Module 2

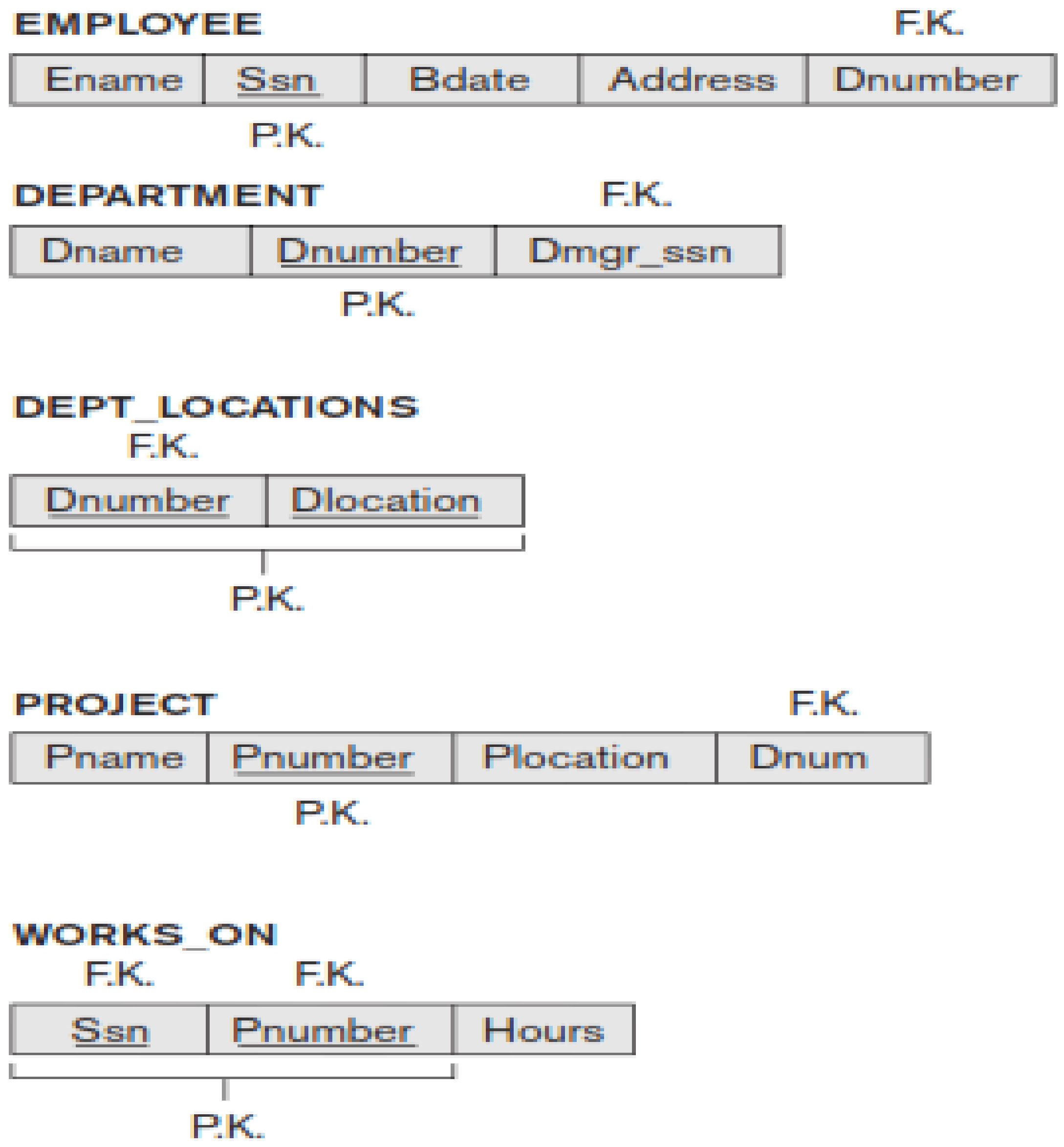
Syllabus

Database Design:- Database Tables and Normalization – The Need for Normalization – The Normalization Process: Inference Rules for Functional Dependencies (proof not needed) - Minimal set of Functional Dependencies - Conversion to First Normal Form, Conversion to Second Normal Form, Conversion to Third Normal Form - Improving the Design – Surrogate Key Considerations - Higher Level Normal Forms: Boyce/Codd Normal Form, Fourth Normal Form, Join dependencies and Fifth Normal Form – Normalization and Database Design

Informal Design Guidelines for Relation Schemas

- Guideline 1. Design a relation schema so that it is easy to explain its meaning.
- Don't combine attributes from multiple entity types and relationship types into a single relation.
- Intuitively, if a relation schema corresponds to one entity type or one relationship type, it is straightforward to explain its meaning.
- Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

Figure 14.1
A simplified COMPANY relational database schema.



Informal Design Guidelines for Relation Schemas

Guideline 2.

- Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations.
- If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

Insert Anomaly

Inserting data into database is not possible because other data is not already there

Delete Anomaly

When delete some information and lose valuable information at the same time

Update Anomaly

Any changes made to data will require to scan all records to make the changes multiple time

Stud_id	Stud_name	Stud_age	Dept_Id	Dept_name	HOD
1	Vineeth	21	3	CSE	Vince
2	Rohith	21	3	CSE	Vince
3	Sarath	22	4	IT	Jiby

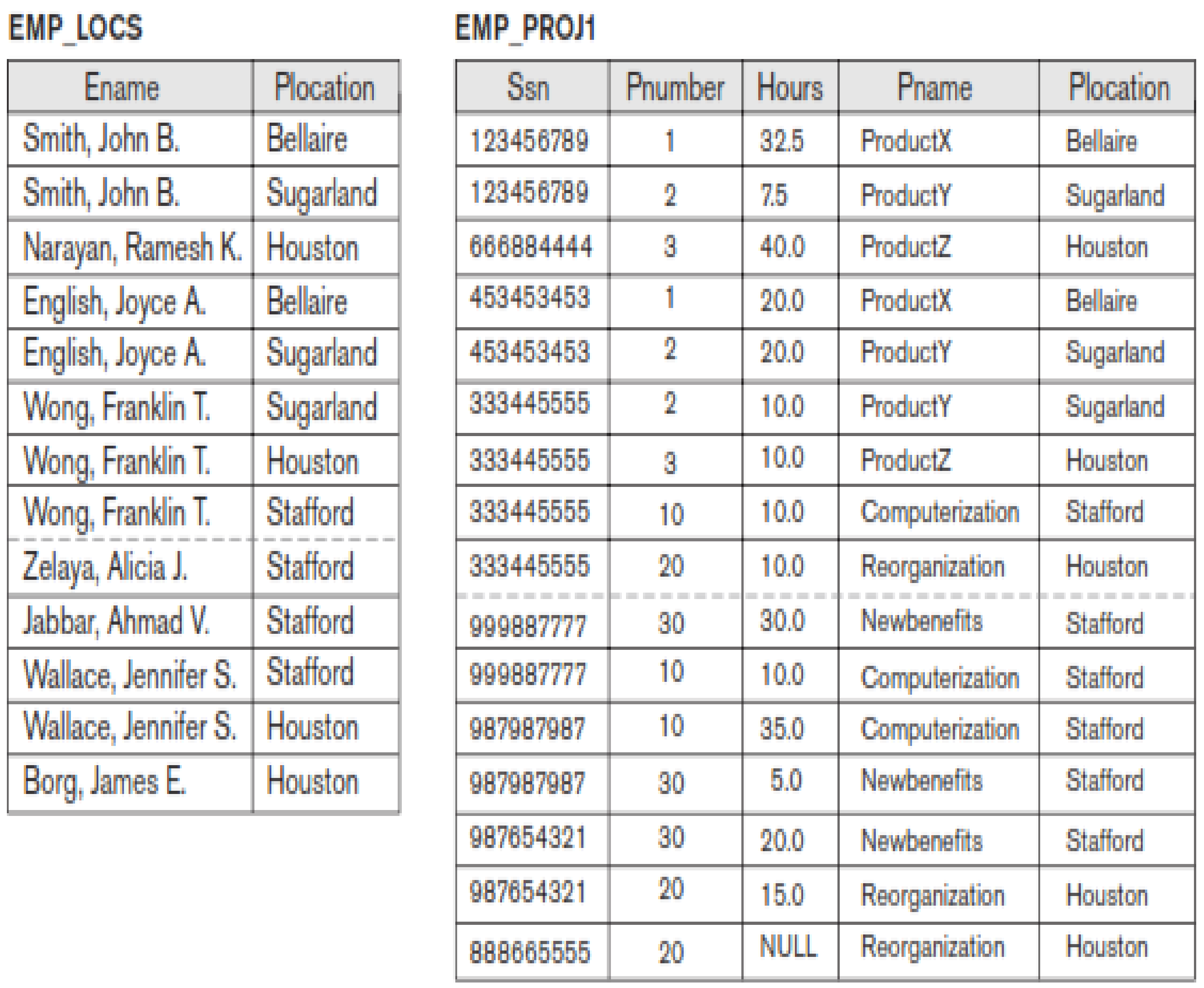
Informal Design Guidelines for Relation Schemas

Guideline 3.

- As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL. If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation.

Guideline 4.

- Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key)pairs in a way that guarantees that no spurious tuples are generated.
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

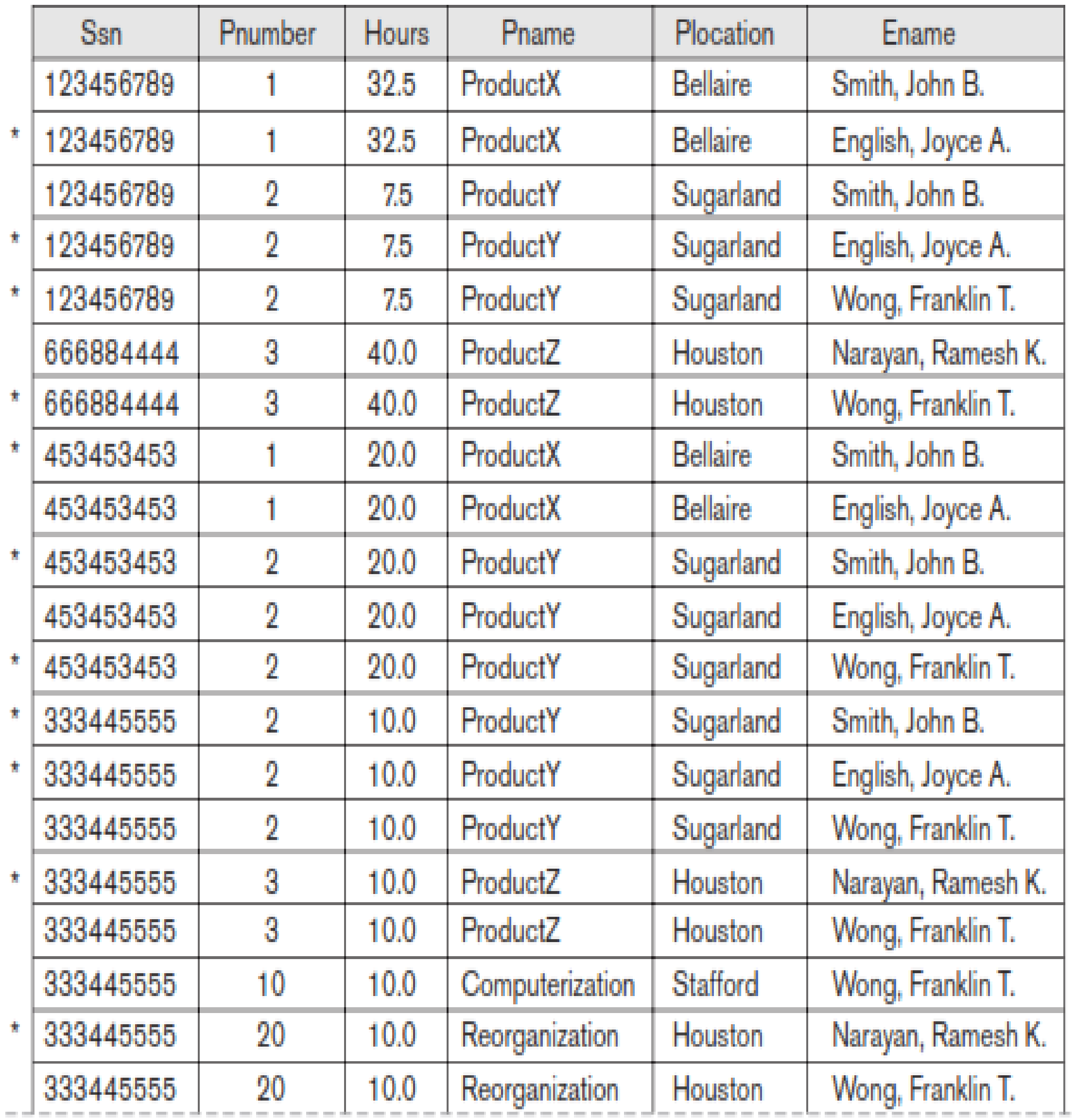


EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston



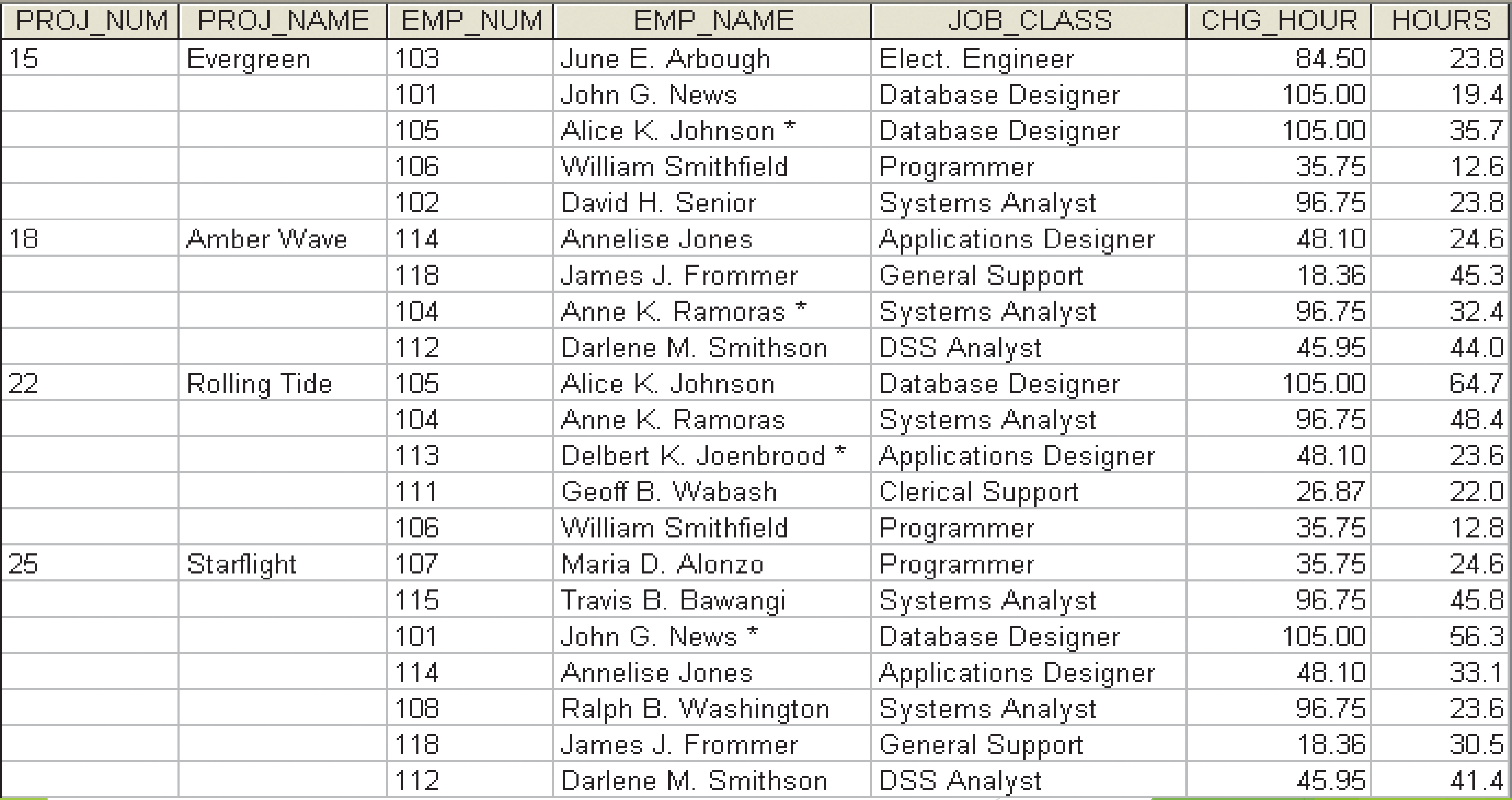
	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

DATABASE TABLES AND NORMALIZATION

- The table is the basic building block of database design
- Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies.
- Normalization works through a series of stages called normal forms. The first three stages are described as first normal form (1NF), second normal form (2NF), and third normal form (3NF).
- Generally, the higher the normal form, the more relational join operations are required to produce a specified output and the more resources are required by the database system to respond to end-user queries

THE NEED FOR NORMALIZATION

- There are two common situations in which database designers use normalization.
- When designing a new database structure based on the business requirements of the end users, the database designer will construct a data model
- After the initial design is complete, the designer can use normalization to analyze the relationships that exist among the attributes within each entity, to determine if the structure can be improved through normalization.
- Alternatively, to modify existing data structures that can be in the form of flat files, spreadsheets, or older database structures.
- An analysis of the relationships among the attributes or fields in the data structure, the database designer can use the normalization process to improve the existing data structure to create an appropriate database design.



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

THE NEED FOR NORMALIZATION

Normalization can be used to avoid following deficiencies

- Some of the primary key attribute include null values(eg :Proj_Num)
- Some of the table have data redundancies and yield insertion,deletion and updation anomalies
- Some of the table entries cause data inconsistency

THE NORMALIZATION PROCESS

- The objective of normalization is to ensure that each table conforms to the concept of well-formed relations—that is, tables that have the following characteristics:
- Each table represents a single subject. For example, a course table will contain only data that directly relate to courses. Similarly, a student table will contain only student data.
- No data item will be *unnecessarily stored in more than one table (in short, tables have minimum controlled* redundancy). The reason for this requirement is to ensure that the data are updated in only one place.
- All nonprime attributes in a table are dependent on the primary The reason for this requirement is to ensure that the data are uniquely identifiable by a primary key value.
- Each table is void of insertion, update, or deletion anomalies. This is to ensure the integrity and consistency of the data.

Functional Dependency

- **Functional Dependency (FD)** is a constraint that determines the relation of set of attributes to another set of attributes in a Database Management System (DBMS). .
- It typically exists between the primary key and non-key attribute within a table.

