decomposable (for some n > 2)

  the relation can be nonloss-decomposed into n projections, but not into m projection for any m < n.
- <e.g.> SPJ (S#, P#, J#); S: supplier, P: part, J: project.
  - Suppose in real world

    if (a) Smith supplies monkey wrenches, and

    (b) Monkey wrenches are used in Manhattan project, and

    (c) Smith supplies Manhattan project.

    then

    (d) Smith supplies Monkey wenches to Manhatan project.

  i.e.

  If (s1, p1, j2), (s2, p1, j1), (s1, p2, j1) appear in SPJ

  Then (s1, p1, j1) appears in SPJ also.

  – no MVD $\twoheadrightarrow$ in 4NF

A Surprise (cont.)

- update problem of SPJ

SPJ:

S#	P#	J#
S1	P1	J2
S1	P2	J1

- If (S2, P1, J1) is to be inserted
  then (S1, P1, J1) must also be inserted

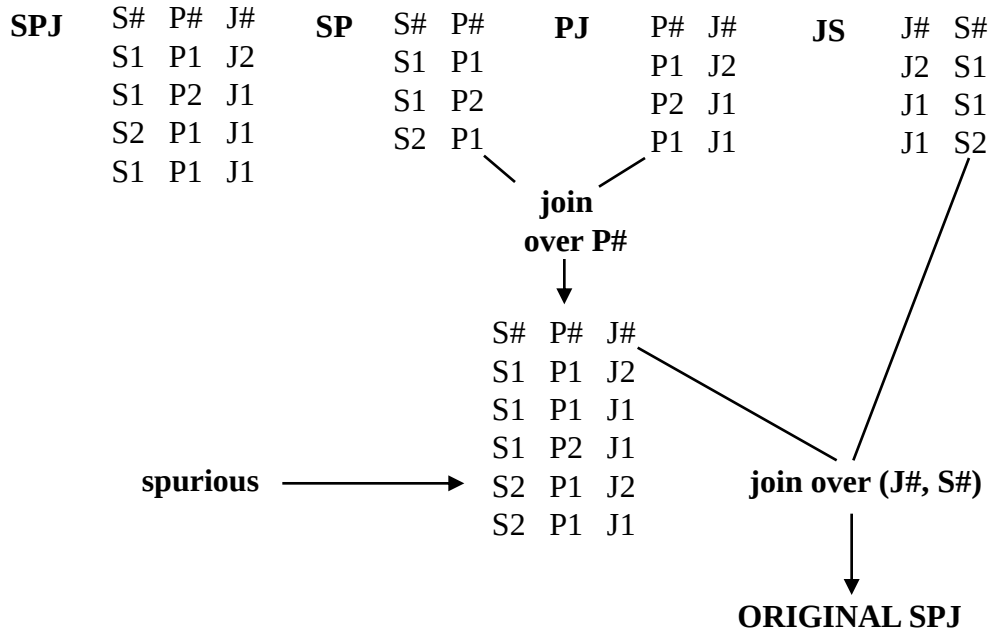
SPJ:

S#	P#	J#
S1	P1	J2
S1	P2	J1
S2	P1	J1
S1	P1	J1

- If (S1, P1, J1) is to be deleted, then one of the following must also be deleted
  (i) (S1, P1, J2): means S1 no longer supplies P1.
  (ii) (S1, P2, J1): means S1 no longer supplies J1.
  (iii) (S2, P1, J1): means J1 no longer needs P1.

A Surprise (cont.)

- SPJ is not 2-decomposable, but is 3-decomposable!



Join Dependency (JD)

- Def: A Relation R satisfies the join dependency (JD)

  * (X, Y, ..., Z)

  iff R is equal to the join of its projections on X, Y, ..., Z,
  where X, Y, ..., Z are subsets of the set of attributes of R.

- <e.g.> SPJ satisfies the JD *(SP, PJ, JS) i.e. SPJ is 3-decomposable.

- MVD is a special case of JD.

  Thm: R (A, B, C) can be nonloss-decomposed into
  ∵ R_1(A, B) and R_2(A, C) iff A $\twoheadrightarrow$ B|C holds.

