DATABASE MANAGEMENT SYSTEM - I

UNIT - III

Normalization

INTRODUCTION

0301203

DATABASE MANAGEMENT SYSTEM - I

0301206

PRACTICAL ON DBMS - I

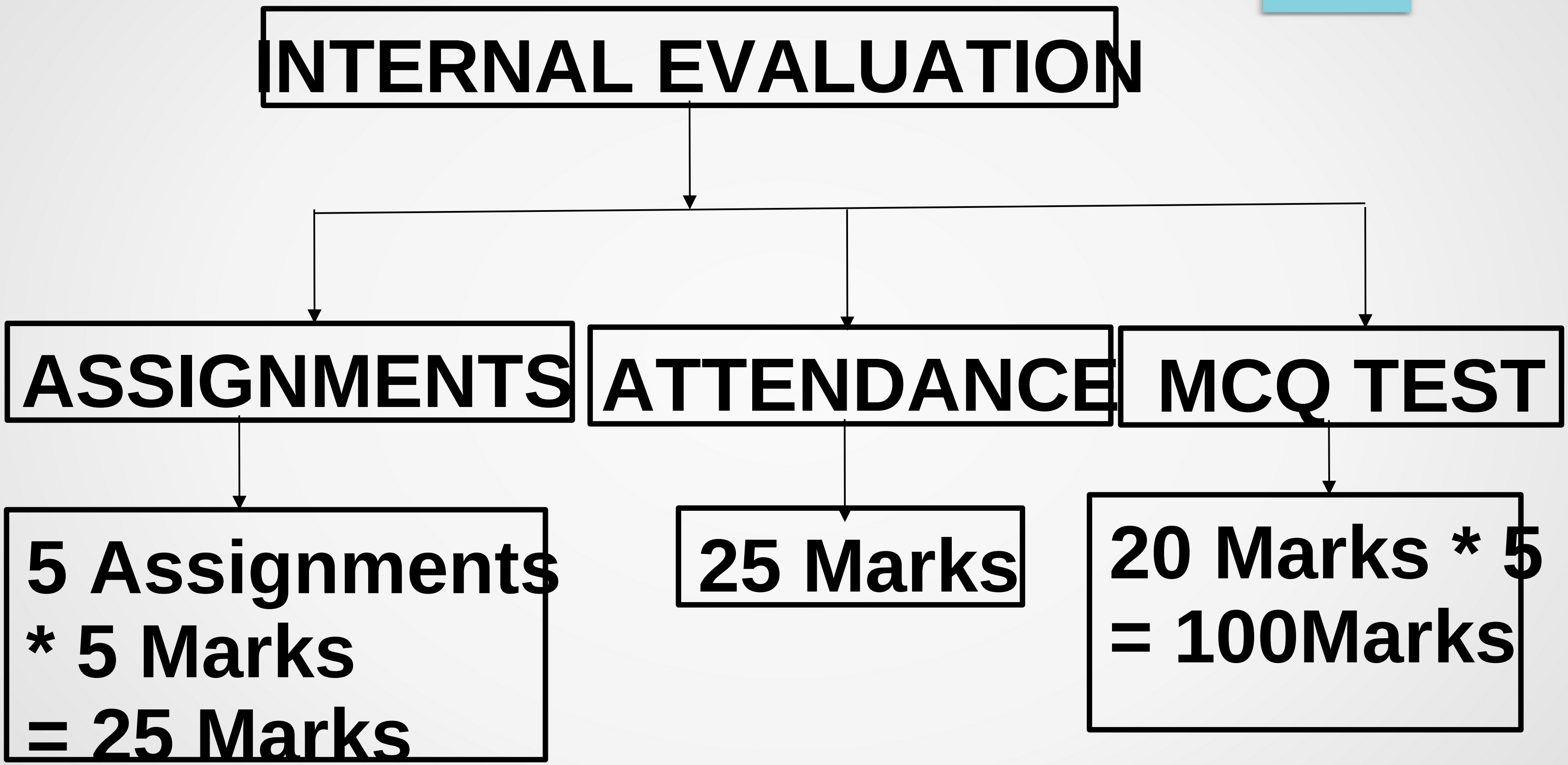
BY:

Prof. (Dr.) Ankit Bhavsar

0301203 Database Management System - I

UNIT	MODULES	WEIGHTAGE
1	Introduction to DBMS	20 %
2	Introduction to RDBMS	20 %
3	Inroduction to Normalization	20 %
4	Open Source Database Management Software	20 %
5	Introduction to MySQL	20 %

INTERNAL EVALUATION



INTERNAL EVALUATION

ASSIGNMENTS

ATTENDANCE

MCQ TEST

5 Assignments * 5 Marks = 25 Marks

25 Marks

20 Marks * 5 = 100Marks

UNIT - 3 Introduction to Normaization

- Need of Normalization
- Normalization Process
- Functional Dependency
- Normalization Forms
- Normalization Conversion
- Create Dependency Diagram

Database Tables and Normalization

Normalization

•Process for **evaluating and correcting table structures to minimize data redundancies.**

•**Reduces data anomalies.**

•It is a **process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.**

- Series of stages called normal forms:
  - **First normal form (1NF)**
  - **Second normal form (2NF)**
  - **Third normal form (3NF)**

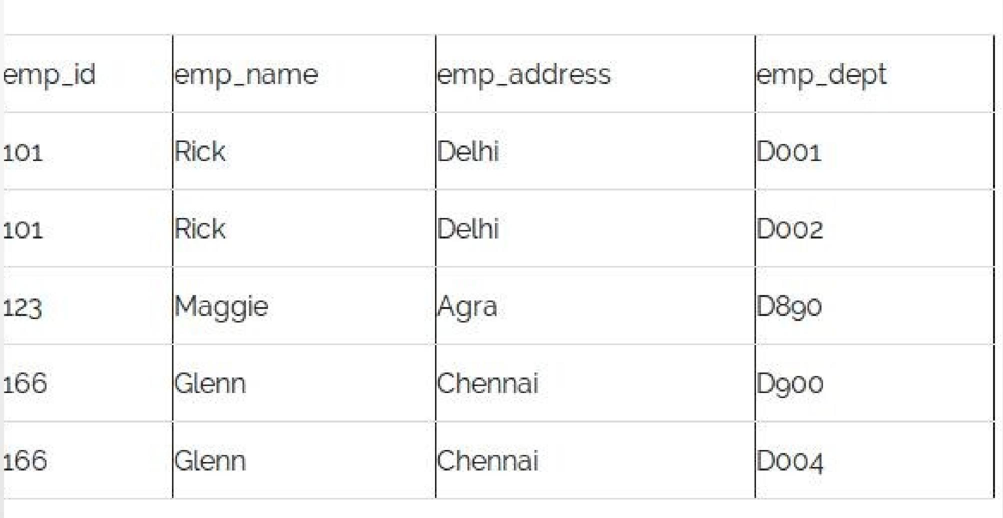
Database Tables and Normalization (cont’d.)

- **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
- **De-normalization produces a lower normal form; that is 3NF is converted to a 2NF through De-normalization.**
  - Increased performance but greater data redundancy

The Need for Normalization

Example

Suppose a manufacturing company stores the employee details in a t**able named employee** that has **four attributes**: **emp_id** for storing employee’s id, **emp_name** for storing employee’s name, **emp_address** for storing employee’s address and **emp_dep**t for storing the department details in which the employee works. At some point of time the table looks like this:



emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

Database Tables and Normalization (cont'd.)

Update anomaly: In the above table we have **two rows for employee Rick** as he belongs to **two departments of the company.** If we want to **update the address** of Rick then we have to update the same **in two rows or the data will become inconsistent.** If somehow, the correct address gets updated in one department but not in other then as per the database, Rick would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly: Suppose a new employee joins the company, **who is under training and currently not assigned to any department then we would not be able to insert the data** into the table if emp_dept field doesn't allow nulls.

Delete anomaly: Suppose, **if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890** would also delete the information of employee Maggie since she is assigned only to this department.

To overcome these anomalies we need to normalize the data.

