Chapter 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)

- Evaluating and correcting table structures to minimize data redundancies
- Reduces data anomalies
- Assigns attributes to tables based on determination
- 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
- Properly designed 3NF structures meet the requirement of fourth normal form (4NF)
- **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

Real world examples:

1) You're given a large data dump, perhaps in .CSV (comma-delimited) format, and need to put it into a database for long-term analytics. But first, you need to 'clean it up', or Normalize it
2) You need to ensure that the data you're collected is properly stored for long-term analytics
3) You need to understand and/or redesign someone else's mess

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
  - No data item will be unnecessarily stored in more than one table
  - All nonprime attributes in a table are dependent on the primary key
  - Each table is void of insertion, update, and deletion anomalies

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)	1 NF 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

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 SPROJ_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	If attribute *B* is functionally dependent on a composite key *A* but not on any subset of composite key, the attribute *B* is fully functionally dependent on *A*.

This is a repeat from Chapter 3, Page 76,
but good to review

Types of Functional Dependencies

- **Partial dependency**: Functional dependence in which the determinant is only part of the primary key
  - Assumption - One candidate key
  - Straight forward
  - Easy to identify
- **Transitive dependency**: An attribute functionally depends on another nonkey attribute

Partial Dependencies can ONLY exist when a table's Primary Key is composed of two or more attributes (page 212)

CENGAGE

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
- Enable reducing data redundancies
- Steps
  - Eliminate the repeating groups
  - Identify the primary key
  - Identify all dependencies

Conversion to First Normal Form (2 of 3)

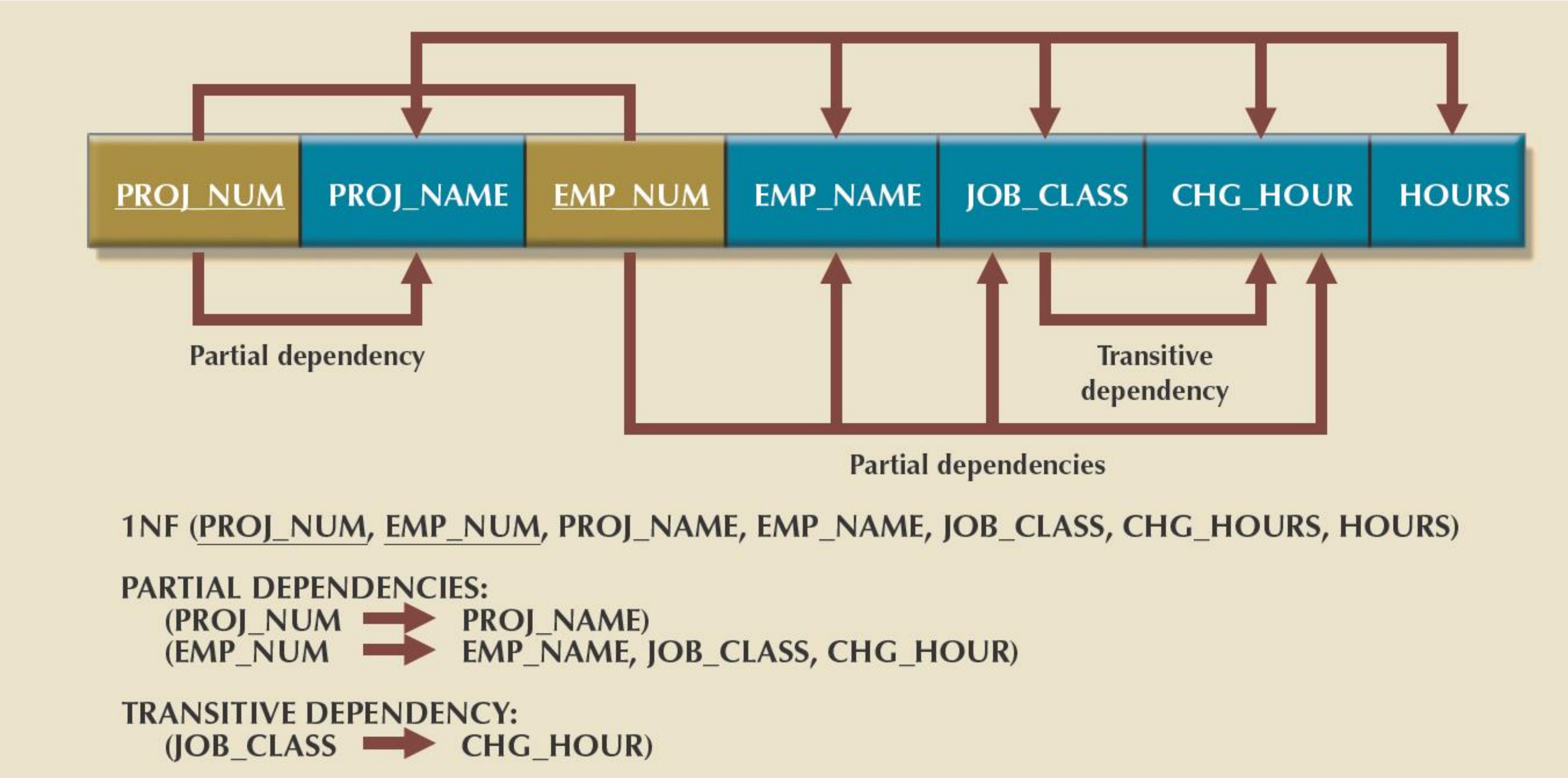
- **Dependency diagram**: Depicts all dependencies found within given table structure
  - Helps to get an overview of all relationships among table's attributes
  - Makes it less likely that an important dependency will be overlooked