  Thm: R (A, B, C) satisfies the JD *(AB, AC)
  iff A $\twoheadrightarrow$ B|C holds.

- <Note>

  JD's are the most general form of dependency possible,
  so long as we concentrate on the dependencies that deal
  with a relation being decomposed via projection and recomposed via join.

Norma Forms: 5NF

- Def: A relation R is in 5NF (or PJ/NF) iff every JD in R is a consequence of the candidate keys of R.
- <e.g.1> Suppose S# and SNAME are candidate keys of

  S (S#, SNAME, STATUS, CITY).

  <i> * ((S#, SNAME, STATUS), (S#, CITY))

  is a consequence of S# (a candidate key of S)

  <ii> * ((S#, SNAME), (S#, STATUS), (SNAME, CITY))

  is a consequence of the candidate keys S# and SNAME.

Norma Forms: 5NF (cont.)

- <e.g.2> Consider SPJ (S#, P#, J#), the candidate key of SPJ is

  (S#, P#, J#).

  However, there exists a JD

  *((S#, P#), (P#, J#), (J#, S#))

  which is not a consequence of (S#, P#, J#)

  → SPJ not in 5NF!

  ⇨ decomposed:

  SP (S#, P#), PJ (P#, J#), JS (J#, S#):

  (no JD in them ∴ all in 5NF!)

Note:

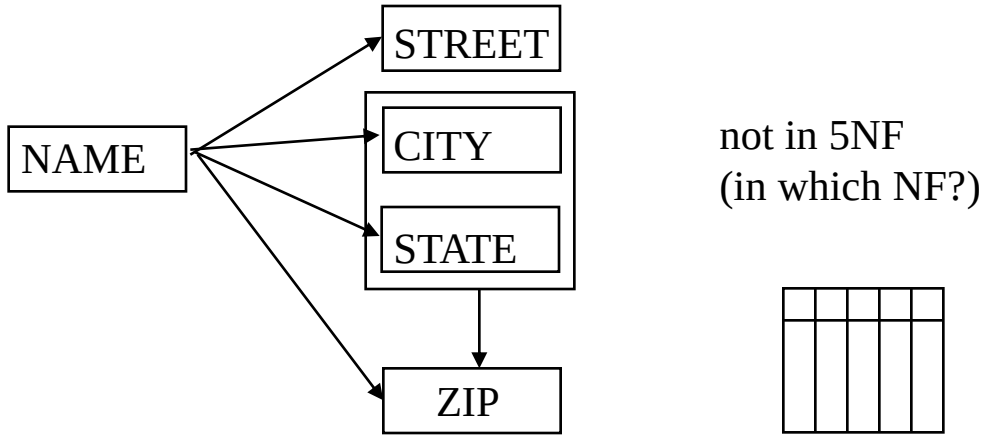
1. Discovering all the JD's is a nontrivial operation.
2. Intuitive meaning of JD may not be obvious.
3. A relation in 4NF but not in 5NF is a pathological case, and likely to be rare in practice.

Concluding Remarks

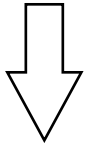
- The technique of non-loss decomposition is an aid to logical database design .
- The overall processes of Normalization:
  - step1: eliminate non-full dependencies.
  - step2: eliminate any transitive FDs.
  - step3: eliminate those FDs in which the determinant is not a candidate key.
  - step4: eliminate any MVDs that are not FDs.
  - step5: eliminate any JDs that are not a consequence of candidate keys.
- General objective:
  - reduce redundancy, and then
  - avoid certain update anomalies.
- Normalization Guidelines are only guidelines.
  - Sometime there are good reasons for not normalizing all the way.

Concluding Remarks (cont.)

<e.g.> **NADDR** (NAME, STREET, CITY, STATE, ZIP)

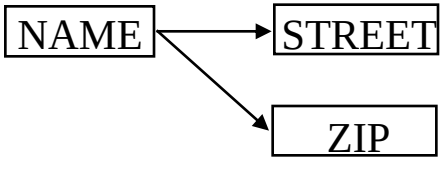


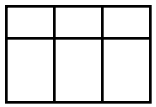
not in 5NF
(in which NF?)



