Unit 7

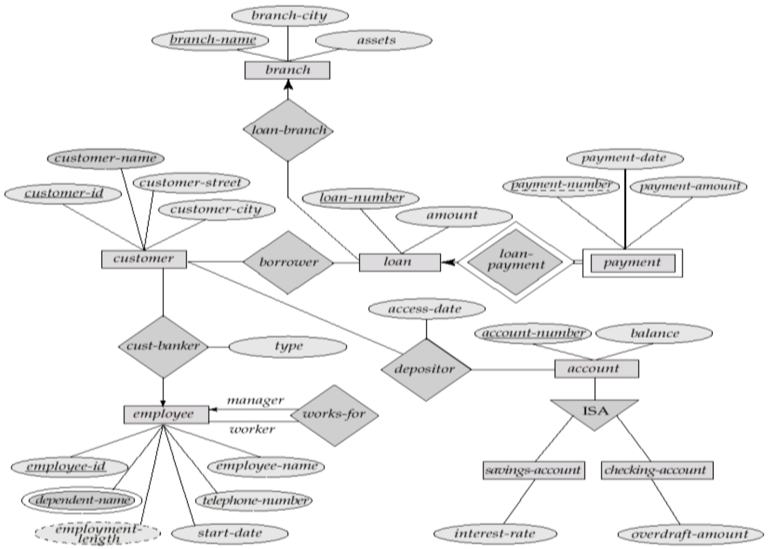
Normalization

(表格正規化)

PART II: 資料庫設計 (Database Design)

- 資料庫問題分析與架構規劃:
  - 若有一大量資料想利用DBMS建資料庫來管理。第一步要分析問題，找到使用者需求
  - 實體-關係模型(Entity-Relationship Model,簡稱E-R Model)是一套資料庫的設計工具。我們可以利用E-R Model分析資料庫問題。它可以把真實世界中複雜的問題中的事物和關係轉化為資料庫中的資料架構
  - 由於利用實體-關係模型設計資料庫時, 並不會牽涉到資料庫的操作、儲存方式等複雜的電腦運作。所以,我們會把心力放在需求分析去規劃想要的資料庫，並以實體-關係圖(E-R Diagram)來呈現
- 資料庫的表格正規化:
  - 實體-關係圖很容易轉化為表格(Tables)，而資料庫就是由許多表格(tables)組成的
  - 這些表格要正規化(Normalization)才能避免將來操作時的異常現象發生
- 設計介面增刪查改資料庫:
  - 如何方便、又有效率的管理存取資料庫是使用者最關心的二個要素
  - 良好的介面設計，可以讓使用者方便的查詢、方便的新增、方便的刪除、方便的修改的處理資料庫

Real-world vs. E-R Model vs. Tables



The real-world enterprise

Semantic Data Model:

Entity-Relationship (E-R) Data Model

1. branch 分公司

branch-name	branch-city	assets
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

2. customer 客戶(存款戶,貸款戶)

customer-name	customer-street	customer-city
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

3. depositor 存款戶

customer-name	account-number
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

Contents

7.1 Introduction

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

Logical Database Design

- Logical Database Design
  - **Semantic Modeling,** eg. E-R model (UNIT 6)
  - **Normalization**
- 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	Smith	20	London
.	.	.	.

P

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

SP

S#	P#	QTY
S1	P1	300
S1	P2	200

or

S'

S#	SNAME	STATUS
S1	Smith	20
S2	.	.
.	.	.

P

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

SP'

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

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

Supplier-and-Parts Database

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

SSP

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)}

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

?

