CST3604 Lecture Notes

Physical Database Design

Database Design – Review of Basic Database Design Principles

(Part 1 of 2)

(Lecture Notes 1A)

Prof. Abel Angel Rodriguez

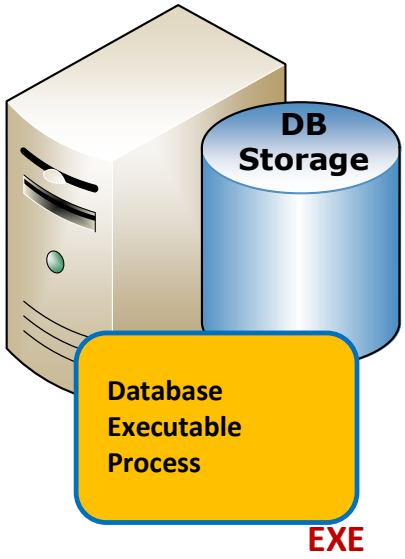
Chapter 1 Review of Database Design Concepts from CST3504

1.1 Review of Basic Concepts

1.1.1 Overview the Basics

Concept of Database Management System (DBMS) in Computer Science

- Physically:
  - A database is a **SOFTWARE** that allows you to organized, store data in an electronic format in a computer system.
  - The data stored in the database can be searched, manipulated, sorted, analyzed and displayed quickly and efficiently
  - Can be of any size and complexity.
  - A database can be looked at as a ***computerized*** record-keeping system



- The users of a database are given the tools to perform a variety of operations on the storage, such as:
  - Inserting, retrieving and updating records into the database
  - Deleting a record from the database
  - Removing or deleting the database tables or files

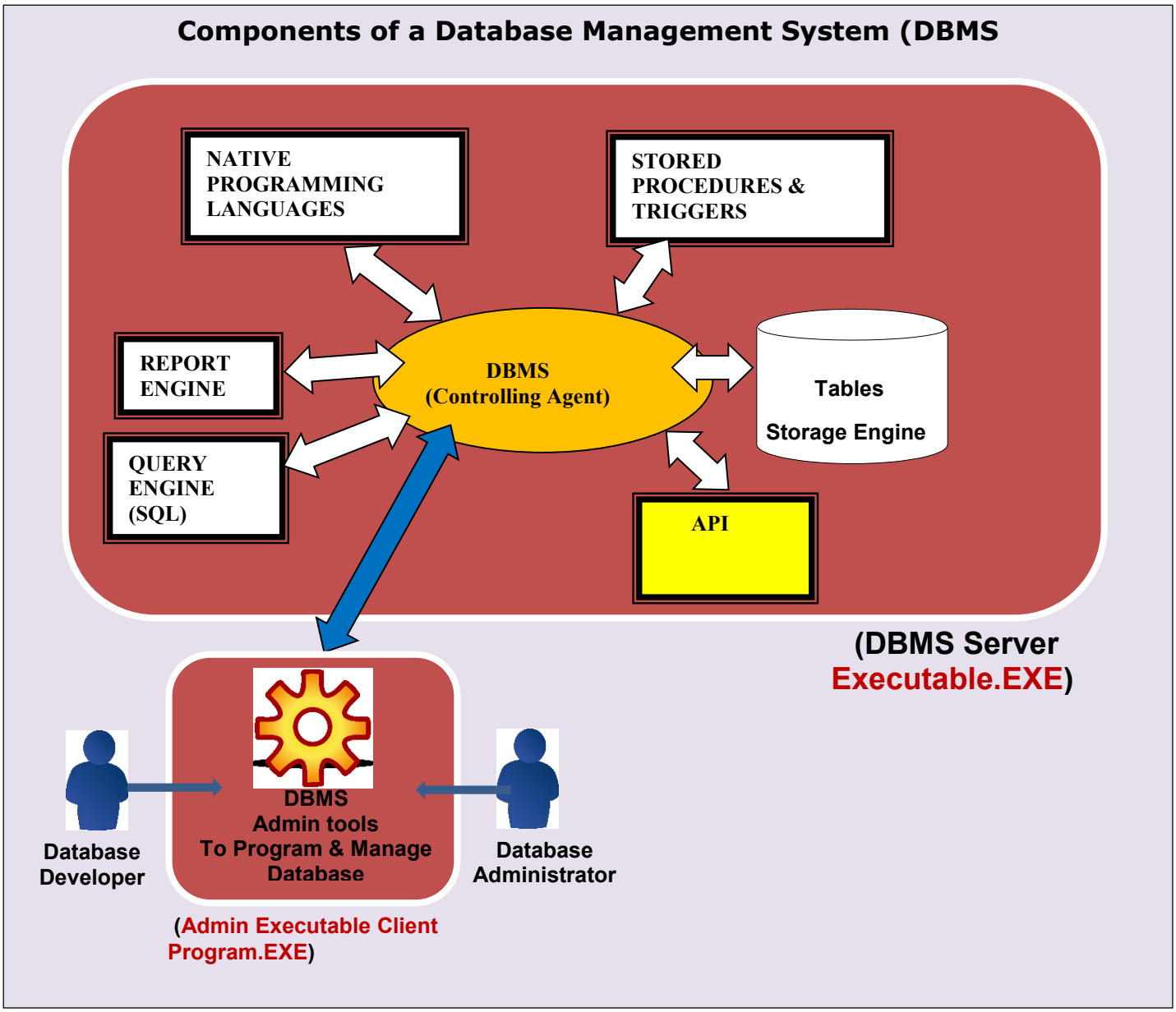
- Modern database programs for business and enterprises are called Data Management Systems (DBMS)
- Examples of current databases are:
  - Business databases which contain information on employees, payrolls, customers, products, inventories, and sales.
  - Databases in schools contain information on courses, students, teachers, and grades.
  - Banks are big users of databases; they are used to store customer information, & all transaction records etc.
  - The telephone companies store telephone account records, transactions & invoices etc.
  - In the new age of the Internet, Web Servers use databases to store e-commerce transactions, products, search engines directory information, etc. This has lead to new storage technology such as Data Warehousing etc.

DBMS Components

- DBMS Characteristics:
  - A DBMS is primarily composed of the following basic components:

Item	Component	Description
1.	**Forms???**	▪ User-Interface Module or Forms to create front-end screens for users to interact with DBMS ▪ IMPORTANT! Most modern DBMS no longer contain this component from my understanding.
2.	**Storage System (Tables)**	▪ Storage System where data is kept ▪ Relations DBMS use tables to store data
3.	**Queries (SQL)**	▪ Engine to enable the querying (searching, updating, inserting & deleting) the data using SQL Language.
4.	**Reporting Services**	▪ Engine to enable the creation/generation of reports or formatted result of database queries and contains useful data for decision-making and analysis.
5.	**Custom(native-code) programming language**	▪ Each DBMS comes with specialized programming language in addition to SQL to write complex Server-side processing (Work done on the DATABASE SERVER and not on CLIENT applicaiton
6.	**Programming Tool: Stored Procedures**	▪ Function-like executable BLOCKS of one or more SQL statement & Native-Code grouped together as one EXECUTABLE UNIT! These EXECUTABLE blocks can be called from CLIENT APPLICATIONS to perform the desired processing/query on the database
7.	**Programming Tools: Triggers**	▪ A special type of **Stored Procedure** that **EXECUTES AUTOMATICALLY** under some specified condition (Similar to Event-Handlers in programming languages)
8.	**Administrative Tools: Database Development Interface/Programs**	▪ User-Interface Programs that allows the ***Database Developer*** program, create databases & execute queries agains the database. ▪ **IMPORTANT!** This is a program or **client executable** that allows user/developer to interface and program the DBMS
9.	**Administrative Tools: Database Administration Interface/program**	▪ User-Interface Program that allows the ***Database Administrator*** manage the DBMS network administration, security, backup & Recovery, etc. ▪ **IMPORTANT!** This is a program or **client executable** that allows user/system administrator to interface and manage the DBMS
10.	**Administrative Tools: Other**	▪ Other tools
11.	**Application Programming Interface (API)**	▪ A set of routines, protocols, and tools for integration by external programs into the existing Database Management Software application. ▪ Opens the door to all kinds or integration into the DBMS by other applications etc.

- Illustration of components of a typical DBMS:



- Examples of Database Management System (DBMS) Servers/Programs are:

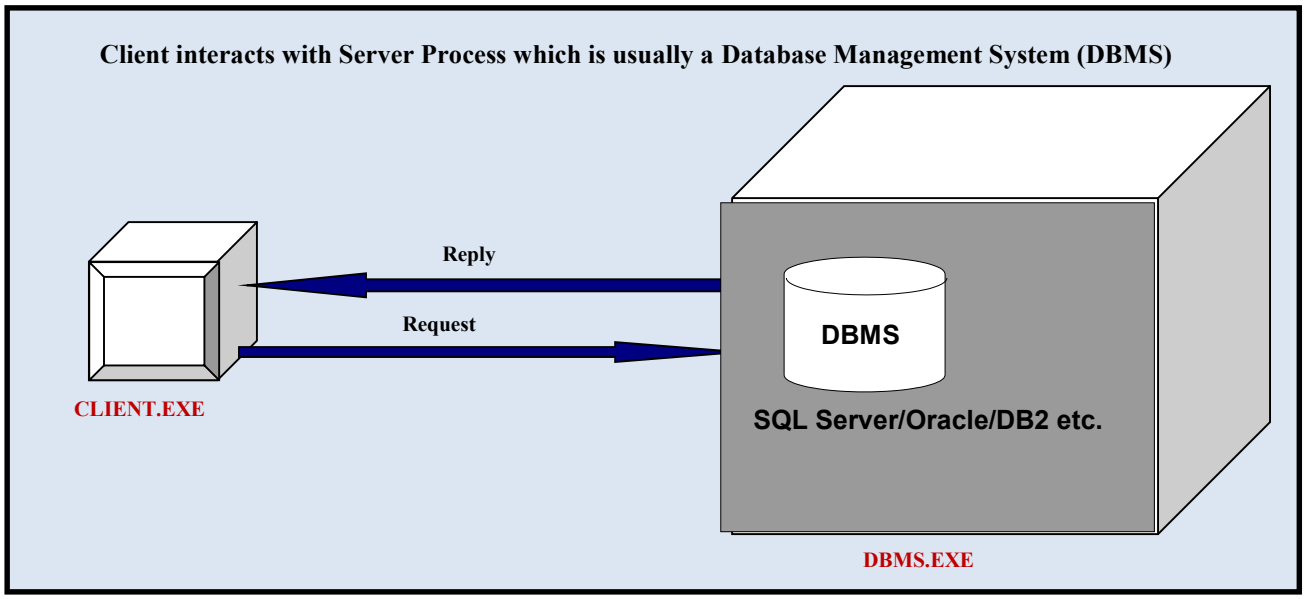
  - **Oracle**
  - **Microsoft SQL Server**
  - **Sybase SQL Server**
  - **MySQL**
  - **IBM DB2**

    - **Note that Microsoft Access is a Personal or Desktop Database, not true DBMS**

Database Application – Typical Client/Server Architecture (Client process & DBMS)

The Architecture

- Illustration of a Database Application architecture:
  - Typically a Client Executable created in Java, C#, other, interacting with a Database Management System (DBMS)



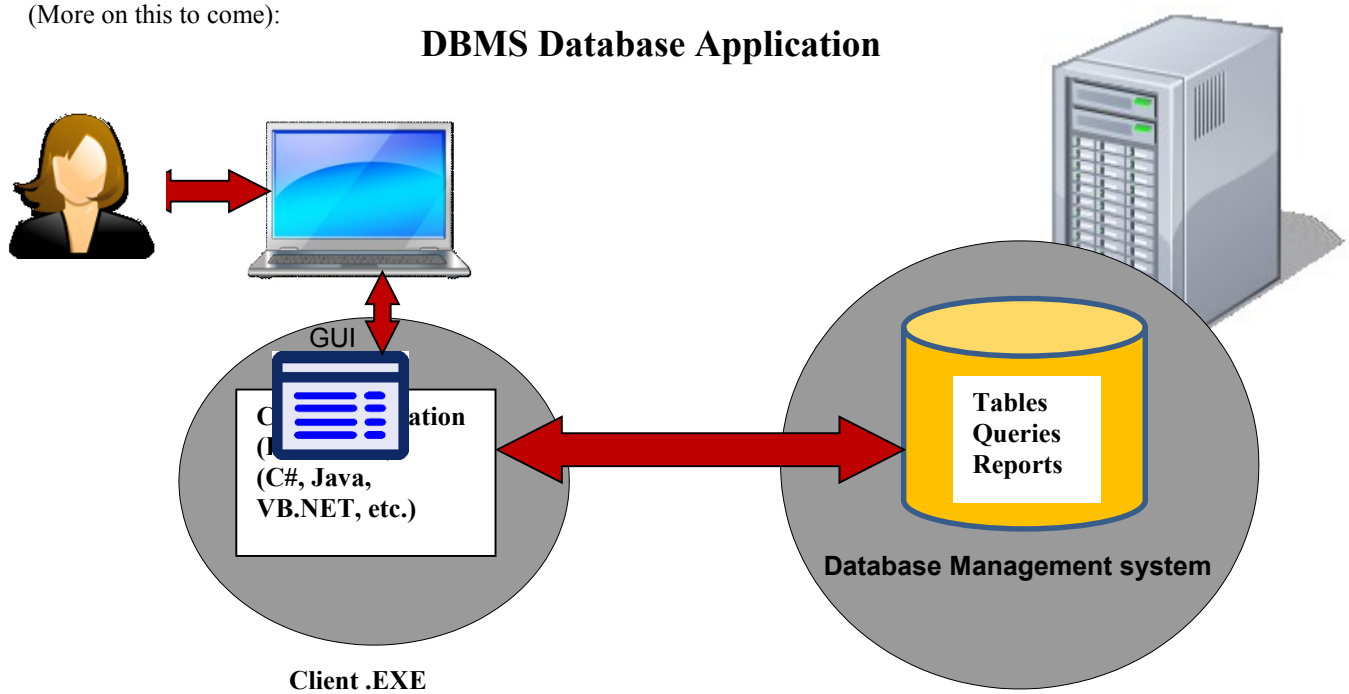
Database Application - Client/Server Application Architecture

  - **IMPORTANT!!!** - Note that there is TWO EXECUTABLE: **A CLIENT.EXE** and a **DATABASE_SERVER.EXE**.

Database Application

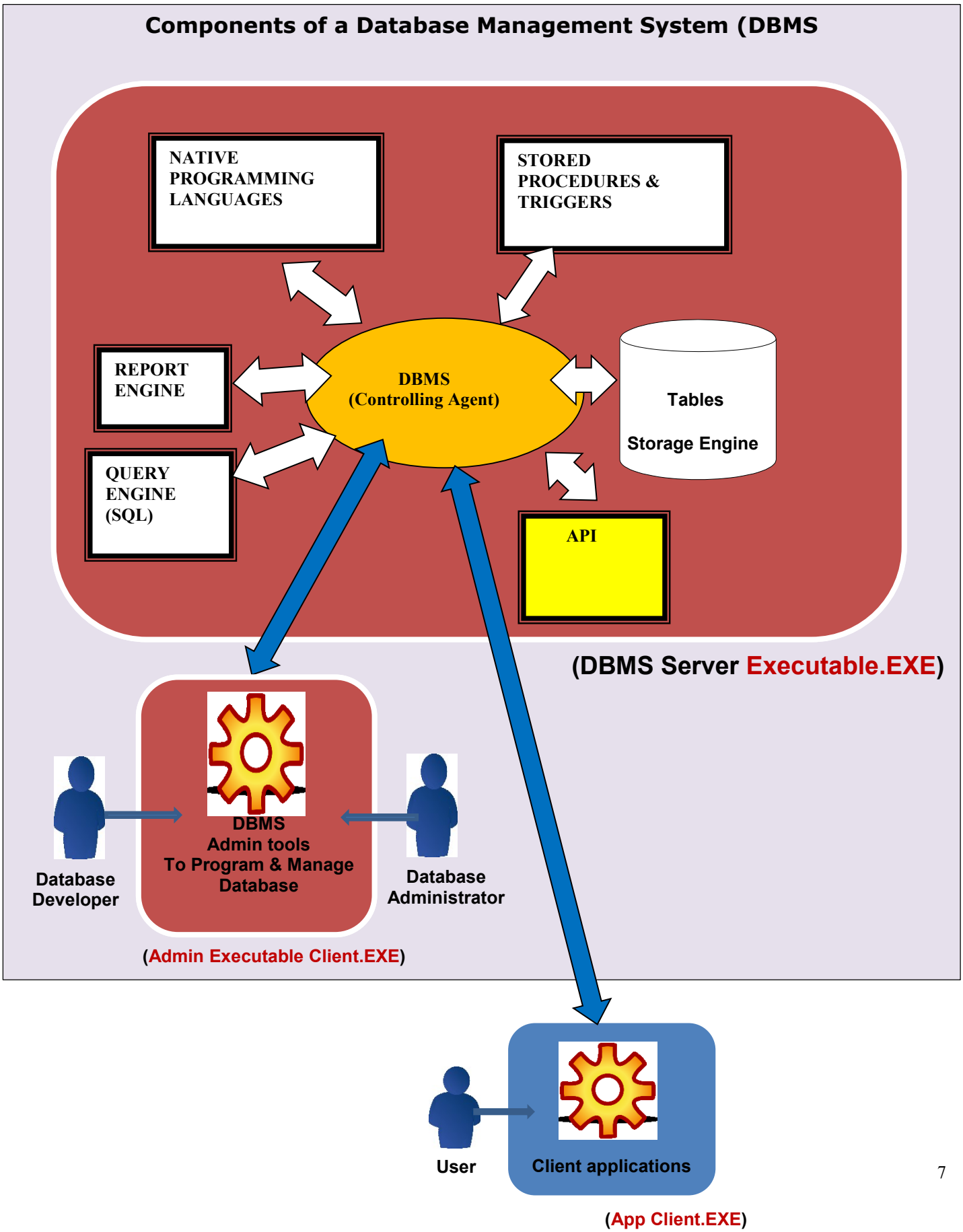
- The most common and ***Business Database Applications*** are those that **CONNECT** to a **Database Management System (DBMS)**.
- ***Database Applications*** are:

  - The application **BUSINESS LOGIC & PROCESSING** is created in languages such as **VB.NET**, **C#**, **JAVA** etc., and **CONNECTS** to ***Database Management System*** (DBMS) like ***Oracle***, ***SQL Servers***, ***MySQL*** etc., in order to retrieve, update, insert and delete data.
  - This is known as a **TWO-TIER CLIENT/SERVER TOPOLOGY** becase there are **TWO EXECUTABLES** involved (More on this to come):



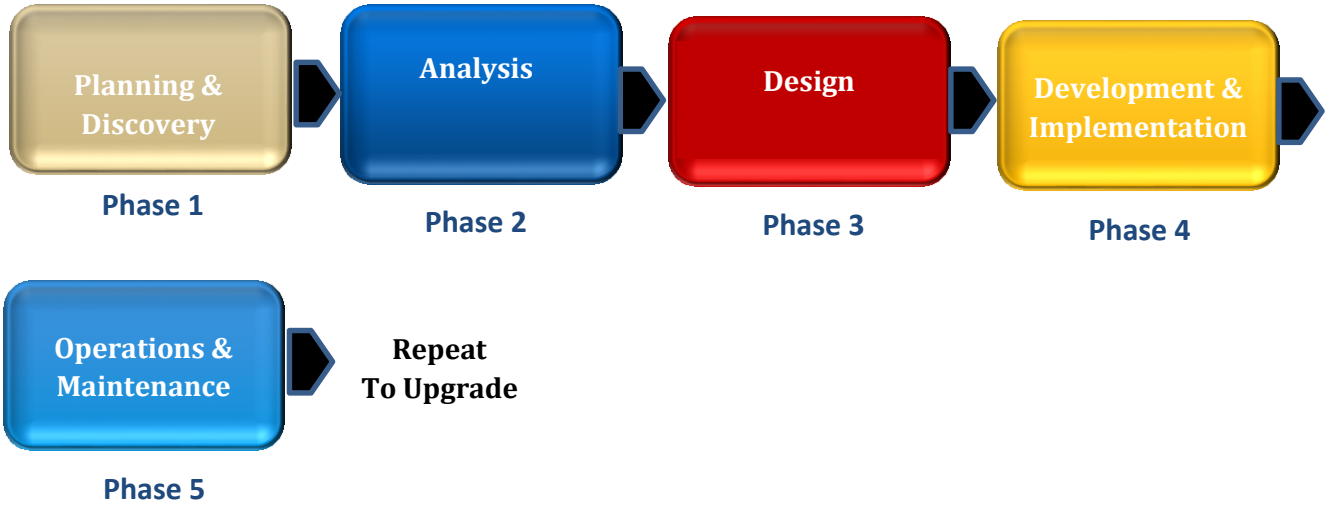
The Big Picture

- Detailed illustration of a Database Application:



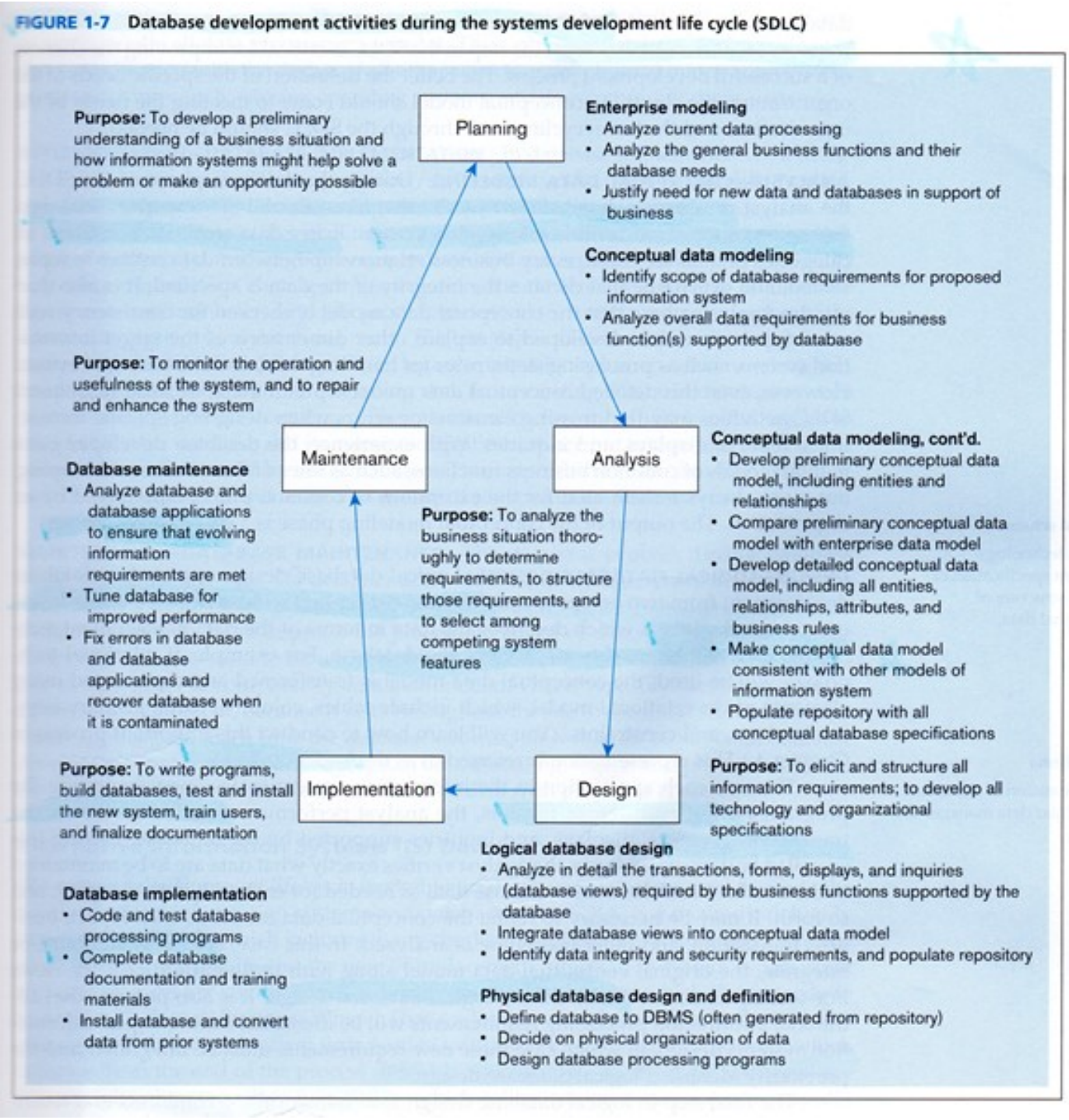
1.1.2 (The HOW) The System Development Life Cycle (Methodology)

- Below is the high-level methodology or the steps required to develop and implement an INFORMATION SYSTEM PROJECT using the System Development Life Cycle (SDLC) approach:



- Figure below are the detailed tasks as shown in your text book of the System Development Life Cycle (SDLC) approach:

Figure 1 – SDLC details as shown in the class textbook



- Table below are **Professor Rodriguez's** the System Development Life Cycle (SDLC) detailed tasks & Project Management tasks to give you a practical real-world approach:

Table 2 – Detailed Tasks by Project Phases

Phases	Description
Phase 1: **Planning & Discovery** 	▪ ***Purpose:*** - *Develop the plan, understand the business and discover existing information systems.* - *Interview, discover, etc.* ▪ ***Database Professional Role:*** - *Database/Systems Analyst* - *Business Analyst* ▪ ***Deliverables:*** 1. ***Primary Database Deliverable – Enterprise Data Model*** *(ER Model) if required or available* 2. *Project plan, methodology or project life-cycle & Documentation*
Phase 2: **Analysis** 	▪ ***Purpose:*** - *Analyze/derive detailed user requirements for data & develop data model to represent requirements.* - *Create a detailed* ***Conceptual Data Model*** ***(Entity-Relational Model - ER Diagram*** *or Enhanced E-R Diagram****)*** ▪ ***Database Professional Role:*** - *Database/Systems Analyst* ▪ ***Deliverables:*** 1. ***Primary Database Deliverable – Detailed Conceptual Data Model:*** ○ *E-R Diagram or Enhanced E-R Diagram*

Phases	Description
Phase 3: **Design** 	▪ ***Purpose:*** - *Develop a detailed design of database Information System based on all specifications and requirements.* - *Create a **Logical Data Model (Logical Schema)*** - *Create a **Physical Data Model (Physical Schema)*** ▪ ***Database Professional Role:*** - *Database Analyst* - *Database Administrator* ▪ ***Deliverables:*** 1. ***Primary Database Deliverable – Logical Data Model (Logical Schema)** (E-R Model or Enhanced E-R Model)* 2. ***Primary Database Deliverable – Physical Data Model (Physical Schema)** (E-R Model or Enhanced E-R Model)*
Phase 4: **Development and Implementation** 	▪ ***Purpose:*** - *Develop & physically implement the design created in design phase.* - *Implement **Physical Schema*** ▪ ***Database Professional Role:*** - *Database Analyst* - *Database Developer* - *Database Administrator* ▪ ***Deliverables:*** 1. ***Fully Tested & Implemented Database Information System Based on Requirements*** 2. ***Users Trained***

Phases	Description
Phase 4: **Operations & Maintenance** 	▪ ***Purpose:*** - *Maintain, operate & backup the Information System* - *Repeat entire Methodology as changes are required* ▪ ***Database Professional Role:*** - *Database Analyst – Re-design for new business requirements & errors in database design* - *Database Developer – Re-design & implement for new business requirements & errors in database design* - *Database Administrator – Maintain, operate, backup & improve performance* ▪ ***Deliverables:*** 1. ***Keep the lights on*** 2. ***Upgrade as needed (Repeat Methodology)*** 3. *Updated Operations & Maintenance Document*

Analysis Phase Review – Conceptual Data Model (ER/EER Diagrams)

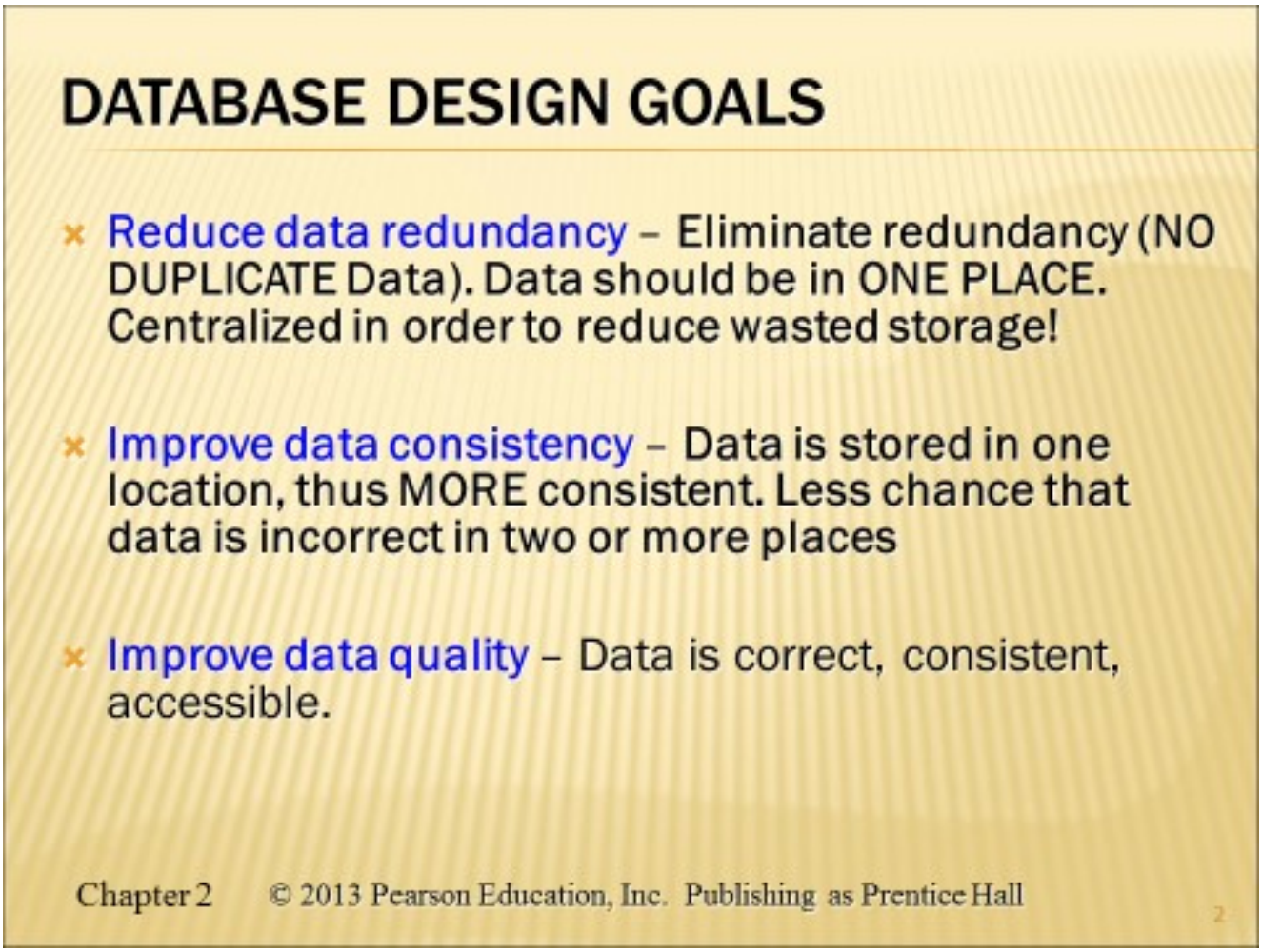
1.2 Summary & Overview of Basic ER Model

1.2.1 Overview of the E-R Model

- The Entity-Relational Model or E-R Model is defined as follows.
  - **E-R Model** – detailed logical representation of the *entities* and their *relationship* in the business environment
  - **E-R Diagram** – A pictorial or graphical representation of the E-R Model. The E-R Diagram uses geometric symbols to create the E-R Model.

