UTS:ENGINEERING AND INFORMATION TECHNOLOGY

UTS
UNIVERSITY OF TECHNOLOGY, SYDNEY

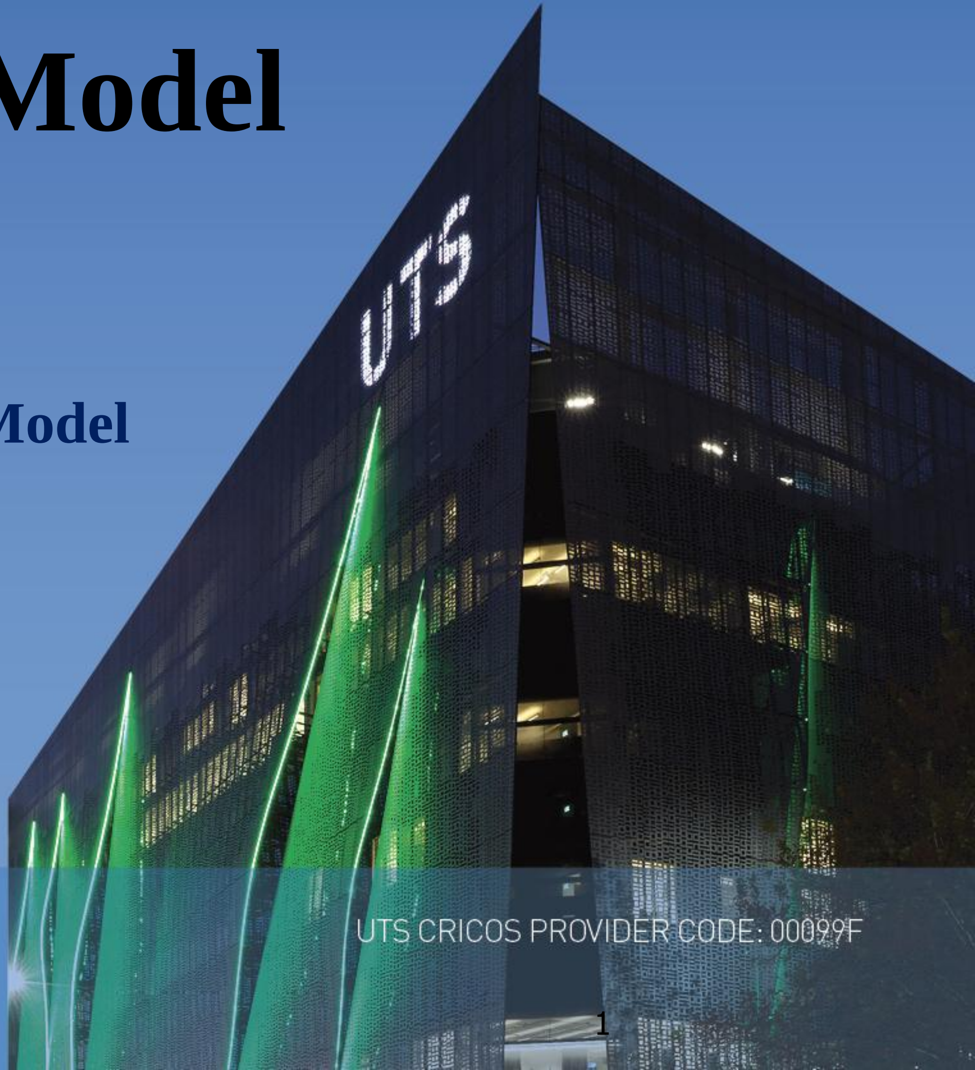
lecture 5: Relational Model

Main reference:

Modern Database Management, 11th Edition
Chapter 4: Logical Database Design and the Relational Model

Subject Coordinator and Instructor:

Dr. Danna (Fahimeh) Ramezani



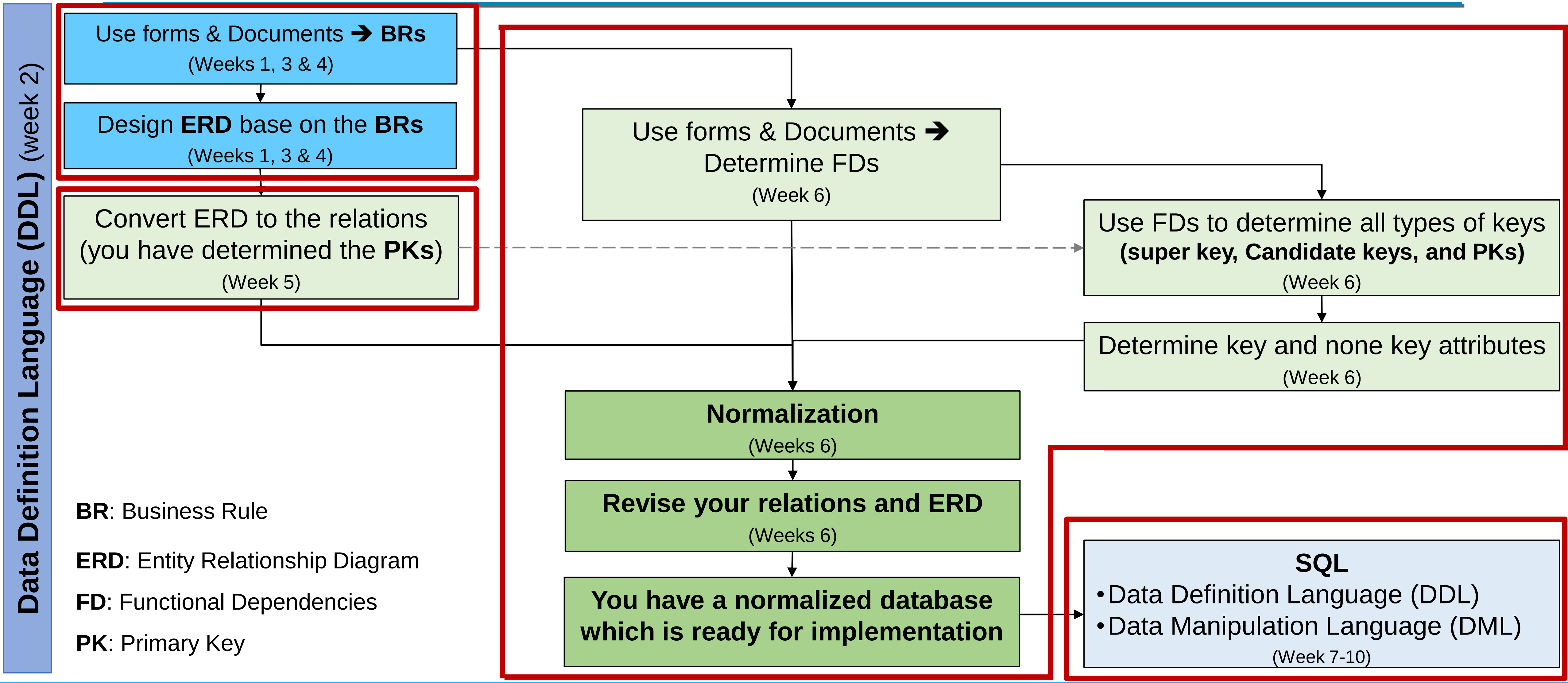
Participations and Discussions

The DF lecture are designed and elaborated to create a collaborative learning environment and engage students in concepts via class activities and discussions.

If you have any question and you don't want to share it in class, send it to us via Discussion Board on ED.

However, it is better to speak out in class ☺

Subject Flowchart



BR: Business Rule

ERD: Entity Relationship Diagram

FD: Functional Dependencies

PK: Primary Key

Subject Overview

- **Design Entity Relationship Diagram (ERD)**
  - **Week 1: Data Modelling I (Conceptual Level):** Entity, Attributes, PK, FK, …
  - **Week 2: Data Definition Language (DDL):** Create tables, constraints, insert, …
  - **Week 3: Data Modelling II (Conceptual Level):** Associative, Weak, …
  - **Week 4: Data Modelling III (Conceptual Level):** Subtype/Supertype
  - **Week 5: Convert ERD to Relations (Logical Level)**
  - **Week 6: Functional Dependencies, and Normalization**
- **Data manipulation**
  - **Week 7: Simple Query**
  - **Week 8: Multiple Table Queries**
  - **Week 9: Subquery**
  - **Week 10: Correlated Subquery**

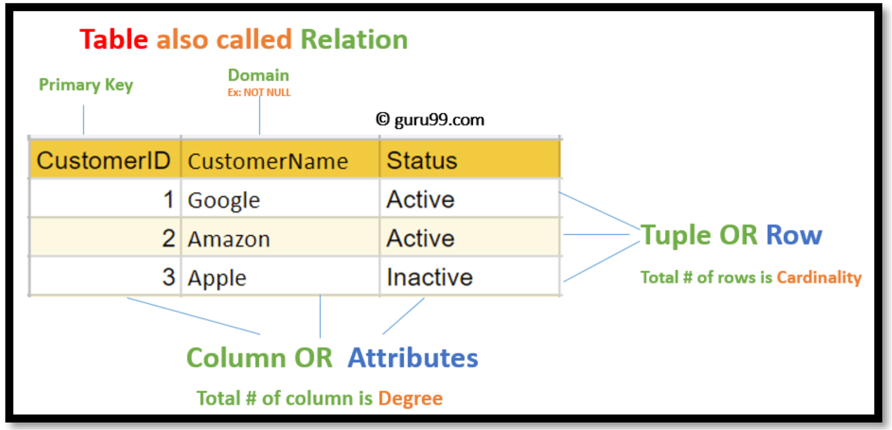
Objectives: Convert ERD to the relations

A **relational model** (a model of a possible database implementation) is **a collection of inter-related tables** that is build up based on an ERD (an abstract concept of our database).

1. **Components of relational model:** Data structure, Data manipulation, Data integrity
2. **Relations**
   2.1. Correspondence with E-R Model (Data structure)
   2.2. Key Fields
   2.3. Integrity Constraints (Data integrity)
      2.3.1. Domain Constraints
      2.3.2. Entity Integrity
      2.3.3. Referential Integrity
3. **Transforming EER Diagrams into Relations**
   3.1. Mapping Regular Entities to Relations
   3.2. Mapping Binary Relationships (1:M, 1:1, M:N (Associative Entities)
   3.3. Mapping Ternary (and n-ary) Relationships
   3.4. Mapping Weak Entities
   3.5. Mapping Unary Relationships
   3.6. Mapping Supertype/Subtype Relationships

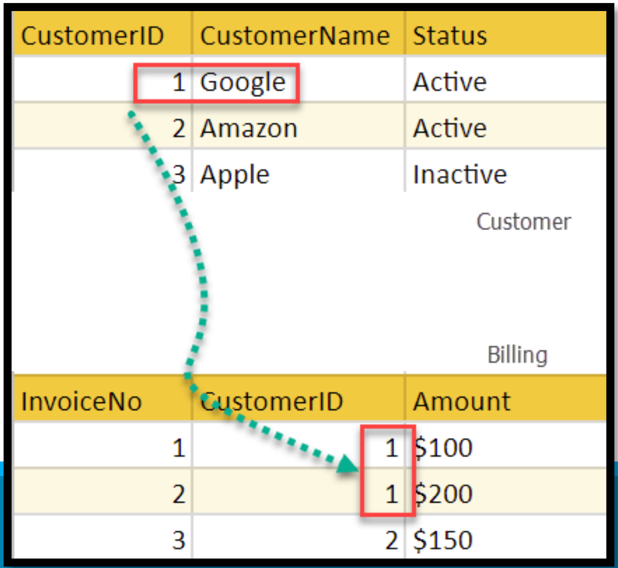
A **relational model** (a model of a possible database implementation) is **a collection of inter-related tables** that is build up based on an ERD (an abstract concept of our database).

1. Components of Relational Model



- **Data structure** (in this video)
  Tables (**relations**), rows, columns

- **Data manipulation** (Lecture 7 -10)
  Powerful SQL operations for retrieving and modifying data



- **Data integrity** (Video 4.2)

  Mechanisms for implementing business rules that maintain integrity of manipulated data → referred to PK/FK

Video 4.1: Relation

Table also called Relation

Primary Key

Domain
Ex: NOT NULL

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Tuple OR Row

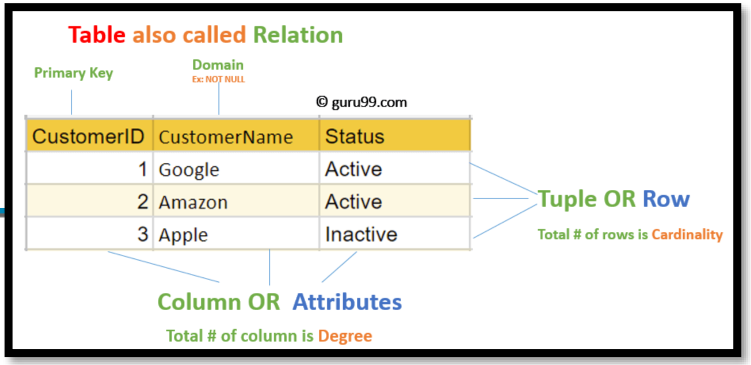
Total # of rows is Cardinality

Column OR Attributes

Total # of column is Degree

Images Reference: https://www.guru99.com/relational-data-model-dbms.html

2. Relation



Images Reference: https://www.guru99.com/relational-data-model-dbms.html

- A relation is a named, two-dimensional **table** of data.
- A table consists of rows (records) and columns (attribute or field).
- Requirements for a table to qualify as a relation:
  - It must have a **unique name**.
  - Every attribute value must be **atomic** (not multivalued (like Skill), not composite (like Address))

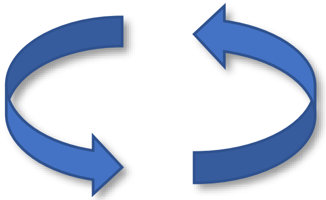
    (*More on this in the next lectures*).
  - Every **row** must be **unique** (can’t have two rows with exactly the same values for all their fields).
  - **Attributes** (columns) in tables must have **unique names**.
  - The order of the columns must be irrelevant.
  - The order of the rows must be irrelevant.

2.1. Relational Model Concepts Correspond to E-R Model

- **Relations (tables)** correspond with **entity types** and with many-to-many relationship types.
- **Rows** correspond with **entity instances** and with many-to-many relationship instances.
- **Columns** correspond with **attributes**.

Entity Types

INSTRUCTOR
Ins_ID
Ins_F_Name
Ins_L_Name



Relations (tables)

Ins_ID	Ins_F_Name	Ins_L_Name
12548	Danna	Ramezani
45476	Ricky	Brown
14475	Jack	Cooper

NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).

2.2. Key Fields (Review)

PK

Customer_ID	Customer_Name	Customer_Street	Customer_City	Customer_State	CustomerPostal_Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Eastern Furniture	Palace Ave	Farmington	NM	NULL

PK / FK

Order_ID	Order_Date	Customer_ID
1001	8/09/2009	4
1002	4/10/2009	3
1003	19/07/2009	1
1004	1/11/2009	6
1005	28/07/2009	4
1006	27/08/2009	4

- Keys are special fields that serve two main purposes:

  - ***Primary keys*** are **unique** identifiers of the relation.

    Examples include Customer ID, etc. This guarantees that all rows are unique.

  - ***Foreign keys*** are identifiers that enable a **dependent** relation (on the many side of a relationship) to refer to its **parent** relation (on the one side of the relationship).

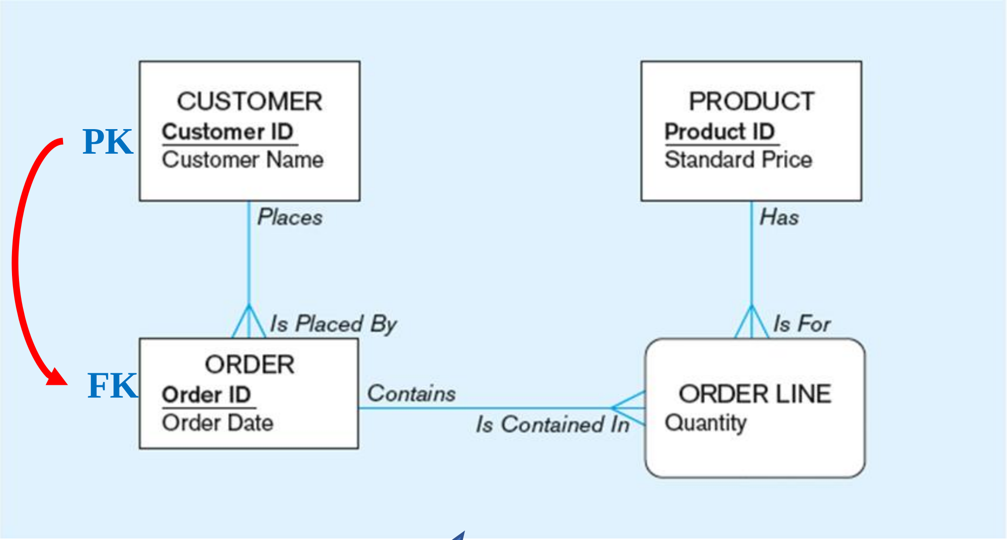
- Keys can be ***simple*** (a single field) or ***composite*** (more than one field).
- Keys usually are used as indexes to speed up the response to user queries *(more on this in Chapter 5).*

Primary key/Foreign key (Review)

BR1: any **customer** can place many orders.

