Normalization

Dr Na Yao

Objectives

- Understand the purpose of normalization and how normalization can be used when designing a relational database.
- Be able to explain the potential problems associated with redundant data in base relations.
- Understand and be able to explain the concept of functional dependency, which describes the relationship between attributes.
- Understand the characteristics of functional dependencies used in normalization.

Objectives

- Be able to identify functional dependencies for a given relation.
- Understand how functional dependencies identify the primary key for a relation.
- Understand how normalization uses functional dependencies to group attributes into relations that are in a known normal form.
- How to identify the most commonly used normal forms, namely First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).

Purpose of Normalization

- Normalization is a technique for producing a set of suitable relations that support the data requirements of an enterprise.

What is a *suitable set of relations*

- the ***minimal*** number of attributes necessary to support the data requirements of the enterprise;
- attributes with a **close logical relationship** 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.

Benefits of using "suitable relations"

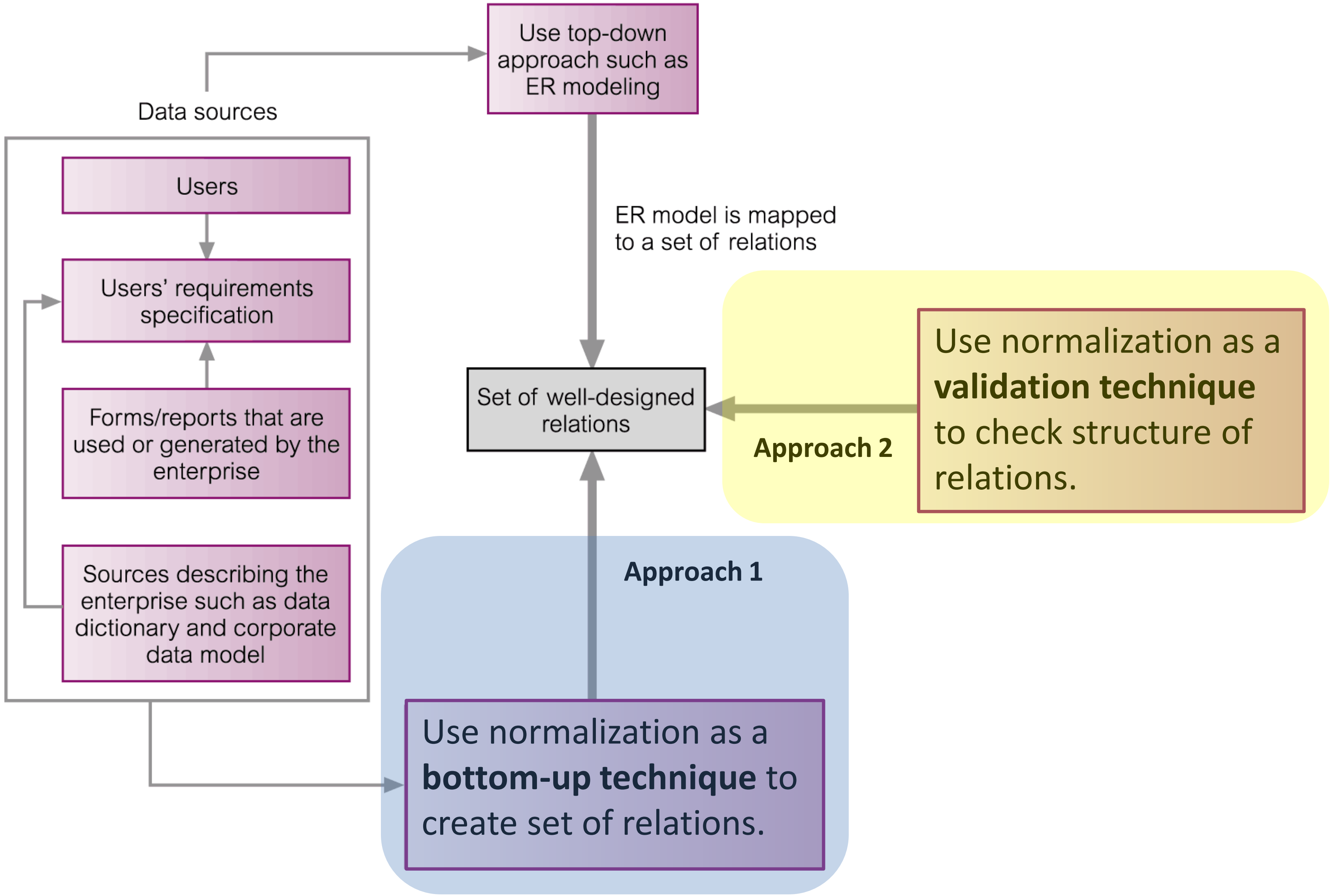
- Easier for the user to access and maintain the data;
- Take up minimal storage space on the computer.