E-R Modeling Design Goals

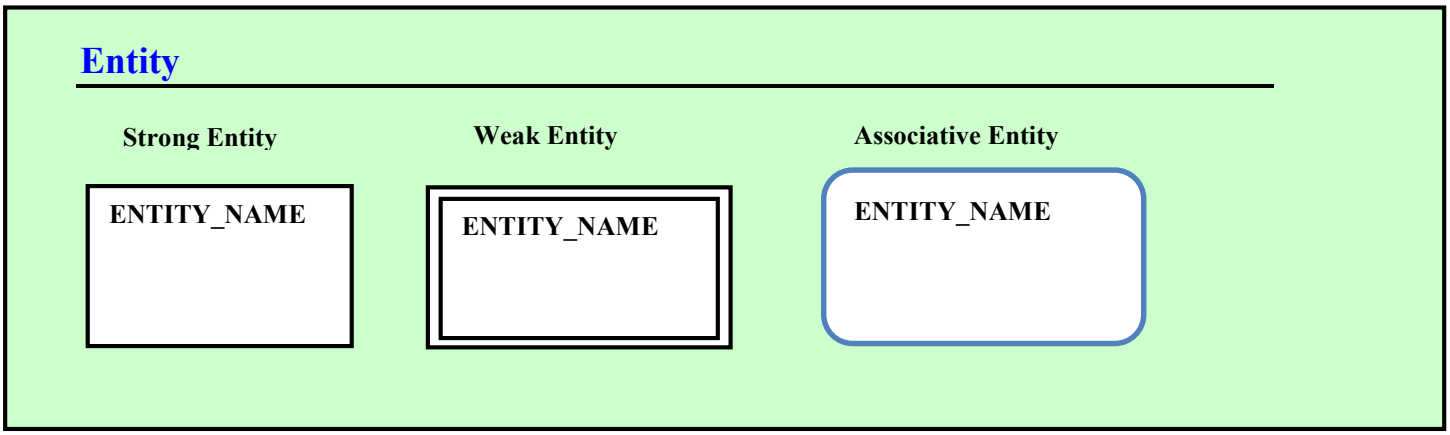
- The following goals or design criteria's are target of a good **E-R Model/Diagram**.



E-R Diagram Notation

- The following three figures illustrates the basic **E-R Diagram** notations most commonly used.

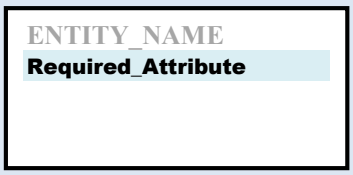
- **E-R Diagrams** for **Entities**:



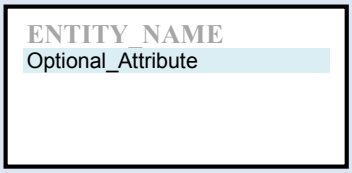
- **E-R Diagrams** for **Attributes**:

Attributes

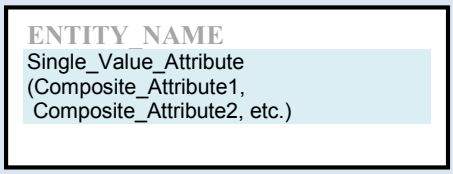
REQUIRED Attribute – Definition: Instance MUST have a value. Syntax: Name in bold font



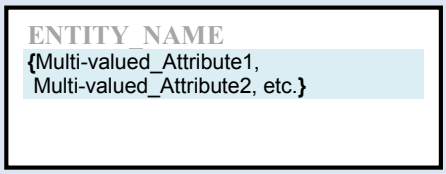
OPTIONAL Attribute – Definition: Instance value is optional. Syntax: Name in normal font



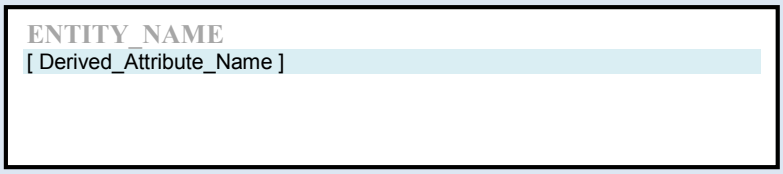
COMPOSITE Attribute – Definition: Instance single value can be broken down into several parts or components
Syntax: Single value & composite components in Parentheses.



MULTI-VALUED Attribute – Definition: Instance can contain more than one value Syntax: Enclose within braces the Single value & multi-values in Parentheses



DERIVED Attribute – Definition: Instance value is calculated from other attributes, or attributes in related Entities etc.
Syntax: Enclose attribute within brackets



- **E-R Diagrams** for **Relationships**:

Relationships

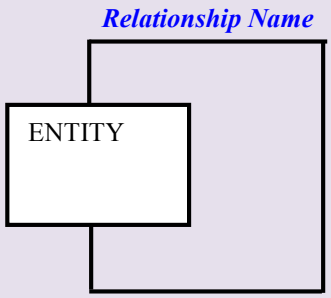
Relationship Basics

- **Relationship** = LINK & NAME between entities that describes how data between them relate.
- Relationship is the combination of the line & **relationship name**

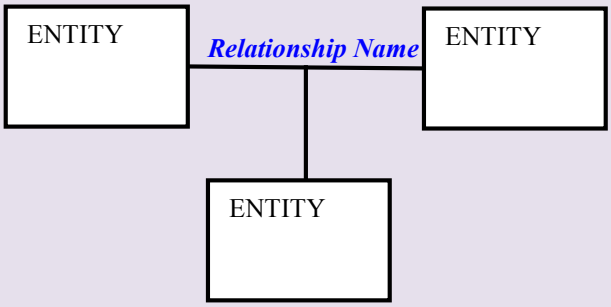
Relationship Degree

Binary Relationship

Unary Relationship



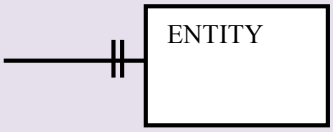
Tenary Relationship



Relationship Cardinality

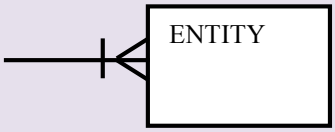
Mandatory One

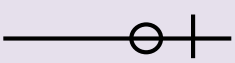
Minimum Cardinality = **One**
Maximum Cardinality = **One**



Mandatory Many

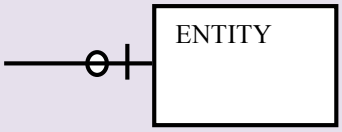
Minimum Cardinality = **One**
Maximum Cardinality = **Many**

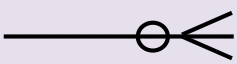




Optional One

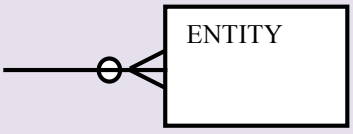
Minimum Cardinality = **0**
Maximum Cardinality = **One**





Optional Many

Minimum Cardinality = **0**
Maximum Cardinality = **Many**

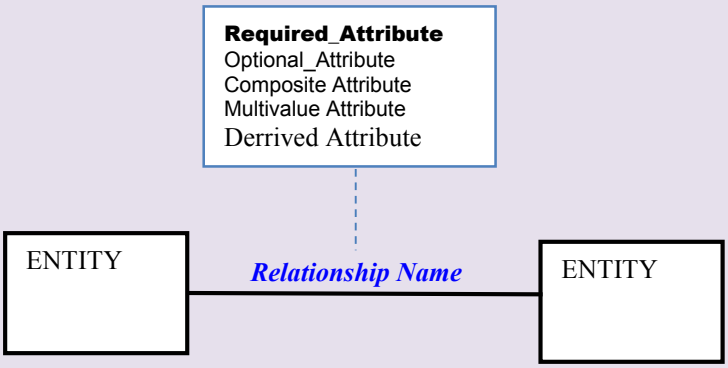


- **E-R Diagrams** for **Relationships (Cont.)**:

Relationships (Cont.)

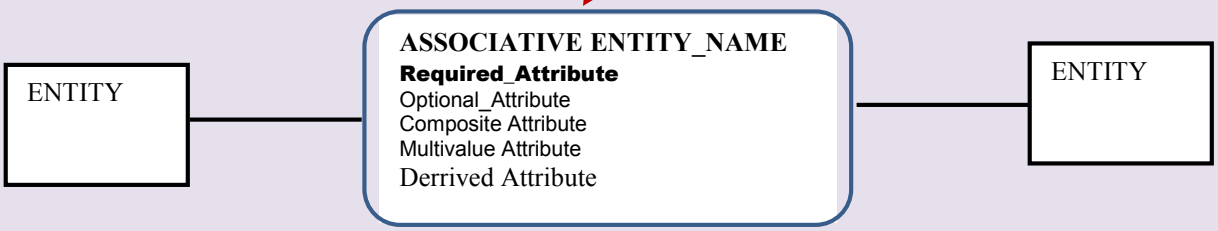
Relationship Attributes

- **Relationship Attributes** = describe features & characteristics of the entities relationships.



Relationship Attributes Converted to Associative Entity

- **Associative Entity** = An ENTITY that represents a relationship.
- A list of relationship attributes shown above, are converted to an ENTITY that relates two entities



1.2.2 Algorithm and Guidance for Creating E-R Models

- Next we will apply a step-by-step approach on creating E-R model based on the concepts learned in this chapter.
- This is something I have put together to help students of the course apply a step approach.
- Creating ER Diagrams requires a thorough understanding of the concepts and apply to the business rules.
- The following foundation is needed

1. Understanding of what are Business Rules
2. **E-R Diagram** components and symbols (Entities, Attributes, Relationships)

E-R Diagram Creation Algorithm

- Algorithm as follows:

Algorithm Steps		If Yes	If No
Step 1 – For each sentence/Business Rule of the problem do the following:			
	Step 1a – Can you identify one or more **ENTITY** in the ***Business Rule***?	1. ***Draw*** & ***Name*** each **ENTITY**. 2. Go to next step	Go to Next step
	Step 1b – Can you identify one or more **entity ATTRIBUTES** in the ***Business Rule***?	1. ***Write*** & ***Name*** each **ATTRIBUTES**. 2. Identify if attribute is one of the following: ▪ Required_Attribute ▪ Optional_Attribute ▪ Composite Attribute ▪ Multivalue Attribute ▪ Derived Attribute 3. Go to next step	Go to Next step
	Step 1c – Can you identify one or more **RELATIONSHIP** in the ***Business Rule***?	1. ***Draw*** & ***Name*** each **RELATIONSHIP**. 2. Go to next step	Go to Next step
	Step 1d – Can you identify one or more **relationship ATTRIBUTE** in the ***Business Rule***?	1. ***Write*** & ***Name*** each ***relational* ATTRIBUTES**. 2. Identify if attribute is one of the following: ▪ Required_Attribute ▪ Optional_Attribute ▪ Composite Attribute ▪ Multivalue Attribute ▪ Derived Attribute 3. Go to next step	Go to Next step
	Step 1e – Can you identify one or more **relationship CARDINALITY** in the ***Business Rule***?	1. ***Draw*** & ***Name*** each **CARDINALITY**. 2. Go to next step	Go to Next step
	Step 1f – Is there opportunity to convert **relationship ATTRIBUTE** to **ASSOCIATED ENTITY**?	1. ***Draw*** & create **ASSOCIATED ENTITY.** 2. ***Write*** & ***Name*** each ***relational* ATTRIBUTES** inside **ASSOCIATED ENTITY** 3. Go to next step	Go to Next step
Step 2 – Repeat step 1 until all business rules have been read and E-R diagrams interconnected.			

1.2.3 Sample Problems – Book Exercises

Exercise #2 (Page 101)

1. For each of the descriptions below, perform the following tasks:

   i) Identify the degree and cardinalities of the relationship.
   ii) Express the relationships in each description graphically with an E-R diagram

a. A book is identified by its ISBN number, and it has a title, a price, and a date of publication. It is published by a publisher, each of which has its own ID number and a name. Each book has exactly one publisher, but one publisher typically publishes multiple books over time.

(2.a.i) This relationship is a degree of 2 (binary). This relationship is One-to-Many from Publisher to Book.

(2.a.ii)

Note: This solution assumes that we wish to track a Publisher even if it does not yet have a Book published.

b. A book (see above in 2a) is written by one or multiple authors. Each author is identified by an author number and has a name and date of birth. Each author has either one or multiple books; in addition, occasionally data are needed also regarding prospective authors who have not yet published any books.

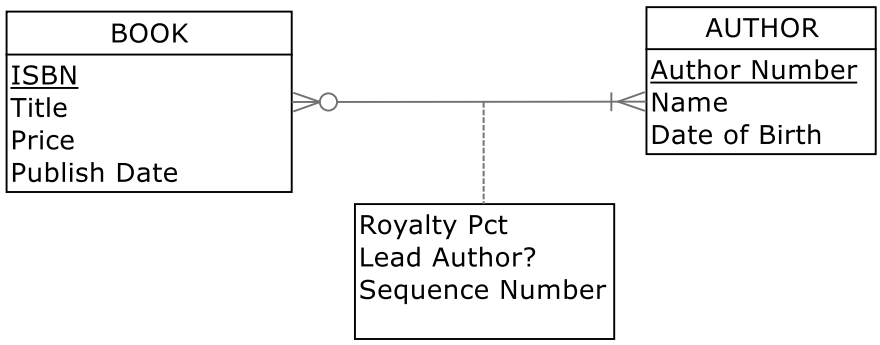
(2.b.i) This relationship is a degree of 2 (binary). This relationship is Many-to-Many from Author to Book.

(2.b.ii)

c. In the context specified above in 2a and 2b, better information is needed regarding the relationship between a book and its authors. Specifically, it is important to record the percentage of the royalties that belong to a specific author, whether or not a specific author is a lead author of the book, and each author's position in the sequence of the book's authors.

(2.c.i) This relationship is a degree of 2 (binary). This relationship is Many-to-Many from Author to Book.

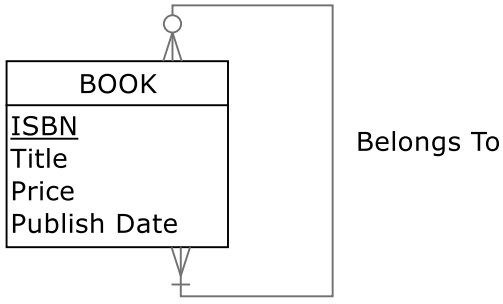
(2.c.ii)



d. A book (see 2a above) can be part of a series, which is also identified as a book and has its own ISBN number. One book can belong to several sets and a set consists of at least one but potentially many books.

(2.d.i) This relationship is a degree of 1 (unary). This relationship is Many-to-Many.

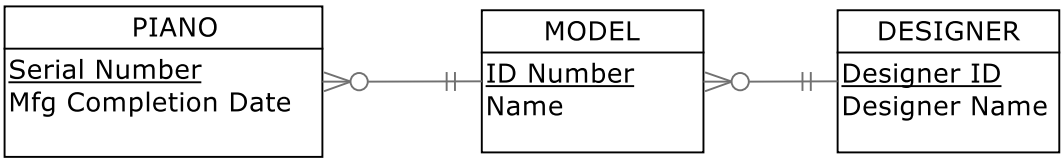
(2.d.ii) This solution assumes that "series" and "sets" are synonymous terms. The question does not require that a series have any special attributes or distinguishing features, so it can be represented in the data model like any other Book instance and identified by ISBN.



e. A piano manufacturer wants to keep track of all the pianos it makes individually. Each piano has an identifying serial number and a manufacturing completion date. Each instrument represents exactly one piano model, all of which have an identification number and a name. In addition, the company wants to maintain information about the designer of the model. Over time, the company often manufactures thousands of pianos of a certain model, and the model design is specified before any single piano exists.

(2.e.i) These relationships have a degree of 2 (binary). These relationships are One-to-Many.

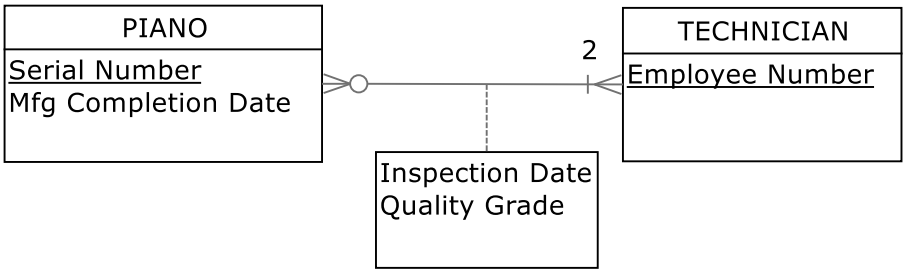
(2.e.ii)



f. A piano manufacturer (see 2e above) employs piano technicians who are responsible for inspecting the instruments before they are shipped to the customers. Each piano is inspected by at least two technicians (identified by their employee number). For each separate inspection, the company needs to record its date and a quality evaluation grade.

(2.f.i) This relationship is a degree of 2 (binary). This relationship is Many-to-Many.

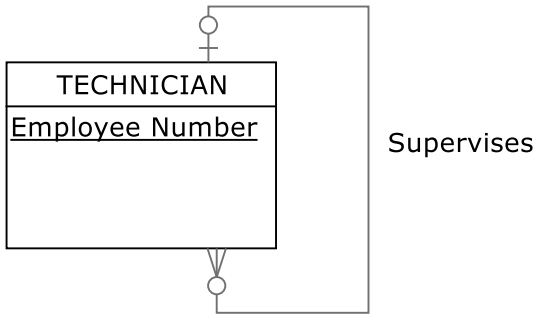
(2.f.ii)



g. The piano technicians (see 2f above) have a hierarchy of reporting relationships: some of them have supervisory responsibilities in addition to their inspection role and have multiple other technicians report to them. The supervisors themselves report to the chief technician of the company.

(2.g.i) This relationship is a degree of 1 (unary). This relationship is One-to-Many.

(2.g.ii) Because the chief technician is not represented as a separate entity type, that person does not have a supervisor, and this leads to the 0 minimum cardinality on the 1 side of the unary relationship.



Exercise #7

7. The entity type STUDENT has the following attributes: Student Name, Address, Phone, Age, Activity, and No of Years. Activity represents some campus-based student activity, and No of Years represents the number of years the student has engaged in this activity. A given student may engage in more than one activity. Draw an ERD for this situation. What attribute or attributes did you designate as the identifier for the STUDENT entity? Why?

Answer

7. *ERD for Student situation:*

Note: Assume Student Name is unique and available to be used as the identifier.

STUDENT
Student Name
Phone
Address
Age
{Activity History (Activity, No of Yrs)}

Exercise #10

MILLENNIUM COLLEGE
GRADE REPORT
FALL SEMESTER 200X

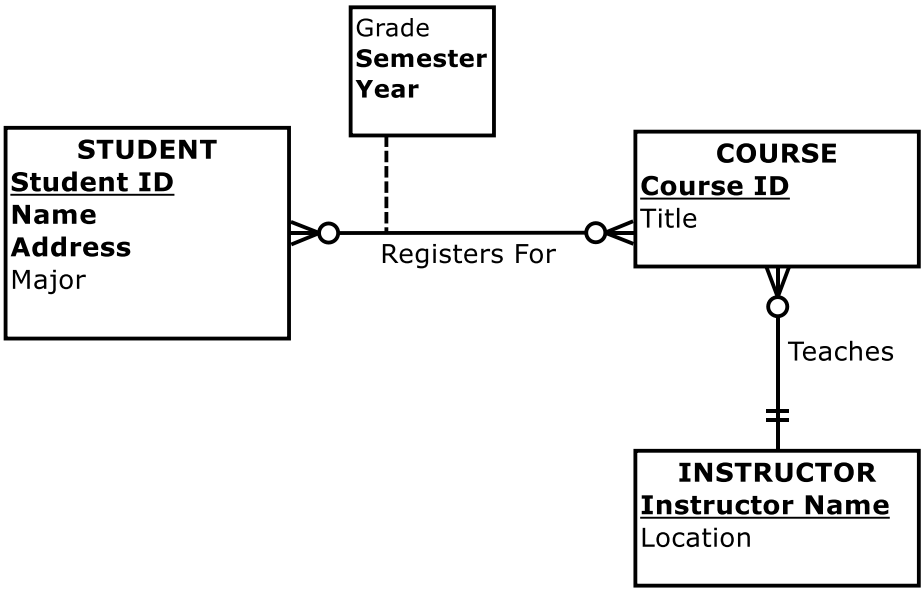
NAME:	Emily Williams	ID: 268300458
CAMPUS ADDRESS:	208 Brooks Hall	
MAJOR:	Information Systems	

COURSE ID	TITLE	INSTRUCTOR NAME	INSTRUCTOR LOCATION	GRADE
IS 350	Database Mgt.	Codd	B104	A
IS 465	System Analysis	Parsons	B317	B

10. Figure 2-26 shows a grade report that is mailed to students at the end of each semester. Prepare an ERD reflecting the data contained in the grade report. Assume that each course is taught by one instructor. Also, draw this data model using the tool you have been told to use in the course. Explain what you chose for the identifier of each entity type on your ERD.

10. *ERD for Figure 26 Grade Report:* Student ID was chosen as the identifier for the STUDENT entity type as it is likely unique. Course ID was chosen as the identifier for the COURSE entity type as it is likely unique. Instructor Name was chosen as the identifier for the INSTRUCTOR entity type and it is assumed to be unique—should discussions during analysis work prove otherwise, it may be wise to create either (a) a composite identifier comprised of Instructor Name and Location, or (b) a new attribute Instructor ID that will be a unique number which can serve as an identifier.

    Note: The addition of Semester and Year attributes on the Registers for relationship allows this diagram (and resulting database) to reflect multiple semesters of data.



1.2.4 Midterm Exam Review Example

- This example will not be listed in these notes or provided to student.
- The professor will display and discuss in class from another document but will not make it available to the students at this time.

