SIT103/SIT772 Data and Information Management

Week 2

Database Design

T3, 2023

Last Week

- Data versus Information
- What is a database and why it is important?
- User data and meta data
- File Systems vs Database
- Database life cycle
- Understanding system's data requirement

  - Context level DFD
- Database design and its importance

Last Week's OnTrack Tasks

- **1.1P** Reflection on three data-driven information systems you use in your daily life

- **1.2P** Installing and setting up MySQL Environment for SQL

  - MySQL community Server

  - MySQL Workbench

Complete and upload the two tasks before the due!

Questions?

Any questions/comments so far

Last week's content

Workshop sessions

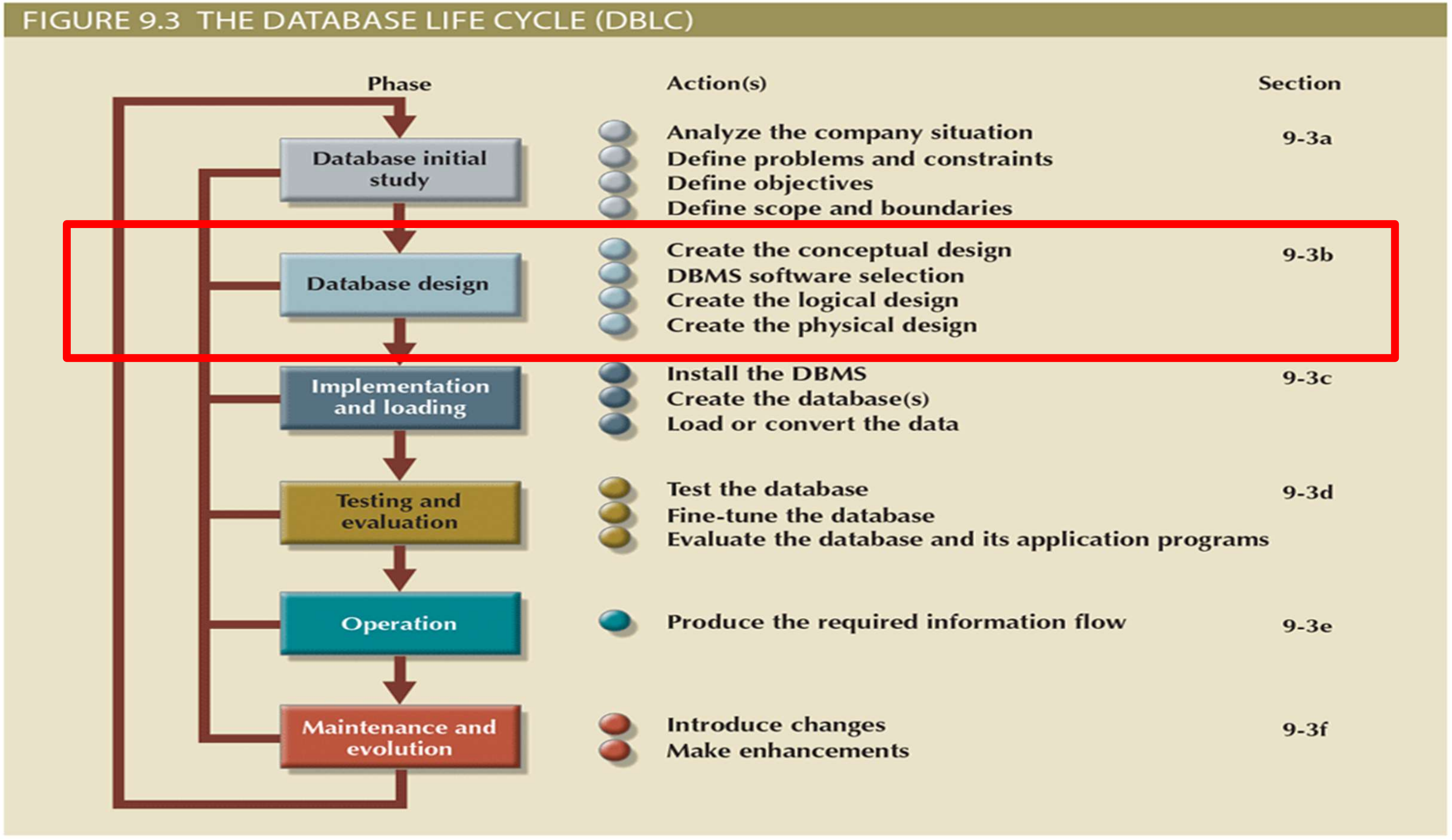
OnTrack tasks

Anything in general about the unit

Database Design

Figure/table is from Coronel & Morris (2018)

FIGURE 9.3 THE DATABASE LIFE CYCLE (DBLC)


Phase
Action(s)
Section
Database initial study
Analyze the company situation
Define problems and constraints
Define objectives
Define scope and boundaries
9-3a
Database design
Create the conceptual design
DBMS software selection
Create the logical design
Create the physical design
9-3b
Implementation and loading
Install the DBMS
Create the database(s)
Load or convert the data
9-3c
Testing and evaluation
Test the database
Fine-tune the database
Evaluate the database and its application programs
9-3d
Operation
Produce the required information flow
9-3e
Maintenance and evolution
Introduce changes
Make enhancements
9-3f

Database Design

- How to implement a database to meet system's data requirement?
- **Modelling** - developing a structure of DB using **Models**
- **Model:** abstraction (simple representation) of complex real-world object/event

Building a House	**Building a DB**
Blueprint, 3D Models	Data Models

- Different types/views of design/models

  - Different users have own views/needs

Relational database

- Proposed by Edgar Frank Codd at IBM in 1970
- Based on "**relational**" model
- Presents data as "**relations**" (a set of related **tables**)
  - A table contains data of a **<u>group of related entities</u>** e.g., Customers
  - A table consists of **records/tuples** (rows) and **attributes** (columns)

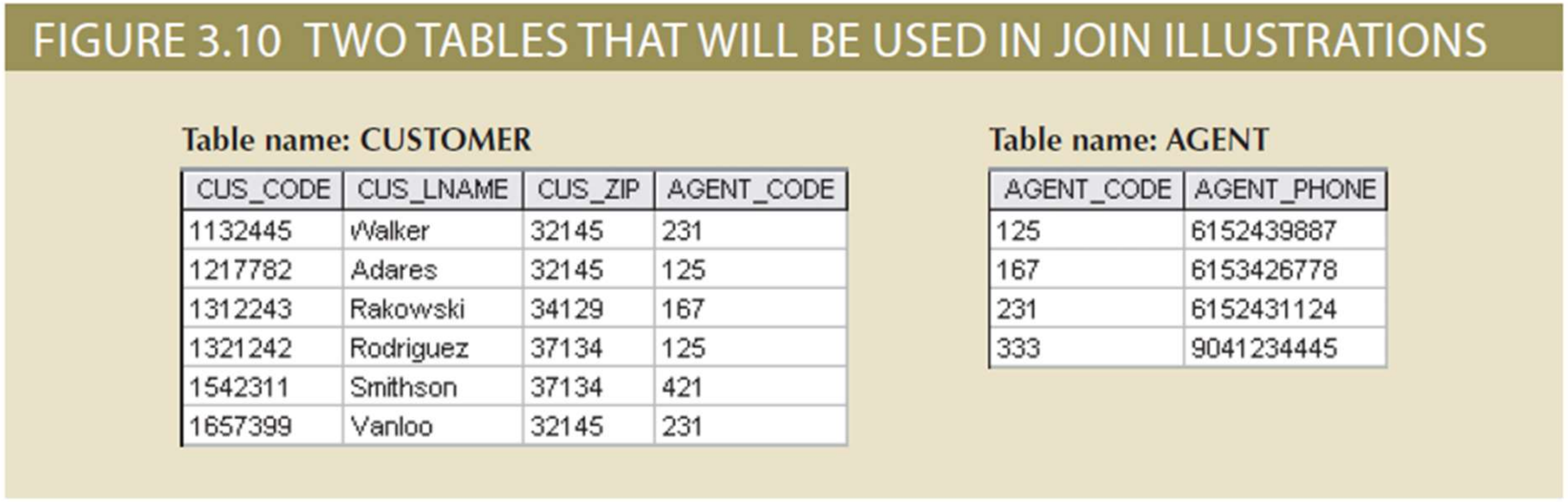


FIGURE 3.10 TWO TABLES THAT WILL BE USED IN JOIN ILLUSTRATIONS

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

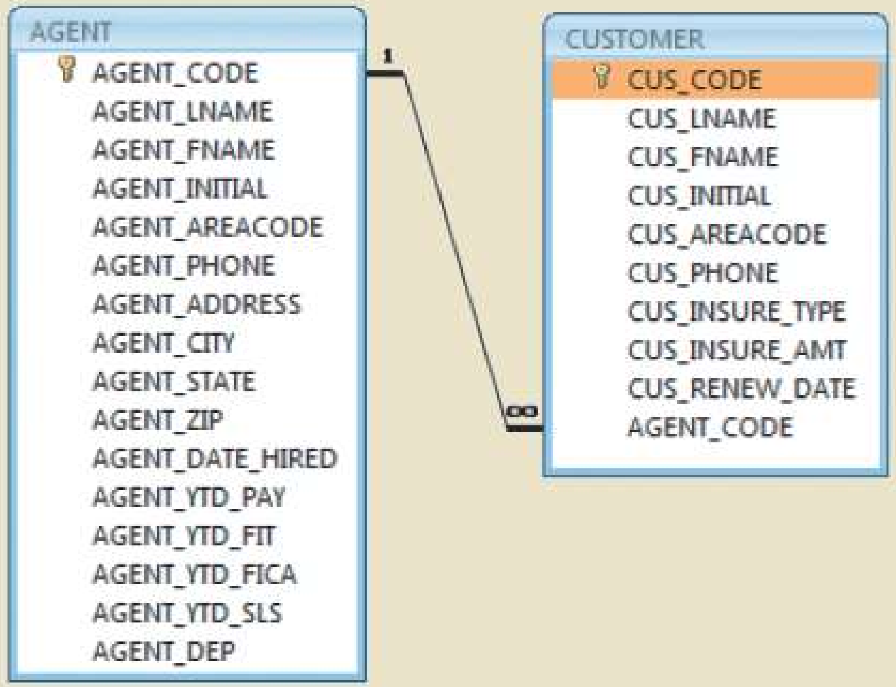
Characteristics of a Table

	Characteristics of a Relational Table
1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (tuple) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the attribute domain.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

Relational Model

- Tables are linked by a common column (e.g., AGENT_CODE in the two tables

FIGURE 2.2 A RELATIONAL DIAGRAM



An agent can have many customers but a customer has only one agent.