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

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

Top Arrows are desirable dependencies



Partial Dependencies will be address in 2NF

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

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

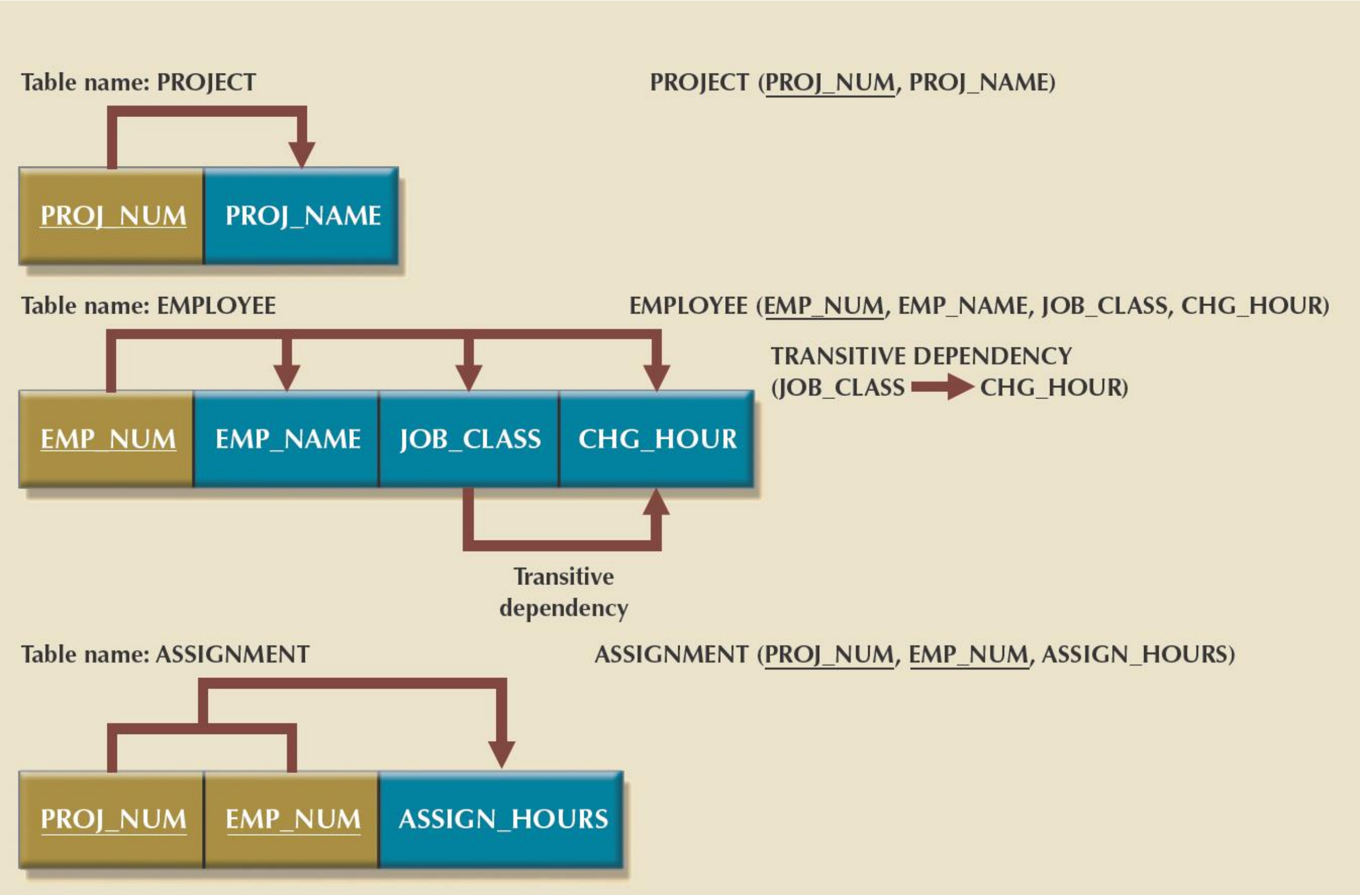
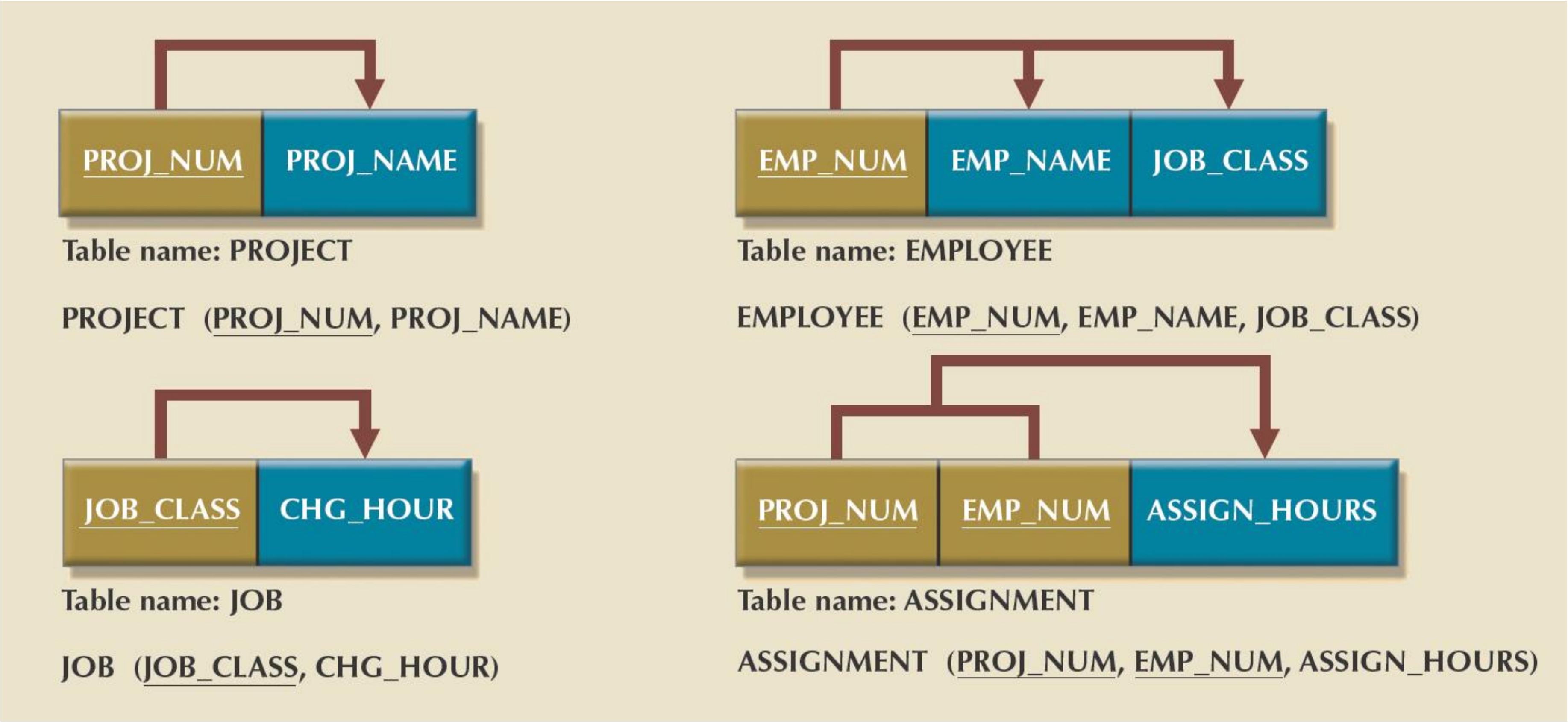