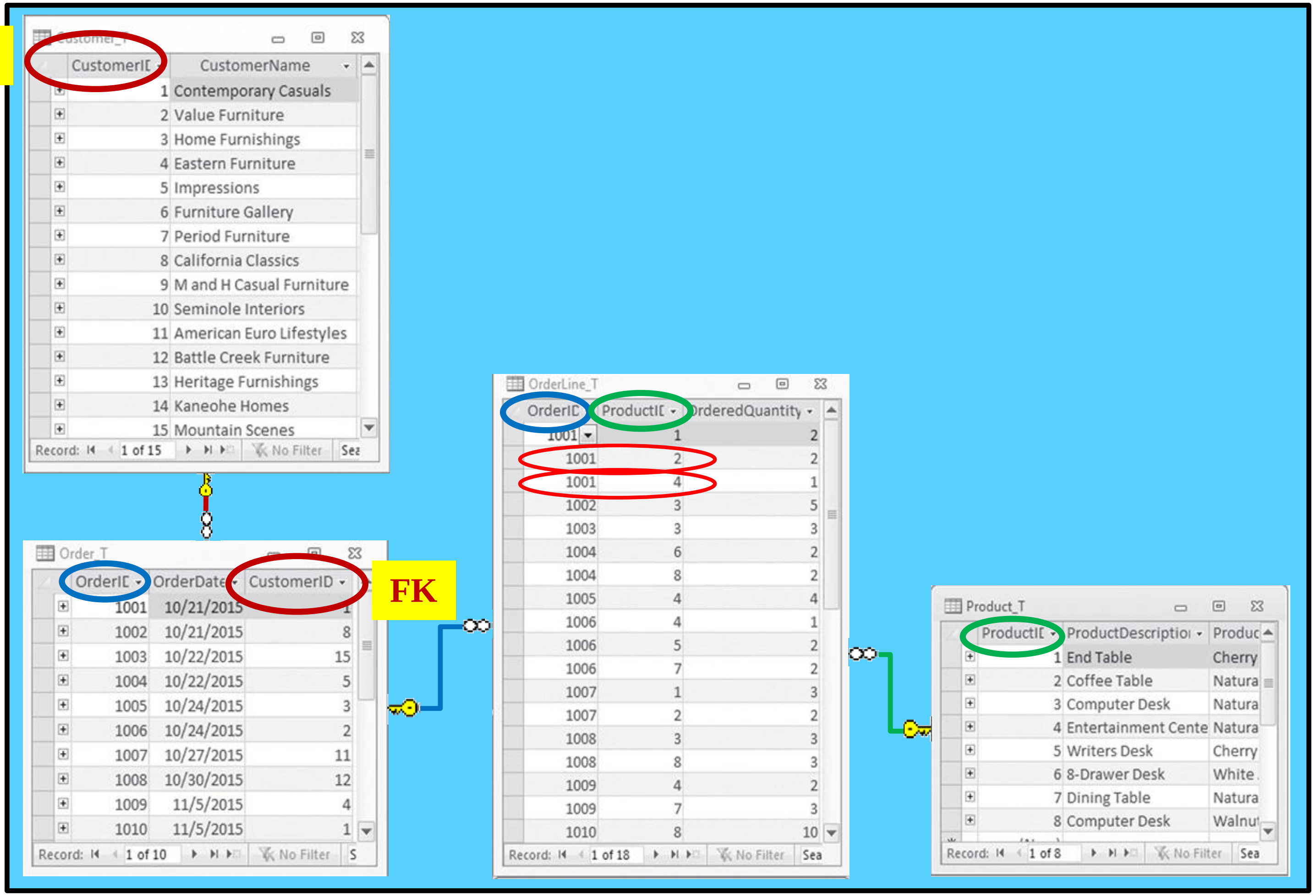
BR2: any **order** can be related to many **products**.

BR3: any **product** can be **ordered** many times.



E-R Model

Relational Model



CustomerID	CustomerName
1	Contemporary Casuals
2	Value Furniture
3	Home Furnishings
4	Eastern Furniture
5	Impressions
6	Furniture Gallery
7	Period Furniture
8	California Classics
9	M and H Casual Furniture
10	Seminole Interiors
11	American Euro Lifestyles
12	Battle Creek Furniture
13	Heritage Furnishings
14	Kaneohe Homes
15	Mountain Scenes

Record: 1 of 15

OrderID	OrderDate	CustomerID
1001	10/21/2015	1
1002	10/21/2015	8
1003	10/22/2015	15
1004	10/22/2015	5
1005	10/24/2015	3
1006	10/24/2015	2
1007	10/27/2015	11
1008	10/30/2015	12
1009	11/5/2015	4
1010	11/5/2015	1

Record: 1 of 10

OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Record: 1 of 18

ProductID	ProductDescription	Produc
1	End Table	Cherry
2	Coffee Table	Natura
3	Computer Desk	Natura
4	Entertainment Cente	Natura
5	Writers Desk	Cherry
6	8-Drawer Desk	White
7	Dining Table	Natura
8	Computer Desk	Walnu

Record: 1 of 8

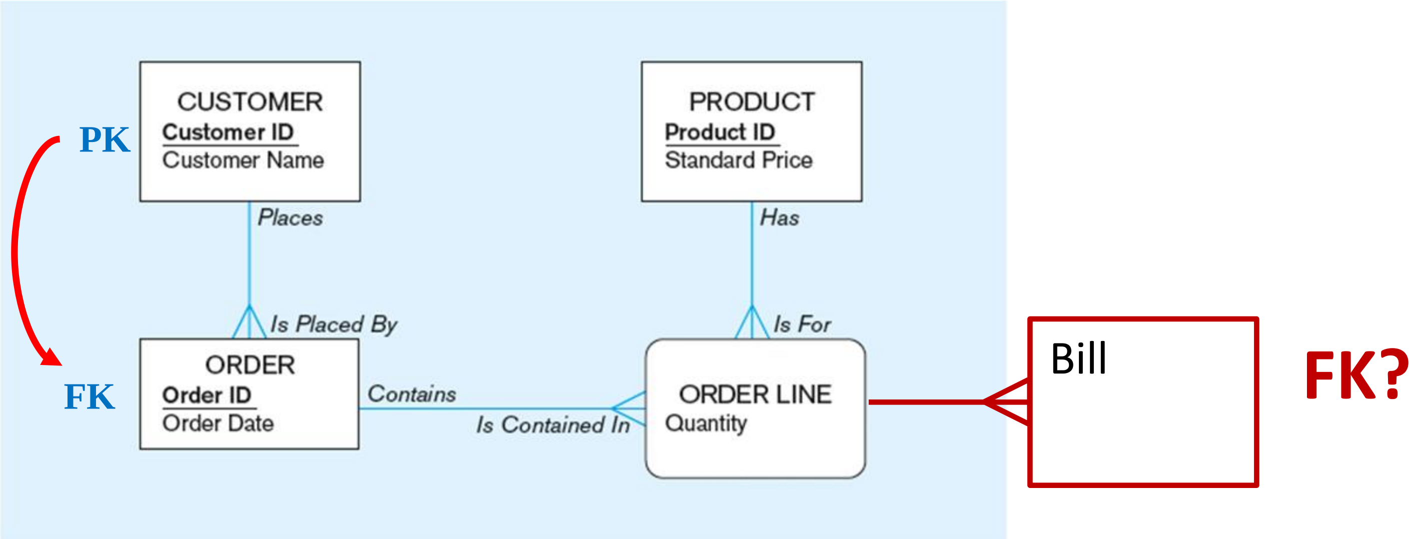
Class Activity 4.1 → in the live lecture

Explain why when there is one-to-many relationship between two entities, **PK** of the entity on the one side will be **FK** on the entity on the many side. Provide your explanation via an example.

Class Activity 4.2 (2 Minutes) → in the live lecture

Remind the rule: in each one-to-many relationships, **PK** of the entity **on one side** is **FK** of the entity **on the many side.**

Question: What would be the FK in a relation (table) that need to have a relationship with OrderLine_T?



Solution to class Activity 4.1 → in the live lecture

Question: What would be the FK in a relation (table) that need to have a relationship with OrderLine_T?

PK

Customer_ID	Customer_Name	Customer_Street	Customer_City	Customer_State	CustomerPostal_Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Eastern Furniture	Palace Ave	Farmington	NM	NULL

Order_ID	Order_Date	Customer_ID
1001	8/09/2009	4
1002	4/10/2009	3
1003	19/07/2009	1
1004	1/11/2009	6
1005	28/07/2009	4
1006	27/08/2009	4

FK

Video 4.2. Integrity Constraints

2.3. Integrity Constraints

Integrity Constraints are applied to facilitate maintaining the **accuracy** and **integrity** of data in the database.

The major types of integrity constraint are:

PK

Customer_ID	Customer_Name	Customer_Street	Customer_City	Customer_State	CustomerPostal_Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Eastern Furniture	Palace Ave	Farmington	NM	NULL

Order_ID	Order_Date	Customer_ID
1001	8/09/2009	4
1002	4/10/2009	3
1003	19/07/2009	1
1004	1/11/2009	6
1005	28/07/2009	4
1006	27/08/2009	4

FK

2.3.1. Domain Constraints (Determines the allowable values for an attribute)

2.3.2. Entity Integrity (PK fields can’t be null)

2.3.3. Referential Integrity (foreign key value MUST match a primary key value)

2.3.1. Domain Constraints

All the values that appear in a **column** of a relation (Table) must be from **the same domain**.

- A **domain** is the **set of values** that may be assigned to an **attribute**.

  Example: the domain value for attribute **Student_ID** is an **integer with 4 digits** then the **domain** can be any integer number between 1000, 9999.

- A **domain definition** is usually consisting of the following components:
  - Domain name
  - Description
  - Data type
  - Size (or length)
  - Allowable values
  - Allowable range

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Domain definitions enforce domain integrity constraints.

2.3.2. Entity Integrity

Entity Integrity:

- No primary key attribute may be null. All primary key fields **MUST have data**.
- The primary key should have a **unique value** for each row in the table.

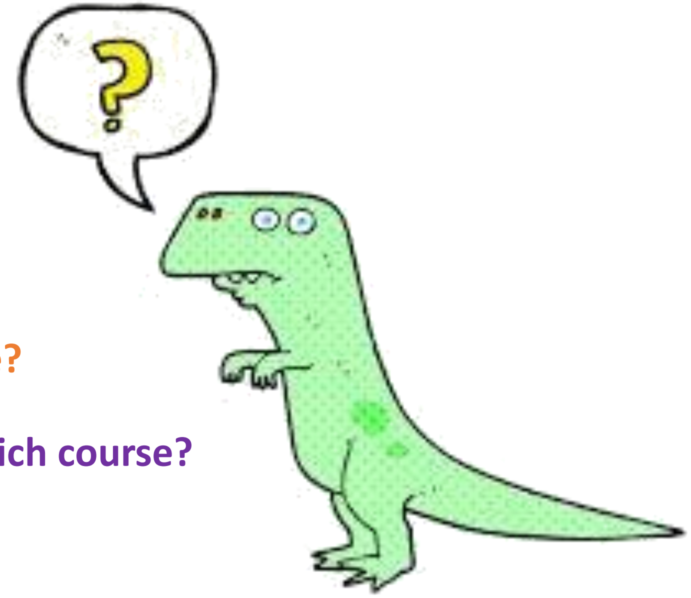
Composite PK

Employee ID	Course ID	Date Completed
null	null	1/1/2017
null	A57	1/2/2016
345	null	5/10/2016
…	…	…
…	…	…

1/1/2017 is a date for what?

1/2/2016 is a date for which employee?

5/10/2016 is a date for completing which course?



2.3.3. Referential Integrity

- **Referential Integrity** states that **any foreign key value** (on the relation of the many side) MUST match **a primary key value** in the relation of the one side.

PK

Customer_ID	Customer_Name	Customer_Street	Customer_City	Customer_State	CustomerPostal_Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Eastern Furniture	Palace Ave	Farmington	NM	NULL

Order_ID	Order_Date	Customer_ID
1001	8/09/2009	4
1002	4/10/2009	3
1003	19/07/2009	1
1004	1/11/2009	6
1005	28/07/2009	4
1006	27/08/2009	4

FK

- Referential Integrity rule is used to maintain the **consistency among rows** between the two tables.

Referential integrity constraints are implemented with foreign key to primary key references.

```
CREATE TABLE Customer_T
        (CustomerID             NUMBER(11,0)    NOT NULL,
        CustomerName            VARCHAR2(25)    NOT NULL,
        CustomerStreet          VARCHAR2(30),
        CustomerCity            VARCHAR2(20),
        CustomerState           CHAR(2),
        CustomerPostalCode      VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                NUMBER(11,0)    NOT NULL,
        OrderDate               DATE DEFAULT SYSDATE,
        CustomerID              NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));
```

Integrity Constraints (Referential Integrity)

SQL table definitions (Figure 4-6)

How referential integrity will be implemented in a relational database?

Referential Integrity rules for delete operation

- **Restrict:** don’t allow delete of “parent” side if related rows exist in “dependent” side
- **Cascade:** automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted
- **Set-to-Null:** set the foreign key in the dependent side to null if deleting from the parent side
  - The foreign key can be null
  - **Set-to-Null** is not allowed for weak and associated entities
  - **Set-to-Null** is not allowed when is related to a mandatory cardinality.

2.3.3. Referential Integrity: Restrict

Restrict: don't allow delete of "parent" side if related rows exist in "dependent" side

- Delete of "parent" side

ON DELETE NO ACTION

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

DELETE

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

Delete Unsuccessful.

Cannot delete, Because it will violate referential integrity as in EMPLOYEE relation, the tuple with DNO=10 causes problem.

- Delete of "dependent" side

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

Referencing Relation

Deletion Successful, because deletion of this tuple causes no violation.

DEPARTMENT

DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Referenced Relation

Issue: You can not delete any data form the parent table.

Photo Reference: http://www.edugrabs.com

2.3.3. Referential Integrity: Cascade

Cascade: automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted.

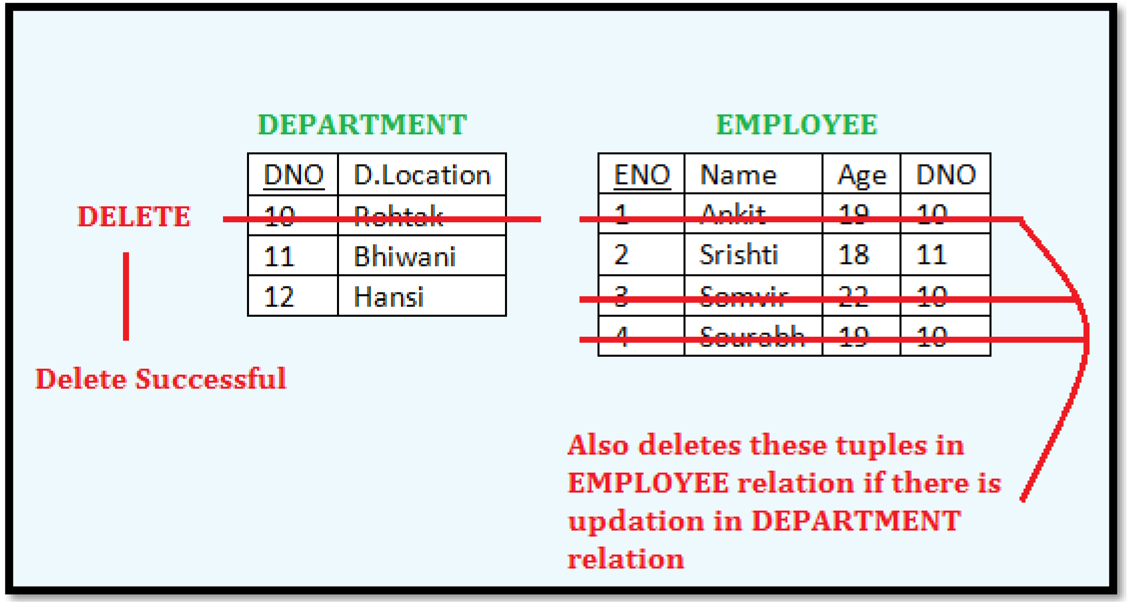


Photo Reference: http://www.edugrabs.com

Issue: Risk of loosing some important data in the dependent table(s)

2.3.3. Referential Integrity: Set-to-Null

Set-to-Null: set the foreign key in the dependent side to null if deleting from the parent side.

Issue: The foreign key can be null, however In the following situation, set-to-null is not allowed:

- → Set-to-Null is not allowed for **weak** and **associated** entities (where FK are part of the key).
- → Set-to-Null is not allowed when is related to a **mandatory** cardinality.

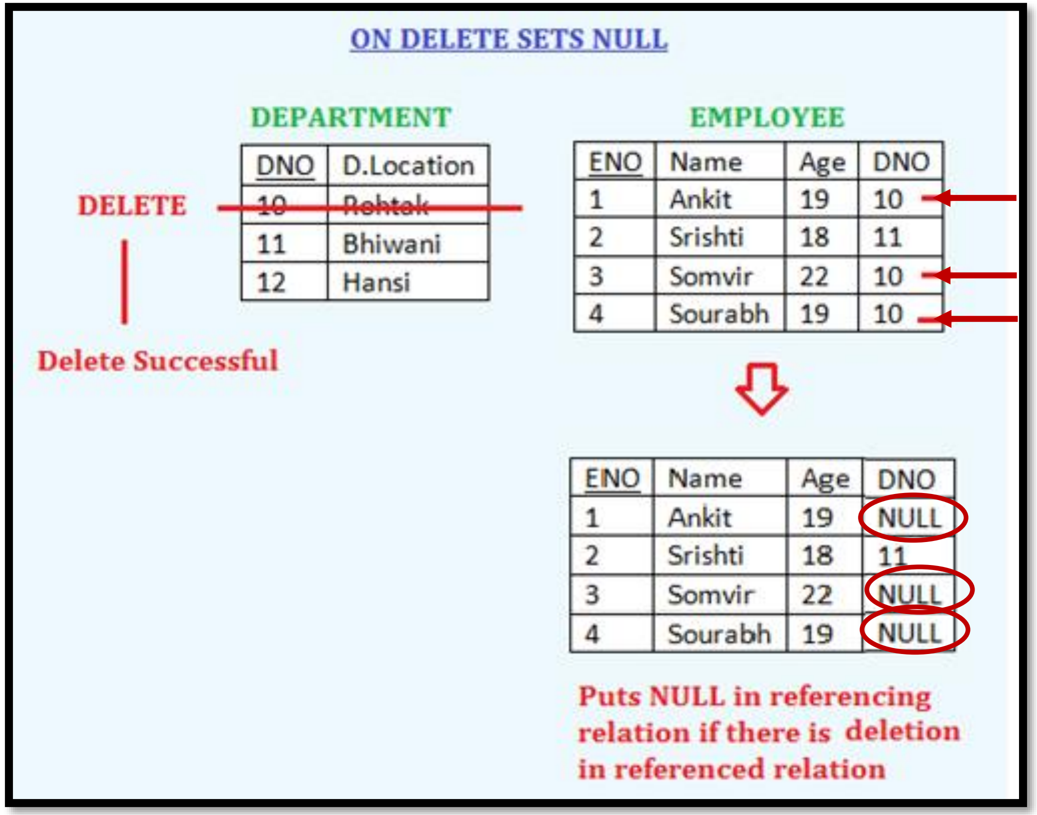


Photo Reference: http://www.edugrabs.com

2.3.3. Referential Integrity: Set-to-Null (cont.)

Set a value of FK to Null is not allowed for **weak** and **associated** entities (where **FK are part of the key**).

Entity Integrity

- No primary key attribute may be null.
- All primary key fields **MUST have unique data**.