Example: Lab 3 sample data sheet

Student number	First name	Last name	Date of Birth	Sex	Nationality	Special needs	Current status	Course

Contract number	Date	Duration	Student name	Student number	Place number	Which Hall/Flat	Room number	Date moving in	Date moving out

How Normalization supports database design



Data Redundancy and Update Anomalies

- One major aim of relational database design is to **group attributes into relations** to ***minimize data redundancy***.
- What are the problems if we have redundant data?

Data Redundancy and Update Anomalies

Design 1:

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

Design 2:

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

Data Redundancy and Update Anomalies

- StaffBranch relation has ***redundant*** data; the details of a branch are repeated for every member of staff.

- In contrast, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation, to represent where each member of staff is located.

Data Redundancy and Update Anomalies

- Relations that contain redundant information may have problems called **update anomalies**:
  - Insertion anomalies
    - what will happen if we want to insert details of new staff of B007?
    - What will happen if we want to insert details of a new branch with no staff?
  - Deletion anomalies
    - What will happen if we delete Mary Howe from StaffBranch?
  - Modification anomalies
    - What will happen if we want to change B003's address?

Decomposition

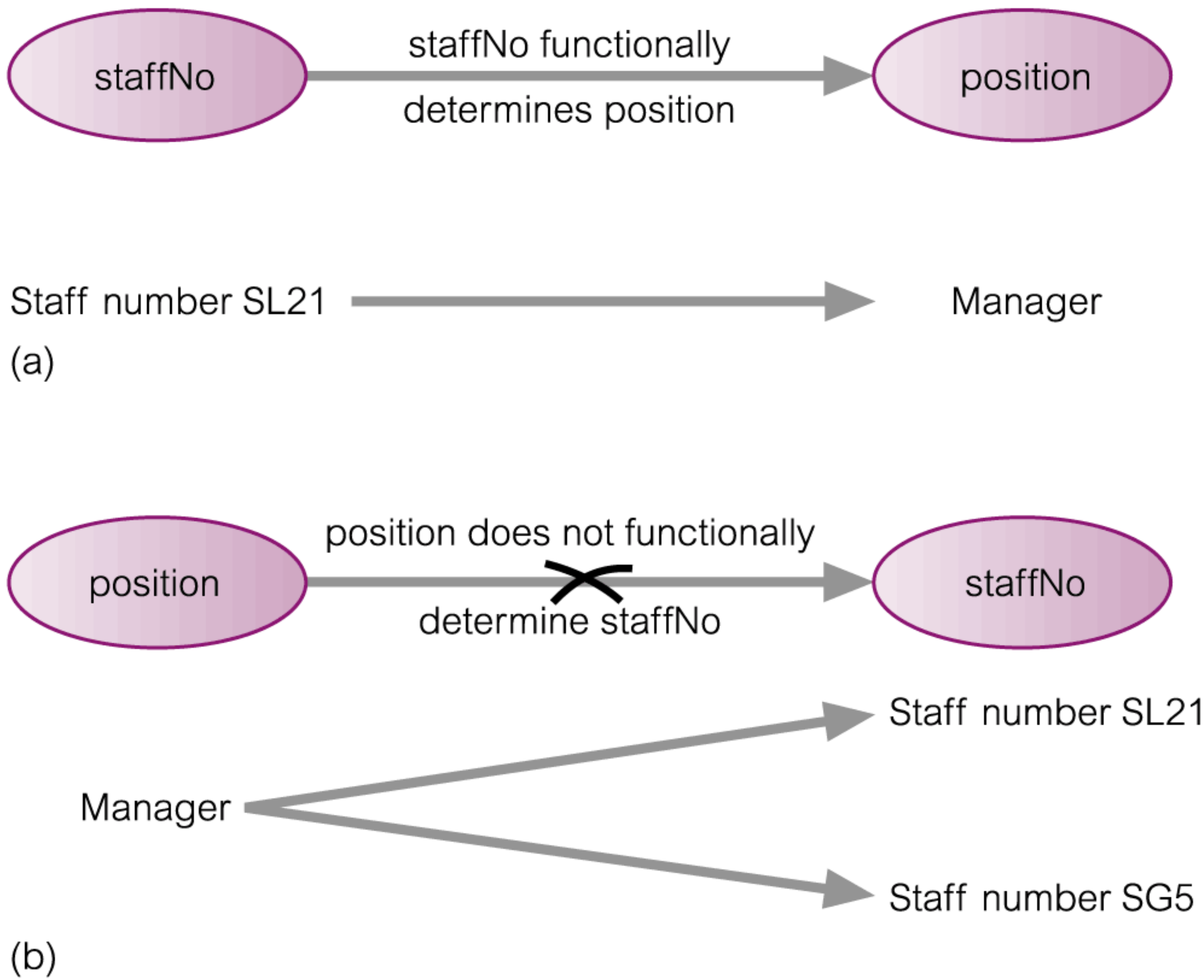
- What can we do to relations with redundant information that are subject to update anomalies?