Figure 6.5 - Third Normal Form (3NF) Conversion Results



CENGAGE

Conversion to Third Normal Form

- Steps
  - Make new tables to eliminate transitive dependencies
    - **Determinant**: Any attribute whose value determines other values within a row
  - Reassign corresponding dependent attributes
- Table is in 3NF when it:
  - Is in 2NF
  - Contains no transitive dependencies

Improving the Design

- Evaluate PK assignments
  - Can you make new PK's and relations to prevent possible errors?
- Evaluate naming conventions
  - Change field names based on new tables that they were moved into
- Refine attribute atomicity (**Atomicity**: Characteristic of an atomic attribute)
  - **Atomic attribute**: Cannot be further subdivided
- Identify new attributes and new relationships
- Refine primary keys as required for data granularity
  - **Granularity**: Level of detail represented by the values stored in a table's row

Figure 6.6 - The Completed Database (1 of 2)

Table name: PROJECT

Table name: JOB

Database name: Ch06_ConstructCo

Table name: PROJECT

PROJ_NUM	PROJ_NAME	EMP_NUM
15	Evergreen	105
18	Amber Wave	104
22	Rolling Tide	113
25	Starflight	101

Table name: JOB

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
500	Programmer	35.75
501	Systems Analyst	96.75
502	Database Designer	105.00
503	Electrical Engineer	84.50
504	Mechanical Engineer	67.90
505	Civil Engineer	55.78
506	Clerical Support	26.87
507	DSS Analyst	45.95
508	Applications Designer	48.10
509	Bio Technician	34.55
510	General Support	18.36

Figure 6.6 - The Completed Database (2 of 2)

Table name: ASSIGNMENT

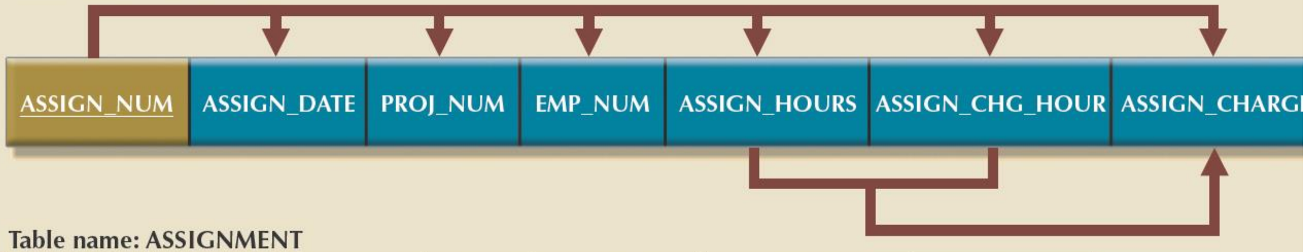


Table name: ASSIGNMENT

ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_HOURS	ASSIGN_CHG_HOUR	ASSIGN_CHARGE
1001	04-Mar-16	15	103	2.6	84.50	219.70
1002	04-Mar-16	18	118	1.4	18.36	25.70
1003	05-Mar-16	15	101	3.6	105.00	378.00
1004	05-Mar-16	22	113	2.5	48.10	120.25
1005	05-Mar-16	15	103	1.9	84.50	160.55
1006	05-Mar-16	25	115	4.2	96.75	406.35
1007	05-Mar-16	22	105	5.2	105.00	546.00
1008	05-Mar-16	25	101	1.7	105.00	178.50
1009	05-Mar-16	15	105	2.0	105.00	210.00
1010	06-Mar-16	15	102	3.8	96.75	367.65
1011	06-Mar-16	22	104	2.6	96.75	251.55
1012	06-Mar-16	15	101	2.3	105.00	241.50
1013	06-Mar-16	25	114	1.8	48.10	86.58
1014	06-Mar-16	22	111	4.0	26.87	107.48
1015	06-Mar-16	25	114	3.4	48.10	163.54
1016	06-Mar-16	18	112	1.2	45.95	55.14
1017	06-Mar-16	18	118	2.0	18.36	36.72
1018	06-Mar-16	18	104	2.6	96.75	251.55
1019	06-Mar-16	15	103	3.0	84.50	253.50
1020	07-Mar-16	22	105	2.7	105.00	283.50
1021	08-Mar-16	25	108	4.2	96.75	406.35
1022	07-Mar-16	25	114	5.8	48.10	278.98
1023	07-Mar-16	22	106	2.4	35.75	85.80