PK

Employee

Employee ID	Employee Name	Birth Date
1234587	Jack	1/1/1960
3459087	Danna	5/2/1985

FK FK

Certificate

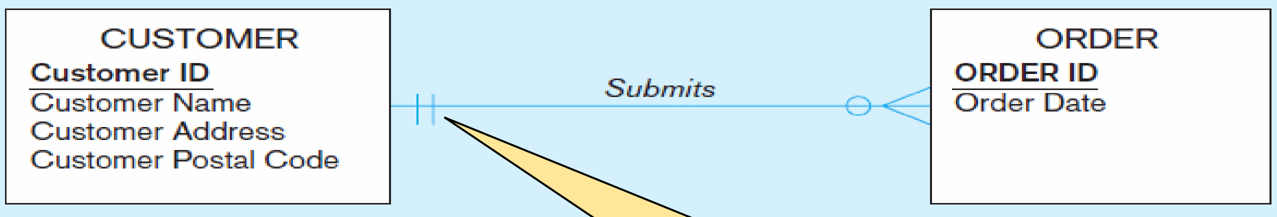
Employee ID	Course ID	Date Completed
1234587	C68	1/1/2017
3459087	A57	1/2/2016
3459087	C68	5/10/2016

PK

Course

Course ID	Course Title	Topic
A57	Knowledge Management	....
C68	Project Management	....

Set a value in FK to Null is not allowed when is related to a mandatory cardinality.



We can't have a null value in the foreign key because of the **mandatory minimum cardinality**.

PK

Customer_ID	Customer_Name	Customer_Street	Customer_City	Customer_State	CustomerPostal_Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furnitures	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Eastern Furniture	Palace Ave	Farmington	NM	NULL

FK

Order_ID	Order_Date	Customer_ID
1001	8/09/2009	4
1002	4/10/2009	3
1003	19/07/2009	1
1004	1/11/2009	6
1005	28/07/2009	4
1006	27/08/2009	4

Where we are … review the path ☺

- We know about the relations …
- We know the integrity constraints …
- Now we are ready to start converting different type of attributes and entities to the relations considering the integrity constraints

Now we know about the **relations** and the **integrity constraints** …

we are ready to start converting different type of entities to the relations …

3. Transforming ERD into Relations

3.1. Mapping Regular Entities to Relations

3.2. Mapping Binary Relationships

- Binary One-to-Many
- Binary One-to-one
- Binary Many-to-Many → Associative Entities

3.3. Mapping Ternary (and n-ary) Relationships

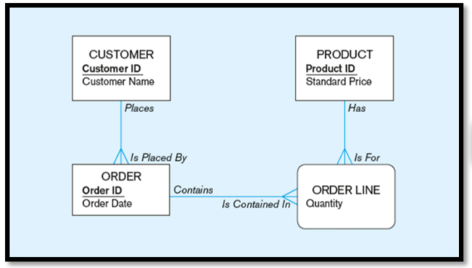
3.4. Mapping Weak Entities

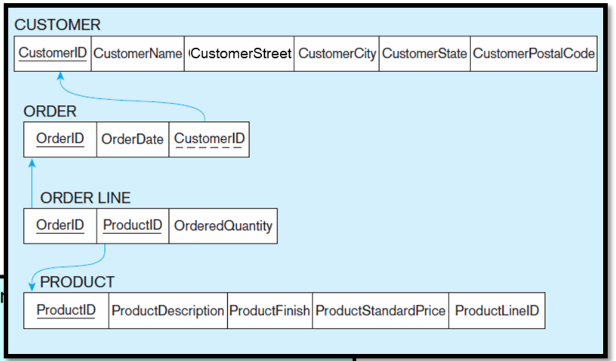
3.5. Mapping Unary Relationships

3.6. Mapping Supertype/Subtype Relationships

Transforming ERD into Relations

Video 4.3: Mapping Regular Entities to Relations





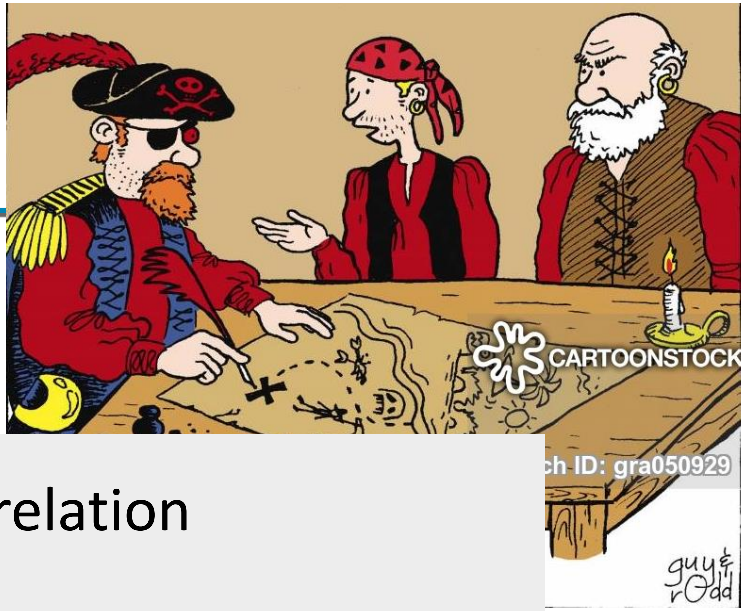
CUSTOMER (CustomerID, CustomerName, CustomerStreet, CustomerCity, Custor

ORDER (OrderID, OrderDate, CustomerID*)
FK (CustomerID) references CUSTOMER

ORDERLINE (OrderID*, ProductID*, OrderQuantity)
FK (OrderID) references ORDER
FK (ProductID) references PRODUCT

PRODUCT (ProductID, ProductDescription, ProductFinish, ProductStandardPrice, ProductLineID)

3.1. Mapping Regular Entities to Relations



3.1.1. Simple attributes: E-R attributes map directly onto the relation

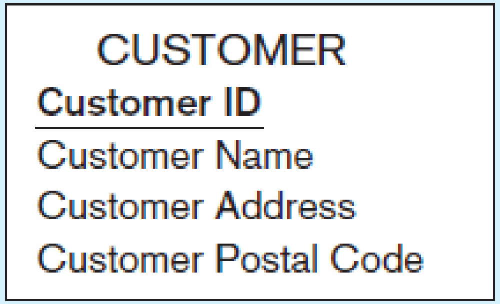
3.1.2. Composite attributes: Use only their simple component attributes

3.1.3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

3.1.1. Mapping a regular entity (Figure 4-8) with simple attribute

3.1.1. Simple attributes: E-R attributes map directly onto the relation

(a) CUSTOMER entity type with simple attributes



(b) CUSTOMER relation

CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

or

CUSTOMER (<u>CustomerID</u>, CustomerName, CustomerAddress, CustomerPostalCode)

3.1.2. Mapping a composite attribute (Figure 4-9)

3.1.2. Composite attributes: Use only their simple component attributes

(a) CUSTOMER entity type with composite attribute

CUSTOMER
<u>Customer ID</u>
Customer Name
Customer Address
(CustomerStreet, CustomerCity, CustomerState)
Customer Postal Code

(b) CUSTOMER relation with address detail

CUSTOMER

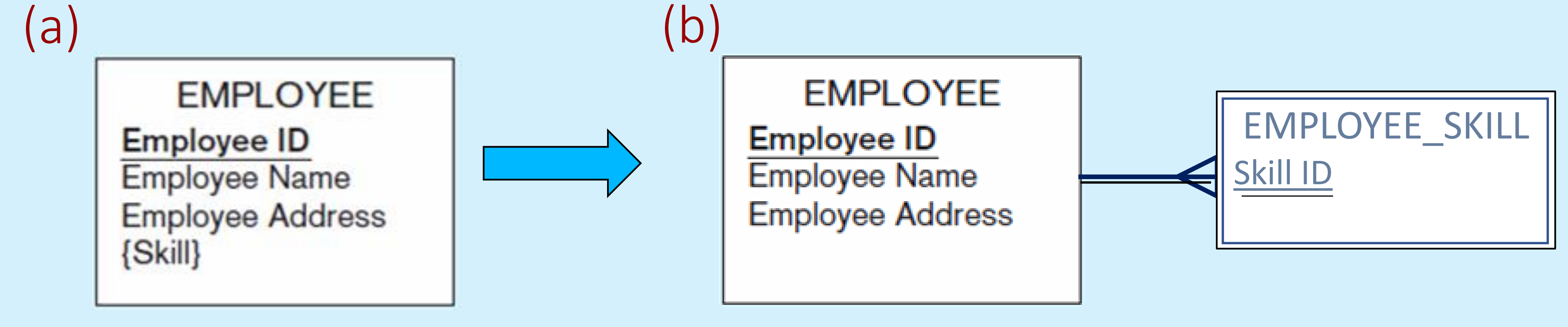
<u>CustomerID</u>	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

or

CUSTOMER (<u>CustomerID</u>, CustomerName, CustomerStreet, CustomerCity, CustomerState, CustomerPostalCode)

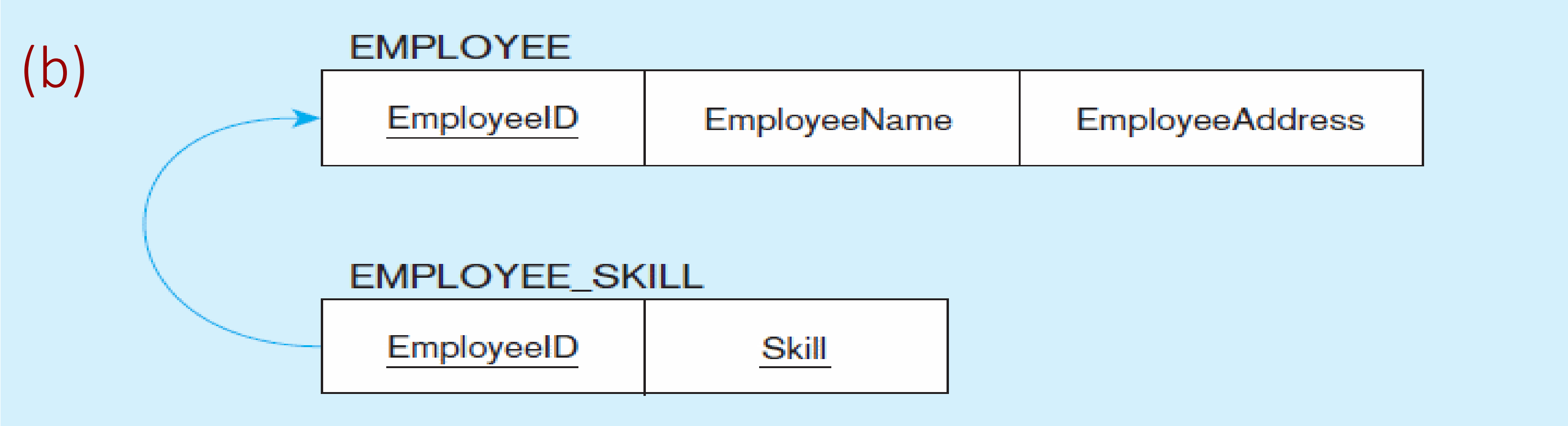
3.1.3. Mapping a multivalued attribute (Figure 4-10)

3.1.3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity.



?

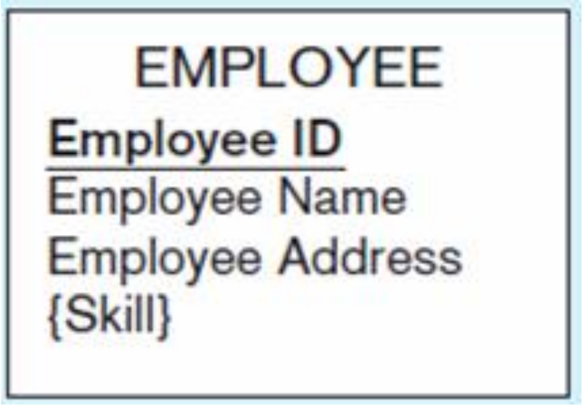
- Multivalued attribute becomes a separate relation with foreign key
- One–to–many relationship between original entity and new relation



Class Activity 4.3 (10 min) → in the live lecture

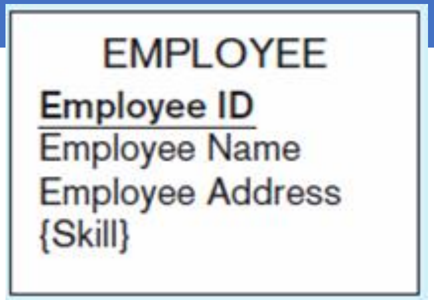
Redesign the following entity, where every employee need to have at least one skill and every skill can be chosen by any employee. The information about skills like skill ID and name need to be stored in the database.

Explain why this need to be redesigned, and then convert your new ERD to the relations.



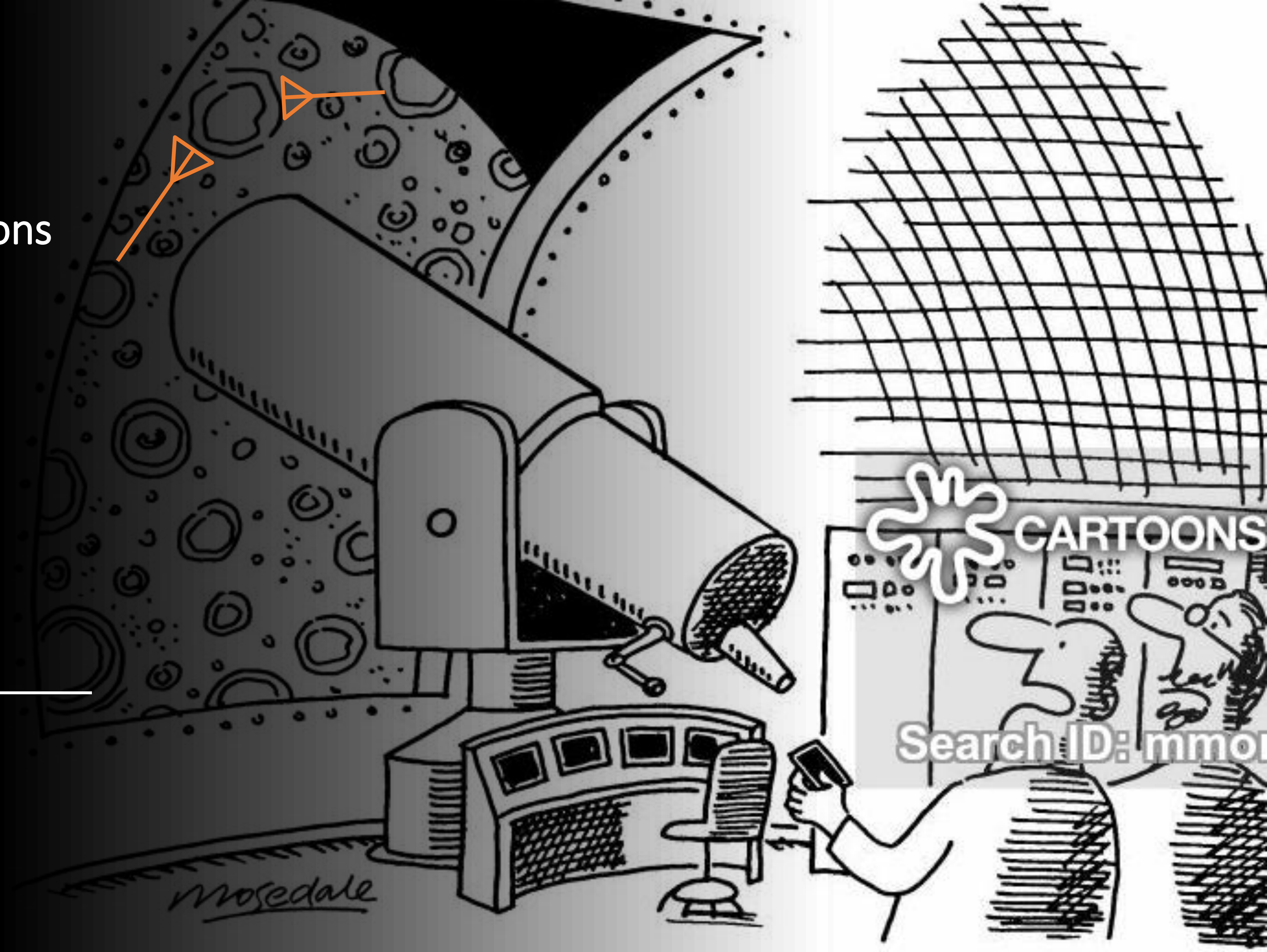
Solution to Class Activity 4.3→ in the live lecture

where every employee need to have at least one skill



Transforming ERD into Relations

Video 4.4: Mapping Binary Relationships



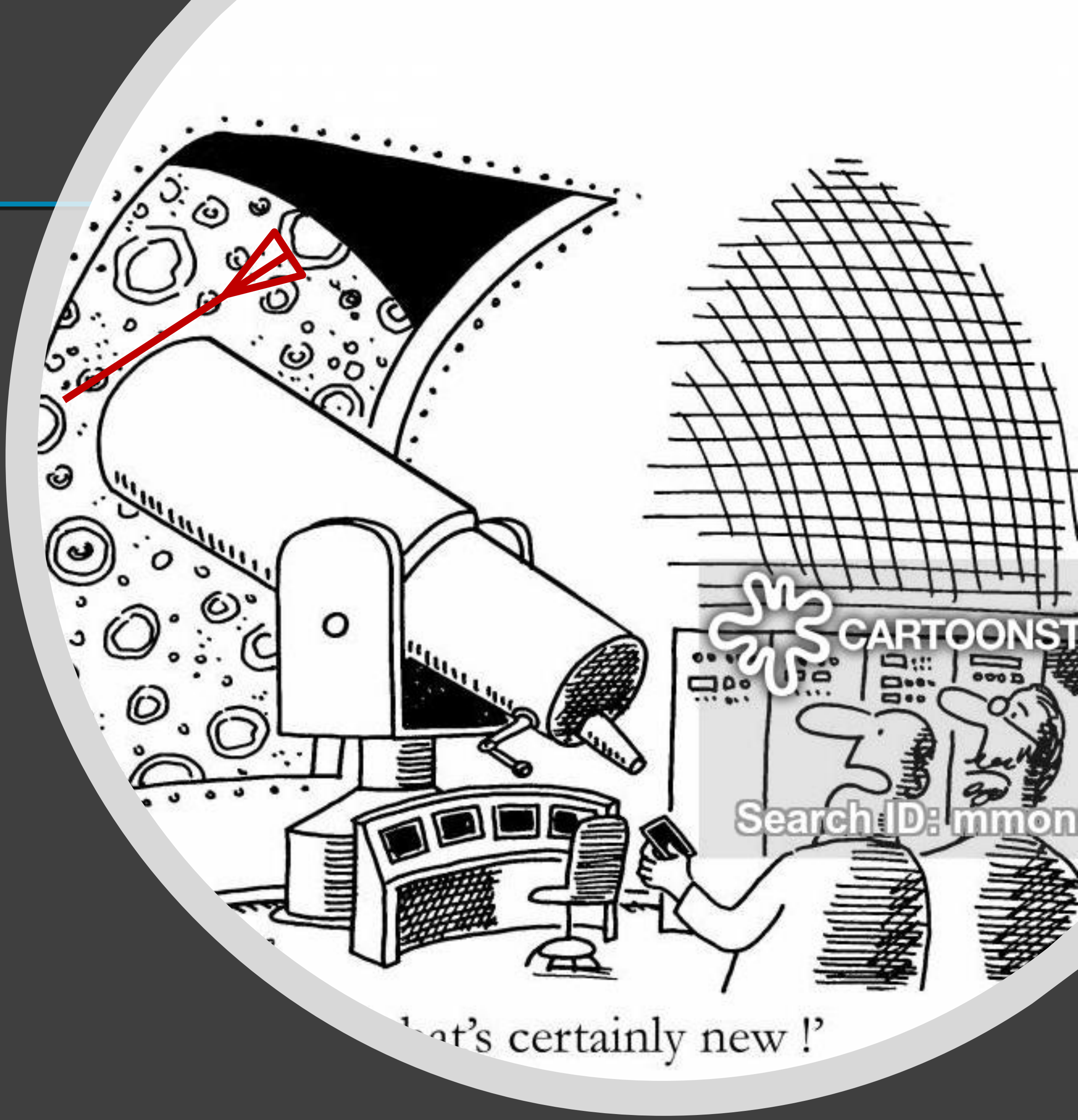
'Well that's certainly new !'

