SECD2523 DATABASE

TOPIC 6B | LOGICAL DATABASE DESIGN

Content adapted from Connolly, T., Begg, C., 2015. Database Systems: A Practical Approach to Design, Implementation, and Management, Global Edition. Pearson Education.

LECTURE LEARNING OUTCOME

By the end of this lecture, students should be able to:

01	**Define the purpose of normalization and how it supports database design**
02	**Identify potential problems associated with relational database design (data redundancy and update anomalies)**
03	**Define functional dependency and its association with normalization**
04	**Identify functional dependencies for a given relation**
05	**Define 1NF, 2NF, 3NF and BCNF**
06	**Perform normalization up until BCNF**

Normalization

- A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.
- The purpose of normalization is to identify a suitable set of relations that support the data requirements of an enterprise.
- The characteristics of a suitable set of relations include the following:
  - the **minimal number of attributes** necessary to support the data requirements of the enterprise.
  - attributes with a close logical relationship (describes as functional dependency) are found in the same relation.
  - **minimal redundancy** with each attribute represented only once with the important exception of attributes that form all or part of foreign keys, which are essential for the joining of related relations.

Benefit of Normalization

- Minimize data **redundancies** in a database, thus will reduce storage space required to store the data
- Reduce data **anomalies**
- Easy for user to **access** data from a database
- Easy for user to **maintain** data

How Normalization Supports Database Design

- 2 approaches for using normalization:
  - Approach 1 – normalization can be used as a **bottom-up** standalone database design technique.
  - Approach 2 – normalization can be used as a **validation** technique to check the structure of relations, using a top-down approach such as ER modeling.

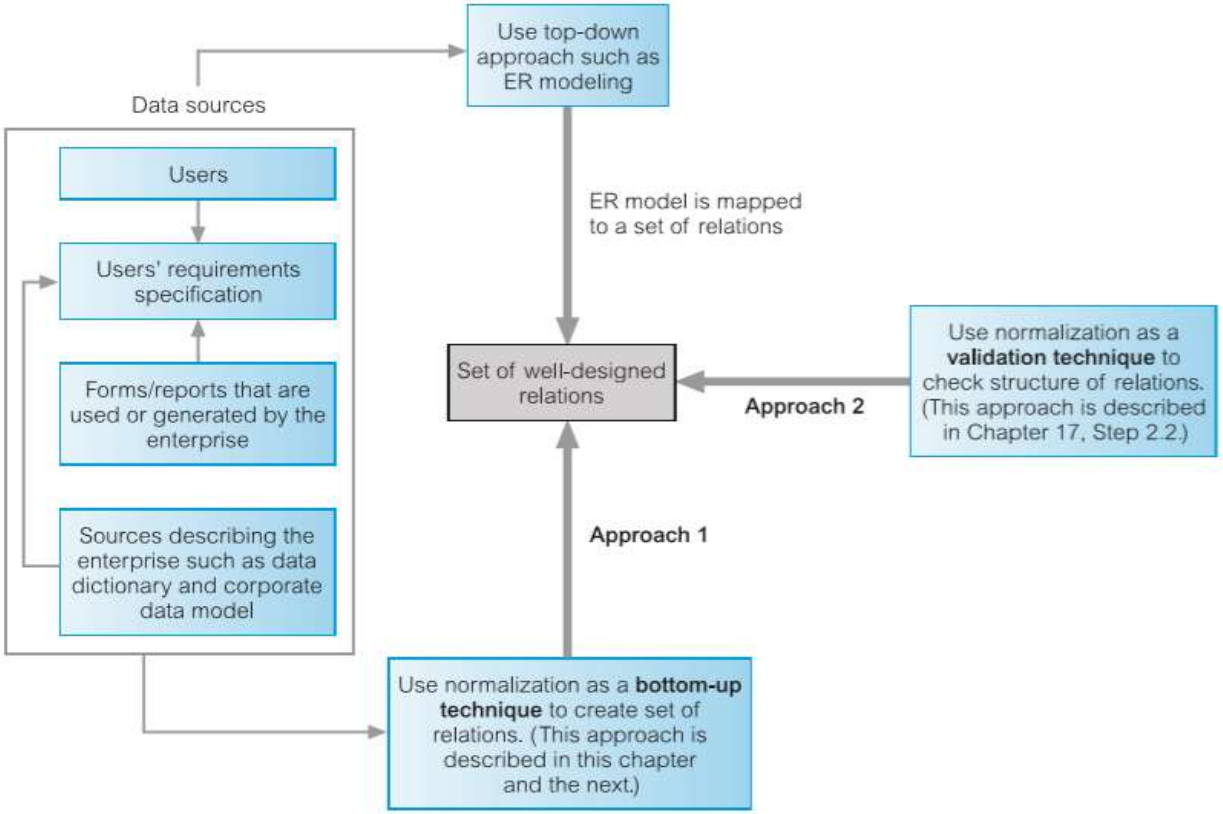
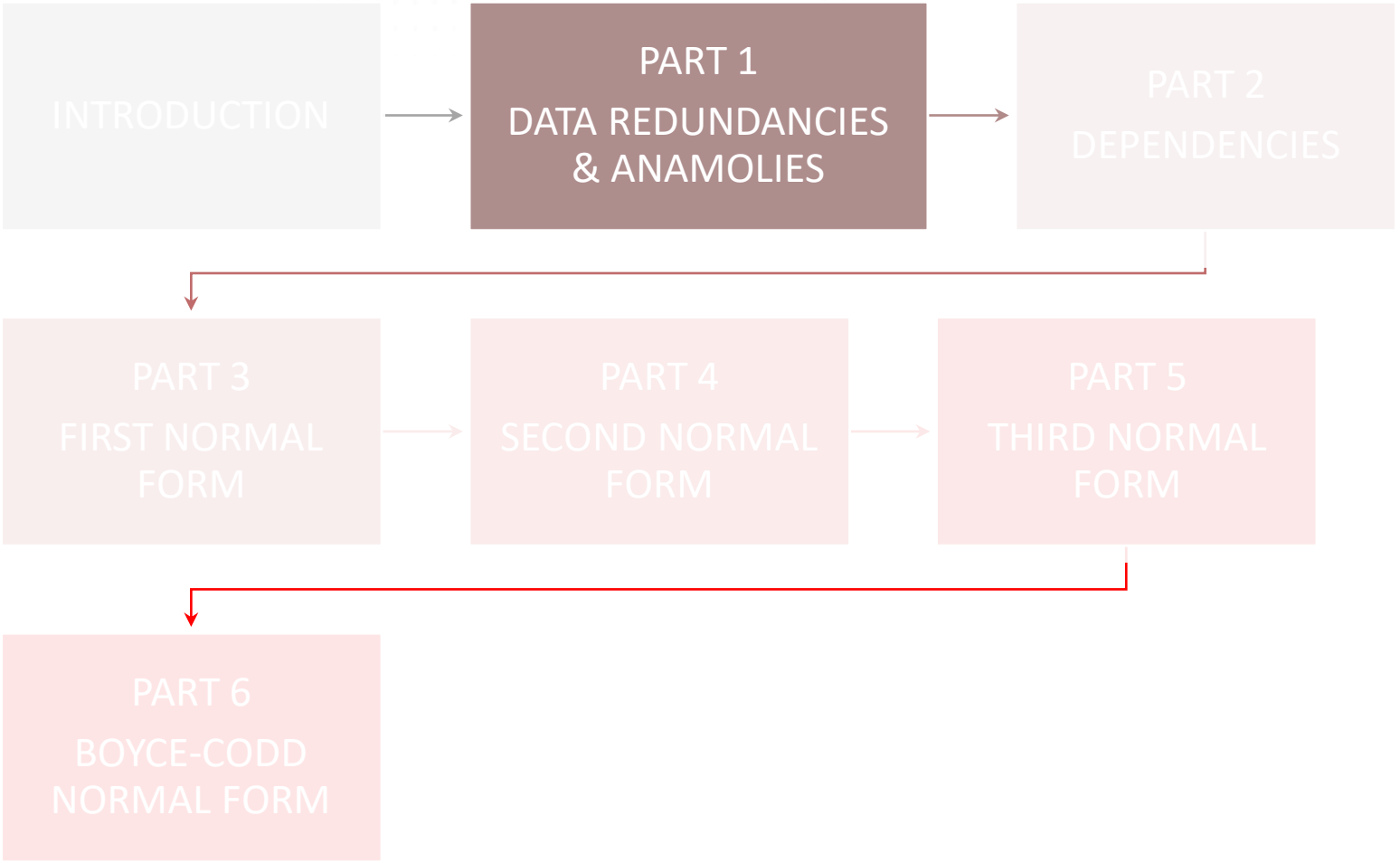


Figure 14.1 How normalization can be used to support database design.

PART 1: DATA ANOMALIES



Before we start…

Need to understand two concepts:

1) DATA REDUNDANCY

2) UPDATE ANOMALIES

To understand the two concepts, we will be referring to the situation below:

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Situation 1	VS	Situation 2
Both information of staffs and branch are in *StaffBranch* relation		Information of staffs in *Staff* relation Information of branch in *Branch* relation

1) DATA REDUNDANCY

- Data redundancy = **repeated details** for the same data
- To **minimize** data redundancy group attributes into relations ii Relational Database (RD).
- This would give benefits for the implemented database, such as:
  - **Minimal** number of operations when update data
  - Reduce the opportunities for data **inconsistencies**
  - Reduce file storage and minimize **costs**

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Situation 1

Both information of staffs and branch are in *StaffBranch* relation

1) DATA REDUNDANCY

- However, certain amount of data redundancy is **required** in Relational Databases
  - copies of Primary Keys (or candidate keys) acting as **Foreign Keys** in related relations to enable the modeling of relationships between data.

Situation 2

Information of staffs in *Staff* relation
Information of branch in *Branch* relation

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Foreign key

Branch

Primary key

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

2) UPDATE ANOMALIES

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Figure 1

Unwanted data redundancy may cause **update anomalies**

For update anomalies, we will look at SITUATION 1:

Look at tuples in row 2, 3 and 5. The details of a branch (B003) are repeated for every member of staff located at that branch data redundancy, hence defeat one of the purpose of building a database.

