The National Higher School of Artificial Intelligence

DATABASES

Chapter 4 : Normalization of Database Tables

Pr. Kamel BOUKHALFA

Slides From the Textbook :
Carlos Coronel and Steven Morris, Database Systems: Design, Implementation, and Management
Tenth Edition

Objectives

- In this chapter, students 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

Database Tables and Normalization

- Normalization
  - Process for evaluating and correcting table structures to minimize data redundancies
    - Reduces data anomalies
  - Series of stages called normal forms:
    - First normal form (1NF)
    - Second normal form (2NF)
    - Third normal form (3NF)

Database Tables and Normalization

- Normalization (continued)
  - 2NF is better than 1NF; 3NF is better than 2NF
  - For most business database design purposes, 3NF is as high as needed in normalization
  - Highest level of normalization is not always most desirable
- Denormalization produces a lower normal form
  - Increased performance but greater data redundancy

The Need for Normalization

- Example: company that manages building projects (Figure 6.1)
  - Each project has its own project number, name, assigned employees, etc.
  - Each employee has an employee number, name, job class
  - Charges its clients by billing hours spent on each contract
  - Hourly billing rate is dependent on employee's position
  - Total charge is a derived attribute and not stored in the table
  - Periodically, report is generated that contains information such as displayed in Table 6.1

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

SOURCE: Course Technology/Cengage Learning

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.

The Need for Normalization

- Structure of data set in Figure 6.1 does not handle data very well
- Table structure appears to work; report is generated with ease
- Report may yield different results depending on what data anomaly has occurred
  - Employee can be assigned to more than one project but each project includes only a single occurrence of any one employee
- Relational database environment is suited to help designer avoid data integrity problems

The Need for Normalization

- PROJECT_NUM, either a PK or part of a PK, contains NULLS
- JOB_CLASS values could be abbreviated differently
- Each time an employee is assigned to a project, all employee information is duplicated
- Update anomalies – Modifying JOB_CLASS for employee 105 requires alterations in two records
- Insertion anomalies – to insert a new employee who has not been assigned to a project requires a phantom project
- Deletion anomalies – If a project has only one employee associated with it and that employee leaves, a phantom employee must be created

The Normalization Process

- 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

TABLE 6.2 Normal Forms

NORMAL FORM	CHARACTERISTIC	SECTION
First normal form (1NF)	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

The Normalization Process (cont'd.)

- Objective of normalization is to ensure that all tables are in at least 3NF
- Higher forms are not likely to be encountered in business environment
- Normalization works one relation at a time
- Progressively breaks table into new set of relations based on identified dependencies

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 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*.

The Normalization Process (cont'd.)

- Partial dependency
  - Exists when there is a functional dependence in which the determinant is only part of the primary key
  - If (A,B)□(C,D); B□C and (A,B) is the PK
    - B□C is a partial dependency because only part of the PK, B, is needed to determine the value of C
- Transitive dependency
  - Exists when there are functional dependencies such that X → Y, Y → Z, and X is the primary key
    - X□Z is a transitive dependency because X determines the value of Z via Y
    - The existence of a functional dependence among non-prime attributes is a sign of transitive dependency

Conversion to First Normal Form

- Repeating group
  - Group of multiple entries of same type can exist for any single key attribute occurrence
- Relational table must not contain repeating groups
- Normalizing table structure will reduce data redundancies
- Normalization is three-step procedure

Conversion to First Normal Form (cont'd.)

- Step 1: Eliminate the Repeating Groups
  - Eliminate nulls: each repeating group attribute contains an appropriate data value
- Step 2: Identify the Primary Key
  - Must uniquely identify attribute value
  - New key must be composed
- Step 3: Identify All Dependencies
  - Dependencies are depicted with a diagram

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

SOURCE: Course Technology/Cengage Learning

Conversion to First Normal Form (cont'd.)