Surrogate Key Considerations

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

The Boyce-Codd Normal Form (BCNF)

- Every determinant in the table should be a candidate key
  - Candidate key - Same characteristics as primary key but not chosen to be the primary key
- Equivalent to 3NF when the table contains only one candidate key
- Violated only when the table contains more than one candidate key
- Considered to be a special case of 3NF

Figure 6.8 - A Table That is in 3NF and not in BCNF

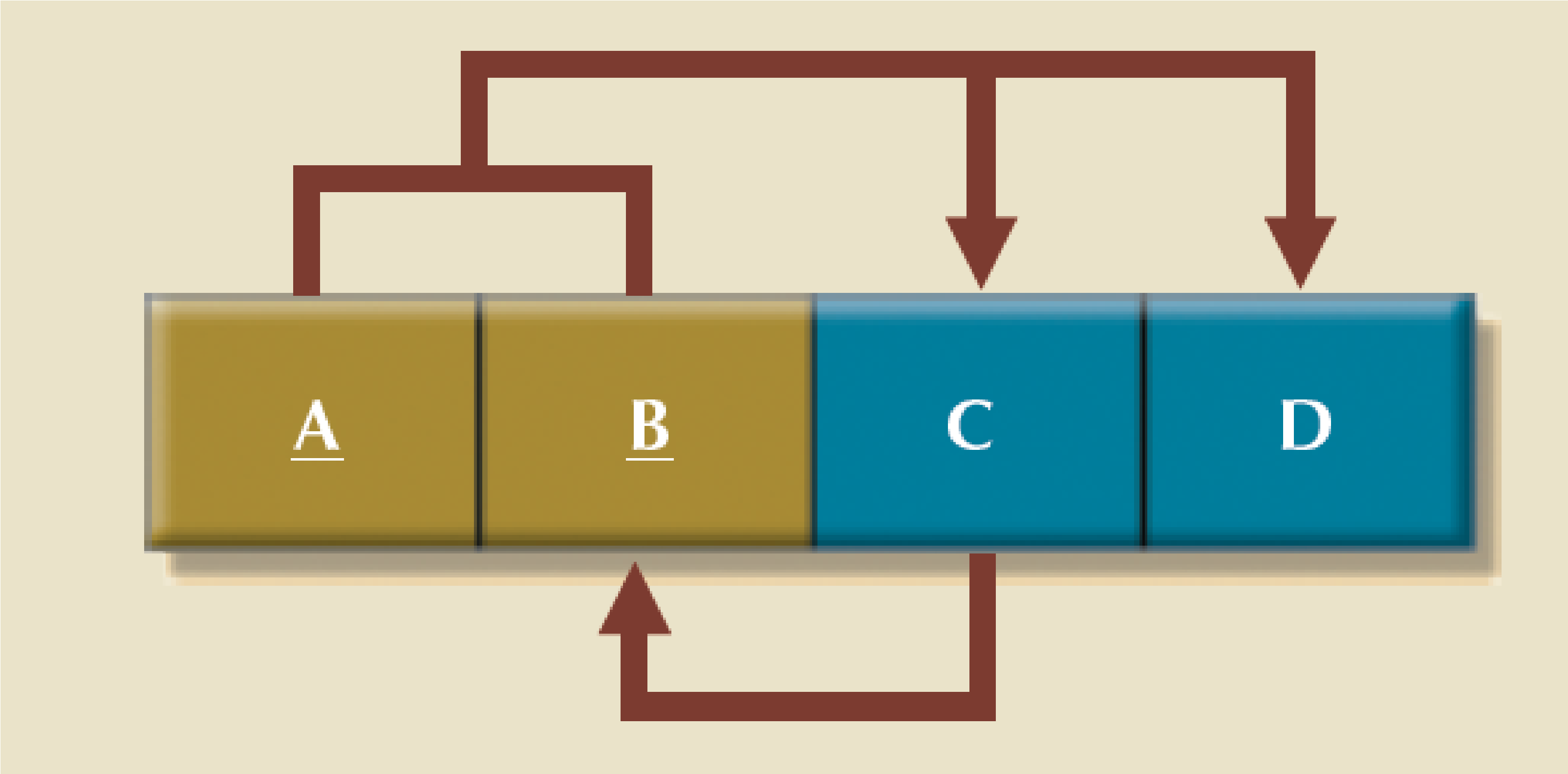


Figure 6.9 – Decomposition to BCNF

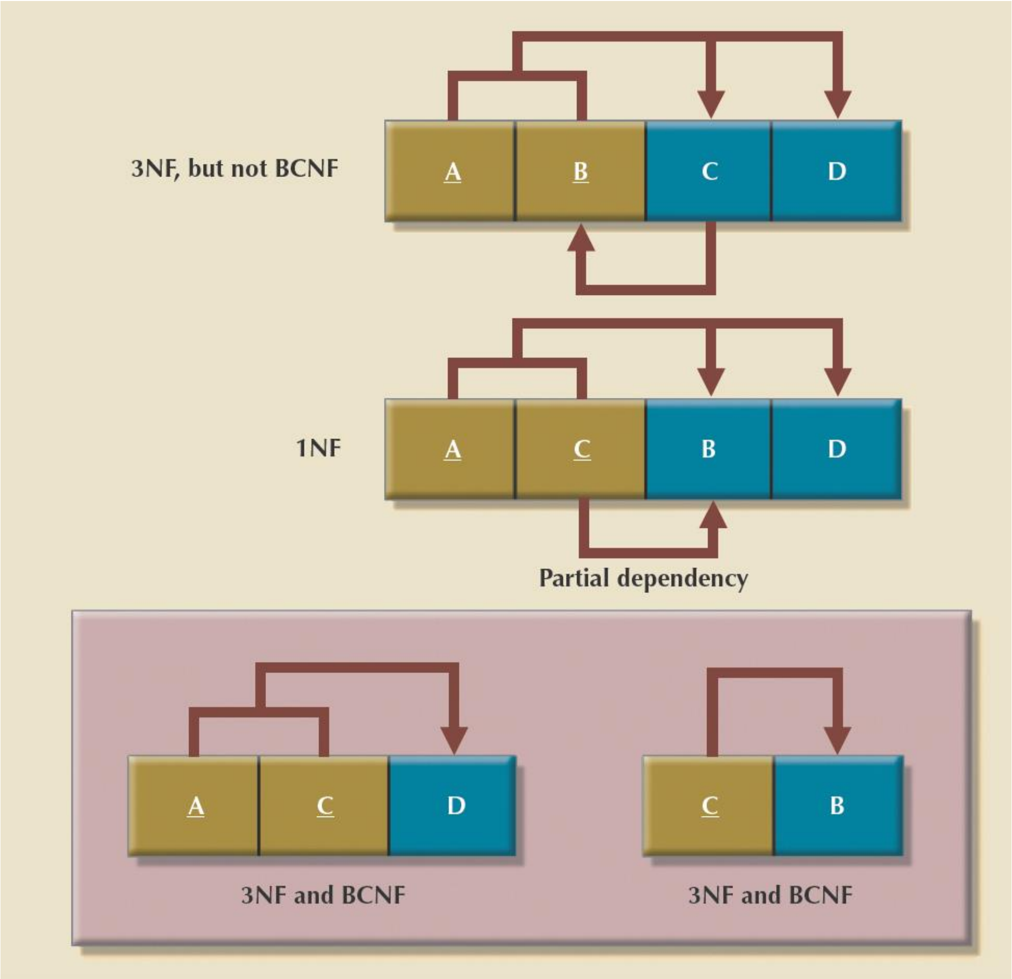


Table 6.5 - Sample Data for a BCNF Conversion

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32546	B

Fourth Normal Form (4NF)

- Table is in 4NF when it:
  - Is in 3NF
  - Has no multivalued dependencies
- Rules
  - All attributes must be dependent on the primary key, but they must be independent of each other
  - No row may contain two or more multivalued facts about an entity

Figure 6.11 - Tables with Multivalued Dependencies

Database name: Ch06_Service

Table name: VOLUNTEER_V1

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	UW	3
10123		4

Table name: VOLUNTEER_V2

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	
10123	UW	
10123		1
10123		3
10123		4

Table name: VOLUNTEER_V3

EMP_NUM	ORG_CODE	ASSIGN_NUM
10123	RC	1
10123	RC	3
10123	UW	4

CENGAGE

Figure 6.12 - A Set of Tables in 4NF (1 of 2)

Database name: CH06_Service

Table name: PROJECT

PROJ_CODE	PROJ_NAME	PROJ_BUDGET
1	BeThere	1023245.00
2	BlueMoon	20198608.00
3	GreenThumb	3234456.00
4	GoFast	5674000.00
5	GoSlow	1002500.00