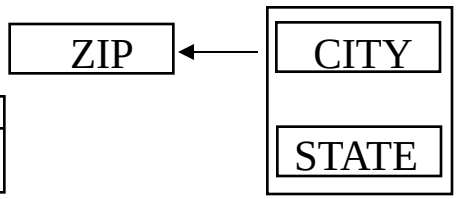
decompose

NSZ (NAME, STREET, ZIP)





ZCS (ZIP, CITY, STATE)



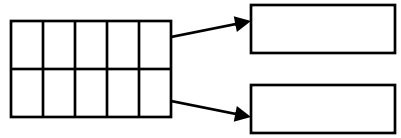
Concluding Remarks (cont.)

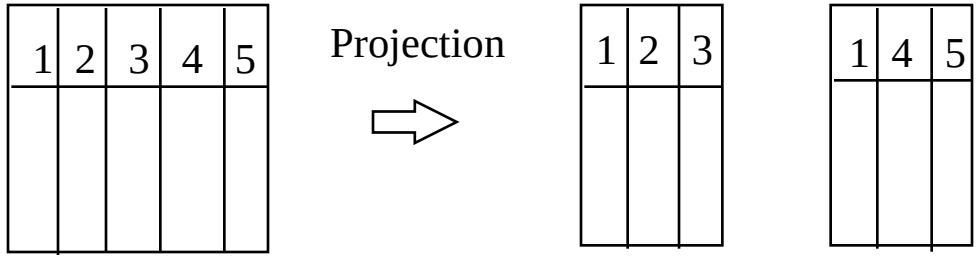
- **However,** (1) STREET, CITY, STATE are almost required together.

  (2) so ZIP do not change very often, such a decomposition seems unlikely to be worthwhile.

- Not all redundancies can be eliminate by projection.
- Research topic: decompose relations by other operator.

e.g. restriction.



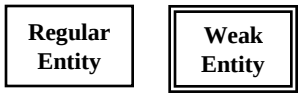


7.7 The Entity/Relationship Model

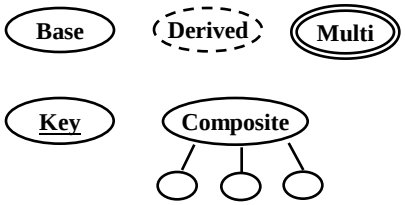
- Proposed by Peter Pin-Shan Chen. "*The Entity-Relationship Model*," in ACM Trans. on Database System Vol. 1, No. 1, pp.9-36, 1976.
- E-R Model contains:
  - **Semantic Concepts** and
  - **E/R Diagram**.

Entity/Relationship Diagram

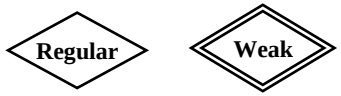
- Entity :



- Property:

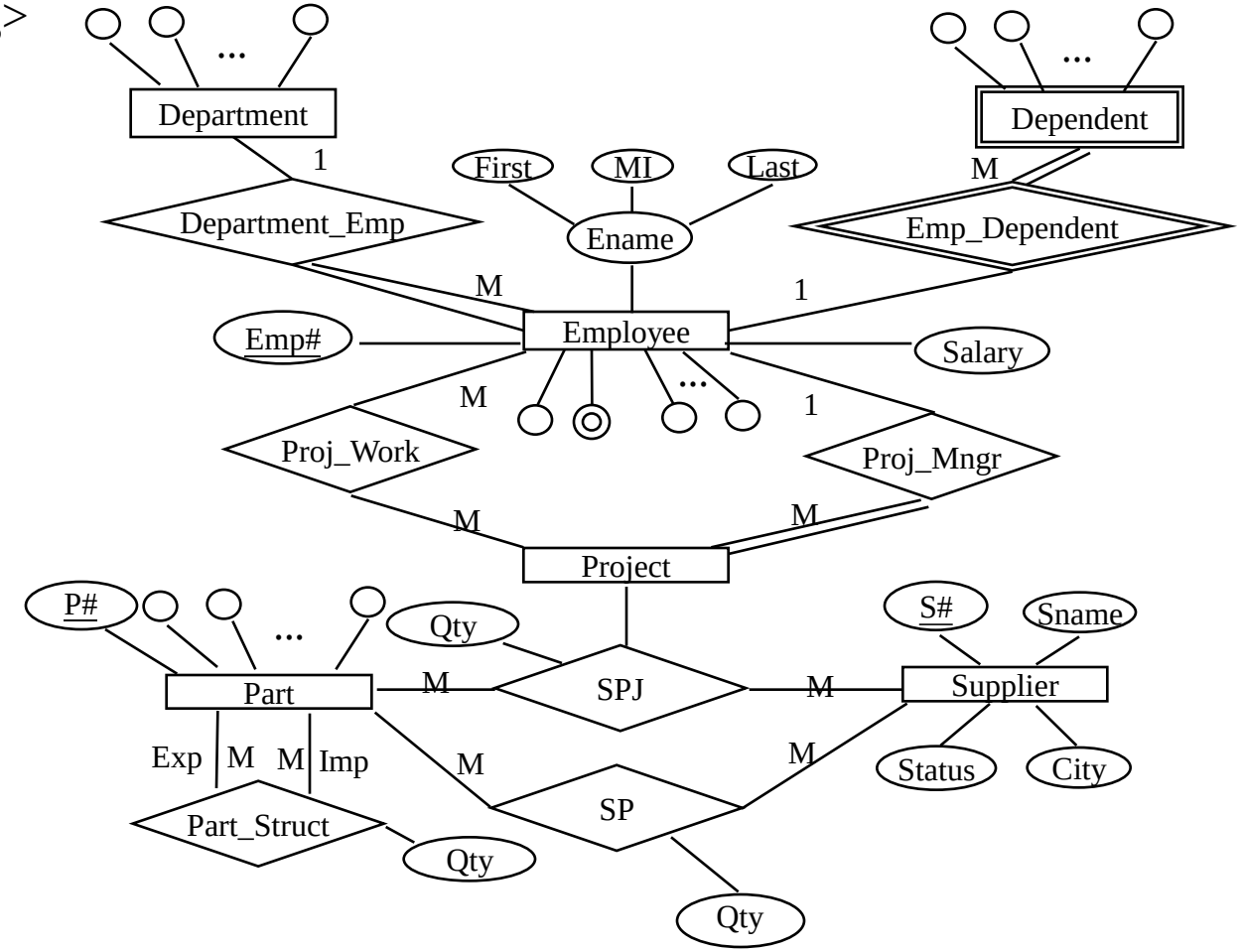


- Relationship:



total ‖ partial |

<e.g>

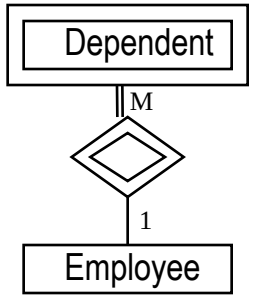


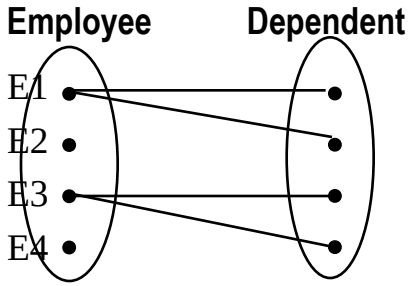
Semantic Concepts

- **Entity**: a thing which can be distinctly identified. e.g. **S**, **P**
  - **Weak Entities**: an entity that is existence-dependent on some other entity. e.g. Dependent
  - **Regular Entities**: an entity that is not weak.
- **Property**:
  - **Simple** or **Composite** Property
  - **key**: unique value
  - **Single** or **Multi-valued**: i.e. repeating data
  - **Missing**: unknown or not applicable.
  - **Base** or **Derived**: e.g. "total quantity"

Semantic Concepts (cont.)