3.3. Mapping Binary Relationships

3.3.1. One-to-Many: Primary key of the entity on the one side becomes a foreign key of the entity on the many side.

3.3.2. One-to-One: Primary key of the entity on the mandatory side becomes a foreign key of the entity on the optional side.

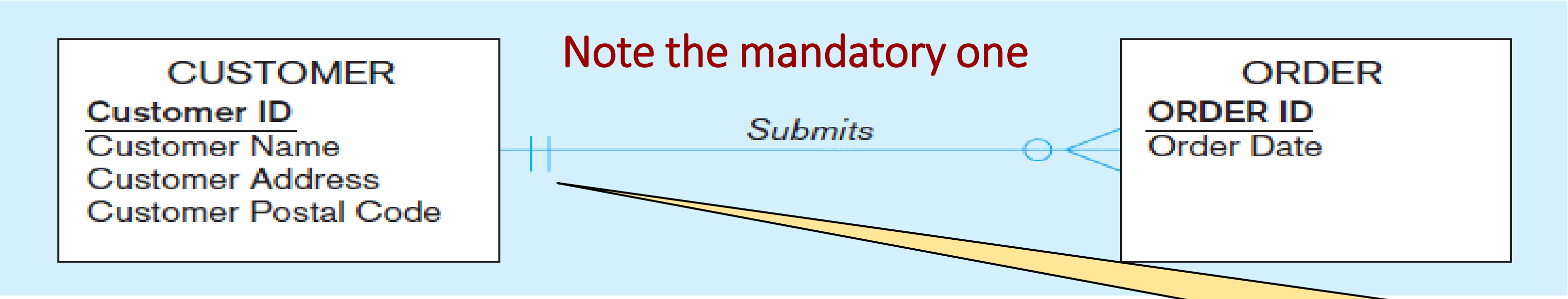
3.3.3. Many-to-Many → We need to create a ***new relation*** **(Associative Entity)** where the primary keys of the related entities will participate in making the composite primary key of the new relation (associative entity).



3.3.1. Mapping a 1:M relationship (Figure 4-12)

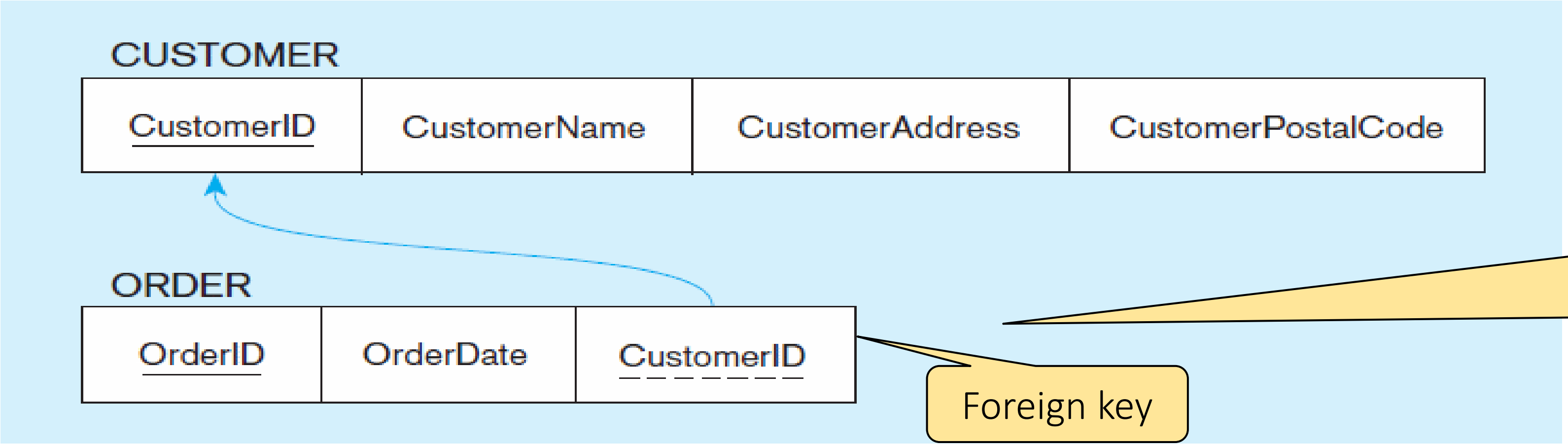
a) Relationship between customers and orders

Note the mandatory one



Answer: no null value in the foreign key… this is because of the mandatory minimum cardinality.

b) Mapping the relationship (The textbook format)

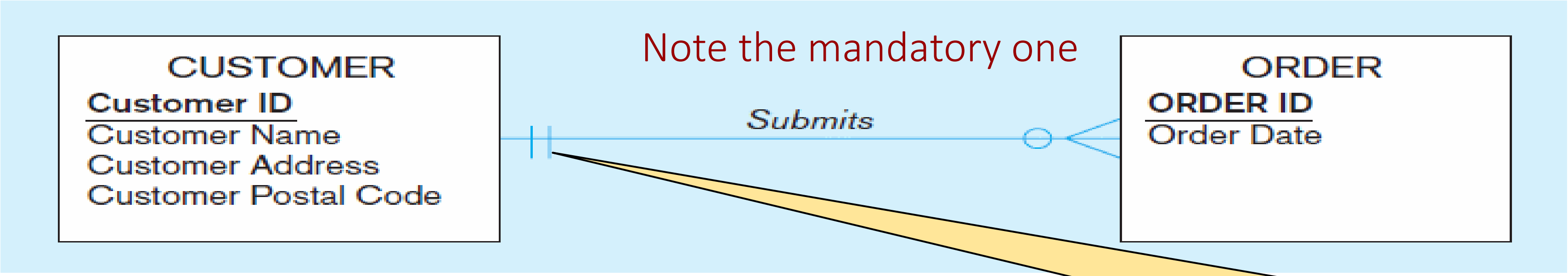


Foreign key

Question:
Can CustomerID in ORDER relation has null values?

3.3.1. Mapping a 1:M relationship- Other Format

a) Relationship between customers and orders



Again, no null value in the foreign key… this is because of the **mandatory minimum cardinality**.

b) Mapping the relationship

CUSTOMER (<u>CustomerID</u>, CustomerName, CustomerAddress, CustomerPostalCode)

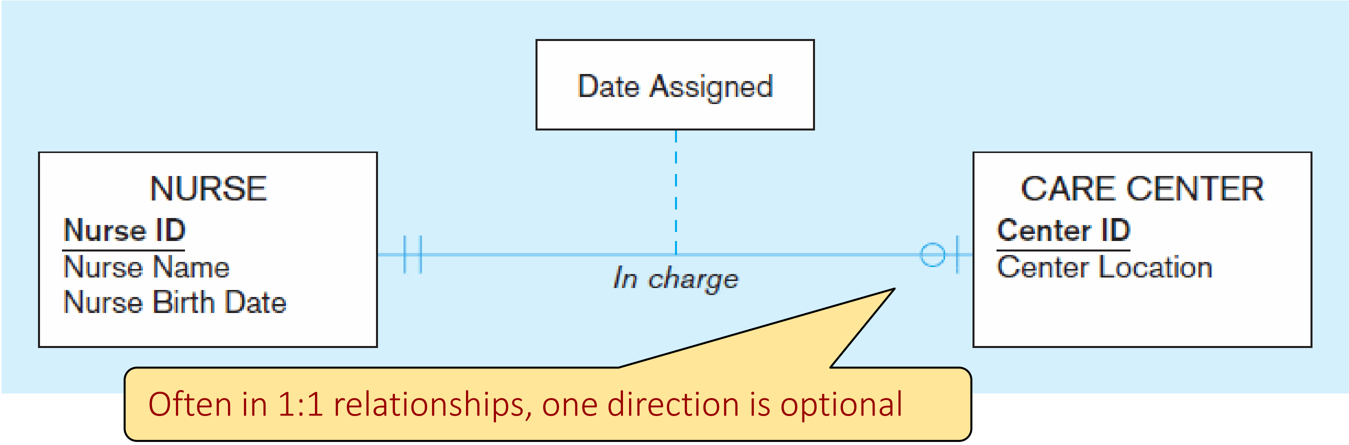
ORDER (<u>OrderID</u>, OrderDate, CustomerID*)
FK (CustomerID) references CUSTOMER

Foreign key

Note: If FK is part of the key, need to be added at the beginning of the relation and **underlined** with a **star**.
If FK is not part of the key, should be added at the end of the relation with a **star**.

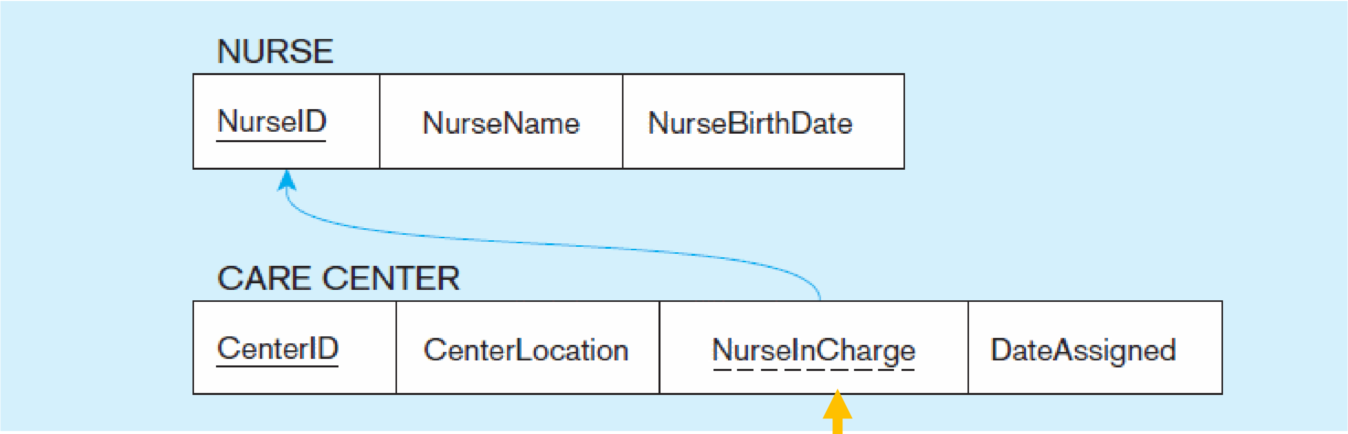
3.3.2. Mapping a binary 1:1 relationship (Figure 4-14)

a) In charge relationship (1:1)



Rule: in 1:1 relationships, PK of the entity on the Mandatory side will be FK in the entity on the Optional side

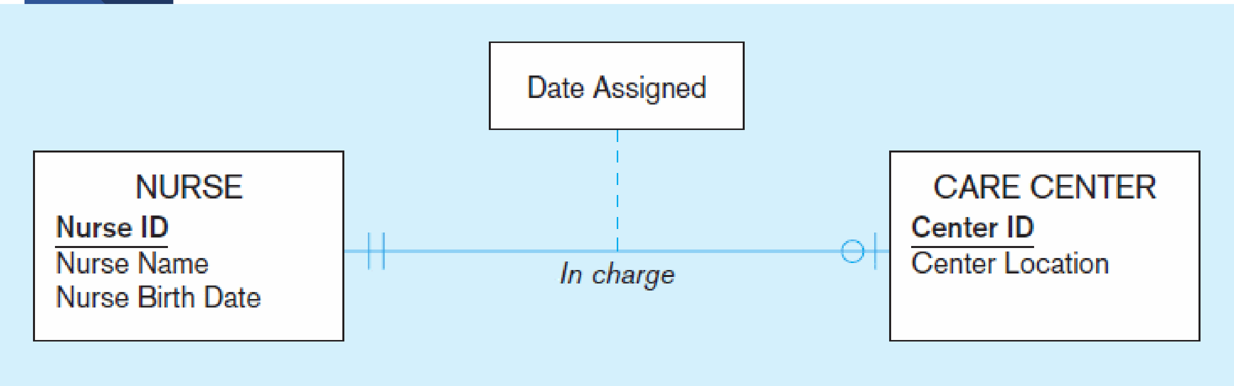
b) Resulting relations

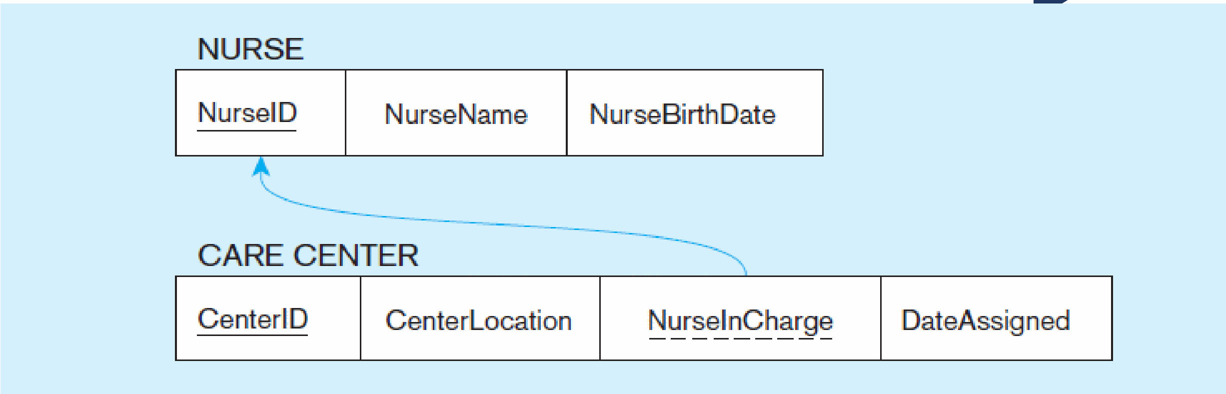


Foreign key goes in the relation on the optional side, matching the primary key on the mandatory side

Example of mapping an associative entity with an identifier (Figure 4-16)

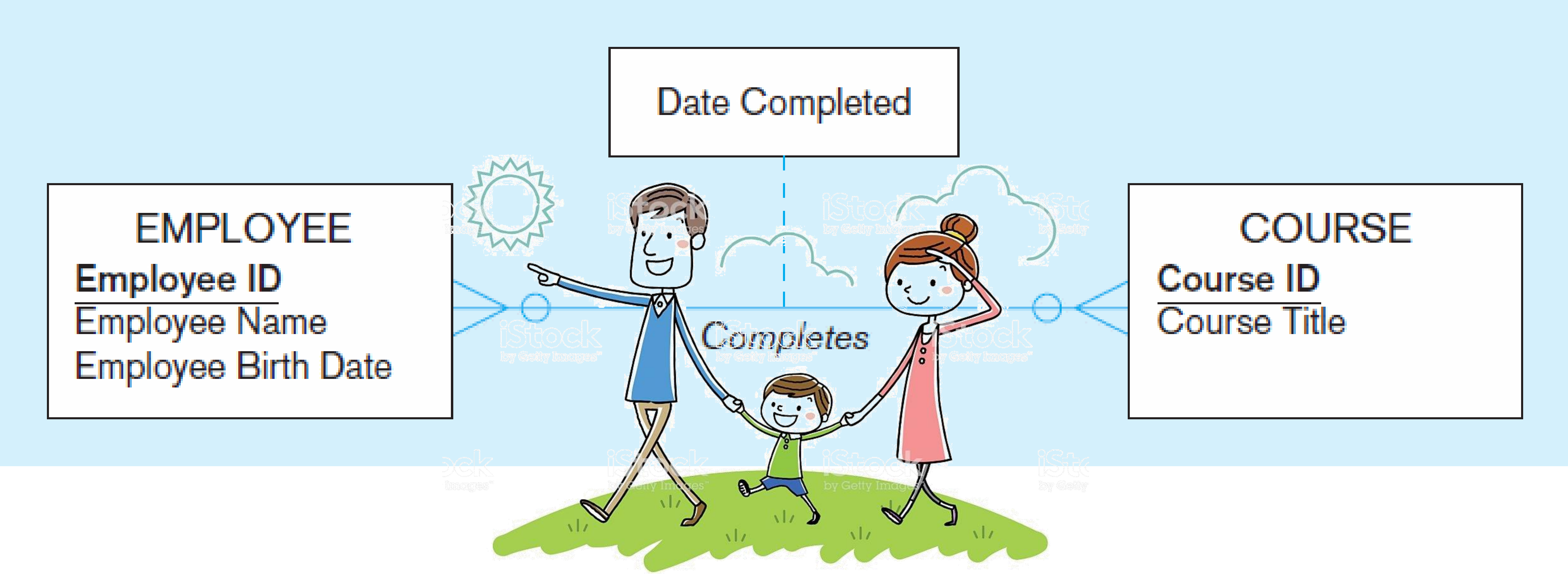
Class Activity 4.4: Convert the following ERD to the relations using the second format. (5 min) → in the live lecture