Table name: ASSIGNMENT

ASSIGN_NUM	EMP_NUM	PROJ_CODE
1	10123	1
2	10121	2
3	10123	3
4	10123	4
5	10121	1
6	10124	2
7	10124	3
8	10124	5

Table name: EMPLOYEE

EMP_NUM	EMP_LNAME
10121	Rogers
10122	O'Leery
10123	Panera
10124	Johnson

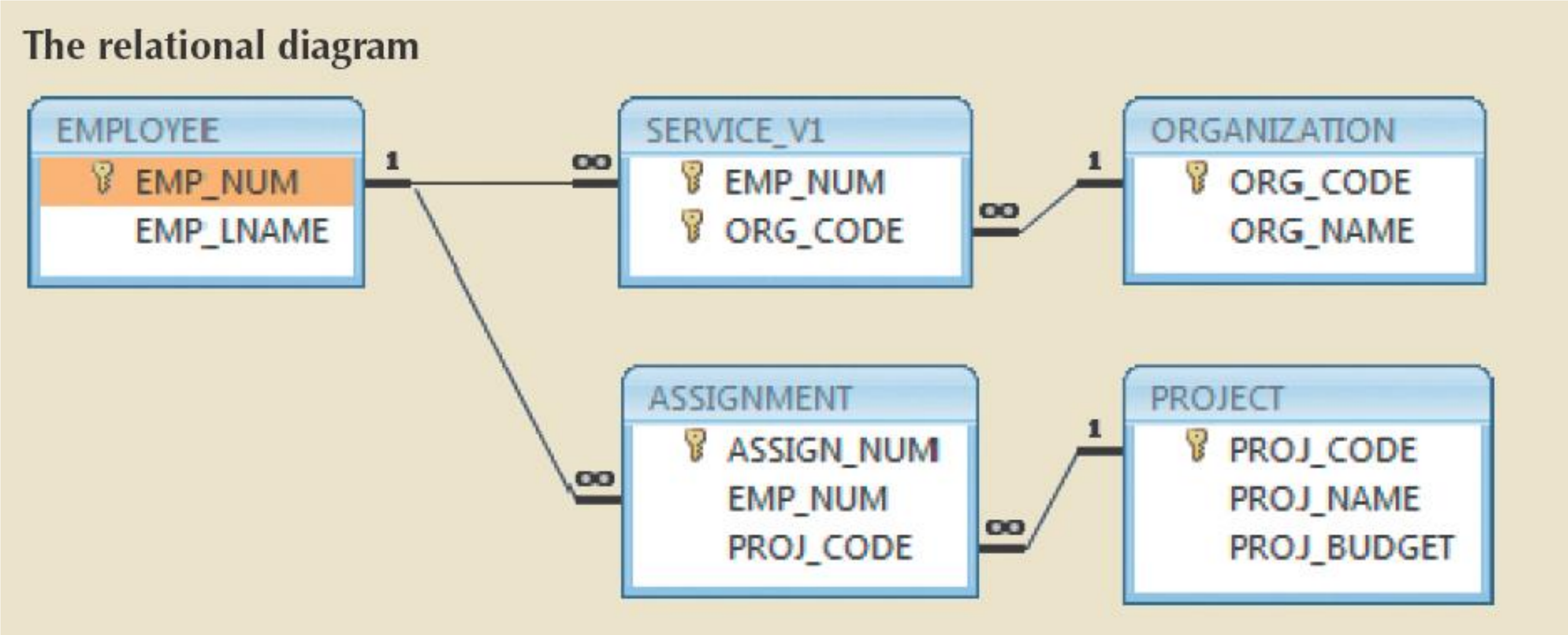
Table name: ORGANIZATION

ORG_CODE	ORG_NAME
RC	Red Cross
UW	United Way
WF	Wildlife Fund

Table name: SERVICE_V1

EMP_NUM	ORG_CODE
10123	RC
10123	UW
10123	WF

Figure 6.12 - A Set of Tables in 4NF (2 of 2)