$X \rightarrow Y$

- The left side of FD is known as a determinant, the right side of the production is known as a dependent.
- **For example:**Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.
- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as: Emp_Id → Emp_Name

Functional Dependency

- A **functional dependency, denoted by X → Y, between two sets of** attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y]=t2[Y]

X	Y
1	1
2	1
3	2
4	3
2	5

Rno	Name	Marks	Dept	Class
1	Xyz	78	CS	A1
2	ABC	60	EE	A1
3	XYZ	78	CS	A2
4	ABC	60	EE	A2

- Rno->Name,Rno->Marks

Functional Dependency

- In a Schema with attributes A,B,C,D and E, following set of functional dependencies are given

A->B,A->C, CD->E,E->A,B->D

Which o the following FD is not implied by the above set

1)CD->AC

2)BD->CD

3)BC->CD

4)AC->BC

- First we need to find closure of determinant
- Closure of set F of FDS is the set F+ of all FDS that can be inferred from F

Functional Dependency

At first,considering ,CD->AC

(CD)+={C,D,E,A,B)

BD->CD

(BD)+={B,D}

BD cannot derive CD,Hence not implied

BC->CD

(BC)+={B,C,D,E,A}

AC->BC

(AC)+={A,C,B,D,E}

Types of Functional Dependency

- A **partial dependency exists when there is a functional dependence in which the** determinant is only part of the primary key (remember we are assuming there is only one candidate key).
- For 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
- A **transitive dependency** exists when there are functional dependencies such that

X → Y, Y → Z, and X is the primary key.

- In that case, the dependency X → Z is a transitive dependency because X determines the value of Z via Y.

Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.
- The Functional dependency has 6 types of inference rule:

Reflexive Rule (IR_1), Augmentation Rule (IR_2), Transitive Rule (IR_3), Union Rule (IR_4), Decomposition Rule (IR_5), Pseudo transitive Rule (IR_6)

Inference Rule (IR):

1.Reflexive Rule (IR_1)

- In the reflexive rule, if Y is a subset of X, then X determines Y.
- If $X \supseteq Y$ then $X \rightarrow Y$

2. Augmentation Rule (IR_2)

- The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.
- If $X \rightarrow Y$ then $XZ \rightarrow YZ$
- **Example:**
- For R(ABCD), **if** $A \rightarrow B$ then $AC \rightarrow BC$

Inference Rule (IR):

3. Transitive Rule (IR_3)

- In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.
- If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4. Union Rule (IR_4)

- Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.
- If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Inference Rule (IR):

5. Decomposition Rule (IR_5)

- Decomposition rule is also known as project rule. It is the reverse of union rule.
- This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.
- If $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

6. Pseudo transitive Rule (IR_6)

- In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

Minimal Set/Minimal Cover of Functional Dependencies (Fc)

- A minimal cover of a set of FD F is a minimal set of FDS Fmin that is equivalent to F

Steps to find minimal cover

1 Decompose the functional dependencies using Decomposition rule(Armstrong's Axiom) i.e. single attribute on right hand side.

2 Remove extraneous attributes from LHS of functional dependencies by calculating the closure of FD's having two or more attributes on LHS.

3 Remove FD's having transitivity.

Minimal Set/Minimal Cover of Functional Dependencies (Fc)

Find minimal cover of {A->C,AB->C,CD->I,C->DI,EC->AB,EI->C}

Step1:

Decompose the functional dependencies using Decomposition rule(Armstrong's Axiom) i.e. single attribute on right hand side.

F1={A->C,AB->C,CD->I,C->D,C->I,EC->A,EC->B,EI->C}

Step2:

Remove extraneous attributes from LHS of functional dependencies by calculating the closure of FD's having two or more attributes on LHS.

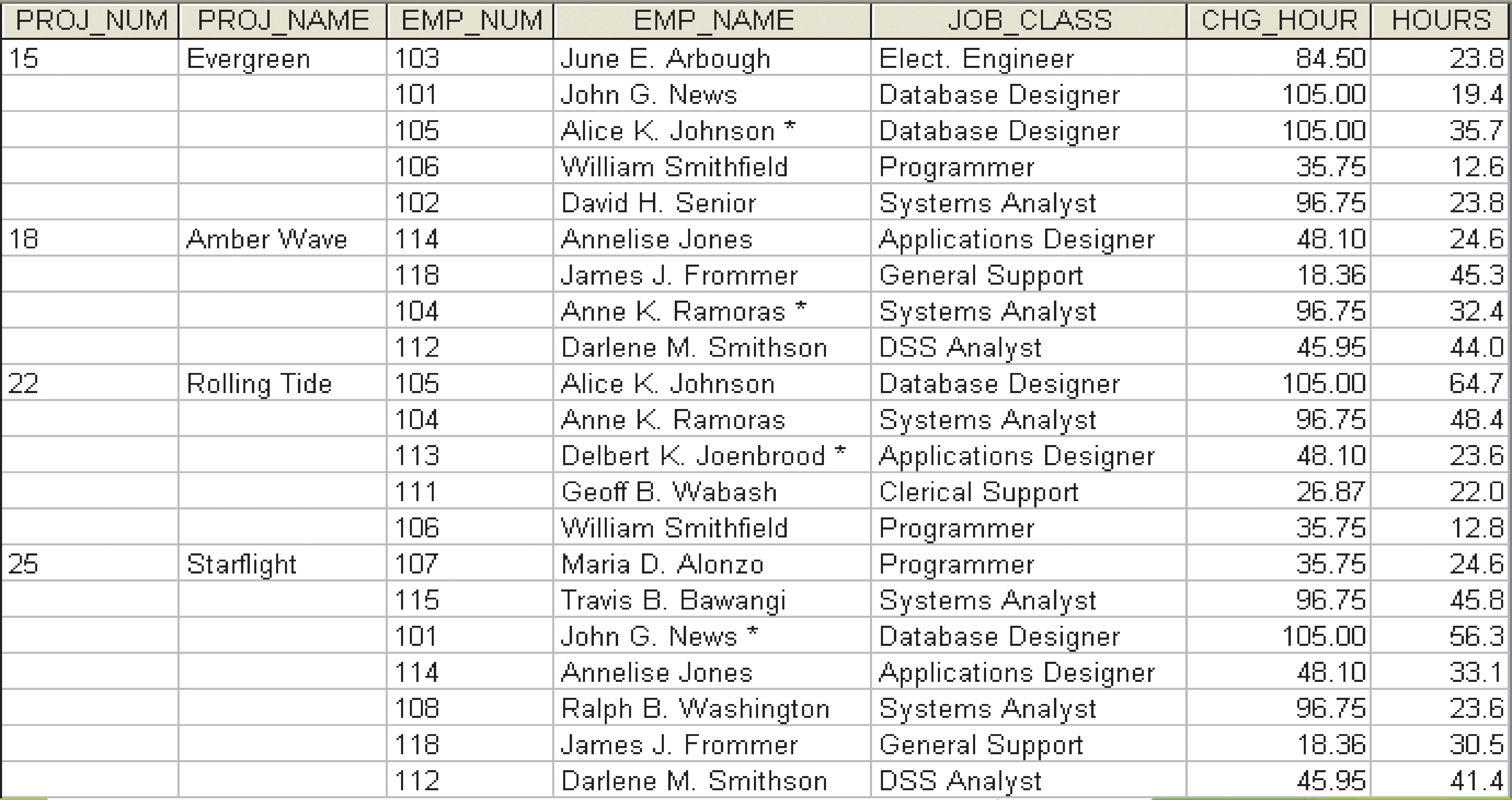
{A->C,C->D,C->I,EC->A,EC->B,EI->C}

Step3:Remove FD's having redundancy.

{A->C,C->D,C->I,EC->A,EC->B}

First Normal Form

- **Normalization of data can be considered a process of analyzing the given relation** schemas based on their FDs and primary keys to achieve the desirable properties of
- (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies
- 1NF states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute.
- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.
- In other words, 1NF disallows relations within relations or relations as attribute values within tuples.
- The only attribute values permitted by 1NF are single **atomic (or indivisible) values**



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

First Normal Form

- The normalization process starts with a simple three-step procedure.

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

Step 2: Identify the Primary Key

- Identify proper primary key that will uniquely identify any attribute value

- For Example, in the given table, key must be composed of a *combination* of PROJ_NUM and EMP_NUM

Step 3: Identify All Dependencies

- The identification of the PK in Step 2 means that we can identify dependency in tables
- PROJ_NUM,EMP_NUM→ PROJ_NAME,EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS
- Other dependency are

