Chapter 14

Normalization

Chapter 14 – Objectives

- The purpose of normalization.
- How normalization can be used when designing a relational database.
- The potential problems associated with redundant data(data that is repeated) in base relations.
- The concept of functional dependency, which describes the relationship between attributes.
- The characteristics of functional dependencies used in normalization.

Chapter 14 - Objectives

- How to identify functional dependencies for a given relation.
- How functional dependencies identify the primary key for a relation.
- How to undertake the process of normalization.
- How normalization uses functional dependencies to group attributes into relations that are in a known normal form.

Chapter 14 - Objectives

- How to identify the most commonly used normal forms, namely First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).
- The problems associated with relations that break the rules of 1NF, 2NF, or 3NF.
- How to represent attributes shown on a form as 3NF relations using normalization.

Purpose of Normalization

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

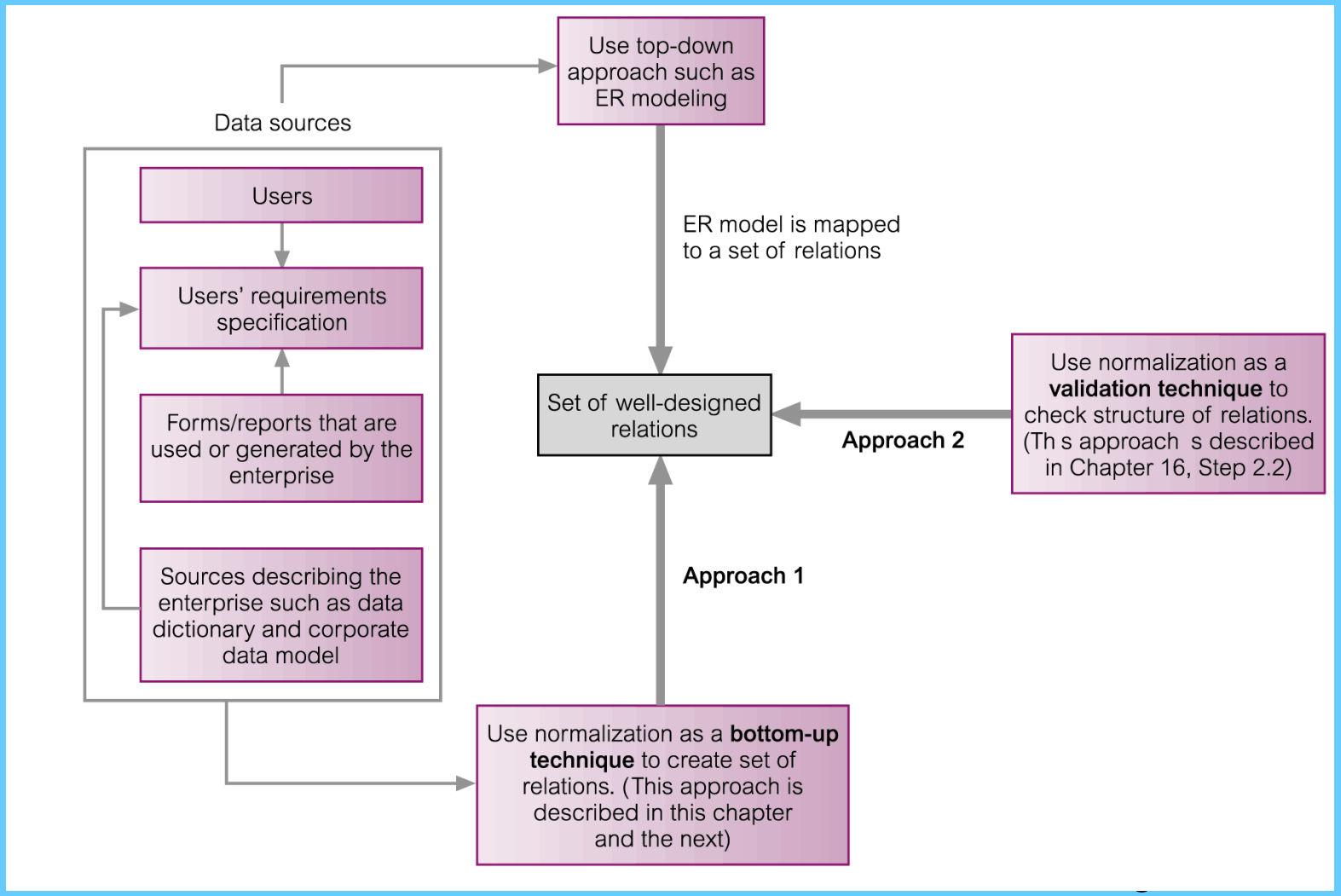
Purpose of Normalization

- **Characteristics of a suitable set of relations include:**
  - **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.**