1.3 Summary & Overview of the Enhanced E-R Model (EER Model) CHAPTER 3

1.3.1 Overview of the E-R Model of Chapter 3

- The **BASIC Entity-Relational Model**:
  - Suitable for modeling most common business problems
- The **ENHANCED Entity-Relational Model**:
  - Today's business environments have changed
  - Relationships are more complex.
  - Data is more complex and bigger!
  - Due to social media and internet sharing and collaboration data is now BIG DATA!
- To cope with these changes, the basic E-R model has been enhanced to accurately represent the complex data encountered in today's business environment.
- The most important modeling enhancement to the E-R model is the following:

- **Supertype** & **subtype** relationships

- Basic concept:

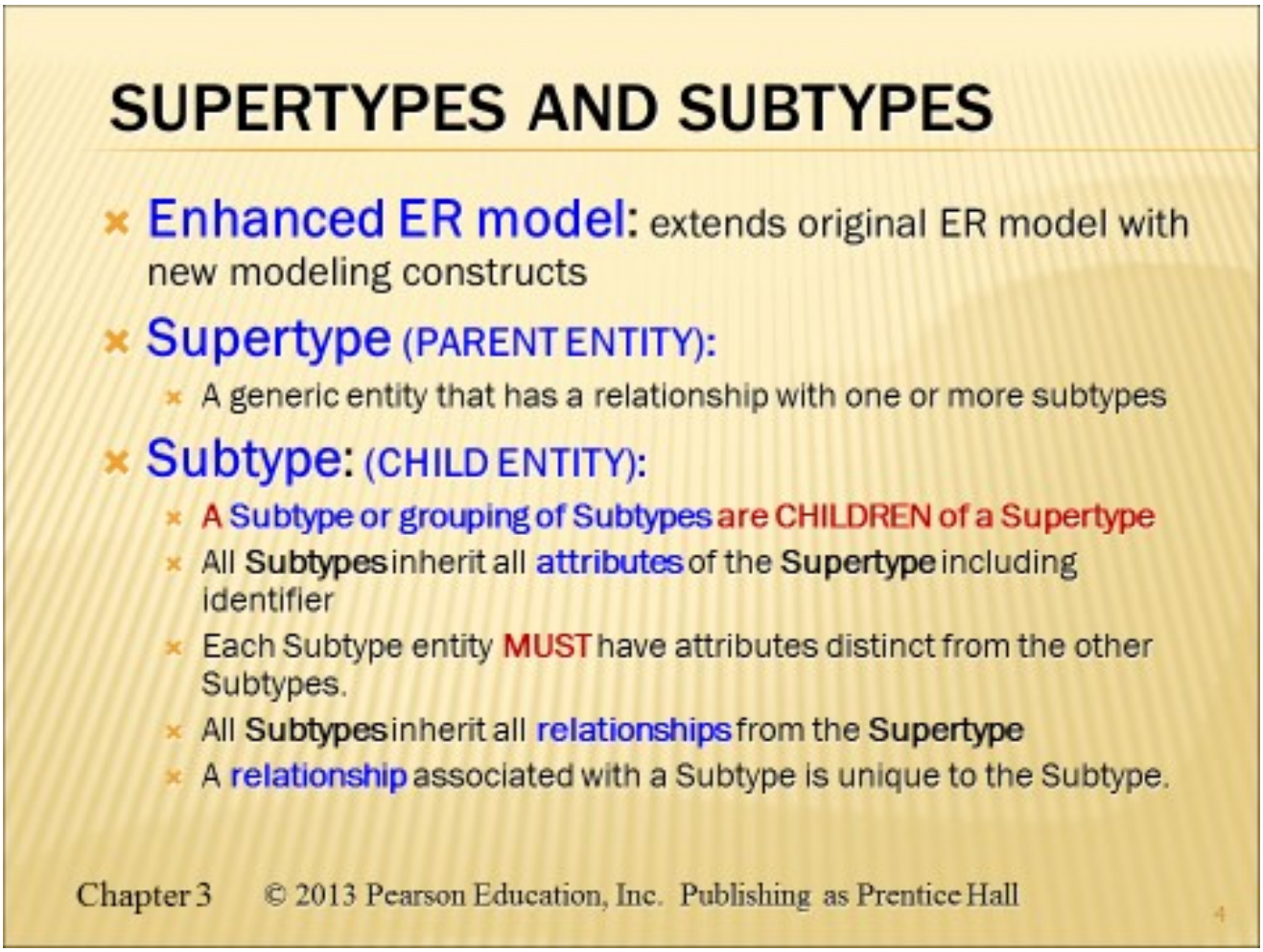


Figure 3-1 Basic notation for supertype/subtype notation

a) EER notation

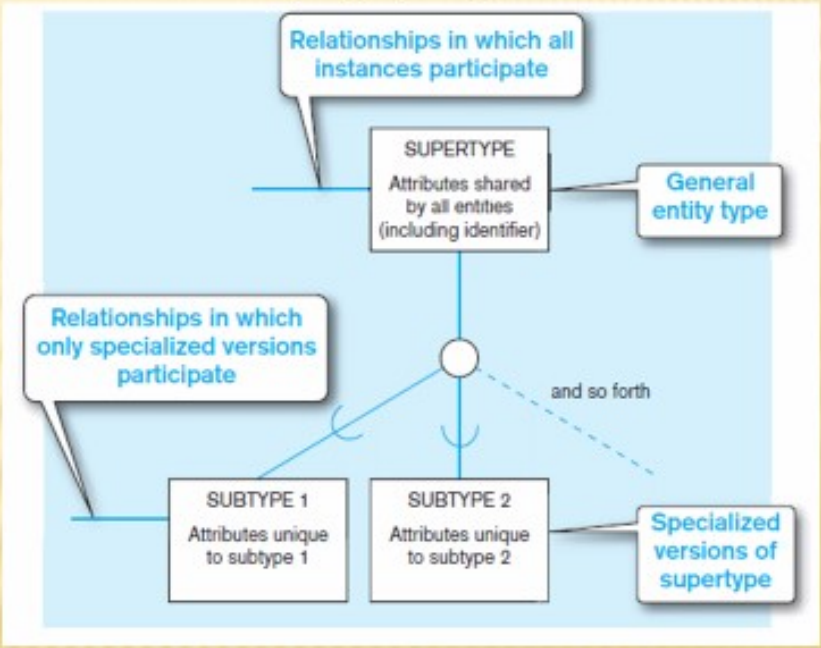
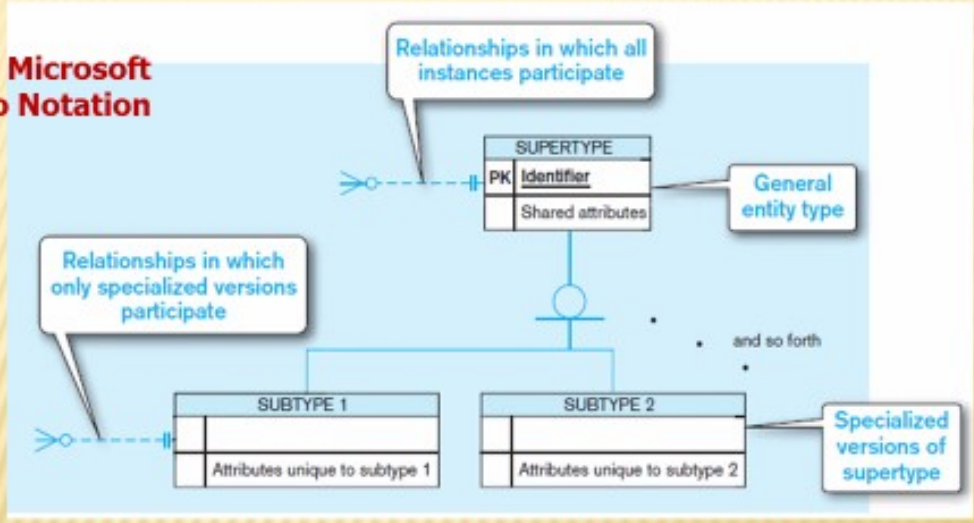
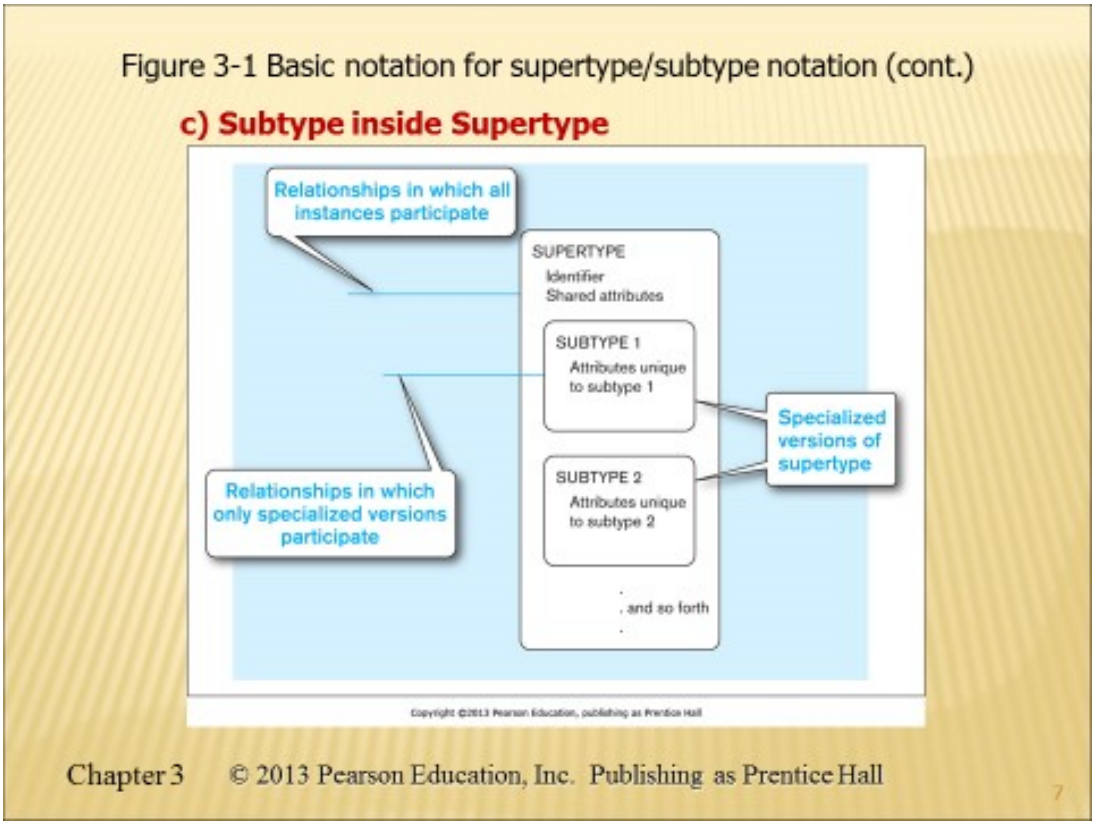


Figure 3-1 Basic notation for supertype/subtype notation (cont.)

b) Microsoft Visio Notation

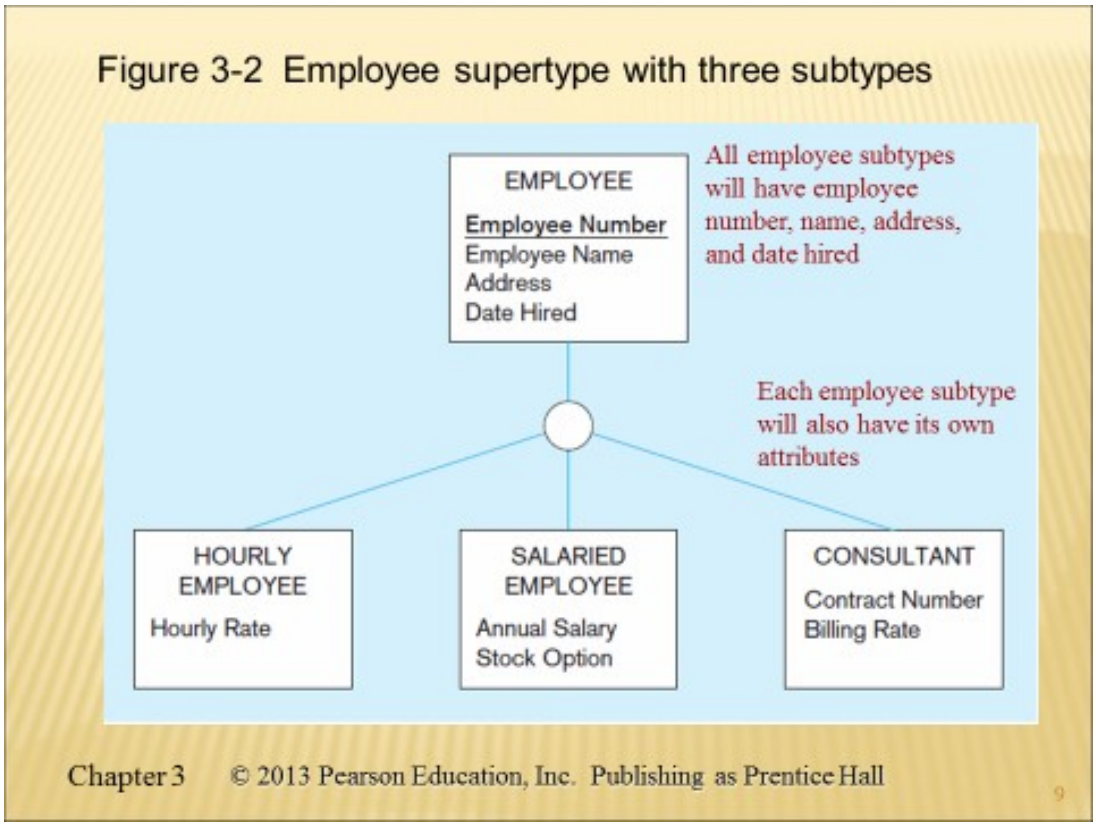


Different modeling tools may have different notation for the same modeling constructs.

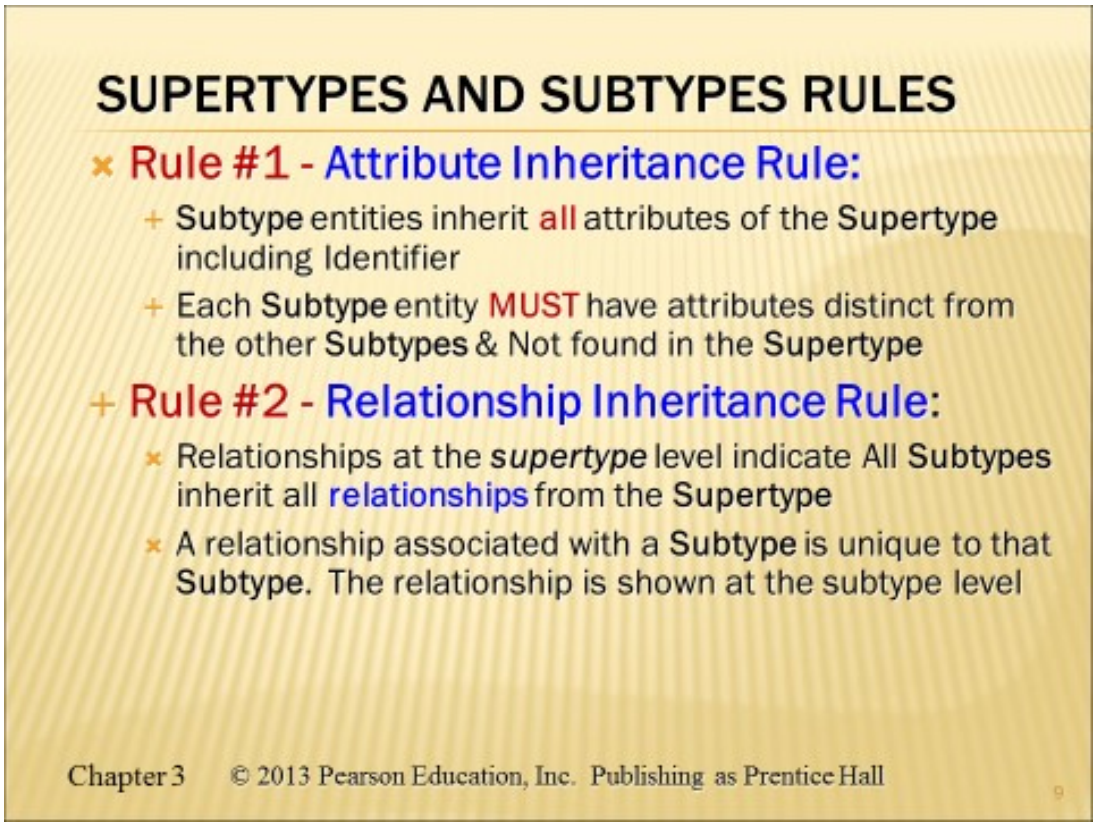


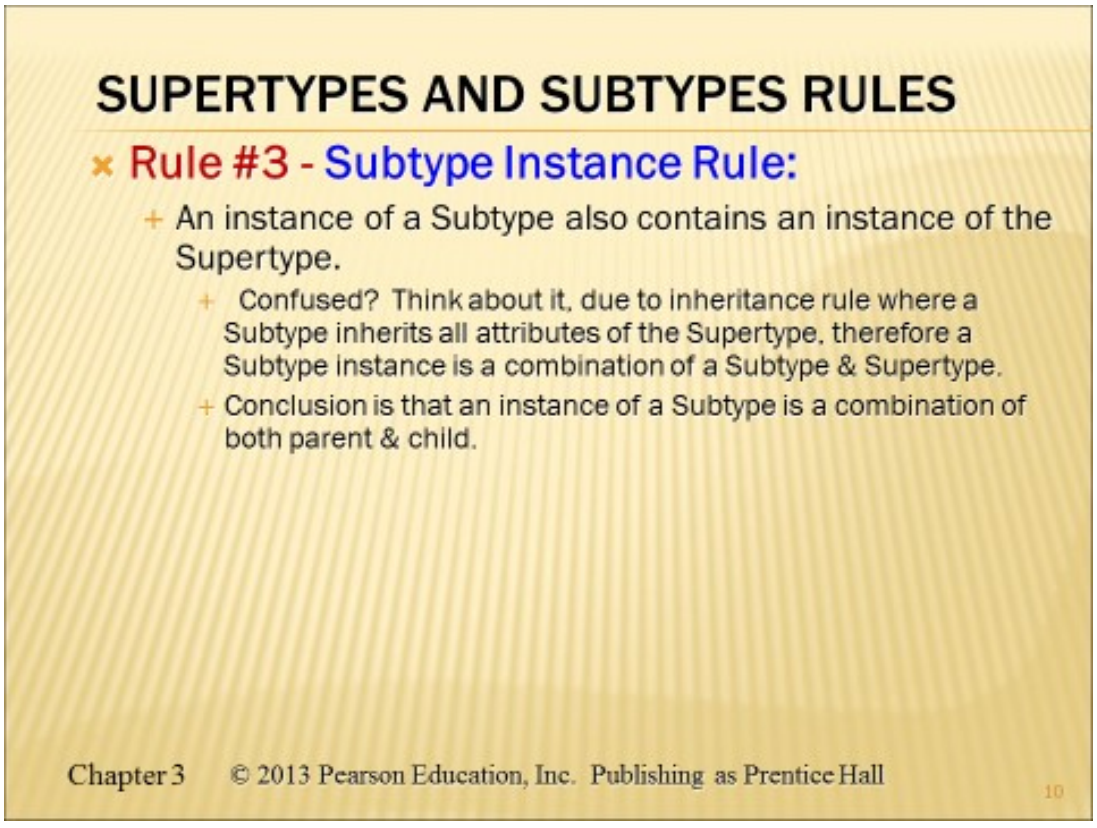
Example

- Example:



- Supertype & Subtype rules:



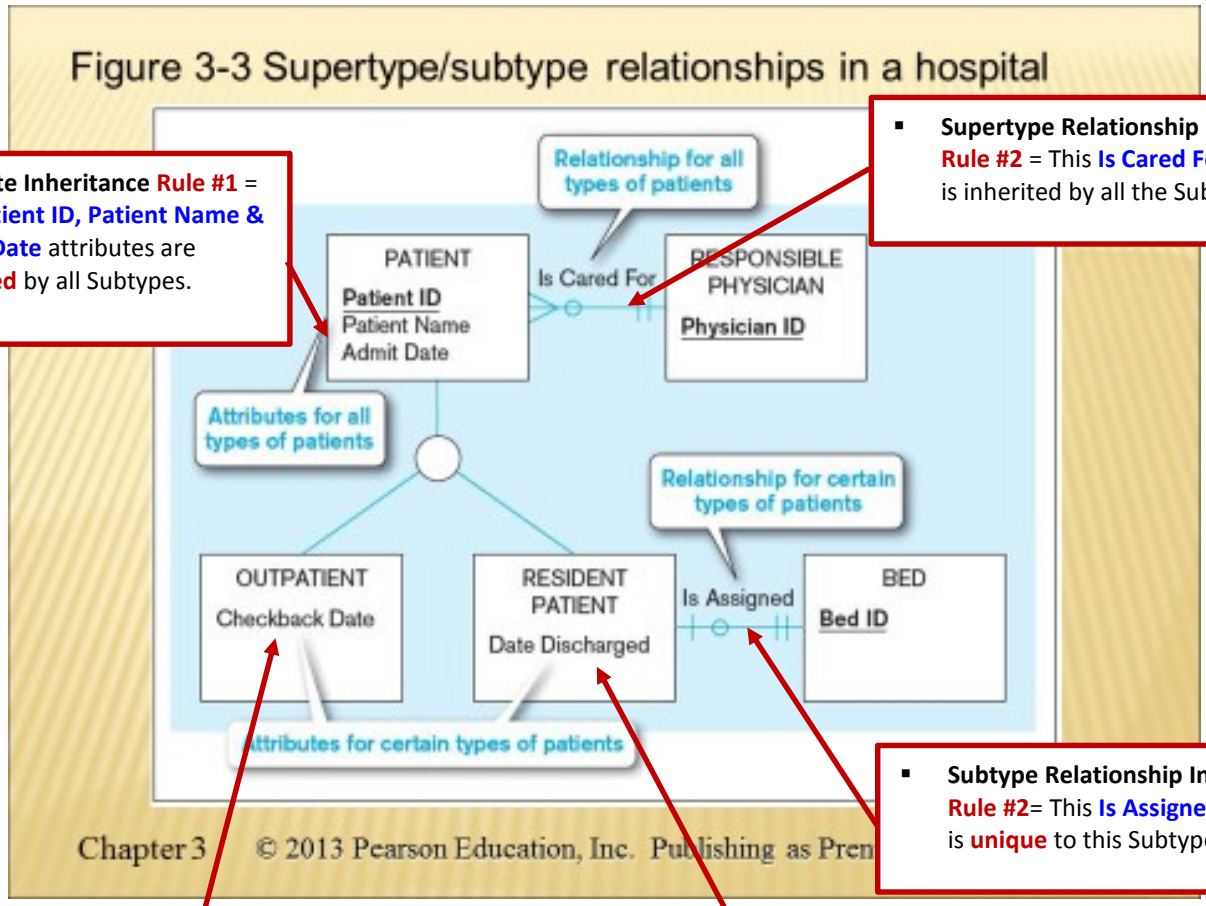


SUPERTYPES AND SUBTYPES RULES

- Rule #4 - When to consider using Supertype & Subtype Relationships:
- When either or all of the following conditions exists:
  1. You have more than one entity that share common attributes (Redundancy exits)
  2. But also have one or more distinct attributes from the other entities
  3. A subtype participates in relationships that are unique to that subtype

Example

- Example of attribute & relationship inheritance rules for :



- **Attribute Inheritance Rule #1** = This **Patient ID, Patient Name & Admit Date** attributes are **inherited** by all Subtypes.

- **Supertype Relationship Inheritance Rule #2** = This **Is Cared For** relationship is inherited by all the Subtypes.

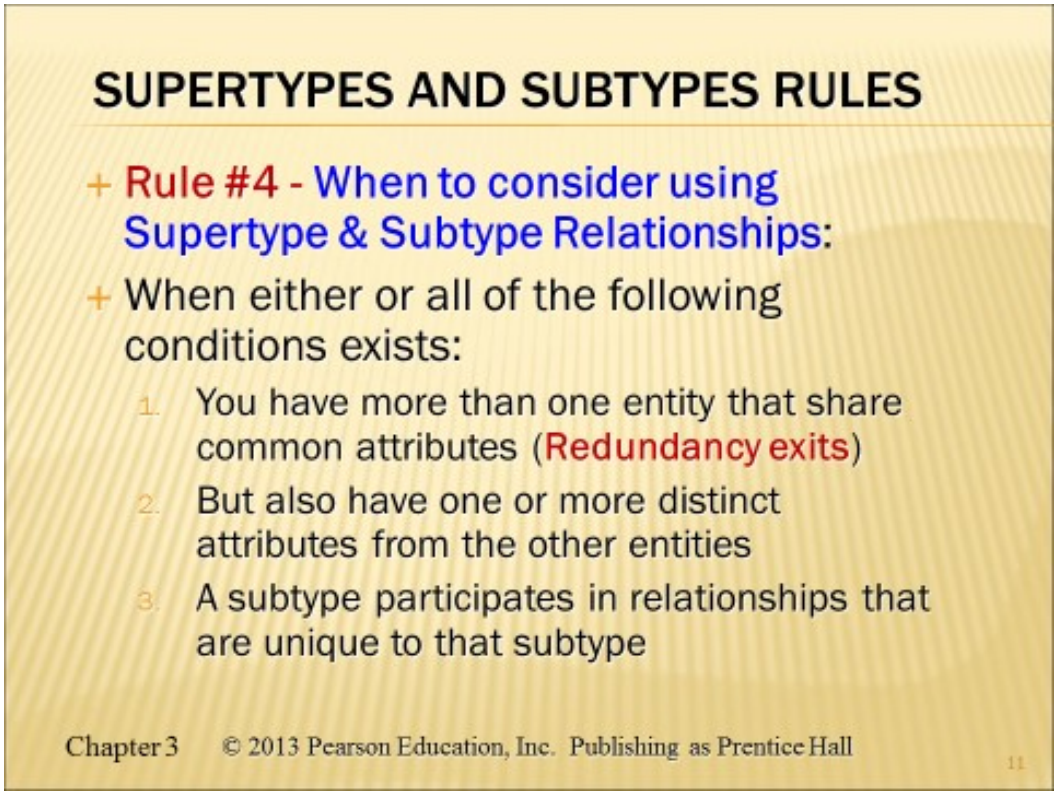
- **Subtype Relationship Inheritance Rule #2**= This **Is Assigned** relationship is **unique** to this Subtype only.

- **Subtype Attribute Inheritance Rule #1** = this attribute **Checkback Date MUST** be **unique** to this Subtype only.

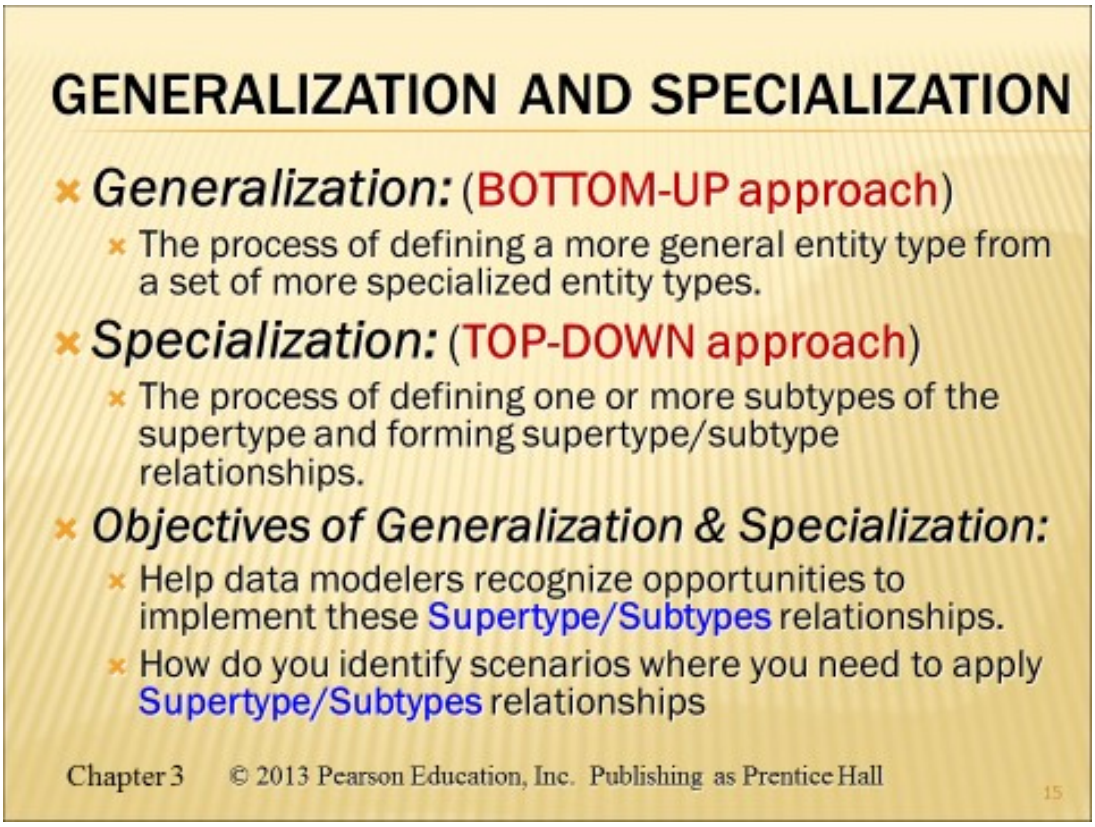
- **Subtype Attribute Inheritance Rule #1** = this attribute **Date Discharged MUST** be **unique** to this Subtype only.