S

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

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

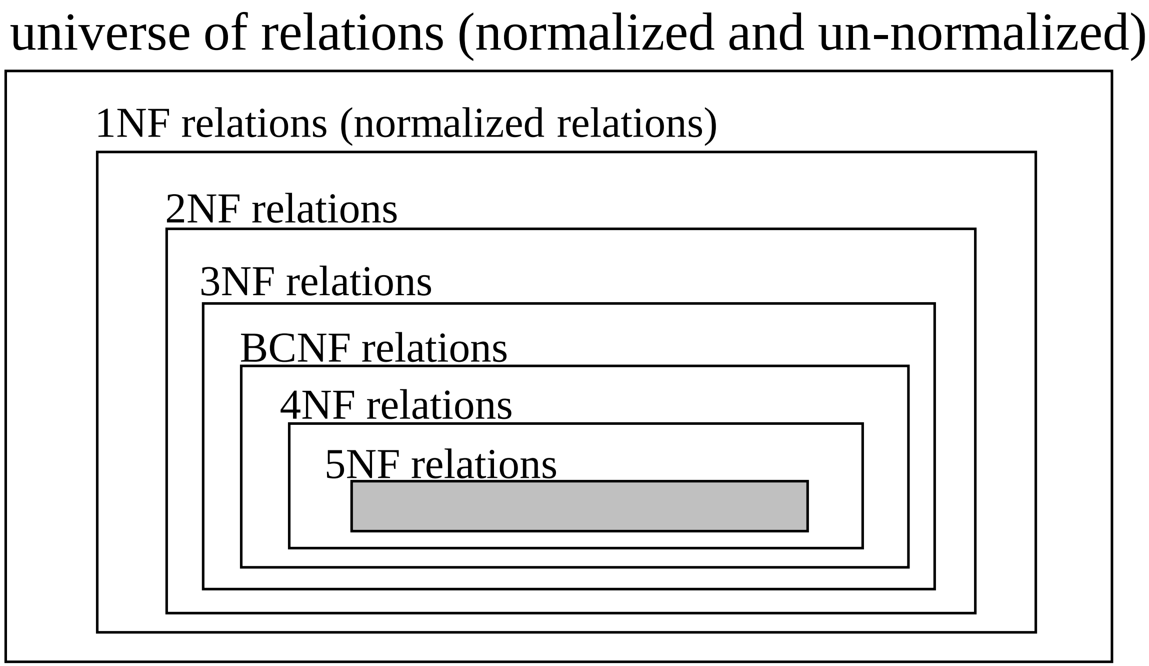
P

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

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 <u>only atomic values</u>.



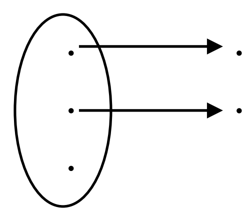
7.2 Functional Dependency

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

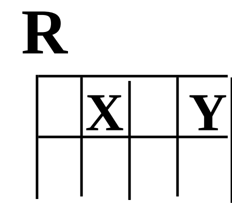
Functional Dependency

- 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.
- Notation:

$$R.X \longrightarrow R.Y$$



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



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

Functional Dependency (cont.)

<e.g.1>

S

S.S# $\longrightarrow$ S.SNAME

S.S# $\longrightarrow$ S.STATUS

S.S# $\longrightarrow$ S.CITY

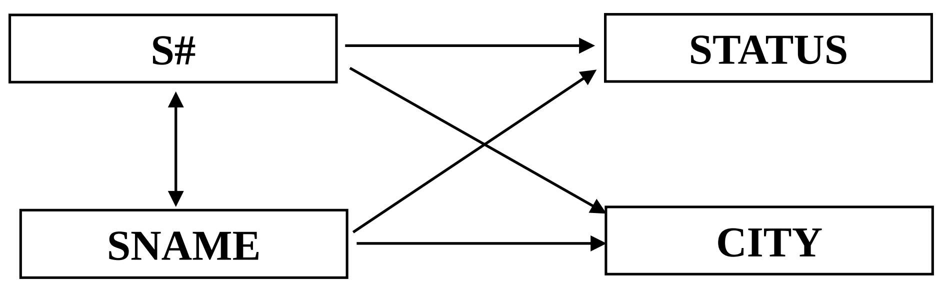
S.STATUS $\not\longrightarrow$ S.CITY

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

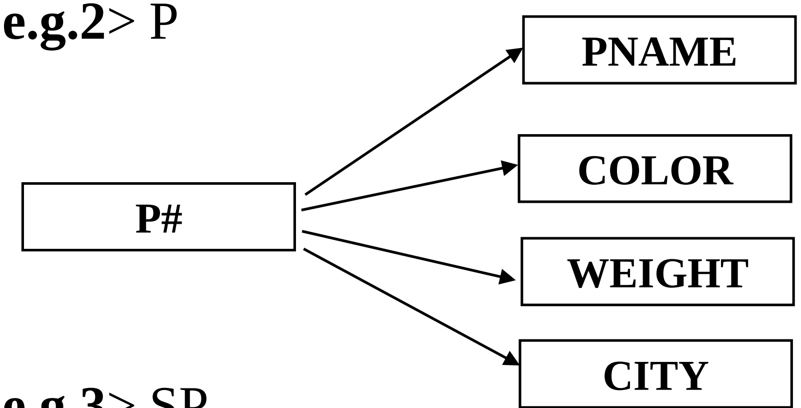
Note: Assume STATUS is some factor of Supplier and no any relationship with CITY.