3.3.3. Mapping an M:N Relationship OR Associative Entities

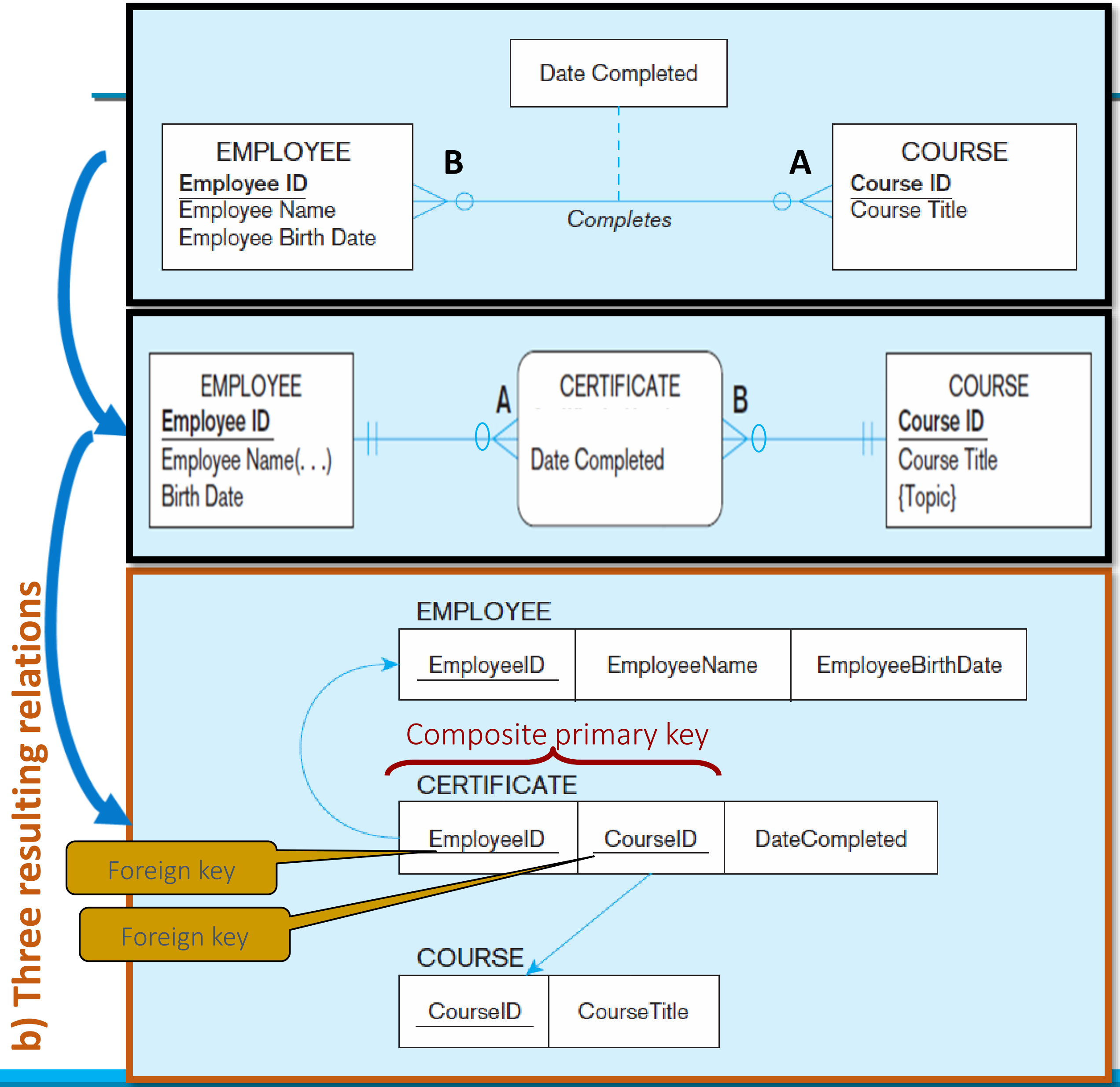
a) Completes relationship (M:N)



The *Completes* relationship will need to become a separate relation.

(Figure 4-13)

3.3.3. Mapping an M:N relationship (Figure 4-13) (cont.)



EMPLOYEE (EmployeeID, EmployeeName, EmployeeBirthDate)

CERTIFICATE (EmployeeID*, CourseID*, DateCompleted)

FK (EmployeeID) references EMPLOYEE

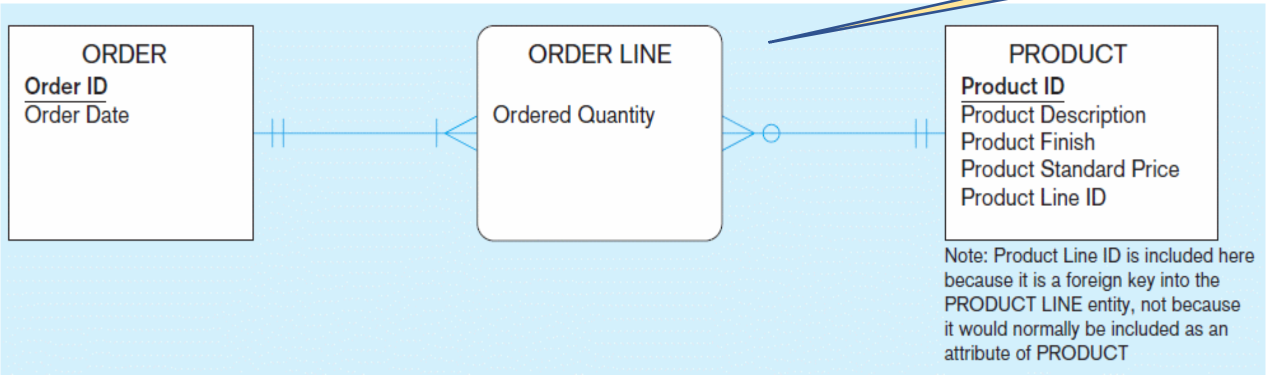
FK (CourseID) references COURSE

COURSE (CourseID, CourseTitle)

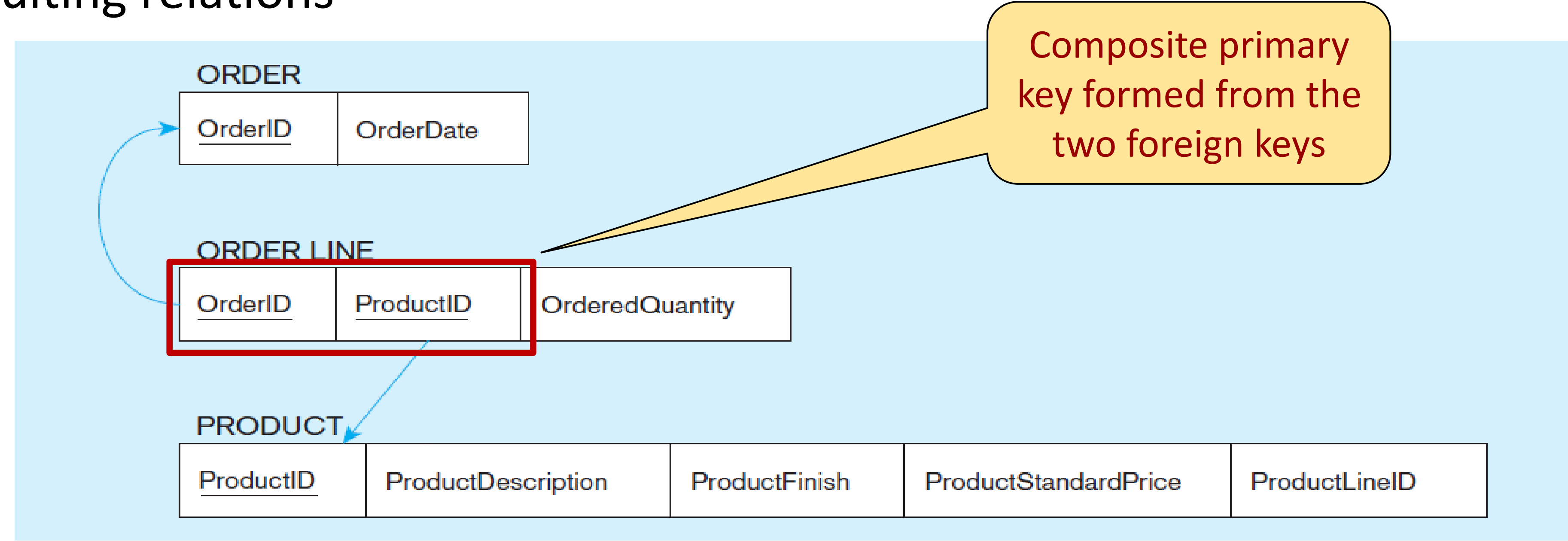
new *intersection relation related to the* ***associative*** *entity*

Another example of mapping an associative entity (Figure 4-1)

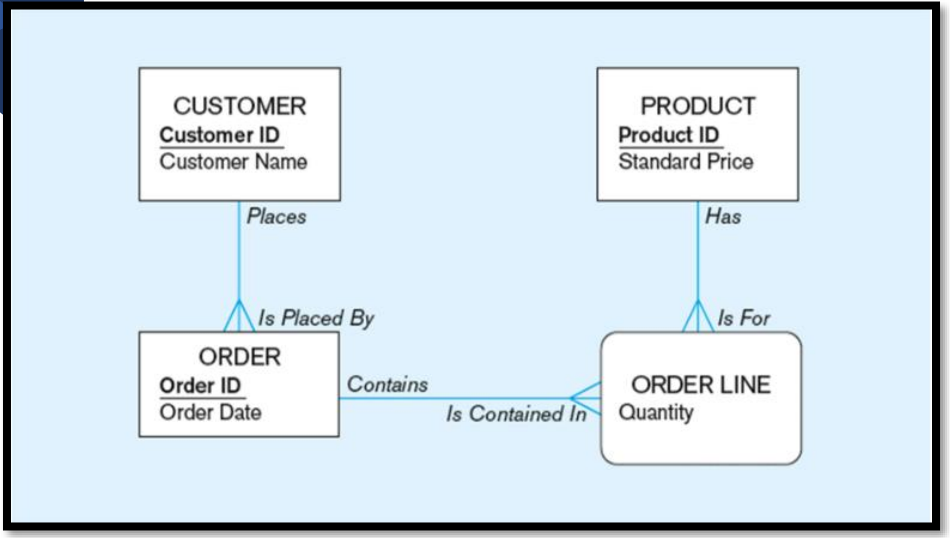
An associative entity



b) Three resulting relations

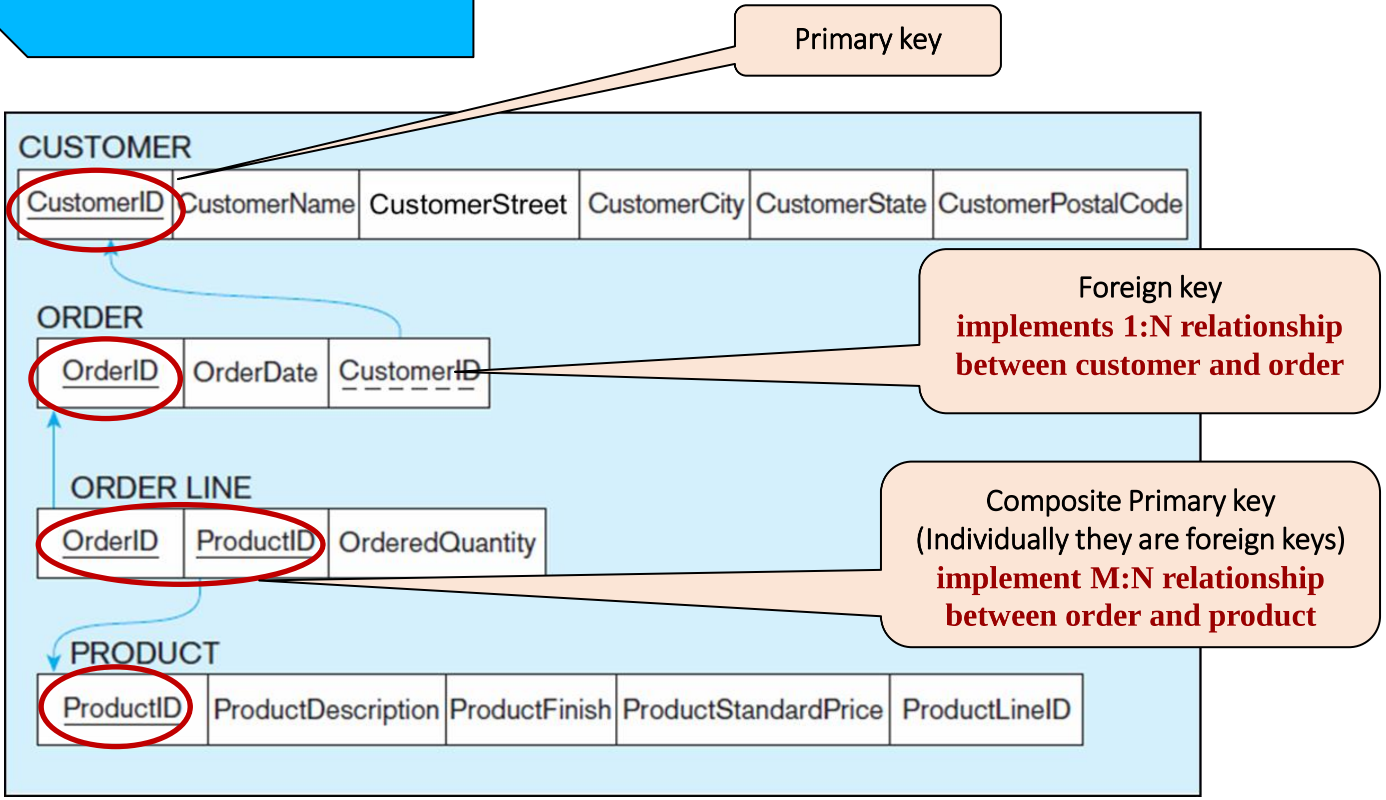


Class Activity 4.5: Convert the following ERD to relations. (10 min) → in the live lecture



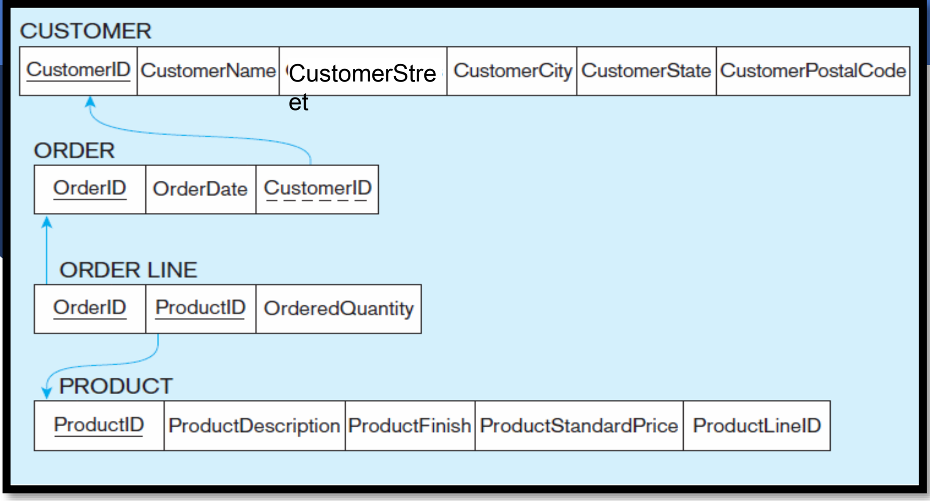
ERD

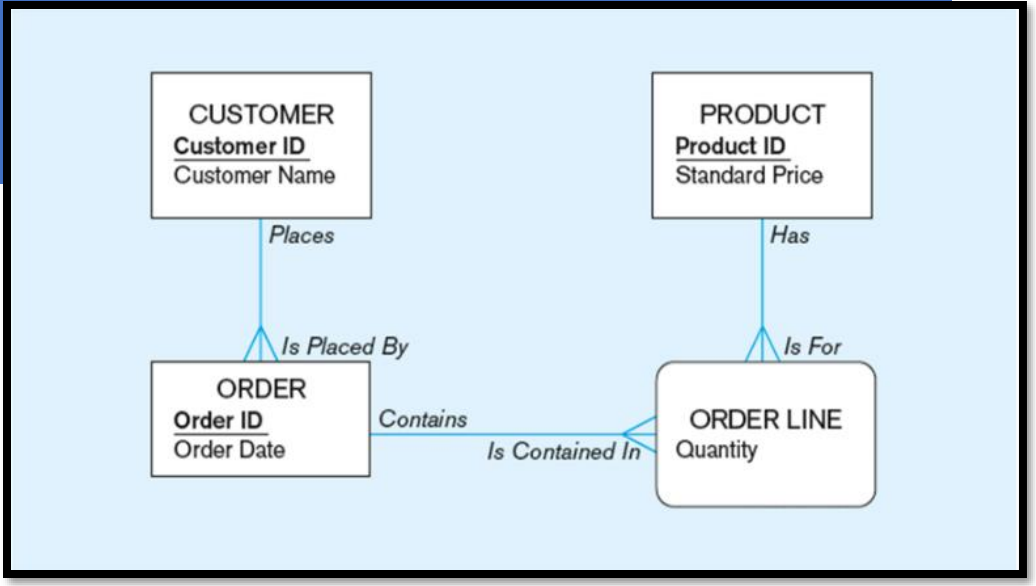
Resulting Relations



Referential integrity constraints are drawn via **arrows** from dependent to parent table

Solution to Class Activity 4.5:
Convert the following ERD to relations.