The Normalization Process

The objective of **normalization is to ensure that each table conforms to the concept of well-formed** relations (tables) with following characteristics:

- **Each table represents a single subject**
- **No data item will be unnecessarily stored in more than one table**
- **All non-prime attributes in a table are dependent on the primary key**
- Each table is void of insertion, update, deletion anomalies

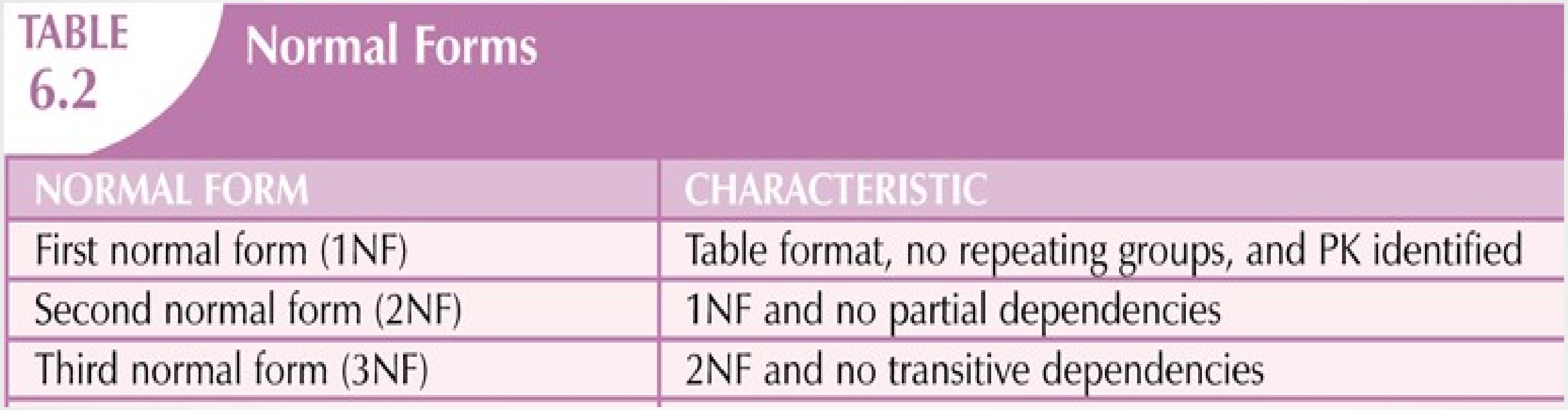


TABLE 6.2 Normal Forms

NORMAL FORM	CHARACTERISTIC
First normal form (1NF)	Table format, no repeating groups, and PK identified
Second normal form (2NF)	1NF and no partial dependencies
Third normal form (3NF)	2NF and no transitive dependencies