3.2.3 Specialization & Generalization – Two Process to Implement Supertypes & Subtypes

- The objectives are to identified scenarios that qualifies for **Supertype** & **Subtypes** relationships implementation based on the following rule:

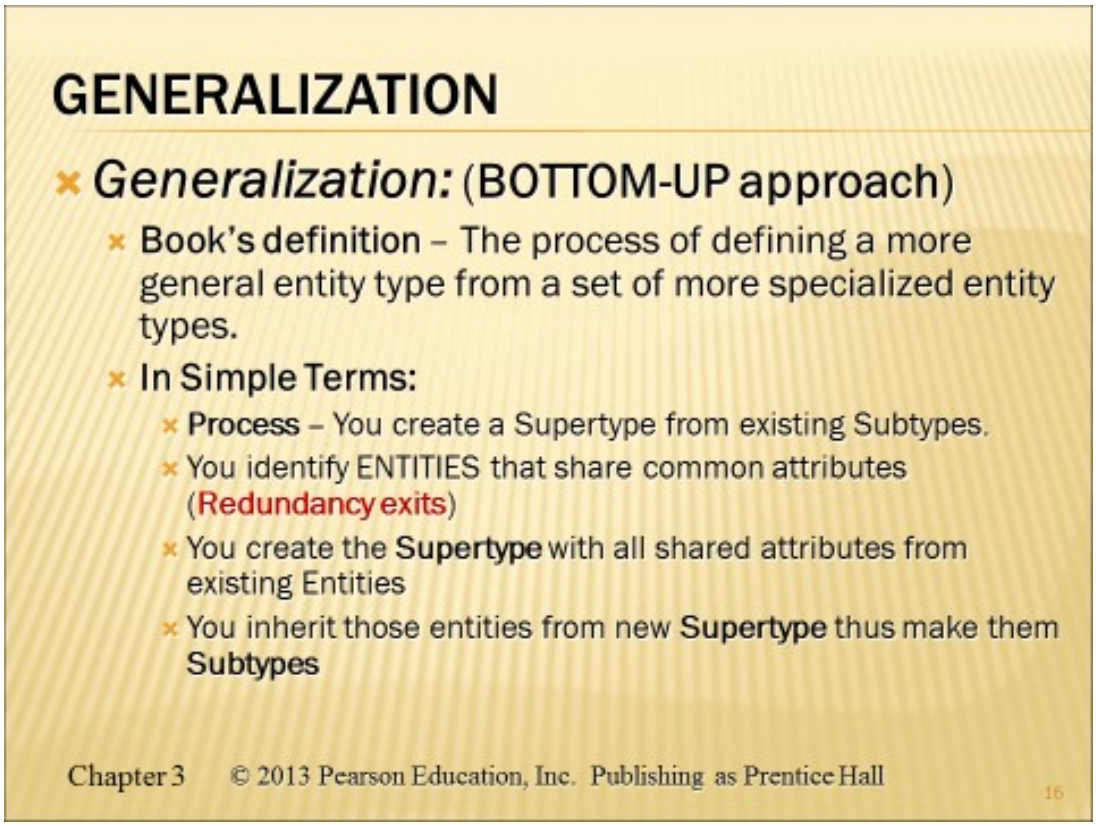


- Next step is to use one of the following processes to model the **supertype/subtype** relationship:



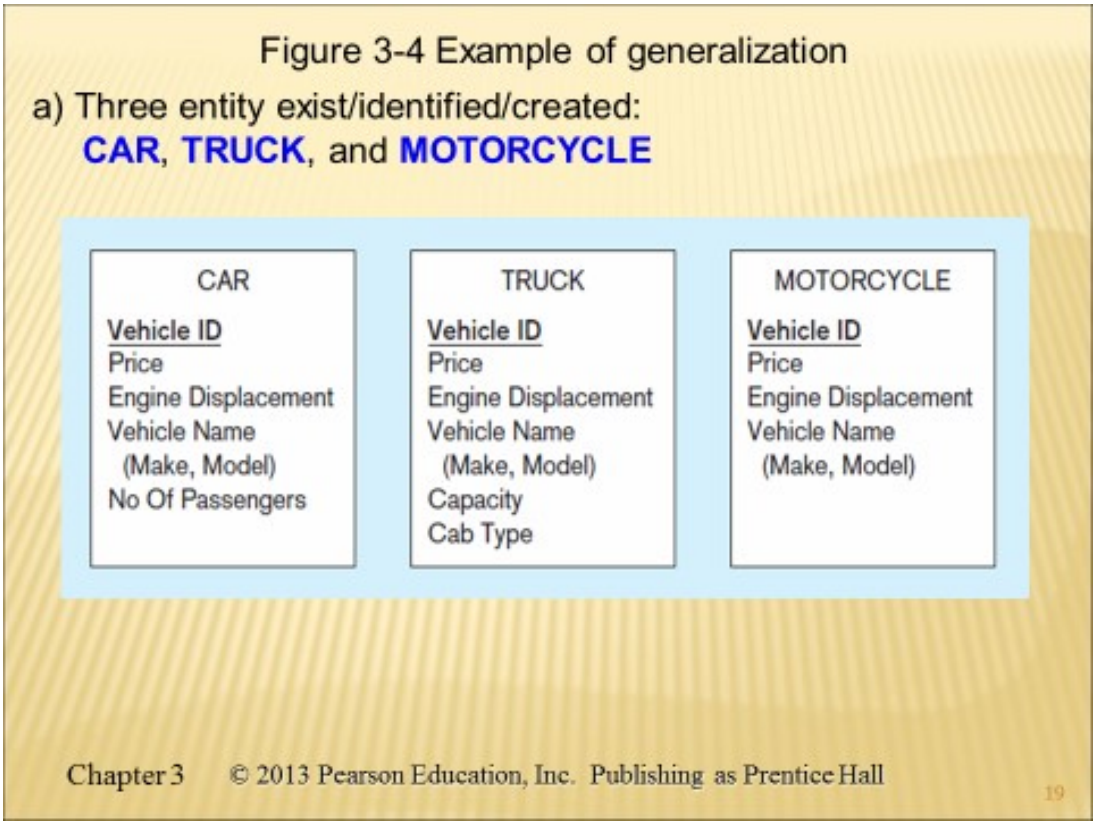
Generalization

- Definition explained:



Example

- Generalization example:



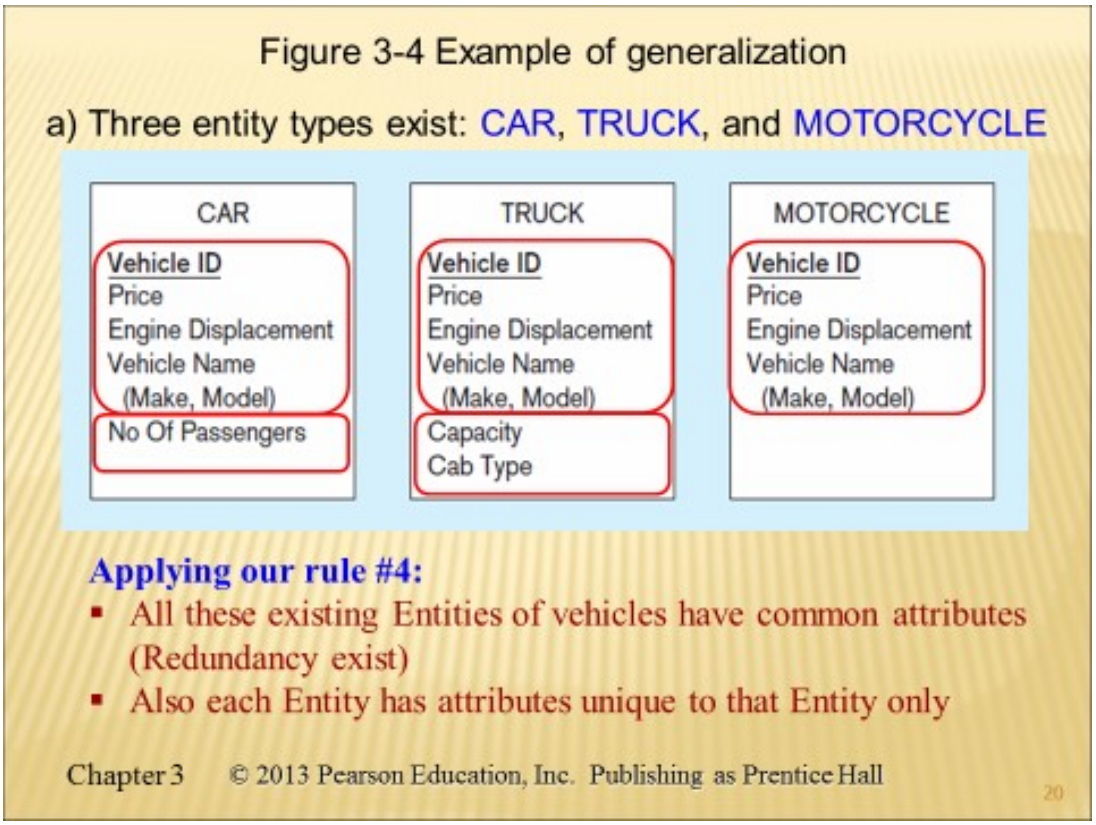
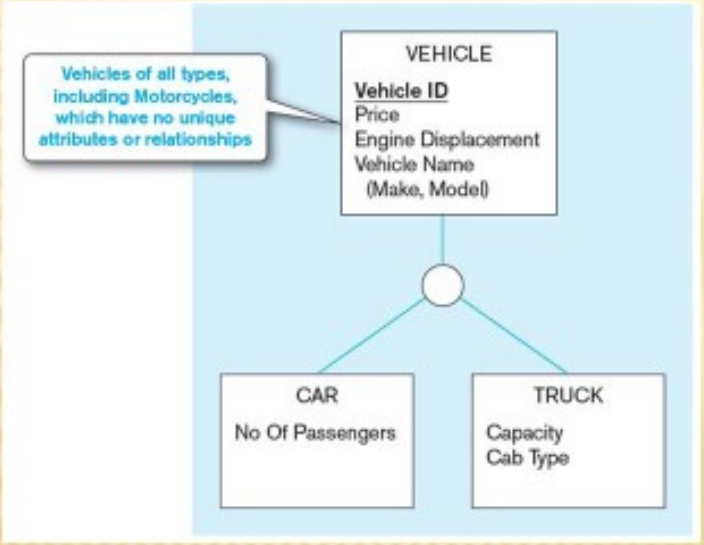


Figure 3-4 Example of generalization (cont.)

b) Generalization to **VEHICLE** supertype



Step 1 - put the SHARED attributes in a supertype

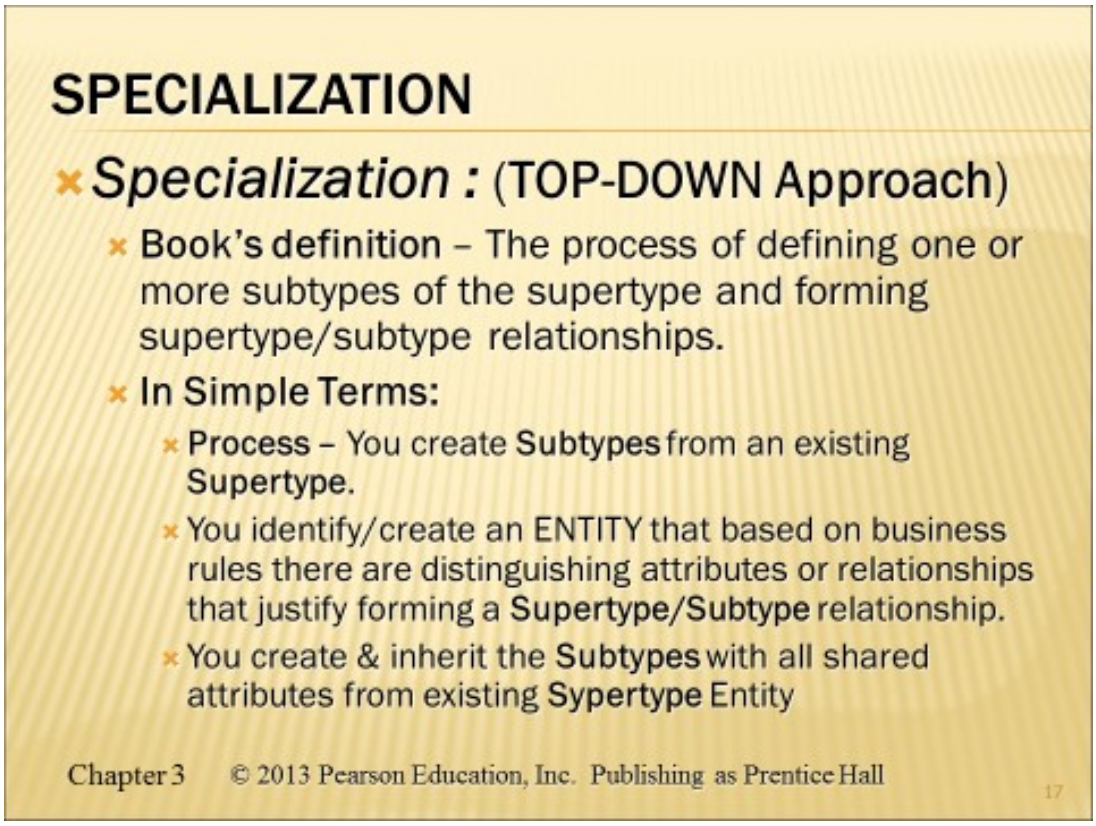
Step 2 – Create supertype & subtype inheritance diagram

Step 3 – Leave the UNIQUE attributes in each subtype

Note: no subtype for motorcycle, since does not meet Rule #4 it has no unique attributes

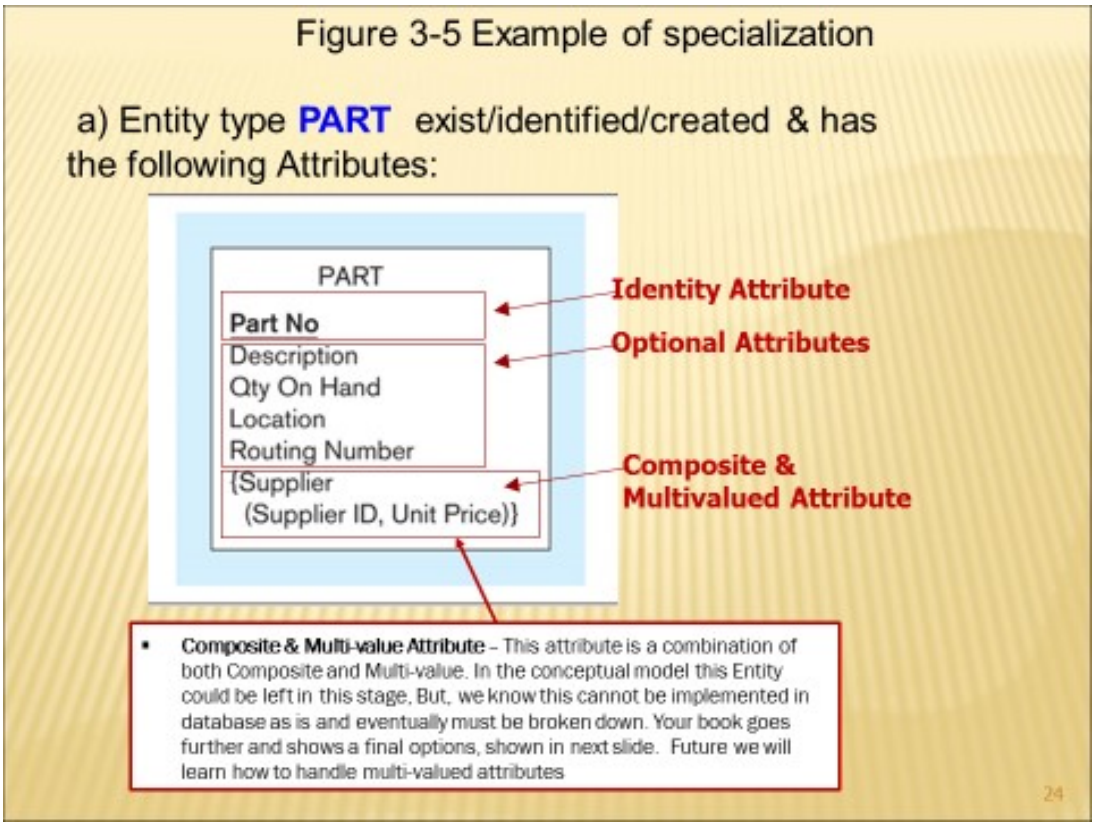
Specialization

- Definition explained



Example

- Generalization example:



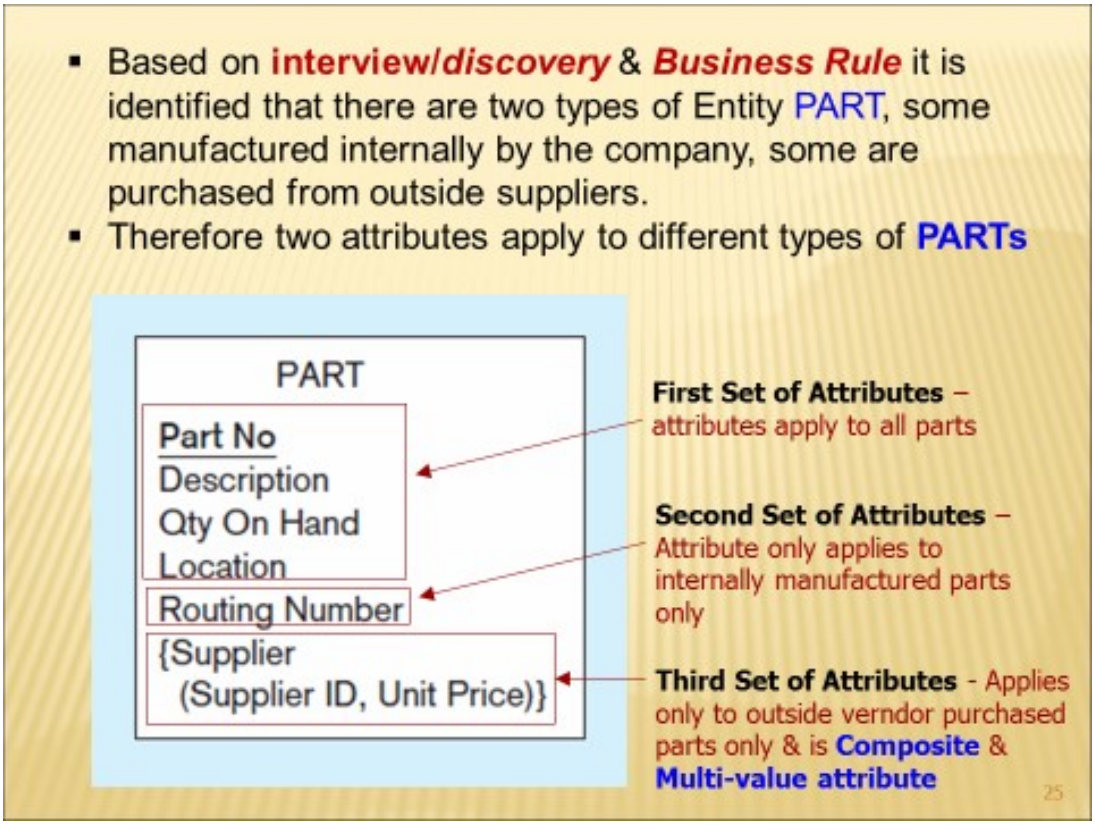
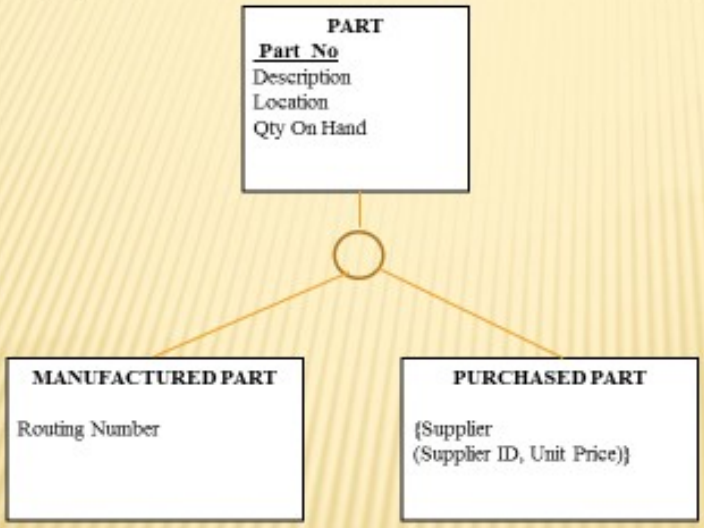


Figure 3-5 Example of specialization (cont.)

b) Specialization to **MANUFACTURED PART** and **PURCHASED PART**



Step 1 – Keep the SHARED attributes in a supertype

Step 2 – Create 2 supertypes & subtype inheritance diagram

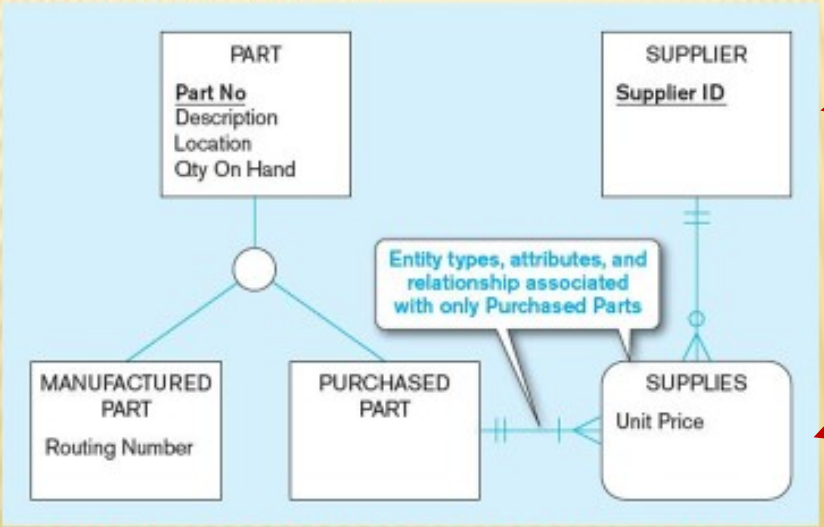
Step 3 – Place the UNIQUE attributes in each subtype

27

Figure 3-5 Example of specialization (cont.)

C) Further breakdown of **Composite/Multivalued Attribute** for **PURCHASED PART**

Step 4 – Composite/Multivalued attributed converted to ENTITY & ASSOCIATIVE ENTITY



- **Composite & Multivalued Attribute**– Part was subdivided/converted to an ENTITY **SUPPLIER**

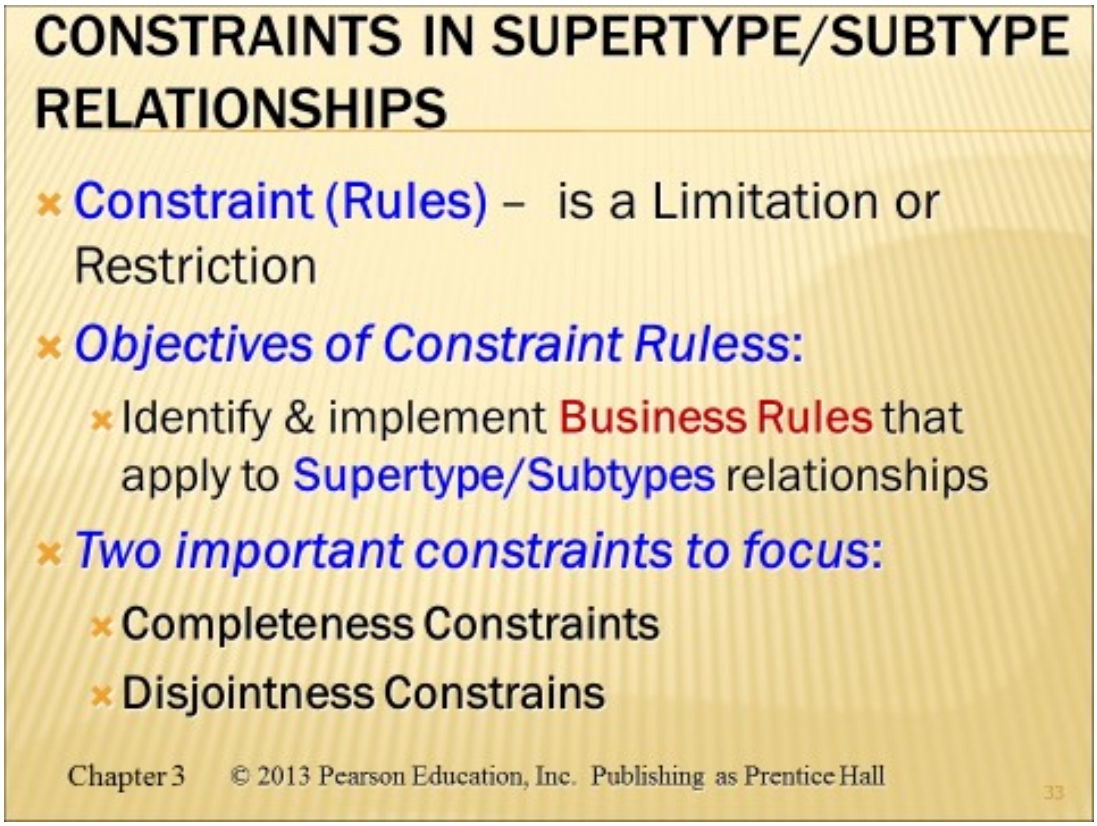
- **Composite & Multivalued Attribute**– part was subdivided/converted to an ASSOCIATIVE ENTITY **SUPPLIES**

Note: multivalued composite attribute was replaced by an associative entity relationship to another entity

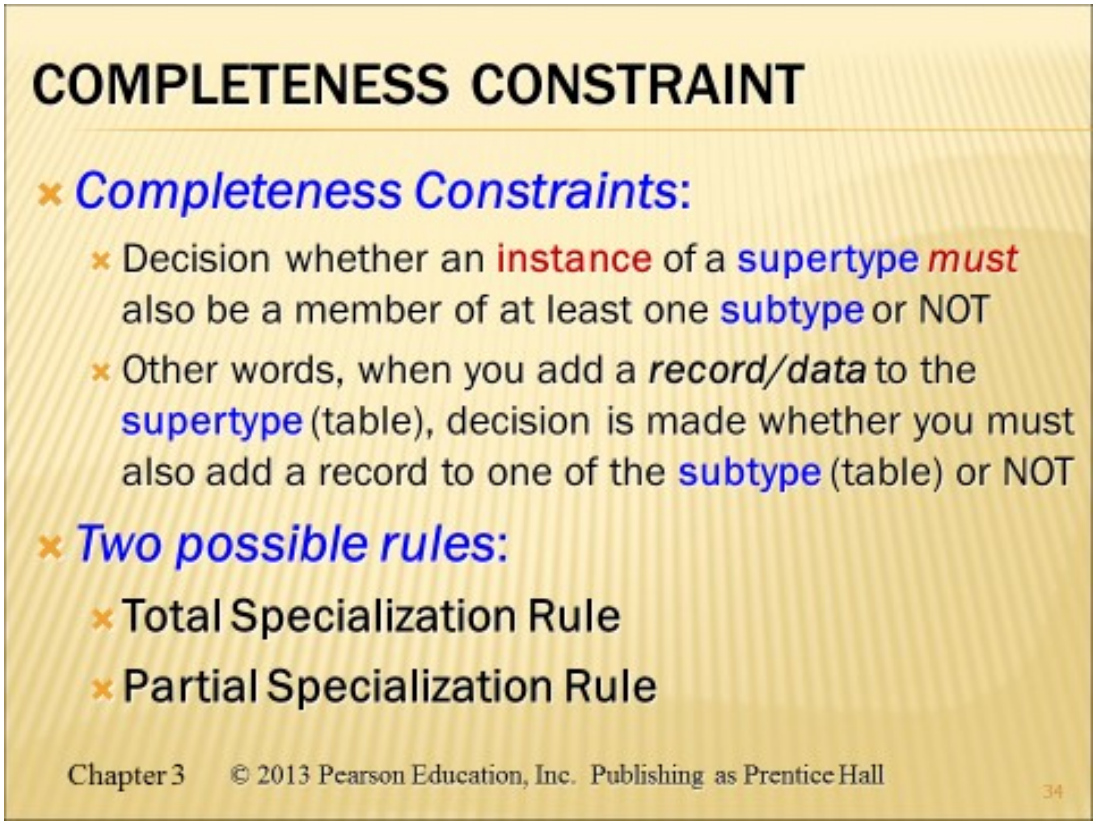
28

3.2.4 Constraints in Supertype & Subtypes Relationships

- Now we focus on another **Supertype** & **Subtypes** relationships topic to assist us in implementing certain types of Business Rules that apply to **Supertype** & **Subtypes** relationships:



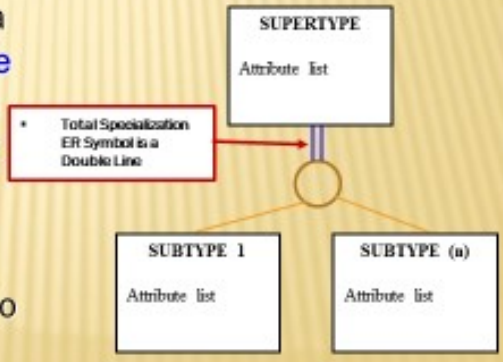
Completeness Constraints



Total Specialization Rule:

COMPLETENESS CONSTRAINT – TOTAL SPECIALIZATION

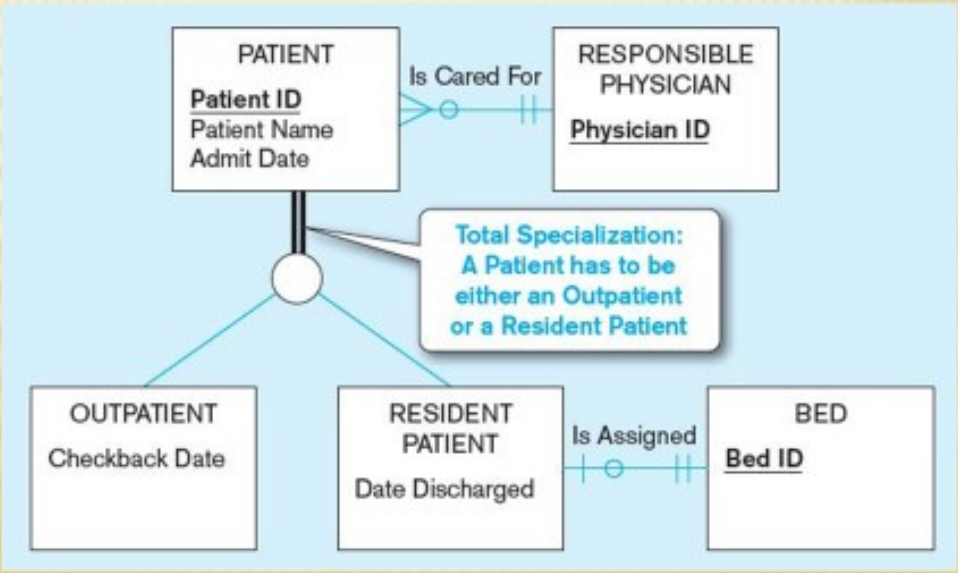
- *Total Specialization Rule*:
  - Each instance of a supertype *MUST* also be a member of at least one subtype
  - Other words, when you add a *record/data* to the supertype (table), you MUST also add a record to one of the subtype (table)
- *E-R Diagram*:
  - Double line from supertype to circle



Chapter 3 © 2013 Pearson Education, Inc. Publishing as Prentice Hall 35

Figure 3-6 Completeness constraints using Patient example

a) **Total specialization rule** – A patient **MUST** be either an **outpatient** or a **resident patient** (**This means no other type of patient exists in hospital. Cannot have just a patient**)



37

Figure 3-6 Completeness constraints using Patient example

a) **Total specialization rule (cont.)** – A patient **MUST** be either an **outpatient** or a **resident patient**:

- This means no other type of patient in hospital.
- Cannot have just a patient, MUST either be **outpatient** or **resident patient**
- This means every time you insert an instance (record) into **PATIENT** table, you **MUST** also insert an instance in **OUTPATIENT** or **RESIDENT PATIENT** table.