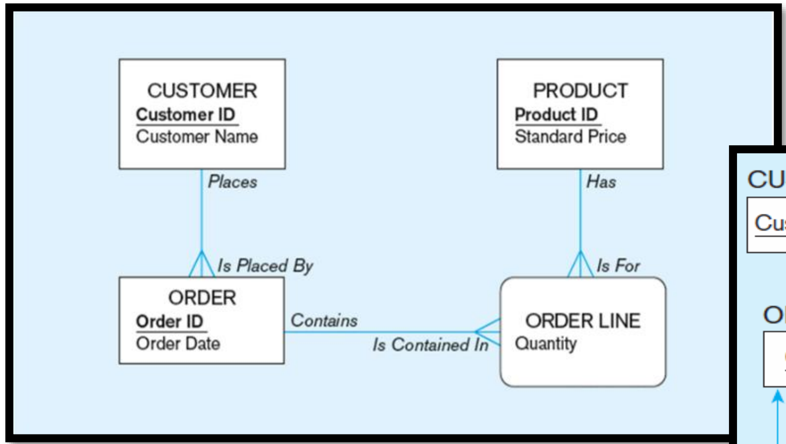


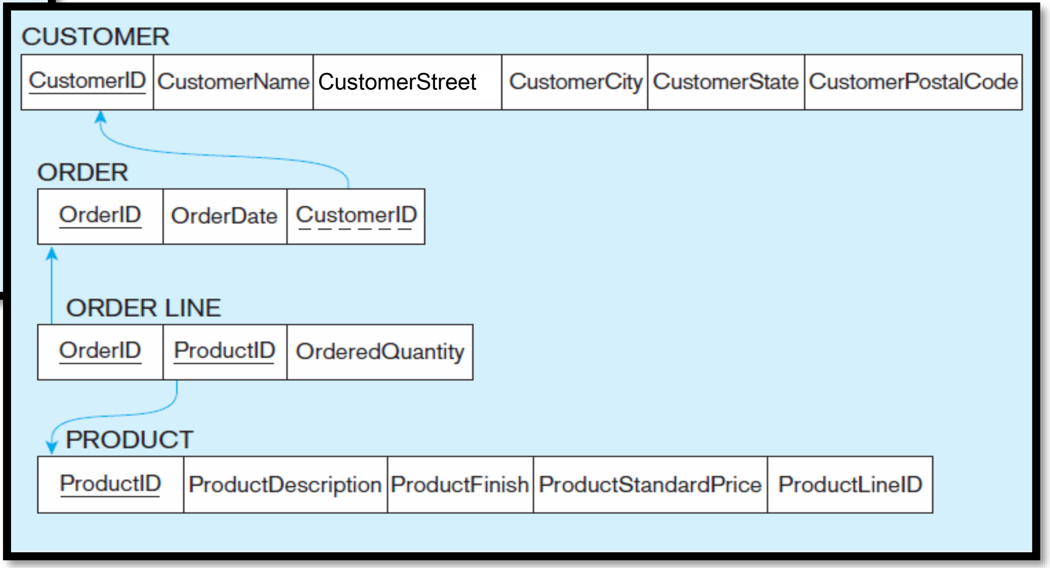


Transforming ERD into Relations

Explore in Class 4.1:

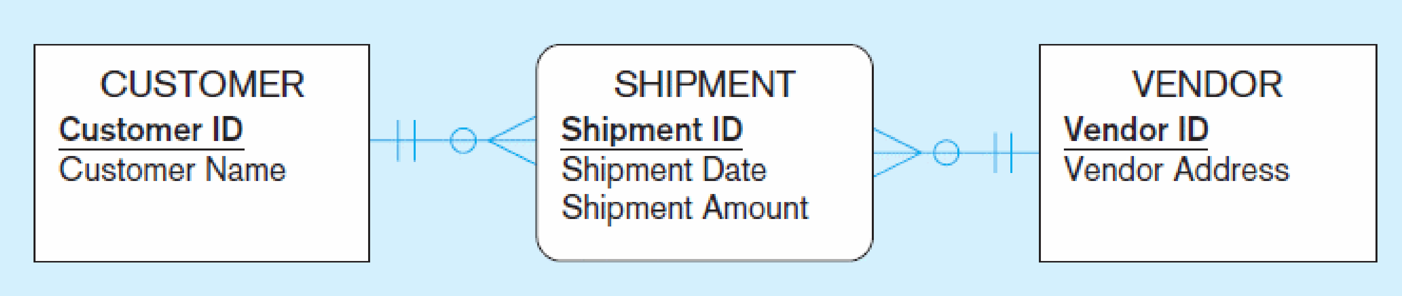
Let's jump to Ed and write some query to create these relations.





Example of mapping an associative entity with an identifier (Figure 4-16)

Class Activity 4.6: Convert the associative entity in this ERD to a relation? (5 min) → in the live lecture



Video 4.5:

Mapping Ternary (and n-ary) Relationships

- One relation for each entity and one for the associative entity

- Associative entity has foreign keys form each entity in the relationship. These FKs will be part of its composite PK.

Question: Will you make a composite PK? or you will generate a surrogate key?

?

Mapping a ternary relationship (Figure 4-19)

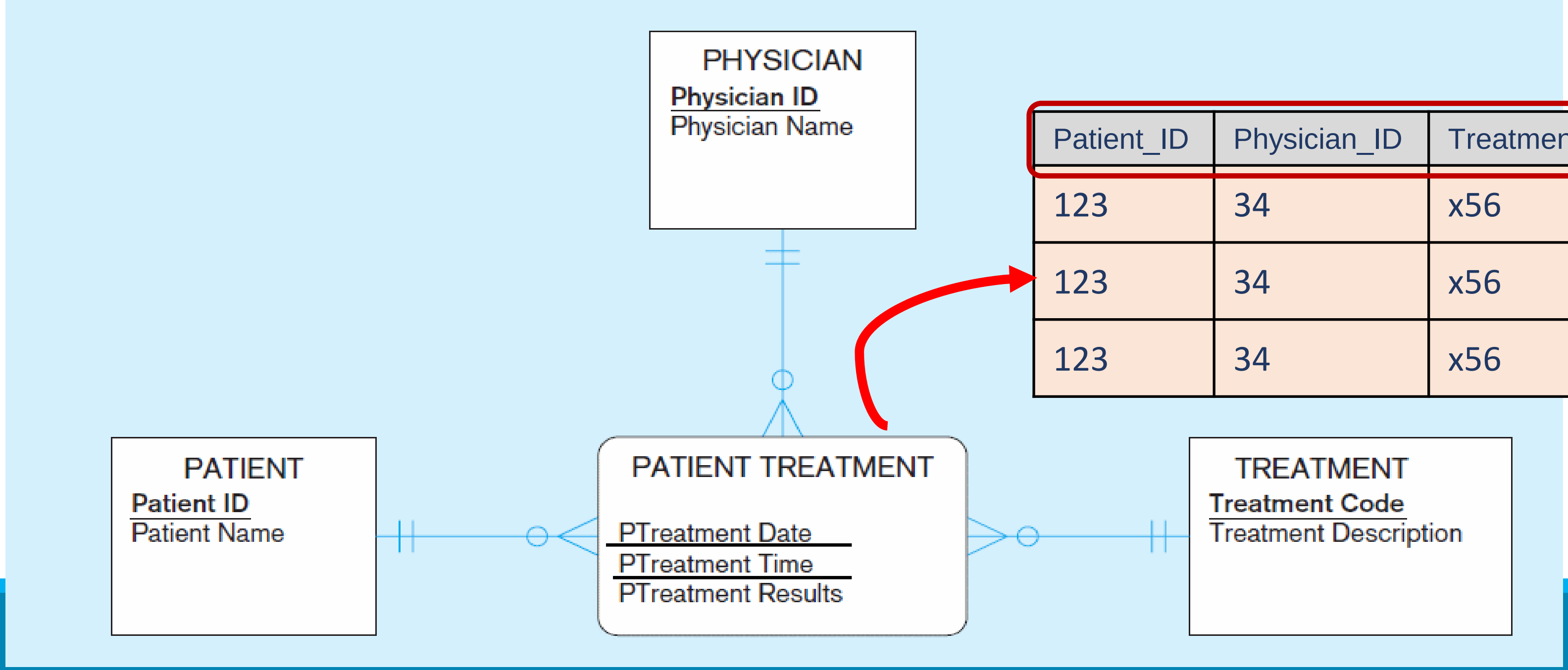
a) PATIENT TREATMENT Ternary relationship with associative entity

BR: Many **physician** can prescribe many **treatment** for many **patient, many times a day.**

So, you may have … a **physician** that prescribed the **same treatment** for the **same patient** on the **same day**.

So, you may have … **multiple records** with **the same** Physician_ID, Treatment_code, Patient_ID, and PTTreatmentDate

That's why we need to include PTreatmentTime as well to make a **unique composite key** …

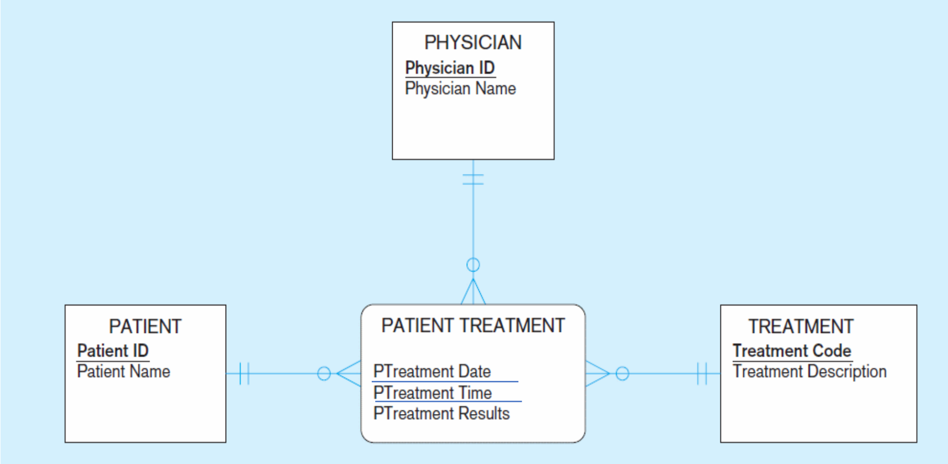


Patient_ID	Physician_ID	Treatment_code	PTreatmentDate	PTreatmentTime	PTreatmentResults
123	34	x56	1/1/2020	12:00	Pain
123	34	x56	1/1/2020	13:00	No pain
123	34	x56	1/1/2020	18:30	Normal

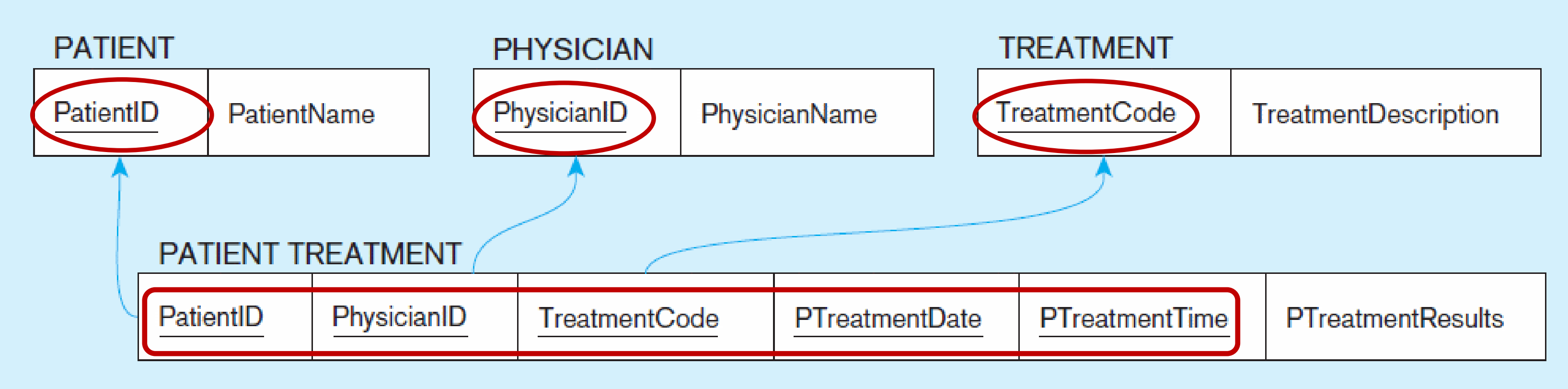
Mapping a ternary relationship (Figure 4-19) (cont.)

BR: Many **physician** can prescribe many **treatment** for many **patient, many times a day.**

Patient_ID	Physician_ID	Treatment_code	PTreatmentDate	PTreatmentTime	PTreatmentResults
123	34	x56	1/1/2020	12:00	Pain
123	34	x56	1/1/2020	13:00	No pain
123	34	x56	1/1/2020	18:30	Normal



b) Mapping the ternary relationship PATIENT TREATMENT



Based on the BR, we may have **multiple records** with **the same** Physicia_ID, Patient_ID, Treatment_code, and PTTreatmentDate

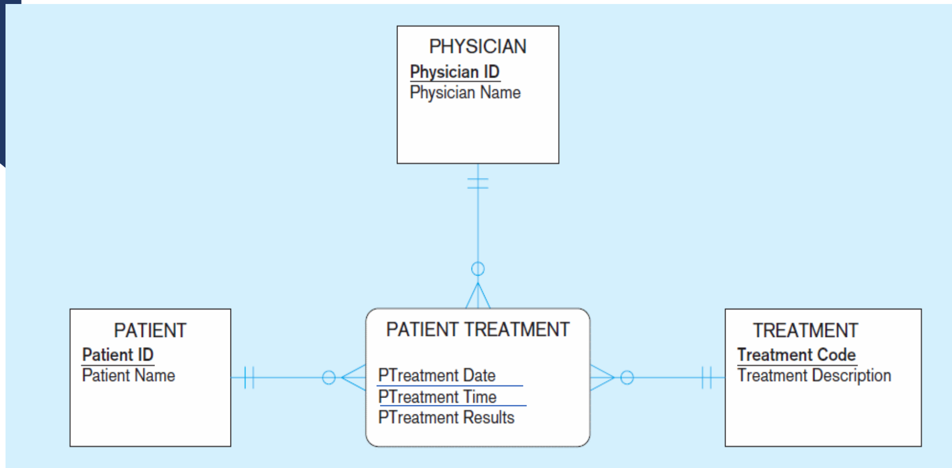
To create a unique PK, treatment **date** and **time** are included in the composite primary key.

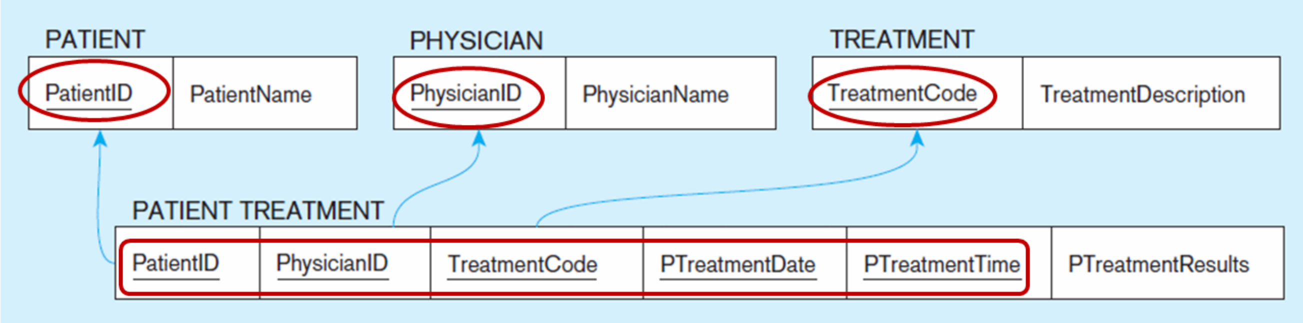
But this makes a very cumbersome key…

It would be better to create a **surrogate** key like Treatment#.

Class Activity 4.7: Convert the following ERD to relations. (10 min)

→ in the live lecture

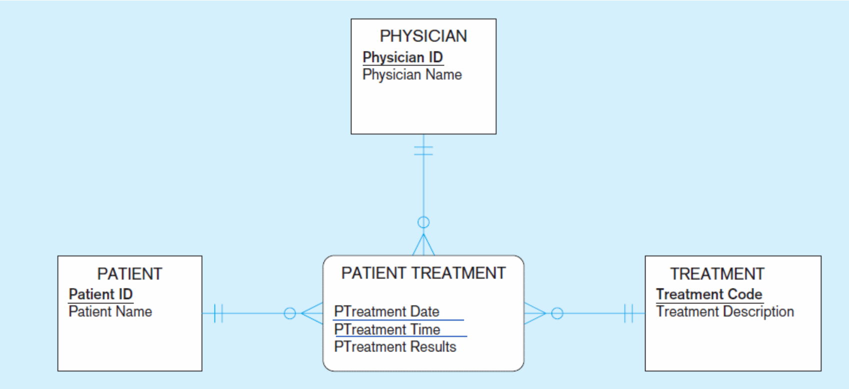


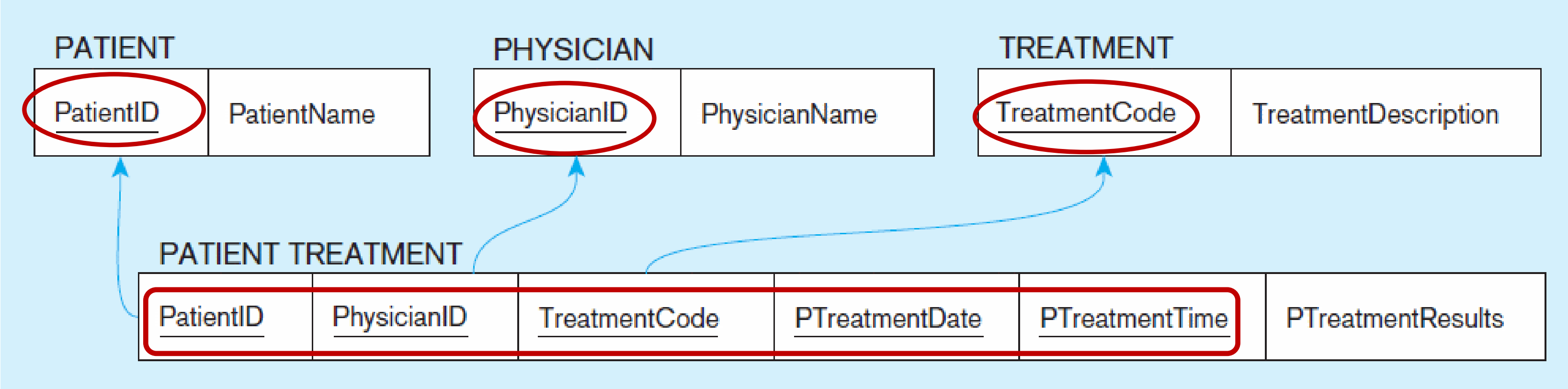


The PK of an Associative entity

Assessment Challenge :

Creating the PATIEN_TTREATMENT table and insert three rows in the created table.





BR: Many **physician** can prescribe many **treatment** for many **patient, many times a day.**

Based on the BR, we may have **multiple records** with **the same** Physicia_ID, Patient_ID, Treatment_code, and PTTreatmentDate

PAUSE

Video 4.6:

Mapping Weak Entities

- The weak entity becomes a separate relation with a foreign key taken from the superior entity
- The primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

NOTE: **Foreign keys can have null values**, **but** the **domain constraint** for the foreign key <u>should NOT allow null value</u> for a weak entity, or an associative entity, or if it is related to a mandatory cardinality.

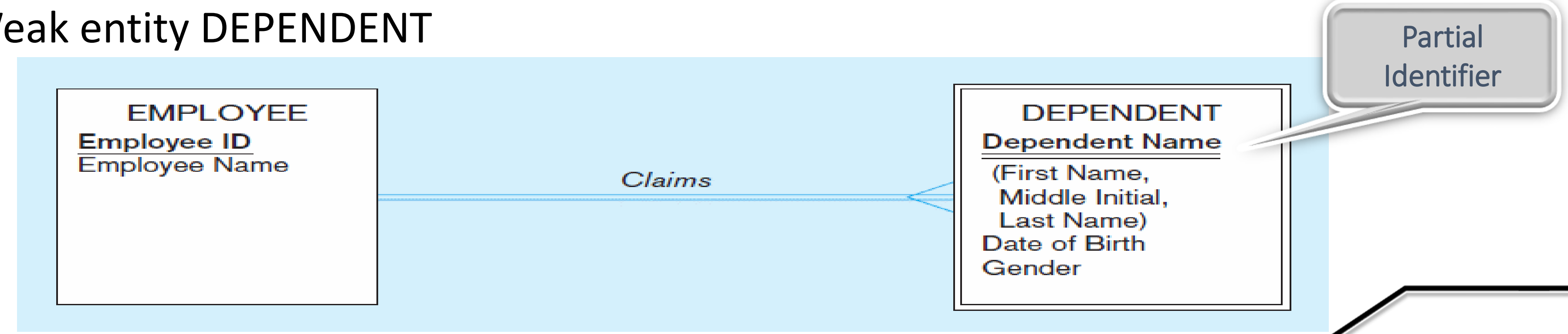
Assessment Challenge 4.2:

This fact refers to which two integrity constraints?

