TRANSFORMING DATA MODELS INTO DATABASE DESIGNS

revised by 김태연

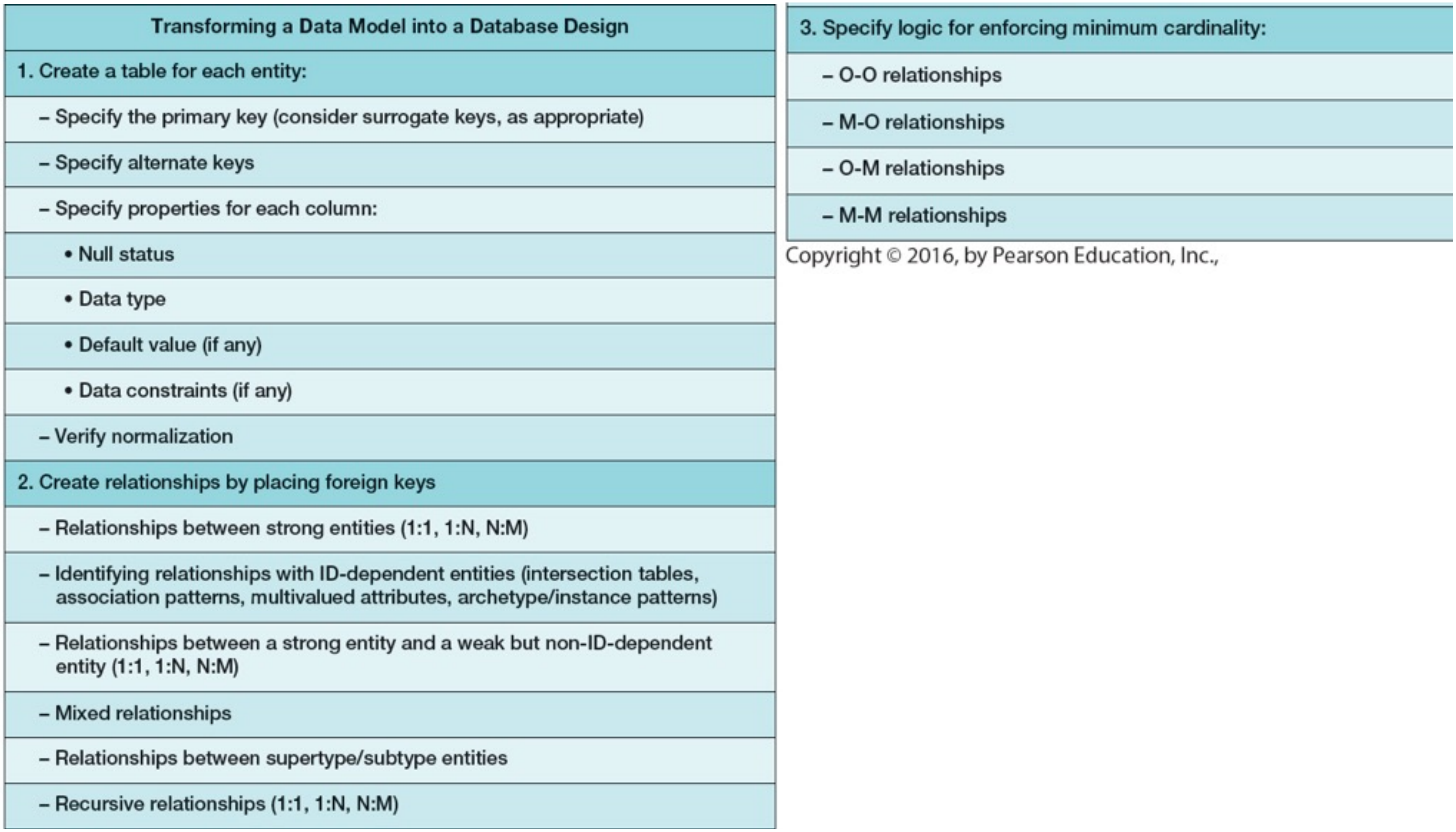
OBJECTIVES

- ➤ To understand how to transform data models into database designs
- ➤ To be able to identify primary keys and understand when to use a surrogate key
- ➤ To understand the use of referential integrity constraints
- ➤ To be able to represent Strong, ID-dependent, 1:1, 1:N, and N:M relationships as tables
- ➤ To be able to represent weak entities as tables
- ➤ To be able to represent supertypes/subtypes as tables
- ➤ To be able to implement referential integrity actions required by minimum cardinalities

PURPOSE OF A DATABASE DESIGN

- A database design is a set of database specifications that can actually be implemented as a database in a DBMS.
- The data model we discussed in Chapter 5 is a generalized, non-DBMS specific design.
- A database design, on the other hand, is a DBMS specific design intended to be implemented in a DBMS product such as Microsoft SQL Server 2012 or Oracle Database 11g Release 2 or MySQL Server 5.6.
- Books on systems analysis and design often identify three design stages:
  - Conceptual design (conceptual schema)
  - Logical design (logical schema)
  - Physical design (physical schema)
- The database design we are discussing is basically equivalent to the logical design, which is defined in these books as the conceptual design implemented in a specific DBMS product.

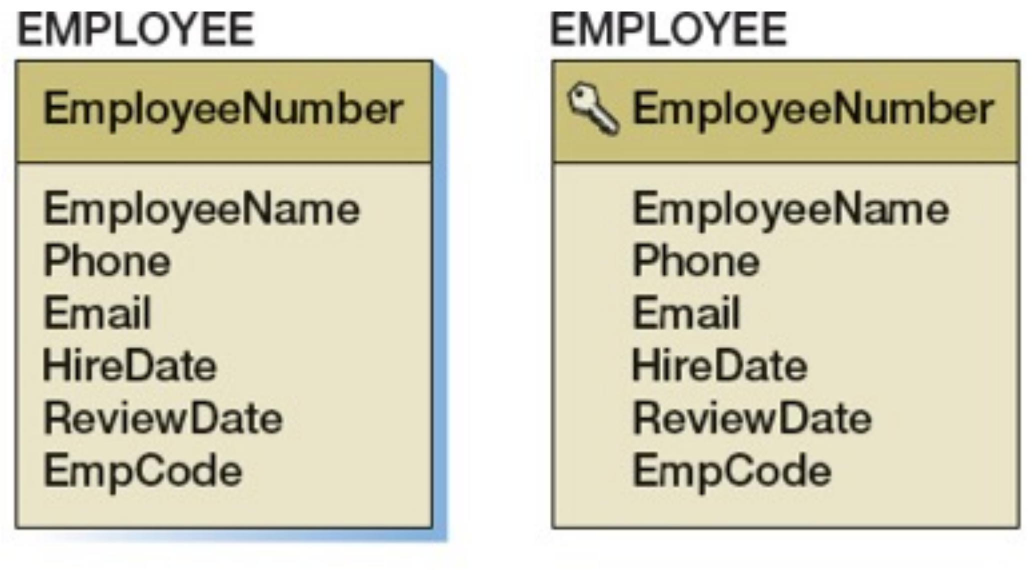
STEPS FOR TRANSFORMING A DATA MODEL INTO A DATABASE DESIGN



Transforming a Data Model into a Database Design
1. Create a table for each entity:
– Specify the primary key (consider surrogate keys, as appropriate)
– Specify alternate keys
– Specify properties for each column:
• Null status
• Data type
• Default value (if any)
• Data constraints (if any)
– Verify normalization
2. Create relationships by placing foreign keys
– Relationships between strong entities (1:1, 1:N, N:M)
– Identifying relationships with ID-dependent entities (intersection tables, association patterns, multivalued attributes, archetype/instance patterns)
– Relationships between a strong entity and a weak but non-ID-dependent entity (1:1, 1:N, N:M)
– Mixed relationships
– Relationships between supertype/subtype entities
– Recursive relationships (1:1, 1:N, N:M)
3. Specify logic for enforcing minimum cardinality:
– O-O relationships
– M-O relationships
– O-M relationships
– M-M relationships