Purpose of Normalization

- **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;**
  - **take up minimal storage space on the computer;**
  - **make it much easier to keep the data correct.**

How Normalization Supports Database Design



Data Redundancy (more than one copy) and Update Anomalies

- **Major aim of relational database design is to group attributes into relations to minimize data redundancy.**

Data Redundancy and Update Anomalies

- **Potential benefits for implemented database include:**
  - **Updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.**
  - **Reduction in the file storage space required by the base relations thus minimizing costs.**

Data Redundancy and Update Anomalies

- **Problems associated with data redundancy are illustrated by comparing the Staff and Branch relations with the StaffBranch relation.**

Data Redundancy and Update Anomalies

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

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

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 potentially suffer from update anomalies.**

- **Types of update anomalies include**
  - **Insertion (e.g. try to insert a new branch(may not have staff there yet))**
  - **Deletion (e.g. delete staffNo SA9(branch info is lost)**
  - **Modification (e.g. change B003 address (redundant))**

Lossless-join and Dependency Preservation Properties

- **Two important properties of decomposition.**
  - ***Lossless-join property* enables us to find any instance of the original relation(exactly) from corresponding instances in the smaller relations.**
  - ***Dependency preservation property* enables us to enforce a constraint on the original relation by enforcing some constraint on each of the smaller relations.**

To get a lossless decomposition, the common attribute (or common attributes) Must be a candidate key in one of the tables.

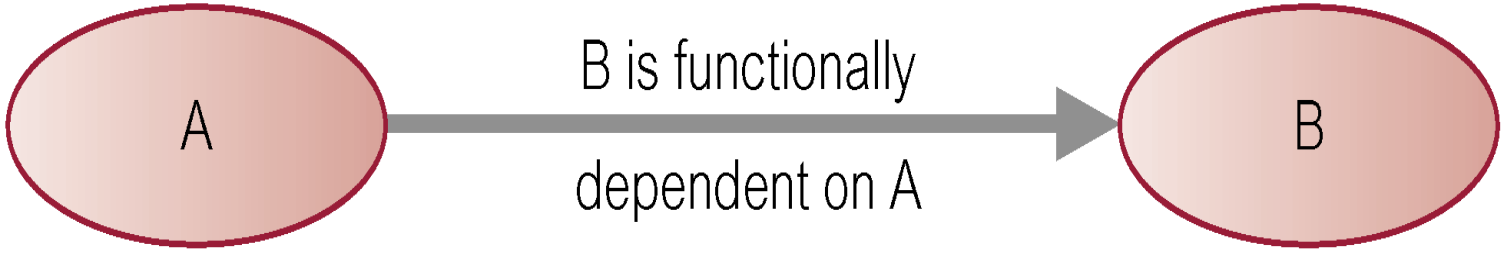
Functional Dependencies

- 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.