Relational Model: Building blocks

- **Entity**: person, place, thing, or event about which data will be collected and stored
  - e.g., Student, Course, Product, Order, Transaction, etc.
- **Attribute**: characteristic of an entity
  - e.g., ID, Name, DoB, Address, etc.
- **Relationship**: association among entities
  - One-to-one (1:1 OR 1..1)
  - One-to-many (1:M OR 1..*)
  - Many-to-many (M:N OR *..*)

E.g., A student is enrolled in one course
An order has many products

- **Constraint**: restriction placed on data
  - Ensures data integrity, e.g., Unique, Not NULL, etc.

E.g., Student ID is unique and can not be NULL

(Dis)Advantages of RDB

- Advantages
  - simple and easy to use
  - limits data redundancy
  - maintains data integrity and security
  - security and access control
  - offers logical/physical independence
- Disadvantages
  - requires good design of structure
  - some systems can be expensive
  - issue with handling unstructured data
  - performance can be an issue

DB design steps

- Identifying Entities, Attributes, Relationships and Constrains

  - Business rules

    - Nouns may translate to Entities or Attributes
    - Verbs may translate to relationships
    - Conditions and requirements may translate to constraints

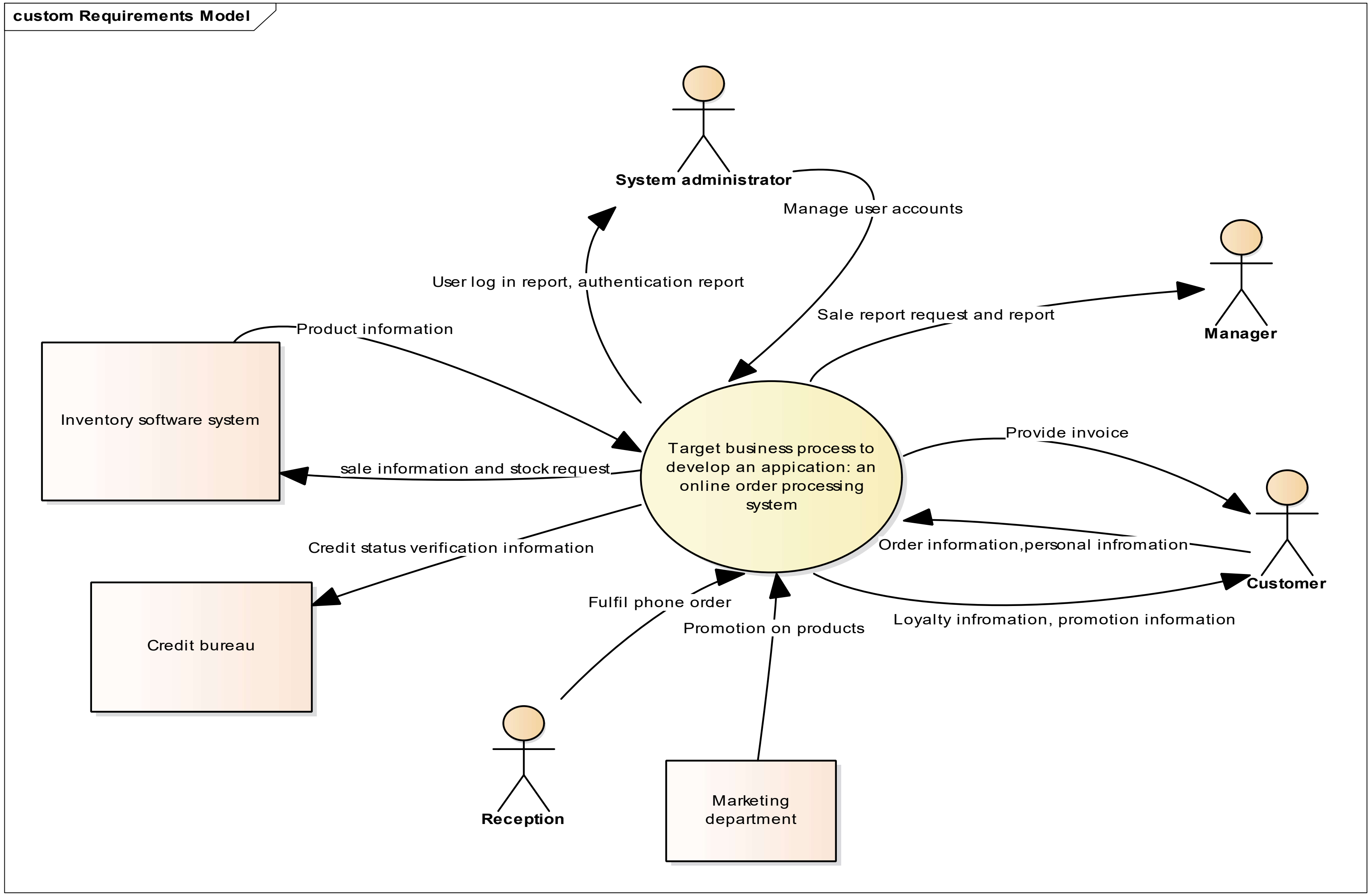
A **customer** is <u>supported</u> by only *one* **agent** and a agent can support *many* customers. Customers and agents are *uniquely identified* by their **ID** and have their **first and last names** along with **address**, **phone**. Customers must have **insure amount** and policy **expiry date** after which the policy has to be renewed.

DB design steps (2)

- Identifying Entities, Attributes, Relationships and Constrains

  - Context Diagram

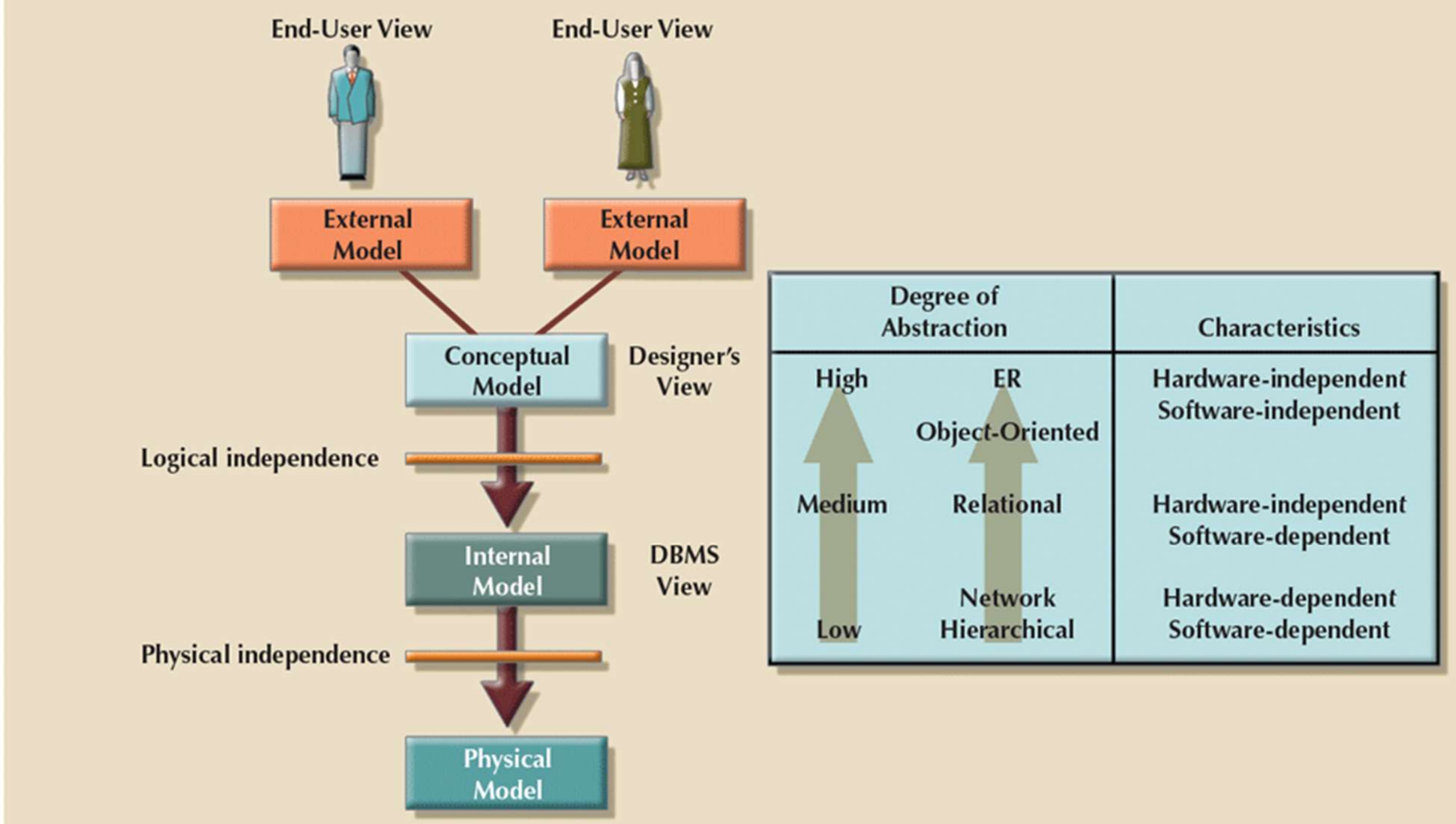
  External Entities, Attributes (input/output)


custom Requirements Model
System administrator
Manage user accounts
User log in report, authentication report
Manager
Sale report request and report
Product information
Inventory software system
sale information and stock request
Target business process to develop an appication: an online order processing system
Provide invoice
Customer
Order information,personal infromation
Credit status verification information
Credit bureau
Fulfil phone order
Promotion on products
Loyalty infromation, promotion information
Reception
Marketing department