CREATE A TABLE FOR EACH ENTITY

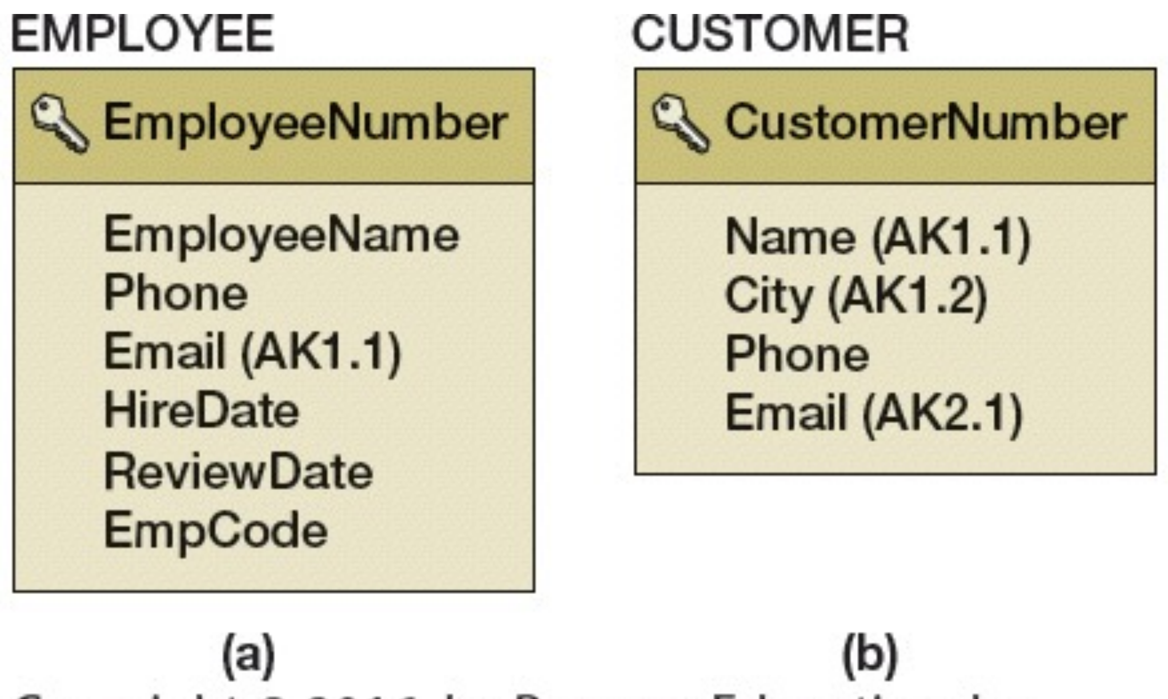
- Create a Table for Each Entity
- Select the Primary Key
    - The ideal primary key is short, numeric, and fixed.
    - Surrogate keys meet the ideal, but have no meaning to users.
- Surrogate key
    - A surrogate key is a DBMS-supplied identifier of each row of a table.
    - Surrogate key values are unique within the table, and they never change.
    - Their values have no meaning to a user.
    - When data are shared among different databases.



(a) EMPLOYEE Entity (b) EMPLOYEE Table

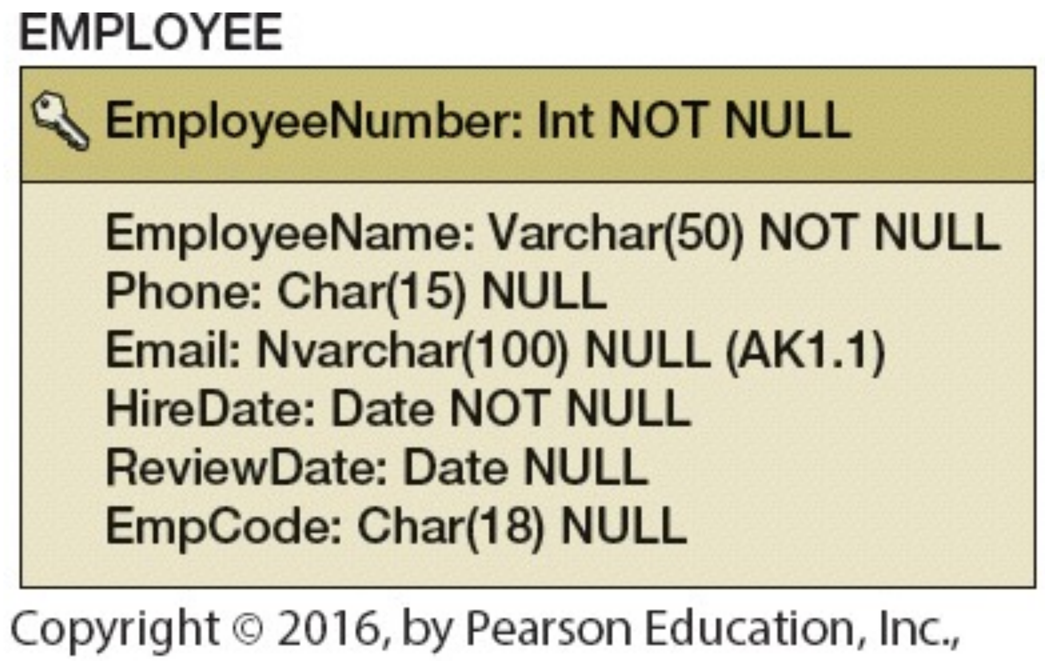
SPECIFY CANDIDATE (ALTERNATE) KEYS

- The terms candidate key and alternate key are synonymous.
- Candidate keys are alternate identifiers of unique rows in a table.
- Will use AK*n*.*m* notation, where *n* is the number of the alternate key, and *m* is the column number in that alternate key.



SPECIFY COLUMN PROPERTIES

- Null Status
  - Null status indicates whether or not the value of the column can be NULL.
- Data Type
  - Generic data types: CHAR(n), VARCHAR(n), DATE, TIME, MONEY, INTEGER, DECIMAL



EMPLOYEE

EmployeeNumber: Int NOT NULL

EmployeeName: Varchar(50) NOT NULL
Phone: Char(15) NULL
Email: Nvarchar(100) NULL (AK1.1)
HireDate: Date NOT NULL
ReviewDate: Date NULL
EmpCode: Char(18) NULL

- MySQL 5.6 Data Types
  - http://dev.mysql.com/doc/refman/5.6/en/data-types.html

ENTITY ATTRIBUTE DISPLAY IN DATA MODELS

- Default Value
  - A default value is the value supplied by the DBMS when a new row is created.

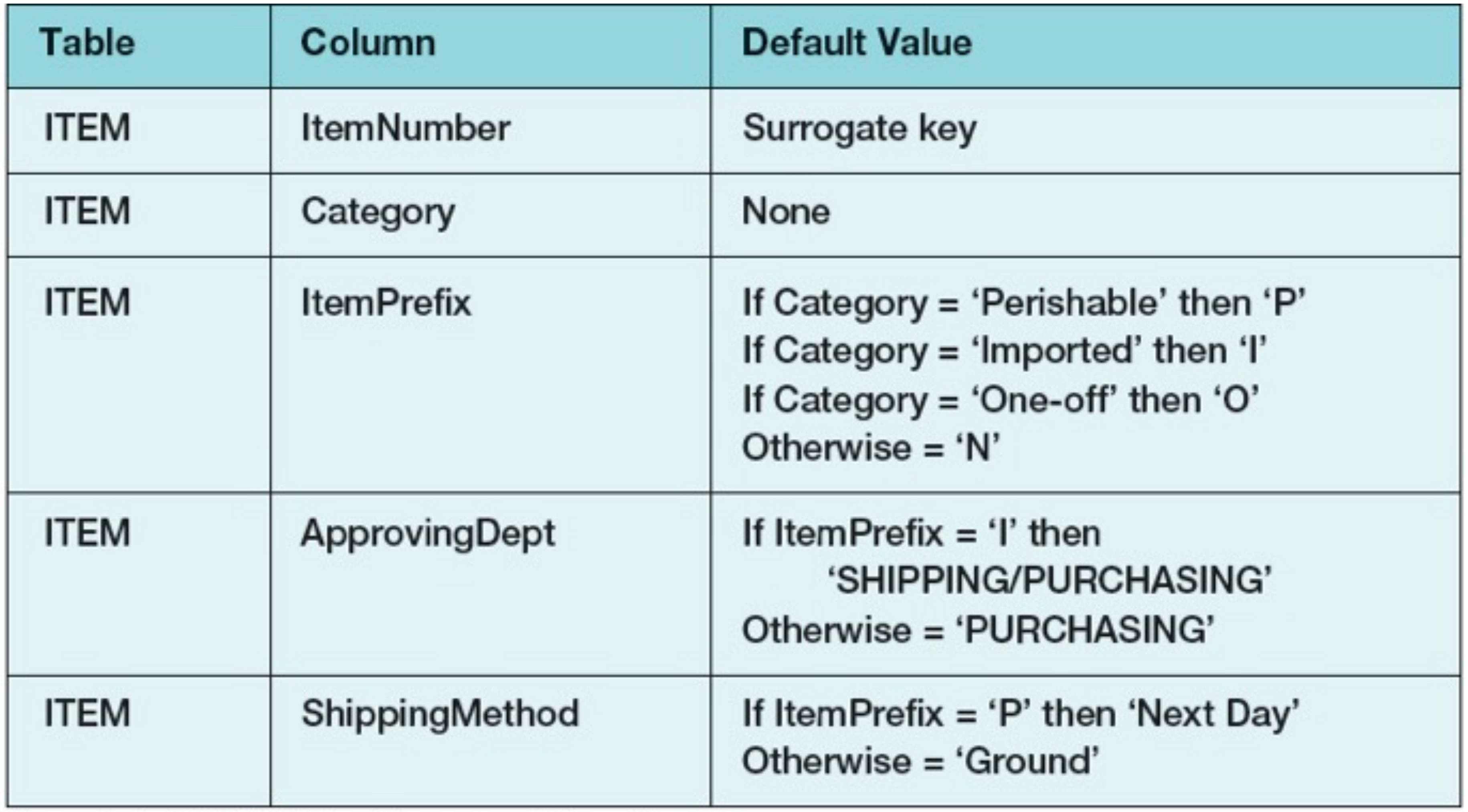


Table	Column	Default Value
ITEM	ItemNumber	Surrogate key
ITEM	Category	None
ITEM	ItemPrefix	If Category = 'Perishable' then 'P' If Category = 'Imported' then 'I' If Category = 'One-off' then 'O' Otherwise = 'N'
ITEM	ApprovingDept	If ItemPrefix = 'I' then 'SHIPPING/PURCHASING' Otherwise = 'PURCHASING'
ITEM	ShippingMethod	If ItemPrefix = 'P' then 'Next Day' Otherwise = 'Ground'

SPECIFY COLUMN PROPERTIES: DATA CONSTRAINTS

- Data constraints are limitations on data values:
  - **Domain constraint**: column values must be in a given set of specific values.
  - **Range constraint**: column values must be within a given range of values.
  - **Intrarelation constraint**: column values are limited by comparison to values in other columns in the same table.
  - **Interrelation constraint**: column values are limited by comparison to values in other columns in other tables (referential integrity constraints on foreign keys).