38

Figure 3-6 Completeness constraints using Patient example

a) Total specialization rule (cont.) –

- **Scenario #1** – Suppose we are registering or inserting a new **outpatient** to the hospital, then we would have to also insert a record to both **PATIENT** & **OUTPATIENT** tables:

PATIENT		
Patient ID	**Patient Name**	**Admit Date**
1111	Joe Smith	03/11/2008
2222	Angel Rod	11/07/2013
3333	Sam Brown	02/04/2014
4444	**Mary Johnson**	**08/24/2015**

OUTPATIENT
Checkback Date
05/21/2008
11/14/2013
03/05/2014
09/24/2015

If record inserted in **Supertype PATIENT**, you MUST also insert record in **Subtype OUTPATIENT**

Figure 3-6 Completeness constraints using Patient example

a) Total specialization rule (cont.) –

- **Scenario #2** – Suppose we are registering or inserting a new **resident patient** to the hospital, then we would have to also insert a record to **PATIENT** & **RESIDENT PATIENT** tables, but in addition the **BED** table since a Mandatory-one cardinality relationship exists between **RESIDENT PATIENT** table & the **BED** table:

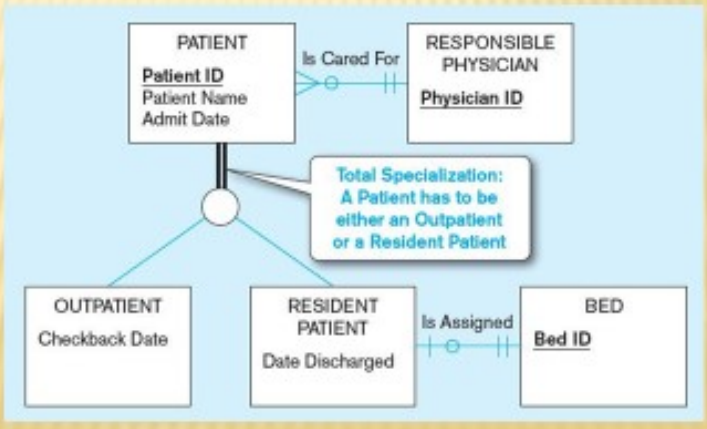
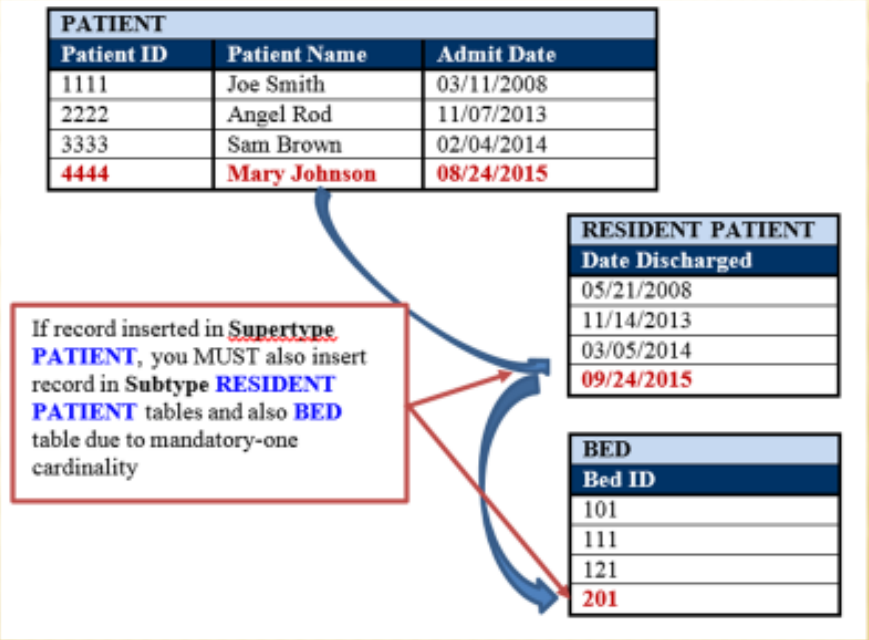


Figure 3-6 Completeness constraints using Patient example

a) Total specialization rule (cont.) – Scenario #2 (Cont.)



Partial Specialization Rule:

COMPLETENESS CONSTRAINT – PARTIAL SPECIALIZATION

- *Partial Specialization Rule*:
  - Each instance of a supertype *DOES NOT NEED TO BE* a member of ANY subtype
  - Other words, when you add a *record/data* to the supertype (table), you DON'T HAVE TO also add a record to any of the subtypes (tables) unless is required for complete a subtype.
- *E-R Diagram*:
  - Single line from supertype to circle

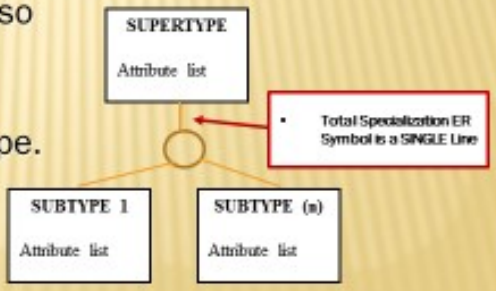


Figure 3-6 Completeness constraints using **Vehicle** example

a) **Partial specialization rule (cont.)** –
   - **Scenario #1** – Suppose we are registering or inserting a new **car** to the shop, then we would have to insert a record to both VEHICLE & CAR tables since a car is composed of both **supertype** and **subtype**:

VEHICLE				
Vehicle ID	**Price**	**Engine Displacement**	**Make**	**Model**
154QMJ6O88PX	35000	1.6	Toyota	Highlander
77EXPL620KZM	26000	1.3	Nissan	Rouge
QNX43923KHSL	45000	2.0	Acura	MDX

CAR
No Of Passengers
6
5
7
8

For a car, if record inserted in Supertype VEHICLE, you MUST also insert record in Subtype CAR. CAR is a subtype of VEHICLE thus you need to insert in BOTH tables.

Figure 3-6 Completeness constraints using **Vehicle** example

a) **Partial specialization rule (cont.)** –

- **Scenario #2** – Suppose we are registering or inserting a new **truck** to the shop, then we would have to insert a record to both **VEHICLE** & **TRUCK** tables since a truck is composed of both **supertype** and **subtype**:

VEHICLE				
Vehicle ID	**Price**	**Engine Displacement**	**Make**	**Model**
154QMJ6O88PX	35000	1.6	Toyota	Highlander
77EXPL620KZM	26000	1.3	Nissan	Rouge
T738RKL8400J	**55000**	**2.5**	**GMC**	**Sierra**

TRUCK	
Capacity	**Cab Type**
1	Regular
.5	Extended
1.5	**Crew**

For a **truck**, if record inserted in **Supertype VEHICLE**, you **MUST** also insert record in **Subtype TRUCK**. **TRUCK** is a subtype of **VEHICLE** thus you need to insert in BOTH tables.

46

Figure 3-6 Completeness constraints using **Vehicle** example

a) **Partial specialization rule (cont.)** –

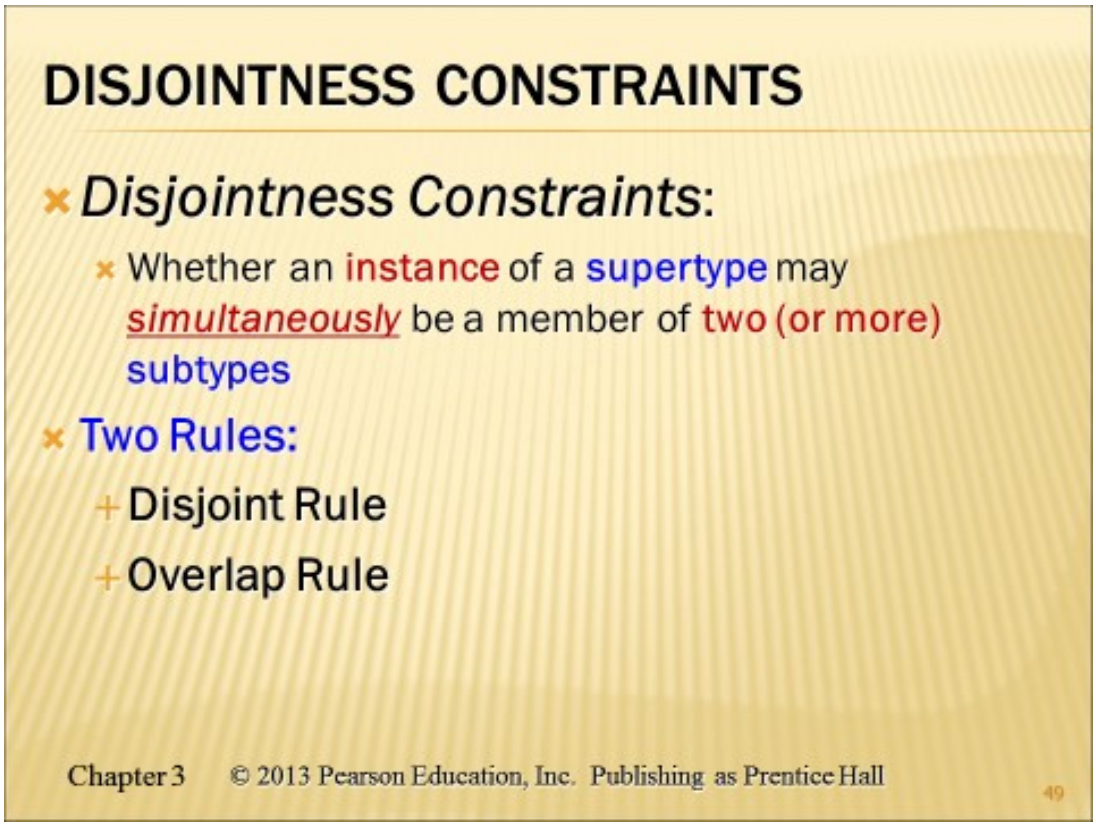
- **Scenario #3** – Suppose we are registering or inserting a new **motorcycle** to the shop, then we would need to insert a record to the **VEHICLE** table ONLY since Partial Specialization rule **DOES NOT** require that a **subtype** is involved:

VEHICLE				
Vehicle ID	**Price**	**Engine Displacement**	**Make**	**Model**
154QMJ6O88PX	35000	1.6	Toyota	Highlander
77EXPL620KZM	26000	1.3	Nissan	Rouge
PY56DR239IUC	**39000**	**3.5**	**Honda**	**VFR1200F**

For a **motorcycle**, there is NO subtype, so you only need to ADD a record to the **Supertype VEHICLE** only!

47

Disjointness Constraints



Disjoint Rule

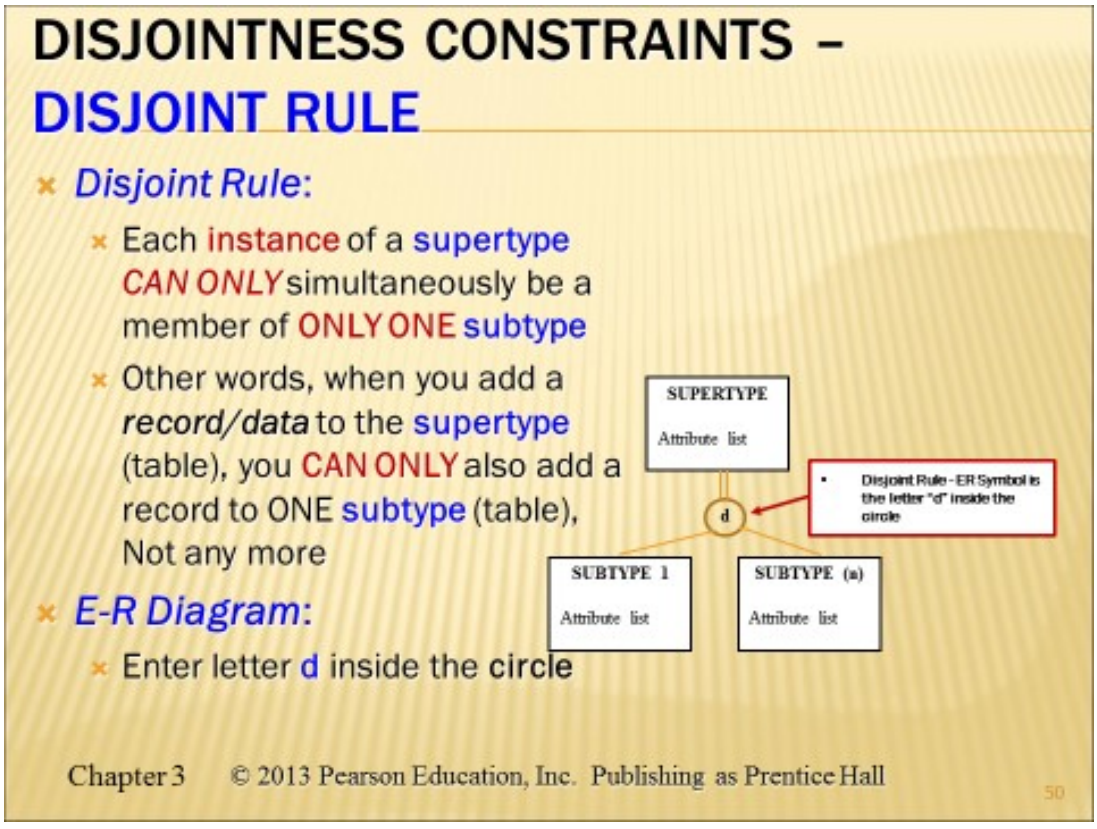


Figure 3-7 disjointness constraints using Patient example

a) **Disjoint rule** – A patient **MUST** be either an **outpatient** or a **resident patient** but not both. This means only ONE membership between **supertype** and **subtype**

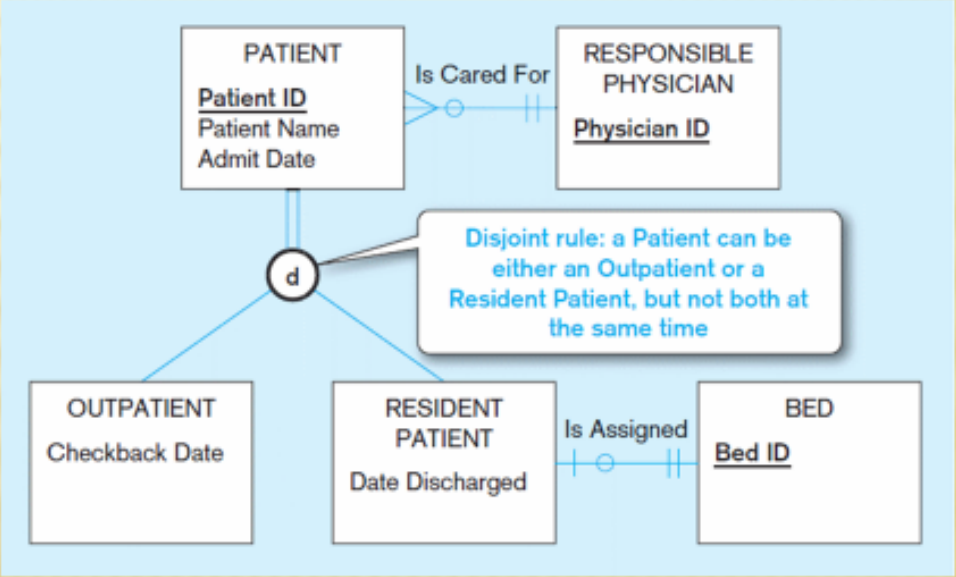


Figure 3-7 disjointness constraints using Patient example

a) Disjoint rule (cont.) –

- **Scenario #1 –** Suppose we are registering or inserting a new **outpatient** to the hospital, then we would have to also insert a record to both **PATIENT** & **OUTPATIENT** tables:

PATIENT		
Patient ID	**Patient Name**	**Admit Date**
1111	Joe Smith	03/11/2008
2222	Angel Rod	11/07/2013
3333	Sam Brown	02/04/2014
4444	**Mary Johnson**	**08/24/2015**

OUTPATIENT
Checkback Date
05/21/2008
11/14/2013
03/05/2014
09/24/2015

If record inserted in **Supertype PATIENT**, you MUST also insert record in **Subtype OUTPATIENT**

Figure 3-7 disjointness constraints using Patient example

a) Disjoint rule (cont.) –

- **Scenario #2 –** Suppose we are registering or inserting a new **resident patient** to the hospital, then we would have to also insert a record to **PATIENT** & **RESIDENT PATIENT** tables, but in addition the **BED** table since a Mandatory-one cardinality relationship exists between **RESIDENT PATIENT** table & the **BED** table:

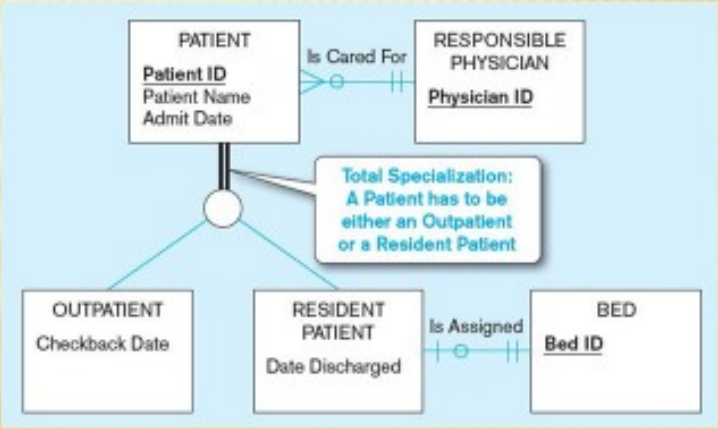


Figure 3-7 disjointness constraints using Patient example

a) Disjoint rule (cont.) – Scenario #2 (Cont.)

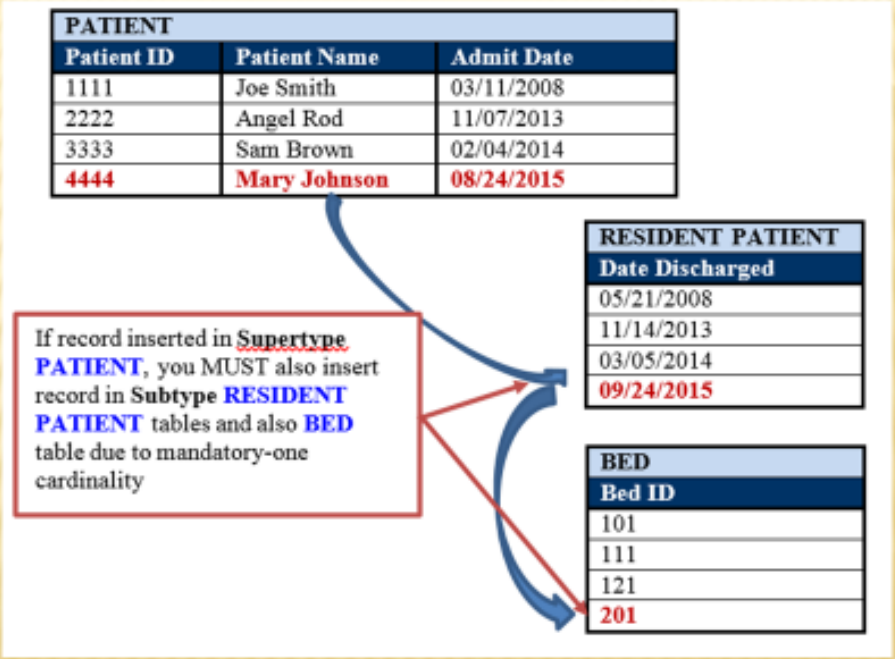
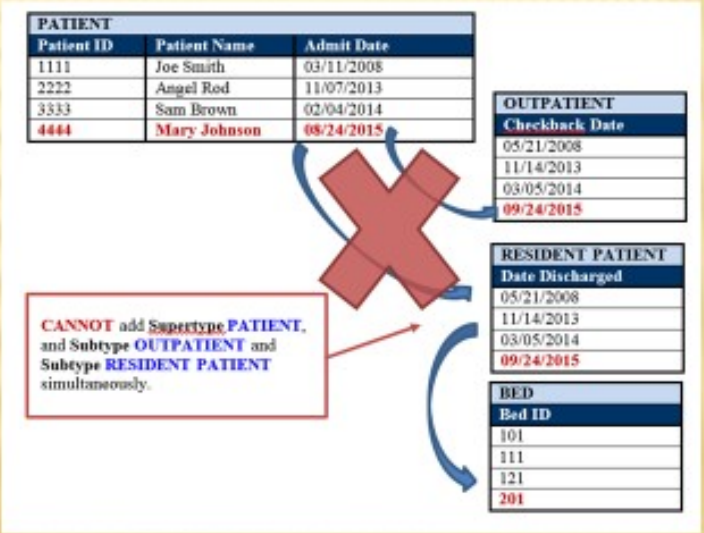


Figure 3-7 disjointness constraints using Patient example

a) Disjoint rule (cont.) –

- **Scenario #3** – Suppose want to register or inserting a new **outpatient** & resident patient simultaneously to the hospital? **Disjoint rule DOES NOT ALLOW THIS!**



Overlap Rule

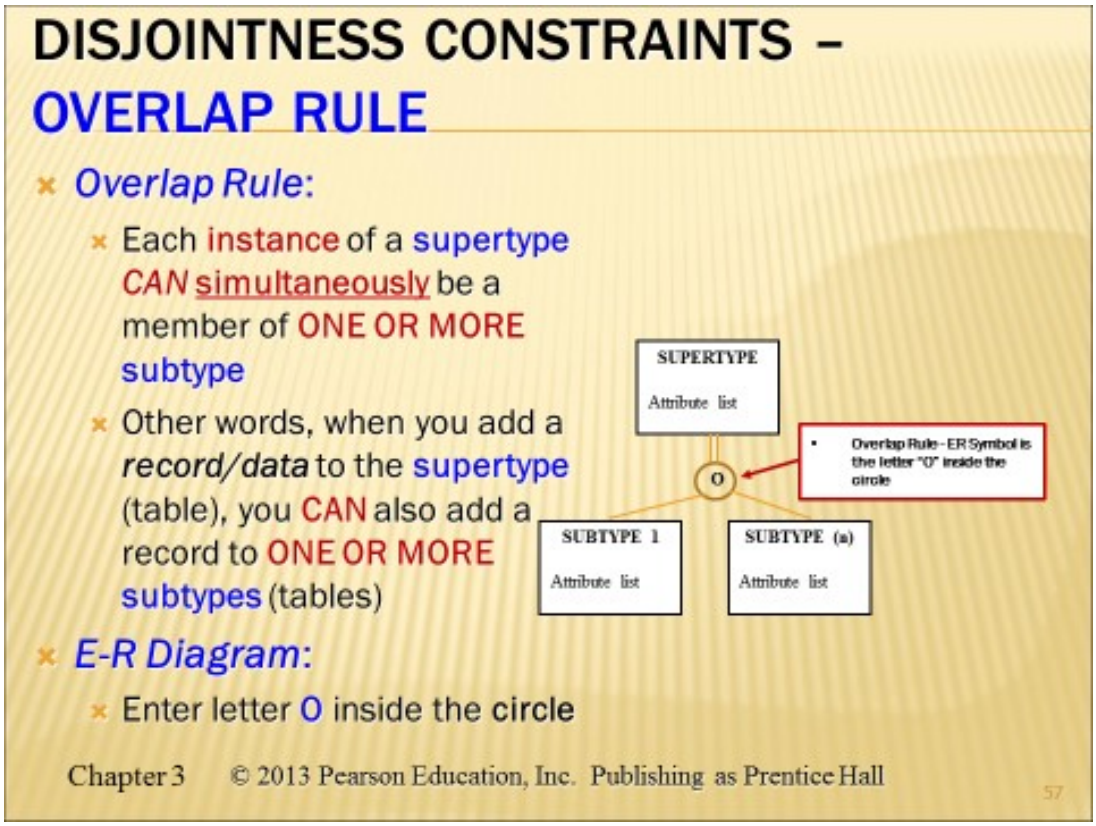
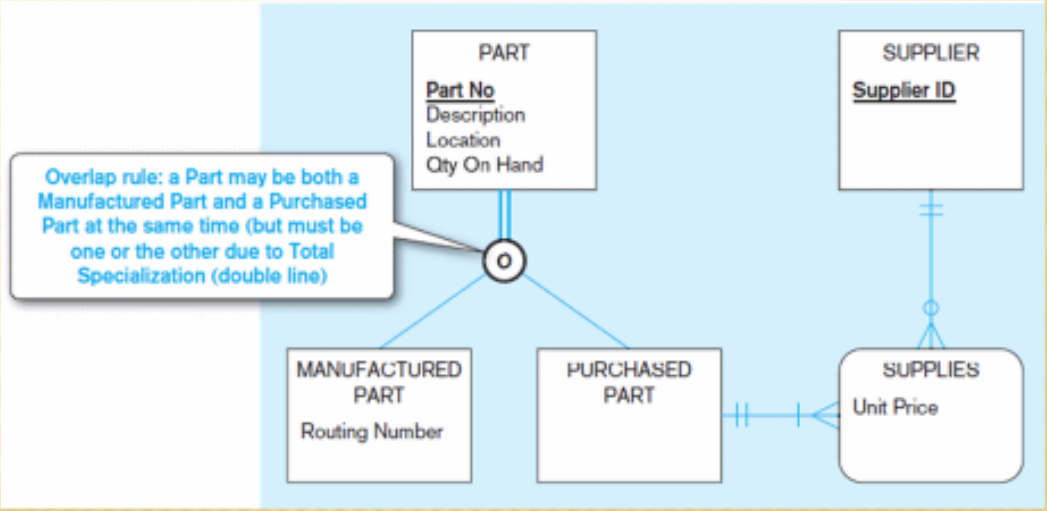


Figure 3-7 disjointness constraints using Manufacture example

a) **Overlap rule** – A part **CAN** be either a **Manufactured** or a **Purchased** part (Can be both simultaneously). This means MULTIPLE membership between **supertype** and **subtypes**



59

Figure 3-7 disjointness constraints using Patient example

b) Overlap rule (cont.) –

- **Scenario #1** – Suppose want to insert a new **PART**, MANUFACTURE PART & PURCHASED PART simultaneously? **YES - Overlap rule DOES NOT ALLOW THIS!**

CAN add Supertype PART, and Subtype MANUFACTURE PART and Subtype PURCHASED PART simultaneously.

