Unit 6:

Normalization of Database Tables

Learning Objectives

- In this chapter, you will learn:
  - What normalization is and what role it plays in the database design process
  - About the normal forms 1NF, 2NF, 3NF, ~~BCNF, and 4NF~~
  - How normal forms can be transformed from lower normal forms to higher normal forms
  - That normalization and ER modeling are used concurrently to produce a good database design
  - That some situations require denormalization to generate information efficiently

Normalization (1 of 2)

- The process of evaluating and correcting table structures to minimize data redundancies
- Reduces data anomalies
- Assigns attributes to tables based on determination
  - Determination is the state in which knowing the value of one attribute makes it possible to determine the value of another (Unit 3).
- Works through a series of stages called Normal forms
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)

Normalization (2 of 2)

- Structural point of view of normal forms
  - Higher normal forms are better than lower normal forms
  - 2NF is better than 1NF; 3NF is better than 2NF
  - For most business database design purposes, 3NF is as high as we need to go in normalization process
  - Highest level of normalization is not always most desirable
- **Denormalization**: Produces a lower normal form
  - Results in increased performance and greater data redundancy

Need for Normalization

- Used while designing a new database structure
  - Analyzes the relationship among the attributes within each entity
  - Determines if the structure can be improved

- Improves the existing data structure and creates an appropriate database design

Normalization Process (1 of 2)

- Objective is to ensure that each table conforms to the concept of well-formed relations
  - Each table represents a single subject
    - For example, a STUDENT table will contain only student data.
  - No data item will be unnecessarily stored in more than one table
    - Minimum controlled redundancy
  - All nonprime attributes in a table are dependent on the primary key
    - The entire primary key; nothing but the primary key
  - Each table is void of insertion, update, and deletion anomalies
    - Ensures integrity and consistency

Prime vs Nonprime Attributes

- Prime attribute
  - An attribute that is part of any candidate key or is the whole key.

- Nonprime (nonkey) attribute
  - An attribute that is not part of any candidate key.

Normalization Process (2 of 2)

- Ensures that all tables are in at least 3NF
- Higher forms are not likely to be encountered in business environment
- Works one relation at a time
- Starts by:
  - Identifying the dependencies of a relation (table)
  - Progressively breaking the relation into new set of relations

Table 6.2 - Normal Forms

NORMAL FORM	CHARACTERISTIC	SECTION
First normal form (1 NF)	Table format, no repeating groups, and PK identified	6.3.1
Second normal form (2NF)	1NF and no partial dependencies	6.3.2
Third normal form (3NF)	2NF and no transitive dependencies	6.3.3
Boyce-Codd normal form (BCNF)	Every determinant is a candidate key (special case of 3NF)	6.6.1
Fourth normal form (4NF)	3NF and no independent multivalued dependencies	6.6.2

RECAP: Dependencies

- **Functional dependence**: Value of one or more attributes determines the value of one or more other attributes
  - **Determinant**: Attribute whose value determines another
  - **Dependent**: Attribute whose value is determined by the other attribute
- **Full functional dependence**: Entire collection of attributes in the determinant is necessary for the relationship

RECAP: Exercise: Full Functional Dependence

- Which of these relationships exhibits full functional dependence

$$STU\_NUM \rightarrow STU\_GPA$$

$$(STU\_NUM, STU\_LNAME) \rightarrow STU\_GPA$$

RECAP: Exercise: Full Functional Dependence

- Which of these relationships exhibits full functional dependence

$$STU\_NUM \rightarrow STU\_GPA$$

$$(STU\_NUM, STU\_LNAME) \rightarrow STU\_GPA$$

Table 6.3 - Functional Dependence Concepts

TABLE 6.3

FUNCTIONAL DEPENDENCE CONCEPTS

CONCEPT	DEFINITION
Functional dependence	The attribute *B* is fully functionally dependent on the attribute *A* if each value of *A* determines one and only one value of *B*. Example: PROJ_NUM → PROJ_NAME (read as *PROJ_NUM functionally determines PROJ_NAME*) In this case, the attribute PROJ_NUM is known as the determinant attribute, and the attribute PROJ_NAME is known as the dependent attribute.
Functional dependence (generalized definition)	Attribute *A* determines attribute *B* (that is, *B* is functionally dependent on *A*) if all (generalized definition) of the rows in the table that agree in value for attribute *A* also agree in value for attribute *B*.
Fully functional dependence (composite key)	If attribute *B* is functionally dependent on a composite key *A* but not on any subset of that composite key, the attribute *B* is fully functionally dependent on *A*.