Category of update anomalies :

- **Insertion anomalies**
- **Deletion anomalies**
- **Modification anomalies**

UPDATE ANOMALIES: Insertion Anomalies

- There are two main types of insert anomalies:

1: To insert the details of new members of staff into the StaffBranch relation

Anomaly due to the design of the relation that cause **difficulties during inserting new data**

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London
SG11	Andy Cole	Supervisor	18000	B007	16 Argyl Rd, London

Argyl > Argyll

Our database will show 2 different address for B007!!

Insertion Anomalies

2) To insert details of a new branch that currently has no members of staff into the StaffBranch relation

Anomaly due to the design of the relation that cause **difficulties during inserting new data**

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London
X	X	X	X	B009	19A McKeon Ave, London

We defined staffNo is a PRIMARY KEY

PK – entity integrity (NO NULL)

Deletion Anomalies

- **If we delete a tuple from the StaffBranch relation that represents the last member of staff located at a branch, the details about that branch are also lost from the database.**

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

We will also delete B007 data (unwanted data loss) because the Branch still exist!!

Anomaly due to the design of the relation that cause **difficulties during deletion of data. Might cause unwanted data loss.**

Modification Anomalies

- **If we want to change the value of one of the attributes of a particular branch in the StaffBranch relation—for example, the address for branch number B003**

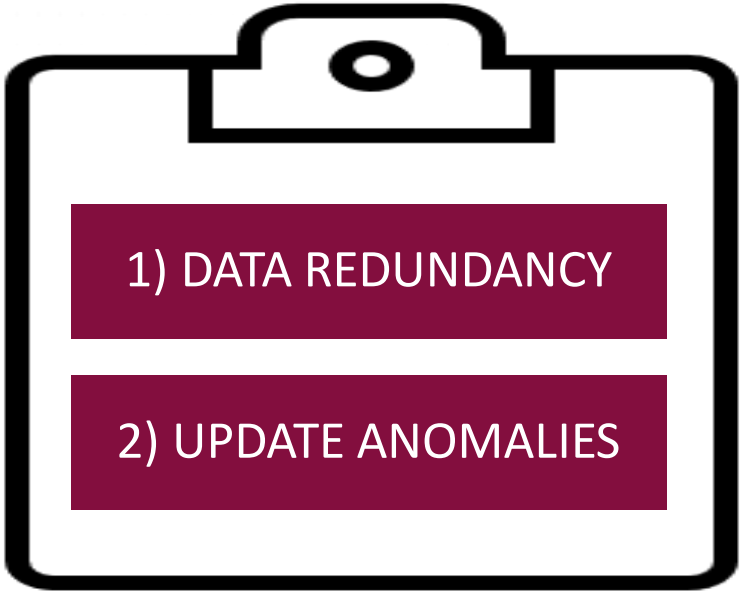
Anomaly due to the design of the relation that cause **difficulties during modification of existing data. Might lead to inconsistency of the database.**

StaffBranch

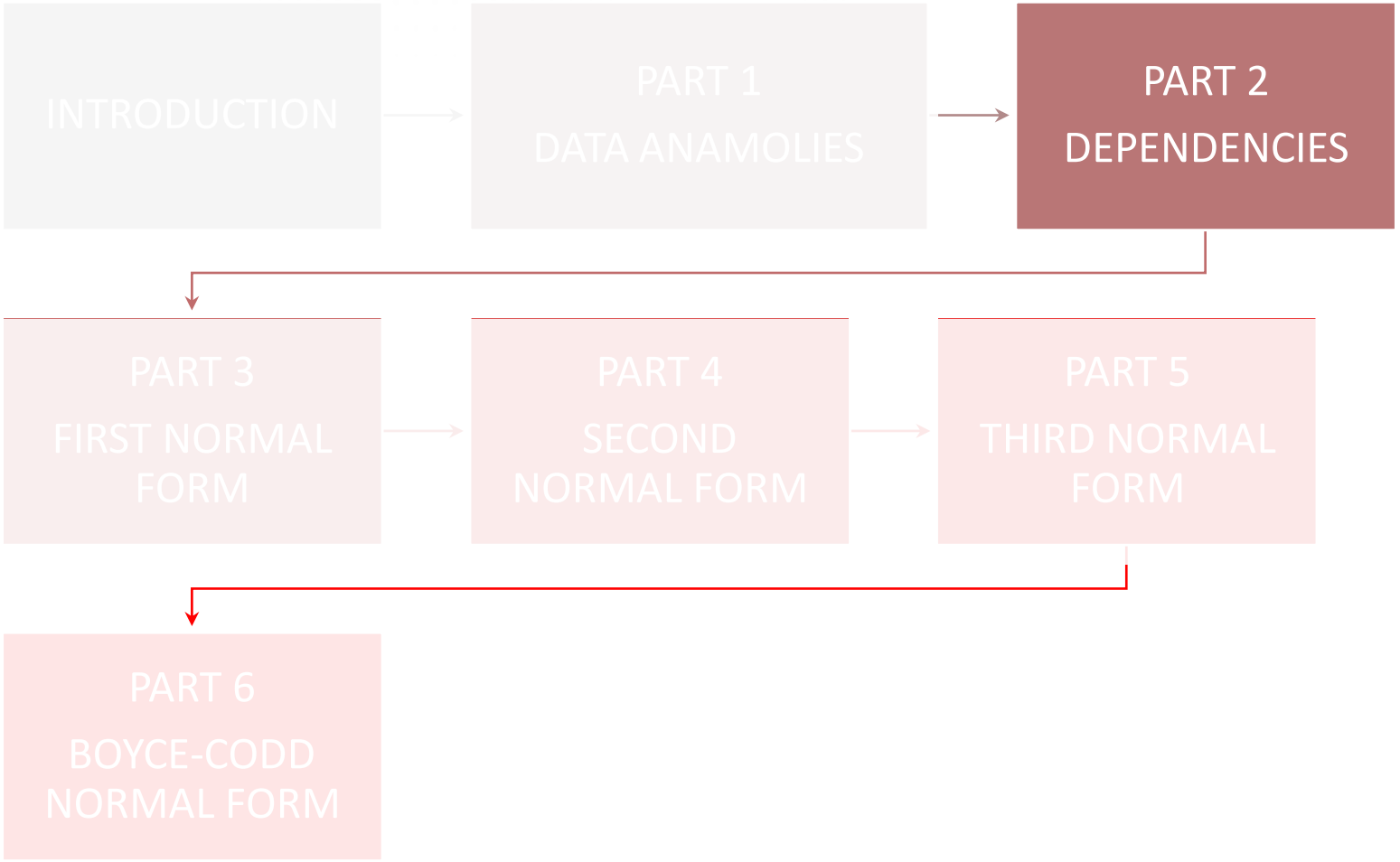
staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	19 McKeon Ave, London
SG14	David Ford	Supervisor	18000	B003	19 McKeon Ave, London
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Accidently forgot to update.
Now we have 2 different address for B003!!

You should understand



PART 2: DEPENDENCIES

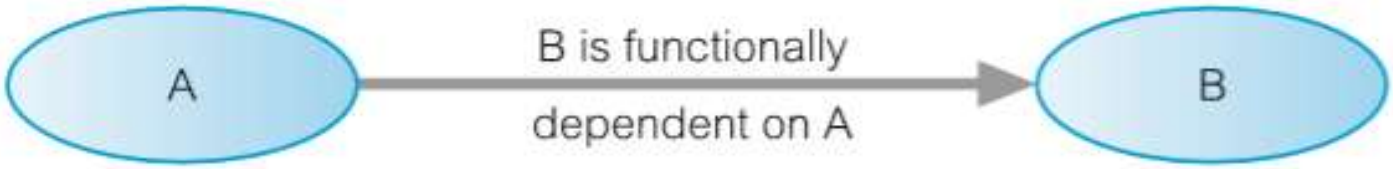


Functional Dependencies (FD)

- An important concept associated with normalization is functional dependency, which describes the relationship between attributes (Maier, 1983).
  - if A and B are attributes of relation R, B is functionally dependent on A (written as: A → B) if each value of A is associated with exactly one value of B. (A and B may each consist of one or more attributes.)
- When a functional dependency is present, the dependency is specified as a constraint between the attributes.
- An alternative way → 'A functionally determines B'

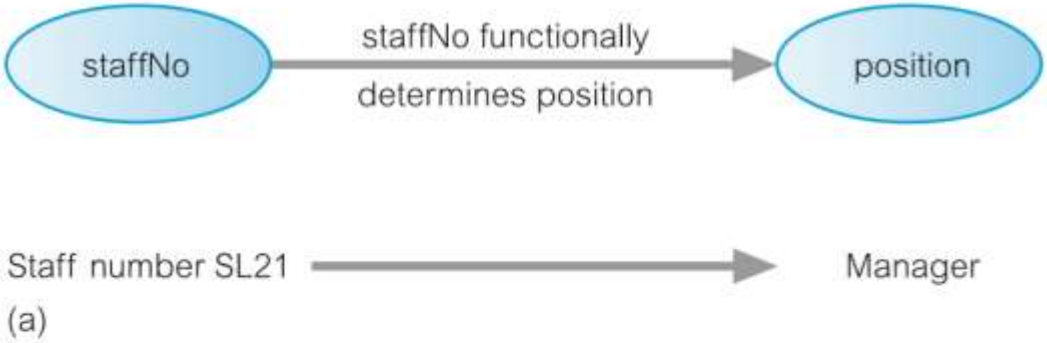
FD: Determinant

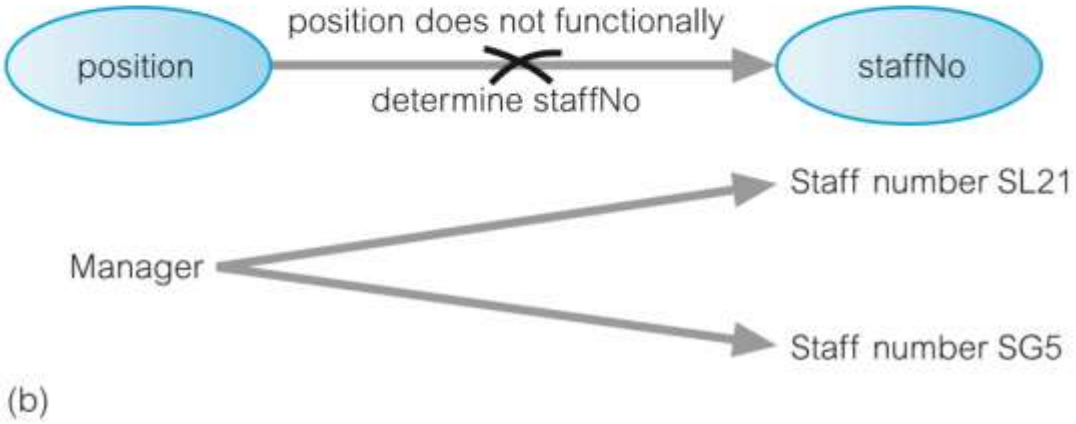
- Determinant: Refers to the attribute, or group of attributes, on the left-hand side of the arrow of a functional