FD Diagram:



Functional Dependency (cont.)

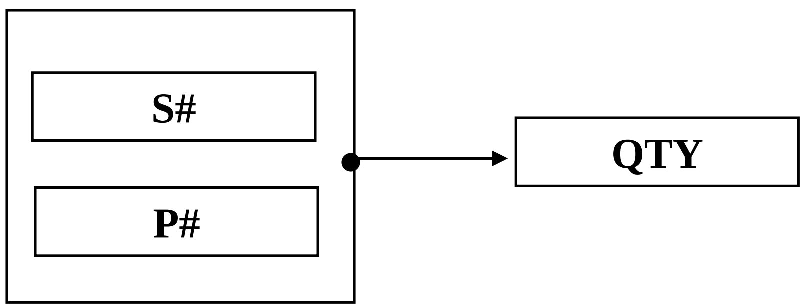
<e.g.2> P



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

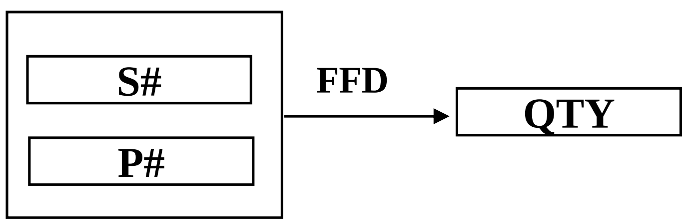
- If ***X*** is a candidate key of ***R***, then all attributes ***Y*** of ***R*** are functionally dependent on ***X***. (i.e. X $\longrightarrow$ Y)

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.

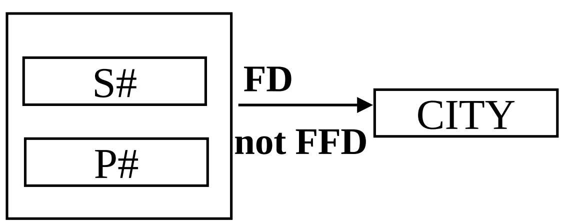
S#		$city_1$	$city_2$
S1		London	Taipei

<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 <u>semantic notion</u>.

<e.g.> S# $\longrightarrow$ 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# $\longrightarrow$ S.S# is not considered!

7.3 First Normal Form (1NF)

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

- Non-key attributes are NOT FFD on primary key. (e.g. QTY, STATUS, CITY in FIRST)

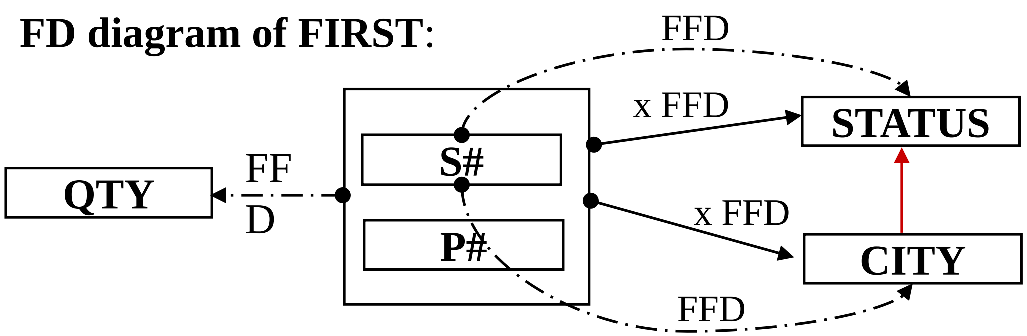
1NF Problem: Update Anomalies! (cont.)

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

2. STATUS is some factor of CITY (ref.p.7-10)

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#) $\not\xrightarrow[\text{FFD}]{}$ STATUS

primary key (S#, P#) $\not\xrightarrow[\text{FFD}]{}$ CITY

7.4 Second Normal Form (2NF)

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
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Normal Form: 2NF