Different views of design

- Based on degrees of data abstraction

FIGURE 2.6 DATA ABSTRACTION LEVELS



Degree of Abstraction		Characteristics
High	ER Object-Oriented	Hardware-independent Software-independent
Medium	Relational	Hardware-independent Software-dependent
Low	Network Hierarchical	Hardware-dependent Software-dependent

Conceptual Design

- **Goal**: design a database independent of database software and physical details
- **Conceptual data model:** describes main data entities, attributes, relationships, and some constraints
- Designed as software and hardware independent
- Covers/captures business rules and system requirements

E.g., Entity Relationship Diagram (ERD)

Logical Design

- **Goal**: design an enterprise-wide database that is based on a specific DBMS software but independent of physical-level details
- Conceptual model is mapped to the specific constructs (format) used by the selected DBMS
- **Logical Model**: Database Schema

  - Includes system implementation, e.g., data types, constraints

  - Some examples of DBMS

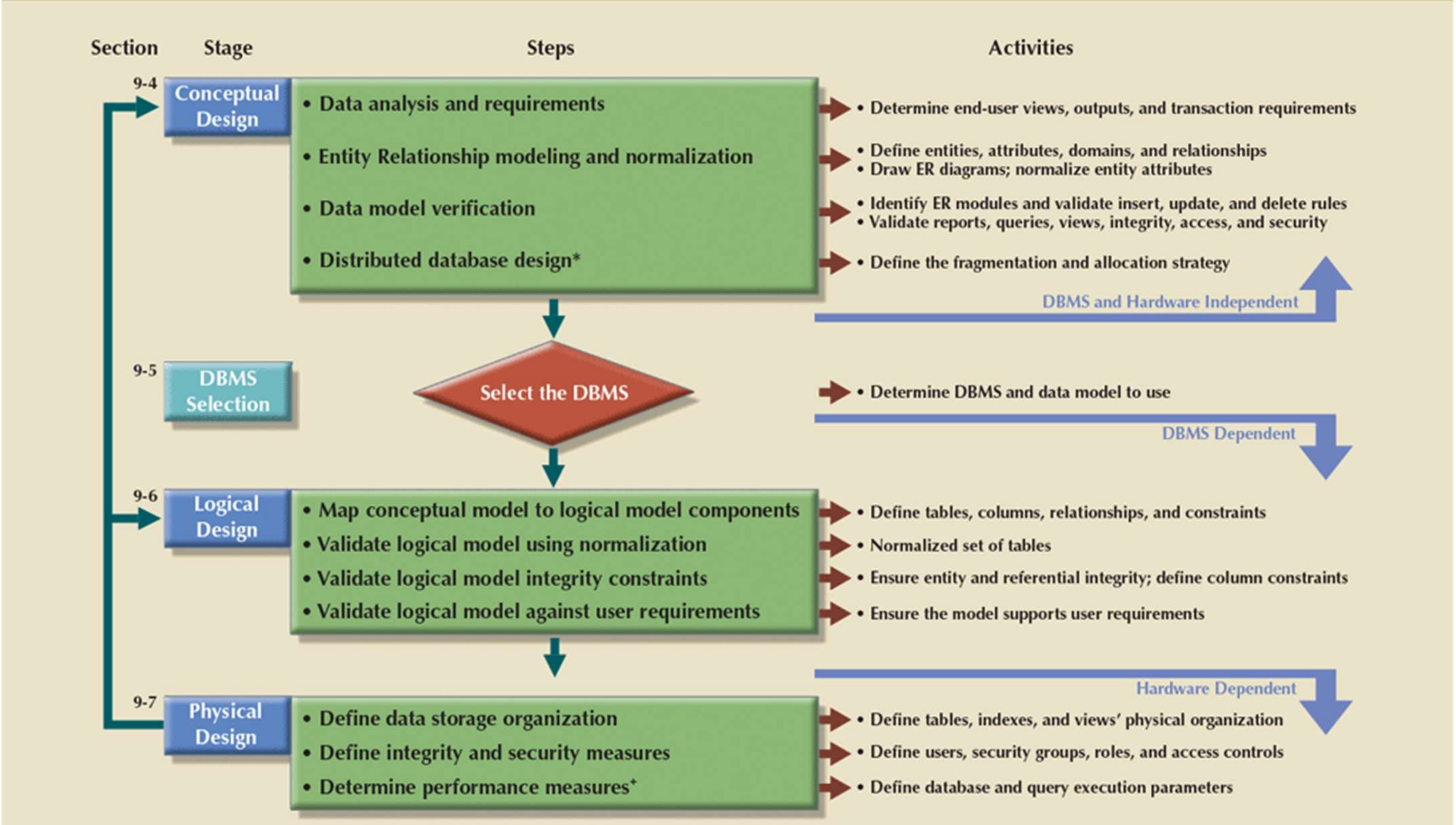
    - Oracle
    - MS SQL
    - **MySQL (used in this unit)**
    - PostgreSQL

Physical Design

- **Goal:** physical storage, organization and access of database; ensures integrity, security, and performance
  - Define data storage organization (allocate space for database etc.)
  - Define integrity and security measures (access right to the users based on their role)
  - Determine performance measures (Fine tuning the database and queries)

Database design

FIGURE 9.6 DATABASE DESIGN PROCESS



Section
Stage
Steps
Activities
9-4
Conceptual Design
• Data analysis and requirements
• Entity Relationship modeling and normalization
• Data model verification
• Distributed database design*
• Determine end-user views, outputs, and transaction requirements
• Define entities, attributes, domains, and relationships
• Draw ER diagrams; normalize entity attributes
• Identify ER modules and validate insert, update, and delete rules
• Validate reports, queries, views, integrity, access, and security
• Define the fragmentation and allocation strategy
DBMS and Hardware Independent
9-5
DBMS Selection
Select the DBMS
• Determine DBMS and data model to use
DBMS Dependent
9-6
Logical Design
• Map conceptual model to logical model components
• Validate logical model using normalization
• Validate logical model integrity constraints
• Validate logical model against user requirements
• Define tables, columns, relationships, and constraints
• Normalized set of tables
• Ensure entity and referential integrity; define column constraints
• Ensure the model supports user requirements
Hardware Dependent
9-7
Physical Design
• Define data storage organization
• Define integrity and security measures
• Determine performance measures⁺
• Define tables, indexes, and views' physical organization
• Define users, security groups, roles, and access controls
• Define database and query execution parameters

* See Chapter 12, Distributed Database Management Systems
⁺ See Chapter 11, Database Performance Tuning and Query Optimization

Keys

- **Key**: one or more attributes, such that the attribute's values uniquely identify each row

- How to identify a key?
  - Based on **determination or "functional dependence"**
  - If you know the value of attribute A, you can look up (**determine**) the value of attribute B, i.e.,

  A determines B **OR** $A \rightarrow B$ **OR** B is functionally dependent on A

- Example

  STU_NUM → STU_LNAME, STU_FNAME, STU_DOB

Keys (2)

- If an attribute determines all other attributes, that attribute is a **simple key**.
- If a combination of attributes determines all other attributes, that combination is a key, called a ***composite* key**
- **Super key:** composite key where at least one subset of attributes is a valid key
- Examples:
  - STU_NUM (Simple key)
  - STU_FNAME + STU_DOB + STU_PHONE (Composite key)
  - STU_NUM + STU_FNAME (Super key)

Keys (3)

- ***Candidate key***
  - Any simple key or minimal composite key
  - e.g. STU_NUM, STU_FNAME+STU_DOB+STU_PHONE
- ***Primary Key (PK)***
  - One of the candidate keys and chosen to be the unique row identifier
  - e.g. STU_NUM
- **Each table must have a Primary Key**

Keys (4)

- ***Natural key***
  - A general accepted identifier for real world objects
  - Familiar to users
  - e.g. Tax File Number, Medicare Number
    Credit Card Number & CVV & Expiry date
- ***Surrogate key***
  - A primary key where numeric values only distinguish entities, and are normally generated by the DBMS and auto-incremented
  - e.g. STU_NUM

NULL Values

- No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes
- A Null can represent
  - An unknown attribute value
  - A known, but missing, attribute value
  - A "not applicable" condition
- Can create problems in function usage and table linkage

You don't know the name of your practical supervisor

You know the name, but it's missing from the DB

Self-study students don't have a practical supervisor

Entity Integrity Constraint

- Condition in which each row in the table has its own unique identity
- All the values in the primary key column must be unique
- No attribute in the primary key can contain a null