CENGAGE

Types of Functional Dependencies -1

- **Partial dependency**: Functional dependence in which the determinant is only part of the primary key (or a candidate key)
  - Assumption - One candidate key
  - Straight forward
  - Easy to identify
  - Example
    - if (A, B) → (C, D), B → C, and (A, B) is the primary key,
    - then the functional dependence B → C is a partial dependency because only part of the primary key (B) is needed to determine the value of C.

Types of Functional Dependencies - 2

- **Transitive dependency**: An attribute functionally depends on another nonkey attribute
  - $X \rightarrow Y$, $Y \rightarrow Z$, are functional dependencies and X is the primary key.
  - the dependency $X \rightarrow Z$ is a transitive dependency
    - because X determines the value of Z via Y.
  - The actual transitive dependency is $X \rightarrow Z$.
  - However, the dependency $Y \rightarrow Z$ signals that a transitive dependency exists.
  - For simplicity, during normalization, we'll refer to the *signaling* dependency ($Y \rightarrow Z$) as the transitive dependency

CASE STUDY: Construction Company

- Manages several building projects
- Each project has
  - a project number, name, assigned employee etc
- An employee has
  - employee number, name, and job classification,

CASE STUDY: Construction Company

- Charges its clients by billing hours spent on each contract
- Hourly billing rate is dependent on employee's position
- Periodically, report is generated that contains information displayed in Table 6.1

TABLE 6.1

A SAMPLE REPORT LAYOUT

PROJECT NUMBER	PROJECT NAME	EMPLOYEE NUMBER	EMPLOYEE NAME	JOB CLASS	CHARGE/ HOUR	HOURS BILLED	TOTAL CHARGE
15	Evergreen	103	June E. Arbough	Elec. Engineer	$ 84.50	23.8	$ 2,011.10
		101	John G. News	Database Designer	$105.00	19.4	$ 2,037.00
		105	Alice K. Johnson *	Database Designer	$105.00	35.7	$ 3,748.50
		106	William Smithfield	Programmer	$ 35.75	12.6	$ 450.45
		102	David H. Senior	Systems Analyst	$ 96.75	23.8	$ 2,302.65
				Subtotal			**$10,549.70**
18	Amber Wave	114	Annelise Jones	Applications Designer	$ 48.10	24.6	$ 1,183.26
		118	James J. Frommer	General Support	$ 18.36	45.3	$ 831.71
		104	Anne K. Ramoras *	Systems Analyst	$ 96.75	32.4	$ 3,134.70
		112	Darlene M. Smithson	DSS Analyst	$ 45.95	44.0	$ 2,021.80
				Subtotal			**$ 7,171.47**
22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.7	$ 6,793.50
		104	Anne K. Ramoras	Systems Analyst	$96.75	48.4	$ 4,682.70
		113	Delbert K. Joenbrood *	Applications Designer	$48.10	23.6	$ 1,135.16
		111	Geoff B. Wabash	Clerical Support	$26.87	22.0	$ 591.14
		106	William Smithfield	Programmer	$35.75	12.8	$ 457.60
				Subtotal			**$13,660.10**
25	Starflight	107	Maria D. Alonzo	Programmer	$ 35.75	24.6	$ 879.45
		115	Travis B. Bawangi	Systems Analyst	$ 96.75	45.8	$ 4,431.15
		101	John G. News *	Database Designer	$105.00	56.3	$ 5,911.50
		114	Annelise Jones	Applications Designer	$ 48.10	33.1	$ 1,592.11
		108	Ralph B. Washington	Systems Analyst	$ 96.75	23.6	$ 2,283.30
		118	James J. Frommer	General Support	$ 18.36	30.5	$ 559.98
		112	Darlene M. Smithson	DSS Analyst	$ 45.95	41.4	$ 1,902.33
				Subtotal			**$17,559.82**
				Total			**$48,941.09**

Note: A * indicates the project leader.

Deficiencies in the Table

- The project number (PROJ_NUM) is apparently intended to be a primary key (PK) or at least a part of a PK, but it contains nulls.
- The table entries invite data inconsistencies.
  - For example, the JOB_CLASS value "Elect. Engineer" might be entered as "Elect.Eng." in some cases, "El. Eng." in others, and "EE" in still others.
- The table displays data redundancies that yield the following anomalies:
  - *Update anomalies*. Modifying the JOB_CLASS for employee number 105 requires many potential alterations, one for each EMP_NUM = 105.
  - *Insertion anomalies*. Just to complete a row definition, a new employee must be assigned to a project.
  - *Deletion anomalies*. Suppose that only one employee is associated with a given project. If that employee leaves the company and the employee data is deleted, then project information will also be deleted.

