Chapter 14

Normalization

Materials To Have Handy

- A paper copy of the StaffBranch relation (pg. 407 of the handout and on or about slide 6)
- A paper copy of the Staff Branch function dependencies (pg. 413 of the handout and on or about slide 23)

What is Normalization?

- A technique to decompose relations into groupings of logically related attributes based on functional dependencies between attributes.
- A bottom-up design technique

Purpose of Normalization

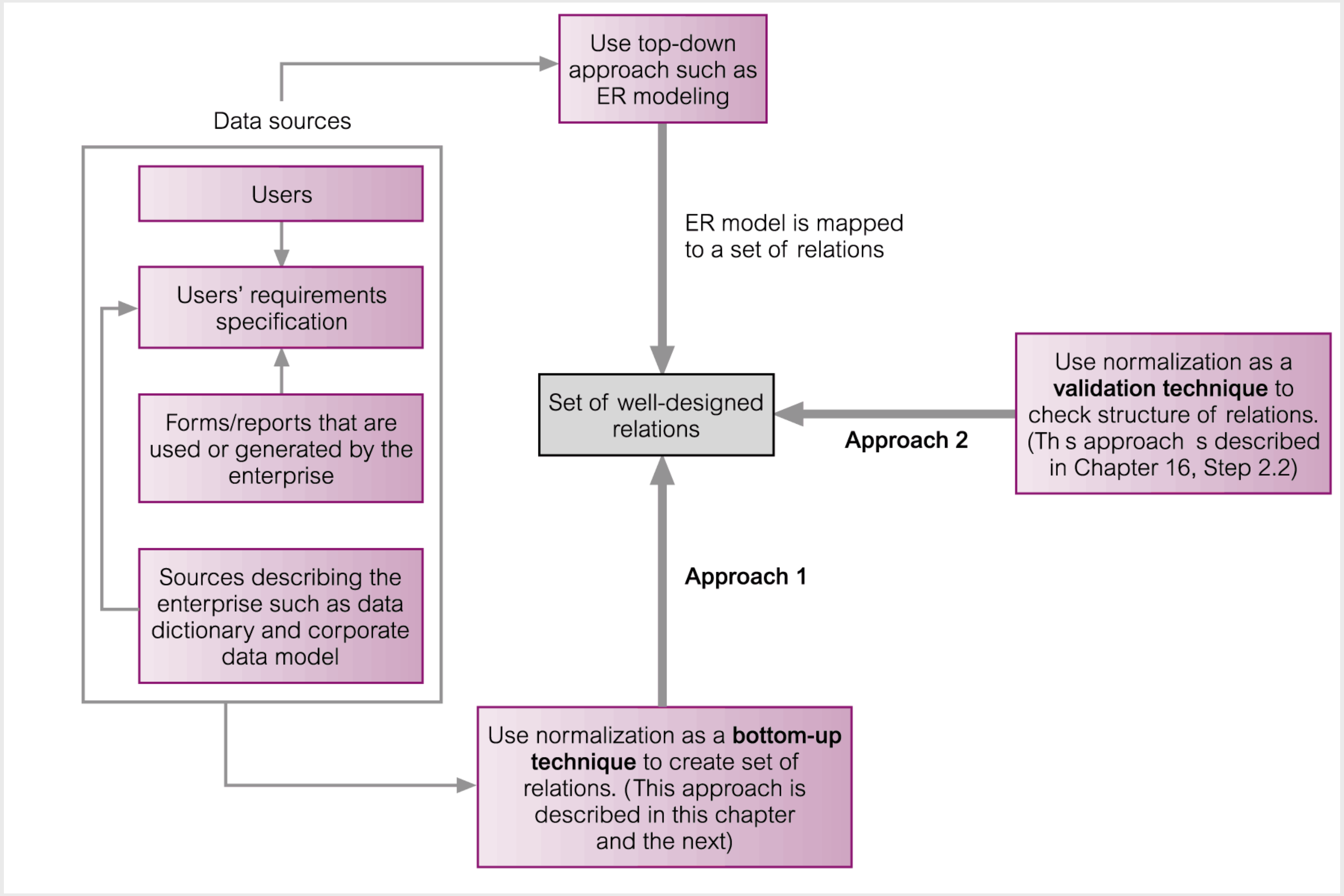
- Normalization attempts to

  minimize the likelihood of introducing inconsistent data into the database

  by

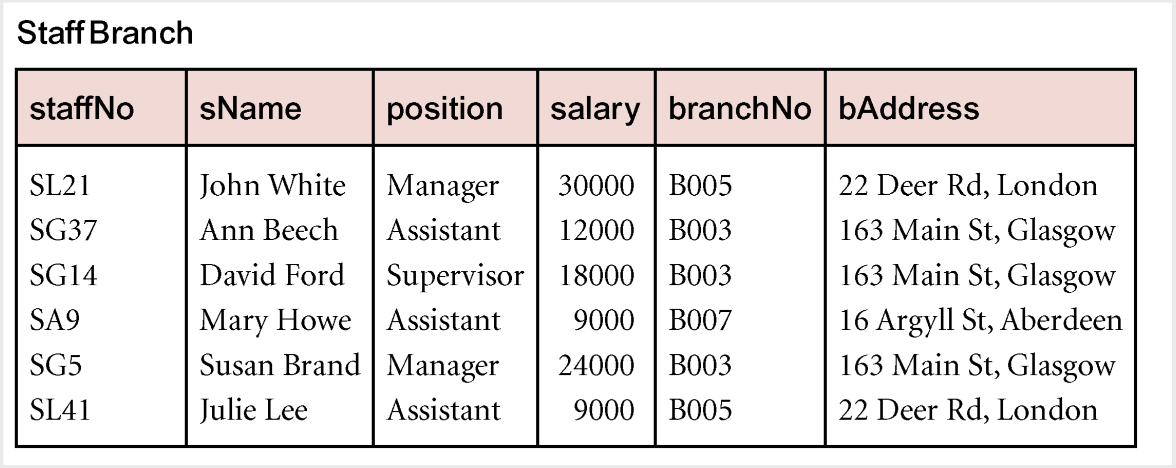
  minimizing the amount of redundancy in the database

How Normalization Supports Database Design


Use top-down approach such as ER modeling
Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise
Sources describing the enterprise such as data dictionary and corporate data model
ER model is mapped to a set of relations
Set of well-designed relations
Approach 2
Use normalization as a validation technique to check structure of relations. (Th s approach s described in Chapter 16, Step 2.2)
Approach 1
Use normalization as a bottom-up technique to create set of relations. (This approach is described in this chapter and the next)

Data Redundancy and Update Anomalies

- **Problems associated with data redundancy are illustrated by looking at the StaffDirectory relation.**



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 Insertion Anomalies

- **StaffDirectory relation has redundant data; the details of a branch are repeated for every member of staff.**
- **If I want to insert a branch without any staff, I must insert NULL values for the staff**
  - **Since NULL values are not allowed in staffNo, a primary key, the insert fails**

Update and Deletion Anomalies

- Update Anomaly: If I change a person’s branch number, I must remember to also change their branch address
- Update Anomaly: If branch’s address changes, it must be updated multiple times
- Deletion Anomaly: If all staff associated with a branch are deleted, the branch details gets deleted as well

Data Redundancy and Update Anomalies

- Relations that contain redundant information may potentially suffer from update anomalies.

