David M. Kroenke and David J. Auer

Database Processing:

Fundamentals, Design, and Implementation

Chapter Six:

Transforming Data Models into Database Designs

Chapter 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 understand the use of referential integrity actions
- To be able to represent ID-dependent, 1:1, 1:N, and N:M relationships as tables
- To be able to represent weak entities as tables

Chapter Objectives

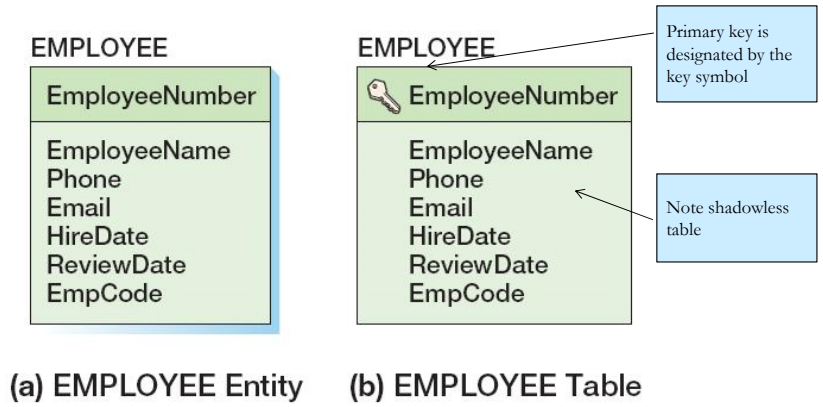
- To be able to represent supertype/subtypes as tables
- To be able to represent recursive relationships as tables
- To be to represent ternary relationships as tables
- To be able to implement referential integrity actions required by minimum cardinalities

Steps for Transforming a Data Model into a Database Design

1. Create a table for each entity:
   - Specify primary key (consider surrogate keys, as appropriate)
   - Specify candidate keys
   - Specify properties for each column:
     - Null status
     - Data type
     - Default value (if any)
     - Specify 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:
   - M-O relationships
   - O-M relationships
   - M-M relationships

Create a Table for Each Entity

EMPLOYEE (EmployeeNumber, EmployeeName, Phone, Email, HireDate, ReviewDate, EmpCode)



(a) EMPLOYEE Entity (b) EMPLOYEE Table

Entities and Tables

- The principle difference between an entity and a table (relation) is that you can express a relationship between entities without using foreign keys.
- This makes it easier to work with entities in the early design process where the very existence of entities and the relationships between them is uncertain.

Select the Primary Key

- **The ideal primary key is short, numeric, and fixed.**
- **Surrogate keys meet the ideal, but have no meaning to users.**

Specify Candidate (Alternate) Keys

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

Specify Candidate (Alternate) Keys

EMPLOYEE

EmployeeNumber
EmployeeName Phone Email (AK1.1) HireDate ReviewDate EmpCode

CUSTOMER

CustomerNumber
Name (AK1.1) City (AK1.2) Phone Email (AK2.1)

Specify Column Properties: Null Status

- **Null status** indicates whether or not the value of the column can be NULL.

EMPLOYEE

EmployeeNumber: NOT NULL
EmployeeName: NOT NULL Phone: NULL Email: NULL (AK1.1) HireDate: NOT NULL ReviewDate: NULL EmpCode: NULL

Specify Column Properties: Data Type

- **Generic data types:**
  - **CHAR(n)**
  - **VARCHAR(n)**
  - **DATE**
  - **TIME**
  - **MONEY**
  - **INTEGER**
  - **DECIMAL**

EMPLOYEE

EmployeeNumber: int
EmployeeName: char(50) Phone: char(15) Email: char(50) (AK1.1) HireDate: datetime ReviewDate: datetime EmpCode: char(18)