VERIFY NORMALIZATION

- The tables should be normalized based on the data model.
- Verify that all tables are:
  - 1NF

    건물점검대장 (건물번호, 점검일자, 점검시간, 건물주소, 점검내용, 직원번호, 직원명, 차량번호)

  - 2NF

    건물점검대장 (건물번호, 점검일자, 점검시간, 점검내용, 직원번호, 직원명, 차량번호)
    건물 (건물번호, 건물주소)

  - 3NF

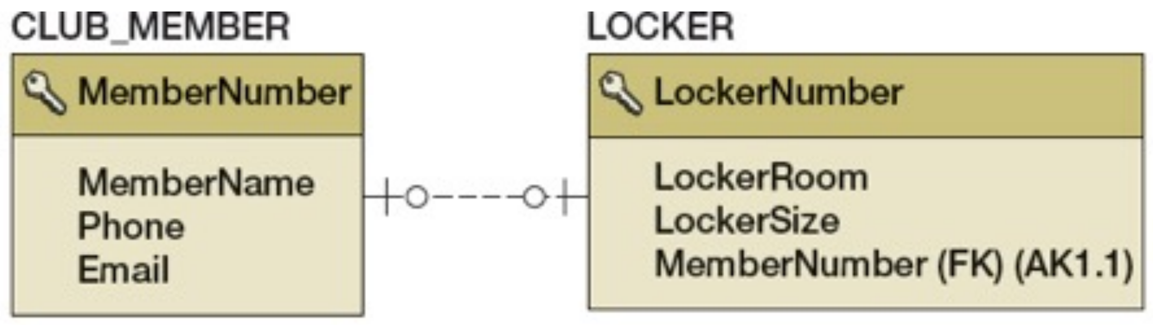
    건물점검대장 (건물번호, 점검일자, 점검시간, 점검내용, 직원번호, 차량번호)
    건물 (건물번호, 건물주소)
    직원 (직원번호, 직원명)

  - BCNF

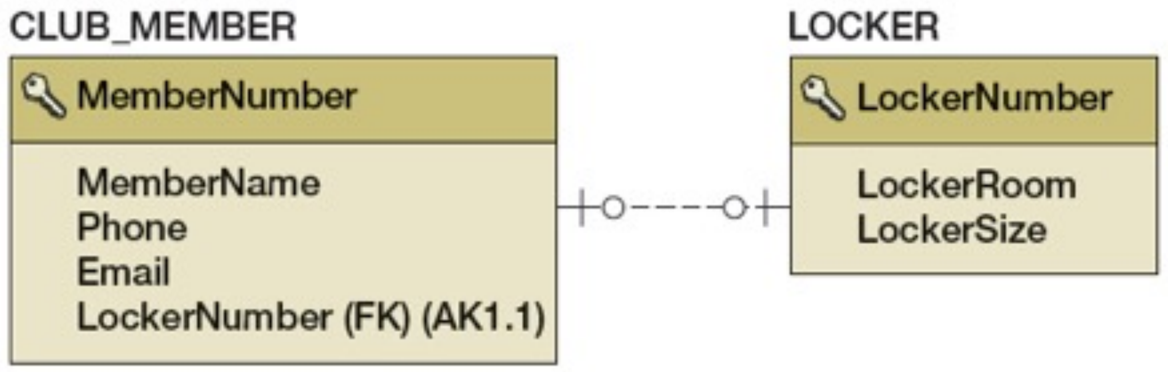
    건물점검대장 (건물번호, 점검일자, 점검시간, 점검내용, 직원번호)
    건물 (건물번호, 건물주소)
    직원 (직원번호, 직원명)
    이용차량 (직원번호, 점검일자, 차량번호)

CREATE RELATIONSHIPS: STRONG ENTITY RELATIONSHIPS

- Place the key of one table in the other table. Enforce the maximum cardinality by defining the foreign key as unique.
- Place the primary key of the parent in the child as a foreign key.