Functional Dependence Concepts

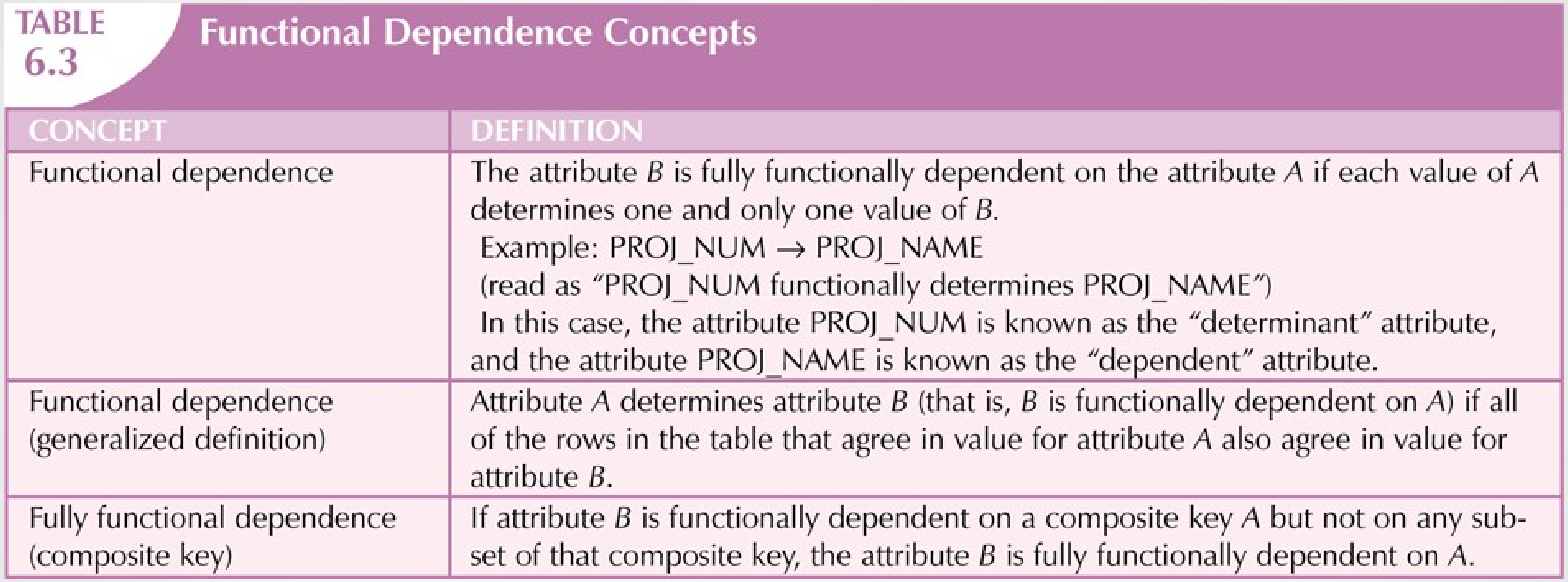


TABLE 6.3 Functional Dependence Concepts

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 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

Two types of functional dependencies:

- Partial dependency
- Transitive dependency

A **partial dependency** exists when there is functional dependence in which the determinant is only part of the primary key.

For example, if (A,B) → (C,D), B→C and (A,B) is a primary key, then the functional dependence B→ C is a partial dependency.

A **transitive dependency** exists when there are functional dependencies such that X→Y, Y→Z, and X is the primary key. Then the dependency X→Z is a transitive dependency because X determines the value of Z via Y.