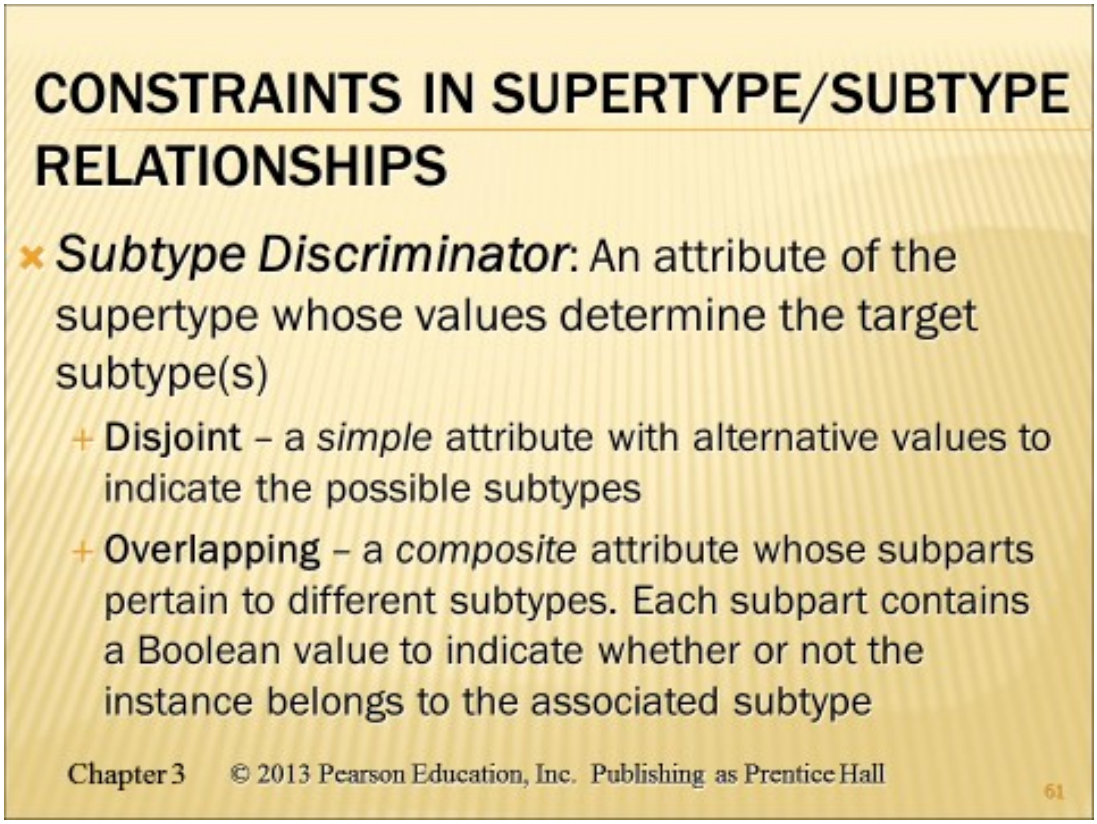
PART			
Part No	**Description**	**Location**	**Qty On Hand**
1000	Joe Smith	Chicago Warehouse	500
1001	Angel Rod	Michigan Warehouse	145
2001	Sam Brown	Virginia Warehouse	1134
2002	**Mary Johnson**	**Virginia Warehouse**	**50**

MANUFACTURED PART
Routing Number
1111
2222
2112
3456

PURCHASED PART
05/21/2008
11/14/2013
03/05/2014
09/24/2015

60

Subtype Discriminators



Disjoint Rule:

- Disjoint Subtypes:

  - Example shown below of previous Employee hierarchy.
  - In this example we have the following constraints:

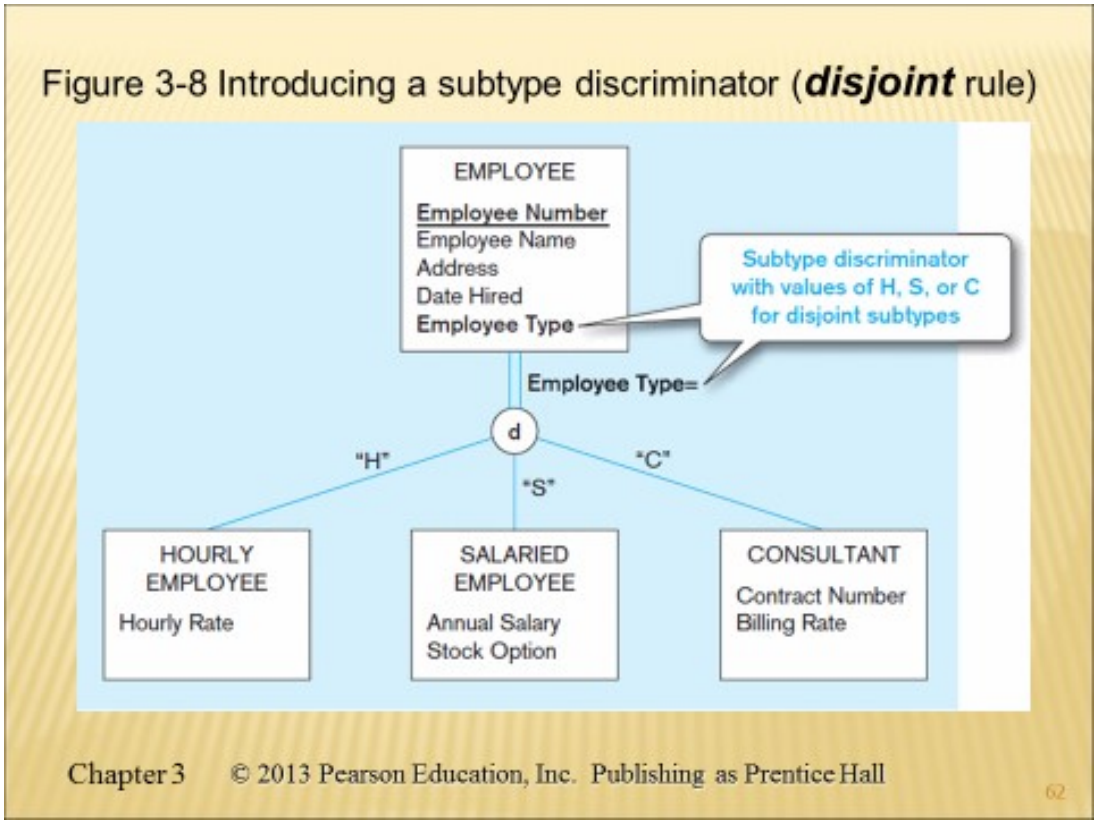
    - **Total Specialization** – Employee must be one of the following 3 types = Hourly, Salaried Employee or Consultant (cannot be just an Employee)
    - **Disjoint Rule** – Each Employee can ONLY BE ONE of the following = Hourly, Salaried Employee or Consultant cannot be any other combination.

  - In this example we have ADDED following **discriminator attribute**:

    - **Employee Type** – Stores the following values: "H" (for hourly employee), "S" (for salary employee) & "C" (for consultant)

  - Now we look at the scenario where we have to ADD a new EMPLOYEE, the following occurs:

    - A new INSTANCE or record is added to the **supertype EMPLOYEE** and based on the value of this **Employee Type** discriminator **attribute**, the appropriate **subtype** INSTANCE is also added.
    - For example if **Employee Type** = **"S"**, then an instance or record is added to both **supertype EMPLOYEE** and **subtype SALARIED EMPLOYEE** ONLY!



Overlapping Subtype Rule:

- Overlapping Subtypes & **discriminator attribute**:

  - In scenarios where subtypes can overlap or a supertype can have many instance of subtypes, then we need specialized version of the **discriminator attribute**
  - In the example shown below of previous parts hierarchy, we have the following constraints:

    - **Total Specialization** – Employee must be one of the following 3 types = Hourly, Salaried Employee or Consultant (cannot be just an Employee)
    - **Overlap Rule** – Each Employee CAN BE ANY OR ALL of the following = Manufacture part, Purchased Part or BOTH!

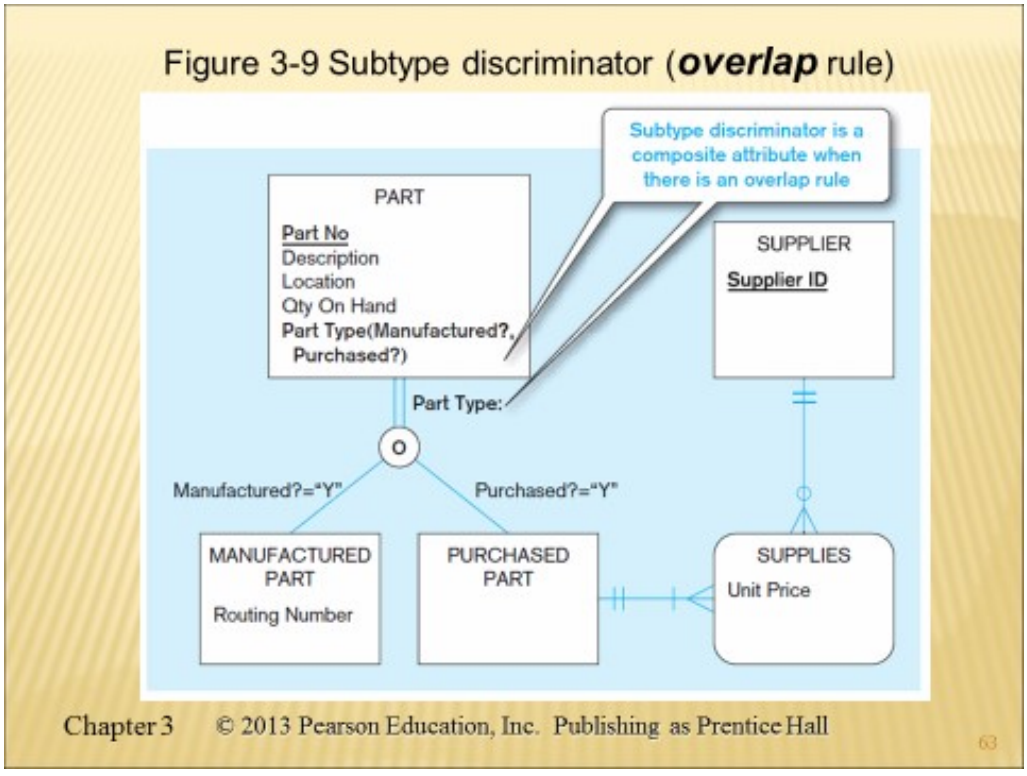
  - In this example we have ADDED following **COMPOSITE BOOLEAN discriminator attribute**:

    - **Employee Type (Manufactured? Purchased?)** – A **COMPOSITE BOOLEAN discriminator attribute** with TWO components **Manufactured? & Purchased?** (question means is Boolean, TRUE or FALSE, YES or NO)
    - Table below shows the possible combinations:

Type of Part	Manufactured?	Purchased?
Manufactured only	"Y"	"N"
Purchased only	"N"	"Y"
Purchased and manufactured	"Y"	"Y"

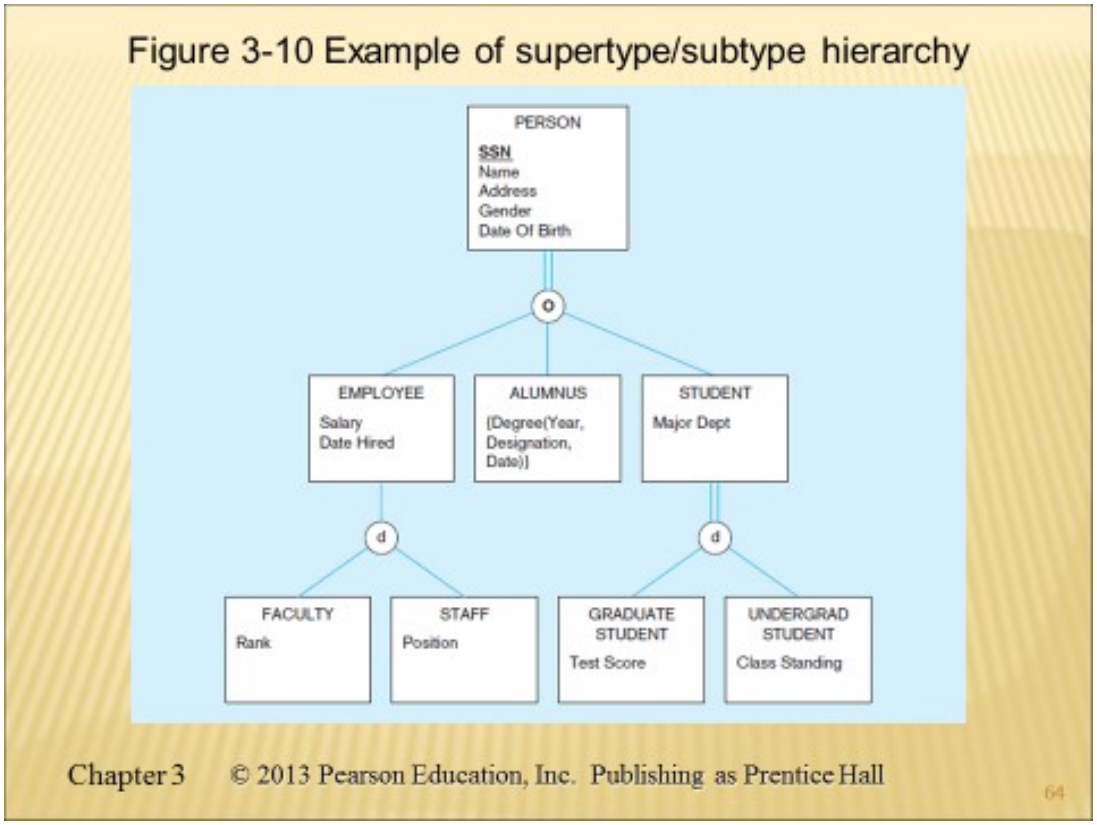
  - Now we look at the scenario where we have to ADD a new PART, the following occurs:

    - A new INSTANCE or record is added to the **supertype PART** and based on the value of both the values of **Manufactured? & Purchased? COMPOSITE BOOLEAN discriminator attribute**, the appropriate **subtype** INSTANCES is also added.
    - For example if **Manufactured** = **"Y"** and **Purchased** = **"Y"** then instance or record are added to ALL THREE **supertype PART**, **subtype PURCHASED PART** and **subtype MANUFACTURED PART**



3.2.5 Supertype & Subtypes Hierarchies

- We can also have scenarios where **Subtypes** can have their own **Subtypes** relationships. In this case the **Subtypes** is a **Supertype** to its own **Subtypes**.
  - See Example below:



3.2.6 EER Model Book Exercise Examples

- We now show examples from the book exercise section.

Example #9

Problem Statement

- Example 9 problem statement:

> 9. A bank has three types of accounts: checking, savings, and loan. Following are the attributes for each type of account:
>
> CHECKING: Acct No, Date Opened, Balance, Service Charge
> SAVINGS: Acct No, Date Opened, Balance, Interest Rate
> LOAN: Acct No, Date Opened, Balance, Interest Rate, Payment
>
> Assume that each bank account must be a member of exactly one of these subtypes. Using generalization, develop an EER model segment to represent this situation using the traditional EER notation, the Visio notation, or the subtypes inside supertypes notation, as specified by your instructor. Remember to include a subtype discriminator.

Solution

- Breaking down problem into parts:
  - Requirements state to use **generalization** – **Bottom-up approach**.
  - Steps:
    1. Identify existing ENTITIES
    2. Create **suppertype** using all shared attributes from existing ENTITIES
    3. Verify that **supertype/subtype** rules are met
    4. Apply inheritance
    5. Implement other requirements for specialization & constraints
    6. Address any other requirements & challenges

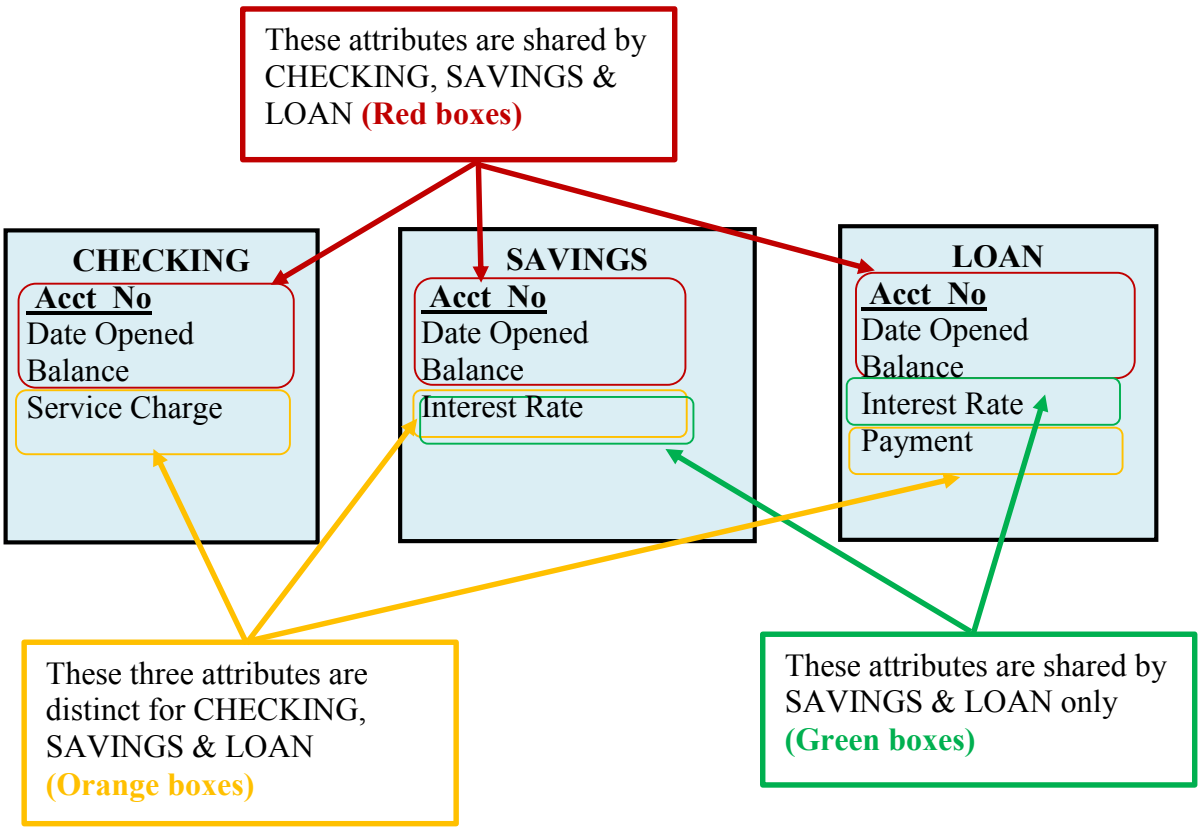
- Existing entities status:

CHECKING
Acct_No
Date Opened
Balance
Service Charge

SAVINGS
Acct_No
Date Opened
Balance
Interest Rate

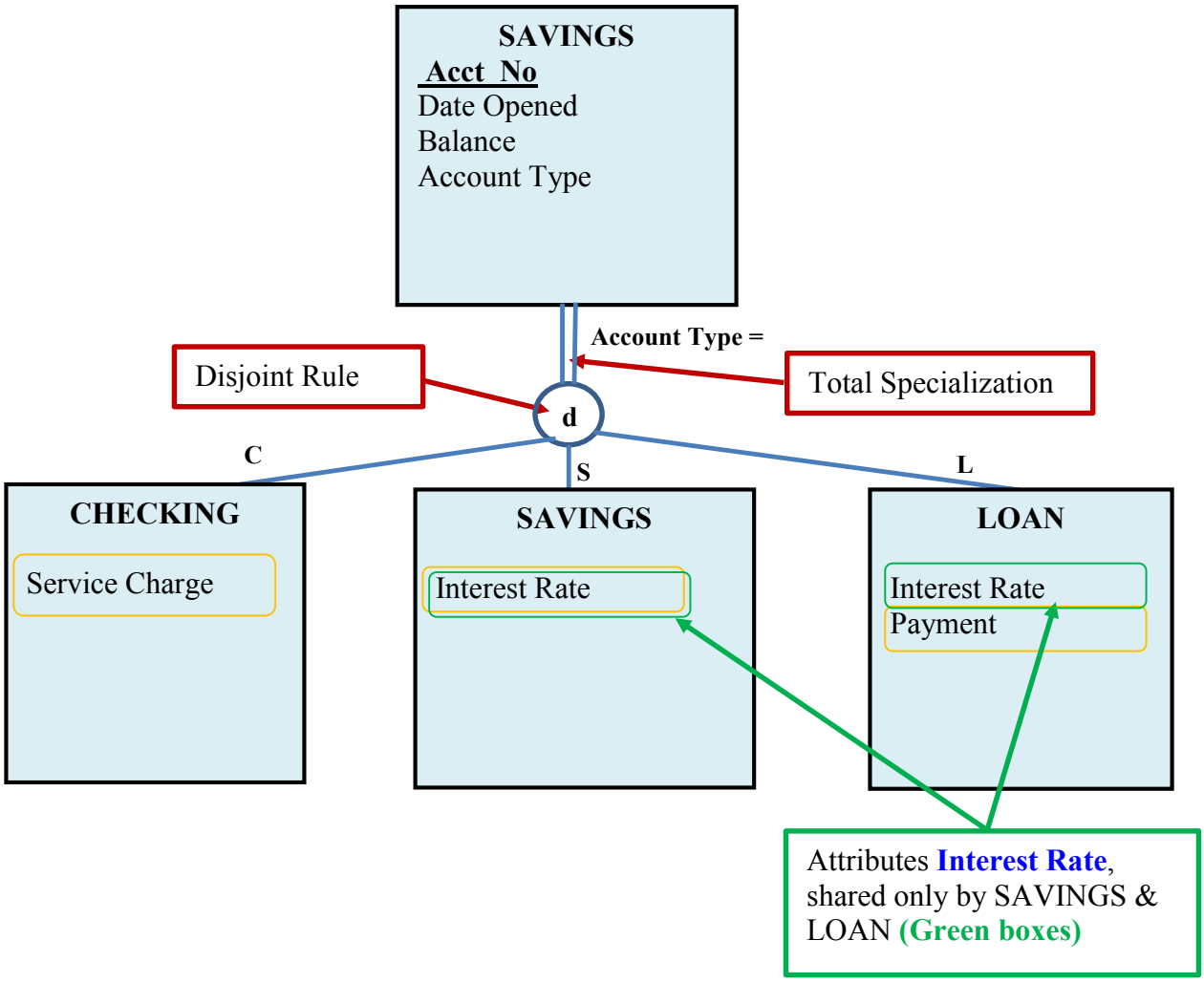
LOAN
Acct_No
Date Opened
Balance
Interest Rate
Payment

- We begin to apply the rules of supertype/subtypes:
  - Step 1 - Identify shared attributes between entities:
    - We have two groups of shared attributes for every ENTITY:
      - First group is between ***CHECKING***, ***SAVINGS*** & ***LOAN*** (Red)
      - Second group between ***SAVINGS*** & ***LOAN*** only. (Green) - We have a different situation here, where there are attributes only shared by two entity not all, which does not meet the requirements of RULE #2. Let's make a note of this and address later
  - Step 2 - Rule #2, ENTITY must contain unique attributes to qualify for Subtype:
    - We have attributes that are distinct for every ENTITY (Orange):
  - See diagram below:



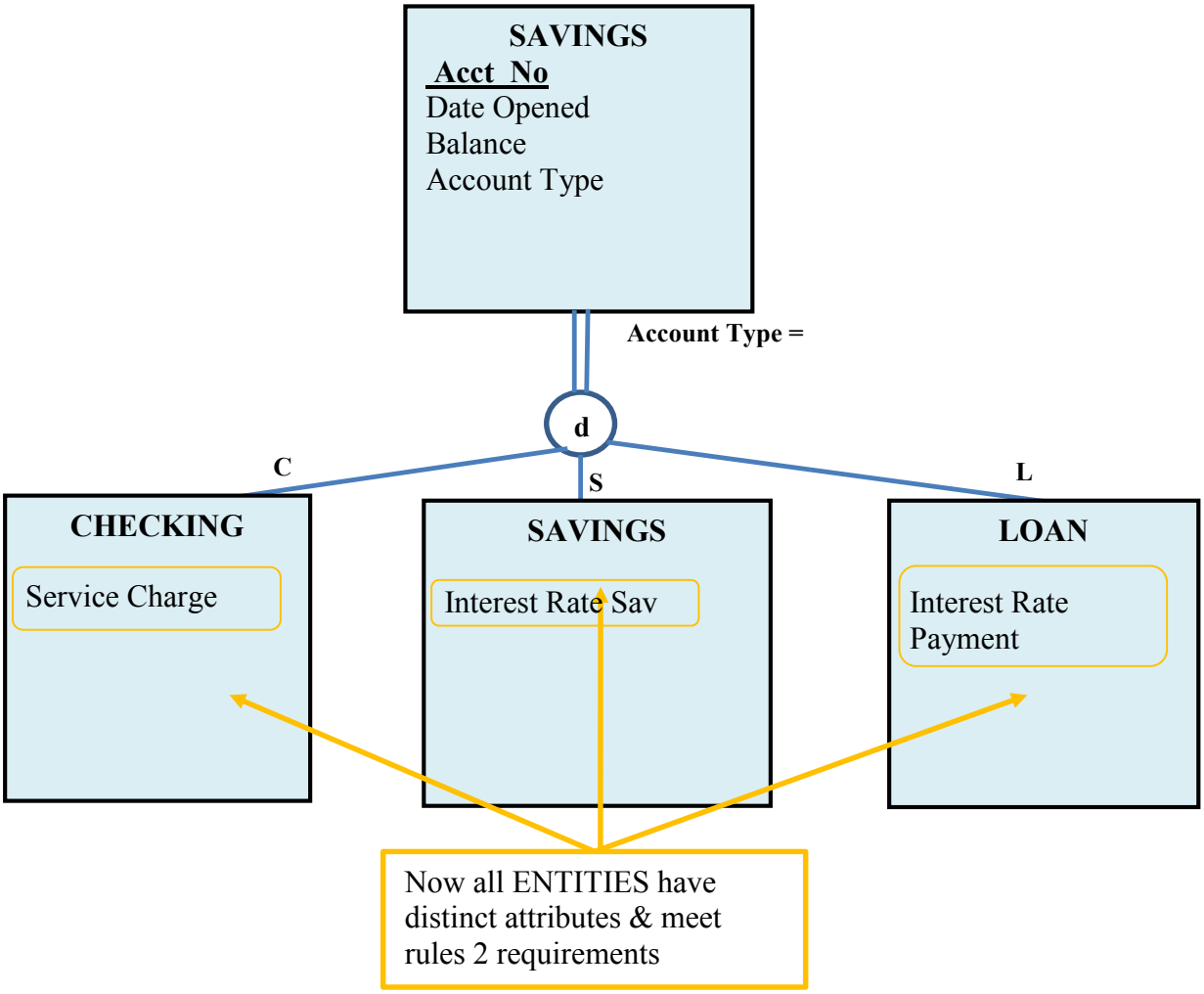
- We address constraints:
  - The requirements state that each bank acct must be a member of at least ***ONE*** of the subtypes:
    - **Completeness Constraint – Total Specialization**
    - **Disjoint rule applies**
  - Requirements state to use **Subtype Discriminator Attribute** & **alternative values** of possible subtype:
    - That means create an attribute in the supertype that will discriminate between accounts using a value to determine the account such as single letters:
      - We will call the **Subtype Discriminator Attribute** = **Account Type**
      - We will use the letters C, S & L as alternative values points

- We make first attempt to implement the hierarchy :
  - Adding these requirements results in diagram below:

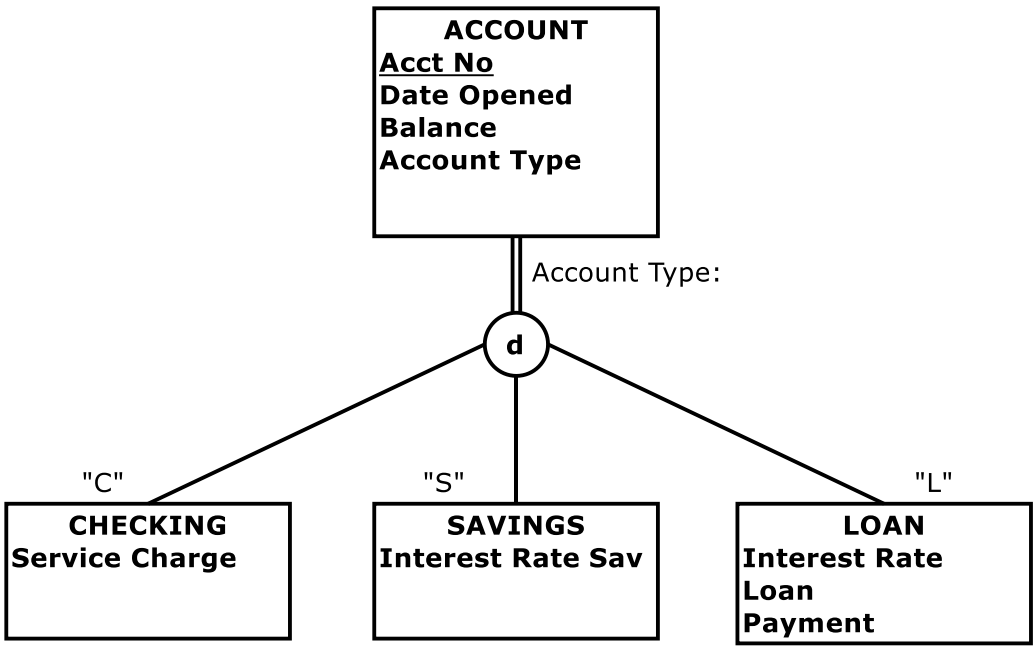


- We now address the challenge of what to do now with the attribute Interest Rate shared among SAVINGS & LOAN only :
  - We could think about adding another **supertype/subtype** hierarchy between SAVINGS & LOAN, but what would we call the supertype? There is really nothing in common with the SAVINGS entity and LOAN entity except that they are ACCOUNTS. Another **supertype/subtype** structure does not really work here right?
  - But if we analyze the business & make some assumption, a SAVING **Interest Rate** is NOT the same as a LOAN **Interest Rate** correct? They can be classified as different attributes.
  - So in this case, we could rename one of these interest rate to make them unique to their respective ENTITY and in this case it does work and we meet the rules requirements.
  - So below diagram renames the SAVING **Interest Rate** to **Interest Rate Sav** and we leave the LOAN **Interest Rate** as is.

- Results show that we now have distinct attributes in every Subtype:



- Official final EER diagram from book's teacher's manual for *Bank situation, standard EER Notation:*



Example #12

Problem Statement

- Example 12 problem statement:

12. Draw an EER diagram for the following problem using this text's EER notation, the Visio notation, or the subtypes inside supertypes notation, as specified by your instructor.