determinant

Example of a Functional Dependency





staffNo functionally determines *position*

- staffNo → position
- Relationship between *staffNo* and *position* is 1:1

position does *not* functionally determine *staffNo*

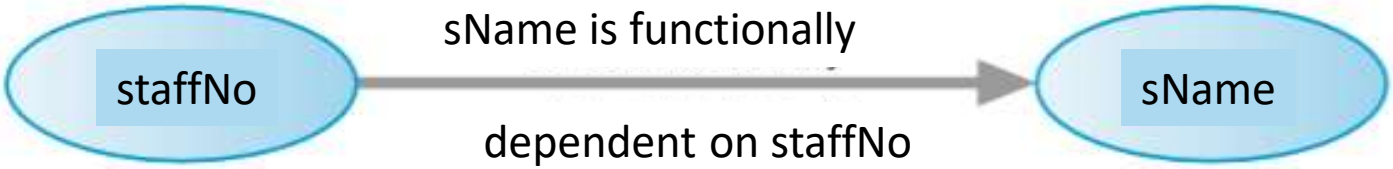
- position ↛ staffNo
- relationship between *position* and *staffNo* is 1:*

Functional Dependencies

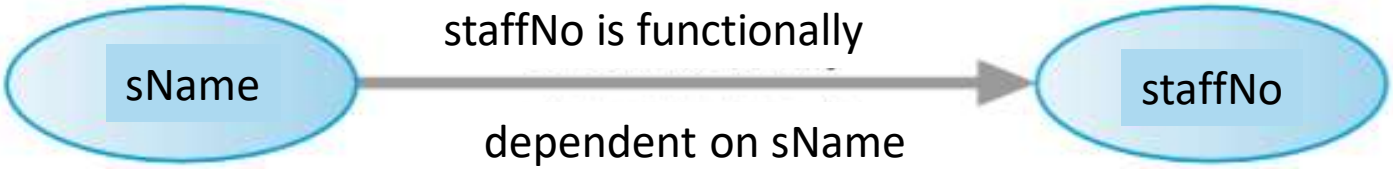
- In normalization we are only interested in FD that "**holds at all time**"
- Indicate a **1:1 relationship** between attributes
- The reason is that we want to identify functional dependencies that hold for all possible values for attributes of a relation as these represent the types of integrity constraints that we need to identify.
- Approach → by understanding the purpose of each attribute in an identified relation.

FUNCTIONAL DEPENDENCIES [Discussion]

- Which of these two FDs **hold at all times**??
  1. staffNo attribute functionally determines the sName attribute
     - staffNo → sName



  2. sName attribute functionally determines the staffNo attributes
     - sName → staffNo



Functional Dependencies

- Characteristics of functional dependencies use in normalization
  - One-to-one relationship exists between the attribute(s) on the left-hand side (determinant) and those on the right-had side of a functional dependency.
  - They hold for all time
  - The determinant has the minimal number of attributes necessary to maintain the dependency with the attribute(s) on the right-hand side → there must be a full functional dependency between the attributes(s) on the left-hand and right-hand sides of the dependency.

Full Functional Dependency

1. Full functional dependency: Indicates that if A and B are attributes of A relation, B is **FULLY FUNCTIONALLY DEPENDENT** on A if B is functionally dependent on A, but not on any proper subset of A.

StaffBranch (staffNo, sName, position, salary, branchNo, bAddress)

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Full Functional Dependency

2) A functional dependency A B is a FULL FUNCTIONAL DEPENDENCY if removal of any attribute from A results in the dependency no longer existing.

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Partial Dependency

A functional dependency A → B is a PARTIAL DEPENDENCY if there is some attribute that can be removed from a and yet the dependency still holds.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Transitive Dependency

A condition where A, B, and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on a via B (provided that a is not functionally dependent on B or C).

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Transitive Dependency

branchNo → bAddress

staffNo → position, salary, branchNo, bAddress)

staffNo
SL21
SG37
SG14
SA9
SG5
SL41

branchNo	bAddress
B005	22 Deer Rd, London
B003	163 Main St, Glasgow
B003	163 Main St, Glasgow
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow
B005	22 Deer Rd, London

- The transitive dependency branchNo → bAddress exists on staffNo via branchNo.
- The staffNo attribute functionally determines the bAddress via the branchNo attribute.
- Neither branchNo nor bAddress functionally determines staffNo.

Identifying Functional Dependencies

- Identifying FDs requires understanding of the meaning and purpose of each attributes and the relationships between the attributes.
- Example:

  StaffBranch (staffNo, sName, position, salary, branchNo, bAddress)

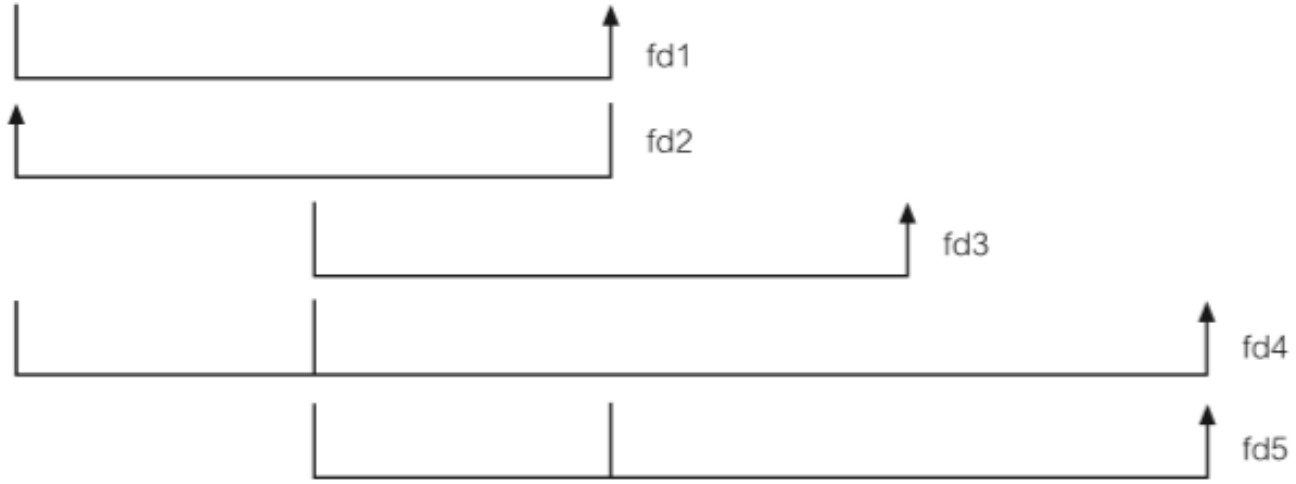
  (Note: for the purpose of discussion, assume that the position held, and the branch can determine the salary)

- FDs:
  - staffNo → sName, position, salary, branchNo, bAddress
  - branchNo → bAddress
  - bAddress → branchNo
  - branchNo, position → salary
  - bAddress, position → salary

Identifying Functional Dependencies

Sample Relation

A	B	C	D	E
a	b	z	w	q
e	b	r	w	p
a	d	z	w	t
e	d	r	w	q
a	f	z	s	t
e	f	r	s	t



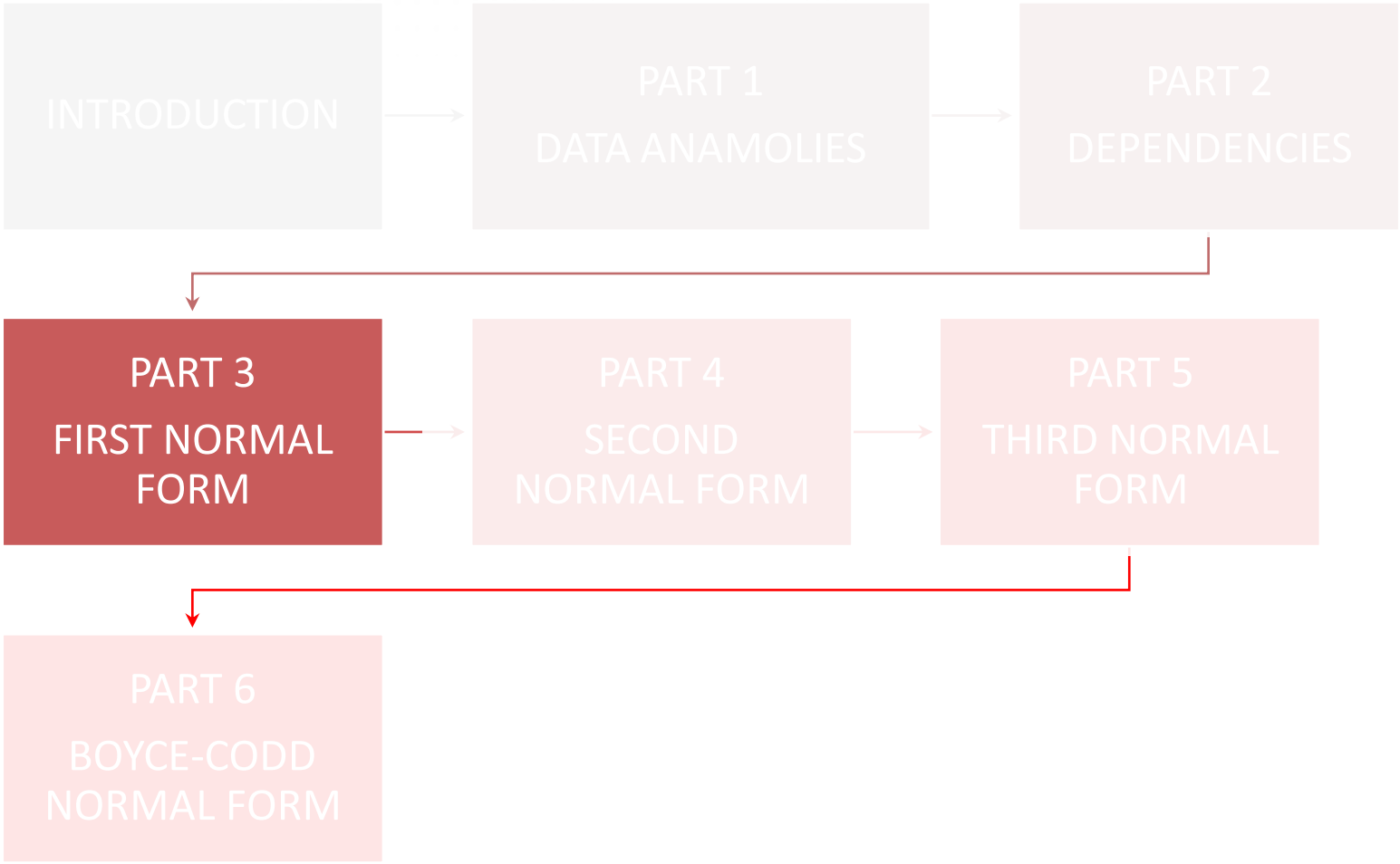
The sample relation displaying data for attributes A,B,C,D and E and the functional dependencies (fd1 to fd4) that exist between these attributes.