- Two important properties of decomposition.
  - ***Lossless-join*** property: ensures that any instance of the original relation can be identified from corresponding instances in the smaller relations.
  - ***Dependency preservation*** property: ensures that a constraint on the original relation can be maintained by enforcing some constraint on each of the smaller relations.

Functional Dependencies

- Important concept associated with normalization.
- Functional dependency describes relationship between attributes.
- For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with exactly one value of B in R.

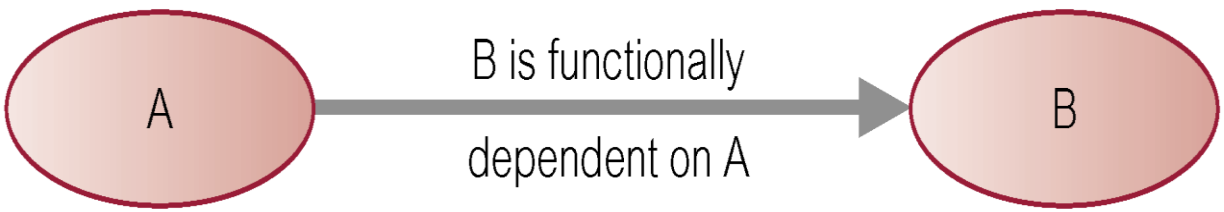
An Example Functional Dependency



Functional Dependencies

- Functional dependency is a property of the meaning or semantics of the attributes in a relation.

- Diagrammatic representation.



- The ***determinant*** of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.

Example Functional Dependency that holds for all Time

- Consider the values shown in staffNo and sName attributes of the Staff relation
- Based on sample data, the following functional dependencies appear to hold.

  staffNo → sName

  sName → staffNo

- However, the only functional dependency that remains true for all possible values for the staffNo and sName attributes of the Staff relation is:

  staffNo → sName

Full Functional Dependency

- Determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side.
- This requirement is called ***full functional dependency***.
- 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.

Partial functional dependency

- Is this a full functional dependency?

  staffNo, sName -> branchNo

- A functional dependency A -> B is a

  partial dependency

  if there is some attribute that can be removed from A and the dependency still holds.