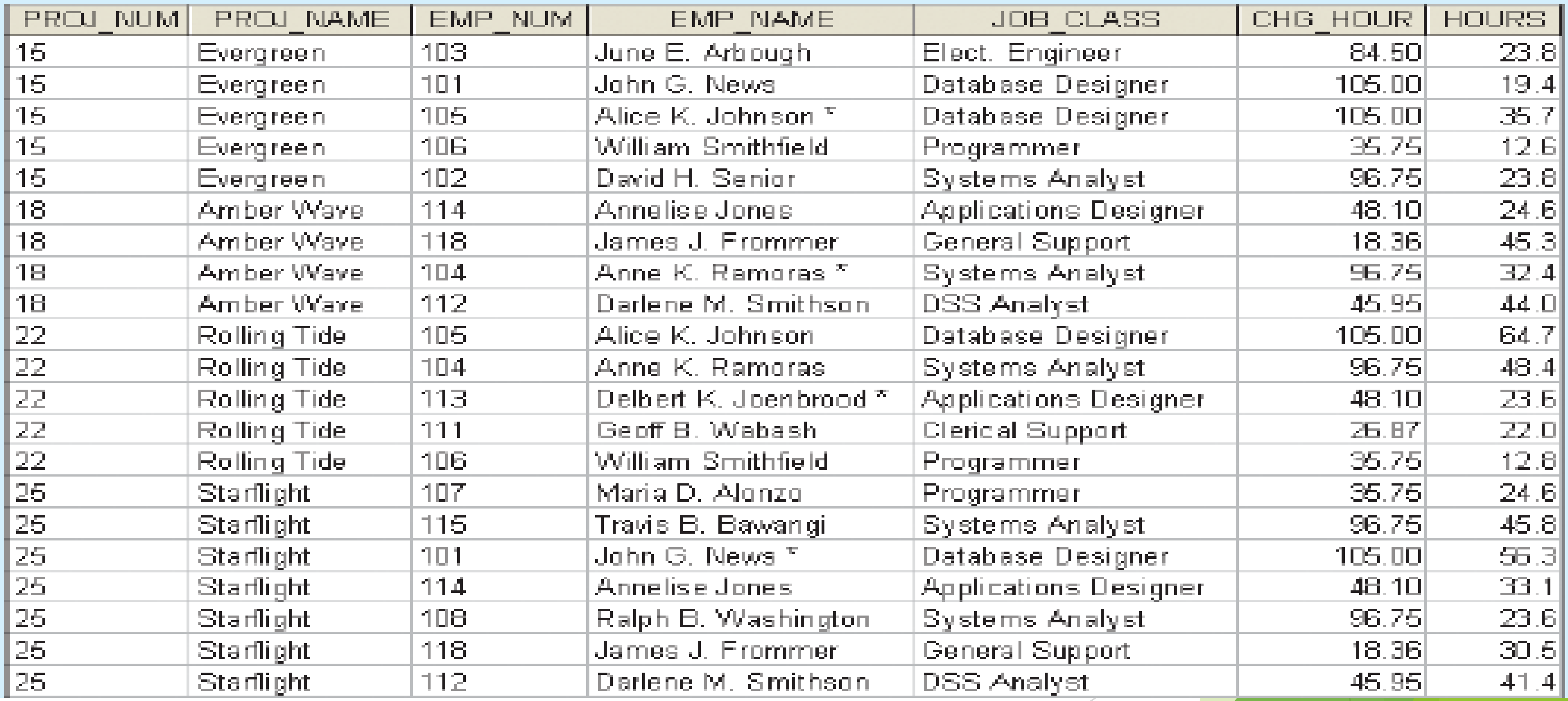
PROJ_NUM → PROJ_NAME

FIGURE 6.2

A table in first normal form

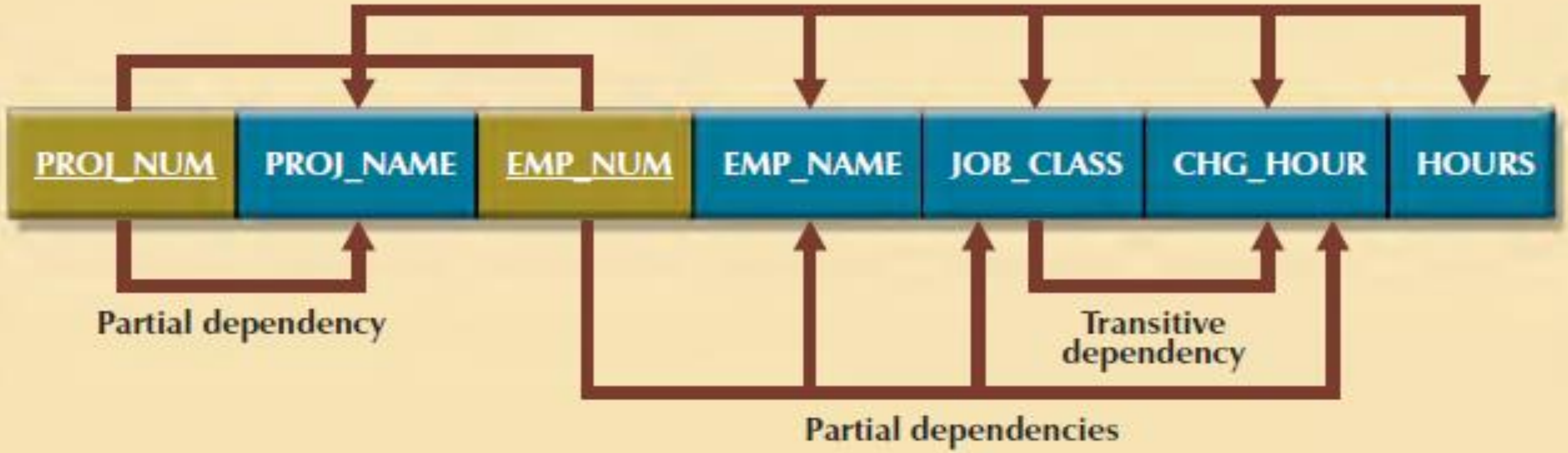
Table name: DATA_ORG_1NF

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 (<u>PROJ_NUM</u>, <u>EMP_NUM</u>, 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)

First Normal Form

- The problem with the 1NF table structure is that it contains partial dependencies—that is, dependencies based on only a part of the primary key.
- A table that contains partial dependencies is subject to data redundancies, and therefore, to various anomalies

Second Normal Form

- A table is in **second normal form (2NF) when:**
- It is in 1NF *and*
- It includes no partial dependencies; that is, no attribute is dependent on only a portion of the primary key.

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. The 1NF-to-2NF conversion is simple

Second Normal Form

Steps for conversion to Second Normal Form

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.
- While these components are placed in the new tables, it is important that they also remain in the original table as well.
- It is important that the determinants remain in the original table because they will be the foreign keys for the relationships that are needed to relate these new tables to the original table

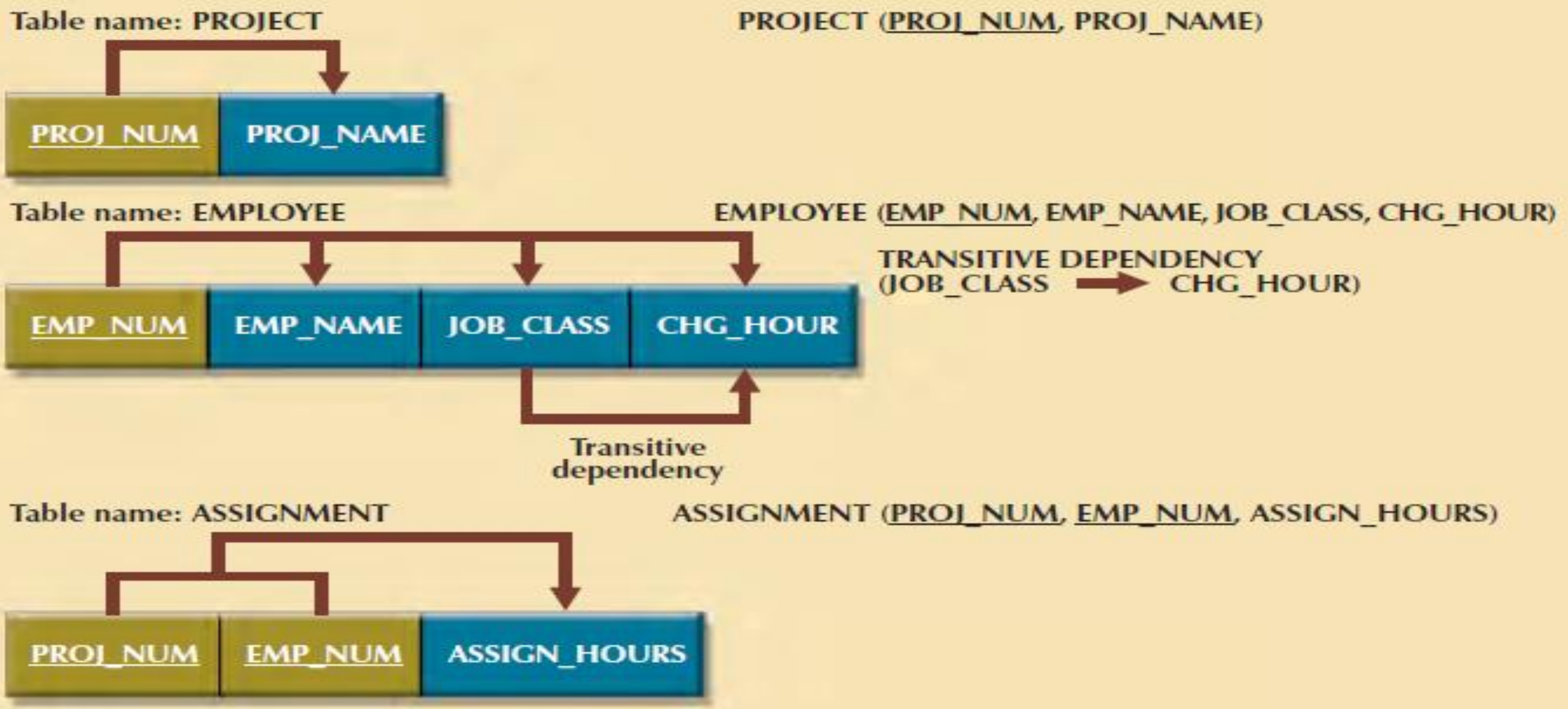
Second Normal Form

Steps for conversion to Second Normal Form

Step 2: Reassign Corresponding Dependent Attributes

- The attributes that are dependent in a partial dependency are removed from the original table and placed in the new table with its determinant.
- Any attributes that are not dependent in a partial dependency will remain in the original table.
- In other words, the three tables that result from the conversion to 2NF are given appropriate names (PROJECT, EMPLOYEE, and ASSIGNMENT) and are described by the following relational schemas:
- 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



Third Normal Form

- A table is in **third normal form (3NF) when:**
- It is in 2NF *and* It contains no transitive dependencies

Conversion to Third Normal Form

- The data anomalies created by the database in 2NF are easily eliminated by completing the following two steps:

Step 1: Make New Tables to Eliminate Transitive Dependencies

Step 2: Reassign Corresponding Dependent Attributes

Third Normal Form

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.
- A **determinant is** any attribute whose value determines other values within a row.
- If we have three different transitive dependencies, we 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

Third Normal Form

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
- Draw a new dependency diagram to show all of the tables you have defined in Steps 1 and 2. Name the table to reflect its contents and function
- Check all of the tables to make sure that each table has a determinant and that no table contains inappropriate dependencies.