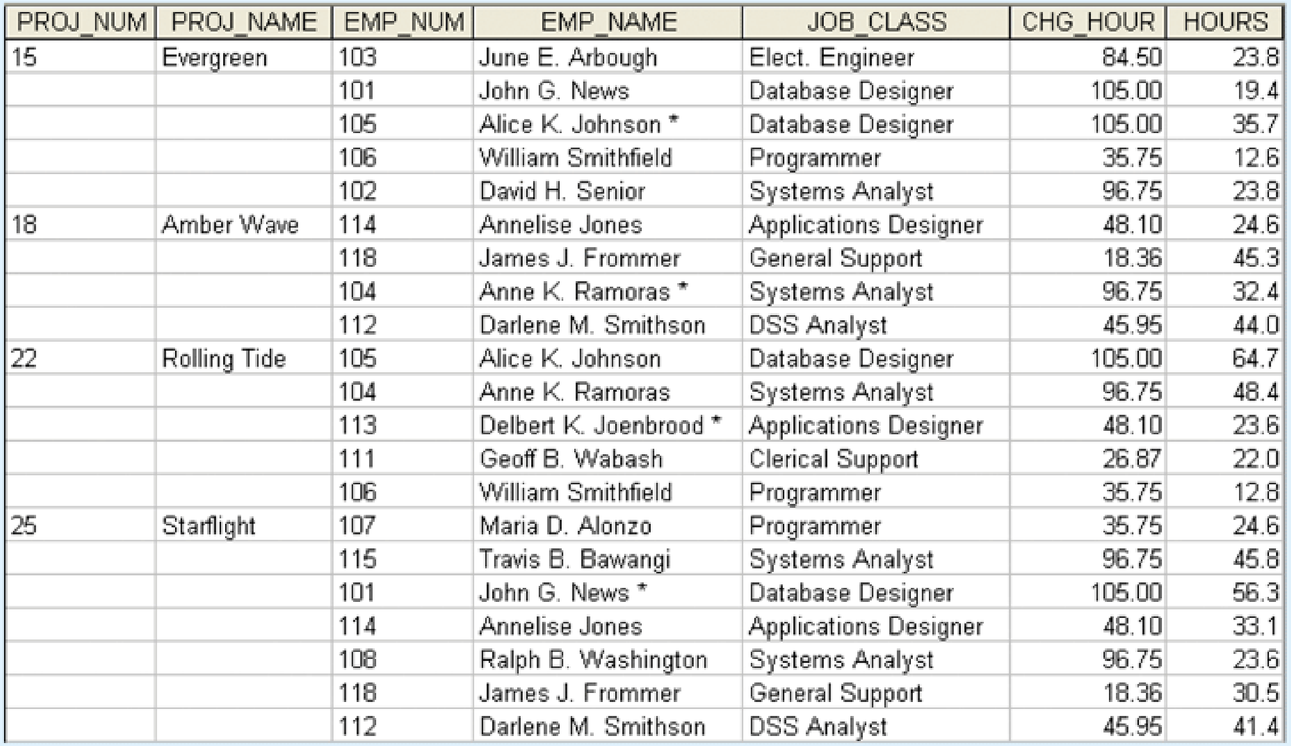
Conversion to First Normal Form

- **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

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

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

Step 1: Eliminate the Repeating Groups (eliminate nulls)