- Types of update anomalies include
  - Insertion
  - Deletion
  - Modification

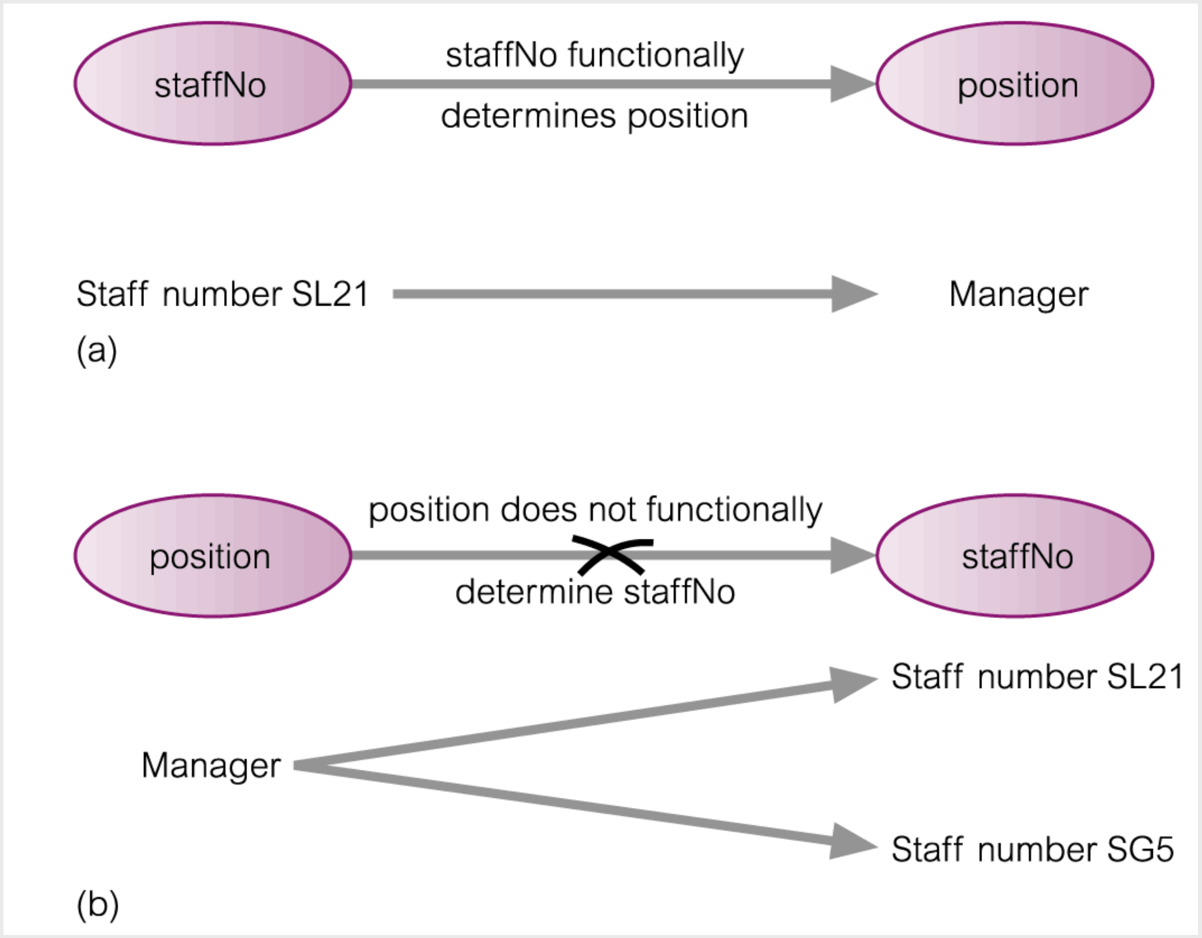
Lossless-join and Dependency Preservation Properties

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

Functional Dependencies

- Functional dependency describes relationship among 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



Determinant

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

Functional Dependencies exist only if they hold for all time

- Consider the values shown in staffNo and sName attributes of the StaffBranch relation
- 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:

$$\text{staffNo} \rightarrow \text{sName}$$

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.*

Example of a Partial Dependency

- **The following functional dependency exists in the StaffBranch relation**

  staffNo, sName → branchNo

- **However, branchNo is also functionally dependent on a subset of (staffNo, sName), namely staffNo. 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 left-hand side and right-hand side attributes**
  - **Holds for *all* time.**
  - **The determinant has the *minimal* number of necessary attributes**

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 $A \rightarrow B$ and $B \rightarrow C$, then C is transitively dependent on A via B

Example Transitive Dependency

- Consider functional dependencies in the StaffDirectory relation

  staffNo → sName, position, salary, branchNo
  branchNo → bAddress

- bAddress has a transitive dependency, on staffNo via branchNo.