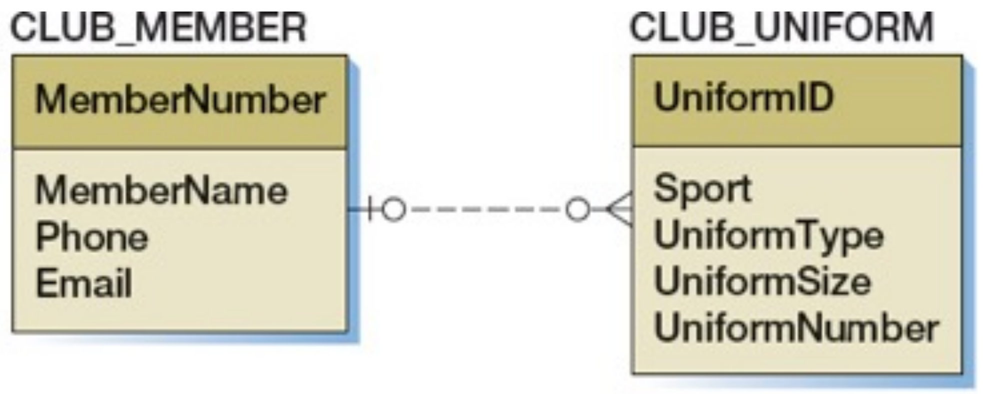


(a) With Foreign Key in LOCKER

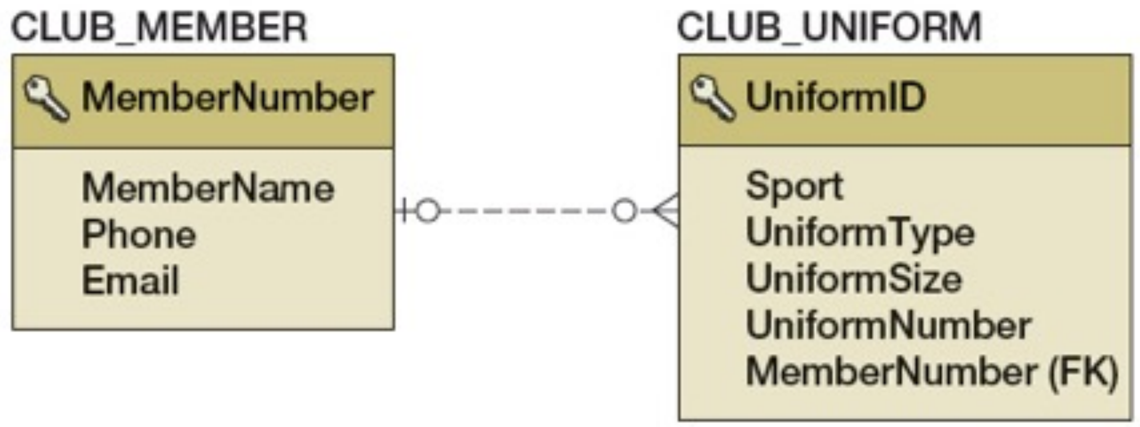


(b) With Foreign Key in CLUB_MEMBER

1:1 Relationships



(a) 1:N Relationship Between Strong Entities



(b) Placing the Primary Key of the Parent in the Child as a Foreign Key

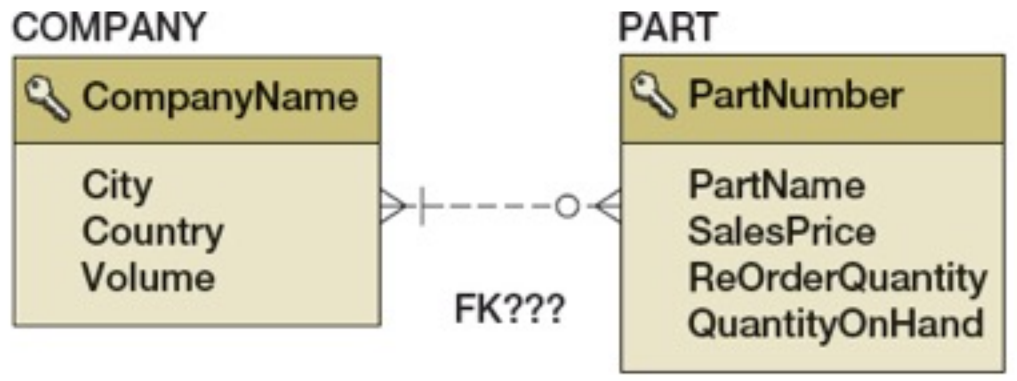
1:N Relationships

CREATE RELATIONSHIPS: STRONG ENTITY RELATIONSHIPS

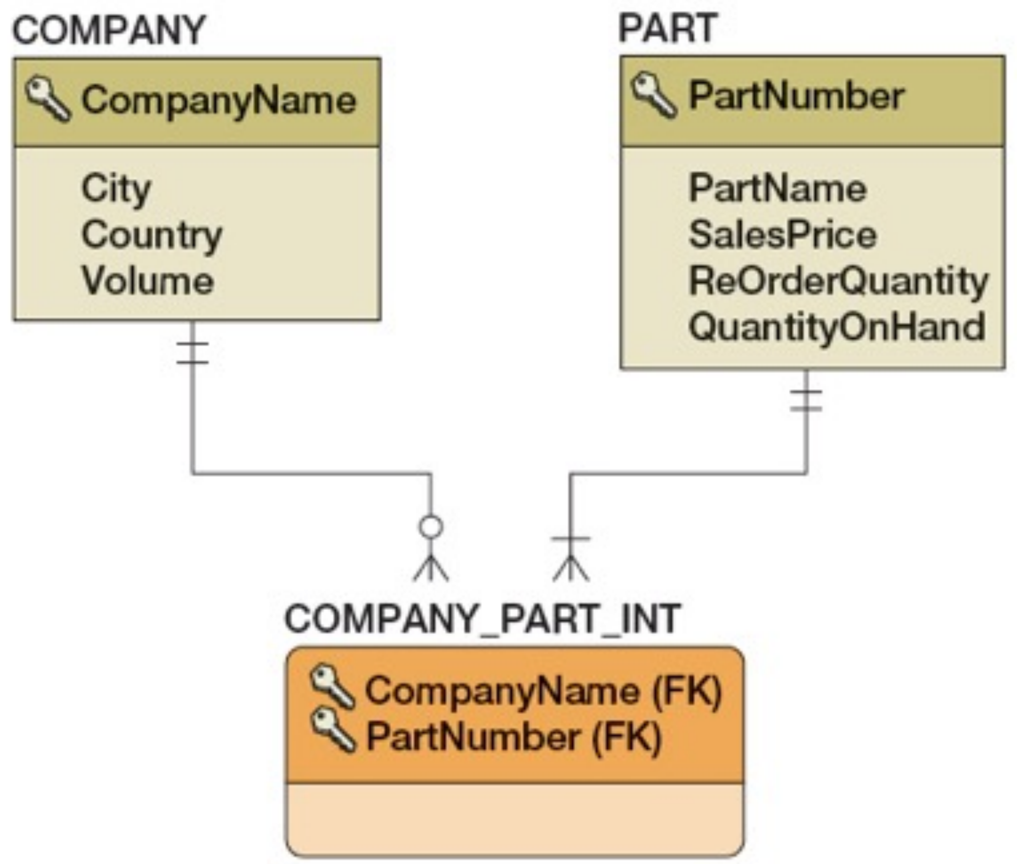
- An intersection table holds only the primary keys of the two tables as foreign keys.
- The intersection holds only the key data; it contains no other user data.

COMPANY

CompanyName	City	PartNumber
Samsung	Suwon	LCD001
LG	Seoul	LCD001
Samsung	Suwon	MEMORY001
Samsung	Suwon	CPU001



(a) The Foreign Key Has No Place in Either Table

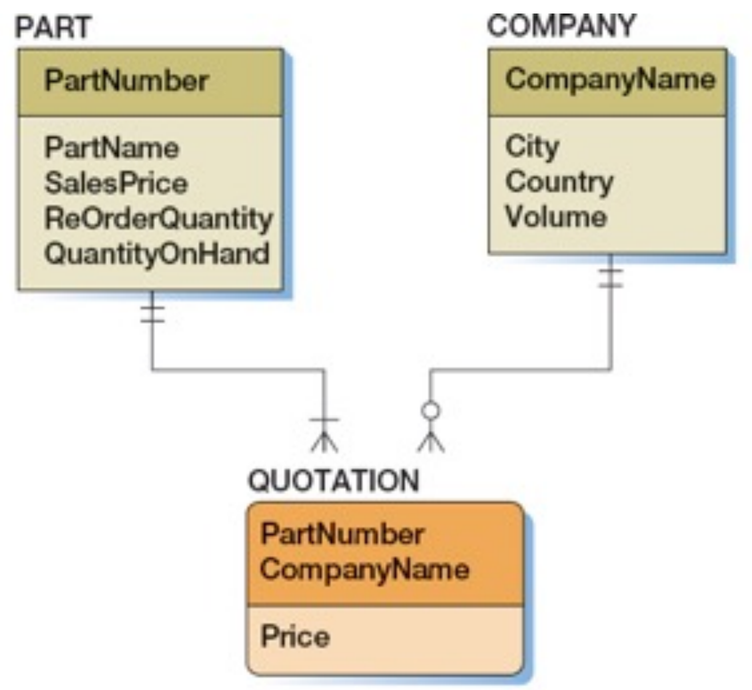


(b) Foreign Keys Placed in ID-Dependent Intersection Table
Copyright © 2016, by Pearson Education, Inc.,

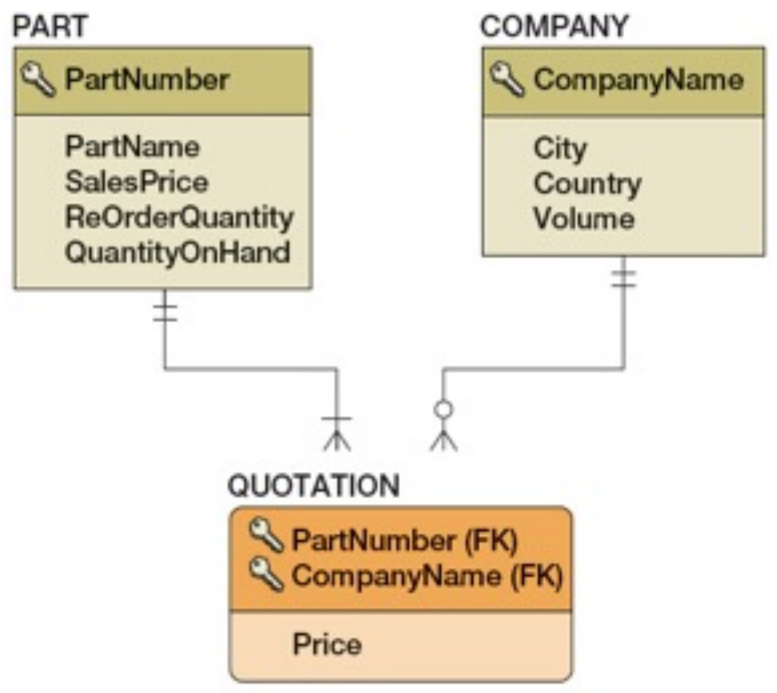
N:M Relationships

RELATIONSHIPS USING ID-DEPENDENT ENTITIES

- The difference between the intersection table and association table is the presence of attributes (user data).
- The need for intersection tables never appears in the users' world.