Linking Tables Together

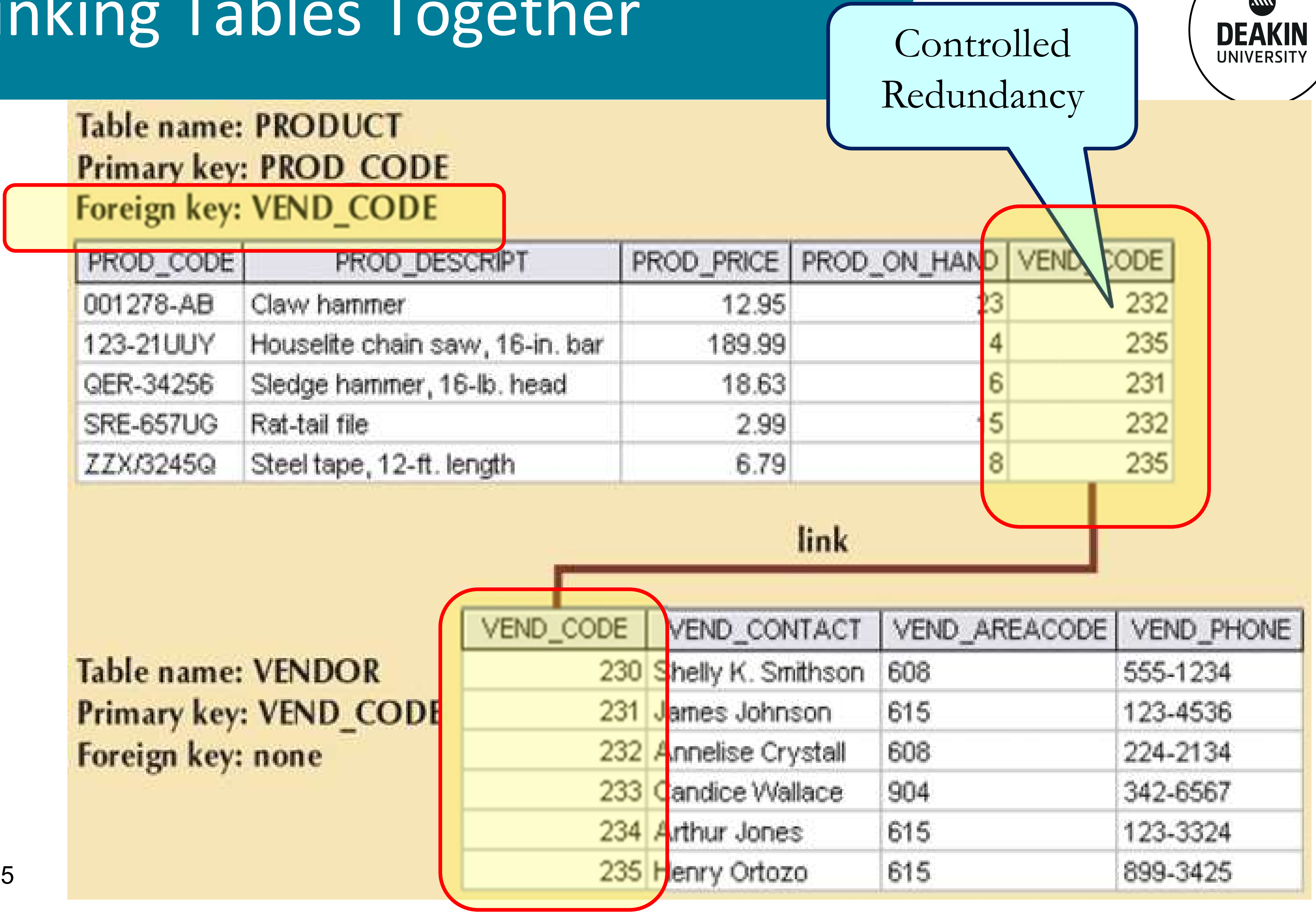


Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	5	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Controlled Redundancy

- **Redundancy** is unnecessary duplication of data
- **Controlled redundancy** makes a relational database work
- Tables that share common attributes enable us to **link tables** together

Referential Integrity Constraint

- **Foreign key (FK)**
  - Attribute(s) whose values match a candidate key (such as primary key) values in the related table
- **Referential integrity**
  - FK contains a value that refers to an existing valid row in another table

Foreign Key: Example

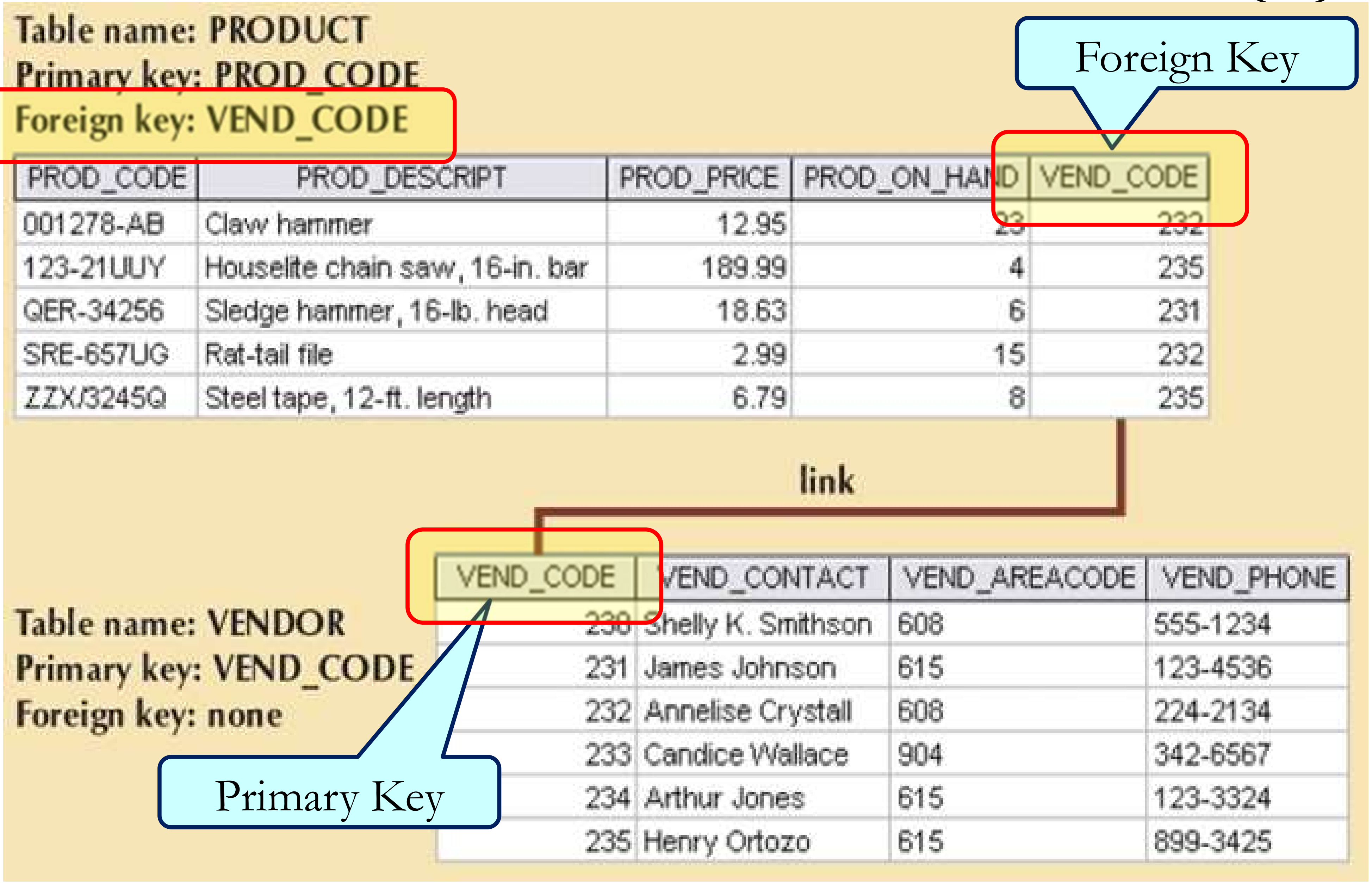

Foreign Key
link
Primary Key

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Integrity Rules

- Primary key
  - All PK entries are unique.
  - No part of a PK may be null.
- Foreign key may have:
  - An entry matching a PK value in the related table.
  - A null entry.

NOTE: Many RDBMSs enforce integrity rules automatically.

Summary of types of keys

Table 3.3	Relational Database Keys
Key Type	**Definition**
Superkey	An attribute or combination of attributes that uniquely identifies each row in a table.
Candidate key	A minimal (irreducible) superkey; a superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries.
Foreign key	An attribute or combination of attributes in one table whose values must either match the primary key in another table or be null.
Secondary key	An attribute or combination of attributes used strictly for data retrieval purposes.

Summary of Integrity rules

Integrity Rules	
Entity Integrity	**Description**
Requirement	All primary key entries are unique, and no part of a primary key can be null.
Purpose	Each row will have a unique identity, and foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null; in short, all invoices are uniquely identified by their invoice number.
Referential Integrity	**Description**
Requirement	A foreign key may have either a null entry, as long as it is not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related (every non-null foreign key value must reference an existing primary key value).
Purpose	It is possible for an attribute not to have a corresponding value, but it will be impossible to have an invalid entry; the enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