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

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

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

- The functional dependencies for the StaffDirectory relation are:

staffNo → sName, position, salary, branchNo

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 StaffDirectory Relation

- 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 StaffDirectory 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 StaffDirectory relation, is staffNo, as *all* other attributes of the relation are functionally dependent on staffNo.**

The Process of Normalization

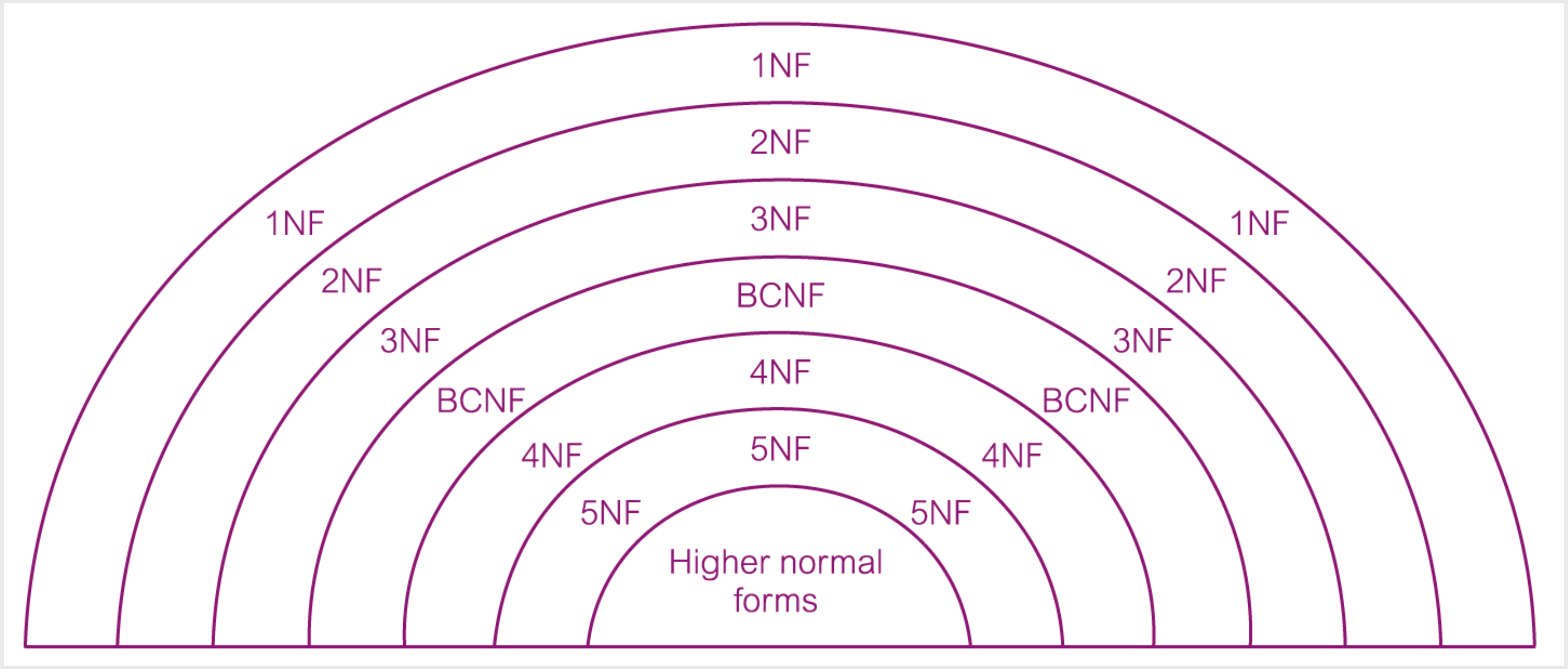
- Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation.

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

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



Unnormalized Form (UNF)

- A table that contains multi-valued attributes or repeating groups (i.e., multiple values in one cell) (see pg. 420 of the handout)
  - Multiple phone numbers for a branch
  - Multiple properties for each renter (client)
- 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.
- Allows 2 or more entities to be conflated in the same relation

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
  - Creating for each row a number of columns equal to the maximum occurrences of a value in a multi-valued attribute.
    - » If there are at most 3 phone #'s, create 3 columns
    - » Enter appropriate data into the 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)

- Conceptual Definition: A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.
- Operational Transformation from 1NF: Remove all partial dependencies on the primary key

1NF to 2NF

- Identify the primary key for the 1NF 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.