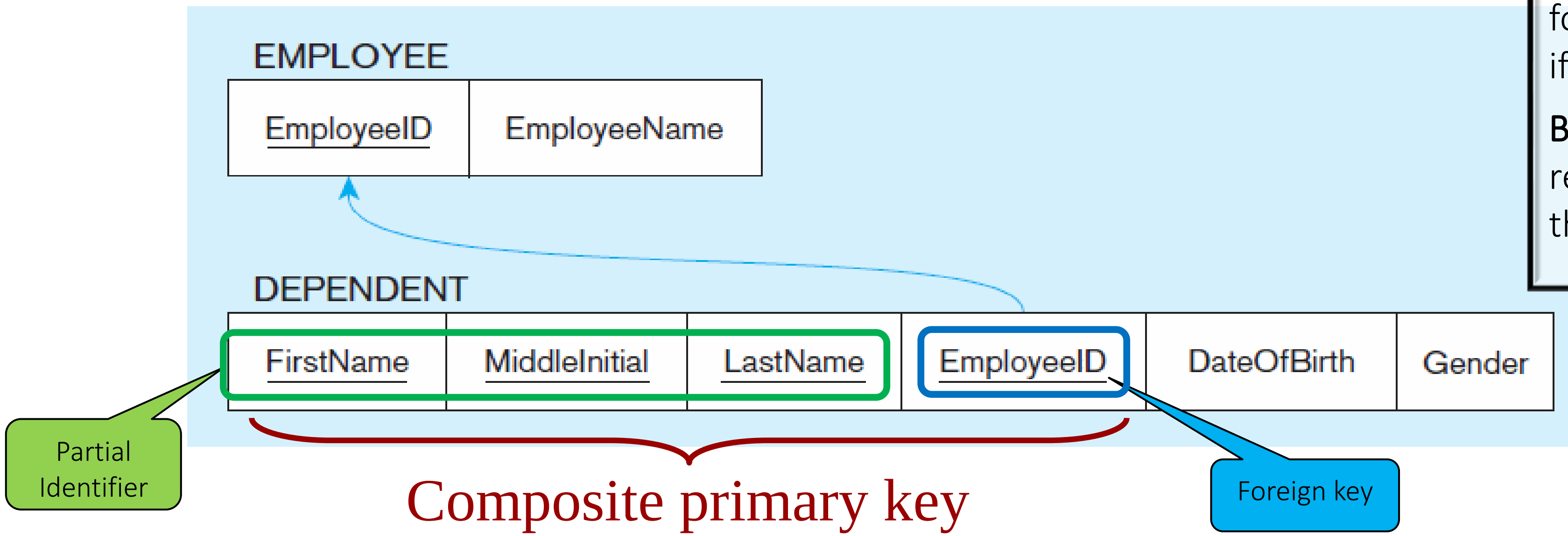
Using which part of the "create table scripts" we can implement this fact?

Example of mapping a weak entity (Figure 4-11)

a) Weak entity DEPENDENT



b) Relations resulting from weak entity



NOTE: the domain constraint for the foreign key <u>should NOT allow *null*</u> value if DEPENDENT is a weak entity.

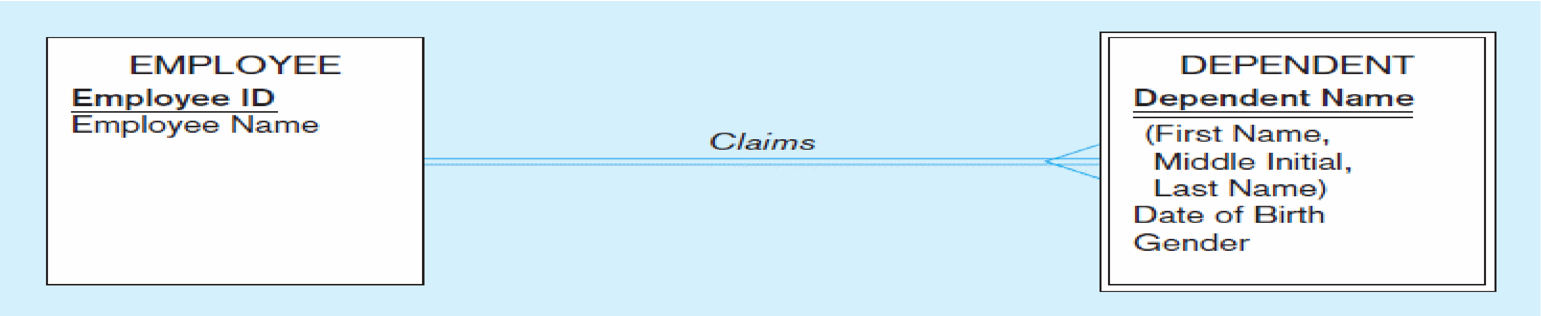
Because in a weak entity the FK that refers to "PK of Strong Entity" is part of the composite PK, so it can't be null.

The PK of an Weak entity

Explore in Class 4.2:

Let’s jump to Ed and write some query to create these relations.

BR: Each Employee have many dependents.



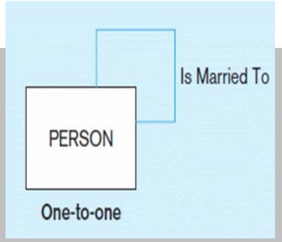
EMPLOYEE (EmployeeID, Employee_F_Name, Employee_L_Name)

DEPENDENT (EmployeeID*, Dependent_F_Name, Dependent_M_Name, Dependent_L_Name, DOB, Gender)
FK (EmployeeID) references EMPLOYEE

NOTE: the domain constraint for the foreign key should NOT allow *null* value if DEPENDENT is a weak entity

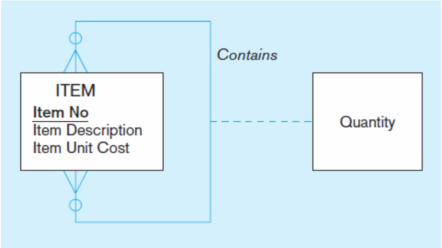
Video 4.7: Mapping Unary Relationships

3.6. Mapping Unary Relationships



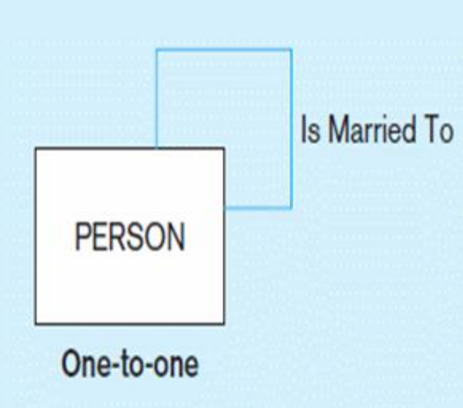
- **One-to-One and One-to-Many**: Recursive foreign key in the same relation

- **Many-to-Many**: Two relations
  - One for the entity type
  - One for an **associative relation** in which the primary key has two attributes, **both taken from the primary key of the entity**

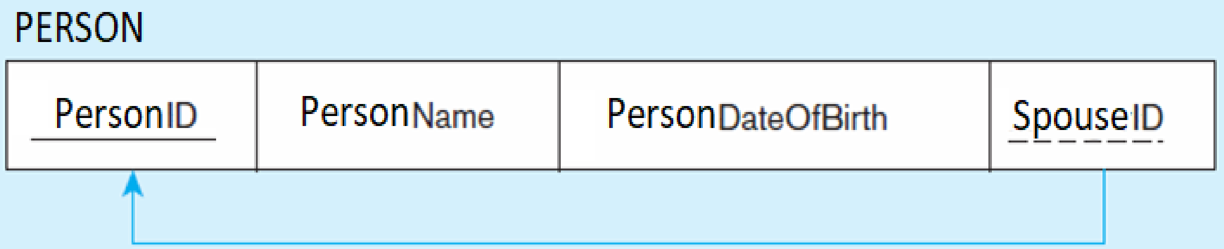


Mapping a unary 1:1 relationship: Recursive foreign key in the same relation

(a) **PERSON** entity with unary relationship



(b) PERSON relation with recursive foreign key



PERSON (<u>PersonID</u>, PersonName, PersonDateOfBirth, SpouseID*)
FK (SpouseID) references PERSON

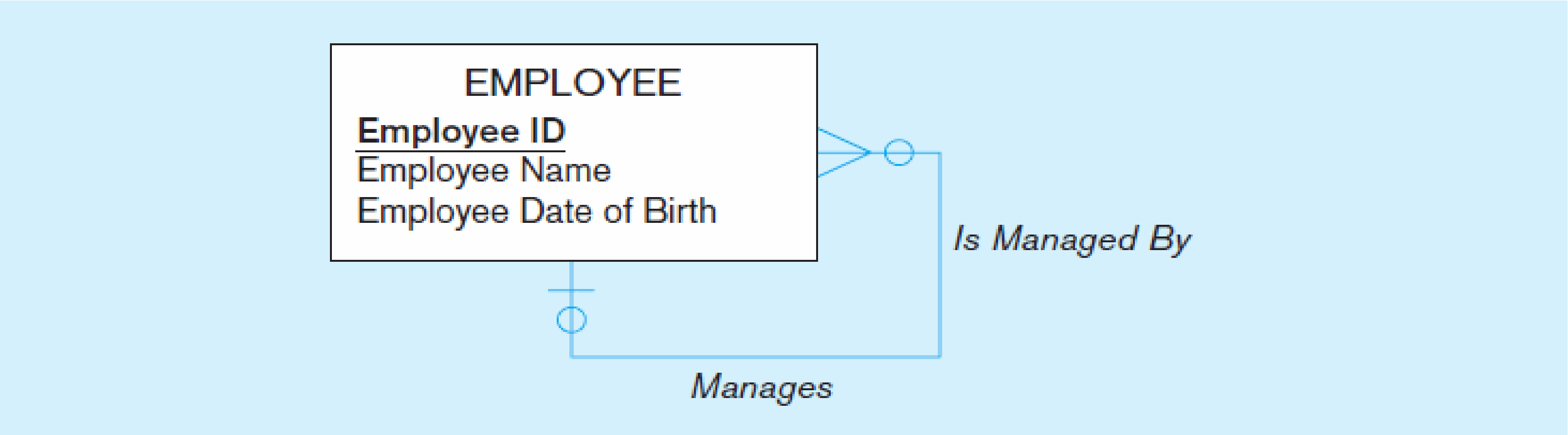
PK FK

PERSON

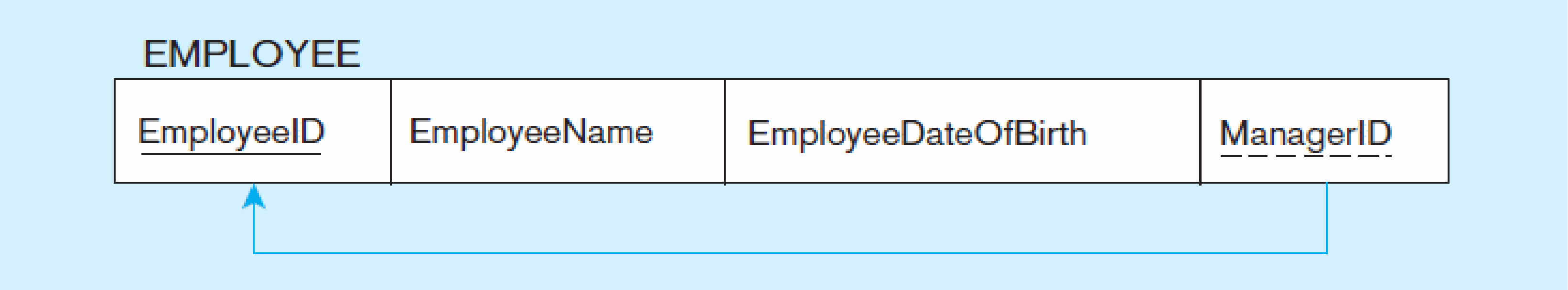
PersonID	PersonName	PersonDateOfBirth	Spouse ID
1234587	Jack	1/1/1980	2387468
3459087	Michael	5/2/1990	
2387468	Sara	5/8/1975	
5745321	Fahimeh	9/2/1983	
8743836	Ricky	5/11/1992	

Mapping a unary 1:N relationship: Recursive foreign key in the same relation

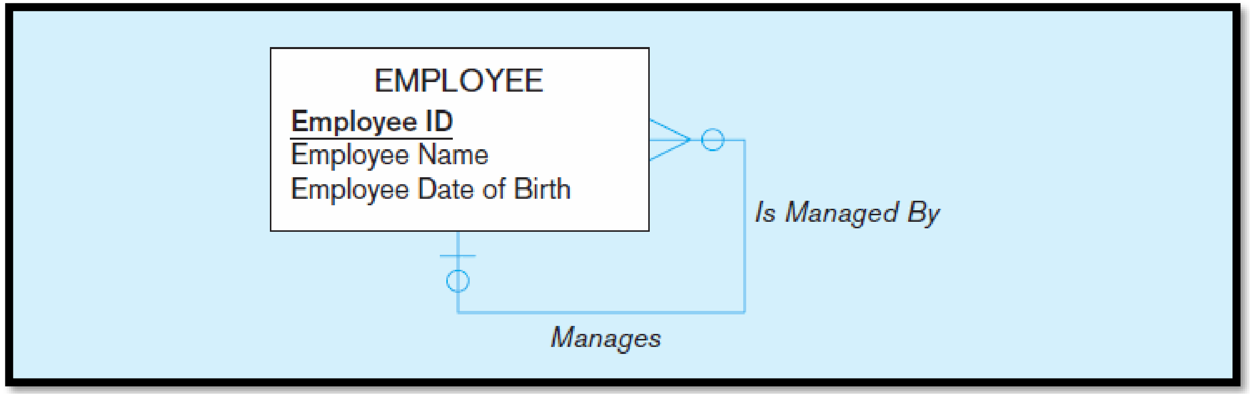
(a) **EMPLOYEE** entity with unary relationship

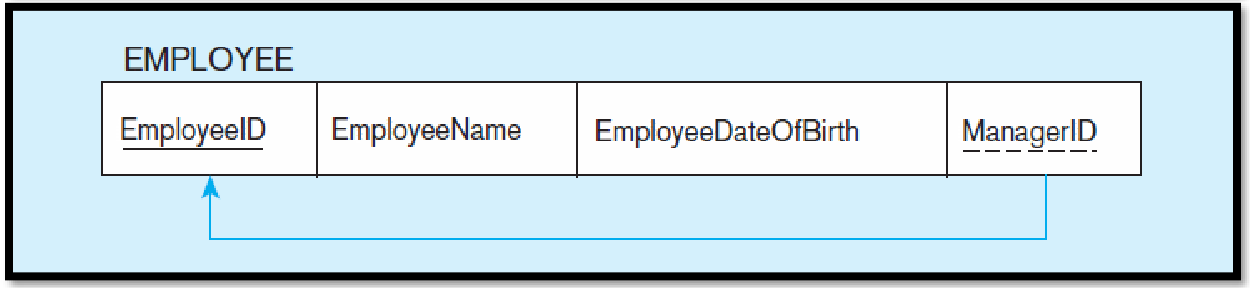


(b) EMPLOYEE relation with recursive foreign key



EMPLOYEE (<u>EmployeeID</u>, EmployeeName, EmployeeDateOfBirth, ManagerID*)
FK (ManagerID) references EMPLOYEE





EMPLOYEE (EmployeeID, EmployeeName, EmployeeDateOfBirth, ManagerID*)
FK (ManagerID) references EMPLOYEE

PK FK

Employee_ID	Employee_Name	Employee_DateOfBirth	Manager_ID
1123	Sara	1.1.2000	7892
1456	Jake	1.1.2000	7892
7892	Fahimeh	1.1.1970	1245
1245	Julia	1.1.1980	…
…	…	…	…

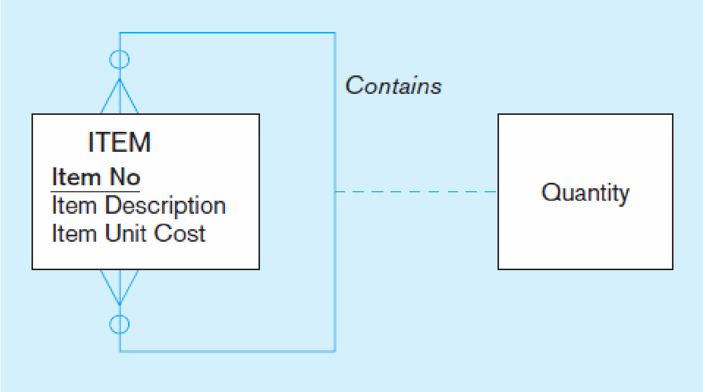
Mapping a unary M:N relationship

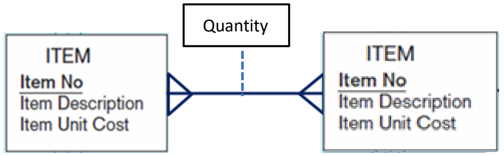
Two relations:

- One for the **entity type**
- One for an **associative relation** in which the **primary key** has two attributes**, both taken from the primary key of the entity**

BR: Any Item can have many Component and any Component will be related to many Items.

