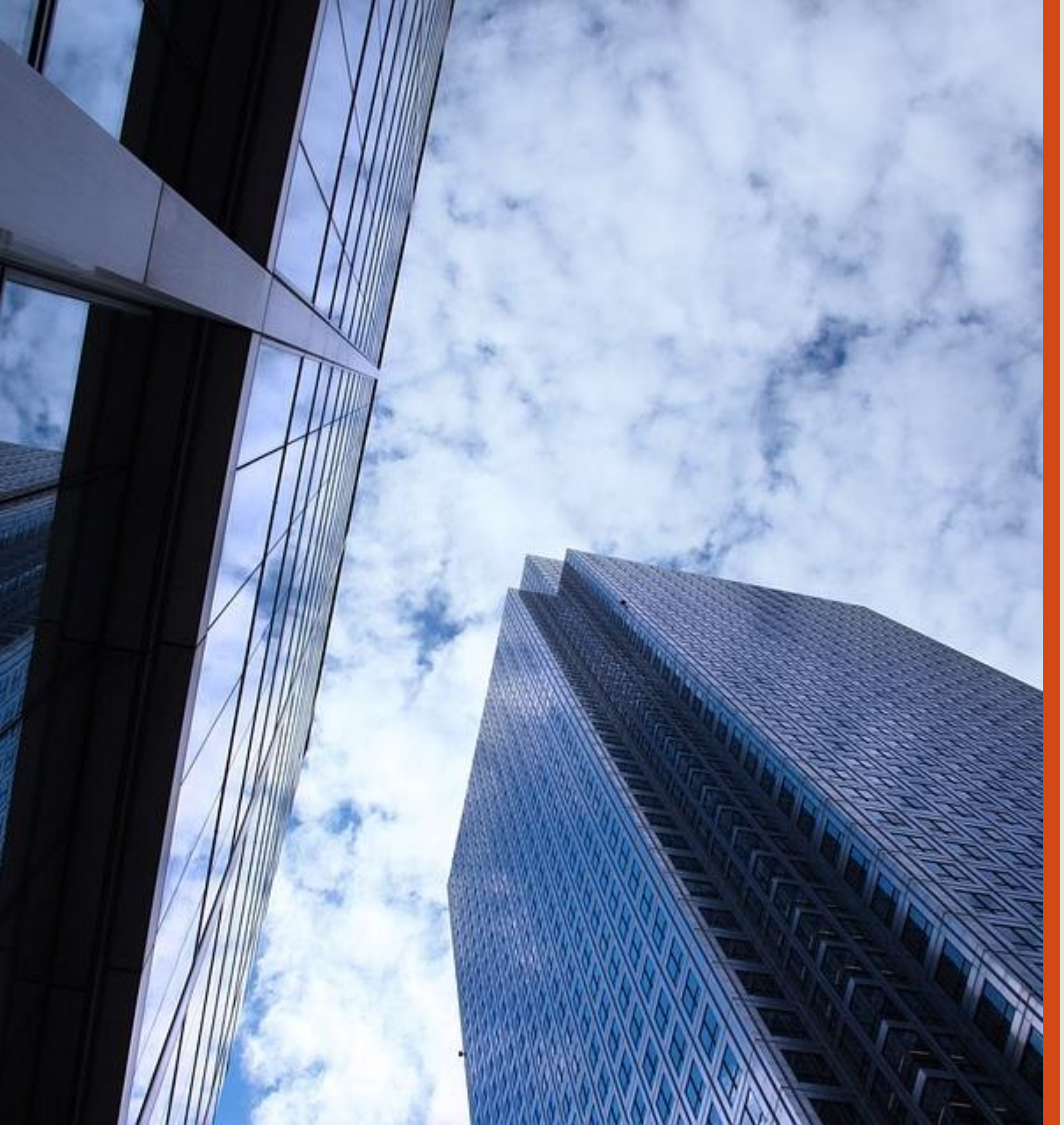


NORMALIZATION
PART 02

Jayathma Chathurangani

ejc@ucsc.cmb.ac.lk

OUTLINE

- Identification of Functional Dependencies
- What is Normalization?

1

IDENTIFICATION OF FUNCTIONAL DEPENDENCIES IN A RELATION

1.1 RECAP - FD

- **X** (Determinant) $\rightarrow$ **Y** (Dependent) means X determines Y.
- Both X and Y represents set of attributes
- If you know X, you can search the table for Y.
- ***Condition for FD:***

If t_1x (value of tuple t_1) and t_2x (value of tuple t_2x) are equal. i.e.