Example: A sample Lease relation with functional dependencies

clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName

Primary Key Dependency
clientNo, propertyNo → rentStart, rentFinish

Partial Dependencies
clientNo → cName
propertyNo → pAddress, rent, ownerNo

Transitive Dependencies
ownerNo → oName

Alternative Candidate Keys
clientNo, rentStart → propertyNo, rentFinish
clientNo, rentFinish → propertyNo, rentStart
propertyNo, rentStart → clientNo, rentFinish
propertyNo, rentFinish → clientNo, rentStart

Example: 1st -> 2nd Normal Form

- The partial dependency of cName on clientNo means that we should place these two attributes in a new relation and remove cName from Lease
- The partial dependency

  propertyNo → pAddress, rent, ownerNo, oName

  means these attributes should be placed in a new relation and remove pAddress, rent, ownerNo, and oName from Lease

1NF -> 2NF: For each partial dependency

- the right hand side attributes are deleted from the original relation and are moved to the new relation
- The left hand side attributes, the determinant, both remain in the original relation and are copied to the new relation
  - The lhs attributes become the foreign key in the new relation
  - This preserves the "lossless join" property
  - This also preserves the dependency because all attributes in the dependency are in the new relation

New Relations

Client

clientNo	cName

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName

Lease

clientNo	propertyNo	rentStart	rentFinish

Note that original Lease relation can be reconstituted via joins

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.**
- **Conceptual Definition: Every entity is in its own relation**

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName

The above relation is not in 3NF because of the transitive dependency ownerNo→oName

2NF to 3NF

- **Identify the functional dependencies in the 2NF relation where the determinant (LHS) is not part of the primary key**

- **Remove these functional dependencies by placing them in a new relation along with a copy of their determinant.**

Example

- We convert ProperyOwner to 3NF by creating a new relation with the attributes (ownerNo, oName) and deleting oName from PropertyOwner

PropertyForRent

propertyNo	pAddress	rent	ownerNo

Owner

ownerNo	oName

Summary

- 3 Types of Functional Dependencies
  - Partial dependencies: used to convert 1NF to 2nd NF
  - Transitive dependencies: used to convert 2nd NF to 3rd NF
  - Candidate key dependencies: one of these dependencies becomes the primary key.
    - » The remaining candidate key dependencies are not used in the normalization process
    - » Once 3rd NF is reached, the original universal relation should only have the attributes in the primary key dependency

General Definitions of 2NF and 3NF

- Previous definitions in these slides applied only to *primary keys*. More general definitions follow:
- 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*.
- 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*.

Trade-Off between PK Normal Form and General Normal Form

- PK Normalization is simpler
- General Normalization provides more opportunities for eliminating redundancies and data inconsistencies

Boyce Codd Normal Form

- When column A depends on column B, then B is unique
  - 3rd normal form requires that non-key columns depend exclusively on unique columns
  - Boyce Codd normal form requires that key columns also depend exclusively on unique columns
  - A relation in 3rd Normal Form is normally in Boyce Codd Normal Form. Only if every attribute is part of a candidate key is it possible for a 3rd NF relation to not be in BC NF

Example of Table That Is Not in Boyce Codd Normal Form

EmployeeID	DriversLicenseNumber	State	Name	PostalCode
489	AB7325	IL	Lisa Ellison	60415
517	N3259211	CA	Sam Snead	90295
600	B16629045	CA	Malia Efrenza	90295
777	8242103	TX	Nadia Shah	75185
929	8242103	FL	Maria Rodriguez	32099
933	AX493200	CA	Jiho Chen	94701

- EmployeeId, (DriversLicenseNumber, State) and (DriversLicenseNumber, PostalCode) are keys
- In the US a postal code uniquely determines a state so PostalCode --> State is a functional dependency
- State depends on PostalCode, but Postal Code is not unique in this relation—notice that there are duplicate (State, Postal Code pairs)

Transformation to Boyce Codd Normal Form

- Move (PostalCode, State) to a separate relation

EmployeeID	DriversLicenseNumber	Name	PostalCode
489	AB7325	Lisa Ellison	60415
517	N3259211	Sam Snead	90295
600	B16629045	Malia Efrenza	90295
777	8242103	Nadia Shah	75185
929	8242103	Maria Rodriguez	32099
933	AX493200	Jiho Chen	94701

PostalCode	State
60415	IL
90295	CA
75185	TX
32099	FL
94701	CA