FIGURE 6.5 Third normal form (3NF) conversion results

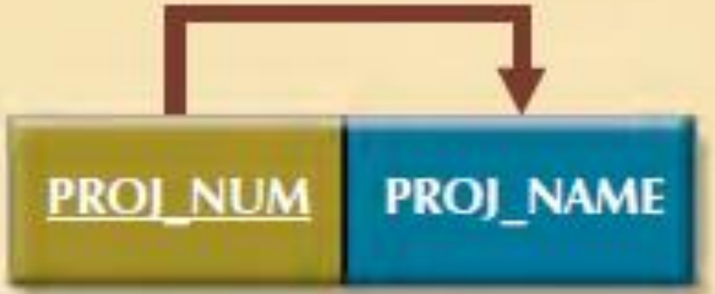


Table name: PROJECT

PROJECT (PROJ_NUM, PROJ_NAME)

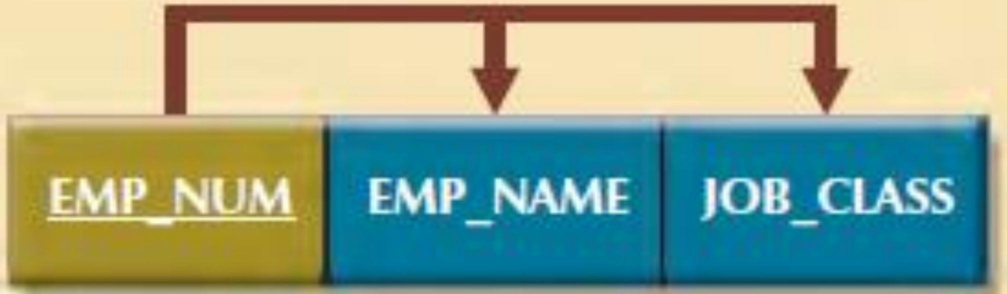


Table name: EMPLOYEE

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

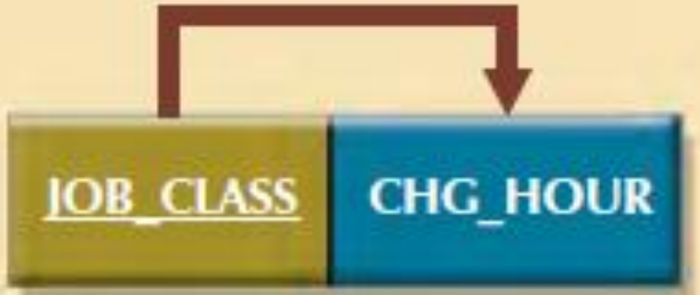


Table name: JOB

JOB (JOB_CLASS, CHG_HOUR)

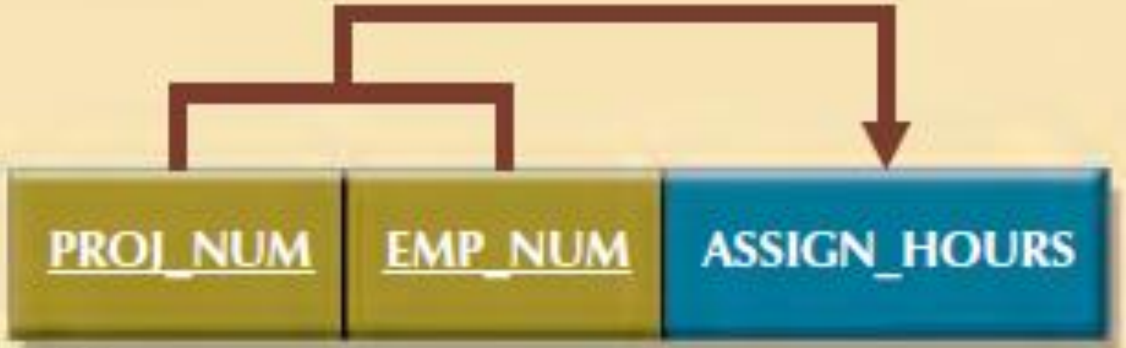


Table name: ASSIGNMENT

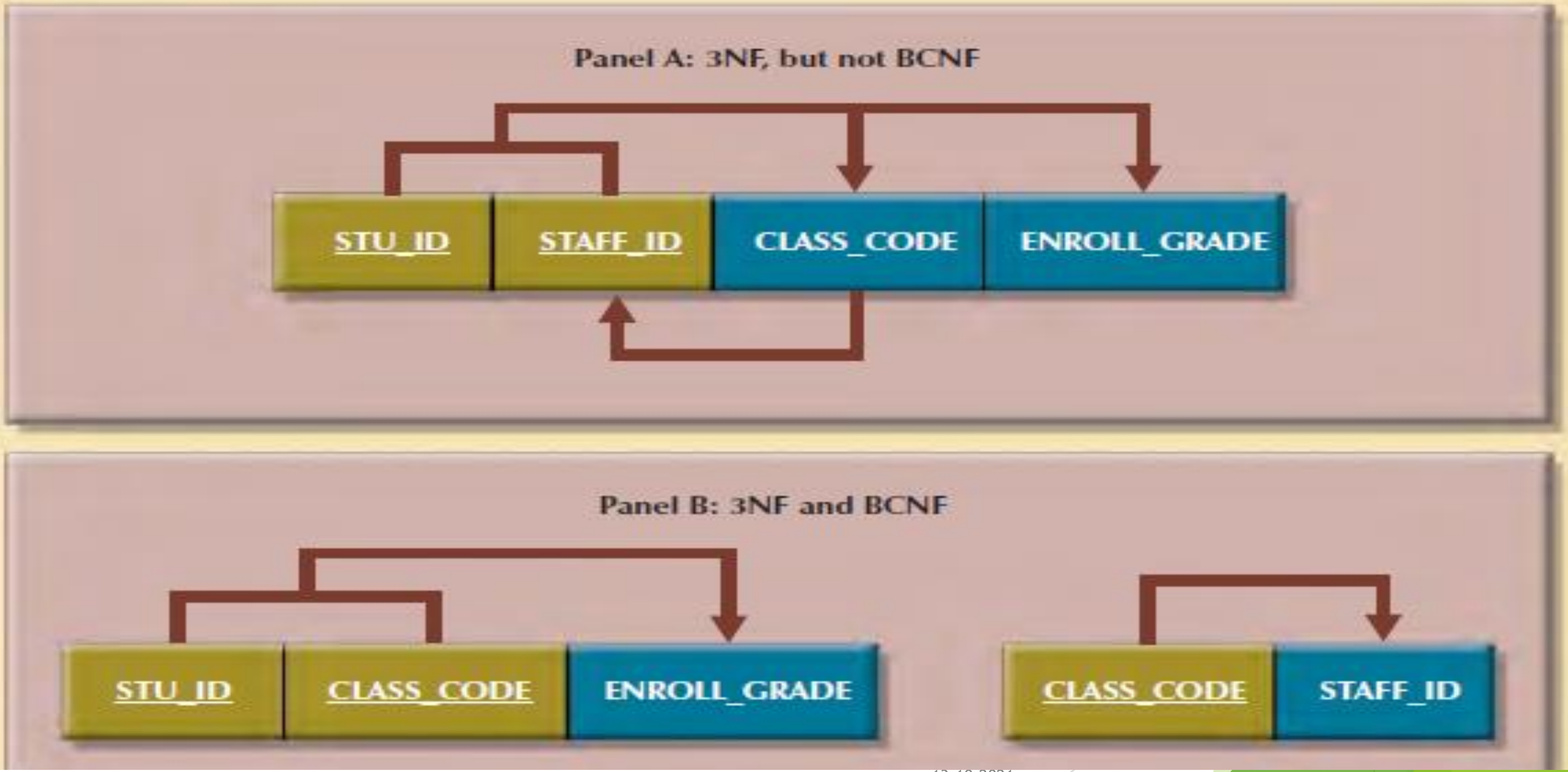
ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

HIGHER-LEVEL NORMAL FORMS

The Boyce- Codd Normal Form (BCNF)

- A table is in Boyce- Codd normal form (BCNF) when every determinant in the table is a candidate key.
- Therefore, when a table contains only one candidate key, 3NF and BCNF are equivalent
- Panel A of Figure shows a structure that is clearly in 3NF, but the table represented by this structure has a major problem, because it is trying to describe two things: staff assignments to classes and student enrollment information.
- Such a dual-purpose table structure will cause anomalies
- For example, if a different staff member is assigned to teach class 32456, two rows will require updates, thus producing an update anomaly and if student 135 drops class 28458

FIGURE 6.9 Another BCNF decomposition



HIGHER-LEVEL NORMAL FORMS

The Boyce- Codd Normal Form (BCNF)

- Information about who taught that class is lost, thus producing a deletion anomaly. The solution to the problem is to decompose the table structure
- Note that the decomposition of Panel B shown in Figure yields two table structures that conform to both 3NF and BCNF requirements

HIGHER-LEVEL NORMAL FORMS

Multi valued Dependency

- Multi valued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute
- A multi valued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least 3 attributes
- Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year

BIKE_MODEL	MANF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

HIGHER-LEVEL NORMAL FORMS

4NF

- A table is in fourth normal form (4NF) when it is in 3NF and has no multi valued dependencies.
- For a dependency A->B,if for a single value of A, multiple value of B exists,then the relation will be a multi valued dependency

STUD_ID	COURSE	HOBBY
21	Computer	Dancing
21	Maths	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

HIGHER-LEVEL NORMAL FORMS

4NF

- The given Student table is in 3NF,but the COURSE and HOBBY are two independent entity. Hence there is no relationship between COURSE and HOBBY
- In the Student relation, a Student with STUD_ID 21 contains two courses, computer and maths and two hobbies, dancing and singing. So there is a multi valued dependency on STUD_ID ,which leads to unnecessary repetition of data
- So make the above table into 4NF,we can decompose it into two tables