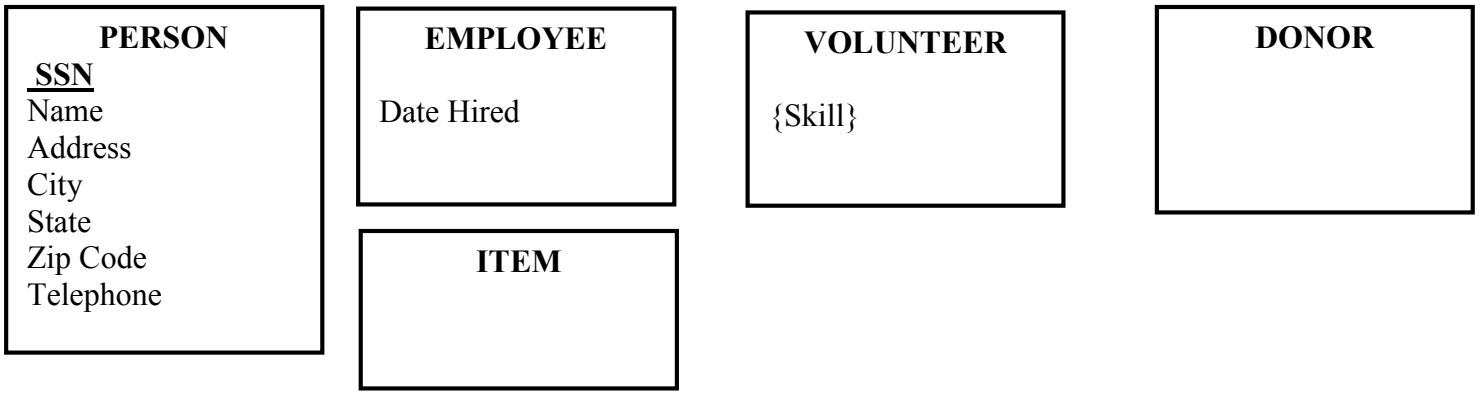
A nonprofit organization depends on a number of different types of persons for its successful operation. The organization is interested in the following attributes for all of these persons: SSN, Name, Address, City/State/Zip, and Telephone. Three types of persons are of greatest interest: employees, volunteers, and donors. Employees have only a Date Hired attribute, and volunteers have only a Skill attribute. Donors have only a relationship (named Donates) with an Item entity type. A donor must have donated one or more items, and an item may have no donors, or one or more donors.

There are persons other than employees, volunteers, and donors who are of interest to the organization so that a person need not belong to any of these three groups. On the other hand, at a given time a person may belong to two or more of these groups (e.g., employee and donor).

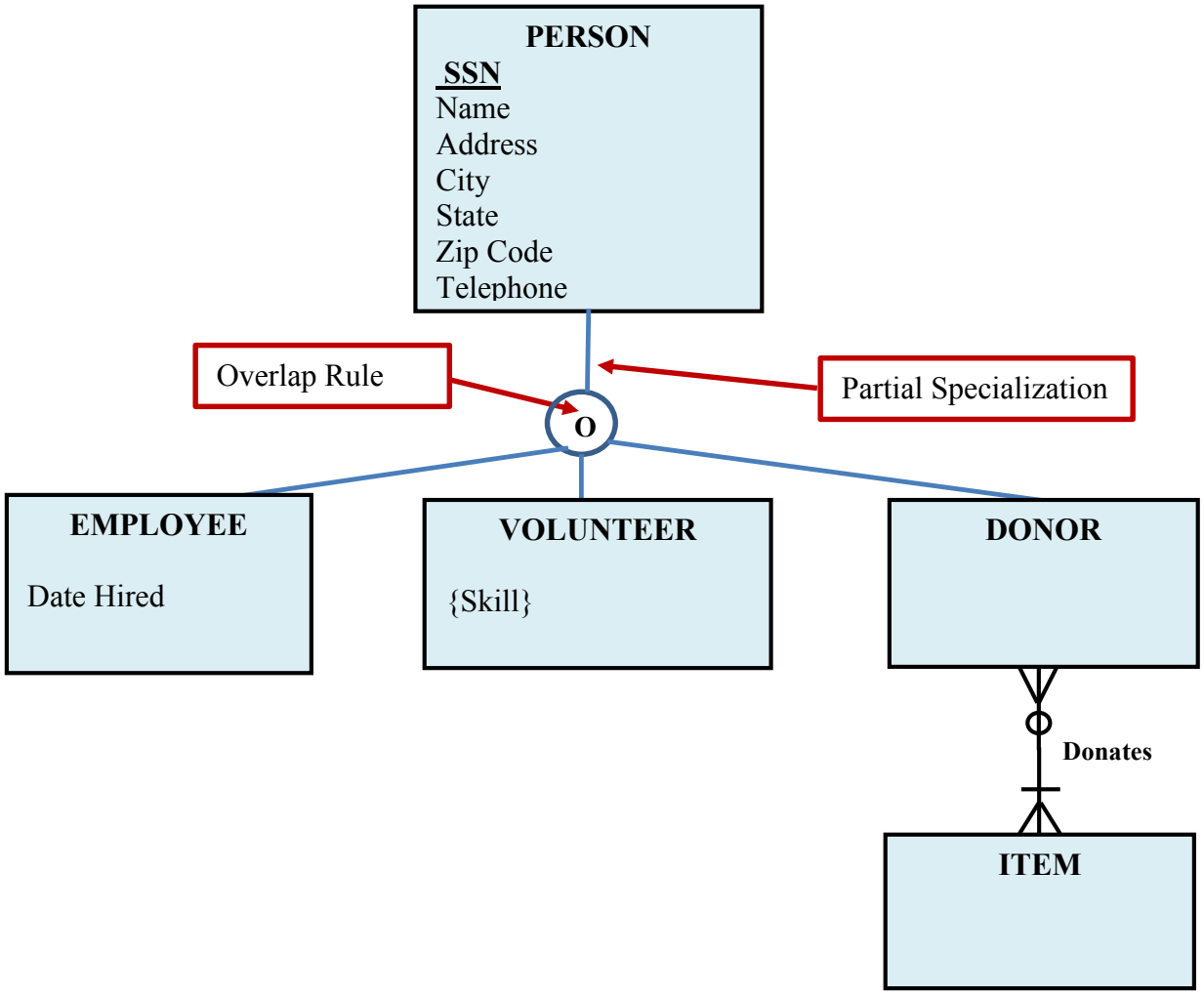
Solution

- Breaking down problem into parts:
  - Problem statement immediately provide information on a supertype PERSON. Therefore we are dealing with **Specialization – Top-Down approach**.
  - Also provides attributes that are shared for all person in the company, or PERSON entity
  - Also provides the persons of **GREATEST** interest or subtype entities EMPLOYEE. VOLUNTEER & DONOR. In addition their respective UNIQUE ATTRIBUTES. One of the unique attribute is Skill which hints mulita-valued attributes.
  - So far the rules of **supertype/subtype** hierarchy have been provided.
  - Note that by stating these are of **GREATEST** interest it implies there can be other persons, thus HINTS of PARTIAL SPECIALIZATION.
  - To confirm **PARTIAL SPECIALIZATION** it states ***"There are other persons of interest… and a person need not to belong to any of these groups"***.
  - In addition, it stats "a person may belong to two or more of these groups (e.g., employee and donor) thus indicates **OVERLAP RULE**.
  - Finally, statement indicates a relationship (DONATES) between DONOR and another entity ITEM, with cardinality rules applied.
  - Steps:
    1. Identify all existing ENTITIES
    2. Create **suppertype** from information provide
    3. Create **supertype/subtype** hierarchy with **subtypes** provided
    4. Apply inheritance
    5. Verify that **supertype/subtype** rules are met
    6. Implement other requirements for specialization & constraints
    7. Address any other requirements & challenges

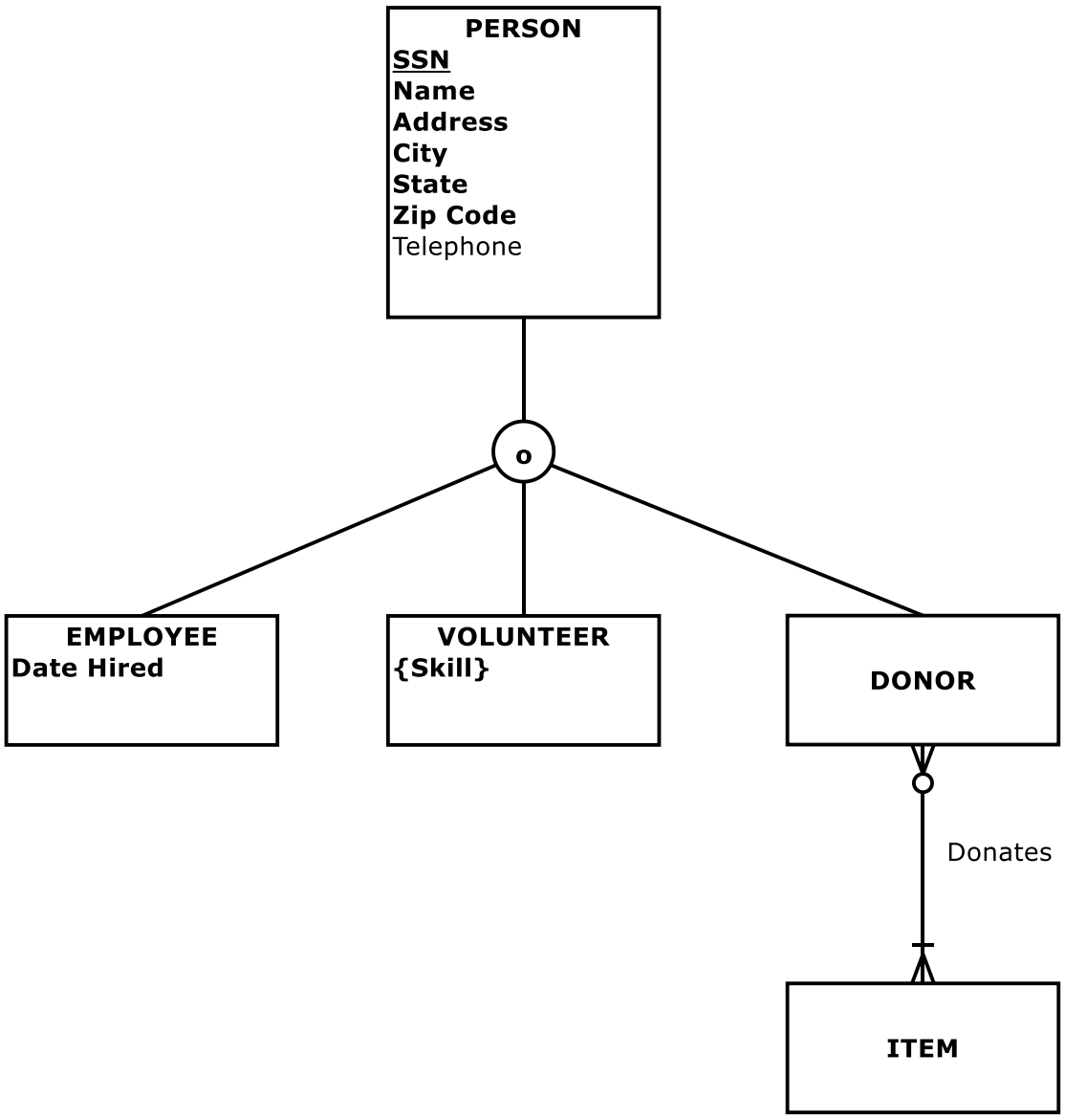
- We draw existing entities & their attributes. Note that we assume that in the entity VOLUNTEER Skill is a multi-valued attribute:



- In this TOP-DOWN-APPROACH we apply the provided rules of **supertype/subtypes**.
- In addition we now know we are dealing with **PARTIAL SPECIALIZATION** and **OVERLAP RULE .**
- Finally the relationship DONATES between DONOR and ITEM, thus we end up with the following diagram:



- Official final EER diagram from book's teacher's manual:



Example #13 (Continuation of problem 12)

Problem Statement

- Example 13 problem statement:

> **13.** Add a subtype discriminator (named Person Type) to the diagram you created in Problem and Exercise 12.

Solution

- Breaking down problem into parts:

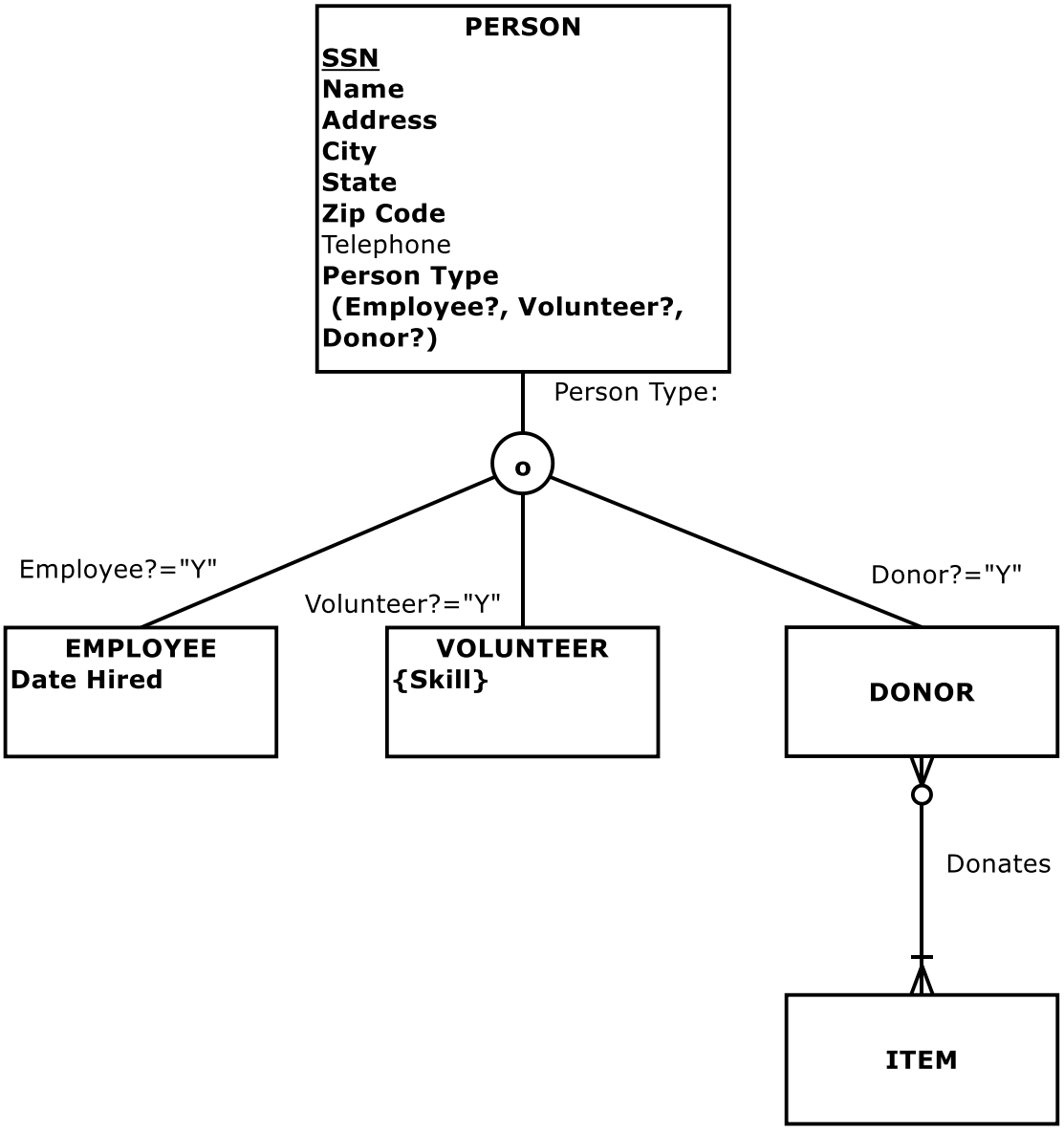
  - We are now extending the EER diagram by adding a **Subtype Discriminator Attribute** & **alternative values**.
  - As per class lecture, we will need a **COMPOSITE** **BOOLEAN** **discriminator attribute** such as Boolean, TRUE or FALSE, YES or NO).
  - Table below shows the possible combination:

Type of person	Employee?	Volunteer?	Donor?
Employee only	Y	N	N
Employee & Volunteer only	Y	Y	N
Employee & Donor only	Y	N	Y
Donor only	N	N	Y
Donor & Volunteer	N	Y	Y
Volunteer Only	N	Y	N
Employee, Volunteer & Donor	Y	Y	Y

  - Now we look at the scenario where we have to ADD a new **PERSON**, the following scenario can occurs:

    - For example if **Employee** = **"Y"**, **Volunteer** = **"Y"** and **Donor** = **"Y"** then instance or record are added to ALL FOUR ENTITIES **supertype PERSON**, **subtype EMPLOYEE**, **subtype VOLUNTEER**, and **subtype subtype DONOR**.

- Official final EER diagram from book's teacher's manual:



Example #14

Problem Statement

- Example 14 problem statement:

14. Develop an EER model for the following situation, using the traditional EER notation, the Visio notation, or the subtypes inside supertypes notation, as specified by your instructor:

A technology company provides offerings to its customers. Offerings are of two separate types: products and services. Offerings are identified by an offering ID and an attribute of description. In addition, products are described by product name, standard price, and date of first release; services are described by name of the company's unit responsible for the service and conditions of service. There are repair, maintenance, and other types of services. A repair service has a cost and is the repair of some product; a maintenance service has an hourly rate. Fortunately, some products never require repair. However, there are many potential repair services for a product. A customer may purchase an offering, and the company needs to keep track of when the offering was purchased and the contact person for that offering with the customer. Unfortunately, not all offerings are purchased. Customers are identified by customer ID and have descriptive data of name, address, and phone number. When a service is performed, that service is billed to some customer. Because some customers purchase offerings for their clients, a customer may be billed for services he or she did not purchase, as well as for ones that were purchased. When a customer is billed for a service (although some may never require a service of any type), the company needs to keep track of the date the service was performed, the date the bill is due, and the amount due.

Solution

- We read the problem statement & break it into parts:

A technology company provides offerings to its customers.

- We have identified the following possible ENTITIES

OFFERING	CUSTOMER	COMPANY
??	??	??

- We continue reading

Offerings are of two separate types: products and services.

- We have identified the following possible ENTITIES and know they are TYPES of OFFERING entity thus we have a **supertype/subtype** hierarchy:

PRODUCT	SERVICE
??	??

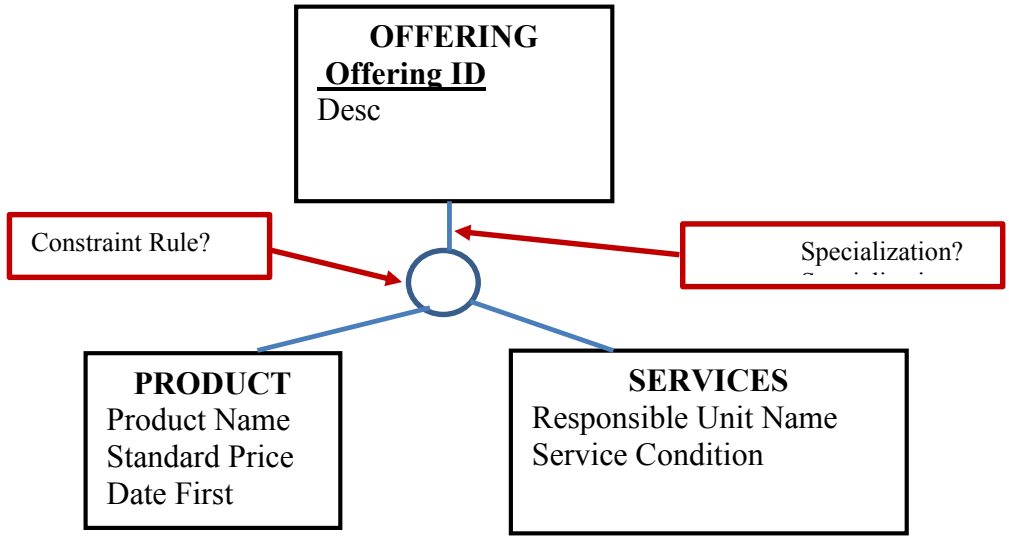
- But we don't have the Specialization or Constraints rules yet.

Offerings are identified by an offering ID and an attribute of description. In addition, products are described by product name, standard price, and date of first release; services are described by name of the company's unit responsible for the service and conditions of service.

- In addition, attribute information for the OFFERING, PRODUCT & SERVICE entities is provide so we modify the following ENTITIES based on attribute descriptions:

OFFERING	PRODUCT	SERVICES
Offering ID	Product Name	Responsible Unit Name
Desc	Standard Price	Service Condition
	Date First	

- Here is what the EER diagram looks at the moment:



- We continue reading

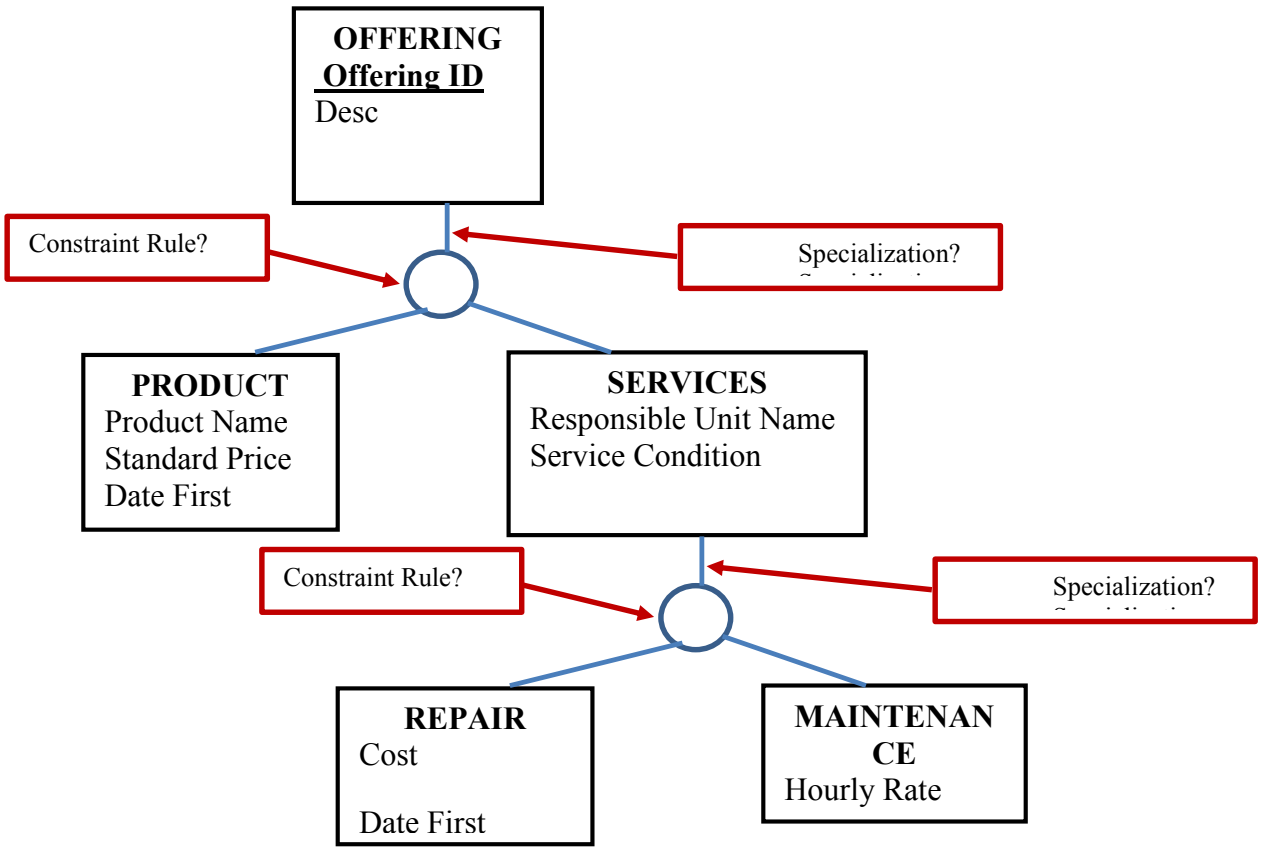
> There are repair, maintenance, and other types of services.

- We have identified the following possible ENTITIES based on the 3 nouns in sentence:

- In addition, these ENTITIES are TYPES OF SERVICES ENTITY, so now we have a supertype/subtype hierarchy under SERVICES. SERVICE is the supertype and MAINTENANCE & REPAIR the subtypes.
- In addition, attribute information for the MAINTENANCE & REPAIR entities is provide so we modify the following ENTITIES based on attribute descriptions:

> A repair service has a cost and is the repair of some product; a maintenance service has an hourly rate.

- Here is what the EER diagram looks at the moment:



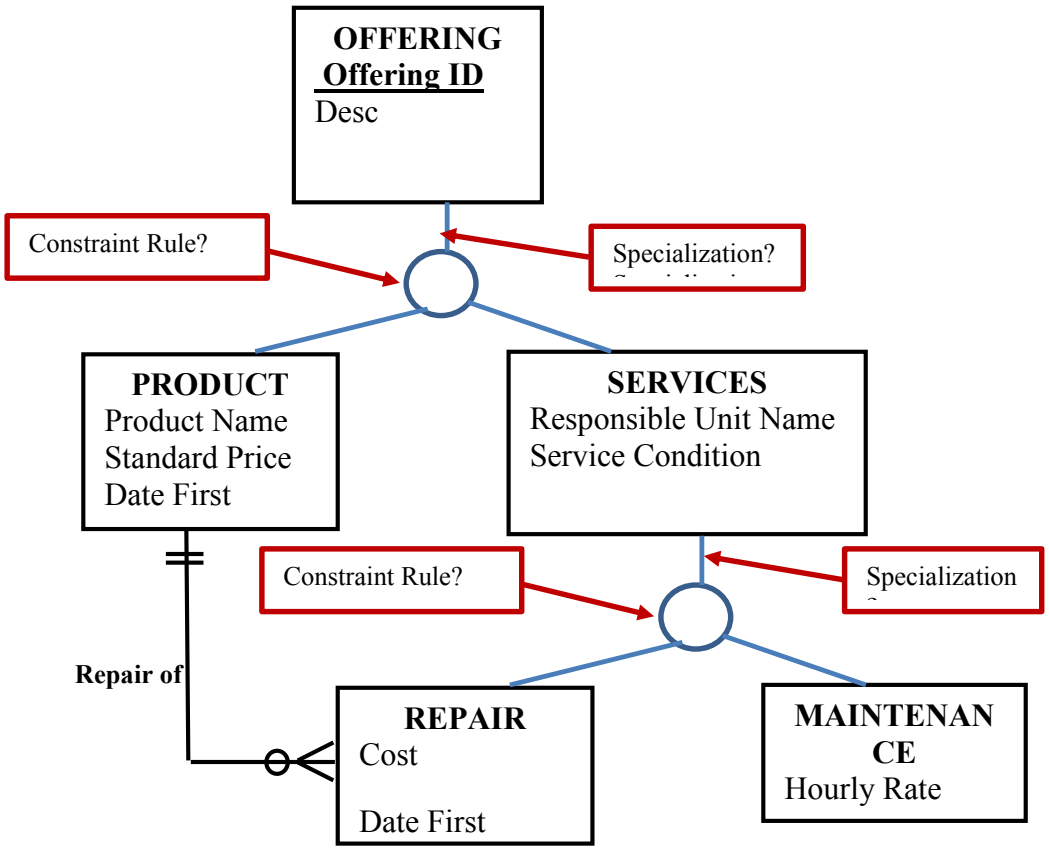
- But very important we have also identified a relationship between REPAIR & PRODUCT ENTITIES with the sentence " A repair service has the cost and is **the repair of some product**". The relationship is REPAIR ***repair of*** a PRODUCT or ***repair of***.

  - Relationship identified = ***repair of*** between REPAIR & PRODUCT
  - Also possible cardinality with the word "some product", so is identifying one product is being repaired.

> Fortunately, some products never require repair. However, there are many potential repair services for a product.

- Based on statement, now we have identified another cardinality within this REPAIR & PRODUCT ***repair of*** relationship:
  - Between PRODUCT to REPAIR = zero to many
  - Between REPAIR to PRODUCT = one to one

- Here is the EER diagram with the relationship between PRODUCT & REPAIR at the moment:



- We continue reading the problem statement:

A customer may purchase an offering, and the company needs to keep track of when the offering was purchased and the contact person for that offering with the customer.

- We have identified another relationship between CUSTOMER & OFFERING ENTITIES with the sentence " A customer may purchase an offering.
- In addition, we have also identified RELATIONSHIP ATTRIBUTES, since we need to keep track of date of purchases & contact person for offering.
- Here is what we have:
  - Relationship identified = ***purchase*** between CUSTOMER & OFFERING
  - Relationship Attributes = Purchase Date & Contact.

- We continue reading to identify any possible cardinalities::

Unfortunately, not all offerings are purchased.

- We have identified possible cardinalities (not much information is provided at this point so I will make some assumptions):
  - Between CUSTOMER to OFFERING = zero to many?
  - Between OFFERING to CUSTOMER = zero to many?