Characteristics of Functional Dependencies

- Main characteristics of functional dependencies:
  - There is a *one-to-one* relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency.
  - Holds 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.

Transitive Dependencies

- ***Transitive dependency***

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).

Example Transitive Dependency

- Consider functional dependencies in the StaffBranch relation

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

- Transitive dependency:

  bAddress is transitively dependent on *staffNo* via *branchNo*

Identifying Functional Dependencies

How 

- By analysing the meaning of each attribute and the relationships between the attributes.
- Normally provided by the enterprise (client): discussions and/or documentation such as the users' requirements specification.
- Database designer may also need to use their common sense and/or experience if there is missing information.

Example - Identifying a set of functional dependencies

- Examine semantics of attributes in StaffBranch relation. Assume that position held and branch determine a member of staff's salary.

Example - Identifying a set of functional dependencies

- Examine semantics of attributes in StaffBranch relation. Assume that position held and branch determine a member of staff's salary.

staffNo -> sName, position, salary, branchNo, bAddress

branchNo -> bAddress

bAddress -> branchNo

branchNo, position -> salary

bAddress, position -> salary

Example - Using sample data to identify functional dependencies

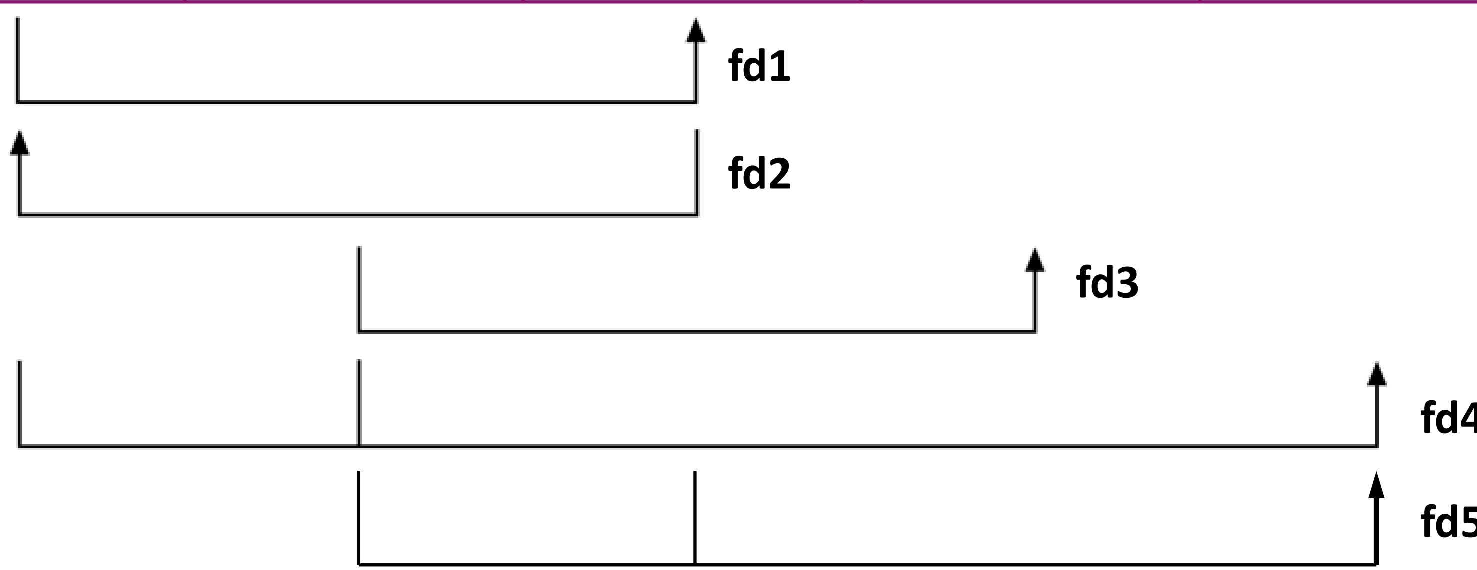
- Consider the data for attributes denoted A, B, C, D, and E in the Sample relation on next slide.