- Dependency diagram:
  - Depicts all dependencies found within given table structure
  - Helpful in getting bird's-eye view of all relationships among table's attributes
  - Makes it less likely that you will overlook an important dependency
  - The arrows above the attributes indicate desirable dependencies (i.e., based on the PK)
  - The arrows below the attributes indicate less desirable dependencies (partial and transitive)

Conversion to First Normal Form

- First normal form describes tabular format:
  - All key attributes are defined
  - 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 are based on part of the primary key
  - Should be used with caution

Conversion to Second Normal Form

- Conversion to 2NF occurs only when the 1NF has a composite key
  - If the 1NF key is a single attribute, then the table is automatically in 2NF
- Step 1: Make New Tables to Eliminate Partial Dependencies
  - For each component of the PK that acts as a determinant in a partial dependency, create a new table with a copy of that component as the PK
  - These components also remain in the original table in order to serve as FKs to the original table
  - Write each key component on a separate line; then write the original composite key on the last line. Each component will become the key in a new table

    PROJ_NUM

    EMP_NUM

    PROJ_NUM EMP_NUM

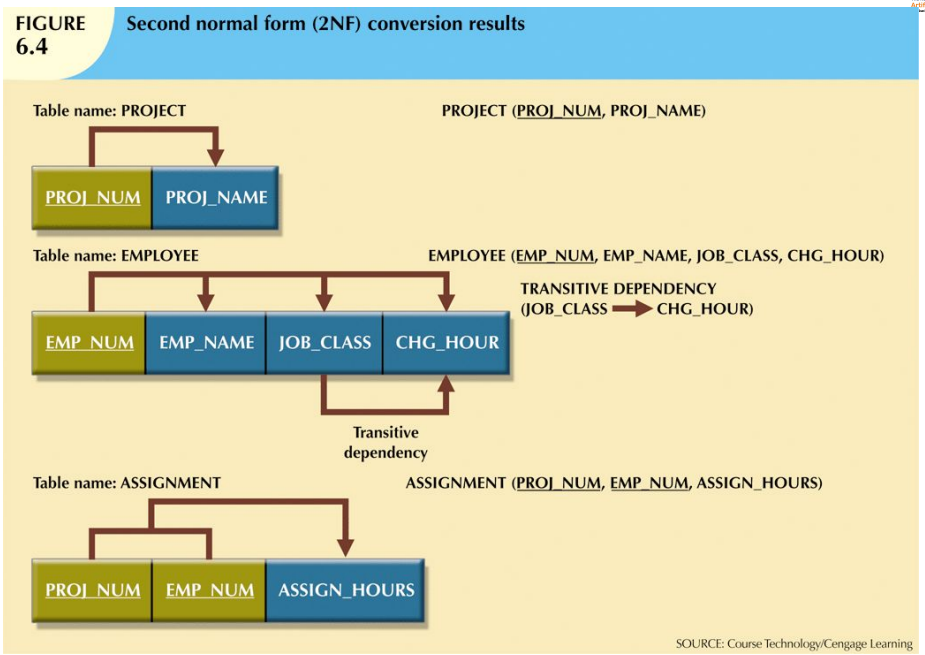
Conversion to Second Normal Form

- Step 2: Reassign Corresponding Dependent Attributes
  - The dependencies for the original key components are found by examining the arrows below the dependency diagram in Fig 6.3
  - The attributes 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 remain in the original table
  - At this point, most anomalies have been eliminated

    PROJECT(**PROJ_NUM**, PROJ_NAME)

    EMPLOYEE(**EMP_NUM**, EMP_NAME, JOB_CLASS, CHG_HOUR)

    ASSIGNMENT(**PROJ_NUM** , **EMP_NUM**, ASSIGN_HOURS)



Conversion to Second Normal Form

- Table is in second normal form (2NF) when:
  - It is in 1NF and
  - It includes no partial dependencies:
    - No attribute is dependent on only portion of primary key

Conversion to Third Normal Form

- Step 1: Make New Tables to Eliminate Transitive Dependencies
  - For every transitive dependency, write its determinant as PK for new table (JOB_CLASS)
    - Determinant: any attribute whose value determines other values within a row
  - The determinant should remain in the original table to serve as a FK