Conversion to First Normal Form (1 of 3)

- **Repeating group**: Group of multiple entries of same type can exist for any single key attribute occurrence
  - Existence proves the presence of data redundancies
  - In Figure 6.1, note that each single project number (PROJ_NUM) occurrence can reference a group of related data entries.
  - For example, the Evergreen project (PROJ_NUM = 15) shows five entries at this point—and those entries are related because they each share the PROJ_NUM = 15 characteristic.
  - Each time a new record is entered for the Evergreen project, the number of entries in the group grows by one.

Conversion to First Normal Form (1 of 3)

- Enable reducing data redundancies

- Steps
  1. Eliminate the repeating groups
  2. Identify the primary key
  3. Identify all dependencies

Conversion to First Normal Form (2 of 3)

- **Dependency diagram**: Depicts all dependencies found within given table structure
  - Helps to get an overview of all relationships among table's attributes
  - Makes it less likely that an important dependency will be overlooked

Conversion to First Normal Form (3 of 3)

- 1NF describes tabular format in which:
  - All key attributes are defined
  - There are no repeating groups in the table
  - All attributes are dependent on the primary key
- All relational tables satisfy 1NF requirements
- Some tables contain partial dependencies
  - Subject to data redundancies and various anomalies

Step 1: Eliminate the Repeating Groups

- Start by presenting the data in a tabular format, where each cell has a single value and there are no repeating groups.
- To eliminate the repeating groups, eliminate the nulls by making sure that each repeating group attribute contains an appropriate data value.
- That change converts the table in Figure 6.1 to 1NF in Figure 6.2.

Step 2: Identify the Primary Key

- PROJ_NUM and EMP_NUM

Step 3: Identify All Dependencies

- PROJ_NUM, EMP_NUM → PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS
  - Based on the primary key
  - PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, and HOURS values are all dependent on—they are determined by—the combination of PROJ_NUM and EMP_NUM.

- PROJ_NUM → PROJ_NAME
  - Partial dependency
- EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR
  - Partial dependence

- JOB_CLASS → CHG_HOUR
  - This dependency exists between two nonprime attributes; therefore, it is a signal that a transitive dependency exists, and we will refer to it as a transitive dependency.

CENGAGE

A Table not in 1NF

FIGURE 6.1 TABULAR REPRESENTATION OF THE REPORT FORMAT

Table name: RPT_FORMAT **Database name: Ch06_ConstructCo**

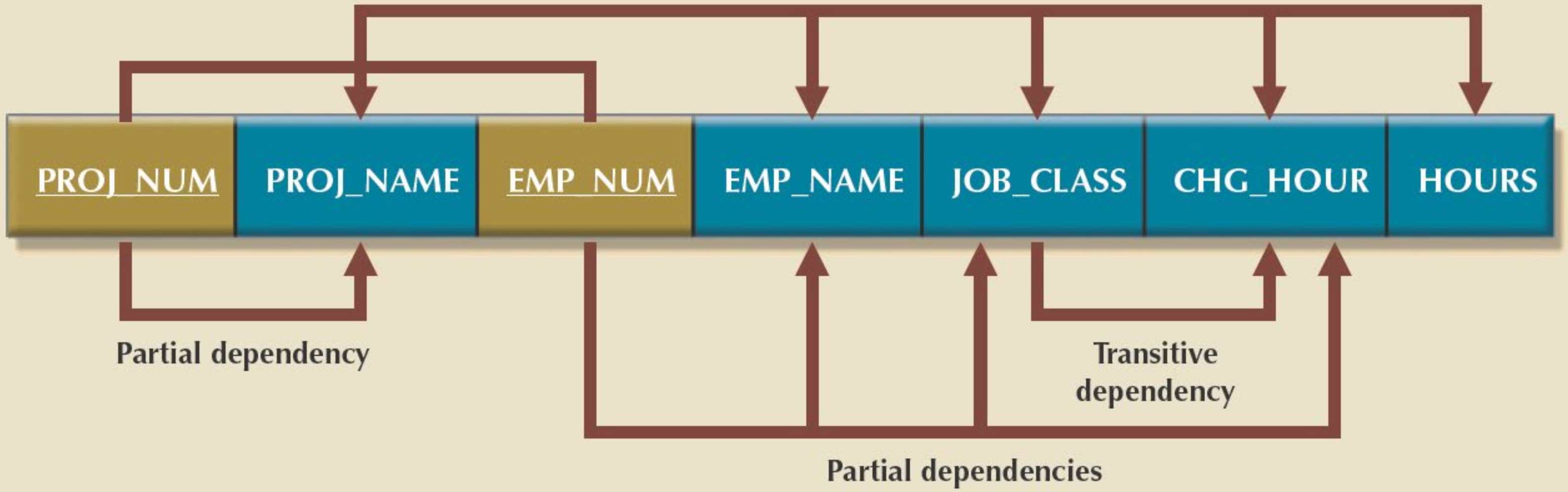
PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