- **Relationship:** association among entities.
  - **participant**: the entity participates in a relationship.
    e.g. Employee, Dependent
  - **degree** of relationship: number of participants.
  - **total participant:** every instance is used.
    e.g. Dependent
  - **partial participant**:
    e.g. Employee



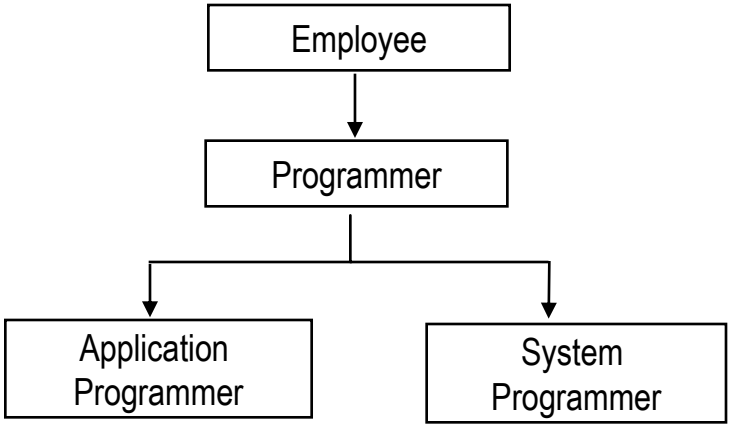


Note: An E/R relationship can be one-to-one, one-to-many, many-to-one, or many-to-many.

Semantic Concepts (cont.)

- **Subtype**: Any given entity can be a subtype of other entity.

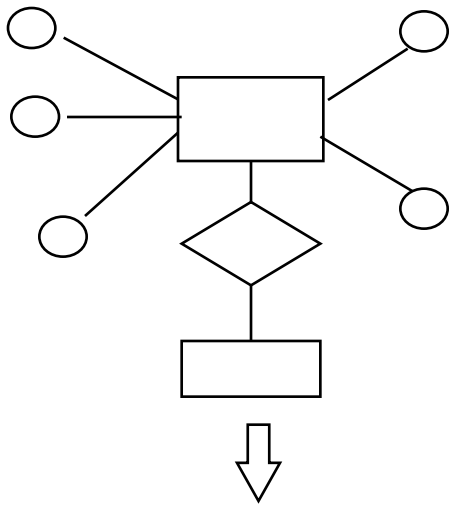
  e.g. An example type hierarchy:



- Programmer $\subseteq$ Employee
- Application Programmer $\subseteq$ Programmer

Transfer E-R Diagram to SQL Definition

ER-Diagram



SQL Definition

(1) Regular Entities → Base Relations

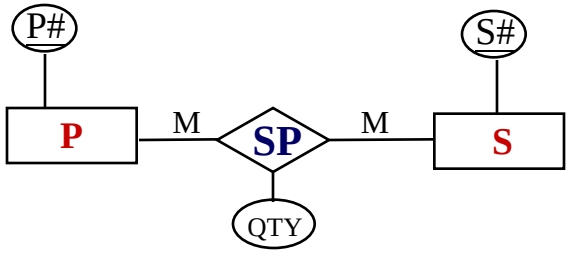
- e.g. Department → DEPT
  Employee → EMP
  Supplier → S
  Part → P
  Project → J
- attributes: properties.
- primary key: key property.
  <e.g> CREATE TABLE EMP
  (EMP# .....
  ⋮
  primary key (EMP#));
  ⋮

(2) Properties → attributes

- multivalued property: normalized first.

Transfer E-R Diagram to SQL Definition (cont.)

(3) Many-to-Many Relationships → Base Relations



<e.g.> Consider:

SUPP-PART → SP

– foreign keys:
key attributes of the participant entity.

– primary key:
(1) combination of foreign keys, or
(2) introduce a new attribute.