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

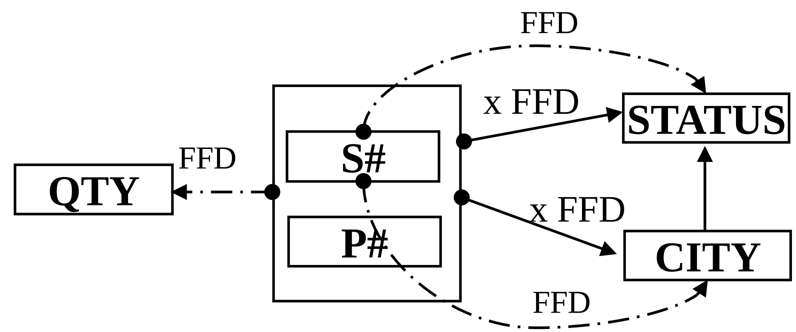
Key:(S#,P#),
Normalized 1NF

- 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. (QTY, STATUS, CITY in FIRST)

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

(S#, P#) $\underset{FFD}{\not\longrightarrow}$ STATUS, and

(S#, P#) $\underset{FFD}{\not\longrightarrow}$ CITY

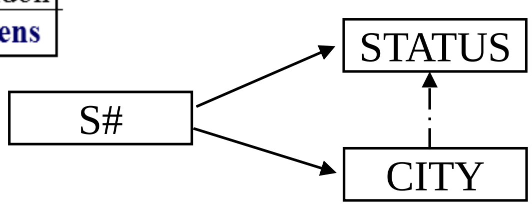


Decompose FIRST into:

SECOND (in 2NF)

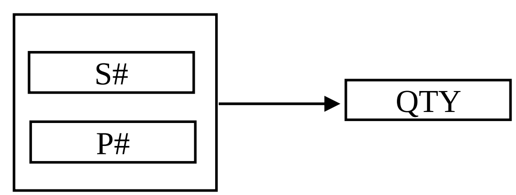
S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	**10**	**Paris**
S4	20	London
S5	**30**	**Athens**

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



FD:
1. S# $\rightarrow$ STATUS
2. S# $\rightarrow$ CITY
3. CITY $\rightarrow$ STATUS

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



FD: 4. (S#, P#) $\rightarrow$ QTY

SP (in 2NF)

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

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
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	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.

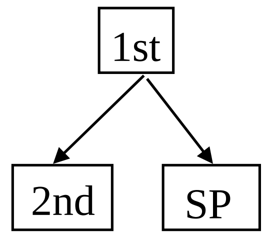
Normal Form: 2NF (cont.)

- 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)

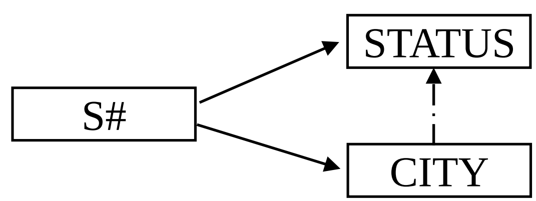
S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

- **Why:**

  S.S# $\longrightarrow$ S.STATUS

  S.S# $\longrightarrow$ S. CITY

  S.CITY $\longrightarrow$ S.STATUS cause a <u>transitive dependency</u>



FD:

1. S# $\rightarrow$ STATUS
2. S# $\rightarrow$ CITY
3. CITY $\rightarrow$ STATUS

7.5 Third Normal Form (3NF)

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

- Def : A relation R is in 3NF iff

  (1) R is in 2NF

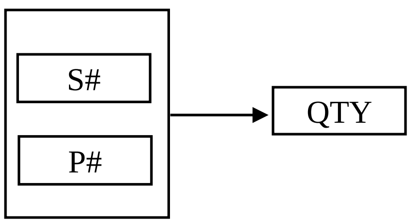
  (2) Every non-key attribute is non-transitively dependent on the primary key.

  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!

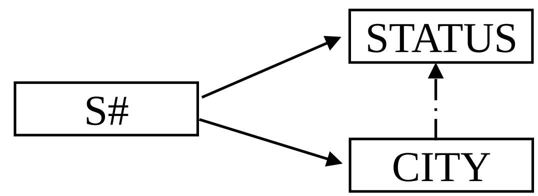
SP FD diagram



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:

S# → CITY

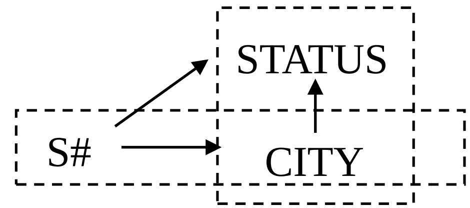
<2> **CS**(CITY, STATUS):

primary key: CITY
FD diagram:

CITY → STATUS

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)