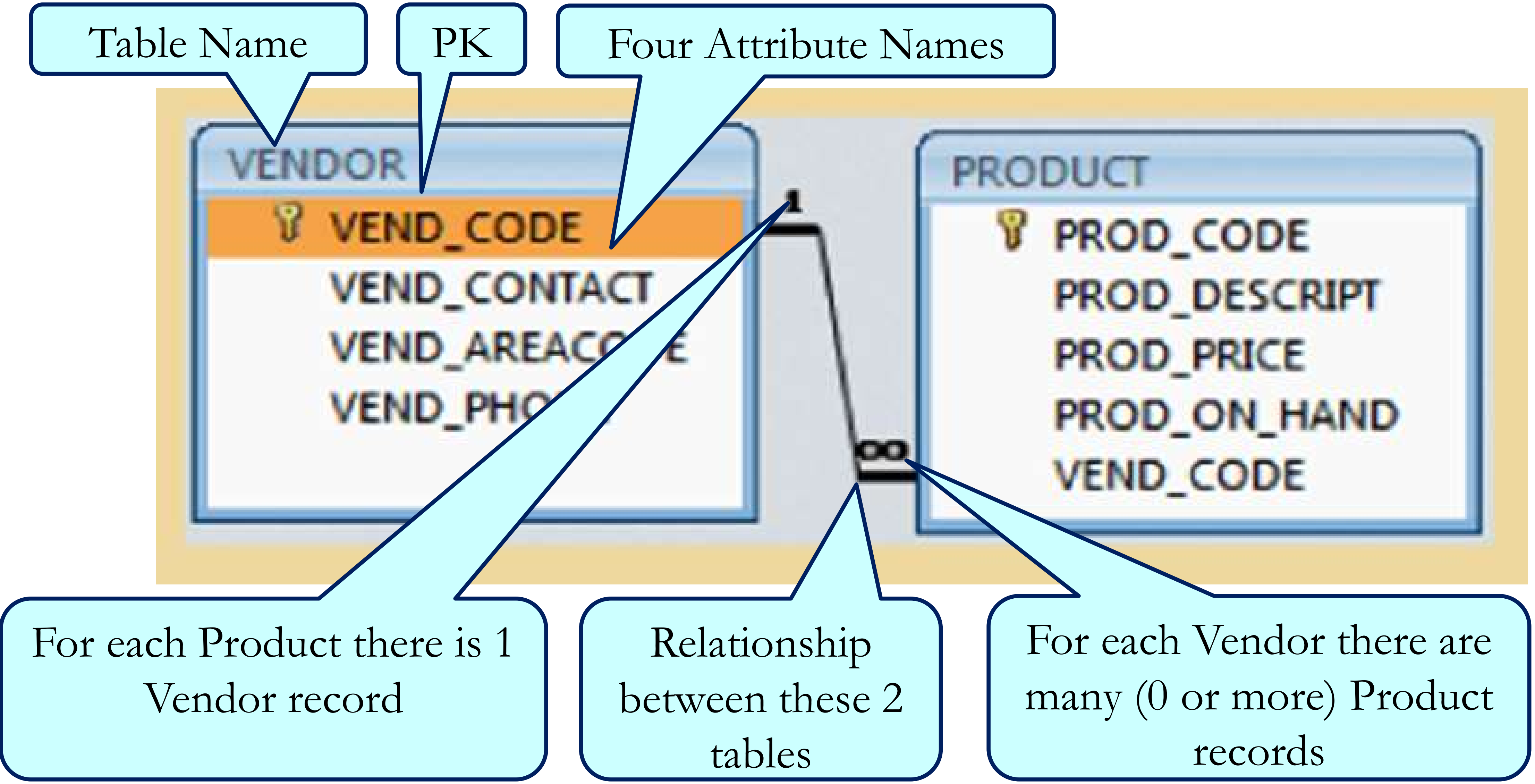
Entity Relationship Diagram (ERD)

- Graphical representation of entity relationship model

  - conceptual model

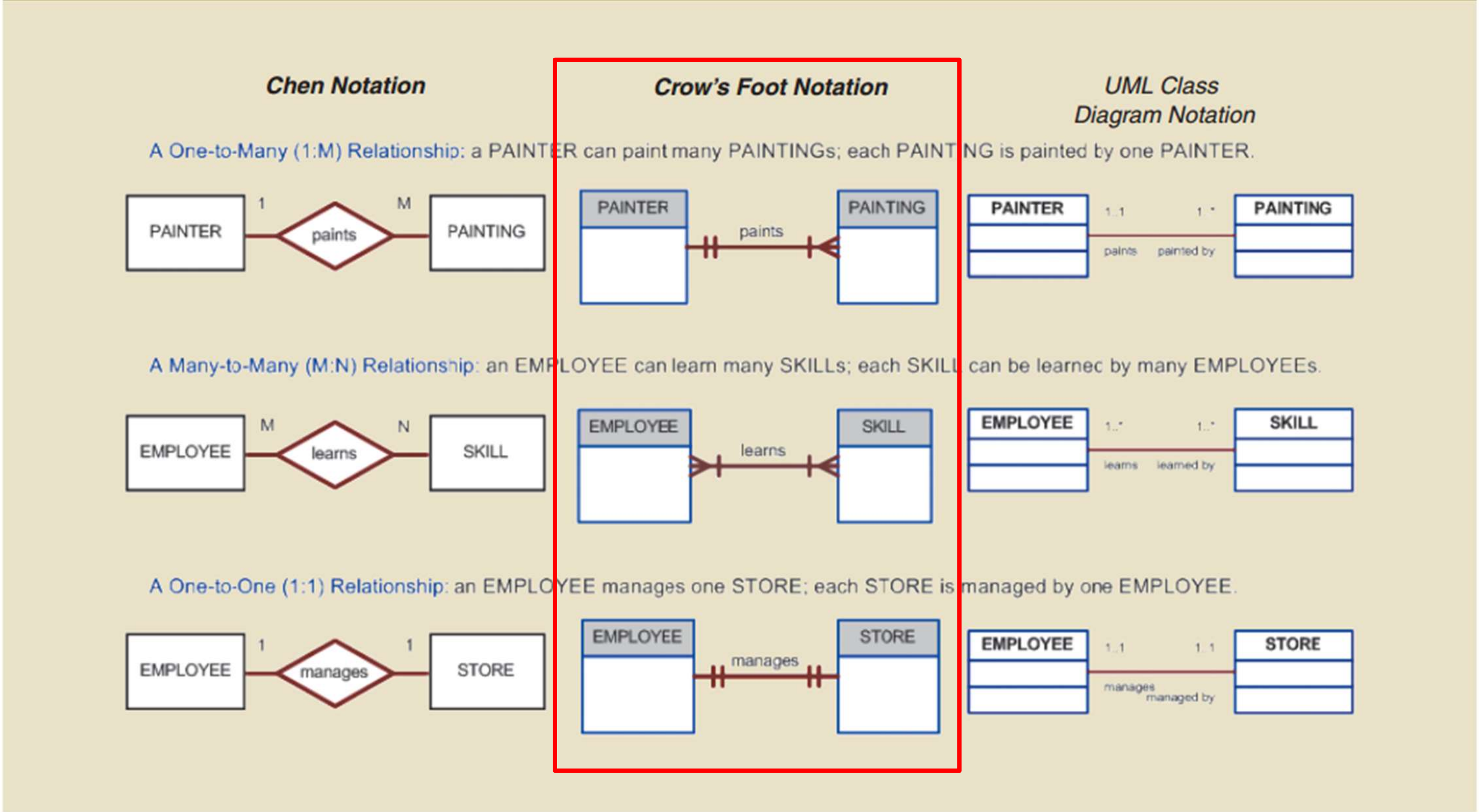
- Represents database structure in terms of **entities** (relation/table), **attributes** and **relationships** (primary and foreign keys)
- Visual modelling yields simplicity
- Very effective for communication
- Limited constraints representation (not all constraints are shown)

Relational Diagram



ERD Notations

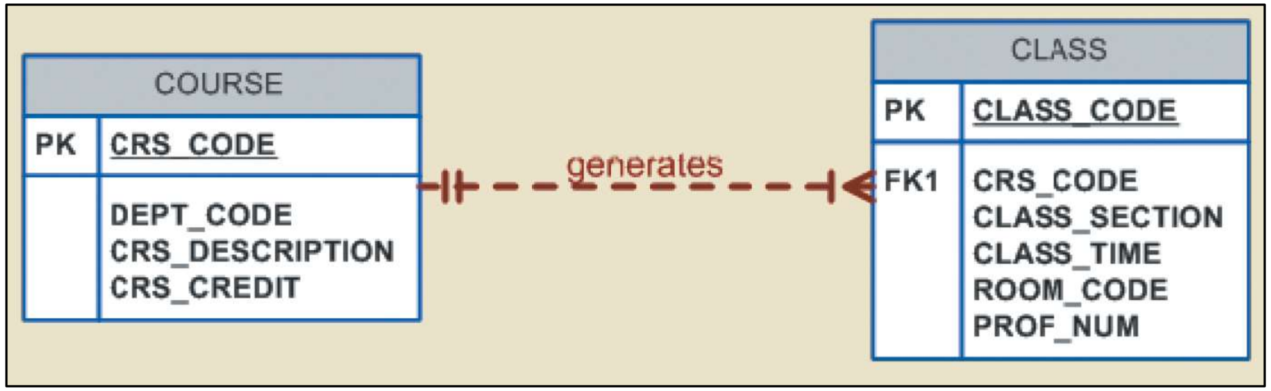
FIGURE 2.3 THE ER MODEL NOTATIONS



Chen Notation
Crow's Foot Notation
UML Class Diagram Notation
A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.
PAINTER
1
paints
M
PAINTING
PAINTER
paints
PAINTING
PAINTER
1..1
1..*
PAINTING
paints
painted by
A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.
EMPLOYEE
M
learns
N
SKILL
EMPLOYEE
learns
SKILL
EMPLOYEE
1..*
1..*
SKILL
learns
learned by
A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.
EMPLOYEE
1
manages
1
STORE
EMPLOYEE
manages
STORE
EMPLOYEE
1..1
1..1
STORE
manages
managed by