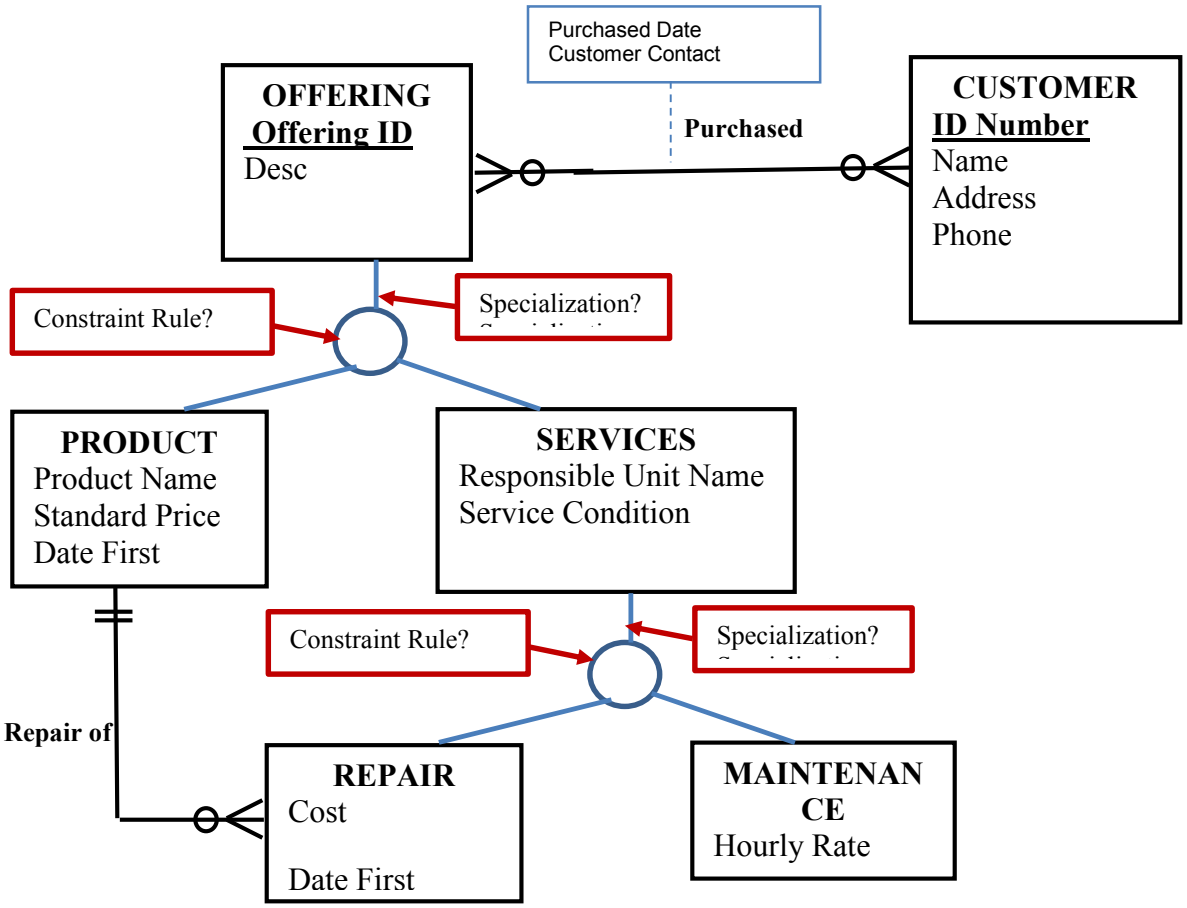
- We continue reading to get more information:

are purchased. Customers are identified by customer ID and have descriptive data of name, address, and phone number.

- Attribute information for the CUSTOMER entity is provide so we modify the following ENTITIES based on attribute descriptions:

CUSTOMER
ID Number
Name
Address
Phone

- Here is the EER diagram with the relationship between CUSTOMER & OFFERING at the moment:



- We continue reading to get more information:

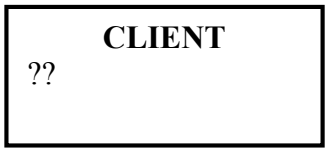
When a service is performed, that service is billed to some customer.

- We have identified another relationship between CUSTOMER & SERVICE ENTITIES with the sentence " billed to **some** customer".
- Also possible cardinality with the word "**some** customer" , may indicate a MANY cardinality.
- Here is what we have:
  - Relationship identified = ***billed to*** between CUSTOMER & SERVICE
  - Cardinality = Many Cardinality between SERVICE & CUSTOMER

- We continue reading to get more information on this relationship:

Because some customers purchase offerings for their clients, a customer may be billed for services he or she did not purchase, as well as for ones that were purchased.

- Based on sentence, we may have a possible new entity CLIENT:



- Also the following business rule is difficult to decode. Let's hold for now and continue reading:

a customer may be billed for services he or she did not purchase, as well as for ones that were purchased.

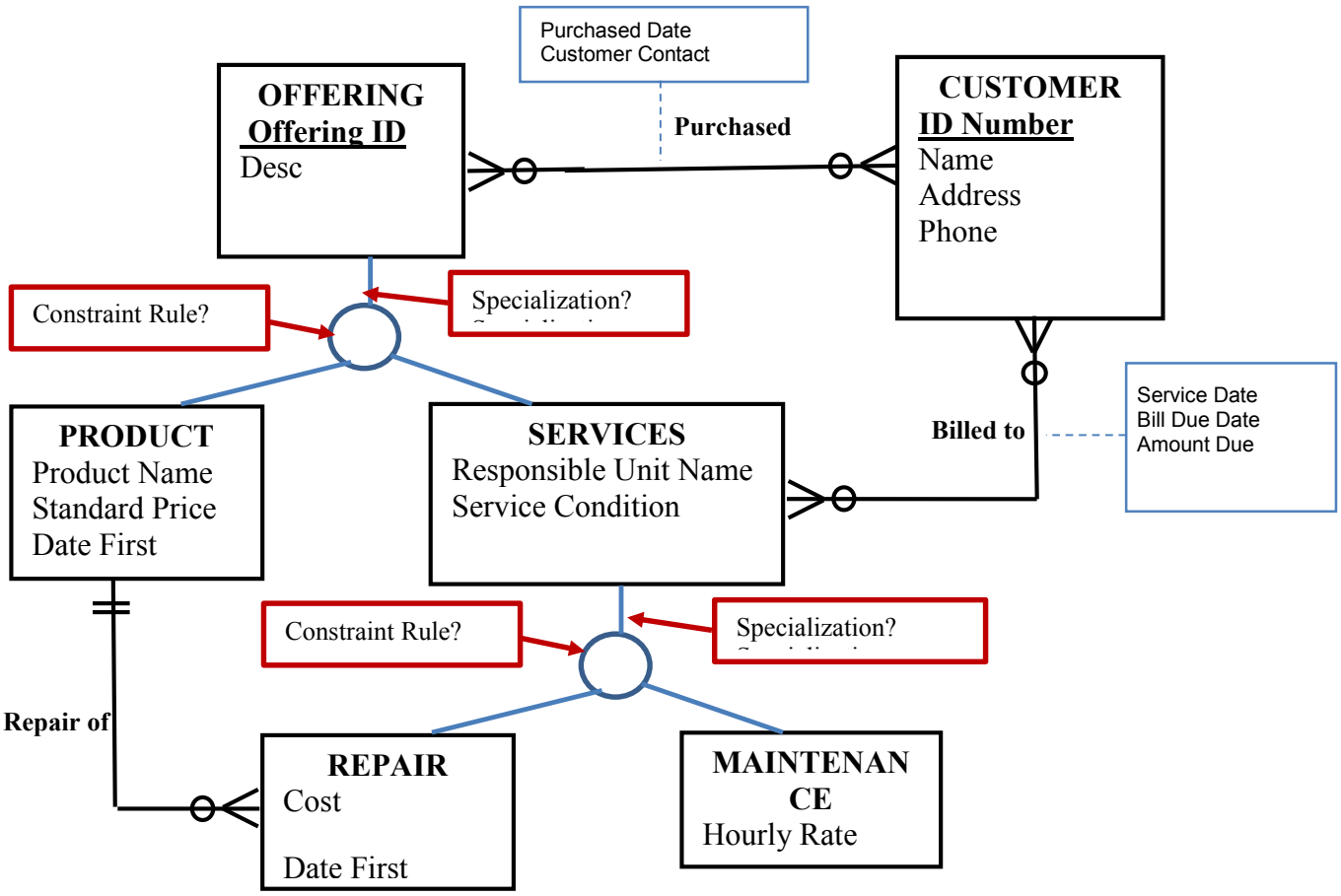
- We continue reading to get more information:

When a customer is billed for a service (although some may never require a service of any type), the company needs to keep track of the date the service was performed, the date the bill is due, and the amount due.

- From above statement we have identified:
  - Relationship attributes for the ***billed to*** relationship between CUSTOMER & SERVICE. The attributes are ***Service Date***, ***Bill Due Date***, ***Amount Due***.
  - In addition, we identified a cardinality: Zero to many between CUSTOMER to SERVICE because some customers may never require a service.

- Here is all the information we have compiled for the ***billed to*** relationship between CUSTOMER & SERVICE:
  - Cardinality = Zero-to-Many?? Between SERVICE & CUSTOMER
  - Cardinality = Zero to Many between CUSTOMER & SERVICE
  - The following relation attributes: ***Service Date***, ***Bill Due Date***, ***Amount Due***

- Here is the EER diagram with the relationship between CUSTOMER & SERVICE at the moment:



- We are almost done, but we have several incomplete items:
  1. **ISSUE #1** –
     a) The **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between OFFERING, SERVICE & PRODUCT has not been defined.
     b) The statement does not indicate if we should add a **Subtype Discriminator** to the solution for the **supertype/subtype** hierarchy between OFFERING, SERVICE & PRODUCT

  2. **ISSUE #2** –
     a) The **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between SERVICE, REPAIR & MAINTENANCE has not been defined.
     b) The statement does not indicate if we should add a **Subtype Discriminator** to the solution for the **supertype/subtype** hierarchy between SERVICE, REPAIR & MAINTENANCE

  3. **ISSUE #3** – The following business rule was difficult to decode for the relationship ***billed to*** relationship between CUSTOMER & SERVICE and we need to make sure is implemented:

> Because some customers purchase offerings for their clients, a customer may be billed for services he or she did not purchase, as well as for ones that were purchased.

IMPLEMENTATION:

- So we go back and read the requirements for the supertype/subtype hierarchies to identify the rules

Addressing Issue #1 - The **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between OFFERING, SERVICE & PRODUCT**:**

- The following statements may provide hint or reason for us to make assumptions:

> Offerings are of two separate types: products and services.
>
> Offerings are identified by an offering ID and an attribute of description. In addition, products are described by product name, standard price, and date of first release; services are described by name of the company's unit responsible for the service and conditions of service.

- Based on the statements I would assume the following:
  - Offerings are two separate types – this would indicate that only products and services are the offerings.
  - But most important reviewing the attribute to the OFFERING entity, would indicate that it will never make sense to have an OFFERING instance on its own. It does not provide enough information. You would need a combination of OFFERING & SERVICE or OFFERING & PRODUCT to really get anything done.
  - Finally Subtype Discriminators are needed since when inserting and instance of a supertype we would need to know which of the subtypes (if any) should this instance be inserted.
  - So we will use the following discriminator attribute = Offering Type
  - And for values we will use = "P" for PRODUCT & "S" for SERVICE.

- Conclusion for the **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between OFFERING, SERVICE & PRODUCT has not been defined:
  - **Total Specialization** – OFFERING cannot be alone, must be combined with PRODUCT or SERVICE
  - **Disjoint Rule** – since an OFFERING cannot be both a SERVICE & PRODUCT since it states there are two **separate** types.
  - **Subtype Discriminator attribute** = Offering Type
  - **Discriminator value** = "P" for PRODUCT & "S" for SERVICE

Addressing Issue #2 - The **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between SERVICE, REPAIR & MAINTENANCE**:**

- The following statements may provide hint or reason for us to make assumptions:

There are repair, maintenance, and other types of services.

A repair service has a cost and is the repair of some product; a maintenance service has an hourly rate.

Fortunately, some products never require repair. However, there are many potential repair services for a product.

- Based on the statements above I would assume the following:
  - The statement "other types of services" indicates there is something else.
  - Initially I assumed there may be another ENTITY:

OTHER?
??

  - But no other description of another service type entity was given. Therefore we would have to assume that there are SERVICES that are NOT part of PART & MAINTENANCE.
  - Therefore the SERVICE entity CAN STAND ON ITS OWN, thus we conclude this could be **PARTIAL SPECIALIZATION**.
  - Also as SERVICE would not makes if it would be BOTH a REPAIR & MAINTENANCE these must be totally separate services, therefore I would assume that for the Constraint Rule we have **DISJOINT RULE.**
  - Finally Subtype Discriminators are needed. So we create discriminator attribute "Service Type" & use "R" for REPAIR & "M" for MAINTENANCE.

- Conclusion for the **Specialization** rules and **Constraint Rules** for the **supertype/subtype** hierarchy between SERVICE, REPAIR & MAINTENANCE:
  - **Partial Specialization**
  - **Disjoint Rule**
  - **Subtype Discriminator attribute** = Service Type
  - **Discriminator value** = "R" for REPAIR & "M" for MAINTENANCE

Addressing Issue #3 – Business rule **in** relationship ***billed to*** relationship between CUSTOMER & SERVICE**:**

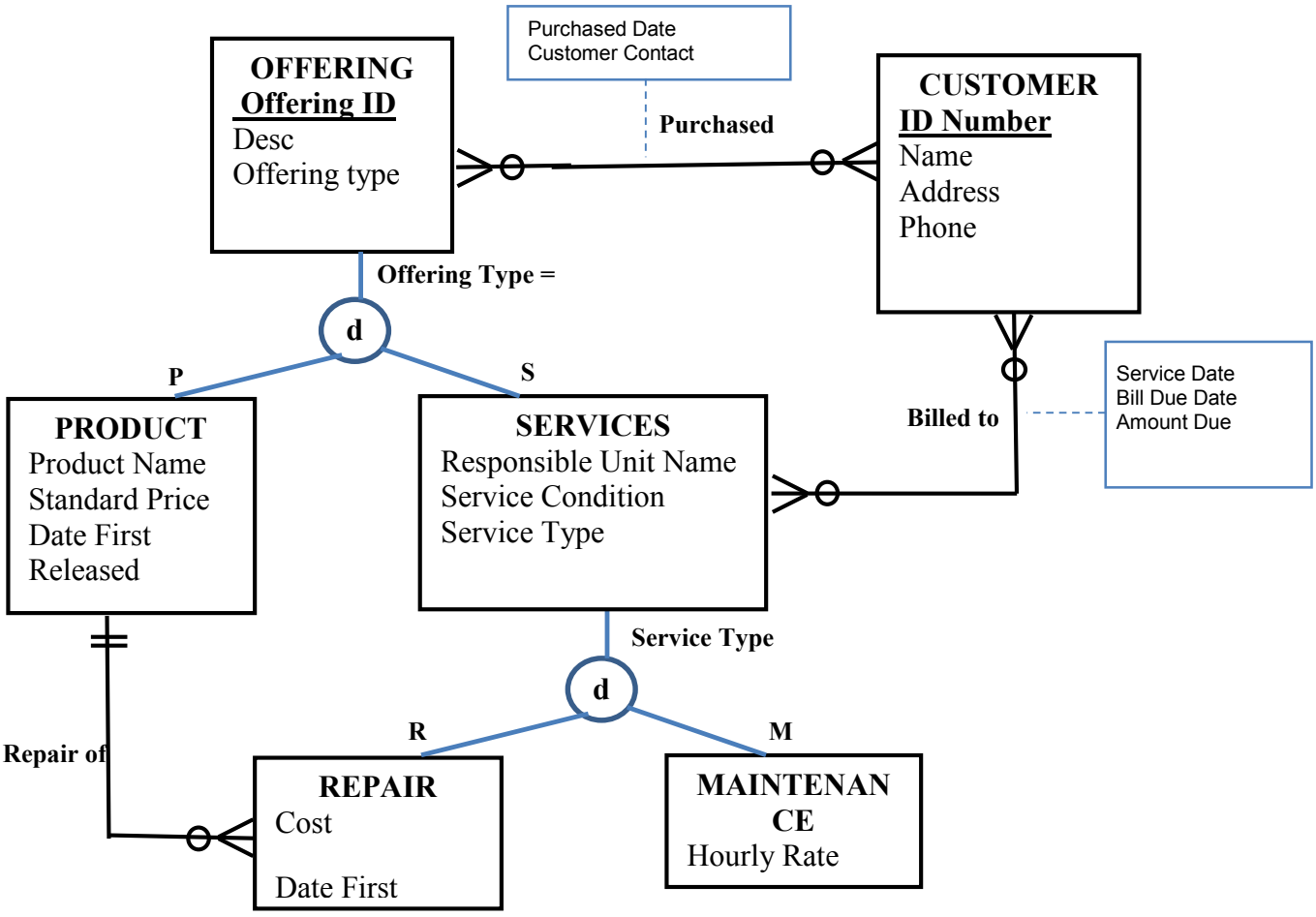
- Reviewing business rule again for the relationship for the relationship ***billed to*** between CUSTOMER & SERVICE :

> Because some customers purchase offerings for their clients, a customer may be billed for services he or she did not purchase, as well as for ones that were purchased.

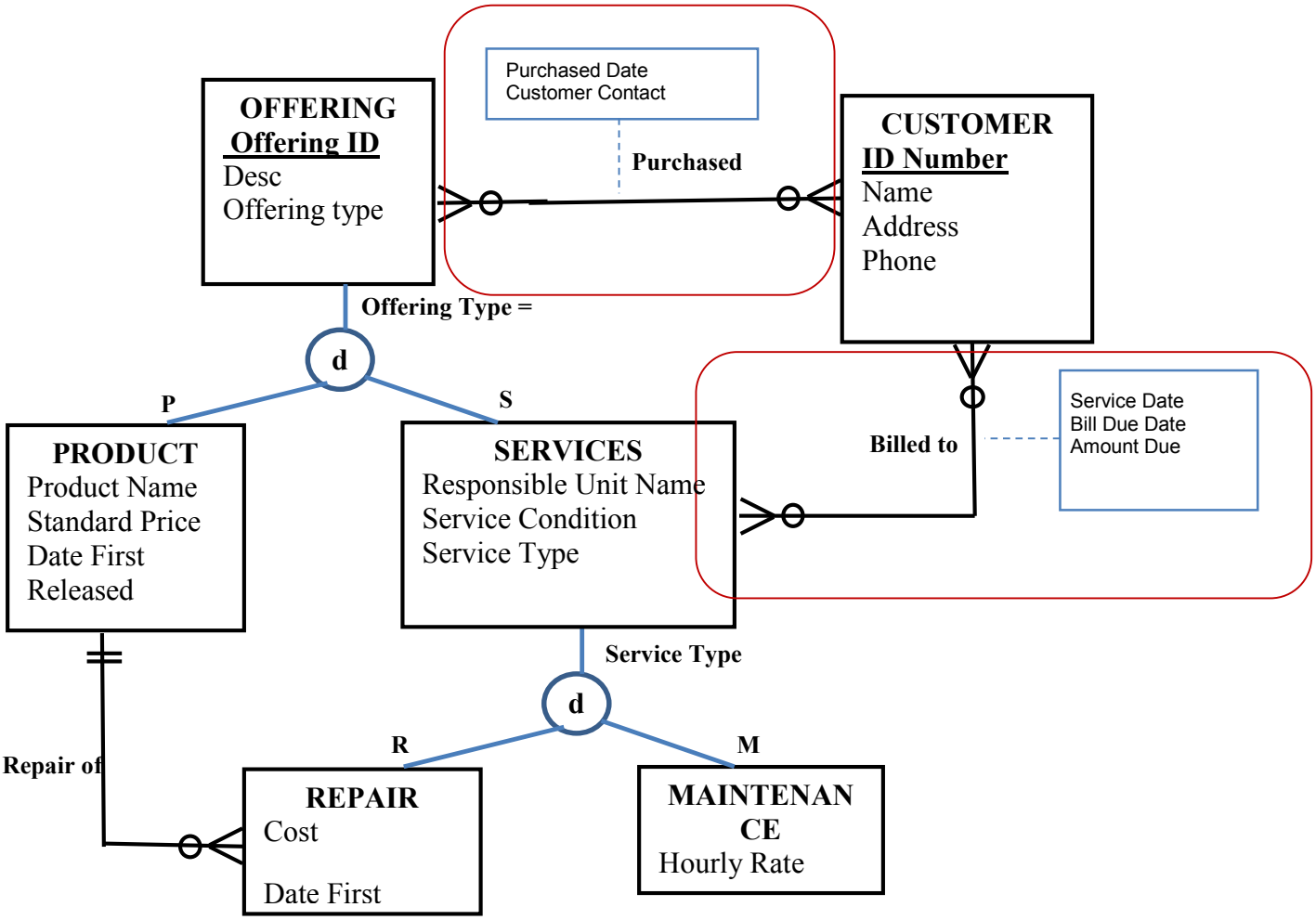
  - Well, this looks like a cardinality stating customer will be mandatory billed. But is not clear how to capture that fact of WHO did the purchase, the customer or its client?
  - Do we need to add information on the Customer's Clients? But none is provided?

- Conclusion:
  - Further investigation and help from experts is needed.
  - Let's leave this business rule out as professor investigates the solution.
  - Let's finalize this example and compare to final answer from the book

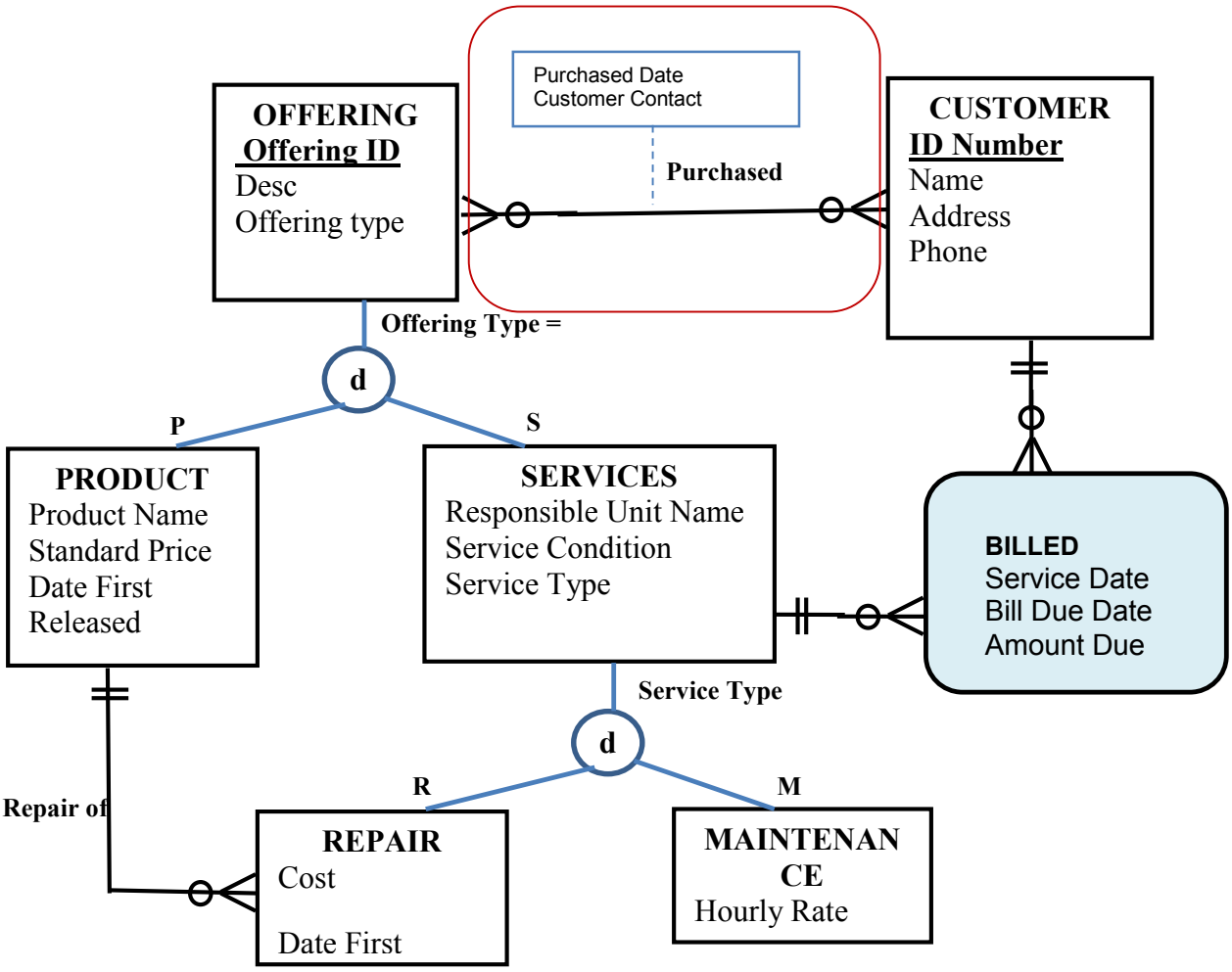
- The EER diagram with the requirements we were able to implement:



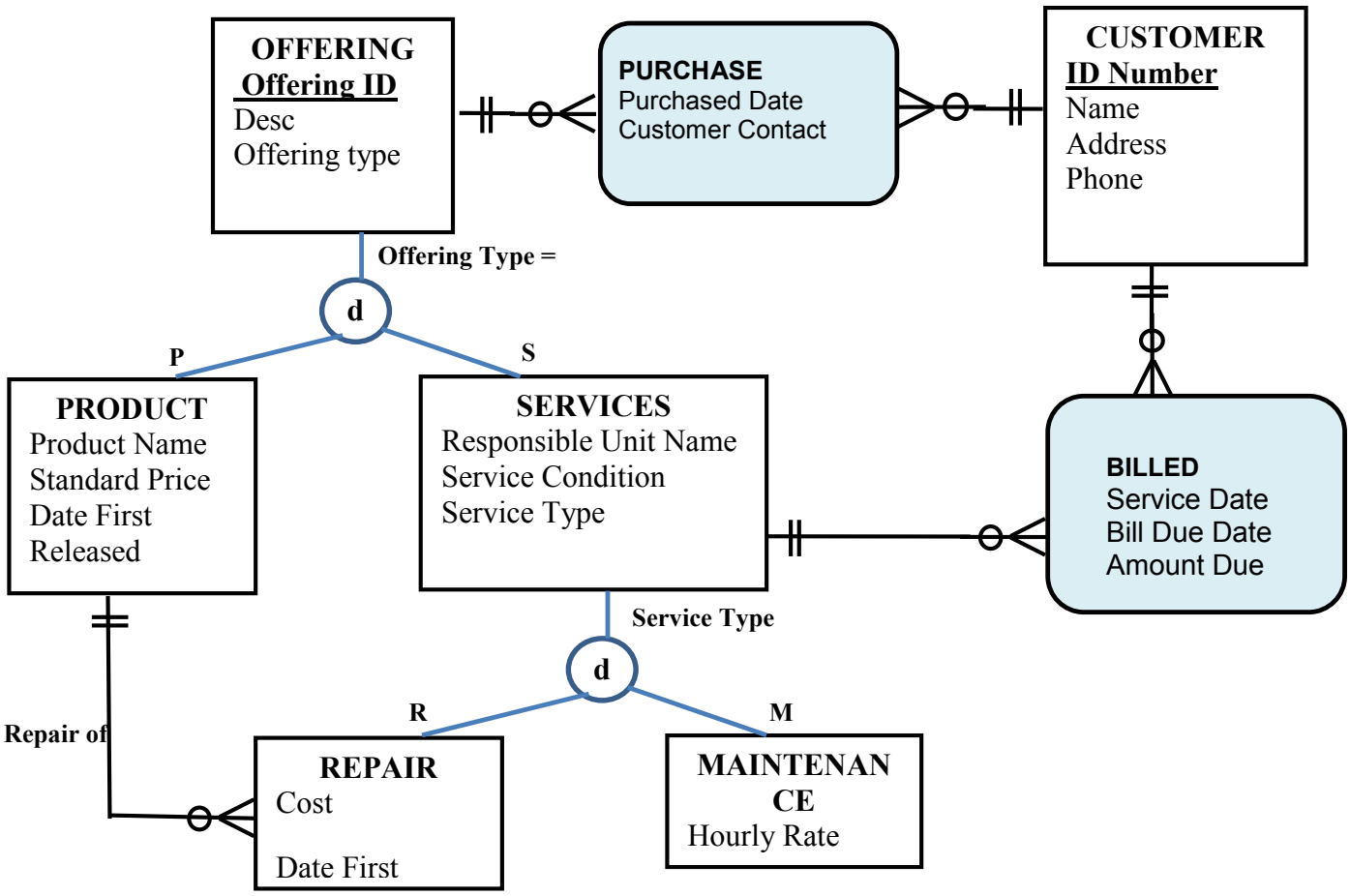
- We have the opportunity to convert some of the relationship attributes that are MANY to MANY into ASSOCIATE ENTITIES
- Highlighted in red are the relationship that qualify for ASSOCIATE ENTITIES:



- First we convert the **billed to** relationship and relationship attribute to an ASSOCIATE entity named BILLED:



- Finally we converted the **Purchased** relationship and relationship attribute to an ASSOCIATE entity named PURCHASE.
- We have our final diagram:



Technology Company ERD