Specify Column Properties: SQL Server 2008 Data Types

Data Type	Description
Binary	Binary, length 0 to 8,000 bytes.
Char	Character, length 0 to 8,000 bytes.
Datetime	8-byte datetime. Range from January 1, 1753, through December 31, 9999, with an accuracy of three-hundredths of a second.
Image	Variable length binary data. Maximum length 2,147,483,647 bytes.
Integer	4-byte integer. Value range from –2,147,483,648 through 2,147,483,647.
Money	8-byte money. Range from –922,337,203,685,477.5808 through +922,337,203,685,477.5807, with accuracy to a ten-thousandth of a monetary unit.
Numeric	Decimal – can set precision and scale. Range –10^38 +1 through 10^38 –1.
Smalldatetime	4-byte datetime. Range from January 1, 1900, through June 6, 2079, with an accuracy of one minute.
Smallint	2-byte integer. Range from –32,768 through 32,767.
Smallmoney	4-byte money. Range from 214,748.3648 through +214,748.3647, with accuracy to a ten-thousandth of a monetary unit.
Text	Variable length text, maximum length 2,147,483,647 characters.
Tinyint	1-byte integer. Range from 0 through 255.
Varchar	Variable-length character, length 0 to 8,000 bytes.

Specify Column Properties: Oracle Database 11g Data Types

Data Type	Description
BLOB	Binary large object. Up to 4 gigabytes in length.
CHAR(n)	Fixed length character field of length *n*. Maximum 2,000 characters.
DATE	7-byte field containing both date and time.
INTEGER	Whole number of length 38.
NUMBER(n,d)	Numeric field of length *n*, *d* places to the right of the decimal.
VARCHAR(n) or VARCHAR2(n)	Variable length character field up to *n* characters long. Maximum value of *n* = 4,000.

Specify Column Properties: MySQL 5.1Data Types I

NumericData Type	Description
BIT (M)	M = 1 to 64
TINYINT	–128 to 127
TINYINT UNSIGNED	0 to 255
BOOLEAN	0 = FALSE; 1 = TRUE
SMALLINT	–32,768 to 32,767
SMALLINT UNSIGNED	0 to 65535
MEDIUMINT	–8,388,608 to 8,388,607
MEDIUMINT UNSIGNED	0 to 16,777,215
INT or INTEGER	–2,147,483,648 to 2,147,483,647
INT UNSIGNED or INTEGER UNSIGNED	0 to 4,294,967,295
BIGINT	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
BIGINT UNSIGNED	0 to 1,844,674,073,709,551,615
FLOAT (P)	P = Precision; 0 to 24
FLOAT (M, D)	Small (single-precision) floating-point number: M = Display width D = Number of significant digits
DOUBLE (M, B)	Normal (double-precision) floating-point number: M = Display width B = Precision; 25 to 53
DEC (M[,D]) or DECIMAL (M[,D]) or FIXED (M[,D])	Fixed-point number: M = Total number of digits D = Number of decimals
Date and Time Data Types	**Description**
DATE	YYYY-MM-DD : 1000-01-01 to 9999-12-31
DATETIME	YYYY-MM-DD HH:MM:SS
	1000-01-01 00:00:00 to 9999-12-31 23:59:59
TIMESTAMP	See documentation.
TIME	HH:MM:SS—00:00:00 to 23:59:59
YEAR (M)	M = 2 or 4 (default)
	IF 2 = 1970 to 2069 (70 to 60)
	IF 4 = 1901 to 2155

Specify Column Properties: MySQL 5.1Data Types II

String Data Types	Description
CHAR (M)	M = 0 to 255
VARCHAR (M)	M = 1 to 255
BLOB (M)	BLOB = Binary Large Object; maximum 65,535 characters
TEXT (M)	Maximum 65,535 characters
TINYBLOB MEDIUMBLOB LONGBLOB TINYTEXT MEDIUMTEXT LONGTEXT	See documentation.
ENUM ('value1', 'value2', . . .)	An enumeration. Only one value, but chosen from list. See documentation.
SET ('value1', 'value2', . . .)	A set. Zero or more values, all chosen from list. See documentation.

Specify Column Properties: Default Value

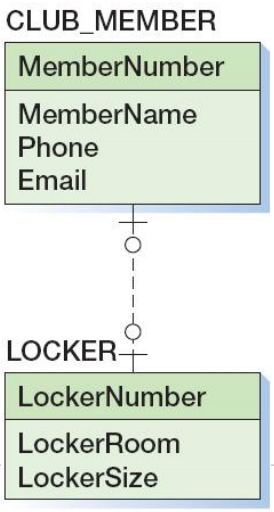
- A **default value** is the value supplied by the DBMS when a new row is created.

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].