Characteristics of Functional Dependencies

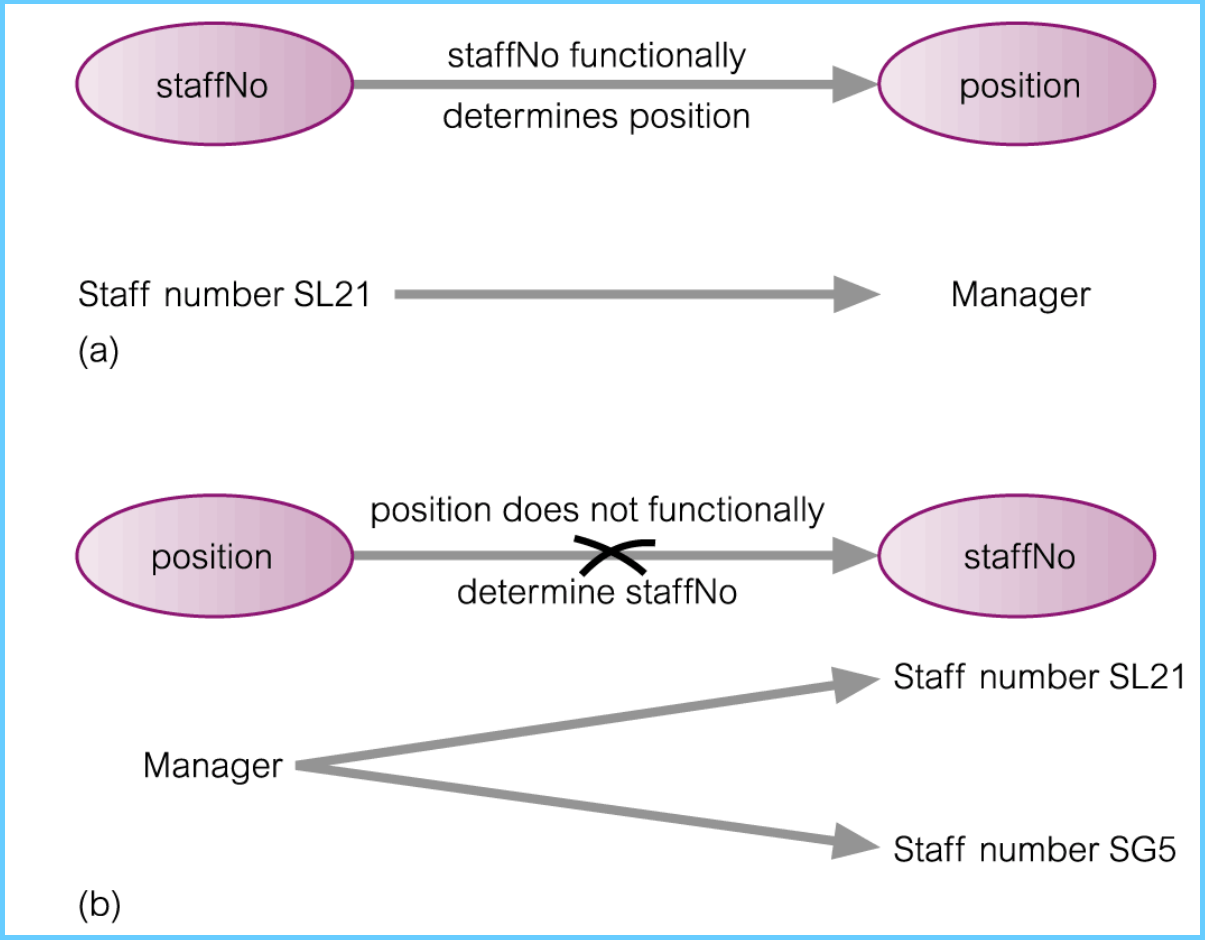
- **Property of the meaning or semantics of the attributes in a relation.**

- **Diagrammatic representation.**



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

An Example Functional Dependency



Example Functional Dependency that holds for all Time

- Consider the values shown in staffNo and sName attributes of the Staff relation (see Slide 12).

- Based on sample data, the following functional dependencies appear to hold.

staffNo → sName

sName → staffNo

Example Functional Dependency that holds for all Time

- **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

Characteristics of Functional Dependencies

- **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.***

Characteristics of Functional Dependencies

- **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.**

Example Full Functional and Partial Dependency

- **Consider the table Viewing (clientNo, propertyNo, clientName, date, comment)**

  clientNo, propertyNo -> everything (PK)
  clientNo -> clientName partial dependency

- **Example above is a *partial dependency.***

Characteristics of Functional Dependencies

- **Main characteristics of functional dependencies used in normalization:**
  - **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

- **Important to recognize a transitive dependency because its existence in a relation can potentially cause update** anomalies.

- **Transitive dependency describes 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).**

Example Transitive Dependency

- Consider functional dependencies in the StaffBranch relation (see Slide 12).

staffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

- Transitive dependency, branchNo → bAddress exists on staffNo via branchNo.

The Process of Normalization

- **Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.**

- **Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.**

Identifying Functional Dependencies

- **Identifying all functional dependencies between a set of attributes is relatively simple if the meaning of each attribute and the relationships between the attributes are well understood.**

- **This information should be provided by the enterprise in the form of discussions with users and/or documentation such as the users' requirements specification.**

Identifying Functional Dependencies

- **Learning the functional dependencies has a Great advantage. The developer MUST learn the enterprise, the business logic. This makes you much more valuable to your company.**

Identifying Functional Dependencies – Read the below!

- However, if the users are unavailable for consultation and/or the documentation is incomplete then depending on the database application, it may be necessary for the database designer to use their common sense and/or experience to provide the missing information.