Conversion to Third Normal Form

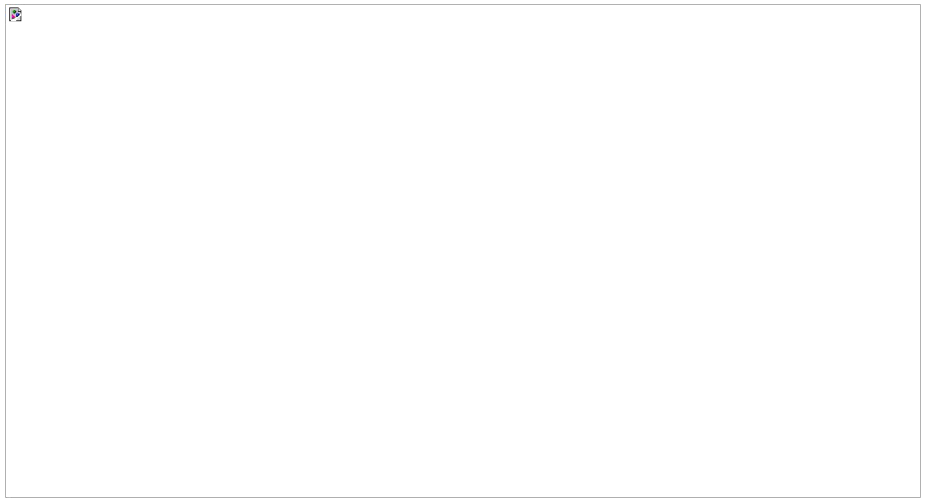
- Step 2: Reassign Corresponding Dependent Attributes
  - Identify attributes dependent on each determinant identified in Step 1
    - Identify dependency
  - Name table to reflect its contents and function

PROJECT(**PROJ_NUM**, PROJ_NAME)

ASSIGNMENT(**PROJ_NUM** , **EMP_NUM**, ASSIGN_HOURS)

EMPLOYEE(**EMP_NUM**, EMP_NAME, JOB_CLASS)

JOB(**JOB_CLASS**, CHG_HOUR)



Conversion to Third Normal Form

- A table is in third normal form (3NF) when both of the following are true:
  - It is in 2NF
  - It contains no transitive dependencies

Conversion to Third Normal Form

1NF->2NF – remove partial dependencies

2NF->3NF – remove transitive dependencies

- In both cases, the answer is create a new table
  - The determinant of the problem dependency remains in the original table and is placed as the PK of the new table
  - The dependents of the problem dependency are removed from the original table and placed as nonprime attributes in the new table

Improving the Design

- Table structures should be cleaned up to eliminate initial partial and transitive dependencies
- Normalization cannot, by itself, be relied on to make good designs
- Valuable because it helps eliminate data redundancies
- If a table has multiple candidate keys and one is a composite key, there can be partial dependencies even when the PK is a single attribute
  - Resolve in 3NF as a transitive dependency

Improving the Design (cont'd.)

- Issues to address, in order, to produce a good normalized set of tables:
  - Evaluate PK Assignments
    - Use JOB_CODE as PK for JOB table rather than JOB_CLASS to avoid data-entry errors when used as a FK in EMPLOYEE (DB Designer /Database Designer)
    - JOB (**JOB_CODE**, JOB_CLASS,CHG_HOUR)
    - Why is JOB_CLASS-->CHG_HOUR not a transitive dependency? (Because JOB_CLASS is a candidate key)

Improving the Design (cont'd.)