FIGURE 6.2 A TABLE IN FIRST NORMAL FORM

Table name: DATA_ORG_1NF **Database name: Ch06_ConstructCo**

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
15	Evergreen	101	John G. News	Database Designer	105.00	19.4
15	Evergreen	105	Alice K. Johnson *	Database Designer	105.00	35.7
15	Evergreen	106	William Smithfield	Programmer	35.75	12.6
15	Evergreen	102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
18	Amber Wave	118	James J. Frommer	General Support	18.36	45.3
18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	96.75	48.4
22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	26.87	22.0
22	Rolling Tide	106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
25	Starflight	115	Travis B. Bawangi	Systems Analyst	96.75	45.8
25	Starflight	101	John G. News *	Database Designer	105.00	56.3
25	Starflight	114	Annelise Jones	Applications Designer	48.10	33.1
25	Starflight	108	Ralph B. Washington	Systems Analyst	96.75	23.6
25	Starflight	118	James J. Frommer	General Support	18.36	30.5
25	Starflight	112	Darlene M. Smithson	DSS Analyst	45.95	41.4

Figure 6.3 - First Normal Form (1NF) Dependency Diagram



1NF (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOURS, HOURS)

PARTIAL DEPENDENCIES:
(PROJ_NUM → PROJ_NAME)
(EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR)

TRANSITIVE DEPENDENCY:
(JOB_CLASS → CHG_HOUR)

Conversion to First Normal Form

- First normal form describes tabular format in which:
  - All key attributes are defined
  - There are no repeating groups in the table
  - All attributes are dependent on primary key
- All relational tables satisfy 1NF requirements
- Some tables contain partial dependencies
  - Dependencies based on only part of the primary key
  - Sometimes used for performance reasons, but should be used with caution
  - Still subject to data redundancies

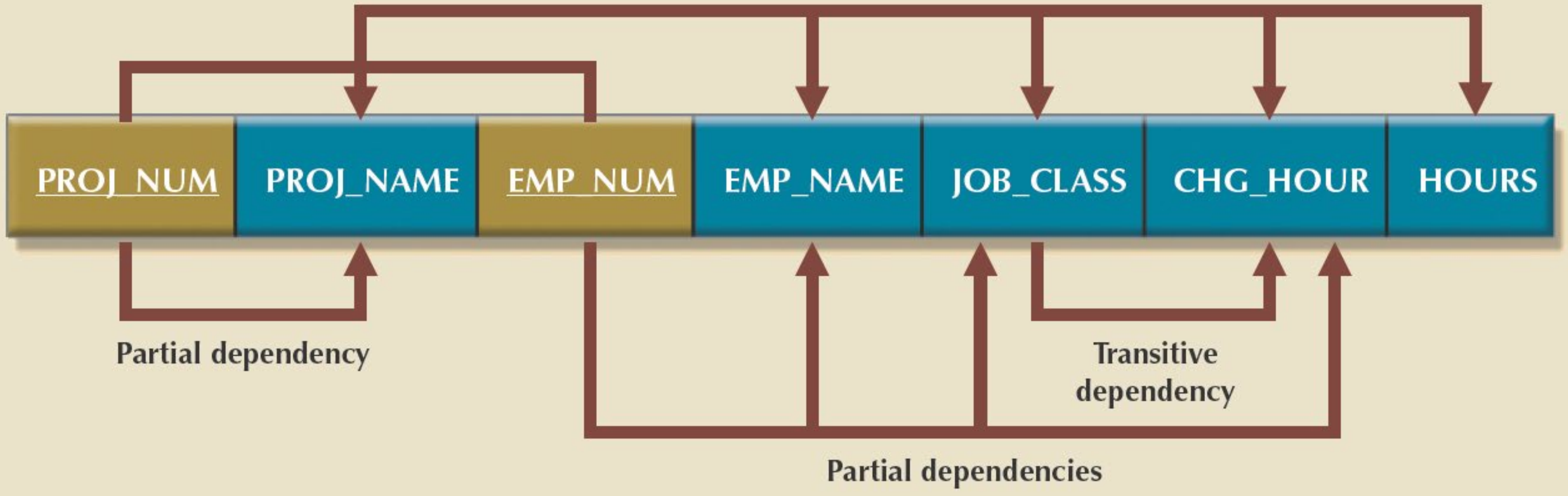
Conversion to Second Normal Form

- Steps
  - Make new tables to eliminate partial dependencies
  - Reassign corresponding dependent attributes
- Table is in 2NF when it:
  - Is in 1NF
  - Includes no partial dependencies