STUD_ID	COURSE
21	Computer
21	Maths
34	Chemistry
74	Biology
59	Physics

STUD_ID	HOBBY
21	dancing
21	singing
34	dancing
74	cricket
59	Hockey

HIGHER-LEVEL NORMAL FORMS

Join Dependency

- If a table can be recreated by joining multiple tables and each of this table have a subset of attributes of the tables, then table is in join dependency

EMPNAME	EMPSKILLS	EMPJOB
Torn	Networking	EJ001
Harry	Web Developmen t	EJ002
Katie	Programmin g	EJ002

EMPNAME	EMPJOB
Torn	EJ001
Harry	EJ002
Katie	EJ002

EMPNAME	EMPSKILLS
Torn	Networking
Harry	Web Developme nt
Katie	Programmi ng

EMPSKILLS	EMPJOB
Networking	EJ001
Web Development	EJ002
Programming	EJ002

HIGHER-LEVEL NORMAL FORMS

5NF/PJNF

- A relation is said to be in 5NF if and only if it satisfies 4NF and no join dependency exists

Consider the relation given below

Company	Product	Suppliers
Godrej	Soap	Vinu
Godrej	Shampoo	Manu
Godrej	Shampoo	Vinu
H.Lever	Soap	Vinu
H.Lever	Shampoo	Manu
H.Lever	Soap	Arjun

HIGHER-LEVEL NORMAL FORMS

5NF/PJNF

Company	Product
Godrej	Soap
Godrej	Shampoo
H.Lever	Soap
H.Lever	Shampoo

Company	Suppliers
Godrej	Vinu
Godrej	Manu
H.Lever	Vinu
H.Lever	Manu
H.Lever	Arjun

Product	Suppliers
Soap	Vinu
Shampoo	Manu
Shampoo	Vinu
Soap	Arjun

IMPROVING THE DESIGN

1Evaluate PK Assignments

- Each time a new employee is entered into the EMPLOYEE table, a JOB_CLASS value must be entered.
- For example, entering DB Designer instead of Database Designer for the JOB_CLASS attribute in the EMPLOYEE table will trigger such a violation.
- Therefore, it would be better to add a JOB_CODE attribute to create a unique identifier. The addition of a JOB_CODE attribute produces the dependency:
- JOB_CODE → JOB_CLASS, CHG_HOUR
- If you assume that the JOB_CODE is a proper primary key, this new attribute does produce the dependency: JOB_CLASS → CHG_HOUR

IMPROVING THE DESIGN

2. Evaluate Naming Conventions

- It is best practice to follow naming conventions outlined in Data Models
- Therefore, CHG_HOUR will be changed to JOB_CHG_HOUR to indicate its association with the JOB table.
- In addition, the attribute name JOB_CLASS does not quite describe entries such as Systems Analyst, Database Designer, and so on; the label JOB_DESCRIPTION fits the entries better.

3 **Refine Attribute Atomicity**

- It is generally good practice to use atomicity requirement
- An **atomic attribute is one that cannot** be further subdivided. Such an attribute is said to display **atomicity.**
- Clearly, the use of the EMP_NAME in the EMPLOYEE table is not atomic because EMP_NAME can be decomposed into a last name, a first name, and an initial.
- By improving the degree of atomicity, you also gain querying flexibility.

IMPROVING THE DESIGN

3 **Refine Attribute Atomicity**

- For example, if you use EMP_LNAME, EMP_FNAME, and EMP_INITIAL, you can easily generate phone lists by sorting last names, first names, and initials.
- Such a task would be very difficult if the name components were within a single attribute.
- In general, designers prefer to use simple, single-valued attributes as indicated by the business rules and processing requirements

4 **Identify New Attributes**

- If the EMPLOYEE table were used in a real-world environment, several other attributes would have to be added. For example, year-to-date gross salary payments Social Security payments, and Medicare payments would be desirable
- An employee hire date attribute (EMP_HIREDATE) could be used to track an employee's job longevity and serve as a basis for awarding bonuses to long-term employees

IMPROVING THE DESIGN

5 **Identify New Relationships**

- The designer must take care to place the right attributes in the right tables by using normalization principles.
- According to the original report, the users need to track which employee is acting as the manager of each project.
- This can be implemented as a relationship between EMPLOYEE and PROJECT. From the original report, it is clear that each project has only one manager.
- Therefore, the system's ability to supply detailed information about each project's manager is ensured by using the EMP_NUM as a foreign key in PROJECT

IMPROVING THE DESIGN

6 **Refine Primary Keys as Required for Data Granularity**

- Granularity refers to the level of detail represented by the values stored in a table's row.
- Data stored at their lowest level of granularity are said to be *atomic data*
- For example, assume that the combination of EMP_NUM and PROJ_NUM is an acceptable (composite) primary key in the ASSIGNMENT table.
- That primary key is useful in representing only the total number of hours an employee worked on a project since its start.
- Using a surrogate primary key such as ASSIGN_NUM provides lower granularity and yields greater flexibility

7 **Maintain Historical Accuracy**

IMPROVING THE DESIGN

8. **Evaluate Using Derived Attributes**

- Storing the derived attribute in the table makes it easy to write the application software

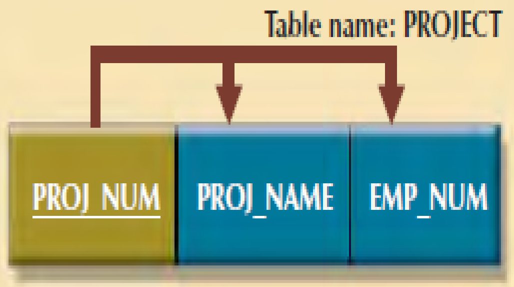
to produce the desired results.

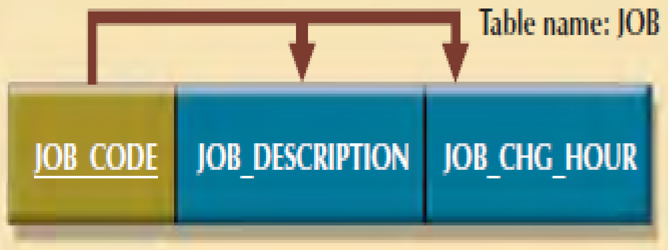
- For example,we can use a derived attribute in the ASSIGNMENT table to store the actual charge made to a project
- That derived attribute, to be named ASSIGN_CHARGE, is the result of multiplying ASSIGN_HOURS by ASSIGN_CHG_HOUR.
- This creates a transitive dependency such that

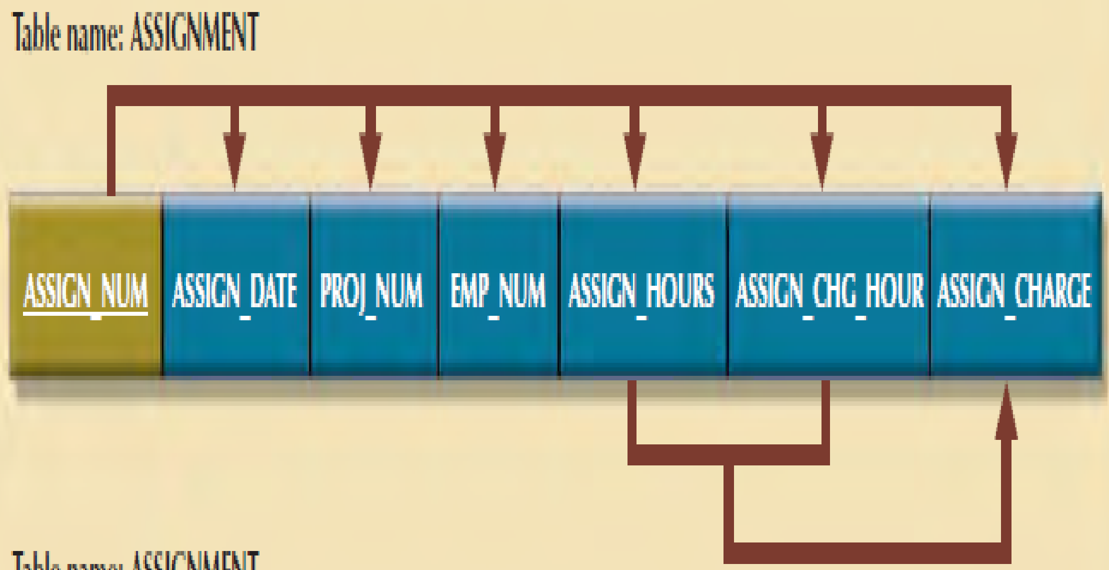
(ASSIGN_CHARGE + ASSIGN_HOURS) → ASSIGN_CHG_HOUR.

- From a database point of view, such derived attribute values can be calculated when they are needed to write reports or invoices
- Also, if many transactions must be reported and/or summarized, the availability of the derived attribute will save reporting time.