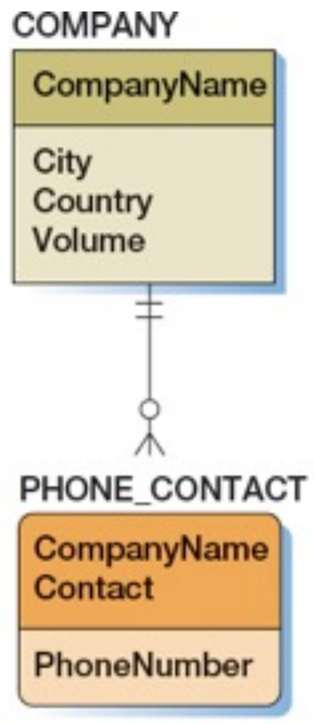


(a) Association Pattern Data Model from Figure 5-22

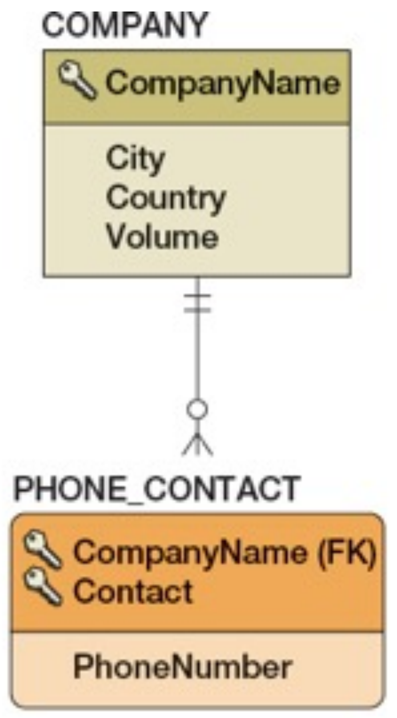


(b) Association Pattern Database Design

Copyright © 2016, by Pearson Education, Inc.,



(a) Data Model with Multivalued Attributes from Figure 5-29



(b) Database Design to Store Multivalued Attributes

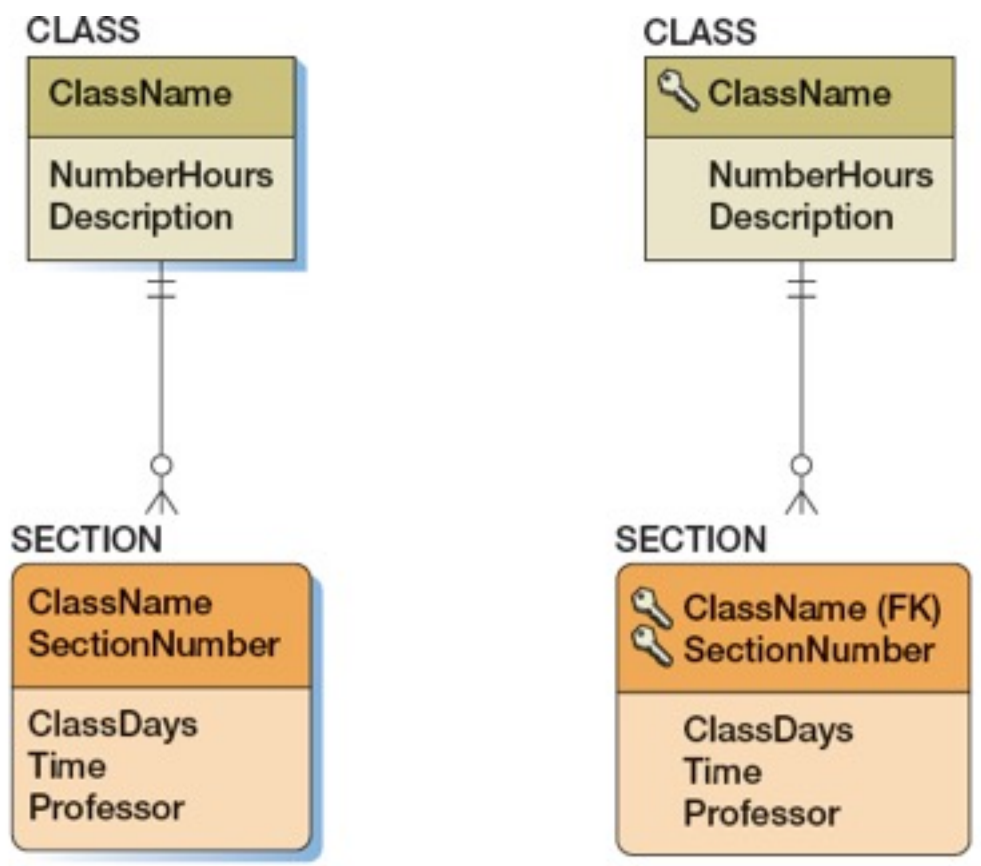
Copyright © 2016, by Pearson Education, Inc.,

Multivalued Attribute

Association Pattern

RELATIONSHIPS USING ID-DEPENDENT ENTITIES

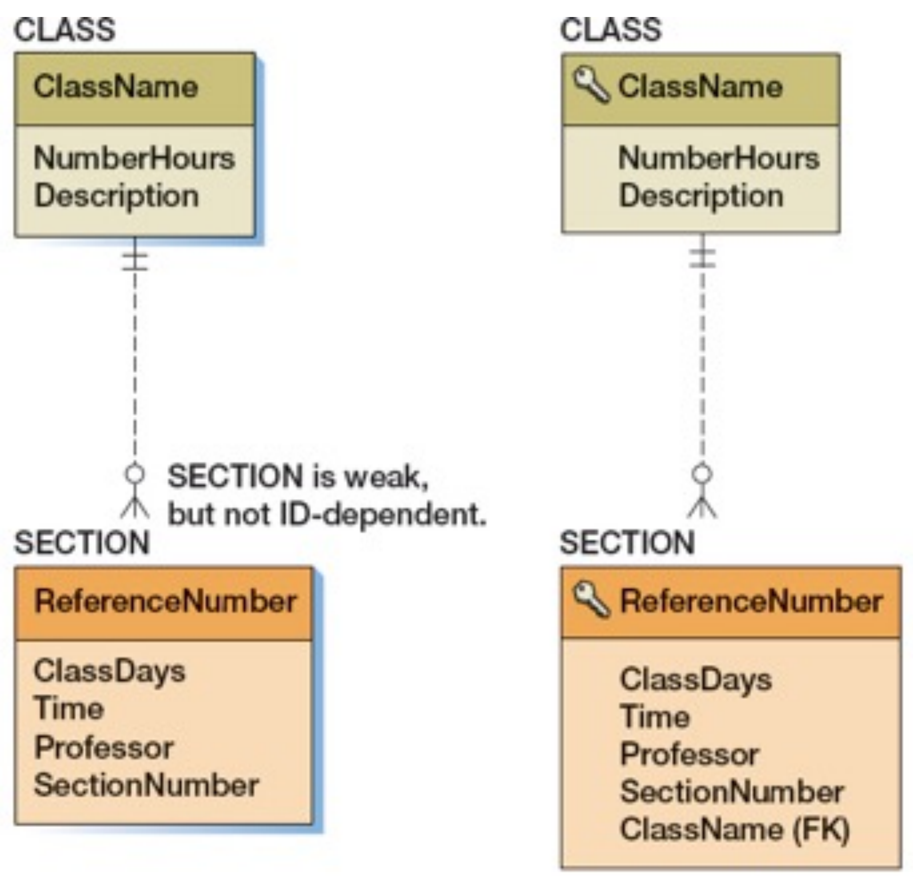
- Place the primary key of the parent in the child as a foreign key.
- Weak but non-ID-dependent entity : The relationship must be transformed using the rules of a 1:N relationship between a strong entity.



(a) Data Model with Archetype/Instance Pattern from Figure 5-30

(b) Database Design for Archetype/Instance Pattern

Archetype/Instance Pattern



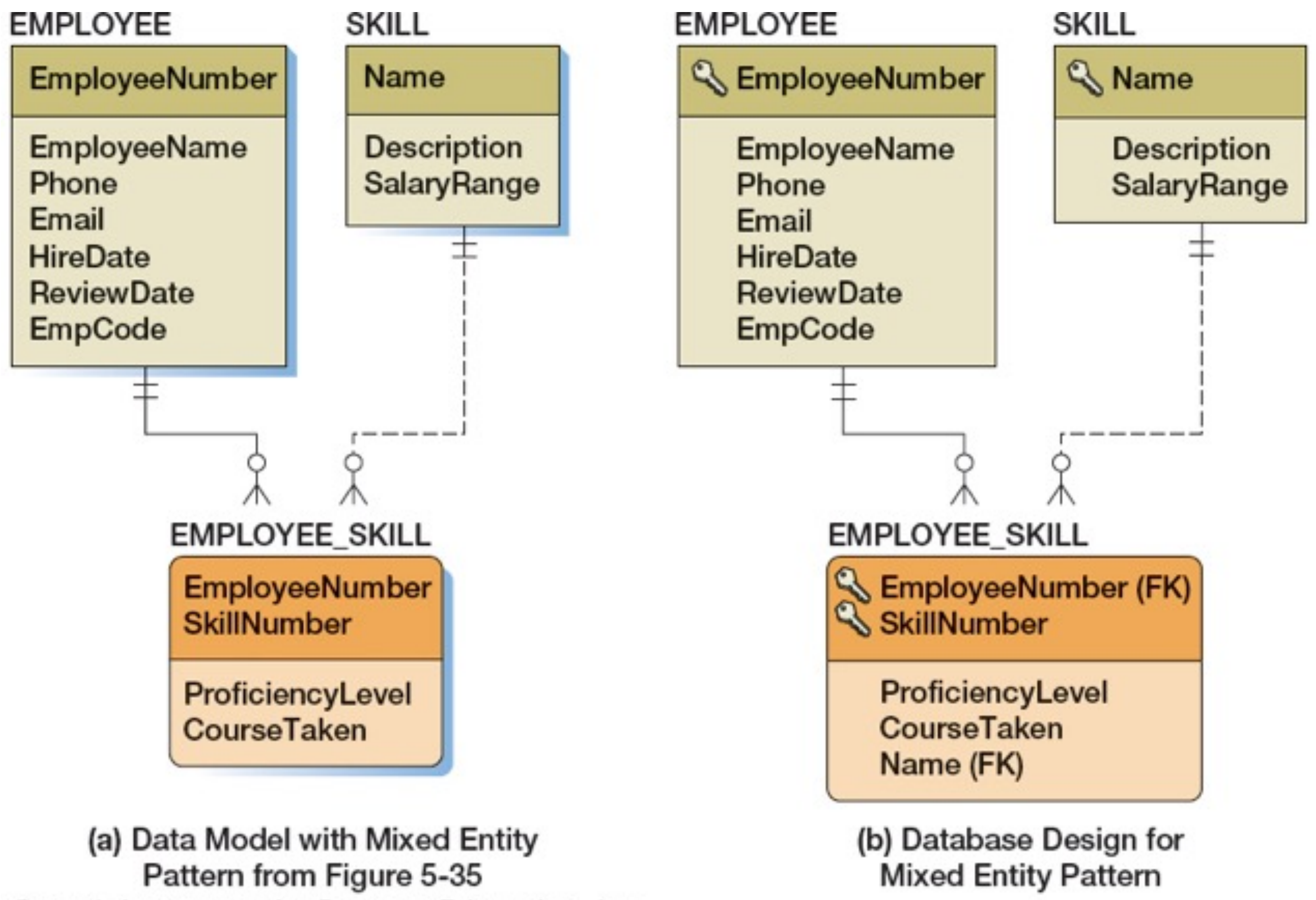
(a) Data Model with Non-ID-Dependent Weak Entity from Figure 5-31