If $\mathbf{t_1x = t_2x}$ then $\mathbf{t_1y = t_2y}$

(Start by checking only if t_1x and t_2x are equal)

- Can determine using semantics or looking at data
- A **full functional dependency** is when all non-key attributes depend on the key attribute.
- A **partial dependency** is when you have a composite primary key (a primary key that is made up of multiple attributes), and on of the non-key attributes functionally dependent on one, but not all of the attributes that makeup the composite primary key.
- A **transitive dependency** is when you have a attribute that is functionally dependent on a attribute that is not the primary key

1.1 RECAP - FD

Determine by looking at data (No information given)

- The data values shown in this relation are representative of all possible values that can be held.
- In table 1: There is a unique Y value for given X.
- In table 2: X cannot determine Y. Because redundant data are not unique.

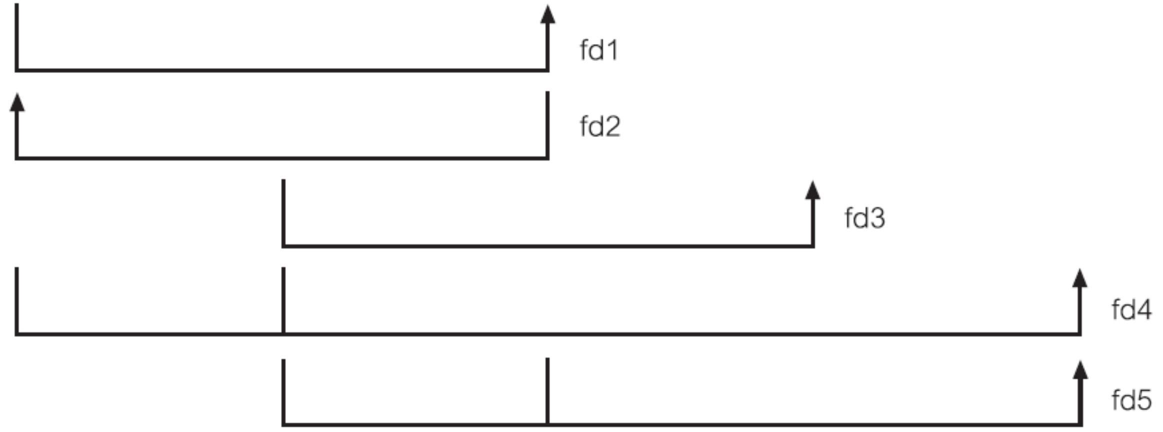
X	Y
1	1
2	1
3	2
4	3

Table 1

X	Y
1	1
2	1
3	2
4	3
2	5

Table 2

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



A ® C (fdl)
C ® A (fd2)
B ® D (fd3)
A, B ® E (fd4)
B, C ® E (fd5)

1.1 RECAP - FD

- We have been given that the **position held** and the **branch** determine a member of staff's **salary**.
- We can identify the functional dependencies based on our understanding of the attributes in the relation as:

staffNo ® sName, position, salary, branchNo, bAddress
branchNo ® bAddress
bAddress ® branchNo
branchNo, position ® salary
bAddress, position ® salary

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

ACTIVITY:
BY LOOKING ONLY AT THE TABLE DATA DETERMINED WHETHER THEY ARE FDS OR NOT?

R No (Roll Number)	Name	Marks	Dept (Department)	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

- R No → Name
- Name → R No
- R No → Marks
- Dept → Course
- Course → Dept
- R No, Name → Marks
- Name → Marks
- Name, Marks → Dept
- Name, Marks → Dept, course

ACTIVITY: FDS? ANSWERS

R No (Roll Number)	Name	Marks	Dept (Department)	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

- R No → Name : **Yes**
- Name → R No : **No**
- R No → Marks : **Yes**
- Dept → Course : **No**
- Course → Dept : **No**
- R No, Name → Marks : **Yes**
- Name → Marks : **Yes**
- Name, Marks → Dept : **Yes**
- Name, Marks → Dept, course : **No**

1.2 RECAP – FD RULES

Armstrong's axioms

- ***Reflexive rule:*** In **X** → **Y**, If **X** is a set of attributes and **Y** is_subset_of **X** **(X ⊇ Y)** , then **X** holds a value of **Y**. Further **X** → **X.**
- ***Transitivity rule:*** This rule is very much similar to the transitive rule in algebra if **X** → **Y** holds and **Y** → **Z** holds, then **X** → **Z** also holds.
- ***Augmentation rule:*** When **X** → **Y** holds, and **C** is attribute set, then **XC** → **YC** also holds. That is adding attributes which do not change the basic dependencies.