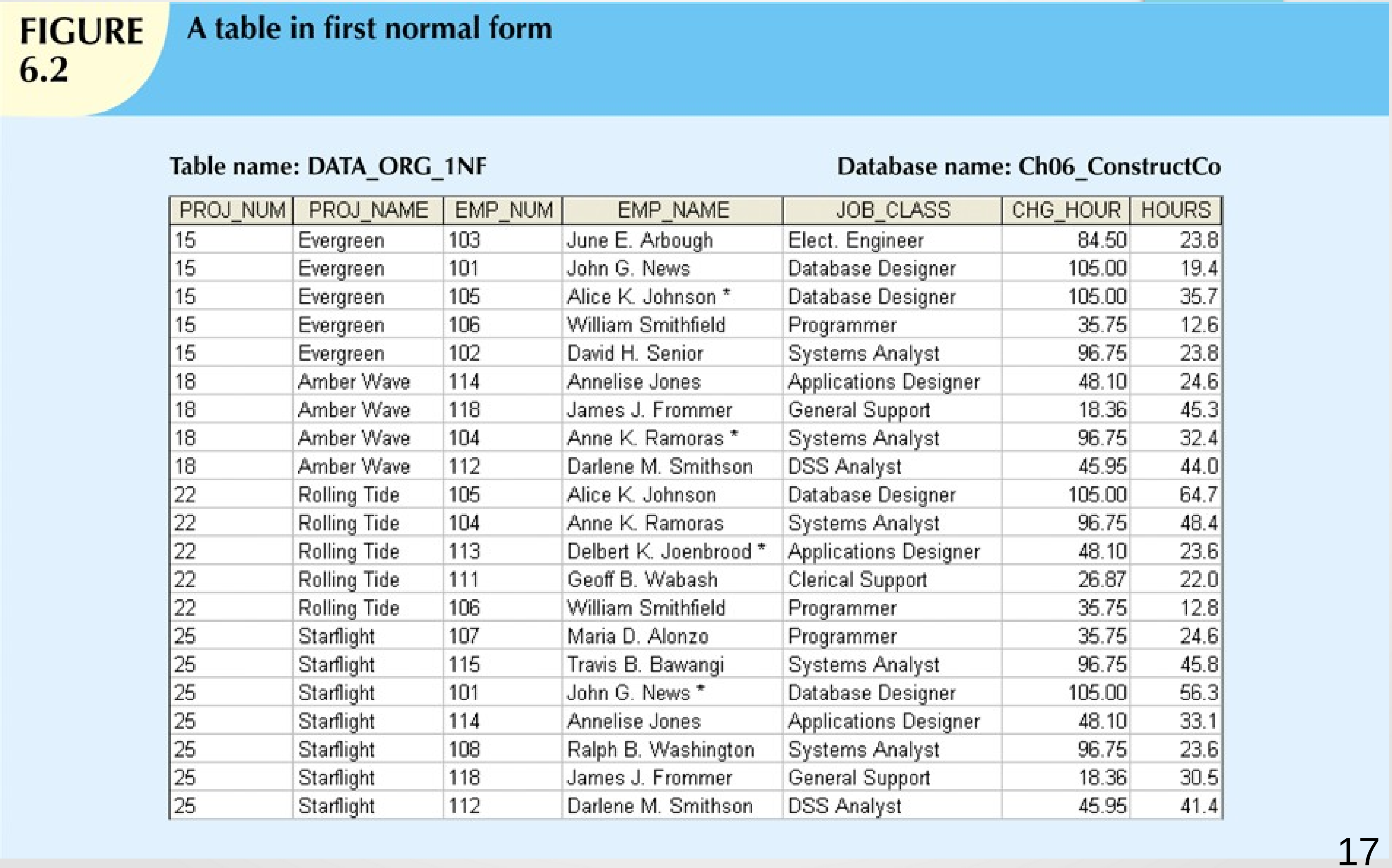


FIGURE 6.2 A table in first normal form

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

Step 2: Identify the Primary Key

- Here, let us take Proj_num as primary key. But if we look at the data, Proj_num is not unique.
- So, to make a proper primary key, a new key must be composed of a combination of Proj_num and Emp_num.

Step 3: Identify All Dependencies

Attributes:

proj_num, emp_num, proj_name, emp_name, job_class, chg_hour, hours

proj_num, emp_num → proj_name, emp_name, job_class, chg_hour, hours

proj_num → proj_name

emp_num → emp_name, job_class, chg_hours

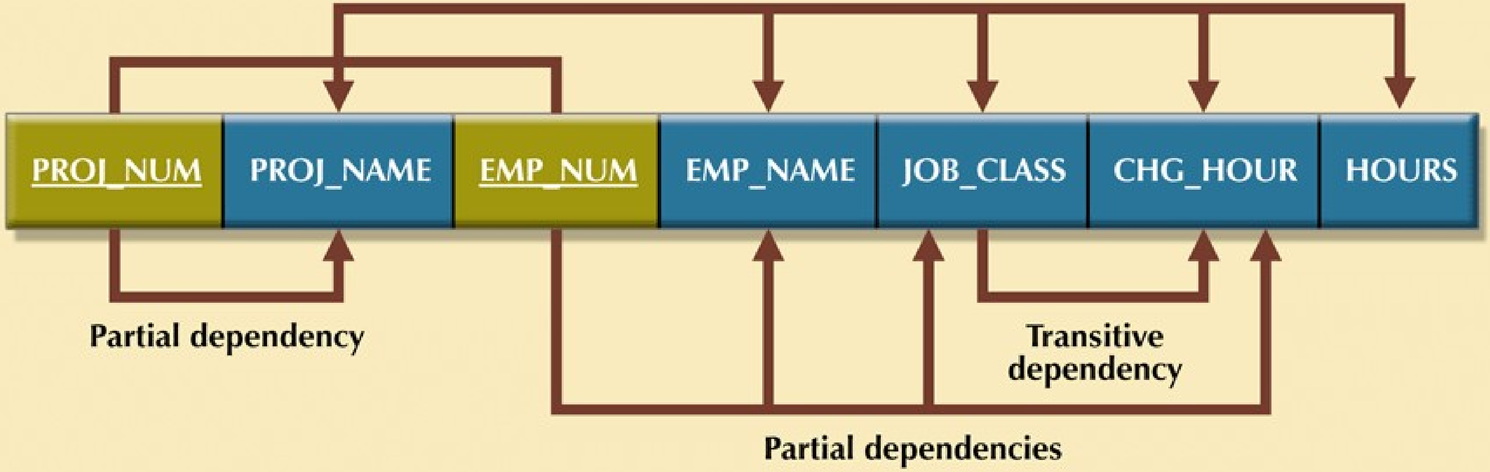
job_class → chg_hours

Now, draw the dependency diagram.

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

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 (cont'd.)

- 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
- Problem of 1NF is some tables contain partial dependencies
  - Dependencies are based on part of the primary key
  - Should be used with caution because can result in anomalies

Conversion to Second Normal Form

Converting to 2NF is done only when the 1NF has a composite primary key

If the 1NF has a single attribute primary key, then the table is automatically in 2NF.