IMPROVING THE DESIGN

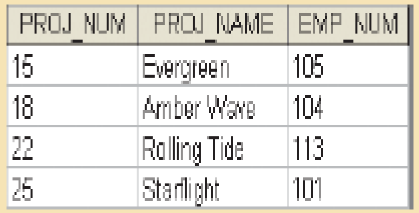






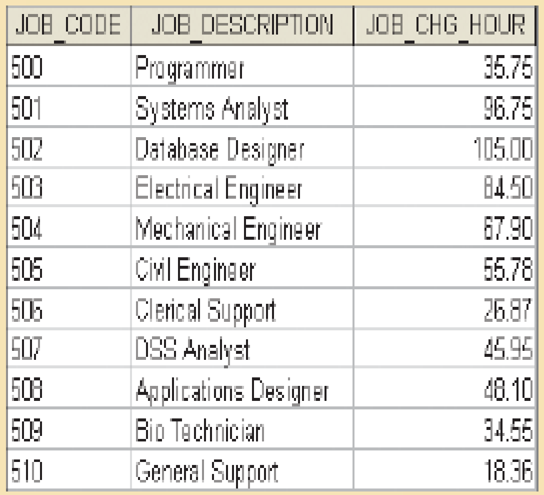
Database name: Ch06_ConstructCo

Table name: PROJECT



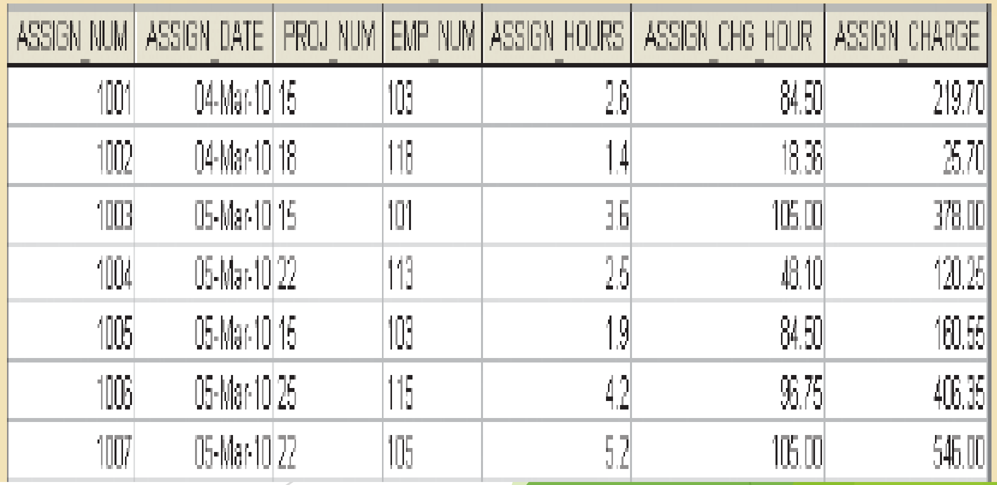
PROJ_NUM	PROJ_NAME	EMP_NUM
15	Evergreen	105
18	Amber Wave	104
22	Rolling Tide	113
25	Starflight	101

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

Table name: ASSIGNMENT



ASSIGN_NUM	ASSIGN_DATE	PROJ_NUM	EMP_NUM	ASSIGN_HOURS	ASSIGN_CHG_HOUR	ASSIGN_CHARGE
1001	04-Mar-10	15	103	2.6	84.50	219.70
1002	04-Mar-10	18	118	1.4	18.36	25.70
1003	05-Mar-10	15	101	3.6	105.00	378.00
1004	05-Mar-10	22	113	2.5	48.10	120.25
1005	05-Mar-10	15	103	1.9	84.50	160.55
1006	05-Mar-10	25	115	4.2	96.75	406.35
1007	05-Mar-10	22	105	5.2	105.00	546.00

SURROGATE KEY CONSIDERATIONS

- At the implementation level, a surrogate key is a system-defined attribute generally created and managed via the DBMS.
- Usually, a system-defined surrogate key is numeric, and its value is automatically incremented for each new row.
- For example, Microsoft Access uses an AutoNumber data type, Microsoft SQL Server uses an identity column, and Oracle uses a sequence object.

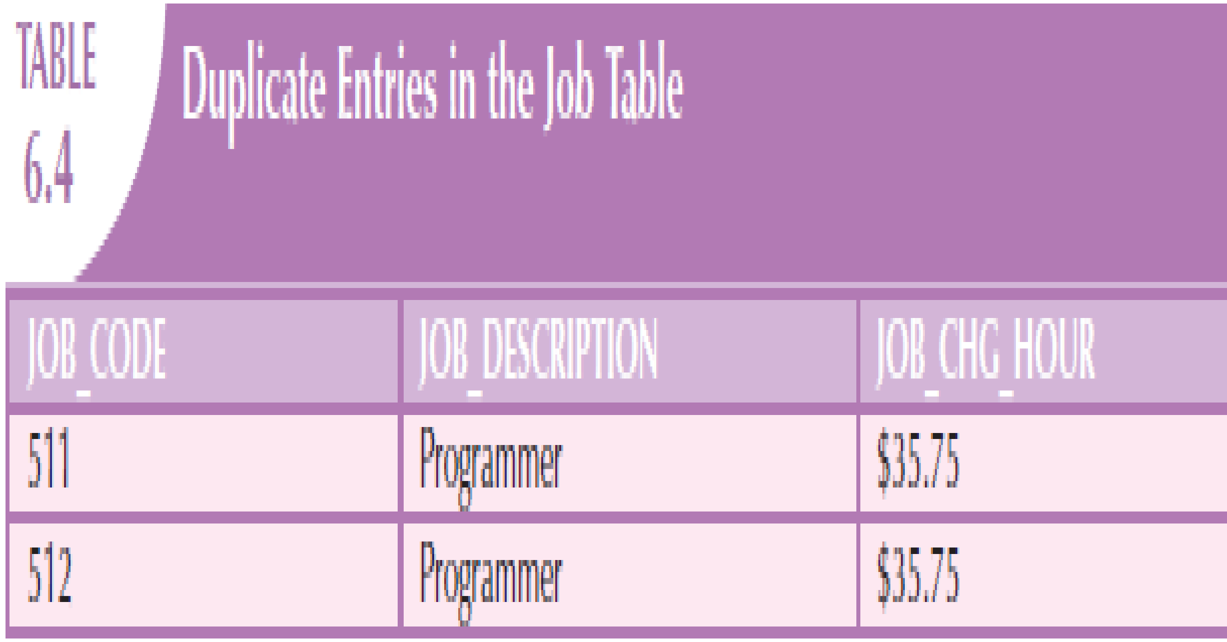


TABLE 6.4 Duplicate Entries in the Job Table

JOB_CODE	JOB_DESCRIPTION	JOB_CHG_HOUR
511	Programmer	$35.75
512	Programmer	$35.75

SURROGATE KEY CONSIDERATIONS

- The data entries in Table are inappropriate because they duplicate existing records—yet there has been no violation of either entity integrity or referential integrity
- This “multiple duplicate records” problem was created when the JOB_CODE attribute was added as the PK.
- In any case, if JOB_CODE is to be the surrogate PK, you still must ensure the existence of unique values in the JOB_DESCRIPTION *through the use of a unique index*

NORMALIZATION AND DATABASE DESIGN

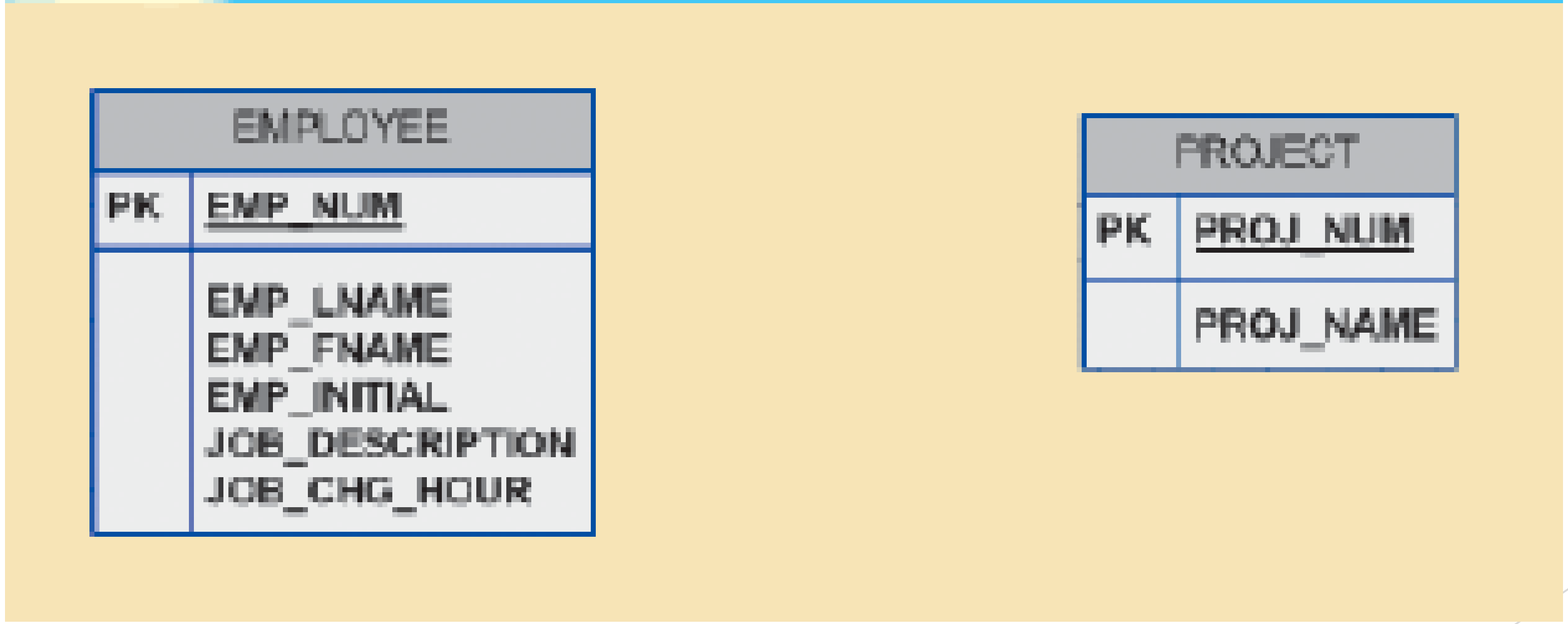
- First, an ERD is created through an iterative process. Begin design by identifying relevant entities, their attributes, and their relationships.
- Then use the results to identify additional entities and attributes.
- Second, normalization focuses on the characteristics of specific entities; that is, normalization represents a micro view of the entities within the ERD
- For Example, the operations of the contracting company whose tables were normalized in the preceding sections. Those operations can be summarized by using the following business rules:
- The company manages many projects.
- Each project requires the services of many employees.
- An employee may be assigned to several different projects.
- Some employees are not assigned to a project and perform duties not specifically related to a project.