- Assume that sample data values shown in relation are <u>*representative of all possible values*</u> that can be held by attributes A, B, C, D, and E. Assume true despite the relatively small amount of data shown in this relation.

Example - Using sample data to identify 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



Example - Using sample data to identify functional dependencies

- Function dependencies between attributes A to E in the Sample relation.

$A \rightarrow C$	(fd1)
$C \rightarrow A$	(fd2)
$B \rightarrow D$	(fd3)
$A, B \rightarrow E$	(fd4)
$B, C \rightarrow E$	(fd5)

Identifying the Primary Key for a Relation using Functional Dependencies

- Main purpose of identifying a set of functional dependencies for a relation is the *identification of **candidate keys***, one of which is selected to be the primary key for the relation.
- To identify all candidate key(s), identify the *attribute (or group of attributes) that uniquely identifies each tuple* in this relation.
- If a relation has more than one candidate key, a candidate key is identified to act as the primary key.
- All attributes that are not part of a candidate key should be functionally dependent on the key.

Example - Identify Primary Key for StaffBranch Relation

- StaffBranch relation has five functional dependencies

  staffNo -> sName, position, salary, branchNo, bAddress

  branchNo -> bAddress

  bAddress -> branchNo

  branchNo, position -> salary

  bAddress, position -> salary

- The determinants are staffNo, branchNo, bAddress, (branchNo, position), and (bAddress, position).
- The only candidate key is staffNo.
- Therefore primary key is staffNo.

Example - Identifying Primary Key for Sample Relation

- Sample relation has five functional dependencies.

  $A \rightarrow C$ (fd1)

  $C \rightarrow A$ (fd2)

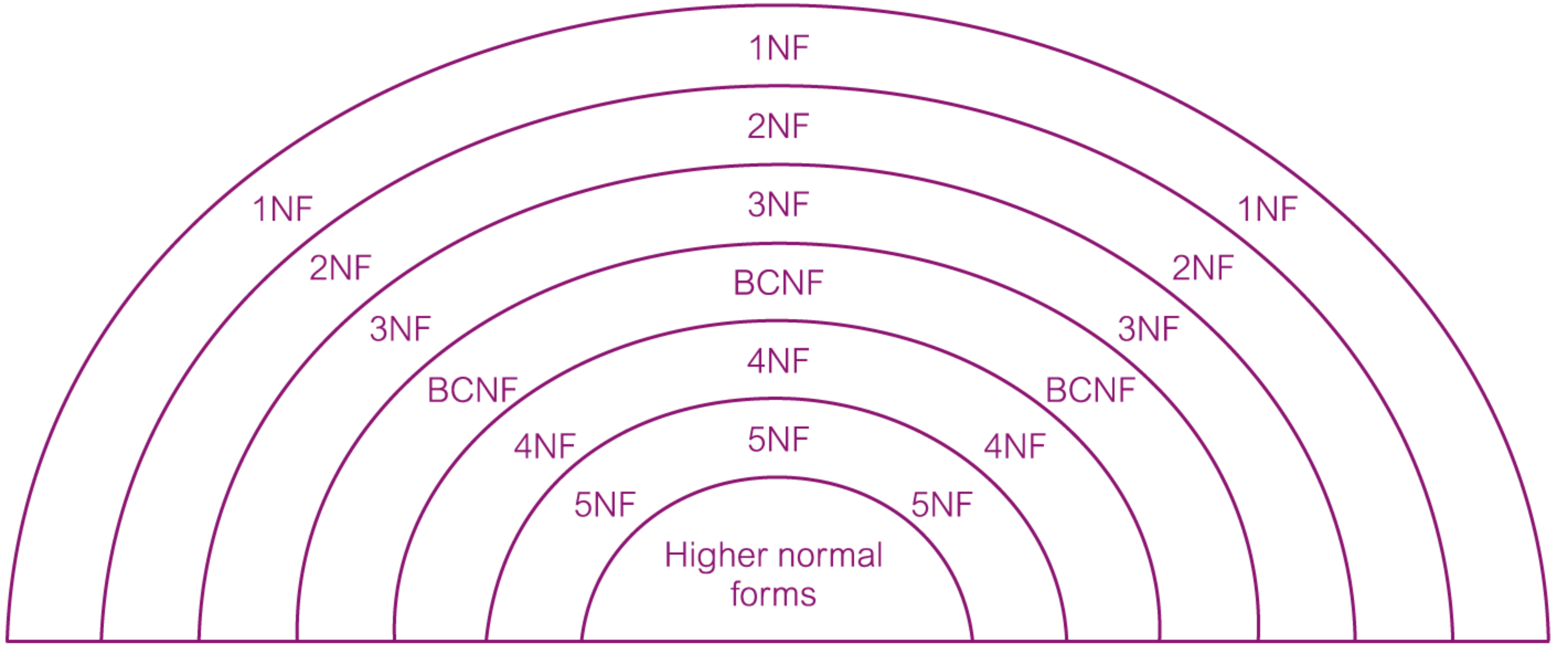
  $B \rightarrow D$ (fd3)