A → C (fd1)
C → A (fd2)
B → D (fd3)
A, B → E (fd4)
B, C → E (fd5)

[SUMMARY]
Identifying Functional Dependency

- To identify the functional dependencies that exist between attributes a, B, C, D, and E, we examine the Sample relation shown in Figure 14.6 and identify when values in one column are consistent with the presence of a particular value in other columns.
- We begin with the first column on the left-hand side and work our way over to the right-hand side of the relation and then we look at combinations of columns; in other words, where values in two or more columns are consistent with the appearance of values in other columns.

PART 3: FIRST NORMAL FORM



The Process of Normalization

- Formal technique for analyzing relations based on their primary key (or candidate keys) and functional dependencies.
- Involves a series of rules that can be used to test each individual relations so that a database can be normalized to any degree.
- A database is said to be a normalized database if all relations are in the “highest” normal form (usually 3NF onwards)
- When a requirement is not met, the relation violating the requirement must be decomposed into relations that individually meet the requirements of normalization.
- Checking rules begin with from 1NF, moving upward until at least 3NF (in this class we will cover until BCNF).

The Process of Normalization

- Three normal forms:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)

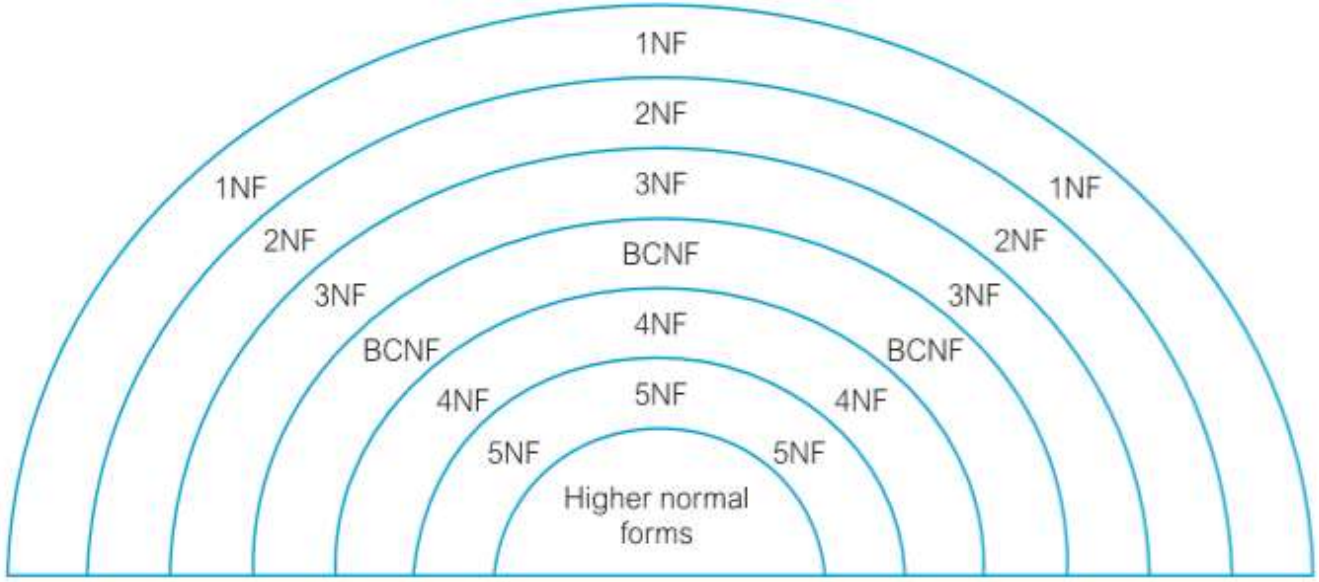


Figure 14.7 Diagrammatic illustration of the relationship between the normal forms.

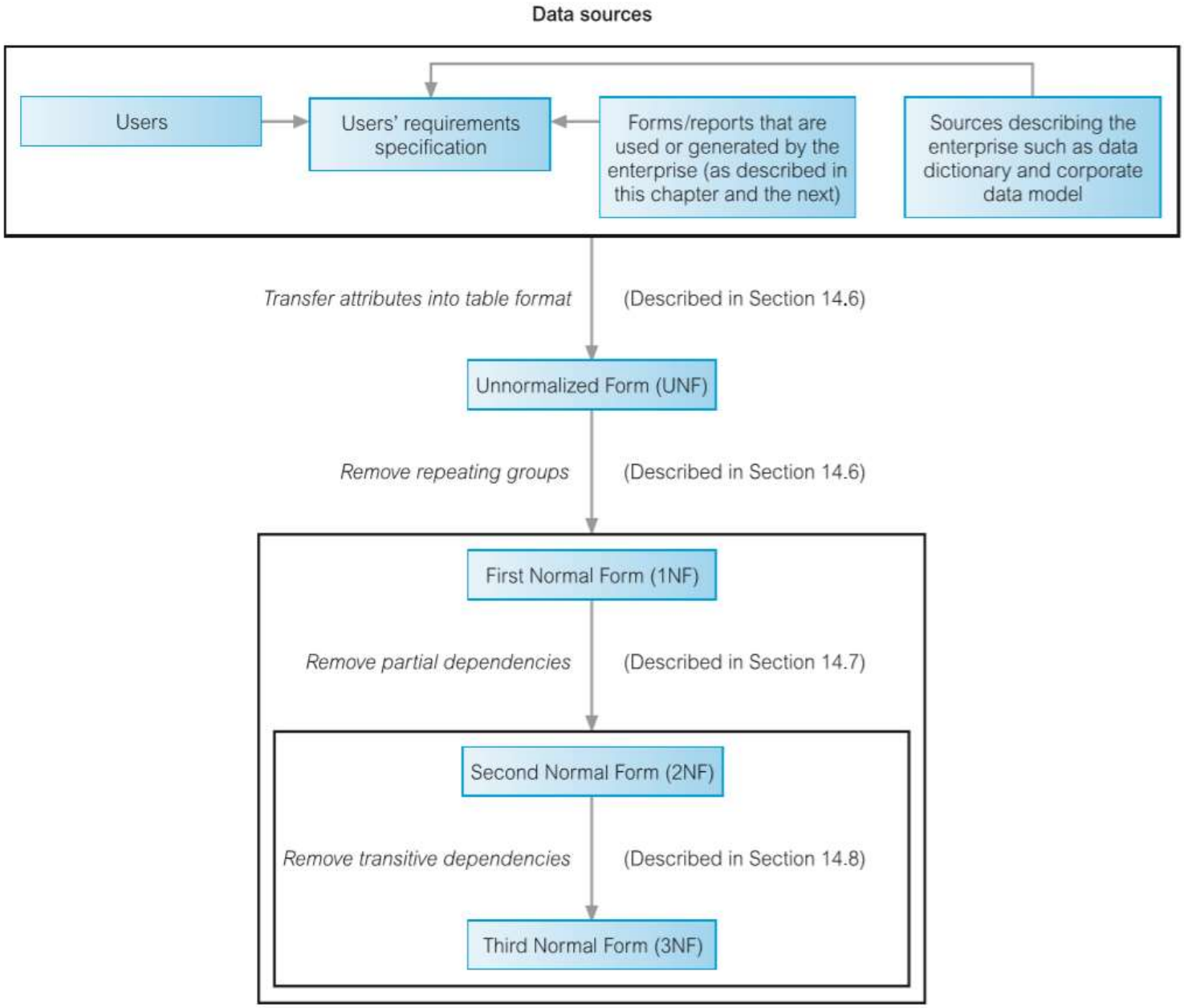


Figure 14.8 Diagrammatic illustration of the process of normalization.