NORMALIZATION AND DATABASE DESIGN

- Some employees are part of a labor pool, to be shared by all project teams. For example, the company's executive secretary would not be assigned to any one particular project.
- Each employee has a single primary job classification. That job classification determines the hourly billing rate.
- Many employees can have the same job classification. For example, the company employs more than one electrical engineer
- Given that simple description of the company's operations, two entities and their attributes are initially defined:
- PROJECT (**PROJ_NUM, PROJ_NAME)**
- EMPLOYEE (**EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, JOB_DESCRIPTION,**JOB_CHG_HOUR)

NORMALIZATION AND DATABASE DESIGN

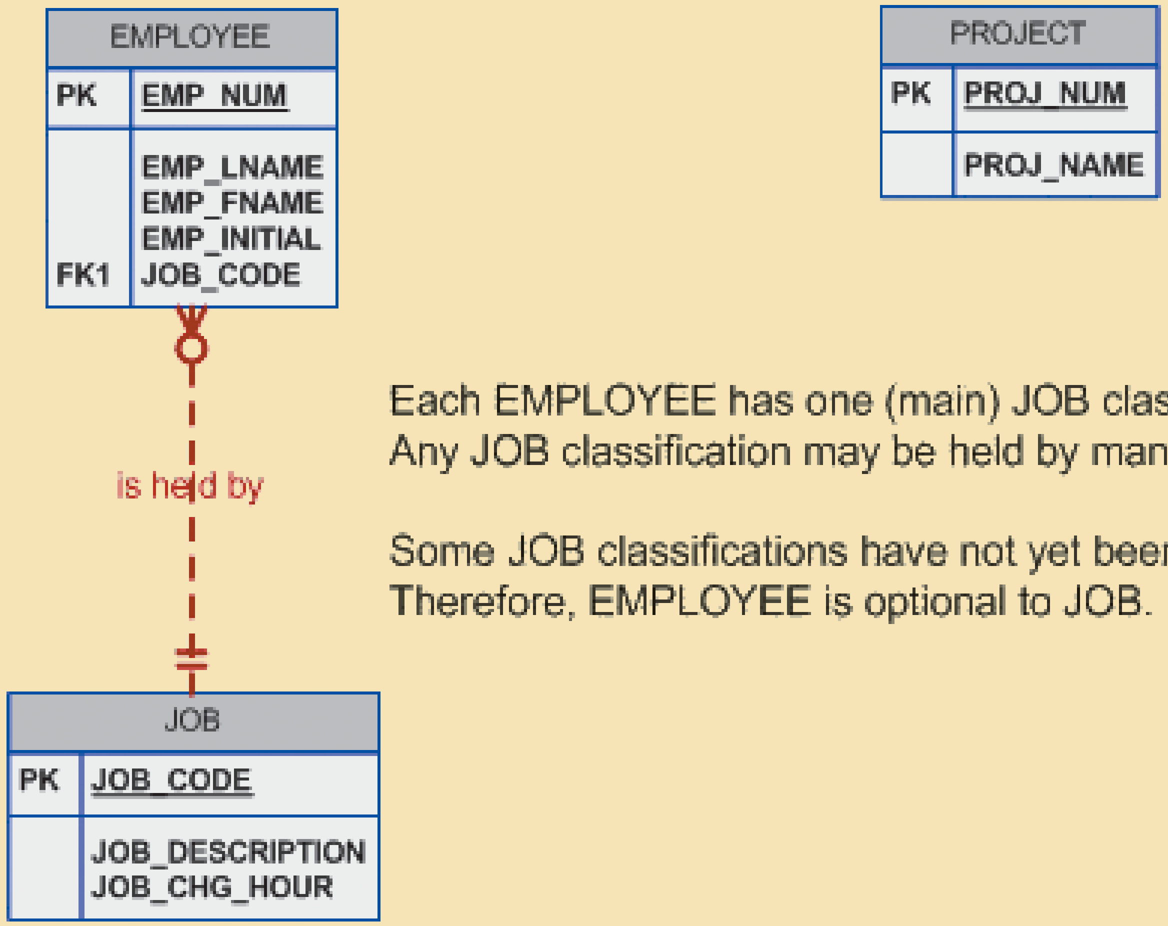
FIGURE 6.12 Initial contracting company ERD



NORMALIZATION AND DATABASE DESIGN

- After creating the initial ERD shown in Figure , the normal forms are defined:
- PROJECT is in 3NF and needs no modification at this point.
- EMPLOYEE requires additional scrutiny. The JOB_ DESCRIPTION attribute defines job classificationssuch as Systems Analyst, Database Designer, and Programmer.
- In turn, those classifications determine the billing rate, JOB_CHG_HOUR. Therefore, EMPLOYEE contains a transitive dependency.
- The removal of EMPLOYEE's transitive dependency yields three entities:
- PROJECT (**PROJ_NUM, PROJ_NAME)**
- EMPLOYEE (**EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, JOB_CODE)**
- JOB (**JOB_CODE, JOB_DESCRIPTION, JOB_CHG_HOUR)**

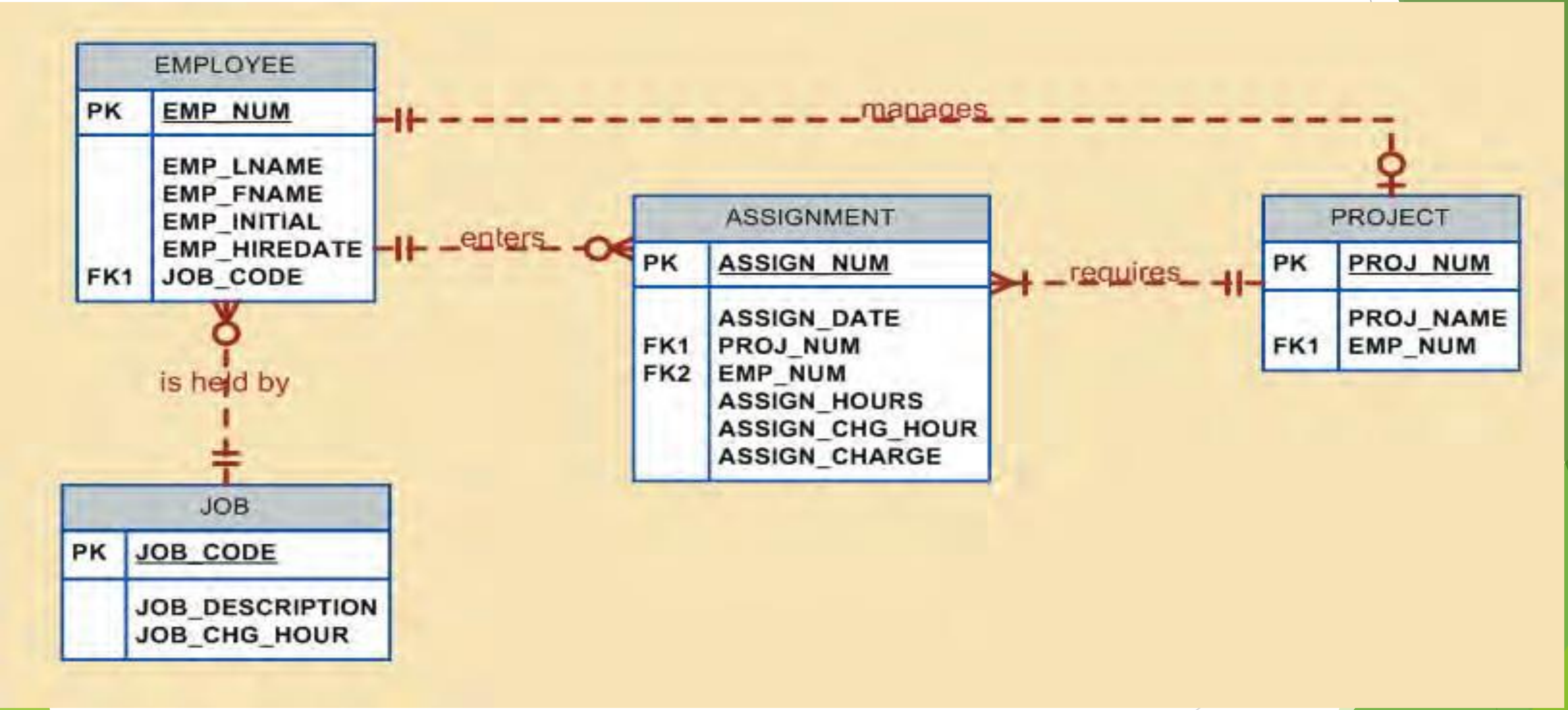
NORMALIZATION AND DATABASE DESIGN



Each EMPLOYEE has one (main) JOB classification.
Any JOB classification may be held by many EMPLOYEEs.

Some JOB classifications have not yet been staffed.
Therefore, EMPLOYEE is optional to JOB.

NORMALIZATION AND DATABASE DESIGN


EMPLOYEE
PK EMP_NUM
EMP_LNAME
EMP_FNAME
EMP_INITIAL
EMP_HIREDATE
FK1 JOB_CODE
manages
enters
is held by
ASSIGNMENT
PK ASSIGN_NUM
ASSIGN_DATE
FK1 PROJ_NUM
FK2 EMP_NUM
ASSIGN_HOURS
ASSIGN_CHG_HOUR
ASSIGN_CHARGE
requires
PROJECT
PK PROJ_NUM
PROJ_NAME
FK1 EMP_NUM
JOB
PK JOB_CODE
JOB_DESCRIPTION
JOB_CHG_HOUR