Create Relationships: 1:1 Strong Entity Relationships

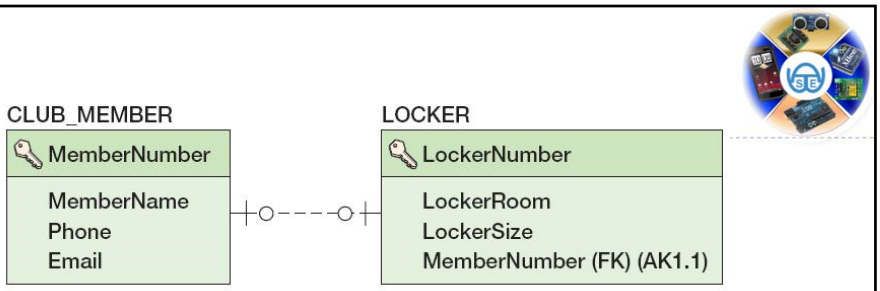


(a) Club Membership Data Entry Form

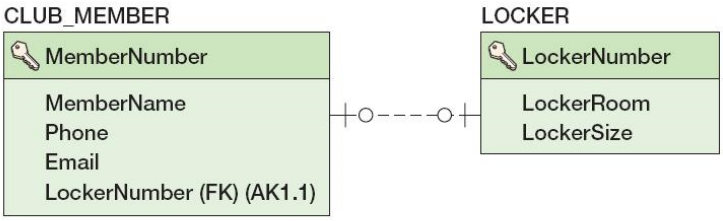
(b) Club Locker Report

CLUB_LOCKERS

LockerRoom	LockerNumber	MemberNumber	MemberName	LockerSize
Mens	2100	1000	Jones	Med
Mens	2115	3000	Wu	Large
Womens	2200	2000	Abernathy	Large
Womens	2217	4000	Lai	Small



(a) With Foreign Key in LOCKER

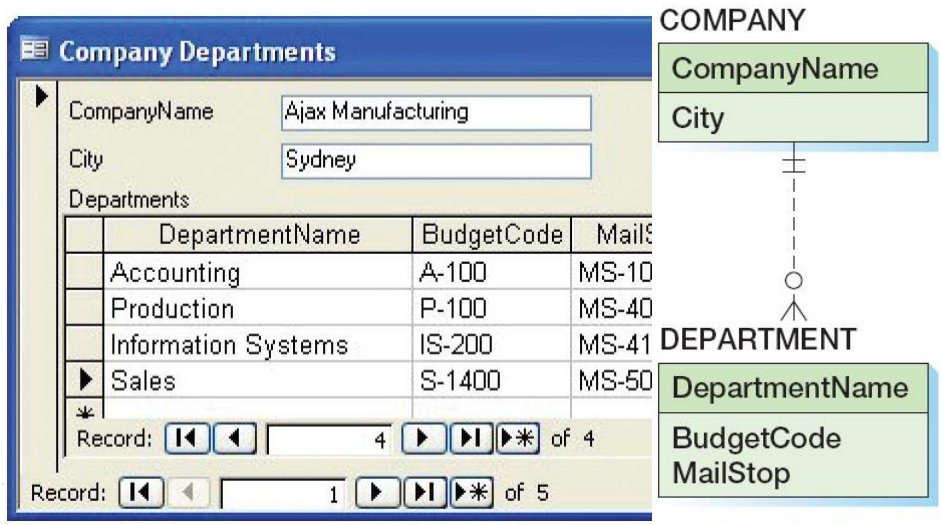


(b) With Foreign Key in CLUB_MEMBER

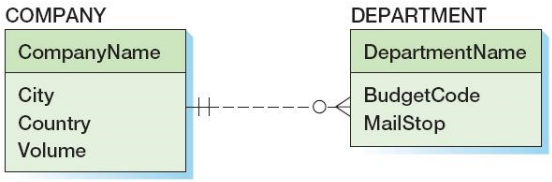
Create Relationships: 1:1 Strong Entity Relationships

- Place the key of one entity in the other entity as a foreign key.
  - Either design will work—no parent, no child.
  - Minimum cardinality considerations may be important.
    - O-M will require a different design than M-O.
    - One design will be very preferable.