First Normal Form (1NF)

- Before the process of normalization, data from source table could be in **unnormalized form (UNF)**, which is a table that contains one or more **repeating groups**.
- **1NF** – A relation in which the intersection of each row and column contains **one and only one value**.
- Repeating group – is an attribute or group of attributes within a table that occurs with multiple values for a single occurrence of the nominated key attribute(s) for that table.
- Two Approach to removing repeating groups from UNF:-
  - By entering appropriate data in the empty columns of rows containing the repeating data (1st approach)
  - By placing the repeating data, along with a copy of the original key attributes(s) in a separate relation. (2nd approach)

UNF: Example

Collection of DreamHome leases

Repeating Groups

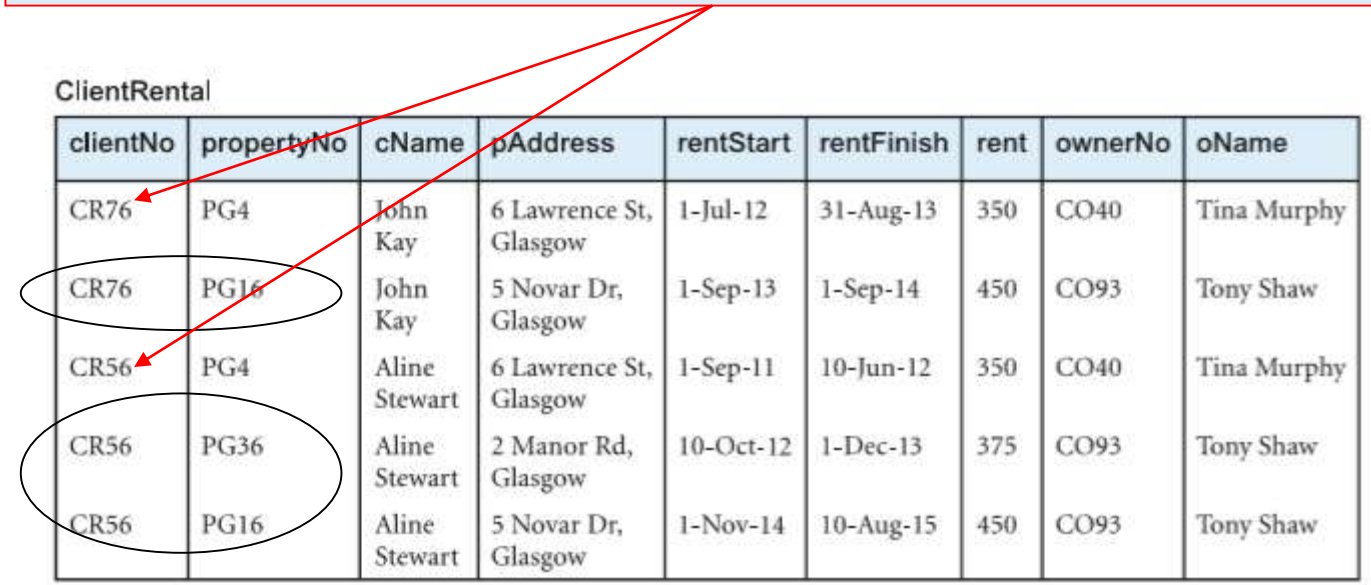
ClientRental

clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	50	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-June-12	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

ClientRental Unnormalized table

1NF Example: 1st Approach

Remove repeating group in UNF table (property rented details) by entering the appropriate client data into each row.



ClientRental

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

1NF relation schema:

ClientRental (clientNo, propertyNo, cName, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

1NF Example: 2nd Approach

Remove repeating group (property rented details) in 1NF table with a copy of the original key attribute(i.e. the PK - clientNo) by replacing them in a separate relation from the non-repeating group relation. Name both relations appropriately.

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

2 newly formed 1NF relations with a single value at the intersection of each row:

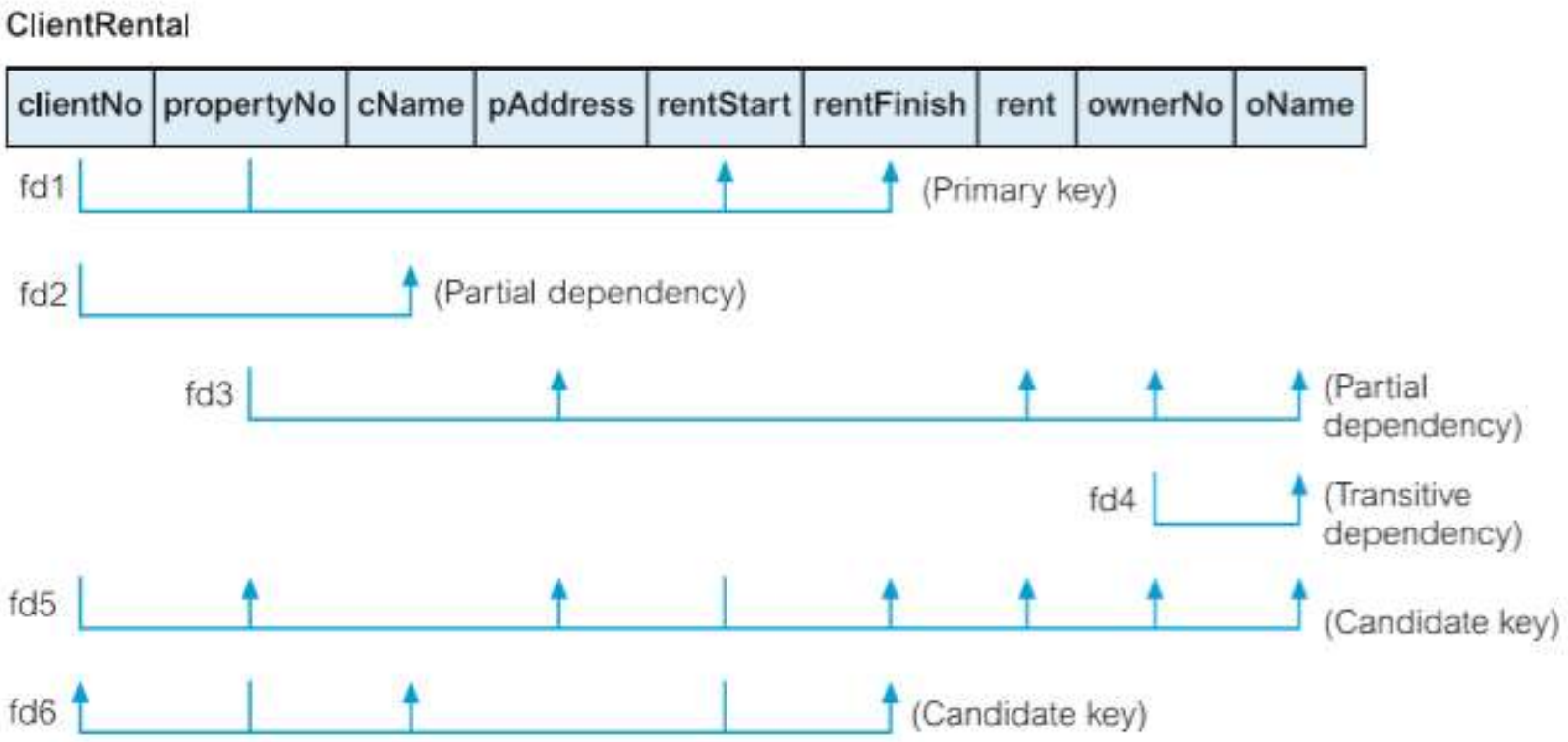
PropertyRentalOwner

clientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

1NF relation schema:

Client (clientNo, cName)
PropertyRentalOwner (clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

1NF: Determine Primary Key (PK)



ClientRental (<u>clientNo</u>, <u>propertyNo</u>, cName, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

First Normal Form (1NF)

Important Notes:

If you are using normalization as a validation technique to validate relations that are derived from conceptual ERM, we can safely say that the derived relations are already in 1NF, where the assumption is a row of record in the relation is considered as one complete tuple.

Already in 1NF

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Starts with UNF

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

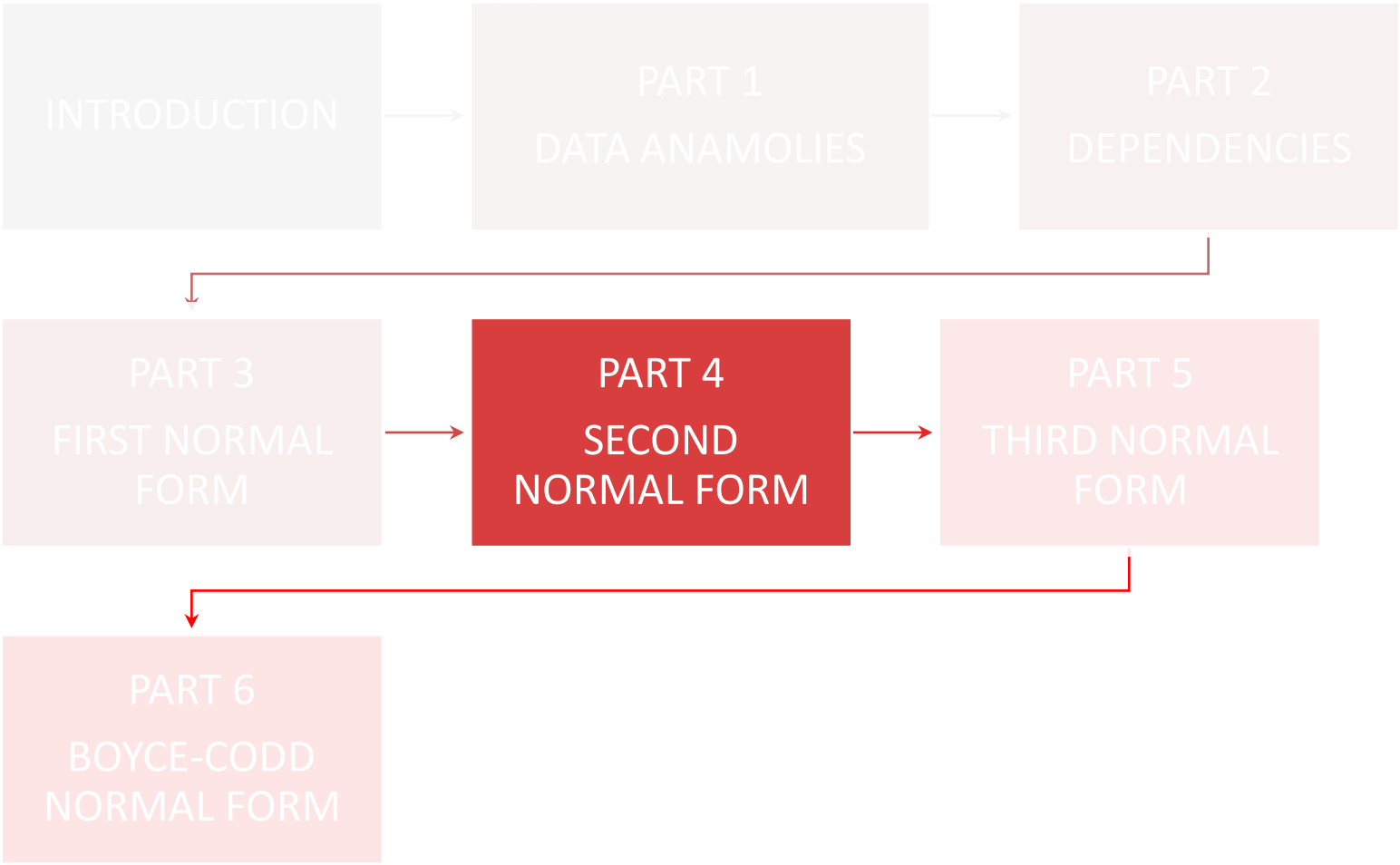
Summary

- **1NF: removing repeating group**

  Identify PK & dependencies within the relation

- 2NF: removing partial dependency
- 3NF: removing transitive dependency
- BCNF: removing non-candidate key determinant