SECONDARY AXIOMS

- ***Union:*** If X→Y and X→Z, then X→YZ
- ***Decomposition:*** If X → YZ then X → Y and X → Z (Split RHS)
- ***Pseudo Transitivity rule:*** If X → Y and YZ → W, then XZ → W (Can write XZ → YZ, then can replace)
- **Composition:** If X → Y and A → B, then XA → YB

1.2 RECAP – FD RULES

EXAMPLE 01 (Assume last row is not there)

- ***Reflexive rule:***

R No → R No

R No, Name → Name

- ***Transitivity rule:***

If **Name → Marks** and

Marks → Dept (If last row is there this is not true)

Then **Name → Dept**

- ***Augmentation rule:***

When **R No → Name** holds, then

R No, Marks → Name, Marks also holds.

R No (Roll Number)	Name	Marks	Dept (Department)	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

1.2 RECAP – FD RULES

EXAMPLE 01 **(Assume last row is not there)**

- ***Union:***

If **R No → Name** and **R No → Marks**

R No → Name, Marks

- ***Decomposition:***

Name, Marks → Dept, Course

Then **Name, Marks → Dept** and **Name, Marks → Course**

- ***Pseudo Transitivity:***

If **R No → Name** and **Name, Marks → Dept**

Then **R No, Marks → Dept**

- ***Composition:***

If **R No → Name** and **Marks → Dept**

Then **R No, Marks → Name, Dept**

R No (Roll Number)	Name	Marks	Dept (Department)	Course
1	a	78	CS	C1
2	b	60	EE	C1
3	a	78	CS	C2
4	b	60	EE	C3
5	c	80	IT	C3
6	d	80	EC	C2

1.3 IDENTIFICATION OF CANDIDATE KEYS (CONTD.)

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

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

bAddress → branchNo

branchNo, position → salary

bAddress, position → salary

Determinants:
staffNo,
branchNo,
bAddress,
(branchNo, position),
(bAddress, position).

Ideal way is find the candidate key and then identify the primary key. But this has only one candidate key by surface browsing the FDs given.

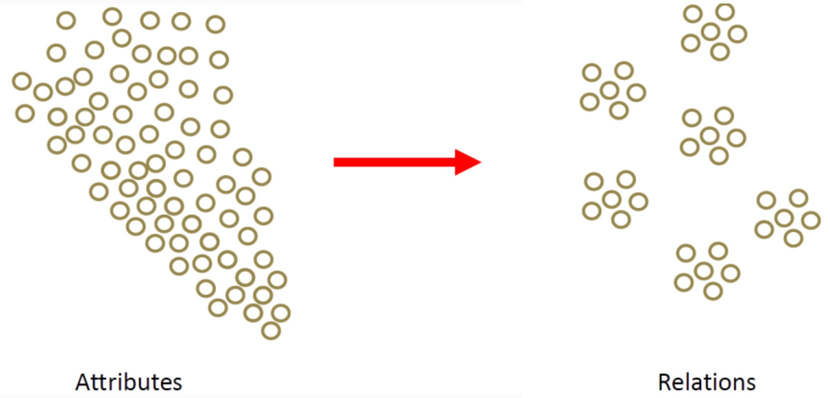
$\boxed{\text{staffNo}}$

3

NORMALIZATION?

3.1 WHAT IS IT?

- **Normalization is a database design technique that begins by examining the relationships (called functional dependencies) between attributes.**
- Attributes describe some property of the data or of the relationships between the data that is important to the enterprise.
- Normalization uses a series of tests (described as normal forms) to help identify the **optimal grouping** for these attributes to ultimately identify a set of suitable relations that supports the data requirements of the enterprise.
- **Normalization is A technique for producing a set of relations with desirable properties, given the data requirements of an enterprise.**