Step 1: Make New Tables to Eliminate Partial Dependencies

- For each component of the primary key that acts as a determinant in a partial dependency,
  - create a new table with a copy of that component as the primary key.
  - they also remain in the original table as well.
    - PROJ_NUM
    - EMP_NUM
    - PROJ_NUM, EMP_NUM
- In other words, the original table is now divided into three tables

Figure 6.3 - First Normal Form (1NF) Dependency Diagram



1NF (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOURS, HOURS)

PARTIAL DEPENDENCIES:
(PROJ_NUM ➡ PROJ_NAME)
(EMP_NUM ➡ EMP_NAME, JOB_CLASS, CHG_HOUR)

TRANSITIVE DEPENDENCY:
(JOB_CLASS ➡ CHG_HOUR)

Step 2: Reassign Corresponding Dependent Attributes

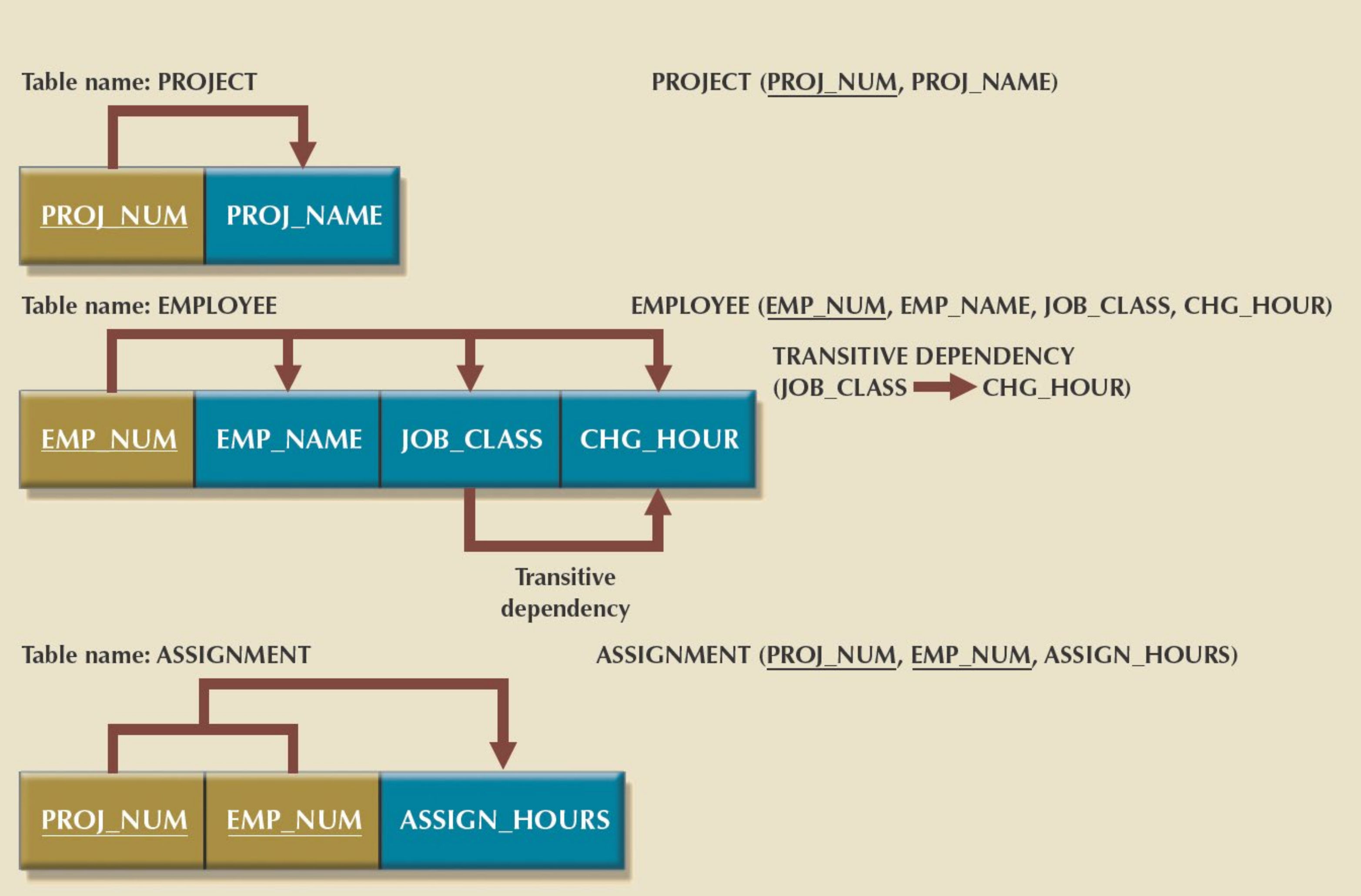
- From the dependency diagram in the 1NF
  - Determine attributes that are dependent in the partial dependencies.
  - The attributes that are dependent in a partial dependency are removed from the original table and placed in the new table with the dependency's determinant.
  - Any attributes that are not dependent in a partial dependency will remain in the original table.