– attributes:
primary key ∪ foreign keys ∪ properties

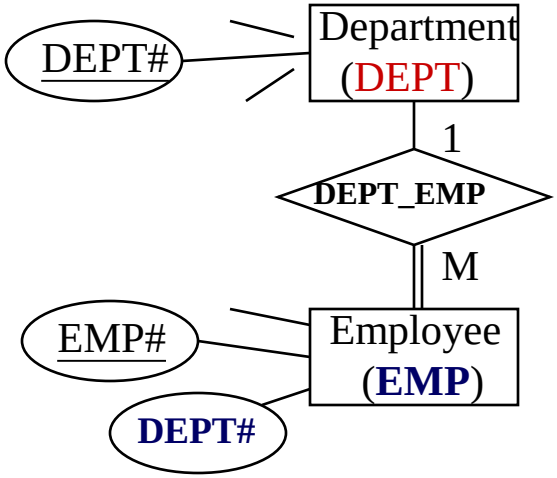
ID	S#	P#	QTY
1			
2			
3			
…			

<e.g> **CREATE TABLE SP**
(S#, ...,
P#, ...,
Qty, ...,
FOREIGN KEY (S#) REFERENCES **S**
NULL NOT ALLOWED
DELETE OF S RESTRICTED
UPDATE OF S.S# CASCADES
FOREIGN KEY (P#) REFERENCES **P**
.....
PRIMARY KEY (S#, P#))

Transfer E-R Diagram to SQL Definition (cont.)

(4) Many-to-One or
One-to-Many or
One-to-One Relationships

→ 不用給**Base-Relation**, 但....



e.g. Consider:

DEPT_EMP(Department_Employee)

1° no any new relation is necessary.

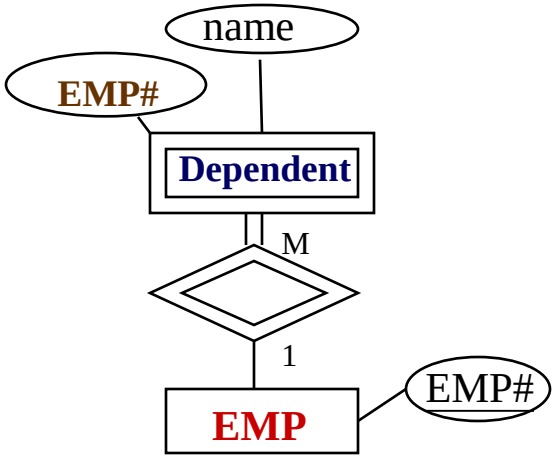
2° Introduce a foreign key in "many" side relation (eg. EMP) that reference the "one" side relation (eg. DEPT).

```
CREATE  TABLE  EMP
 ( EMP#, ...
   DEPT#, ...
    ......
   FOREIGN KEY (DEPT#) REFERENCES DEPT
    ......
   );
```

Transfer E-R Diagram to SQL Definition (cont.)

(5) Weak Entities → Base Relation + Foreign Key

- foreign key: key property of its dependent entity
- primary key: e.g. EMP# e.g. name
  - (1) combination of foreign key and its own primary key
  - (2) introduce a new attribute



e.g. **CREATE TABLE DEPENDENT**

(EMP#, …

name,

.....

FOREIGN KEY **(EMP#)** REFERENCES **EMP**

NULL NOT ALLOWED

DELETE OF EMP CASCADES

UPDATE OF EMP.EMP# CASCADES

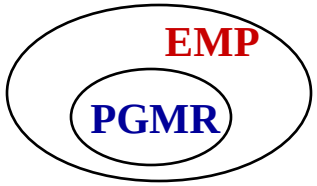
PRIMARY KEY (EMP#, DEPENDENT_NAME));

Transfer E-R Diagram to SQL Definition (cont.)

(6) Supertypes and Subtypes→ Base Relation + Foreign Key

<e.g.> **CREATE TABLE EMP**

(EMP#, ...
DEPT#, ...
SALARY, ...
.....
PRIMARY KEY (EMP#), ...);



CREATE TABLE PGMR

(EMP#, ...
LANG#, ...
SALARY, ...
.....
PRIMARY KEY (EMP#)
FOREIGN KEY (EMP#) REFERENCES **EMP**
.....);

Home Work: Term Project

- Design and implementation an useful, complete, and “real” database system.
- Steps:
  - Take any data you are familiar with. (from your work or ?)
  - System Analysis represented by flow chart
  - By using the E-R model to analysis and describe your data.
  - Design logical database, user interface, and more
  - Design the database system as “complete and real” as possible.
- Due:
  - Demo and
  - A Comprehensive Report