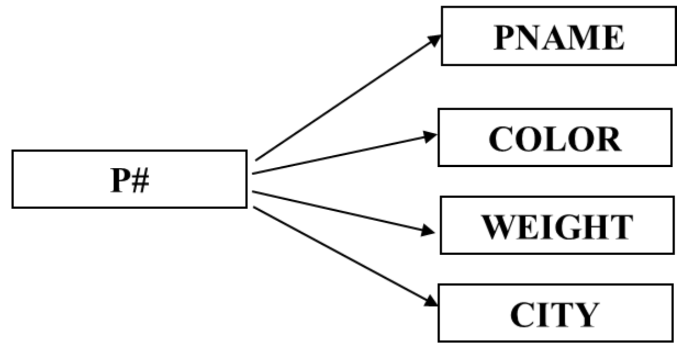
CITY	STATUS
Athens	30
London	20
Paris	10
Rome	50

Normal Forms: 3NF (cont.)

- Consider 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



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

- 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.

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

Normal Forms: 3NF (cont.)

- Note:

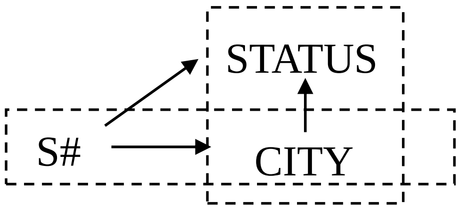
  (1) Any 2NF diagram can always be reduced to a collection of 3NF relations.

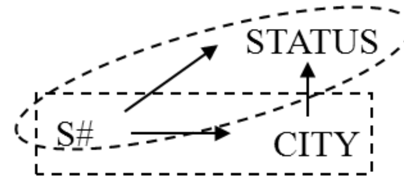
  (2) The reduction process from 2NF to 3NF is *non-loss* decomposition.

  (3) The collection of 3NF relations may contain "*more information*" than the equivalent 2NF relation.

SECOND

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





7.6 Good and Bad Decomposition

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

SC (in 3NF)

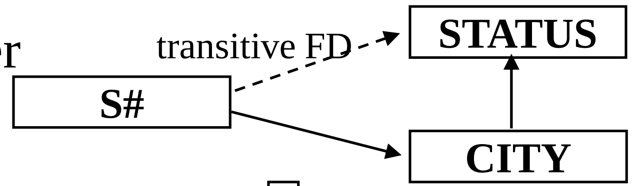
S#	CITY
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

SS (in 3NF)

S#	STATUS
S1	20
S2	10
S3	10
S4	20
S5	30

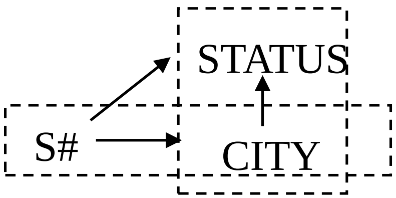
Good and Bad Decomposition

- Consider



Suppose 1. CITY is the main office of the supplier.

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

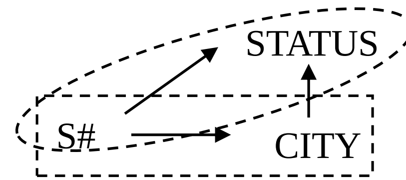


SC (in 3NF)

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

CS (in 3NF)

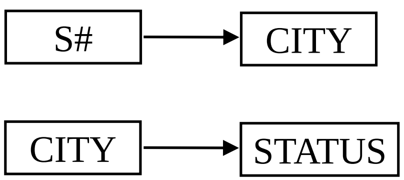
CITY	STATUS
Athens	30
London	20
Paris	10
Rome	50



① Decomposition **A:**

SC:

CS:



Good !

(Rome, 50) can be inserted.

SC (in 3NF)

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

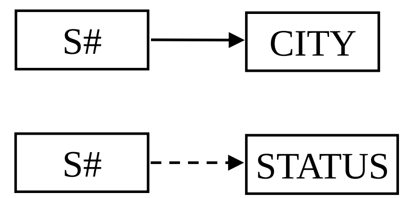
SS (in 3NF)

S#	CITY
S1	20
S2	10
S3	10
S4	20
S5	30

② Decomposition **B**:

SC:

SS:



Bad !

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

Good and Bad Decomposition (cont.)