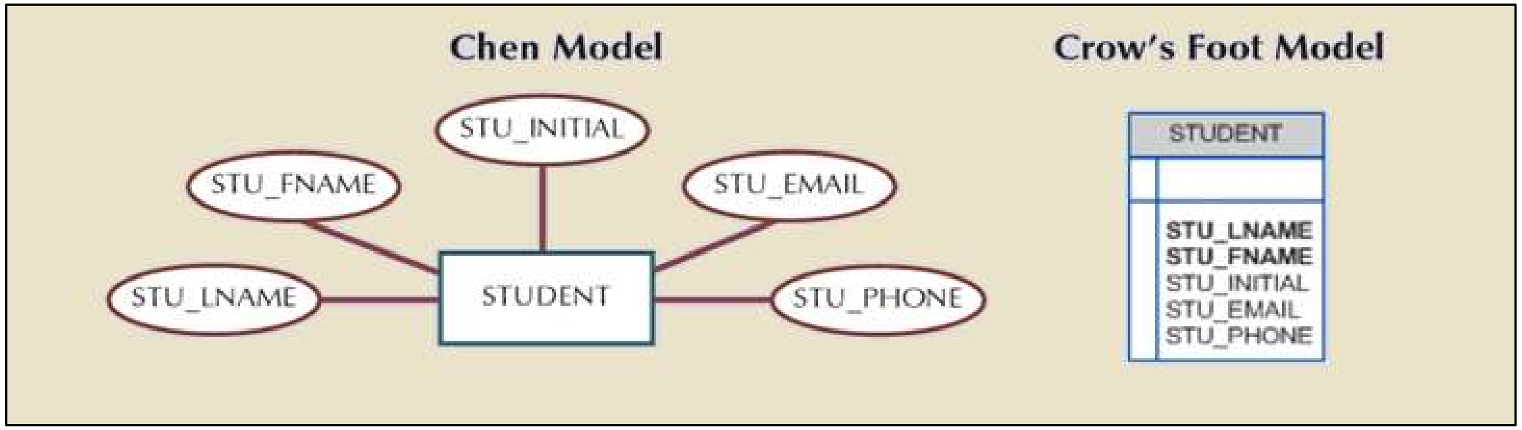
Entity relationships diagram (ERD): Entities in Crow-Foot notations

- An entity:
  - corresponds to a **table**, is represented by a **rectangle** containing the entity's name (e.g. COURSE, CLASS)
  - is usually written in capital letters, and is a **noun**
  - **PK, FK and attributes are written in the rectangle**



ERD: Attributes

- Attributes are **<u>characteristics</u>** of an entity
- ***Chen notation***: attributes are represented by ovals connected to entity rectangle with a line
  - Each oval contains the name of the attribute it represents
- ***Crow's Foot notation***: attributes written in attribute box below entity rectangle

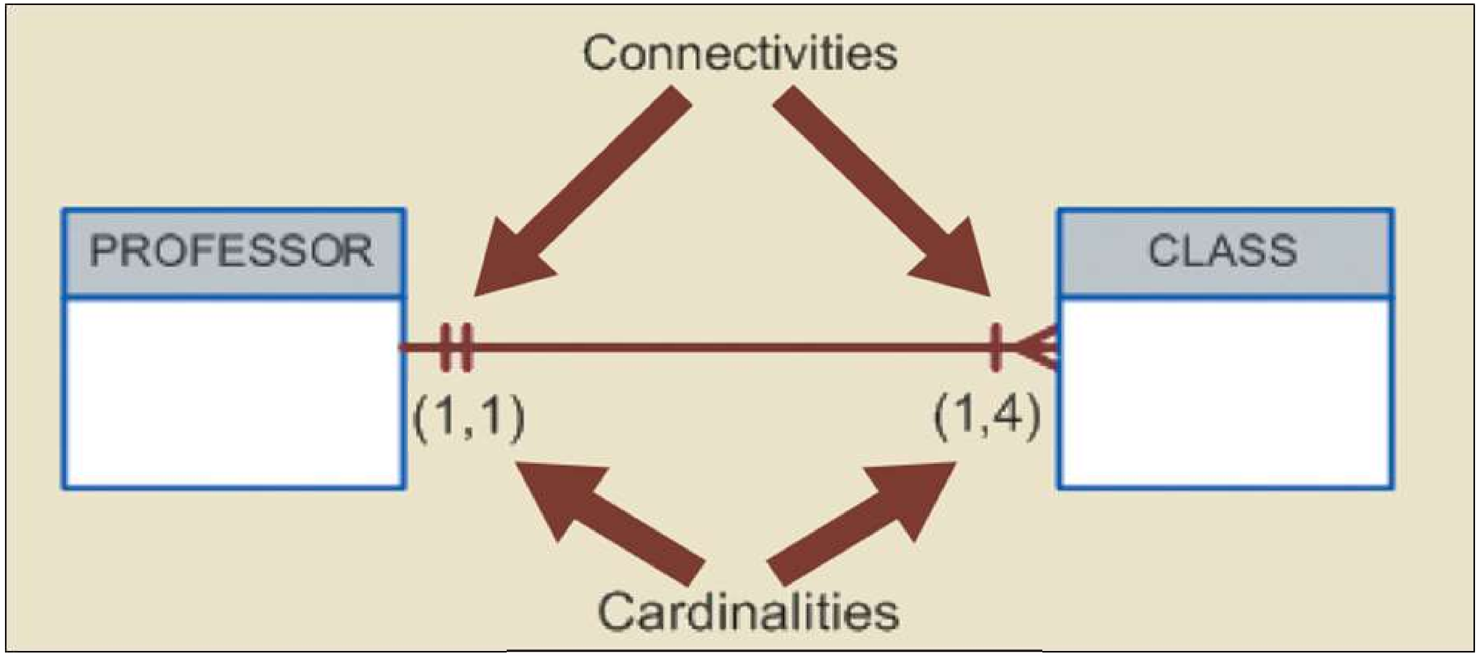


ERD: Relationships

- **Relationship**
  - – An association between 2 (or more) entities,
- **Connectivity**
  - – Describes the relationship <u>classification</u> i.e. 1:1, 1:M or M:N
  - – <u>Consider both directions</u> of the relationship
- **Cardinality**
  - – A property that assigns a specific value to connectivity
  - – Denotes a specific range (minimum, maximum) of entity occurrences associated with one occurrence of the related entity

ERD: Connectivity and Cardinality

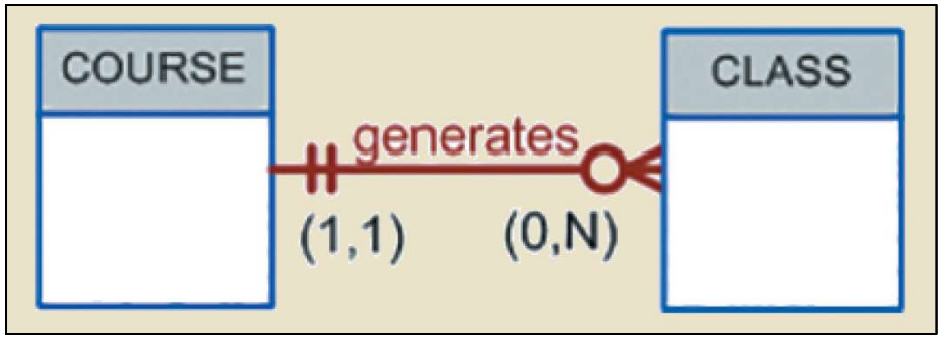
- A class is taught by a professor at a given time in a given place. (cardinality at professor side 1:1)
- A professor teaches at least one class and up to four classes.
- Here, professor and class has (one to many) relationship ; Connectivity

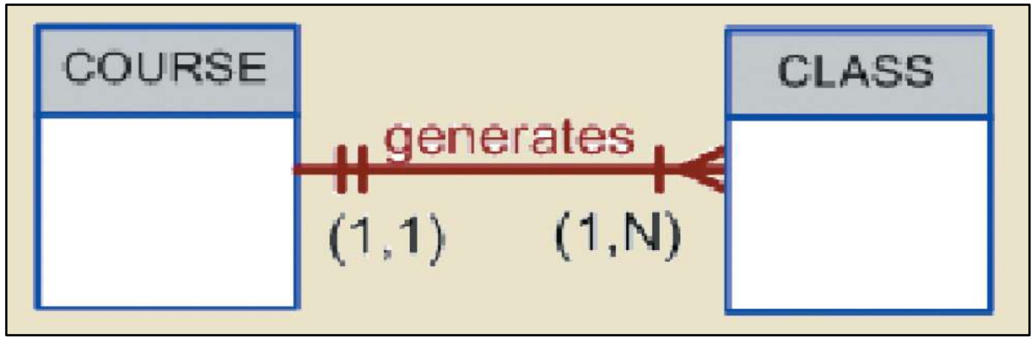


- (minimum, maximum)

ERD: Relationship Participation

- Participation in an entity relationship is either optional or mandatory
- **<u>Optional</u>**:
  - One entity occurrence **<u>does not require</u>** a corresponding entity occurrence in a particular relationship
- **<u>Mandatory</u>**:
  - One entity occurrence **<u>requires</u>** a corresponding entity occurrence in a particular relationship



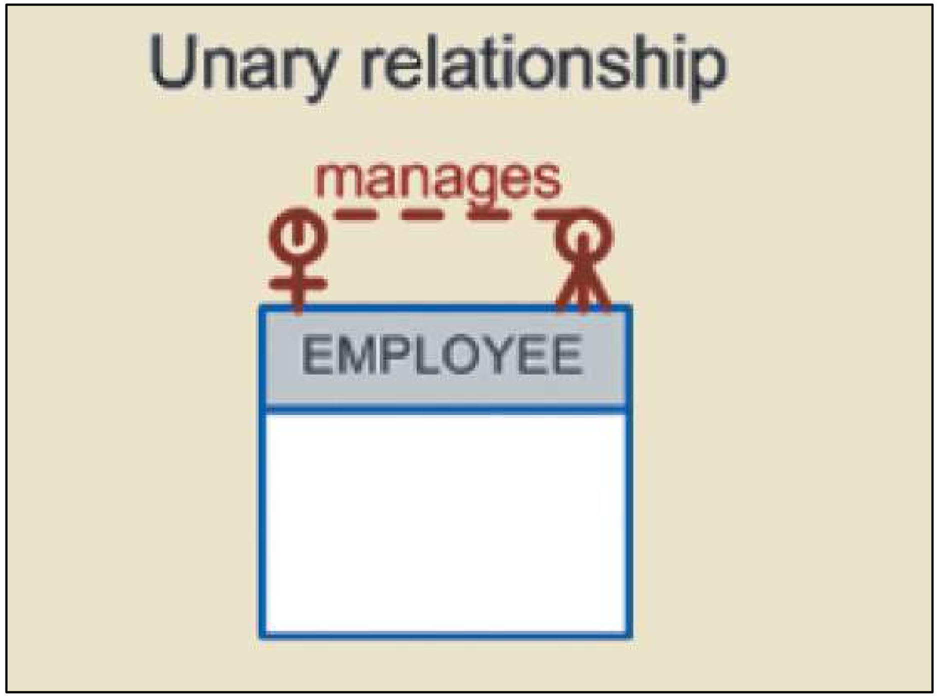


ERD: Relationship Degree

- The degree of a relationship is the **number of entities** associated with the relationship
  - **Unary** relationship (1 entity)
  - **Binary** relationship (2 entities)
  - **Ternary** relationship (3 entities)