PART 4: SECOND NORMAL FORM



Second Normal Form (2NF)

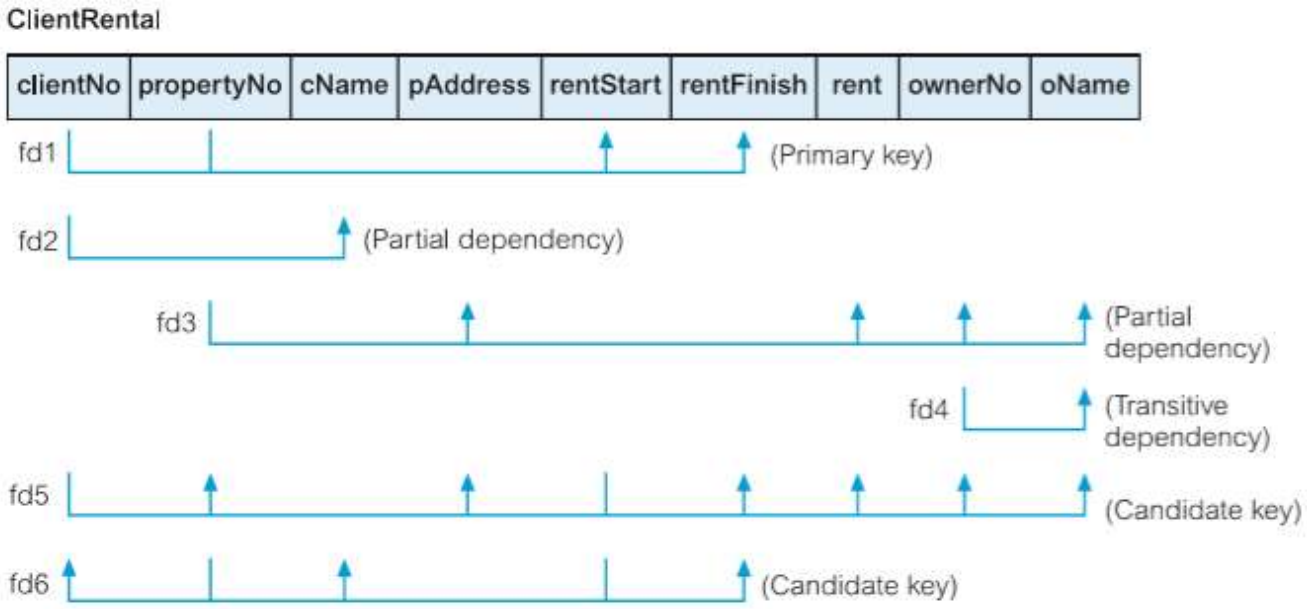
- **2NF:** A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key.
- A 1NF relation with a single attribute PK is automatically a 2NF relation.
- A 1NF relation with composite PK MAY or MAY NOT BE a 2NF relation.
- The normalization of 1NF relations to 2NF involves the removal of partial dependencies.
  - If a partial dependency exists, remove the partially dependent attribute(s) from the relation by replacing them in a new relation along with a copy of determinant

1NF to 2NF: What to do?

- Identify the functional dependencies in the relation.
- Observe the FDs, if partial dependencies exist on the primary key, move all partial dependencies attributes from the relation by placing them in a new relation along with a copy of their determinant.

2NF Example

- Let us use the 1NF ClientRental relation (obtained from the 1st approach).
  - list its FDs
  - determine the PK for ClientRental



2NF Example

FDs list in ClientRental

FD1 : clientNo, propertyNo → rentStart, rentFinish

FD2 : clientNo → cName

FD3 : propertyNo → pAddress, rent, ownerNo, oName

FD4 : ownerNo → oName

FD5 : clientNo, rentStart → propertyNo, pAddress, rentFinish, rent, ownerNo, oName

FD6 : propertyNo, rentStart → clientNo, cName, rentFinish

2NF Example

Identify FDs with existence of partial dependencies on PK

1. Is there any FD such that the determinant is a subset of PK?
2. If so, then that FD is not a Full FD, hence the relation violates the 2NF rule.

PK is clientNo, propertyNo

Subsets of PK:

{clientNo}, {propertyNo}, {clientNo, propertyNo}

determinant in FD2 and FD3 is subset of PK

FDs list in ClientRental

FD1 : clientNo, propertyNo → rentStart, rentFinish

FD2 : clientNo → cName (*partial dependency)*

FD3 : propertyNo → pAddress, rent, ownerNo, oName (*partial dependency)*

FD4 : ownerNo → oName

FD5 : clientNo, rentStart → propertyNo, pAddress, rentFinish, rent, ownerNo, oName

FD6 : propertyNo, rentStart → clientNo, cName, rentFinish

2NF Example

3. Move all partial FDs attributes from the relation
   i. Create new relation for each partial FDs
   ii. Name all relations appropriately

2NF Relations:

From FD2: **Client (clientNo, clientName)**

From FD3: **PropertyOwner (propertyNo, pAddress, ownerNo, oName)**

The rest : **Rental (clientNo, propertyNo, rentStart, rentFinish)**

These are our latest set of relations. All of them are now in 2NF as every non-PK attribute is fully FD on the PK of the relation

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

Figure 14.14 Second normal form relations derived from the ClientRental relation.

Summary

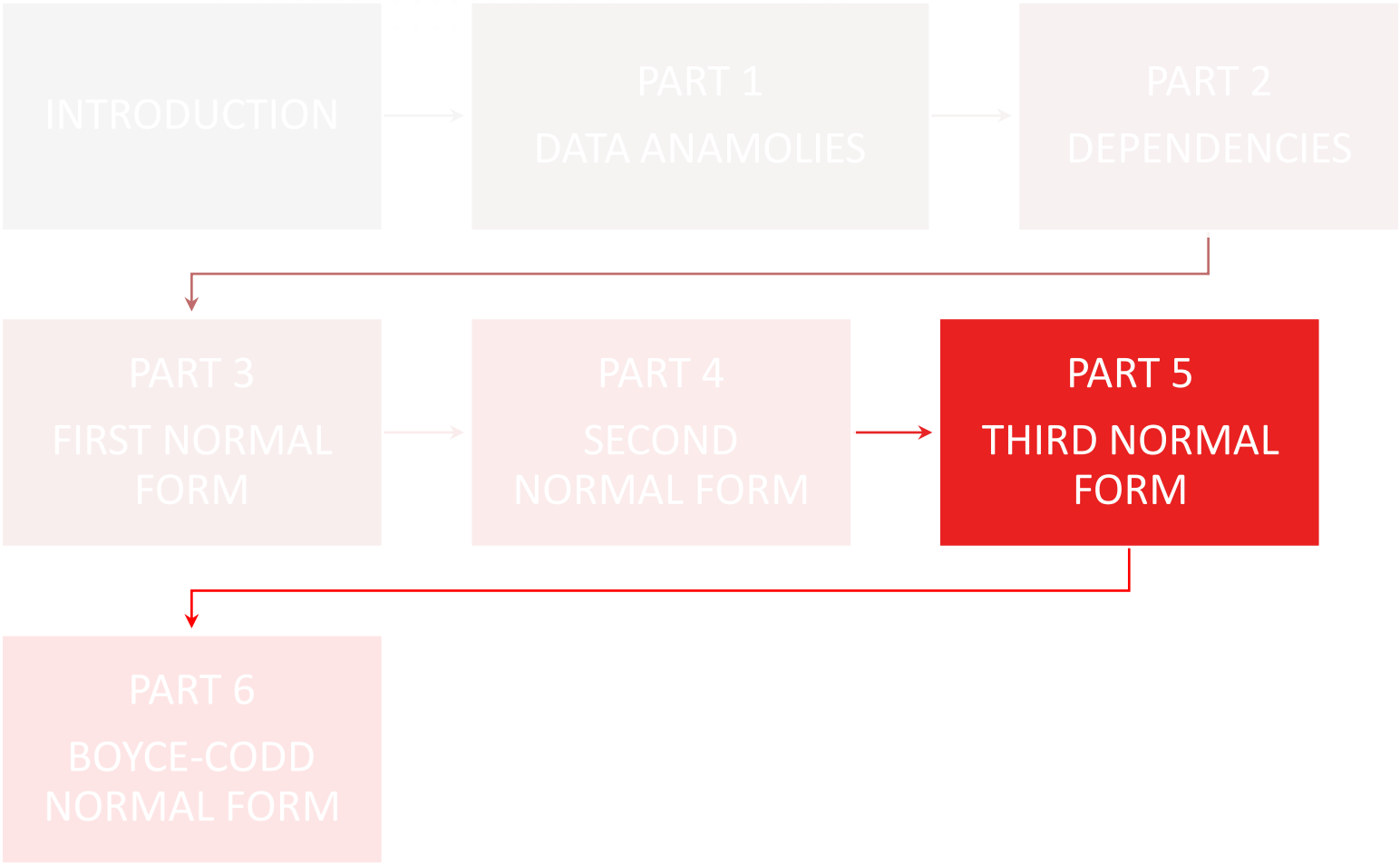
1NF: removing repeating group

2NF: removing partial dependency

3NF: removing transitive dependency

BCNF: removing non-candidate key determinant

PART 5: THIRD NORMAL FORM



Third Normal Form (3NF)

- **3NF**: A relation that is in 1NF and 2NF and in which no non-primary key attribute is **transitively dependent** on the primary key.
- 2NF relations may still suffer from **update anomalies**.
- This update anomaly is caused by a transitive dependency.
- The normalization of 2NF into 3NF involves the removal of transitive dependency.
- If a transitive dependency exists, move the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

2NF to 3NF: What to do?

- List FDs for all 2NF relations.
- Examine each FDs by looking for transitive dependencies
- If transitive dependencies exist on the primary key, remove them by placing them in a new relation along with a copy of their determinant.