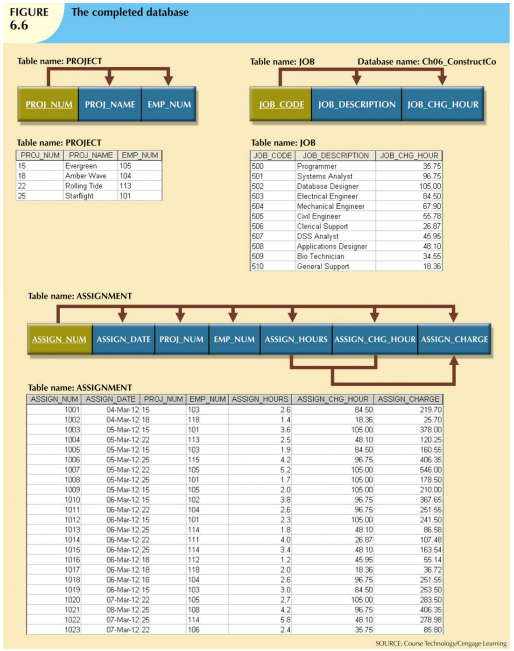
- Evaluate Naming Conventions
  - CHG_HOUR should be JOB_CHG_HOUR
  - JOB_DESCRIPTION is a better than JOB_CLASS
- Refine Attribute Atomicity
  - Atomic attribute – one that can not be further subdivided
    - EMP_NAME is not atomic
- Identify New Attributes
  - YTD gross salary, social security payments, hire date

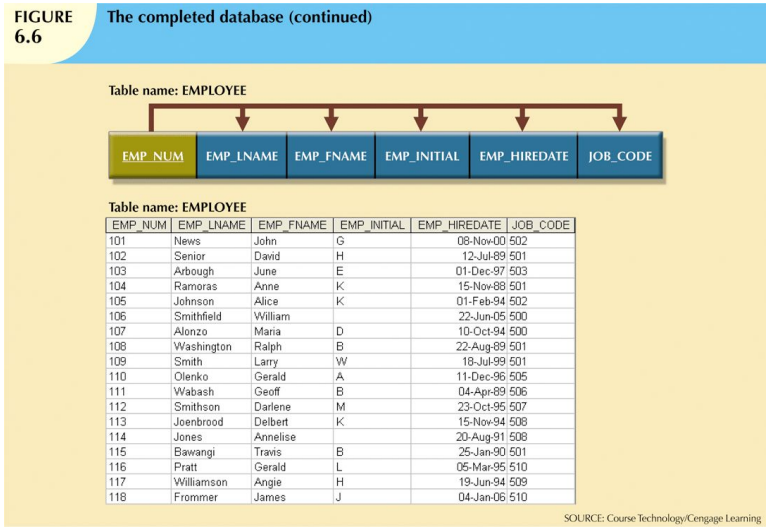
Improving the Design (cont'd.)

- Identify New Relationships
  - To track the manager of each project, put EMP_NUM as a FK in PROJECT
- Refine Primary Keys as Required for Data Granularity
  - What does ASSIGN_HOURS represent ? Yearly total hours, weekly, daily?
  - If need multiple daily entries for project and emp number, then use a surrogate key ASSIGN_NUM to avoid duplication of the PK key EMP_NUM, PROJ_NUM, ASSIGN_DATE

Improving the Design (cont'd.)

- Maintain Historical Accuracy
  - An employee's job charge could change over the lifetime of a project. In order to reconstruct the charges to a project, another field with the job charge and date active is required
- Evaluate Using Derived Attributes
  - Store rather than derive the charge if it will speed up reporting





Higher-Level Normal Forms

- Tables in 3NF perform suitably in business transactional databases
- Higher-order normal forms are useful on occasion
- Two special cases of 3NF:
  - Boyce-Codd normal form (BCNF)
  - Fourth normal form (4NF)

The Boyce-Codd Normal Form

- Every determinant in table is a candidate key
  - Has same characteristics as primary key, but for some reason, not chosen to be primary key
- When table contains only one candidate key, the 3NF and the BCNF are equivalent
- BCNF can be violated only when table contains more than one candidate key

The Boyce-Codd Normal Form

- Most designers consider the BCNF as a special case of 3NF
- Table is in 3NF when it is in 2NF and there are no transitive dependencies
- Table can be in 3NF and fail to meet BCNF
  - No partial dependencies, nor does it contain transitive dependencies
  - A nonkey attribute is the determinant of a key attribute