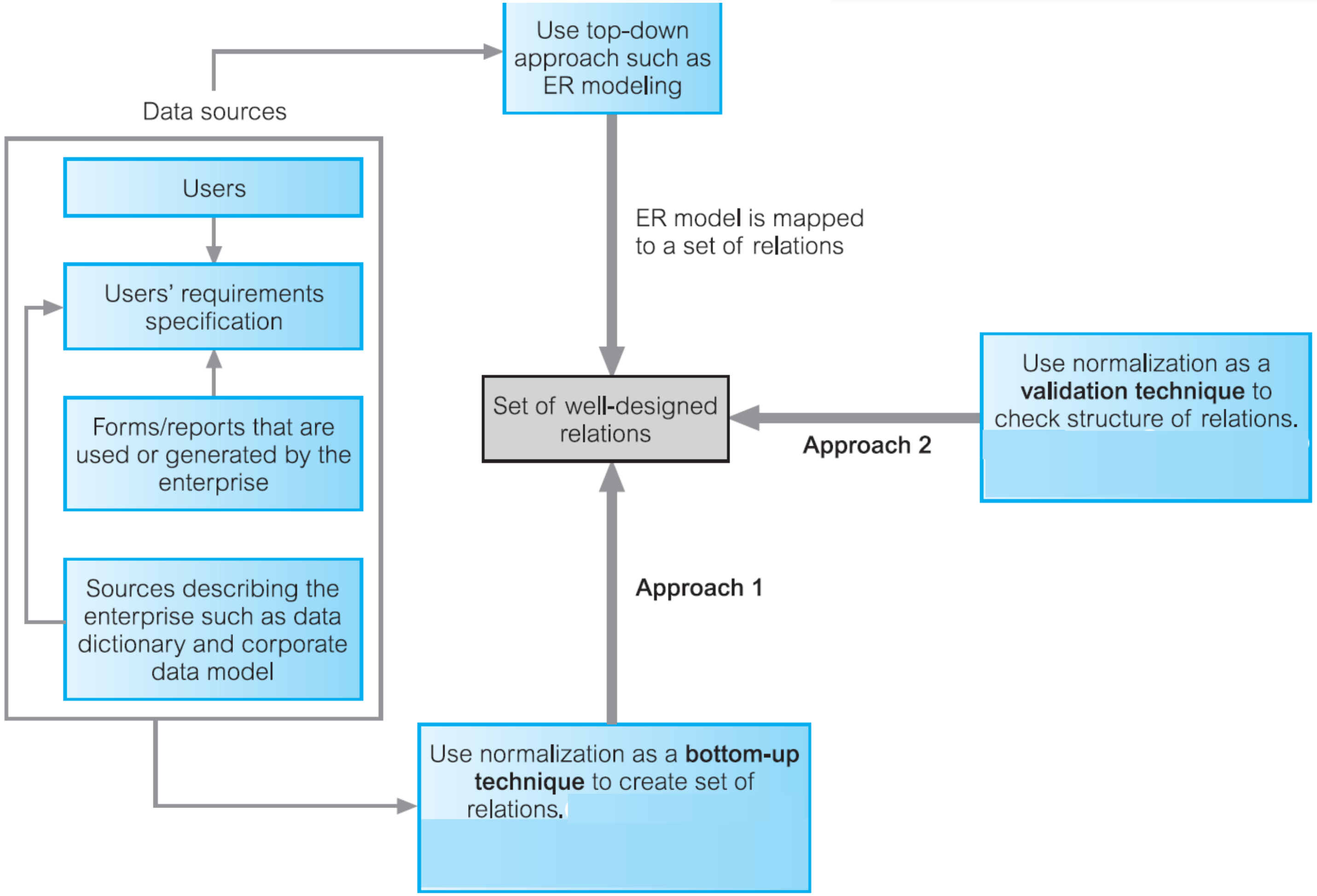
3.2 PURPOSE

- 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 (described 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.
- The benefits of using a database that has a suitable set of relations is that the database will be easier for the user to access and maintain the data, and take up minimal storage space in the computer.

3.3 HOW IT SUPPORTS DATABASE DESIGN?

- Normalization is a formal technique that can be used at any stage of database design. However, in this lesson we highlight two main approaches for using normalization,
  - ✓ Approach 1 : normalization can be used as a bottom-up standalone database design technique, and
  - ✓ Approach 2 : normalization can be used as a validation technique to check the structure of relations, which may have been created using a top-down approach such as ER modeling.
- No matter which approach is used, the goal is the same; creating a set of well-designed relations that meet the data requirements of the enterprise.

3.3 HOW IT SUPPORTS DATABASE DESIGN? (CONTINUED)



- Normalization as a bottom-up standalone technique (Approach 1) is often limited by the level of detail that the database designer is reasonably expected to manage.
- However, this limitation is not applicable when normalization is used as a validation technique (Approach 2), as the database designer focuses on only part of the database, such as a single relation, at any one time.
- Therefore, no matter what the size or complexity of the database, normalization can be usefully applied.