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

Figure 6.13 - Initial Contracting Company ERD

CENGAGE

Figure 6.14 - Modified Contracting Company ERD

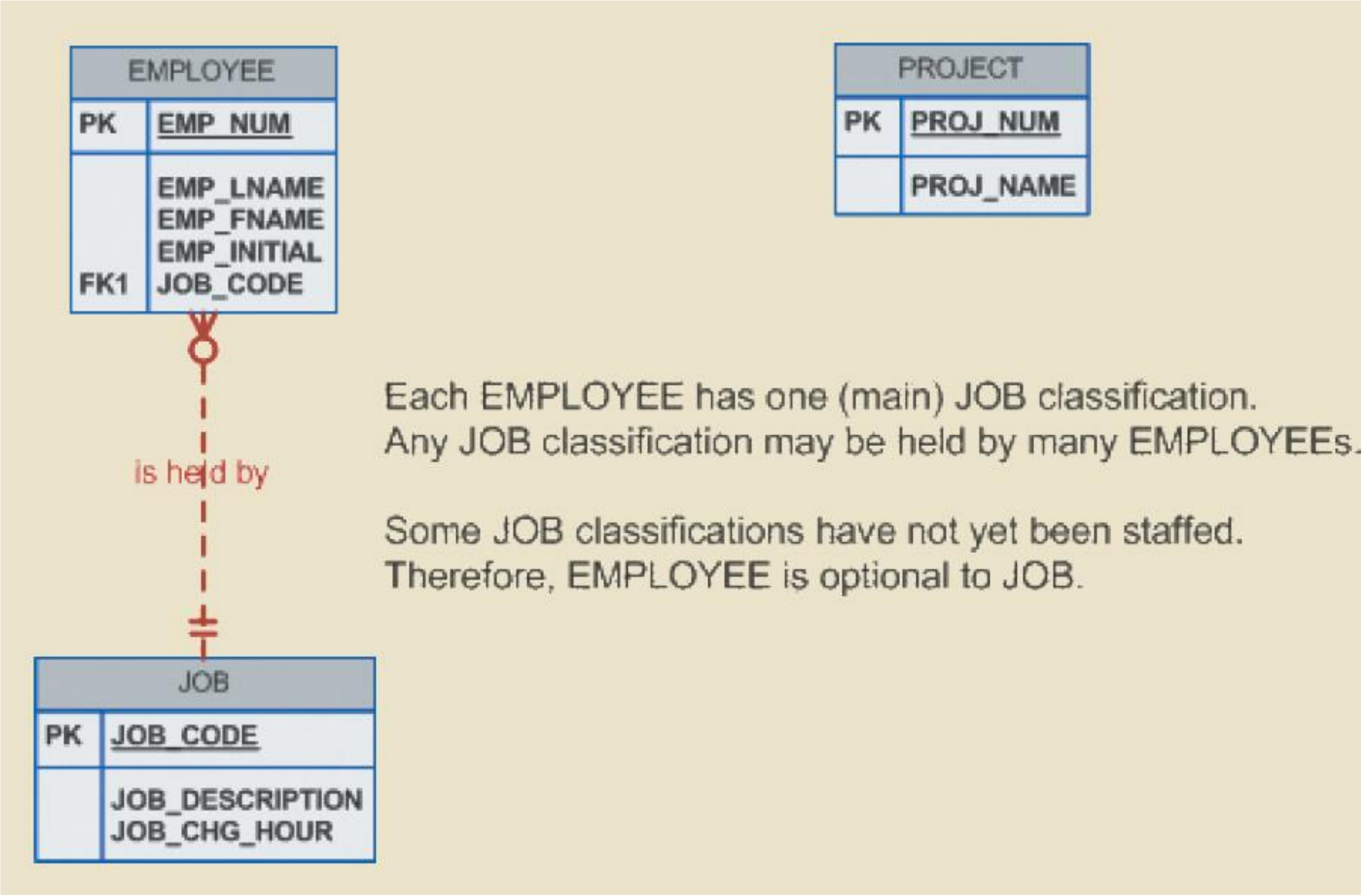


Figure 6.15 - Incorrect M:N Relationship Representation

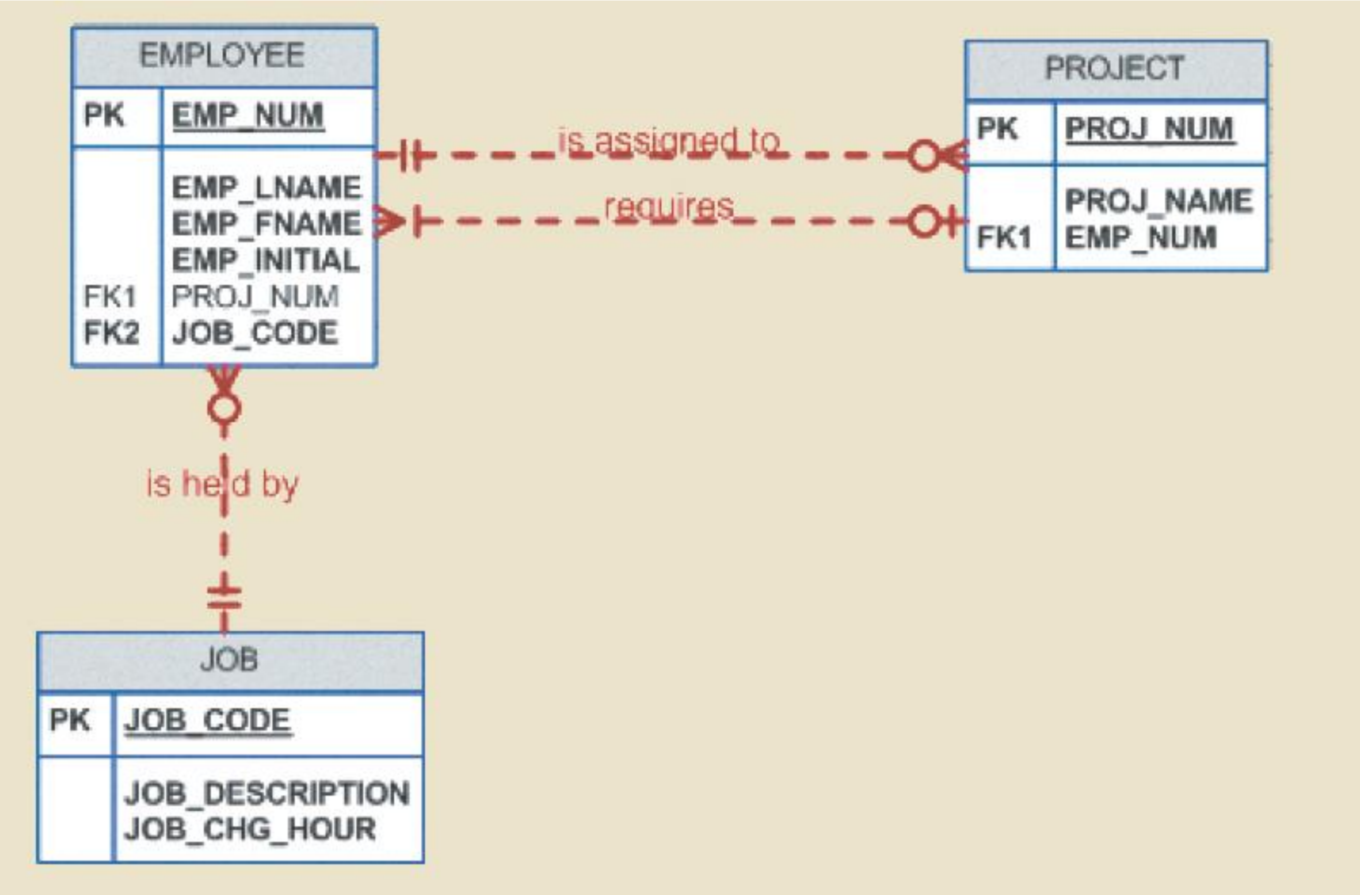


Figure 6.16 - Final Contracting Company ERD

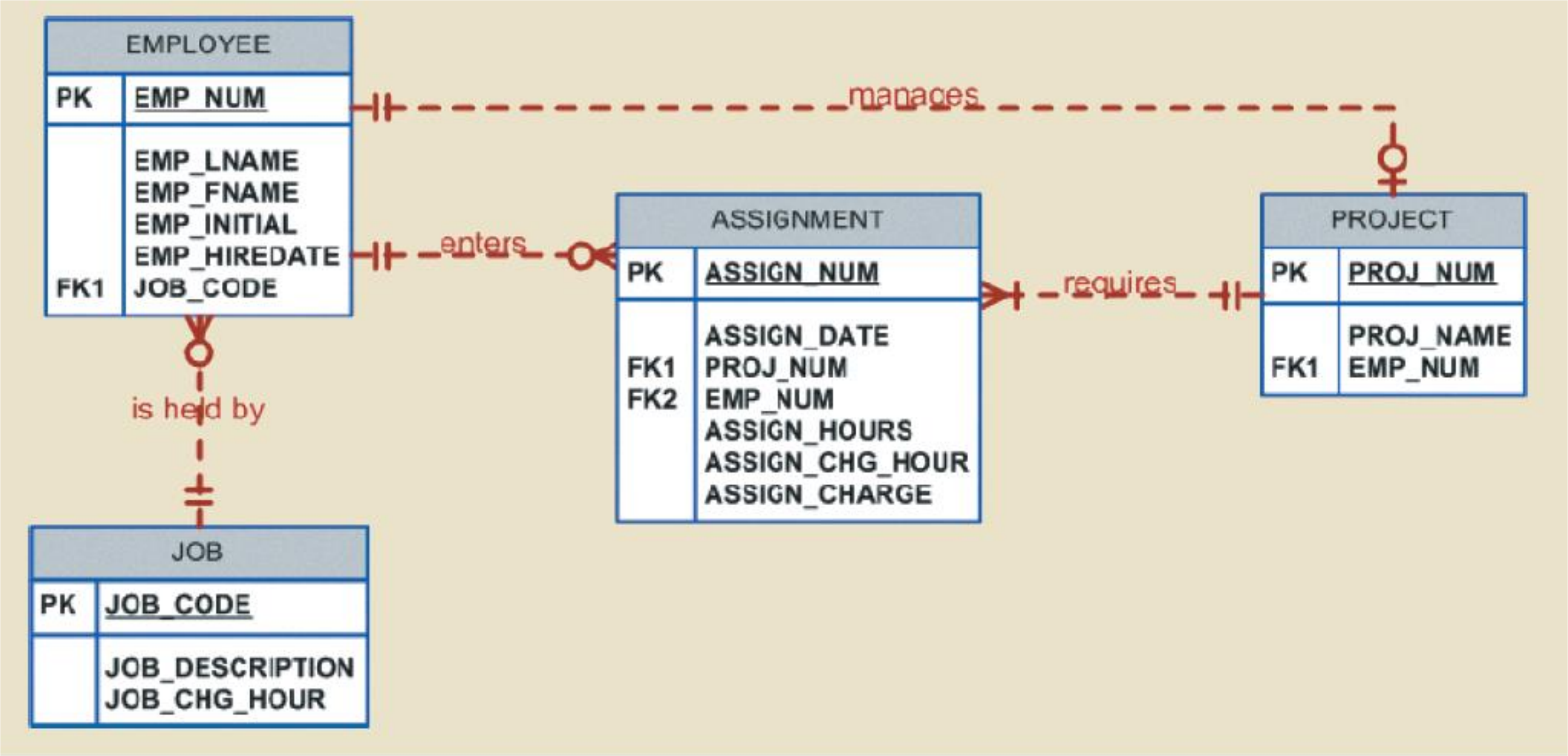


Figure 6.17 - The Implemented Database

Table name: EMPLOYEE

Database name: Ch06_ConstructCo

EMP_NUM	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_HIREDATE	JOB_CODE
101	News	John	G	08-Nov-00	502
102	Senior	David	H	12-Jul-89	501
103	Arbough	June	E	01-Dec-97	503
104	Ramoras	Anne	K	15-Nov-88	501
105	Johnson	Alice	K	01-Feb-94	502
106	Smithfield	William		22-Jun-05	500
107	Alonzo	Maria	D	10-Oct-94	500
108	Washington	Ralph	B	22-Aug-89	501
109	Smith	Larry	W	18-Jul-99	501
110	Olenko	Gerald	A	11-Dec-96	505
111	Wabash	Geoff	B	04-Apr-89	506
112	Smithson	Darlene	M	23-Oct-95	507
113	Joenbrood	Delbert	K	15-Nov-94	508
114	Jones	Annelise		20-Aug-91	508
115	Bawangi	Travis	B	25-Jan-90	501
116	Pratt	Gerald	L	05-Mar-95	510
117	Williamson	Angie	H	19-Jun-94	509
118	Frommer	James	J	04-Jan-06	510

Table name: JOB

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
500	Programmer	35.75
501	Systems Analyst	96.75
502	Database Designer	105.00
503	Electrical Engineer	84.50