Conversion of 1NF to 2NF has two steps:

Step 1: Make New Tables to Eliminate Partial Dependencies

Step 2: Assign Corresponding Dependent Attributes

Conversion to Second Normal Form

- Step 1: Make New Tables to Eliminate Partial Dependencies
  - Write each key component of primary key on separate line, then write original (composite) key on last line
  - Each component will become key in new table
  - Create a new table with a copy of that component as the primary key.

proj_num

emp_num

proj_num, emp_num

- So,now the original table is divided into three tables.

PROJECT, EMPLOYEE, ASSIGNMENT

Conversion to Second Normal Form

- Step 2: Assign Corresponding Dependent Attributes
  - Determine attributes that are dependent on other attributes
  - At this point, most anomalies have been eliminated
  - The attributes that are dependent in a partial dependency are removed from the original table and placed in the new table with its determinant.
- PROJECT (**proj_num** , proj_name)
- EMPLOYEE (**emp_num**, emp_name, job_class, chg_hour)
- ASSIGNMENT (**proj_num, emp_num**, assign_hours)

FIGURE 6.4 Second normal form (2NF) conversion results

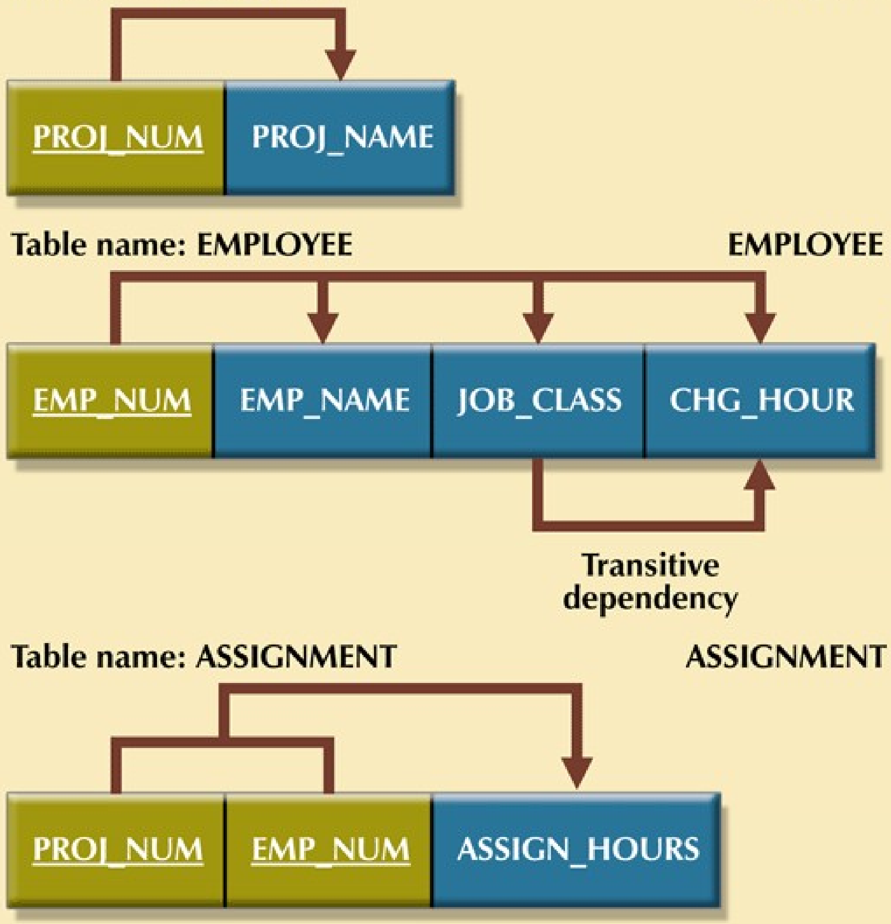


Table name: PROJECT

PROJECT (PROJ_NUM, PROJ_NAME)

Table name: EMPLOYEE

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

TRANSITIVE DEPENDENCY
(JOB_CLASS → CHG_HOUR)

Table name: ASSIGNMENT

ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

Conversion to Second Normal Form (cont’d.)

- 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
  - **Determinant:** any attribute whose value determines other values within a row.
- If you have three transitive dependencies, you will have three determinants.
- Here,

Job_class is the determinant

Conversion to Third Normal Form (cont'd.)