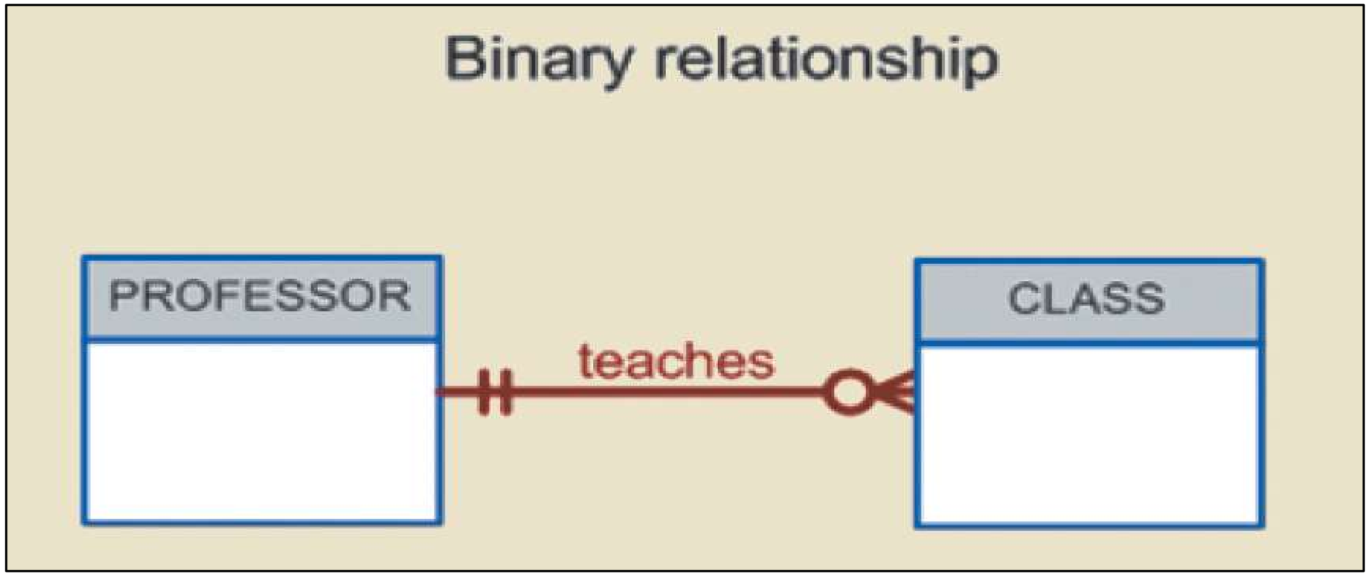
Unary Relationship

- EMPLOYEE requires another EMPLOYEE to be the **manager** – that is, EMPLOYEE has a **relationship with itself**.
- Such a relationship is known as a **recursive relationship**.



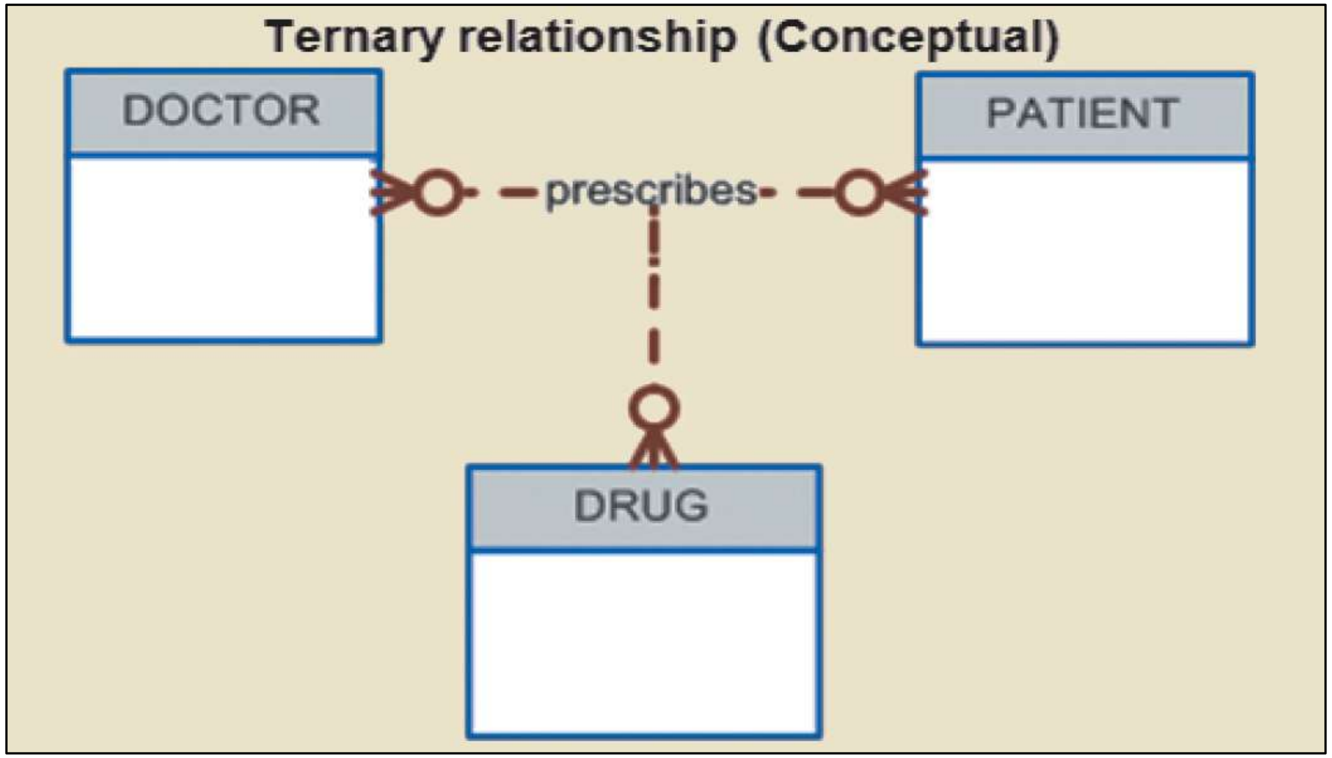
Binary Relationship

- Two entities are associated in a relationship.
- the most common type of relationship.
- “A PROFESSOR teaches one or more CLASSes” represents a binary relationship.



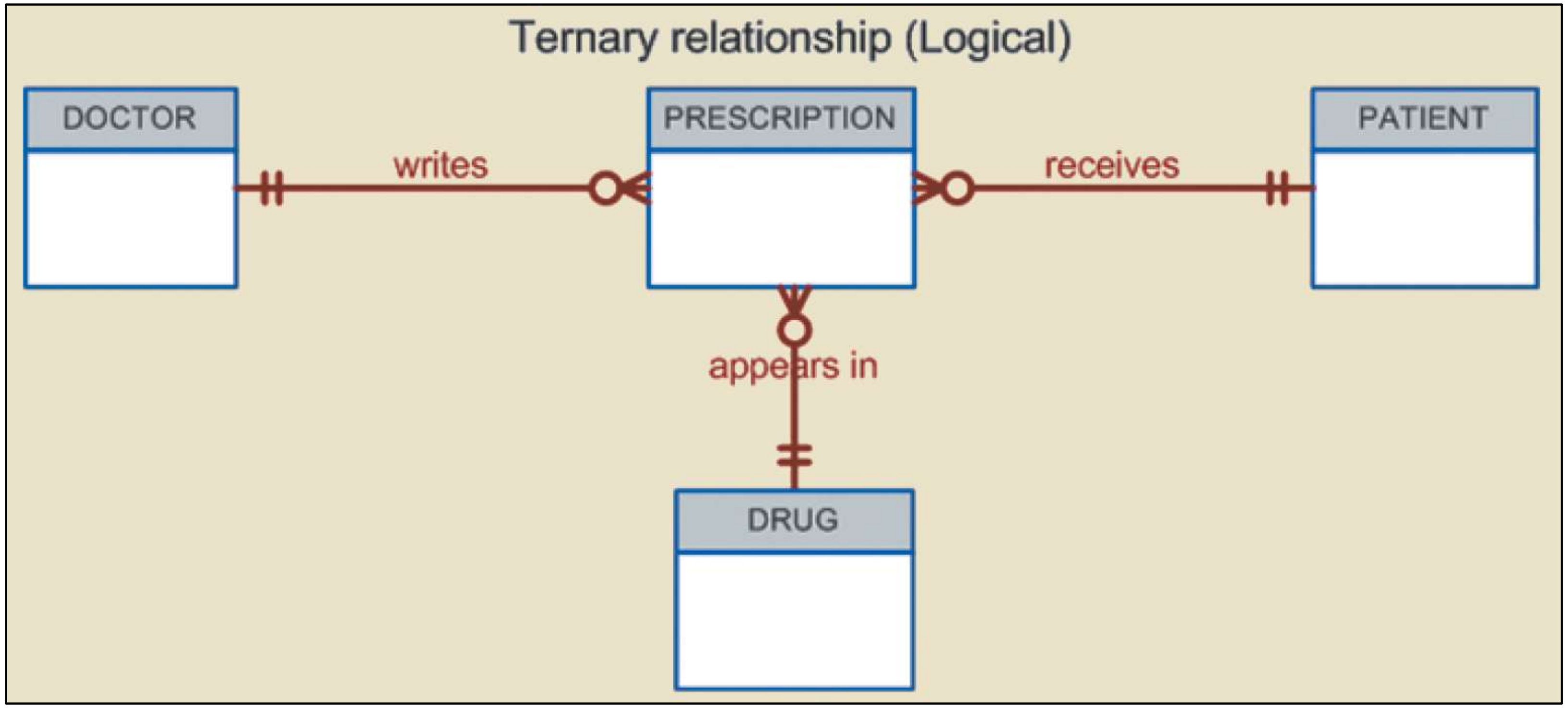
Ternary Relationship

- An association among three different entities.
- “A DOCTOR prescribes a DRUG for a PATIENT”. The relationship here is conceptual, requires more rules to make it implementable.