504	Mechanical Engineer	67.90
505	Civil Engineer	55.78
506	Clerical Support	26.87
507	DSS Analyst	45.95
508	Applications Designer	48.10
509	Bio Technician	34.55
510	General Support	18.36

Table name: PROJECT

PROJ_NUM	PROJ_NAME	EMP_NUM
15	Evergreen	105
18	Amber Wave	104
22	Rolling Tide	113
25	Starflight	101

Table name: ASSIGNMENT

ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_HOURS	ASSIGN_CHG_HOUR	ASSIGN_CHARGE
1001	04-Mar-16	15	103	2.6	84.50	219.70
1002	04-Mar-16	18	118	1.4	18.36	25.70
1003	05-Mar-16	15	101	3.6	105.00	378.00
1004	05-Mar-16	22	113	2.5	48.10	120.25
1005	05-Mar-16	15	103	1.9	84.50	160.55
1006	05-Mar-16	25	115	4.2	96.75	406.35
1007	05-Mar-16	22	105	5.2	105.00	546.00
1008	05-Mar-16	25	101	1.7	105.00	178.50
1009	05-Mar-16	15	105	2.0	105.00	210.00
1010	06-Mar-16	15	102	3.8	96.75	367.65
1011	06-Mar-16	22	104	2.6	96.75	251.55
1012	06-Mar-16	15	101	2.3	105.00	241.50
1013	06-Mar-16	25	114	1.8	48.10	86.58
1014	06-Mar-16	22	111	4.0	26.87	107.48
1015	06-Mar-16	25	114	3.4	48.10	163.54
1016	06-Mar-16	18	112	1.2	45.95	55.14
1017	06-Mar-16	18	118	2.0	18.36	36.72
1018	06-Mar-16	18	104	2.6	96.75	251.55
1019	06-Mar-16	15	103	3.0	84.50	253.50
1020	07-Mar-16	22	105	2.7	105.00	283.50
1021	08-Mar-16	25	108	4.2	96.75	406.35
1022	07-Mar-16	25	114	5.8	48.10	278.98
1023	07-Mar-16	22	106	2.4	35.75	85.80

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

Table 6.6 – Common Denormalization Examples

CASE	EXAMPLE	RATIONALE AND CONTROLS
Redundant data	Storing ZIP and CITY attributes in the AGENT table when ZIP determines CITY (see Figure 2.2)	Avoid extra join operations Program can validate city (drop-down box) based on the zip code
Derived data	Storing STU_HRS and STU_CLASS (student classification) when STU_HRS determines STU_CLASS (see Figure 3.28)	Avoid extra join operations Program can validate classification (lookup) based on the student hours
Preaggregated data (also derived data)	Storing the student grade point average (STU_GPA) aggregate value in the STUDENT table when this can be calculated from the ENROLL and COURSE tables (see Figure 3.28)	Avoid extra join operations Program computes the GPA every time a grade is entered or updated STU_GPA can be updated only via administrative routine
Information requirements	Using a temporary denormalized table to hold report data; this is required when creating a tabular report in which the columns represent data that are stored in the table as rows (see Figures 6.17 and 6.18)	Impossible to generate the data required by the report using plain SQL No need to maintain table Temporary table is deleted once report is done Processing speed is not an issue

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

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.