Step 2: Reassign Corresponding Dependent Attributes

- Give the new tables appropriate names
- They are described by the following relational schemas:
    - PROJECT (**<u>PROJ_NUM</u>**, PROJ_NAME)
    - EMPLOYEE (**<u>EMP_NUM</u>**, EMP_NAME, JOB_CLASS, CHG_HOUR)
    - ASSIGNMENT (**<u>PROJ_NUM</u>**, **<u>EMP_NUM</u>**, ASSIGN_HOURS)

- By leaving the determinants in the original table as well as making them the primary keys of the new tables, primary key/foreign key relationships have been created.

Figure 6.4 - Second Normal Form (2NF) Conversion Results



Conversion to Third Normal Form

- Steps
  - Make new tables to eliminate transitive dependencies
  - Reassign corresponding dependent attributes
- Table is in 3NF when it:
  - Is in 2NF
  - Contains no transitive dependencies

Step 1: Make New Tables to Eliminate Transitive Dependencies

- For every transitive dependency, write a copy of its determinant as a primary key for a new table.
- If you have three different transitive dependencies, you will have three different determinants.
- As with the conversion to 2NF, it is important that the determinant remain in the original table to serve as a foreign key.
- Our example contains only one transitive dependency, therefore we write its determinant:
  - JOB_CLASS

Step 2: Reassign Corresponding Dependent Attributes

- Identify the attributes that are dependent on each determinant identified in Step 1.
- Place the dependent attributes in the new tables with their determinants and remove them from their original tables.
- In this example, eliminate CHG_HOUR from the EMPLOYEE table
  - JOB (**<u>JOB_CLASS</u>**, CHG_HOUR) …new table
  - EMPLOYEE (**<u>EMP_NUM</u>**, EMP_NAME, JOB_CLASS)