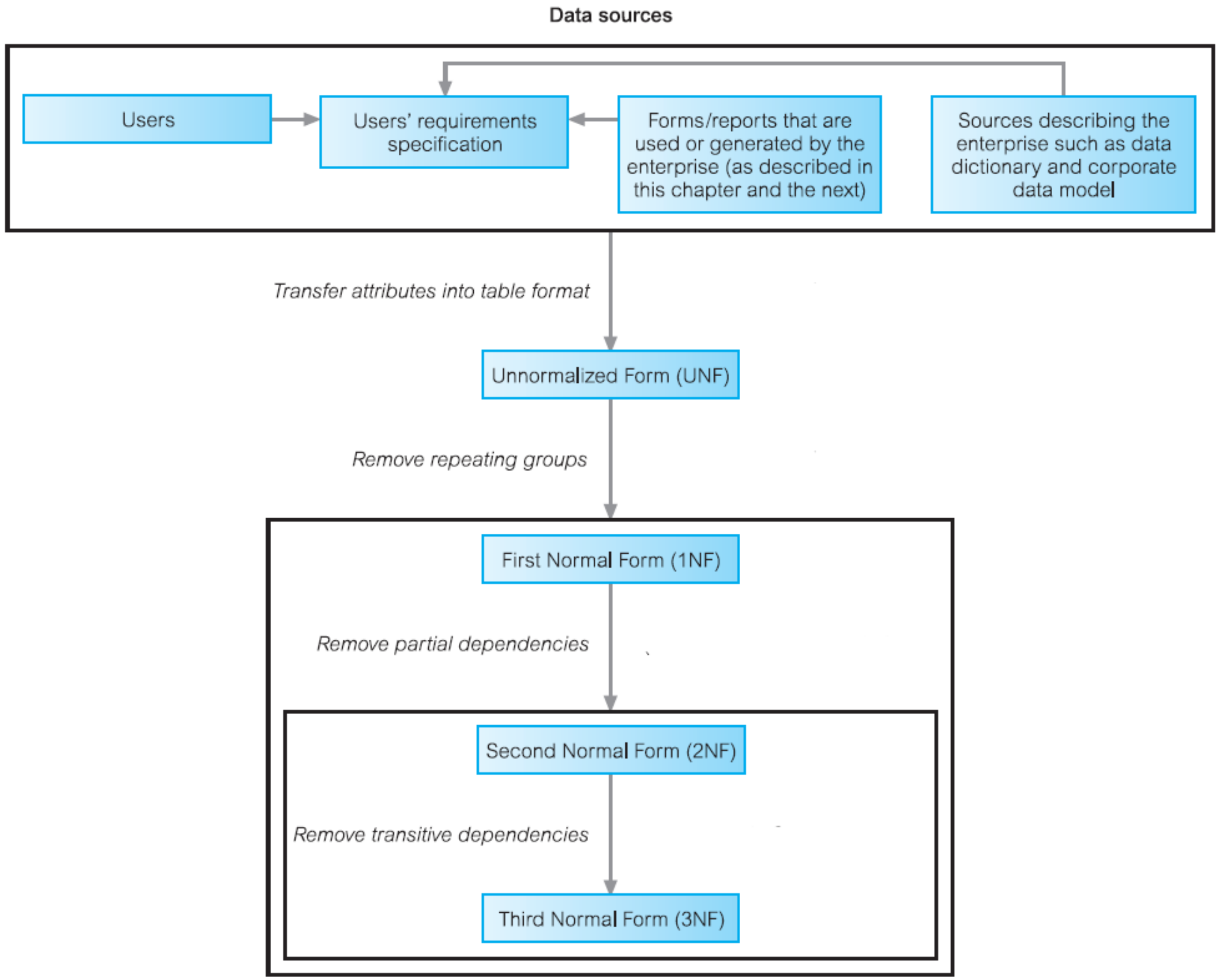
NORMALIZATION AS A BOTTOM-UP APPROACH

- Normalization is a **bottom-up approach** because it starts with a set of unstructured or partially structured data and progressively refines it into well-defined relational tables. The process begins with an existing set of attributes, dependencies, or a large unnormalized table, and then applies normalization rules step by step:
  1. **Starts with raw data** – Normalization begins with a flat, often redundant table or dataset.
  2. **Focuses on attributes and dependencies** – It looks at functional dependencies, anomalies, and redundancies.
  3. **Applies Normal Forms step by step** – It proceeds systematically through **1NF** → **2NF** → **3NF** → **BCNF**, each step refining the structure.
  4. **Results in well-structured tables** – After normalization, the data is split into multiple related tables with minimized redundancy.
- Since normalization refines an already existing structure by **breaking down** a large unstructured or redundant dataset into **smaller, optimized tables**, it follows a **bottom-up** approach.