Example - Identifying a set of functional dependencies for the StaffBranch relation

- **Examine semantics of attributes in StaffBranch relation (see Slide 12). Assume that position held and branch determine a member of staff's salary.**

Example - Identifying a set of functional dependencies for the StaffBranch relation

- **With sufficient information available, identify the functional dependencies for the StaffBranch relation as:**

  staffNo → sName, position, salary, branchNo, bAddress

  branchNo → bAddress

  bAddress → branchNo

  branchNo, position → salary

  bAddress, position → salary

Identifying the Primary Key for a Relation using Functional Dependencies

- **Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.**

- **An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.**

Example - Identify Primary Key for StaffBranch Relation

- StaffBranch relation has five functional dependencies (see Slide 32).

- The determinants are staffNo, branchNo, bAddress, (branchNo, position), and (bAddress, position).

- To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation.

Example - Identifying Primary Key for StaffBranch Relation

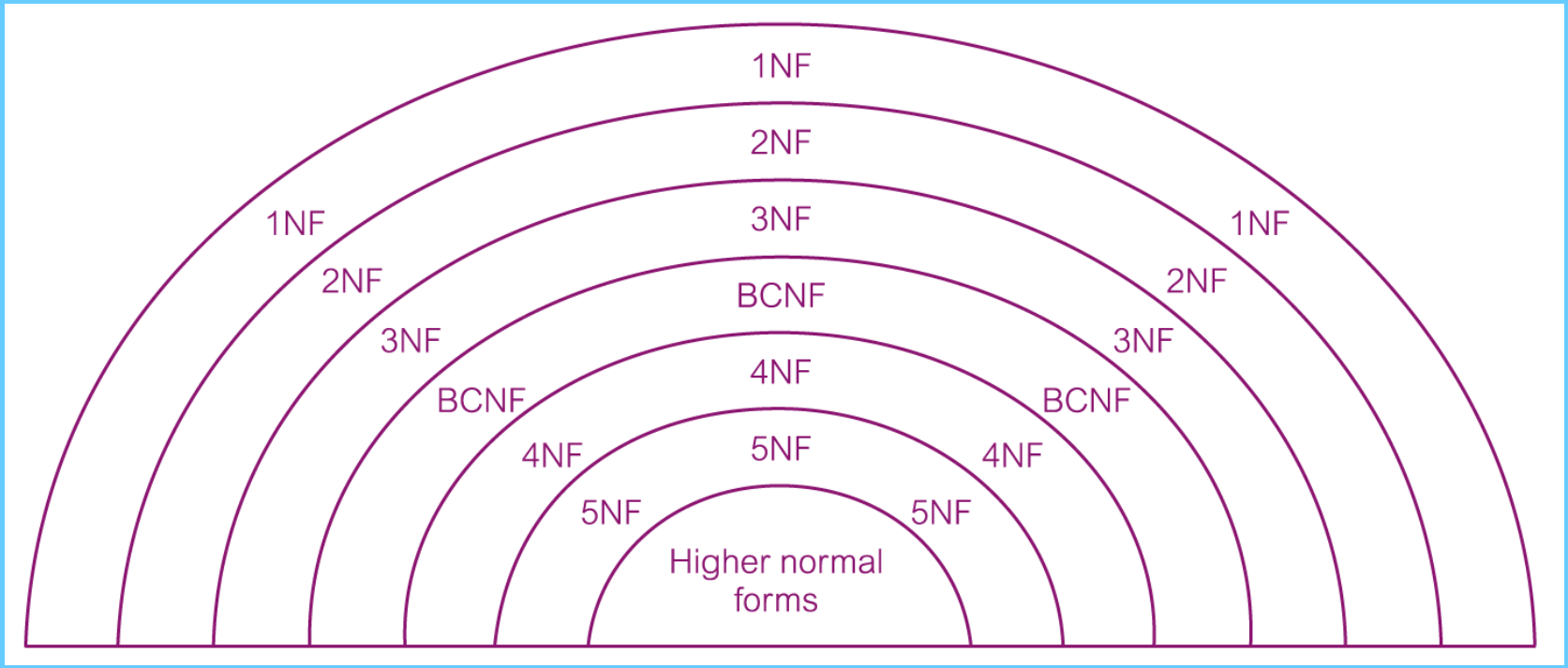
- **All attributes that are not part of a candidate key should be functionally dependent on the key.**

- **The only candidate key and therefore primary key for StaffBranch relation, is staffNo, as *all* other attributes of the relation are functionally dependent on staffNo.**