(b) Database Design for Non-ID-Dependent Weak Entity

Archetype/Instance Pattern - Weak, Non-ID-Depenenty Entity

RELATIONSHIPS USING ID-DEPENDENT ENTITIES

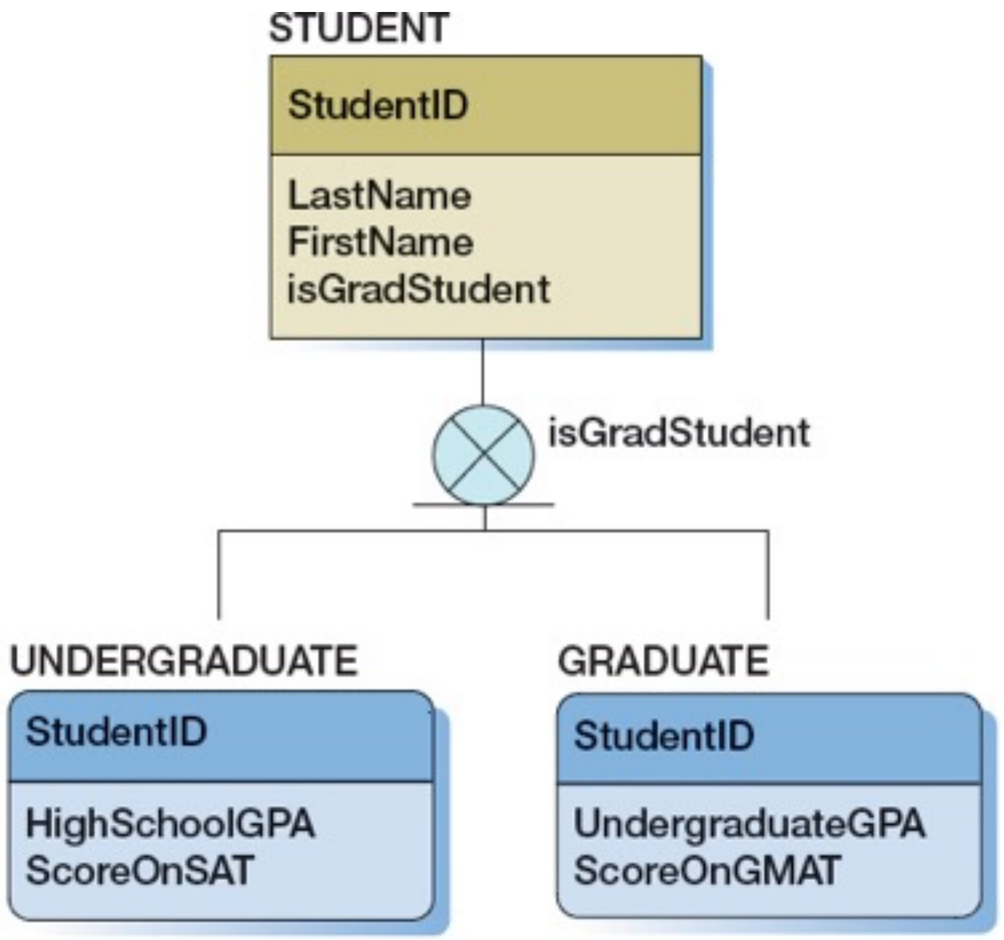
- The design transformation process for all HAS-A relationships can be summarized by the phrase, "Place the primary key of the parent in the child as a foreign key."



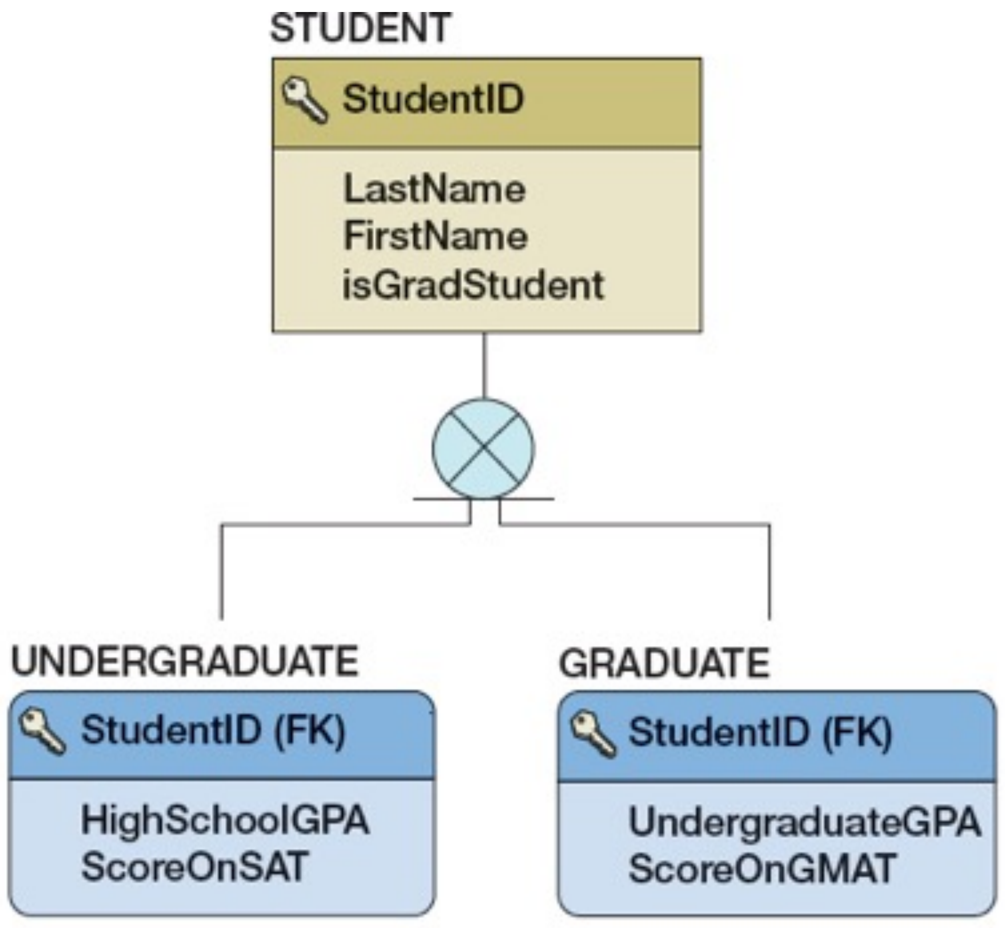
Mixed Entity

SUBTYPE RELATIONSHIPS

- A subtype and its supertype are representations of the same underlying entity. Because of this equivalence, the keys of all subtype tables are identical to the key of the supertype table.



(a) Data Model of the Supertype/Subtype Relationship from Figure 5-20(a)



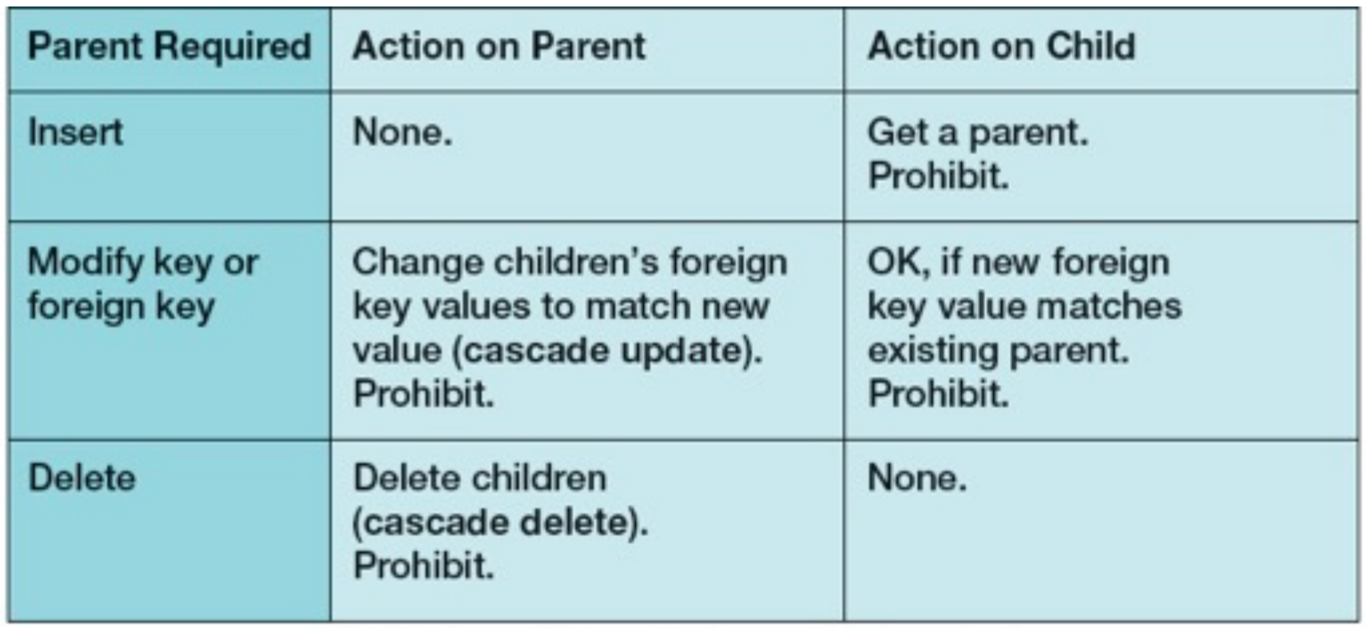
(b) Database Design for the Supertype/Subtype Relationship

DESIGN FOR MINIMUM CARDINALITY

- Relationships can have the following types of minimum cardinality:
    - O-O: parent optional and child optional
    - **M-O: parent mandatory and child optional**
    - O-M: parent optional and child mandatory
    - M-M: parent mandatory and child mandatory
- We will use the term **action** to mean a minimum cardinality enforcement action.
- No action needs to be taken for O-O relationships.