3NF: Example

2NF relations:

Client (clientNo, clientName)

i. FD2 : clientNo → cName

Rental (clientNo, propertyNo, rentStart, rentFinish)

i. FD1 : clientNo, propertyNo → rentStart, rentFinish

ii. FD5' : clientNo, rentStart → propertyNo, rentFinish

iii. FD6' : propertyNo, rentStart → clientNo, rentFinish

PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)

i. FD3 : propertyNo → pAddress, rent, ownerNo, oName

ii. FD4 : ownerNo → oName *(Transitive dependency)*

Look at PropertyOwner relation:

→ In FD3, propertyNo → ownerNo and propertyNo → oName

→ In FD4, ownerNo → oName

→ oName is TD on propertyNo via ownerNo

3NF: Example

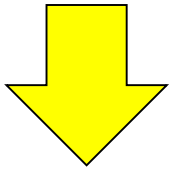
- Move transitive dependency attributes from PropertyOwner by placing the attributes in a new relation
  - Owner (ownerNo, oName)
- Rename the original relation (PropertyOwner) appropriately and place the remaining attributes
  - PropertyForRent (propertyNo, pAddress, rent, ownerNo)

3NF: Example

PropertyOwner (2NF)

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw



PropertyForRent (3NF)
Owner (3NF)

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

3NF: Example

- These are our final set of normalized relations in the 3NF derived from the ClientRental relation.
  - Client (clientNo, cName)
  - Rental (clientNo, propertyNo, rentstart, rentFinish)
  - PropertyForRent (propertyNo, pAddress, rent, ownerNo)
  - Owner (ownerNo, oName)

Summary

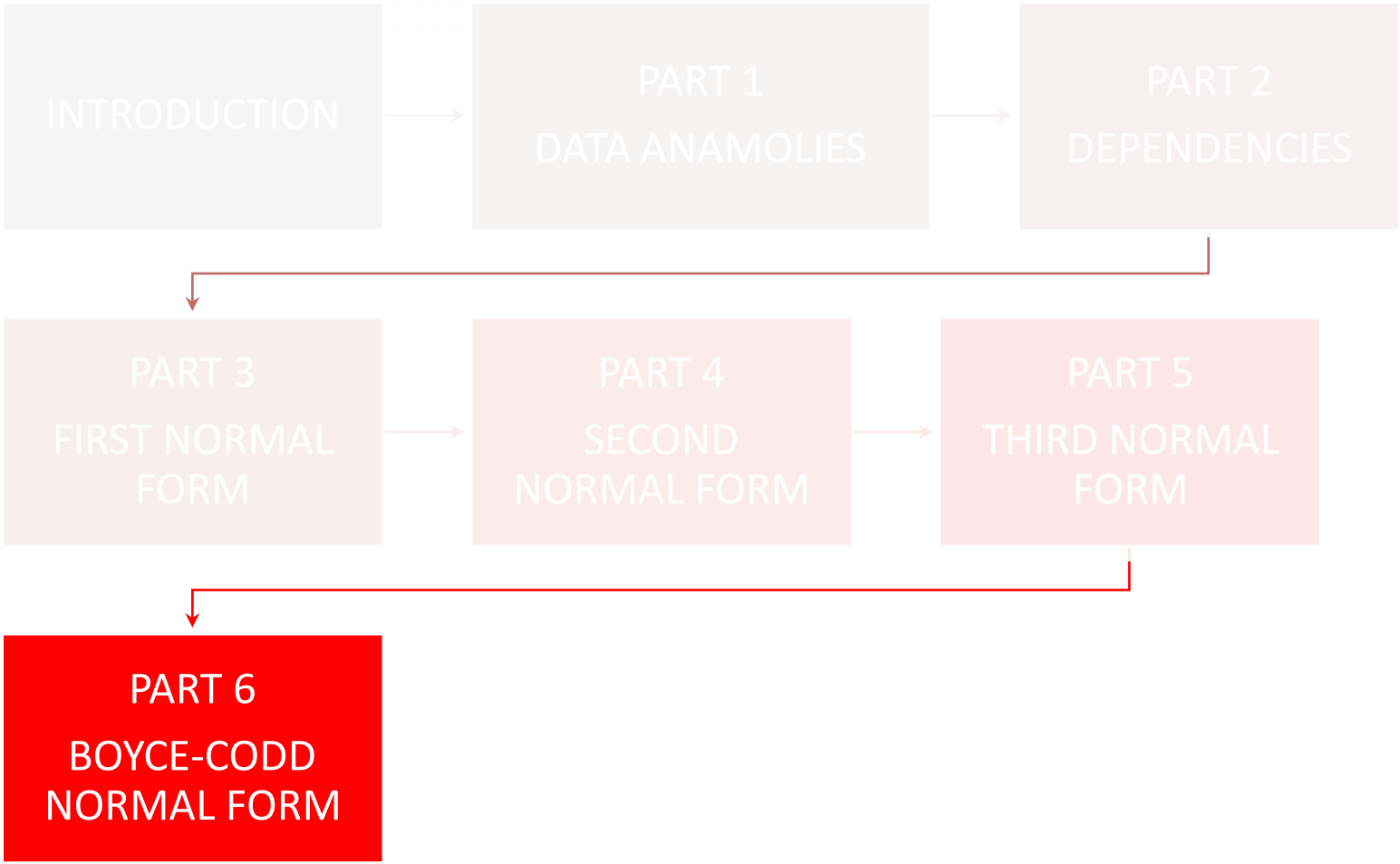
1NF: removing repeating group

2NF: removing partial dependency

3NF: removing transitive dependency

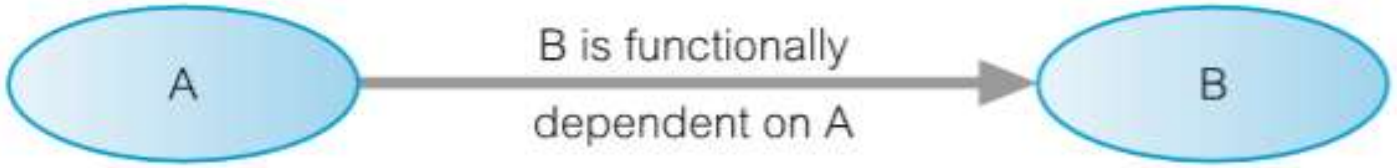
BCNF: removing non-candidate key determinant

PART 6: BOYCE-CODD NORMAL FORM



Boyce-Codd Normal Form (BCNF)

- Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with the general definition of 3NF.
- Boyce–Codd normal form (BCNF)
  - A relation is in BCNF if and only if every determinant is a candidate key.



determinant

NON-CANDIDATE KEY

Boyce-Codd Normal Form (BCNF)

- Difference between 3NF and BCNF is that for a functional dependency A → B:
  - 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key.
  - Whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
- Every relation in BCNF is also in 3NF. However, a relation in 3NF is not necessarily in BCNF.

BCNF: Example

- In this example, we extend the DreamHome case study to include a description of client interviews by members of staff.

ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13-May-14	10.30	SG5	G101
CR56	13-May-14	12.00	SG5	G101
CR74	13-May-14	12.00	SG37	G102
CR56	1-Jul-14	10.30	SG5	G102

Figure 15.1 ClientInterview relation.

The ClientInterview relation has the following functional dependencies:

fd1	clientNo, interviewDate → interviewTime, staffNo, roomNo	(Primary key)
fd2	staffNo, interviewDate, interviewTime → clientNo	(Candidate key)
fd3	roomNo, interviewDate, interviewTime → staffNo, clientNo	(Candidate key)
fd4	staffNo, interviewDate → roomNo	

Non-candidate key determinant

This relation is in 1NF, 2NF, 3NF, **BUT NOT** BCNF, because fd4 determinant is not a CK

BCNF: Example

Interview

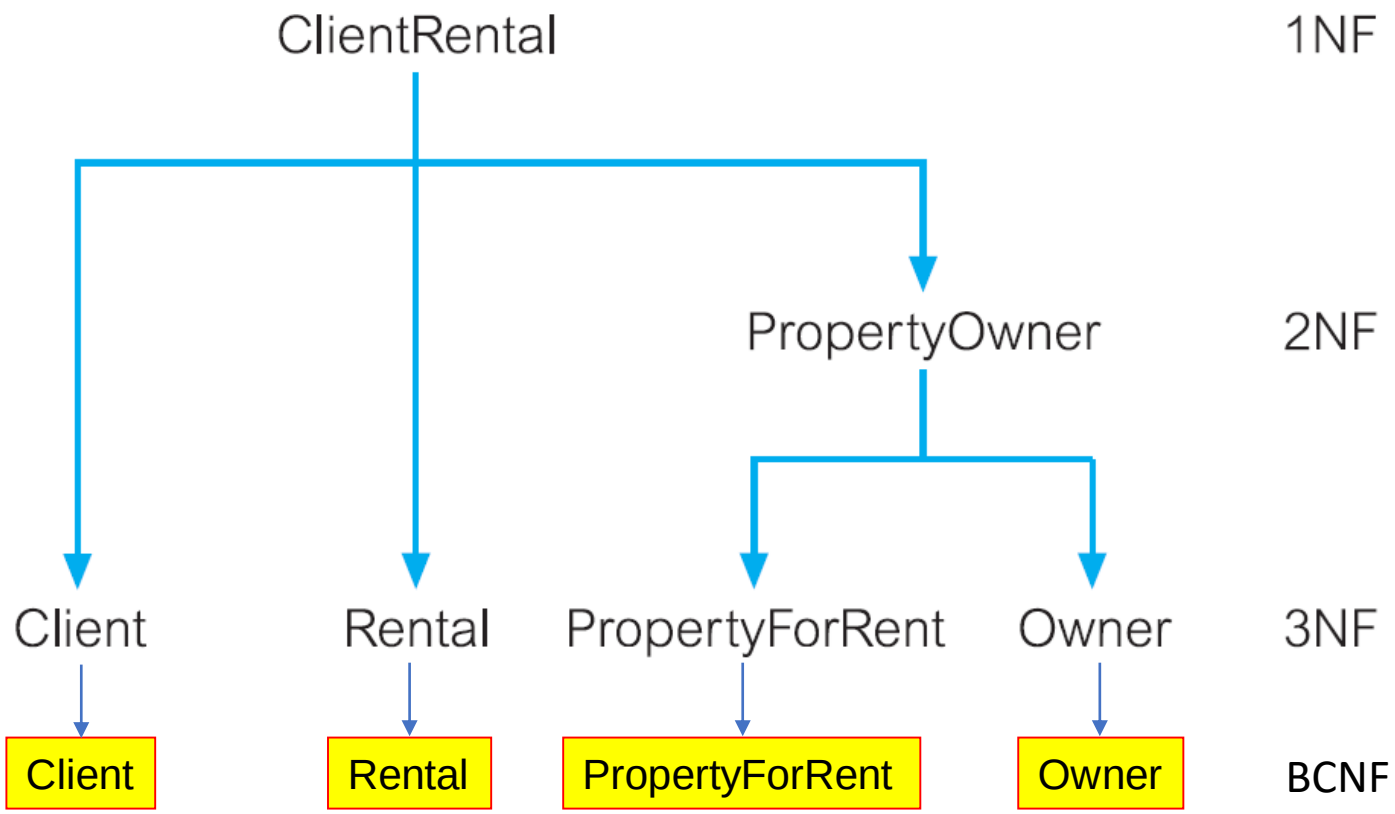
clientNo	interviewDate	interviewTime	staffNo
CR76	13-May-14	10.30	SG5
CR56	13-May-14	12.00	SG5
CR74	13-May-14	12.00	SG37
CR56	1-Jul-14	10.30	SG5