  $A, B \rightarrow E$ (fd4)

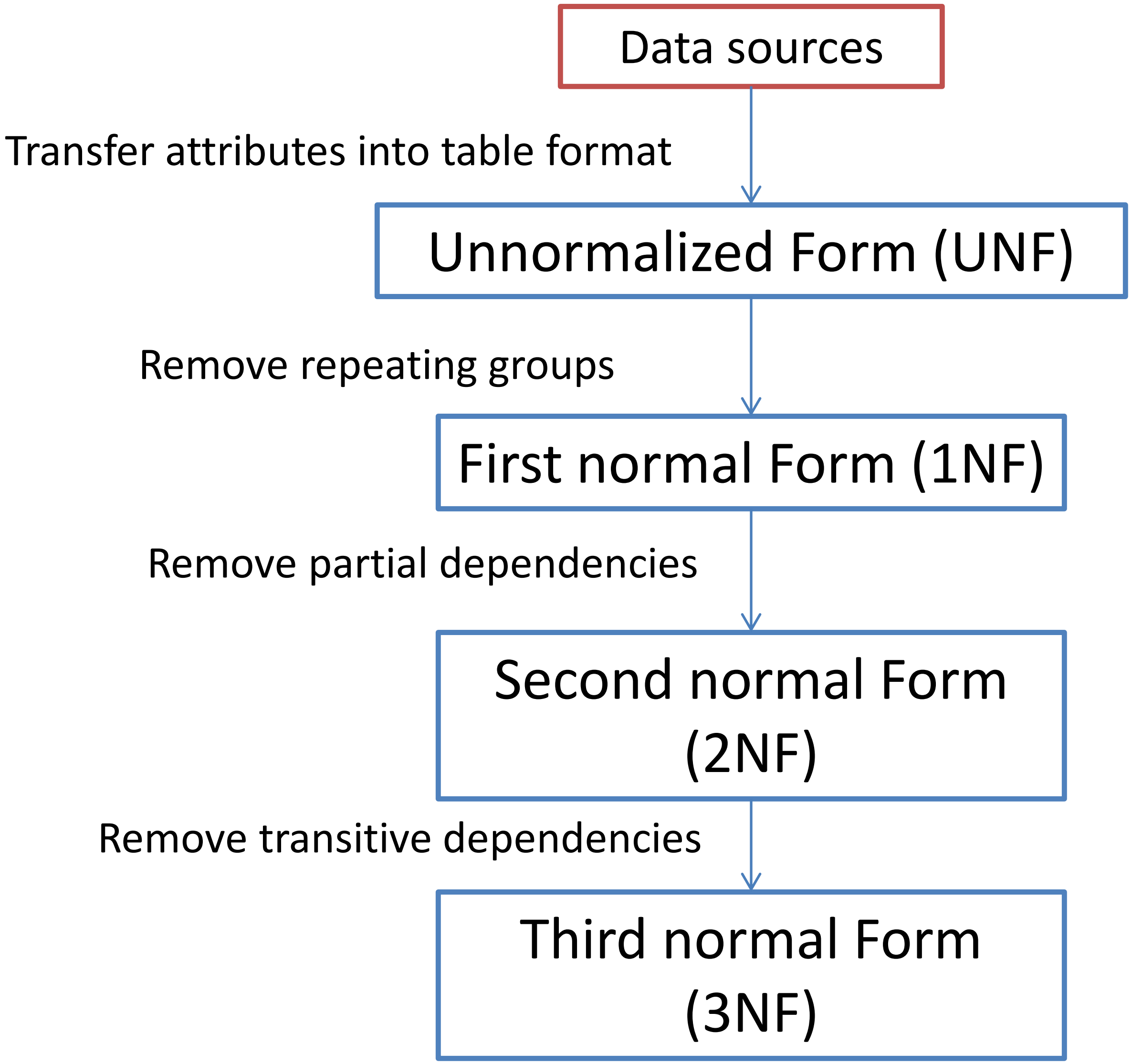
  $B, C \rightarrow E$ (fd5)