Conversion to Third Normal Form

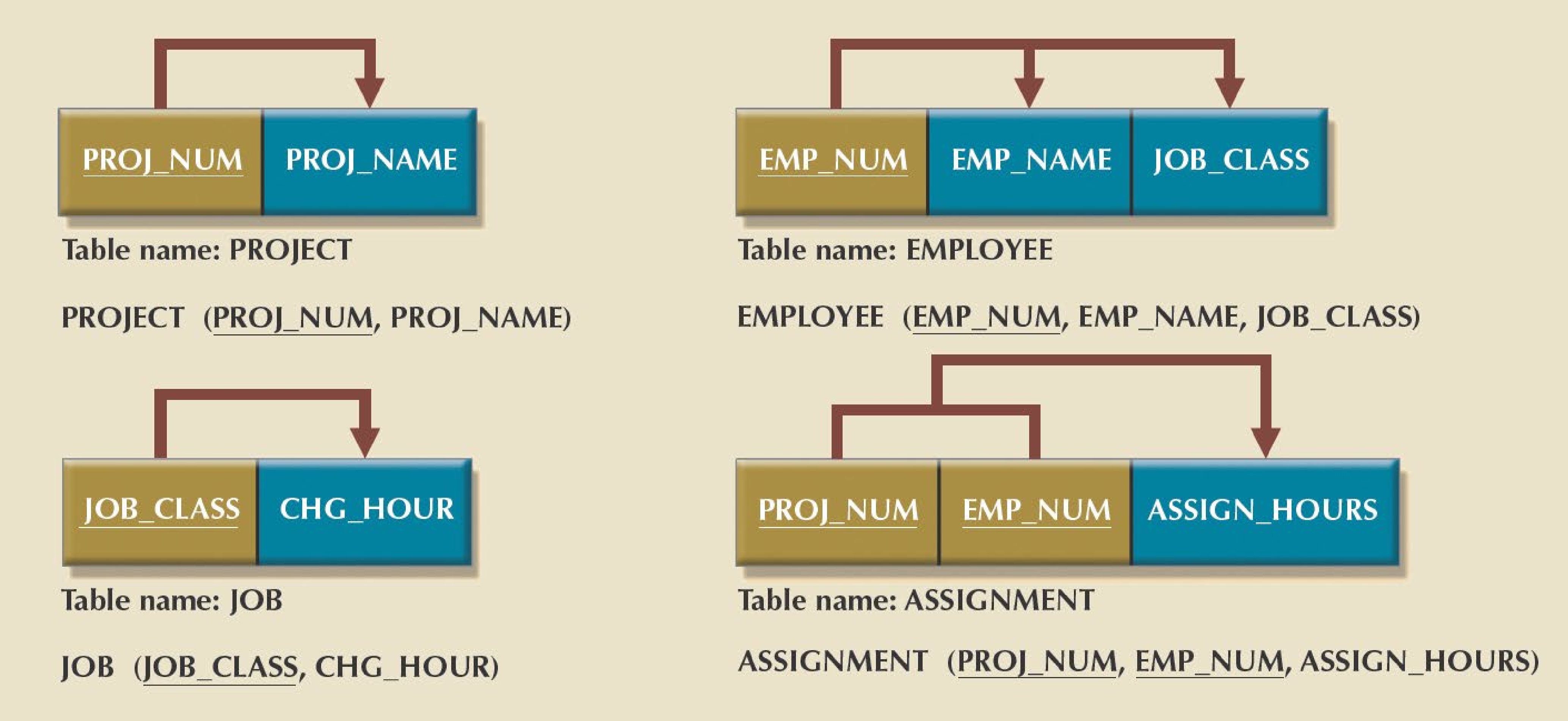
- Draw a new dependency diagram
- Name the table to reflect its contents and function. In this case, JOB seems appropriate.
- Check all of the tables to make sure that each table has a determinant and that no table contains inappropriate dependencies.

Conversion to Third Normal Form

- PROJECT (**<u>PROJ_NUM</u>**, PROJ_NAME)
- EMPLOYEE (**<u>EMP_NUM</u>**, EMP_NAME, JOB_CLASS)
- JOB (**<u>JOB_CLASS</u>**, CHG_HOUR)
- ASSIGNMENT (**<u>PROJ_NUM, EMP_NUM</u>**, ASSIGN_HOURS)

- Note that this conversion has eliminated the original EMPLOYEE table's transitive dependency.
- The tables are now said to be in third normal form (3NF).

Figure 6.5 - Third Normal Form (3NF) Conversion Results



CENGAGE

Points to Note - 1

- It is imperative that 2NF be achieved before moving on to 3NF;
  - be certain to resolve the partial dependencies before resolving the transitive dependencies.
- The example shown made an assumption that each table has only one candidate key, which is the primary key.
- If a table has multiple candidate keys, then the overall process remains the same, but there are additional considerations.

Points to Note - 2

- For example, if a table has multiple candidate keys and one of them is a composite key, the table can have partial dependencies based on this composite candidate key, even when the primary key chosen is a single attribute.
  - Those dependencies would be perceived as transitive dependencies and would not be resolved until 3NF.
  - However, with you can recognize all candidate keys and their dependencies as such, and resolve them appropriately.

Points to Note - 3

- The existence of multiple candidate keys can also influence the identification of transitive dependencies.
- Previously, a transitive dependency was defined to exist when one nonprime attribute determined another nonprime attribute.
- In the presence of multiple candidate keys, the definition of a nonprime attribute as an attribute that is not a part of any candidate key is critical.
- If the determinant of a functional dependence is not the primary key but is a part of another candidate key, then it is not a nonprime attribute and does not signal the presence of a transitive dependency.