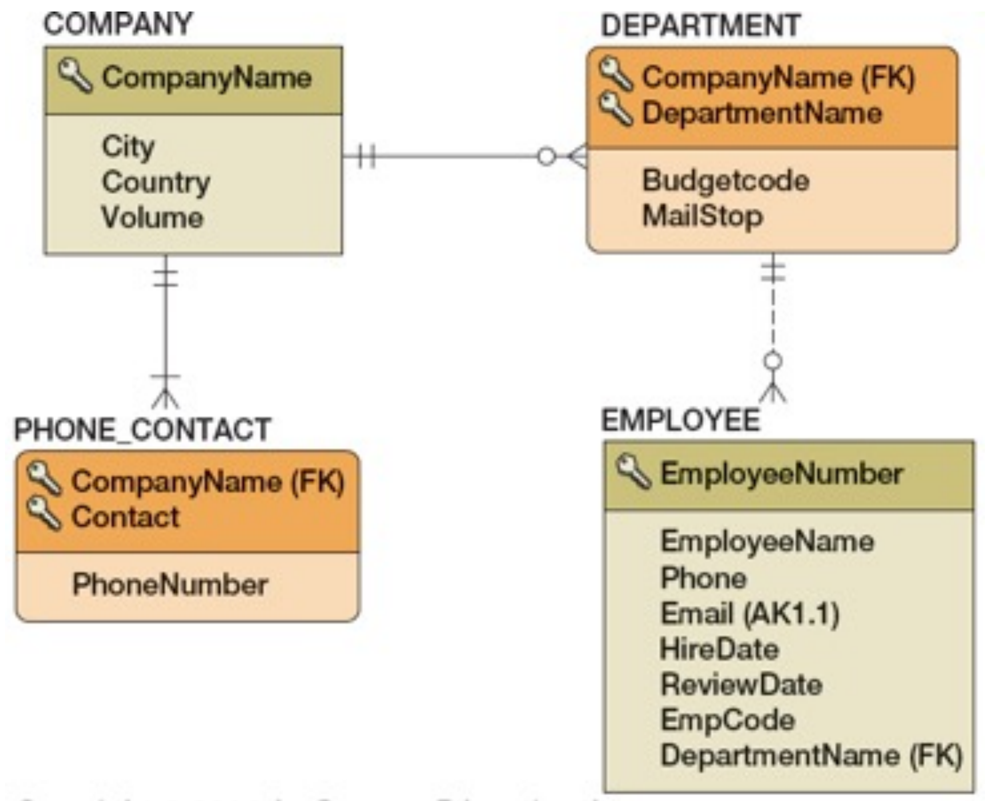
ACTIONS WHEN THE PARENT IS REQUIRED: ACTION ON PARENT

- When a new parent is created, nothing needs to be done. No child row can yet be dependent upon the new row.
- If the primary key of the parent changes, then any existing children will become orphans.
- To prevent the creation of orphans, either the foreign key values must be changed to match the new value of the parent's primary key or the modification to the parent's primary key must be prohibited.



Parent Required	Action on Parent	Action on Child
Insert	None.	Get a parent. Prohibit.
Modify key or foreign key	Change children's foreign key values to match new value (cascade update). Prohibit.	OK, if new foreign key value matches existing parent. Prohibit.
Delete	Delete children (cascade delete). Prohibit.	None.

(a) Actions When the Parent Is Required



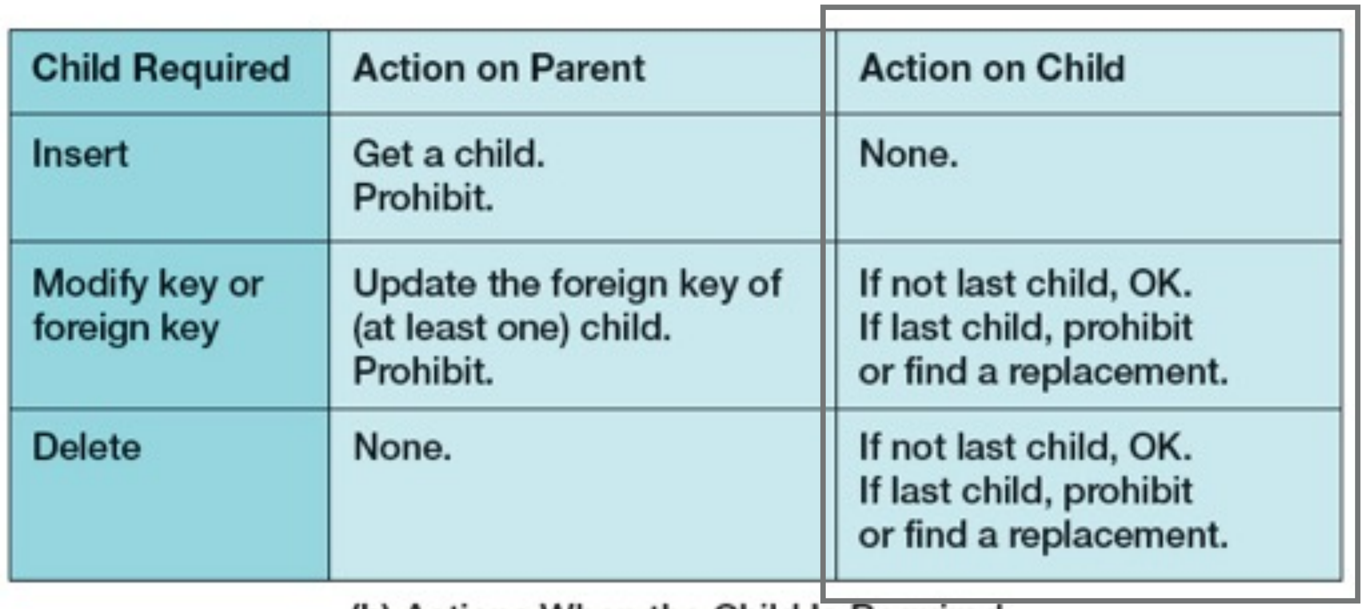
ACTIONS WHEN THE PARENT IS REQUIRED: ACTION ON CHILD

- The policy of propagating a change from the parent's primary key to the children's foreign key is called **cascading updates**.
- If that row has children, and if the deletion is allowed, then the children will become orphans.
- When such a delete attempt is made, either the children must be deleted as well or the deletion must be prohibited.
- Deleting the children along with the parent is called **cascading deletions**.

- If the parent is required, then when a new child row is created, the new row must have a valid foreign key value.
- With regards to modifications to the foreign key, the new value must match a value of the primary key in the parent.

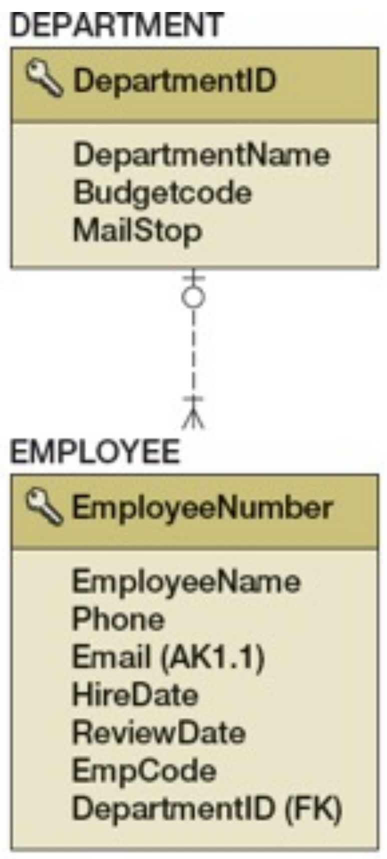
ACTIONS WHEN THE CHILD IS REQUIRED: ACTION ON PARENT

- If the child is required, then we cannot create a new parent without also creating a relationship to a child.
- If the child is required, then to modify the parent's primary key, either the key of at least one child must also be changed or the update must be disallowed.
- If the child is required and the parent is deleted, no action need be taken.