ER MODELING AS A TOP-DOWN APPROACH

- Entity-Relationship (ER) modeling is a **top-down approach** because it begins with a conceptual overview of the data, considering the entire system and how entities relate to each other before defining tables.
  1. **Starts with high-level abstractions** – ER modeling begins by identifying **entities, relationships, and attributes** at a conceptual level.
  2. **Defines relationships first** – The primary focus is on how different entities interact rather than on specific table structures.
  3. **Progressively refines into schema** – The ER diagram is converted into a relational schema only after the conceptual design is completed.
  4. **Decomposes into relational tables** – Once relationships and entities are identified, the schema is converted into tables.
- Since ER modeling **starts from a broader system perspective** and **drills down** into specific tables and relationships, it is considered a **top-down** approach.

3.4 PROCESS OF NORMALIZATION (CONTINUED)



3.6 PROCESS OF NORMALIZATION

- Normalization is a formal technique for analyzing relations based on their primary key (or candidate keys) and functional dependencies (Codd, 1972b).
- The technique involves a series of rules that can be used to test individual relations so that a database can be normalized to any degree. When a requirement is not met, the relation violating the requirement must be decomposed into relations that individually meet the requirements of normalization.
- Three normal forms were initially proposed called First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).
- Subsequently, R. Boyce and E. F. Codd introduced a stronger definition of third normal form called Boyce–Codd Normal Form (BCNF) (Codd, 1974).

3.6 PROCESS OF NORMALIZATION (CONTINUED)

- In this chapter we describe only the first three normal forms.
- Normalization is often executed as a series of steps. Each step corresponds to a specific normal form that has known properties. As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.
- For the relational data model, it is important to recognize that it is only First Normal Form (1NF) that is critical in creating relations; all subsequent normal forms are optional.
- However, to avoid the update anomalies discussed, it is generally recommended that we proceed to at least Third Normal Form (3NF).

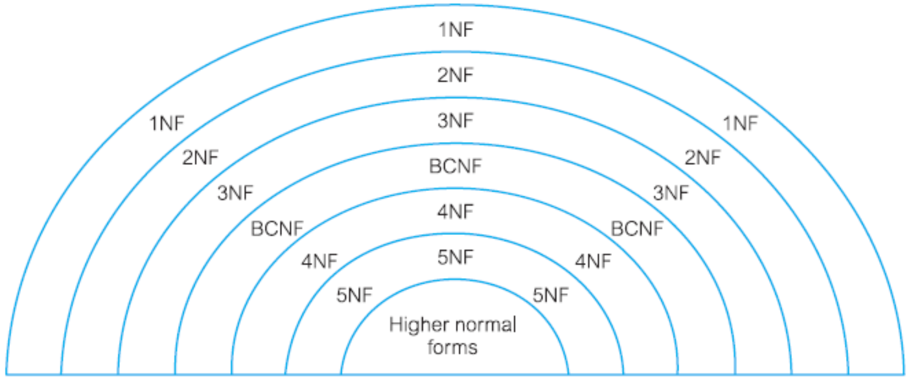
3.7 NORMALIZATION IN SUMMARY

- **Purpose:**

  to identify a suitable set of relations that support the data requirements of an enterprise

- **Characteristics(relations)**

  - the minimal number of attributes
  - attributes with a close logical relationship
  - Minimal redundancy with each attribute



- **Process: analyzing relations based on;**

  - their primary key
  - their functional dependencies

- Normalization is executed as a series of steps. Each step corresponds to a specific normal form that has known properties
- As depicted in the diagram higher normal forms should satisfy the lower normal form requirements
- Can increase quality by adding more normal forms. But Performance degrades.

3.7 NORMALIZATION IN SUMMARY

- **Advantages**

  - Reduction of data redundancy within tables
  - Reduce data storage space
  - Reduce inconsistency of data
  - Reduce update cost
  - Remove many to many relationship
  - Improve flexibility of the system

- **Disadvantages**

  - Increase join
  - Reduce efficiency
  - Increase use of indexes
  - Increase complexity of the system

WRAP UP AND THANK YOU.