StaffRoom

staffNo	interviewDate	roomNo
SG5	13-May-14	G101
SG37	13-May-14	G102
SG5	1-Jul-14	G102

Figure 15.2 The Interview and StaffRoom BCNF relations.

Example of dreamhome



Another example

- Given the following table, normalize up to BCNF.

StaffPropertyInspection

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG4	6 Lawrence St, Glasgow	18-Oct-12 22-Apr-13 1-Oct-13	10.00 09.00 12.00	Need to replace crockery In good order Damp rot in bathroom	SG37 SG14 SG14	Ann Beech David Ford David Ford	M231 JGR M533 HDR N721 HFR
PG16	5 Novar Dr, Glasgow	22-Apr-13 24-Oct-13	13.00 14.00	Replace living room carpet Good condition	SG14 SG37	David Ford Ann Beech	M533 HDR N721 HFR

Another example

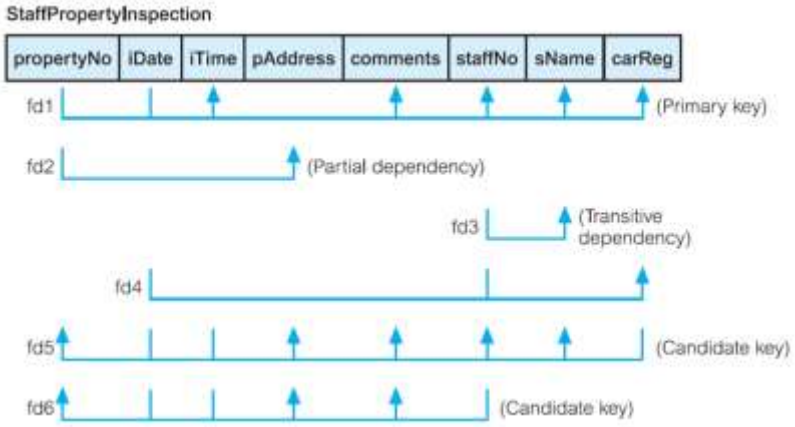
- UNF to 1NF:
  - StaffPropertyInspection (**propertyNo**, **iDate**, iTime, pAddress, comments, staffNo, sName, carReg)

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg
PG4	18-Oct-12	10.00	6 Lawrence St, Glasgow	Need to replace crockery	SG37	Ann Beech	M231 JGR
PG4	22-Apr-13	09.00	6 Lawrence St, Glasgow	In good order	SG14	David Ford	M533 HDR
PG4	1-Oct-13	12.00	6 Lawrence St, Glasgow	Damp rot in bathroom	SG14	David Ford	N721 HFR
PG16	22-Apr-13	13.00	5 Novar Dr, Glasgow	Replace living room carpet	SG14	David Ford	M533 HDR
PG16	24-Oct-13	14.00	5 Novar Dr, Glasgow	Good condition	SG37	Ann Beech	N721 HFR

Another example

- Dependency



fd1	propertyNo, iDate → iTime, comments, staffNo, sName, carReg	(Primary key)
fd2	propertyNo → pAddress	(Partial dependency)
fd3	staffNo → sName	(Transitive dependency)
fd4	staffNo, iDate → carReg	
fd5	carReg, iDate, iTime → propertyNo, pAddress, comments, staffNo, sName	(Candidate key)
fd6	staffNo, iDate, iTime → propertyNo, pAddress, comments	(Candidate key)

Another example

1NF:

StaffPropertyInspection (**propertyNo**, **iDate**, iTime, pAddress, comments, staffNo, sName, carReg)

fd1	propertyNo, iDate → iTime, comments, staffNo, sName, carReg	(Primary key)
fd2	propertyNo → pAddress	(Partial dependency)
fd3	staffNo → sName	(Transitive dependency)
fd4	staffNo, iDate → carReg	
fd5	carReg, iDate, iTime → propertyNo, pAddress, comments, staffNo, sName	(Candidate key)
fd6	staffNo, iDate, iTime → propertyNo, pAddress, comments	(Candidate key)

2NF:

PropertyInspection (**propertyNo**, **iDate**, iTime, comments, staffNo, sName, carReg)
Property (**propertyNo**, pAddress)

Property Relation

fd2	propertyNo → pAddress

PropertyInspection Relation

fd1	propertyNo, iDate → iTime, comments, staffNo, sName, carReg
fd3	staffNo → sName
fd4	staffNo, iDate → carReg
fd5′	carReg, iDate, iTime → propertyNo, comments, staffNo, sName
fd6′	staffNo, iDate, iTime → propertyNo, comments

Transitive dependency

Another example

3NF:

PropertyInspection (**propertyNo**, **iDate**, iTime, comments, staffNo, carReg)
Staff (**staffNo**, sName)
Property (**propertyNo**, pAddress)

Property Relation
fd2 propertyNo → pAddress

Staff Relation
fd3 staffNo → sName

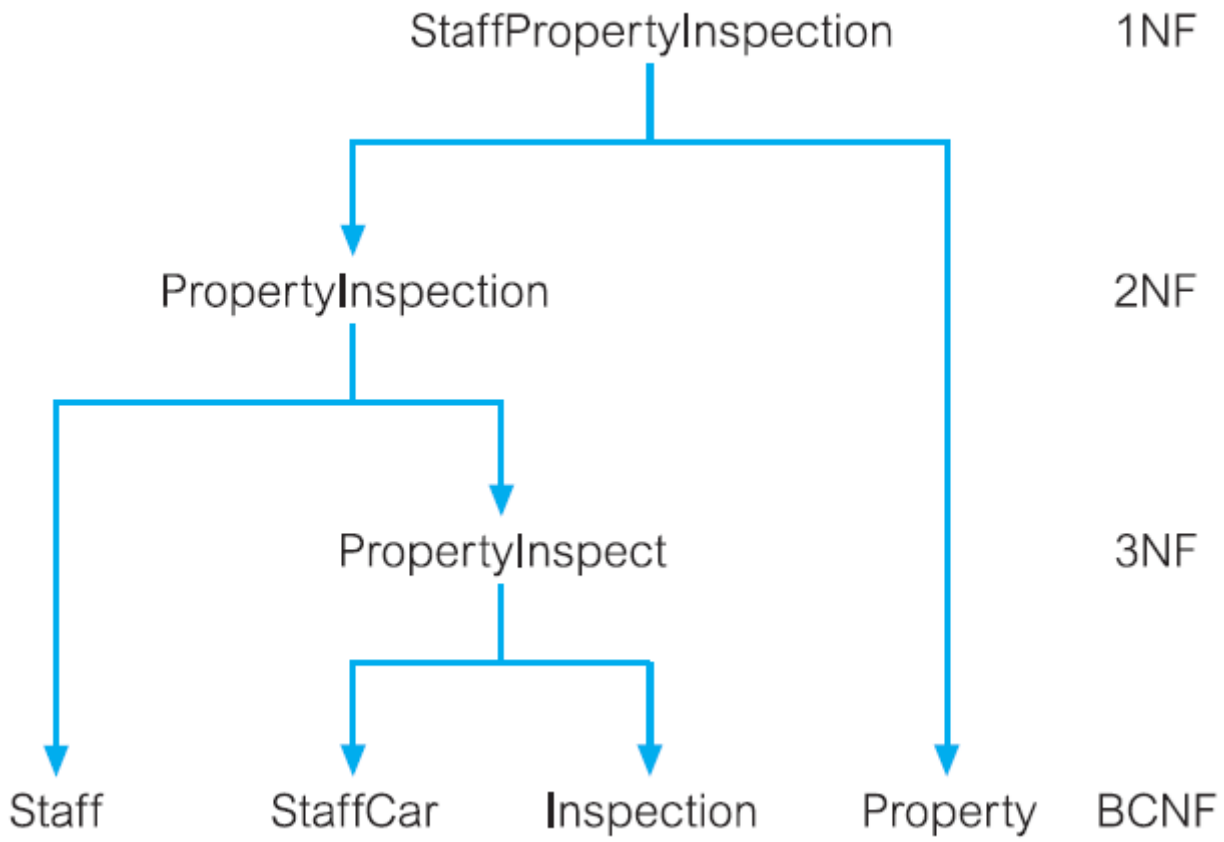
PropertyInspect Relation
fd1′ propertyNo, iDate → iTime, comments, staffNo, carReg
fd4 staffNo, iDate → carReg
fd5′ carReg, iDate, iTime → propertyNo, comments, staffNo
fd6′ staffNo, iDate, iTime → propertyNo, comments

Non candidate key

BCNF:

Inspection (**propertyNo**, **iDate**, iTime, comments, staffNo)
StaffCar (**staffNo**, **iDate**, carReg)
Staff (**staffNo**, sName)
Property (**propertyNo**, pAddress)

Another Example



Summary

1NF: removing repeating group

2NF: removing partial dependency

3NF: removing transitive dependency

BCNF: removing non-candidate key determinant

Innovating Solutions
Menginovasi Penyelesaian