- Step 2: Reassign Corresponding Dependent Attributes
  - Identify attributes dependent on each determinant identified in Step 1
    - Identify dependency and place the dependent attributes in the new tables with their determinants and remove them from their original tables.

    emp_num → emp_name, job_class
  - Name table to reflect its contents and function

FIGURE 6.5 Third normal form (3NF) conversion results

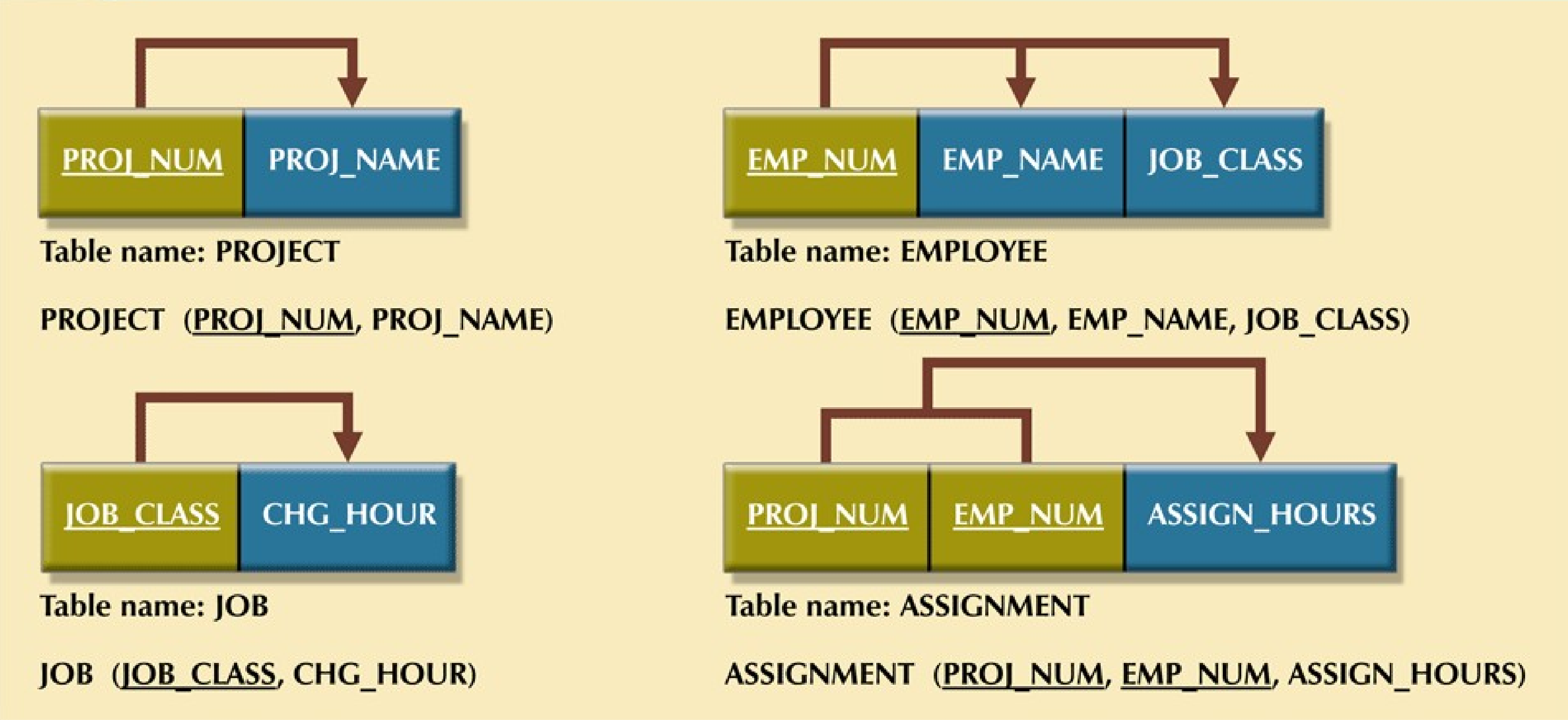


Table name: PROJECT

PROJECT (PROJ_NUM, PROJ_NAME)

Table name: EMPLOYEE

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

Table name: JOB

JOB (JOB_CLASS, CHG_HOUR)

Table name: ASSIGNMENT

ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

Conversion to Third Normal Form (cont’d.)

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