- The determinants in the Sample relation are A, B, C, (A, B), and (B, C).
- The only determinants that functionally determines all the other attributes of the relation are (A, B) and (B, C).
- Two candidate keys: (A, B) and (B, C)

The Process of Normalization



The Process of Normalization



Unnormalized Form (UNF)

- A table that contains one or more repeating groups.

- To create an unnormalized table
  - Transform the data from the information source (e.g. form) into table format with columns and rows.

UNF example

ClientRental

Client No	cName	property No	pAddress	rentStart	rentFinish	rent	owner No	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-07	31-Aug-08	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-08	1-Sep-09	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-06	10-Jun-07	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-07	1-Dec-08	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-09	10-Aug-10	450	CO93	Tony Shaw

First Normal Form (1NF)

- A relation in which the intersection of each row and column contains one and only one value.

UNF to 1NF

- Nominate an attribute or group of attributes to act as the key for the unnormalized table.
- Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).

UNF to 1NF

- Remove the repeating group by
  - Entering appropriate data into the empty columns of rows containing the repeating data (‘flattening’ the table).

  Or by

  - Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

1NF

ClientRental

Client No	cName	property No	pAddress	rentStart	rentFinish	rent	owner No	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-07	31-Aug-08	350	CO40	Tina Murphy
CR76	John Kay	PG16	5 Novar Dr, Glasgow	1-Sep-08	1-Sep-09	450	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-06	10-Jun-07	350	CO40	Tina Murphy
CR56	Aline Stewart	PG36	2 Manor Rd, Glasgow	10-Oct-07	1-Dec-08	375	CO93	Tony Shaw
CR56	Aline Stewart	PG16	5 Novar Dr, Glasgow	1-Nov-09	10-Aug-10	450	CO93	Tony Shaw

Second Normal Form (2NF)

- Based on the concept of full functional dependency.
- Full functional dependency indicates that if
  - A and B are attributes of a relation,
  - B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.

- 2NF: A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.

1NF to 2NF - Steps

1. Identify the functional dependencies in the relation.

2. Identify the primary key for the 1NF relation.

3. If partial dependencies exist, remove the partially dependent attributes from the relation by placing the attributes in a new relation along with a copy of their determinant.

1NF to 2NF

Client No	property No	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName

Step 1: identify functional dependencies

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

Step 2: identify primary key

2NF

Step 3: identify partial dependencies, then remove them

Client (<u>clientNo</u>, cName)

PropertyOwner(<u>propertyNo</u>, pAddress, rent, ownerNo, oName)

Rental(<u>clientNo</u>, <u>propertyNo</u>, rentStart, rentFinish)

Third Normal Form (3NF)

- Based on the concept of transitive dependency.
- Transitive Dependency is a condition where
  - A, B and C are attributes of a relation such that if A $\rightarrow$ B and B $\rightarrow$ C,
  - then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).
- 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 to 3NF - Steps

1. Identify functional dependencies in the relation.

2. Identify the primary key in the 2NF relation.

3. If transitive dependencies exist, remove the transitively dependent attributes from the relation by placing the attributes in a new relation along with a copy of their determinant.