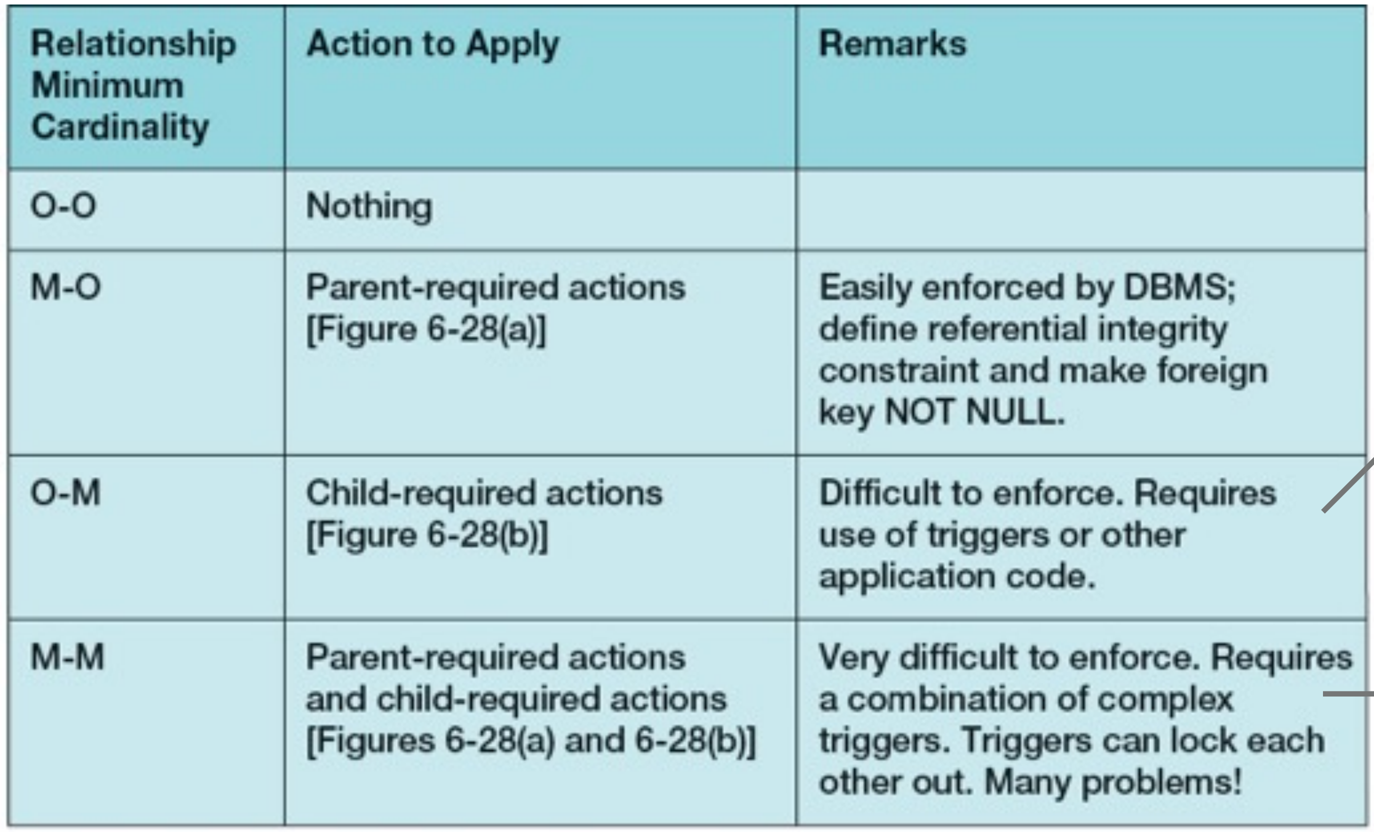
Child Required	Action on Parent	Action on Child
Insert	Get a child. Prohibit.	None.
Modify key or foreign key	Update the foreign key of (at least one) child. Prohibit.	If not last child, OK. If last child, prohibit or find a replacement.
Delete	None.	If not last child, OK. If last child, prohibit or find a replacement.

(b) Actions When the Child Is Required



SUMMARY OF MINIMUM CARDINALITY DESIGN

- Fortunately, The actions of M-O relationship are easy to enforce using facilities available in most DBMS products.
- It turns out that we can enforce these actions with just two limitations.
- First, we need to define a referential integrity constraint that ensures that every foreign key value has a match in the parent table.
- Second, we make the foreign key column NOT NULL.



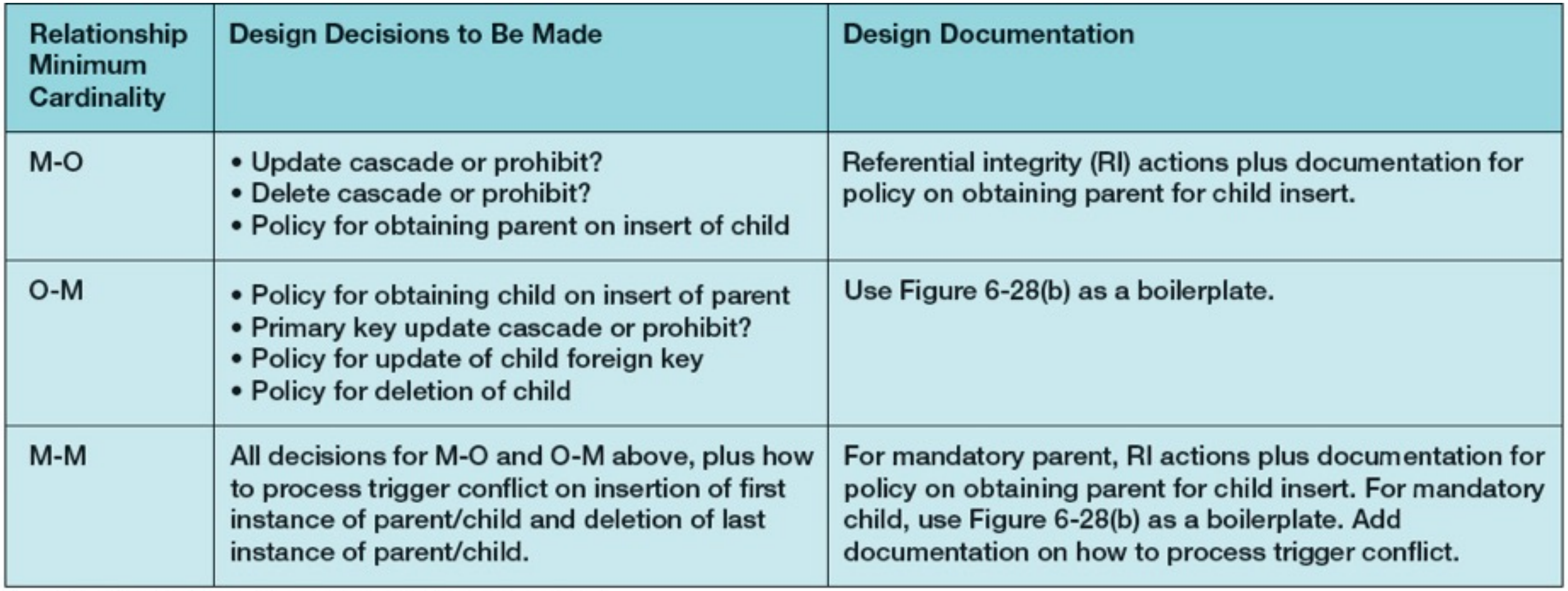
Relationship Minimum Cardinality	Action to Apply	Remarks
O-O	Nothing	
M-O	Parent-required actions [Figure 6-28(a)]	Easily enforced by DBMS; define referential integrity constraint and make foreign key NOT NULL.
O-M	Child-required actions [Figure 6-28(b)]	Difficult to enforce. Requires use of triggers or other application code.
M-M	Parent-required actions and child-required actions [Figures 6-28(a) and 6-28(b)]	Very difficult to enforce. Requires a combination of complex triggers. Triggers can lock each other out. Many problems!

If the row is the last child, then the trigger must do one of the following:
- Delete the parent.
- Find a substitute child.
- Disallow the update or delete.

The best advice is to avoid M-M relationships if you can.

DOCUMENTING THE MINIMUM CARDINALITY DESIGN

- Because enforcing minimum cardinality can be complicated, and because it often involves the creation of triggers or other procedures, clear documentation is essential.



Relationship Minimum Cardinality	Design Decisions to Be Made	Design Documentation
M-O	• Update cascade or prohibit? • Delete cascade or prohibit? • Policy for obtaining parent on insert of child	Referential integrity (RI) actions plus documentation for policy on obtaining parent for child insert.
O-M	• Policy for obtaining child on insert of parent • Primary key update cascade or prohibit? • Policy for update of child foreign key • Policy for deletion of child	Use Figure 6-28(b) as a boilerplate.
M-M	All decisions for M-O and O-M above, plus how to process trigger conflict on insertion of first instance of parent/child and deletion of last instance of parent/child.	For mandatory parent, RI actions plus documentation for policy on obtaining parent for child insert. For mandatory child, use Figure 6-28(b) as a boilerplate. Add documentation on how to process trigger conflict.