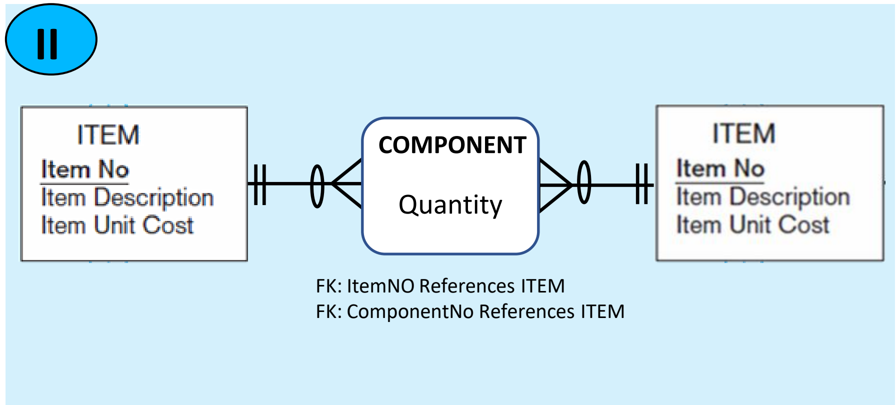
There is a M:N relationship between ITEM and COMPONENT

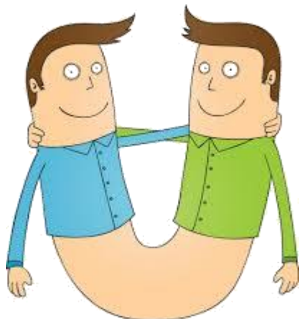


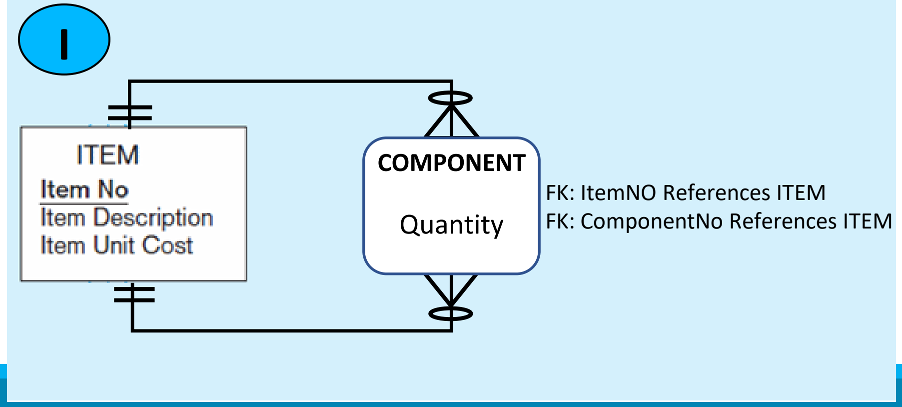


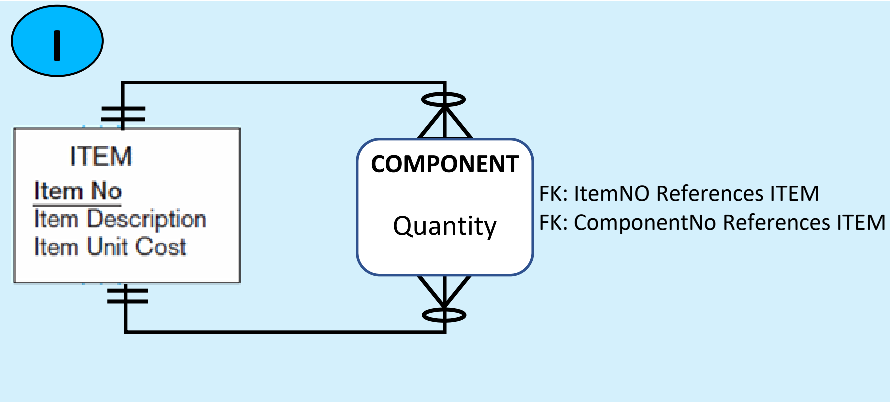
ERD II is just provided to clarify ERD I.

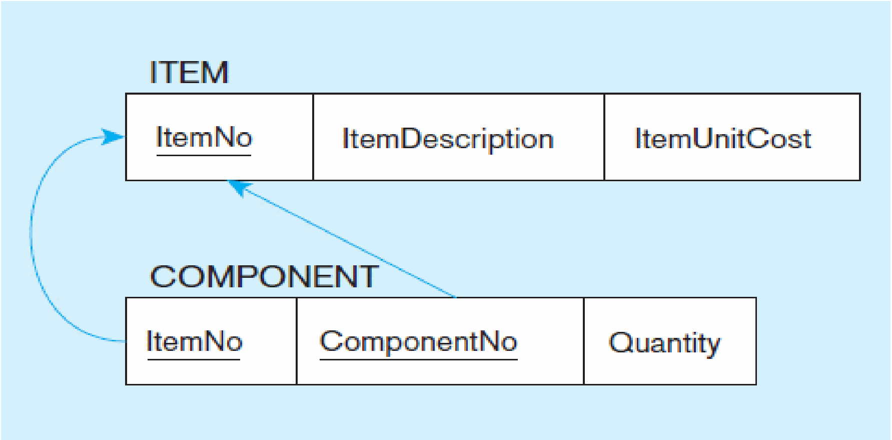
ERD II is not a standard ERD









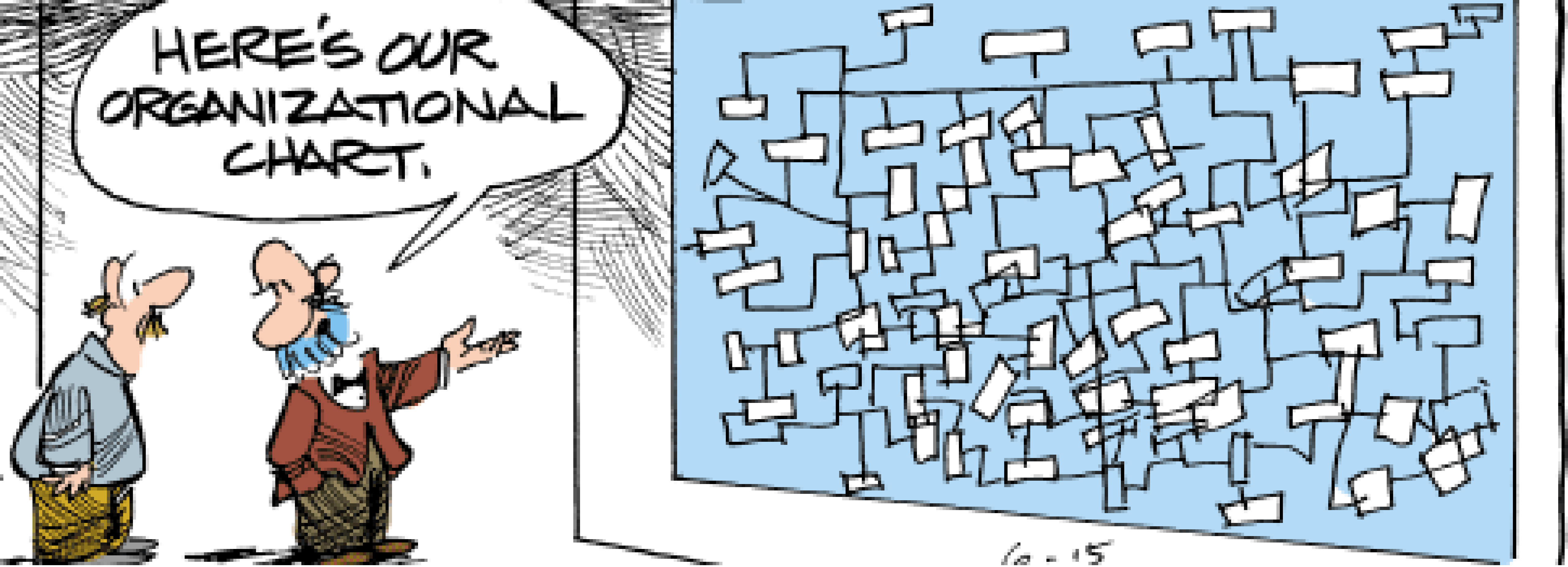


ITEM

Item_No	Item_Description	Item_Unit_Cost
12	Wheel	50
13	Spoke	0.5
14	Rim	30
15	Valve	5

COMPONENT

Item_No	Component_No	Quantity
12	13	30
12	14	1
12	15	1



Transforming ERD into Relations

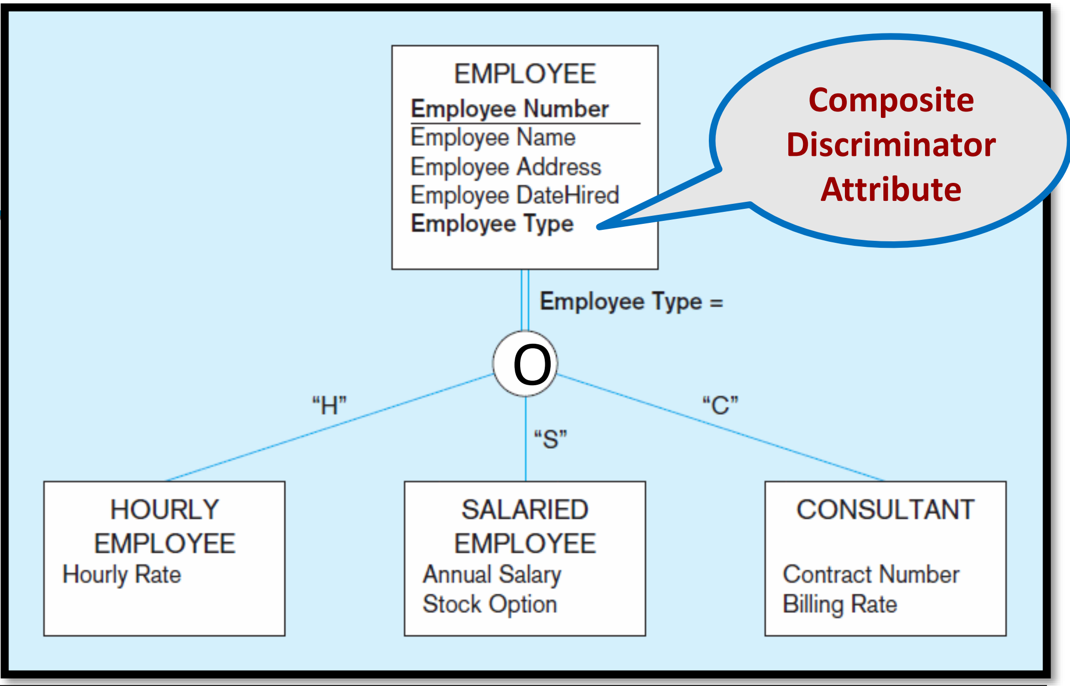
Video 4.8:
Mapping Supertype/Subtype Relationships

3.7. Mapping Supertype/Subtype Relationships

- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table

Supertype/subtype relationships (Figure 4-20)

- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table



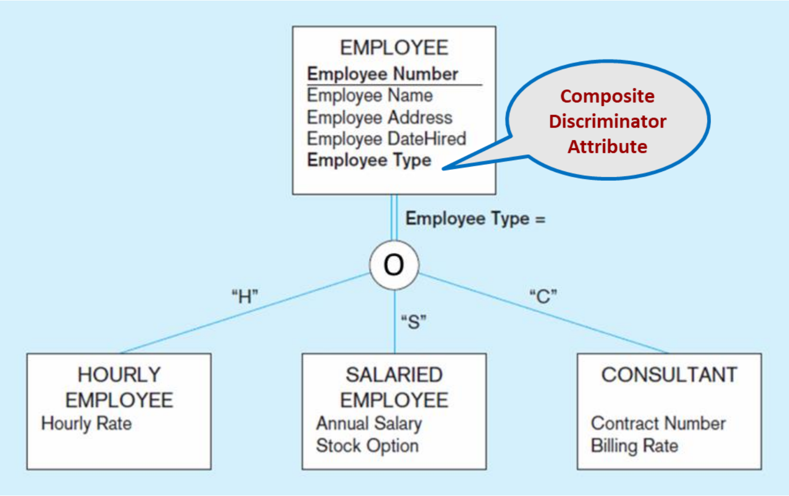
EMPLOYEE (Employee_Number, Employee_Name, Employee_Street, Employee_City, Employee_State, Employee_CustomerPostalCode, Employee_Date_Hired, **Employee_Type**)

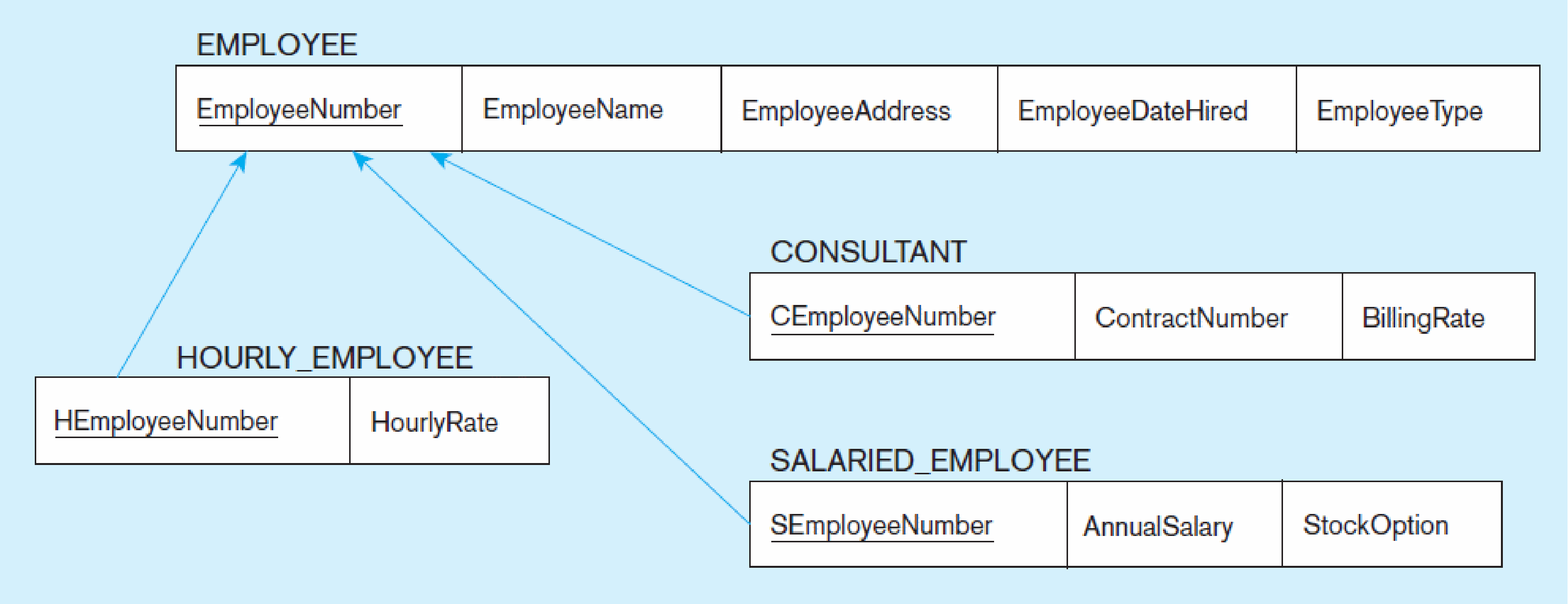
HOURLY_EMPLOYEE (H_Employee_Number*, Hourly_Rate)
FK (H_Employee_Number) references EMPLOYEE

SALARIED_EMPLOYEE (S_Employee_Number*, Annual_Salary, Stock_Option)
FK (S_Employee_Number) references EMPLOYEE

HOURLY_EMPLOYEE (C_Employee_Number*, Contract_Number, Billing_Rate)
FK (C_Employee_Number) references EMPLOYEE

Mapping supertype/subtype relationships to relation





(Figure 4-21)

These are implemented as one-to-one relationships.

Note: This is the best method to map supertype/subtypes to relations. There are other two methods that will be discussed in the related tutorial.

Introducing a subtype discriminator (overlap rule)

EMPLOYEE (Employee_Number, Employee_Name, Employee_Street, Employee_City, Employee_State, Employee_CustomerPostalCode, Employee_Date_Hired, **Employee_Type**)

HOURLY_EMPLOYEE (H_Employee_Number*, Hourly_Rate)
FK (H_Employee_Number) references EMPLOYEE

SALARIED_EMPLOYEE (S_Employee_Number*, Annual_Salary, Stock_Option)
FK (S_Employee_Number) references EMPLOYEE

HOURLY_EMPLOYEE (C_Employee_Number*, Contract_Number, Billing_Rate)
FK (C_Employee_Number) references EMPLOYEE

Composite Discriminator Attribute

Employee_Number	Employee_Name	Employee_Street	Employee_City	Employee_State	Employee__CustomerPostalCode	Date_Hire	Employe_type (H? S? C?)
1123	Sara			UTS		1/1/2014	YNY
1456	Jake			32/50 …		5/8/2013	NYN
7892	Fahimeh			12/97 …		2/3/2013	NNY

O

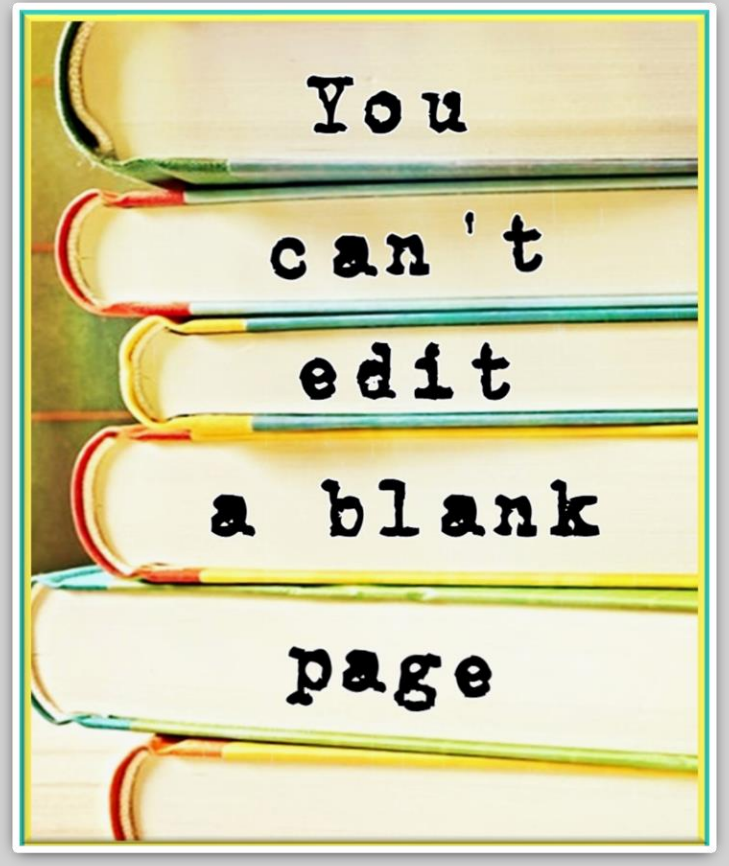
H_Employee_Number	Hourly_Rate
1123	80

S_EmployeeNumber	Annual_Salary	Stock_Option
1456	70000	0.2

C_Employee_Number	Contract_Number	Billing_Rate
7892	9856	50
1123	9812	30

Summary

- List properties of relations.
- Transform E-R and EER diagrams to relations.
- Create tables with entity and relational integrity constraints.



Next Lecture…

1. **Terms to know to Do Normalization**
    1.1. Functional Dependencies
    1.2. Keys: Super-key, Candidate key and Primary Key
    1.3. Determining Candidate Keys from FDs
    1.4. Partial Functional Dependencies
    1.5. Transitive Functional Dependencies
2. **Data Normalization and Well-Structured Relations**
3. **Steps in normalization**
4. **First Normal Form**
5. **Second Normal Form**
6. **Third Normal Form**