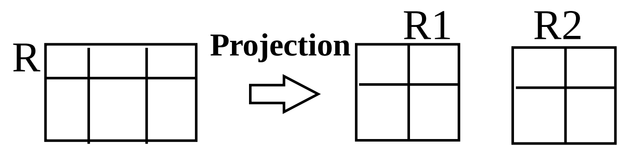
- Independent Projection: (by Rissanen '77, ref[10.6])

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.>



- <u>Decomposition **A**:</u>

SECOND(S#, STATUS, CITY) (R) $\longrightarrow$ $\begin{cases} \text{SC (S\#, CITY) (R1)} \\ \text{CS (CITY, STATUS) (R2)} \end{cases}$

(1) FD in SECOND (R):

S# $\longrightarrow$ CITY

CITY $\longrightarrow$ STATUS

S# $\dashrightarrow$ STATUS

FD in SC and CS

R1: S# $\longrightarrow$ CITY

R2: CITY $\longrightarrow$ STATUS

$\Rightarrow$ S# $\dashrightarrow$ STATUS

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

$\therefore$ SC and CS are independent $\rightarrow$ Decomposition **A** is good!

Good and Bad Decomposition (cont.)

Decomposition B:

SECOND(S#, STATUS, CITY) $\longrightarrow \begin{cases} \text{SC (S\#, CITY)} \\ \text{SS (S\#, STATUS)} \end{cases}$

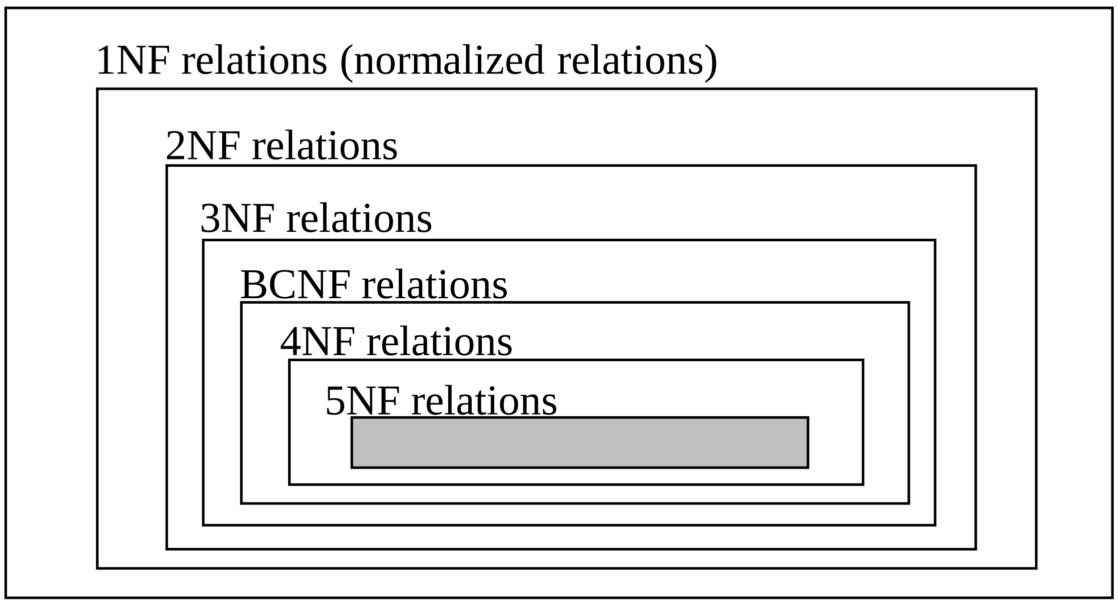
(1) FD in SC and SS:

$$\begin{matrix} S\# \longrightarrow CITY \\ S\# \longrightarrow STATUS \end{matrix} \;\not\Rightarrow\; CITY \dashrightarrow STATUS$$

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

Normal Forms

universe of relations (normalized and un-normalized)



Unit 18 More on Normalization

- 18.1 Introduction
- 18.2 Functional Dependency
- 18.3 First, Second, and Third Normal Forms (1NF, 2NF, 3NF)
- 18.4 Boyce/Codd Normal Form (BCNF)
- 18.5 Fourth Normal Form (4NF)
- 18.6 Fifth Normal Form (5NF)

EX.part2.2: Tables and SQL

- **Reduction E-R Model to Relational Tables**
  - Refer to UNIT 6, Sec. 7
  - Transfer your E-R model to Tables
- **Check each Table to see if it is a**
  - 1NF,
  - 2NF,
  - 3NF
- **Design Query**
  - Using SQL to define and create Tables
  - Design some queries to access your database
  - Using SQL to query your database
  - …

end of unit 7