Ternary Relationship (1)

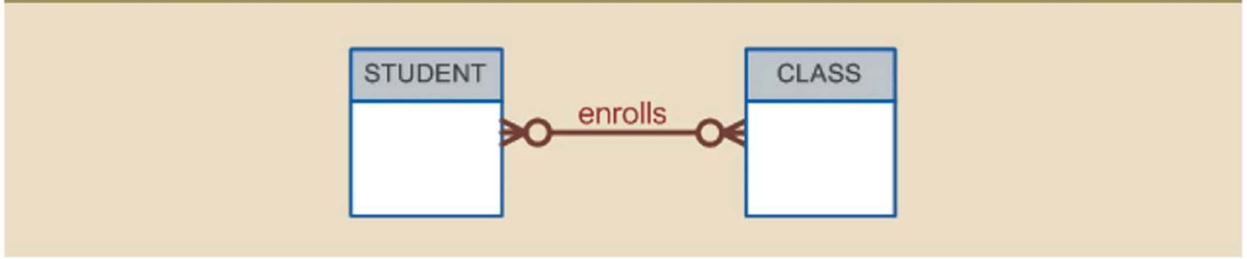
- A DOCTOR writes one or more PRESCRIPTIONs.
- A PATIENT may receive one or more PRESCRIPTIONs.
- A DRUG may appear in one or more PRESCRIPTIONs.



Associative (Composite) Entities

- Used to implement an M:N relationship between two or more entities (Prescription in the previous slide)
- Composed of the primary key attributes of each parent entity
- May also contain additional attributes that play no role in connective process (e.g., grade)

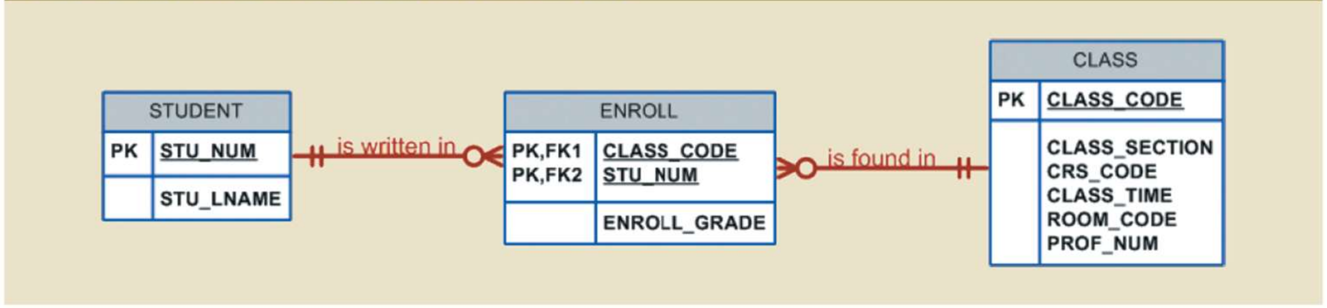
FIGURE 4.24 THE M:N RELATIONSHIP BETWEEN STUDENT AND CLASS



Business case

Implemented case

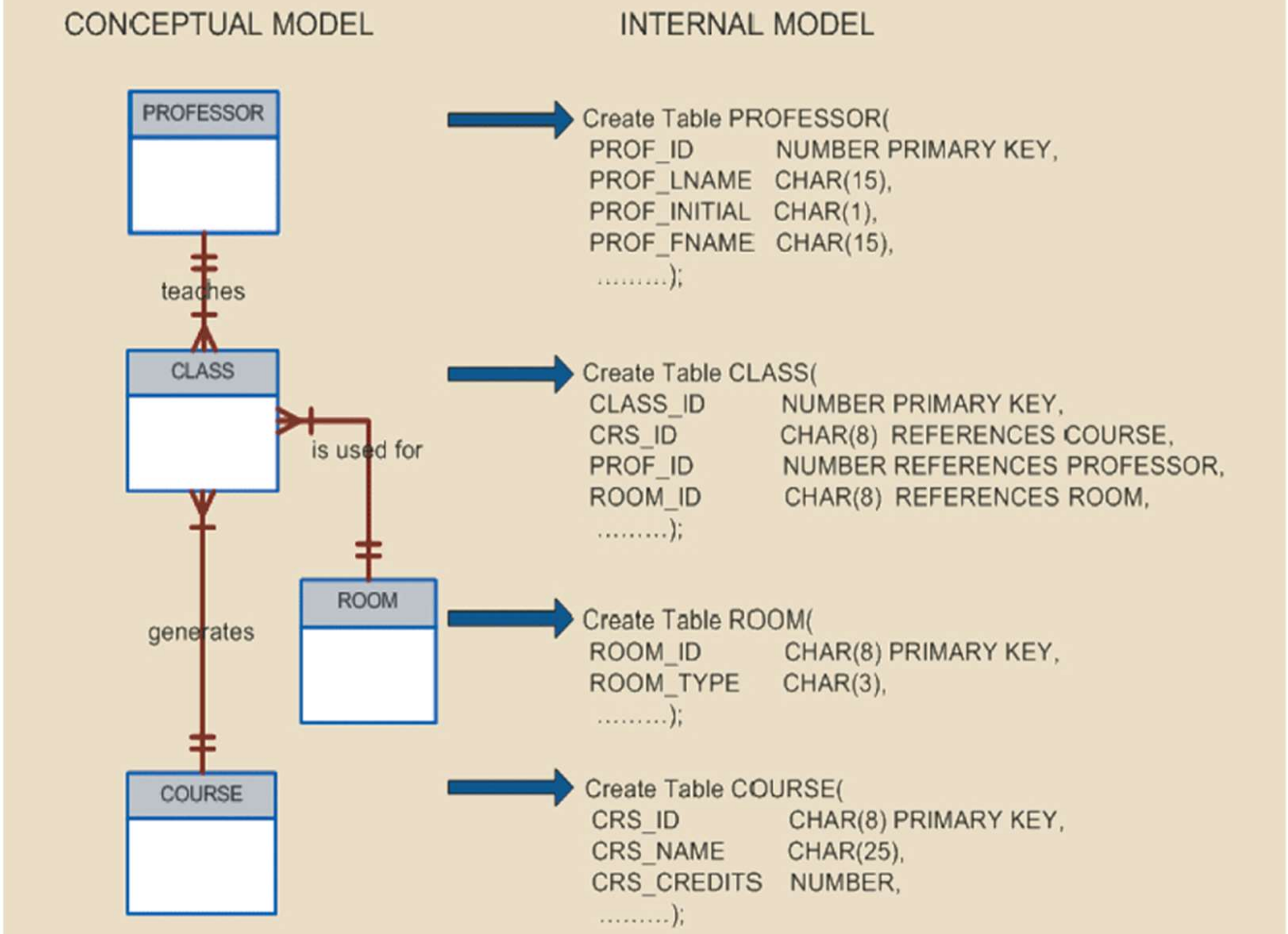
FIGURE 4.25 A COMPOSITE ENTITY IN AN ERD



Database schema

Conceptual relational diagram to Logical database schema

FIGURE 2.9 INTERNAL MODEL FOR TINY COLLEGE



```
Create Table PROFESSOR(
 PROF_ID        NUMBER PRIMARY KEY,
 PROF_LNAME     CHAR(15),
 PROF_INITIAL   CHAR(1),
 PROF_FNAME     CHAR(15),
 .........);

Create Table CLASS(
 CLASS_ID       NUMBER PRIMARY KEY,
 CRS_ID         CHAR(8)  REFERENCES COURSE,
 PROF_ID        NUMBER REFERENCES PROFESSOR,
 ROOM_ID        CHAR(8)  REFERENCES ROOM,
 .........);

Create Table ROOM(
 ROOM_ID        CHAR(8) PRIMARY KEY,
 ROOM_TYPE      CHAR(3),
 .........);

Create Table COURSE(
 CRS_ID         CHAR(8) PRIMARY KEY,
 CRS_NAME       CHAR(25),
 CRS_CREDITS    NUMBER,
 .........);
```

Summary

- Database design – Conceptual and logical design
- Relational model – Entity, Attribute, Relationships, and constraints
- Keys (composite, super, candidate, primary, natural, surrogate, foreign, secondary)
- Integrity Rules – Entity and referential integrity
- Entity Relationship Diagram
- Associative Entities

This Week's OnTrack Tasks

- 2.1P Database Modelling Tools

  - Basics of relational database modelling

  - Modelling tools – Lucid Chart and MS Visio

- Please check the task sheet and start working on it

Next Week

- More on relational model and ERD

Thank you

See you next week

Any questions/comments?

Readings and References:

- Chapter 2-4, Chapter 9

Database Systems : Design, Implementation, & Management 13TH EDITION, by Carlos Coronel, Steven Morris