2NF to 3NF

Step 1: identify functional dependencies

Step 2: identify primary key

Client (<u>clientNo</u>, cName)

Rental(<u>clientNo</u>, <u>propertyNo</u>, rentStart, rentFinish)

PropertyOwner(<u>propertyNo</u>, pAddress, rent, ownerNo, oName)

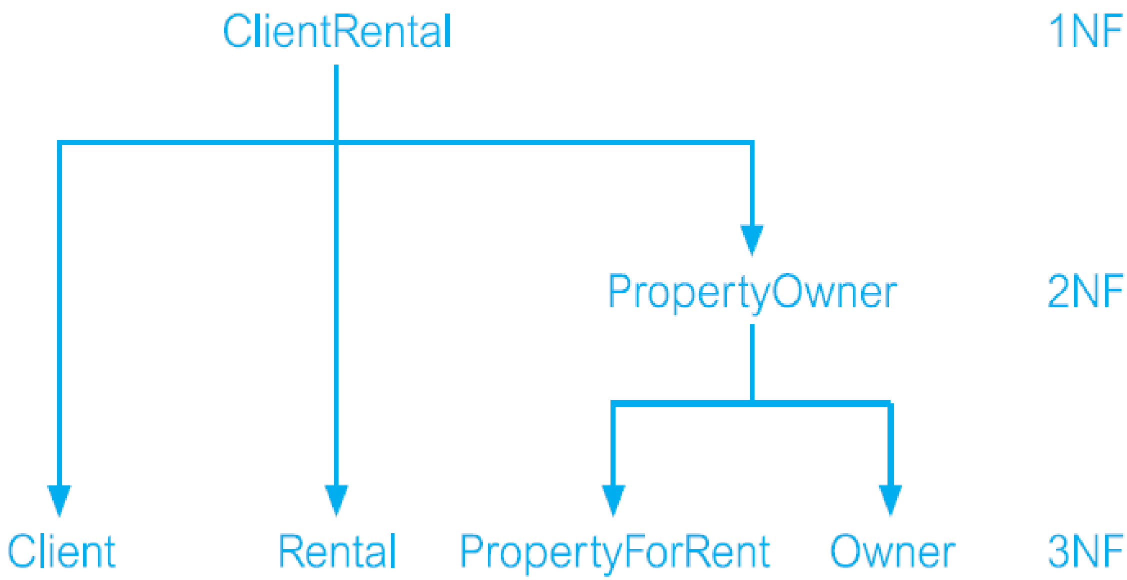
3NF

Step 3: identify transitive dependencies, then remove them

Client (clientNo, cName)

Rental(clientNo, propertyNo, rentStart, rentFinish)

Decomposition of ClientRental 1NF to 3NF



General Definitions of 2NF and 3NF

- Second normal form (2NF)
  - A relation that is in first normal form and every *non-candidate-key* attribute is fully functionally dependent on any *candidate key*.
- Third normal form (3NF)
  - A relation that is in first and second normal form and in which no *non-candidate-key* attribute is transitively dependent on any *candidate key*.

Normalization exercise

Examine the table shown below. This table represents the hours worked per week for temporary staff at each branch of a company. Assume that sample data values shown in relation are *representative of all possible values*.

staffNo	*branch No*	*branchAddress*	*name*	*position*	*hoursPer Week*
S4555	B002	City Center Plaza, Seattle, WA 98122	Ellen Layman	Assistant	16
S4555	B004	16 - 14th Avenue, Seattle, WA 98128	Ellen Layman	Receptionist	10
S4612	B002	City Center Plaza, Seattle, WA 98122	Dave Sinclair	Trainee	10
S4612	B004	16 - 14th Avenue, Seattle, WA 98128	Dave Sinclair	Assistant	10

a) Is this table in 2NF? **Explain** your answer.

b) Describe and illustrate the process of normalising the data shown in this table to third normal form (3NF).