FIGURE 6.7 A table that is in 3NF but not in BCNF

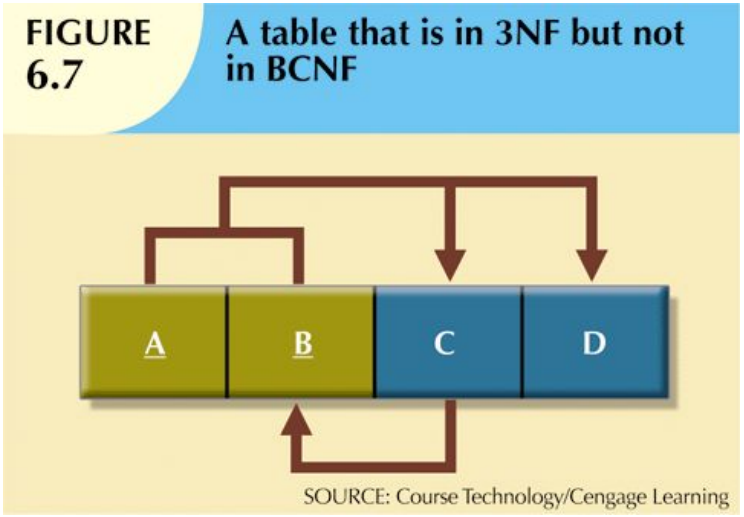
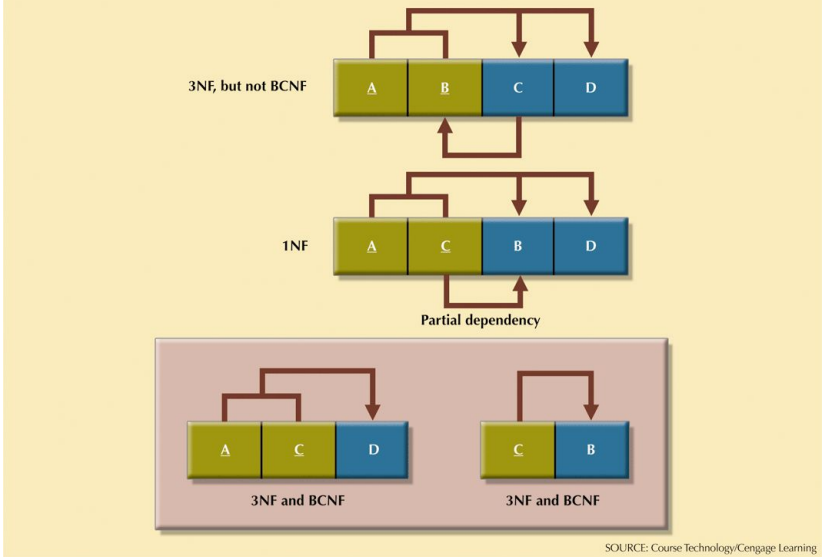


TABLE 6.5 Sample Data for a BCNF Conversion

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	32456	B

FIGURE 6.8 Decomposition to BCNF



Normalization and Database Design

- Normalization should be part of the design process
- Make sure that proposed entities meet required normal form before table structures are created
- Many real-world databases have been improperly designed or burdened with anomalies
- You may be asked to redesign and modify existing databases

Normalization and Database Design

- ER diagram
  - Identify relevant entities, their attributes, and their relationships
  - Identify additional entities and attributes
- Normalization procedures
  - Focus on characteristics of specific entities
  - Micro view of entities within ER diagram
- Difficult to separate normalization process from ER modeling process

Normalization and Database Design

- Given the following business rules:
  - The company manages many projects
  - Each project requires the services of many employees
  - An employee may be assigned to several projects
  - Some employees are not assigned to a project and perform non-project related duties. Some employees are part of a labor pool and shared by all project teams
  - Each employee has a single primary job classification which determines the hourly billing rate]
  - Many employees can have the same job classification.