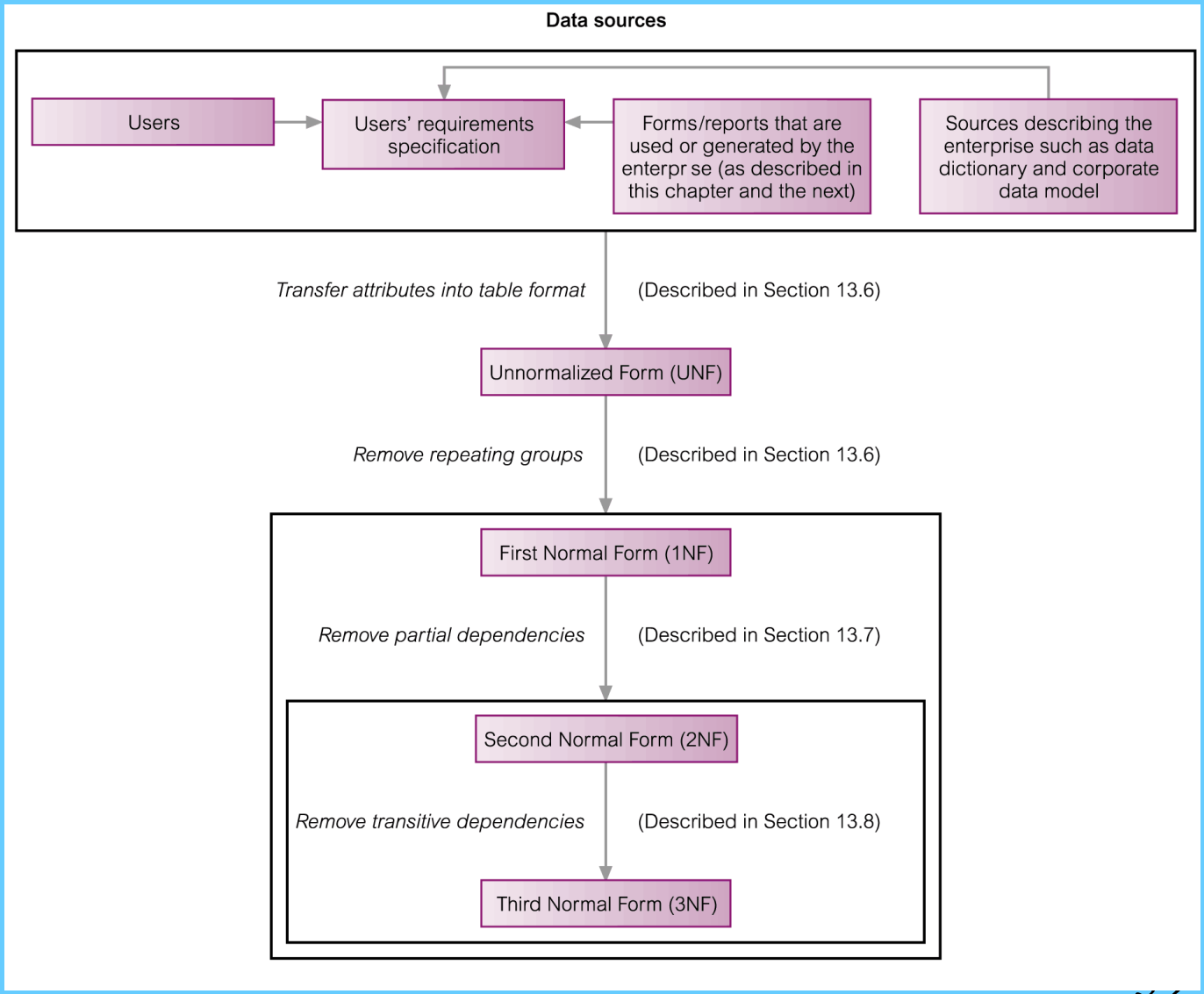
The Process of Normalization

- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.

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.

First Normal Form (1NF)

- A relation in which the intersection of each row and column contains one and only one value (no repeating groups, no multi-valued attributes).

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.

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.

Second Normal Form (2NF)

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

1NF to 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.
- If partial dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant.

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).

Third Normal Form (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

- Identify the primary key in the 2NF relation.
- Identify functional dependencies in the relation.
- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant.

General Definitions of 2NF and 3NF (Stronger, additional constraints)

- **Second normal form (2NF)**
  - **A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on *any candidate key*.**

    E.G., employeeID and SS# are both candidate keys

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