NEXT: Improving the Database Design

Improving the Design

- Evaluate PK assignments
- Evaluate naming conventions
- Refine attribute atomicity
- Identify new attributes
- Identify new relationships
- Refine primary keys as required for data granularity
- Maintain historical accuracy
- Evaluate using derived attributes

Evaluate PK

- EMPLOYEE (<u>**EMP_NUM**</u>, EMP_NAME, JOB_CLASS)
- JOB (<u>**JOB_CLASS**</u>, CHG_HOUR)
- This can create referential integrity problems
  - E.g. entering DB Designer instead of Database Designer
- Create surrogate key
  - JOB_CODE
  - JOB(<u>**JOB_CODE**</u>, JOB_CLASS, CHG_HOUR)
  - Note that JOB_CLASS → CHG_HOUR is not a transitive dependency
    - Because JOB_CLASS is a candidate key

Naming Conventions

- Change CHG_HOUR to JOB_CHG_HOUR
  - To show its association to the JOB table
- JOB_CLASS to JOB_DESCRIPTION
  - A better fit for the attribute
  - Stores values such as Database designer, Systems analyst

Refine attribute atomicity

- Atomic attribute: Cannot be further subdivided
- EMP_NAME to
  - EMP_LNAME
  - EMP_FNAME
  - EMP_INITIALS
- Helps queries, we can sort details based on EMP_LNAME

Identify new attributes

- Look at the employee table, are there other attributes that can be stored for an employee ?
  - Hire_date
  - Salary
  - Etc.

- Do the same for all tables

Identify new relationships

- Do we need a relationship between EMPLOYEE and PROJECT?
  - An EMPLOYEE manages a PROJECT

Surrogate Key Considerations

- Used by designers when the primary key is considered to be unsuitable
- System-defined attribute
- Created and managed via the DBMS
- Have a numeric value which is automatically incremented for each new row

Normalization and Database Design

- Normalization should be part of the design process
- Proposed entities must meet required the normal form before table structures are created
- Principles and normalization procedures to be understood to redesign and modify databases
  - ERD is created through an iterative process
  - Normalization focuses on the characteristics of specific entities

Denormalization (1 of 2)

- Design goals
  - Creation of normalized relations
  - Processing requirements and speed
- Number of database tables expands when tables are decomposed to conform to normalization requirements
- Joining a larger number of tables:
  - Takes additional input/output (I/O) operations and processing logic
  - Reduces system speed

Denormalization (2 of 2)

- Defects in unnormalized tables
  - Data updates are less efficient because tables are larger
  - Indexing is more cumbersome
  - No simple strategies for creating virtual tables known as views

- Conflicts between design efficiency, information requirements, and processing speed are often resolved through compromises that may include denormalization
- Use denormalisation cautiously
- Understand why—under some circumstances—unnormalised tables are better choice

Table 6.7 - Data-Modeling Checklist (1 of 4)

BUSINESS RULES
• Properly document and verify all business rules with the end users. • Ensure that all business rules are written precisely, clearly, and simply. The business rules must help identify entities, attributes, relationships, and constraints. • Identify the source of all business rules, and ensure that each business rule is justified, dated, and signed off by an approving authority.

Table 6.7 - Data-Modeling Checklist (2 of 4)

DATA MODELING

Naming conventions: All names should be limited in length (database-dependent size).
• Entity names:
    • Should be nouns that are familiar to business and should be short and meaningful
    • Should document abbreviations, synonyms, and aliases for each entity
    • Should be unique within the model
    • For composite entities, may include a combination of abbreviated names of the entities linked through the composite entity
• Attribute names:
    • Should be unique within the entity
    • Should use the entity abbreviation as a prefix
    • Should be descriptive of the characteristic
    • Should use suffixes such as _ID, _NUM, or _CODE for the PK attribute
    • Should not be a reserved word
    • Should not contain spaces or special characters such as @, !, or &
• Relationship names:
    • Should be active or passive verbs that clearly indicate the nature of the relationship

CENGAGE

Table 6.7 - Data-Modeling Checklist (3 of 4)

Entities: • Each entity should represent a single subject. • Each entity should represent a set of distinguishable entity instances. • All entities should be in 3NF or higher. Any entities below 3NF should be justified. • The granularity of the entity instance should be clearly defined. • The PK should be clearly defined and support the selected data granularity.
Attributes: • Should be simple and single-valued (atomic data) • Should document default values, constraints, synonyms, and aliases • Derived attributes should be clearly identified and include source(s) • Should not be redundant unless this is required for transaction accuracy, performance, or maintaining a history • Nonkey attributes must be fully dependent on the PK attribute

Table 6.7 - Data-Modeling Checklist (4 of 4)

Relationships: • Should clearly identify relationship participants • Should clearly define participation, connectivity, and document cardinality
ER model: • Should be validated against expected processes: inserts, updates, and deletions • Should evaluate where, when, and how to maintain a history • Should not contain redundant relationships except as required (see attributes) • Should minimize data redundancy to ensure single-place updates • Should conform to the minimal data rule: All that is needed is there, and all that is there is needed.