Normalization and Database Design

- We initially define the following entities

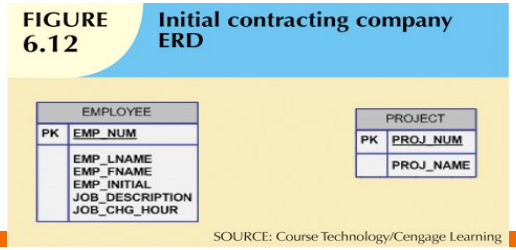
  PROJECT(**PROJ_NUM**, PROJ_NAME)

  EMPLOYEE(**EMP_NUM**,EMP_LNAME, EMP_FNAME, EMP_INITIAL, JOB_DESCRIPTION, JOB_CHG_HOUR)
- PROJECT is in 3NF and needs no modification
- EMPLOYEE contains a transitive dependency so we now have

  PROJECT(**PROJ_NUM**, PROJ_NAME)

  EMPLOYEE(**EMP_NUM**,EMP_LNAME, EMP_FNAME, EMP_INITIAL, JOB_CODE)

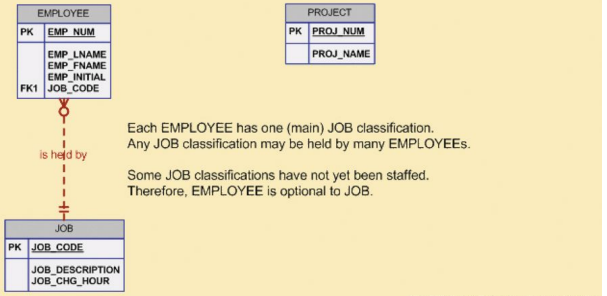
  JOB(**JOB_CODE**, JOB_DESCRIPTION, JOB_CHG_HOUR)



Normalization and Database Design

- EMPLOYEE contains a transitive dependency so we now have
  PROJECT(**PROJ_NUM**, PROJ_NAME)
  EMPLOYEE(**EMP_NUM**,EMP_LNAME, EMP_FNAME, EMP_INITIAL, JOB_CODE)
  JOB(**JOB_CODE**, JOB_DESCRIPTION, JOB_CHG_HOUR)

FIGURE 6.13 Modified contracting company ERD



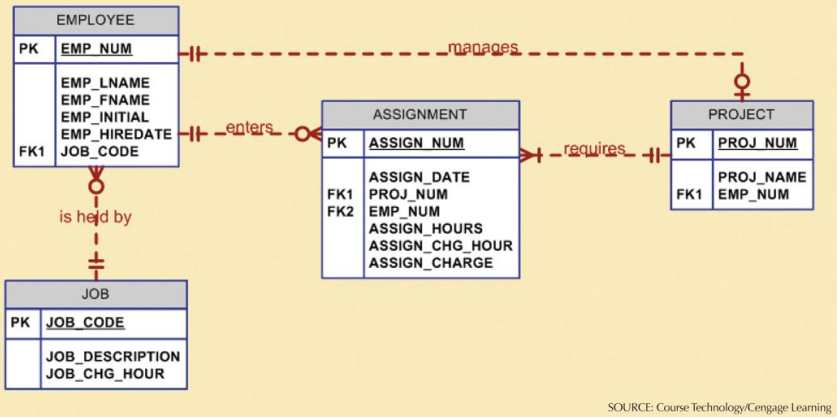
Normalization and Database Design

- To represent the M:N relationship between EMPLOYEE and PROJECT, we could try two 1:M realtionships
  - An employee can be assigned to many projects
  - Each project can have many employees assigned to it

Normalization and Database Design

- As this M:N can not be implemented, we include the ASSIGNMENT entity to track the assignment of employees in projects

FIGURE 6.15 Final contracting company ERD



Normalization and Database Design

- ASSIGN_HOURS is assigned to ASSIGNMENT
- A "manages" relationship is added to in order to keep detailed information about each project's manager
- Some additional attributes are added to maintain additional information

PROJECT(**PROJ_NUM**, PROJ_NAME,EMP_NUM)

EMPLOYEE(**EMP_NUM**,EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_HIREDATE, JOB_CODE)

JOB(**JOB_CODE**, JOB_DESCRIPTION, JOB_CHG_HOUR)

ASSIGNMENT(**ASSIGN_NUM**, ASSIGN_DATE, PROJ_NUM, EMP_NUM, ASSIGN_HOURS, ASSIGN_CHG_HOUR, ASSIGN_CHARGE)