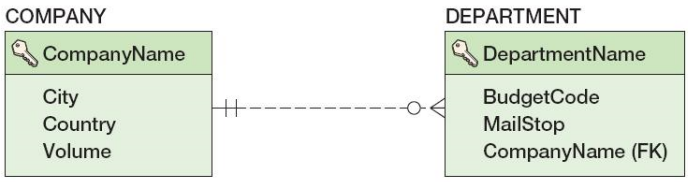
Create Relationships: 1:N Strong Entity Relationships



Create Relationships: 1:N Strong Entity Relationships



(a) 1:N Relationship Between Strong Entities



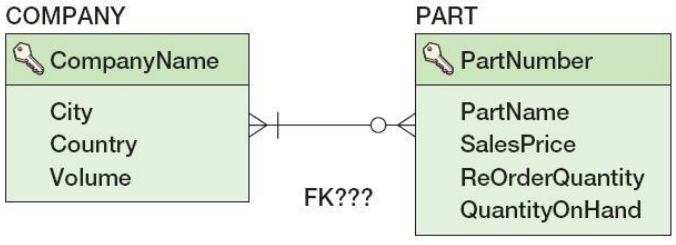
(b) Placing the Primary Key of the Parent in the Child as a Foreign Key

Create Relationships: 1:N Strong Entity Relationships

- Place the primary key of the table on the one side of the relationship into the table on the many side of the relationship as the foreign key.
- The *one* side is the parent table and the *many* side is the child table, so "place the key of the parent in the child."

Create Relationships: N:M Strong Entity Relationships

- In an N:M strong entity relationship there is no place for the foreign key in either table.
  - A COMPANY may supply many PARTs.
  - A PART may be supplied by many COMPANYs.



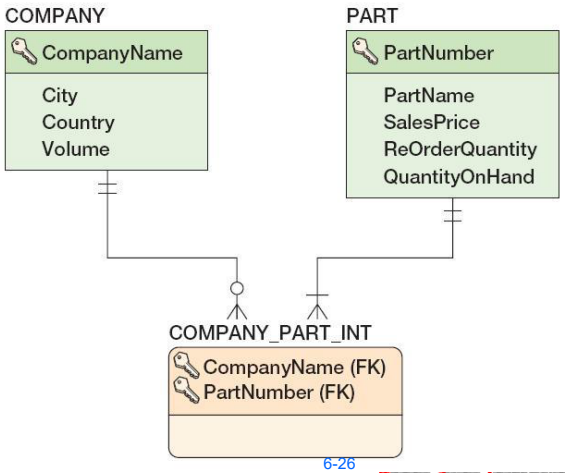
Create Relationships: N:M Strong Entity Relationships

- The solution is to create an intersection table that stores data about the corresponding rows from each entity.
- The intersection table consists only of the primary keys of each table which form a composite primary key.
- Each table's primary key becomes a foreign key linking back to that table.

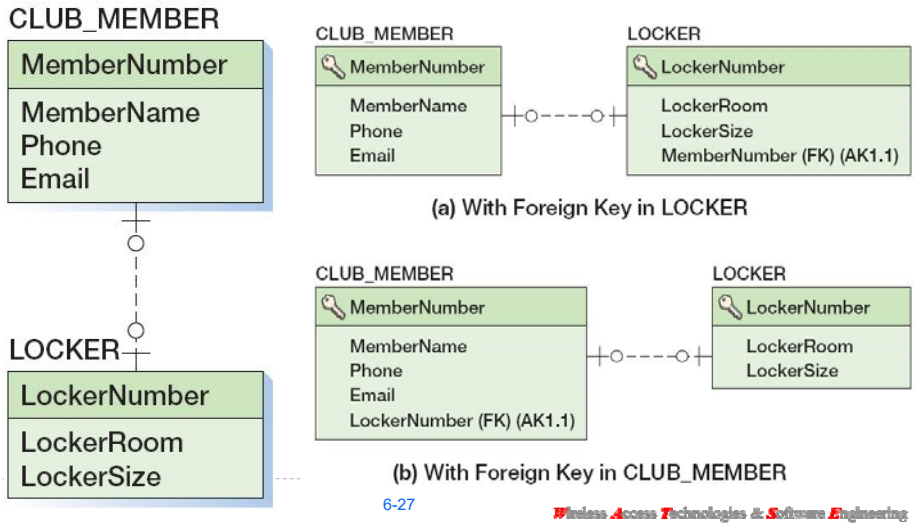
COMPANY_PART_INT (*CompanyName*, *PartNumber*)

Create Relationships: N:M Strong Entity Relationships

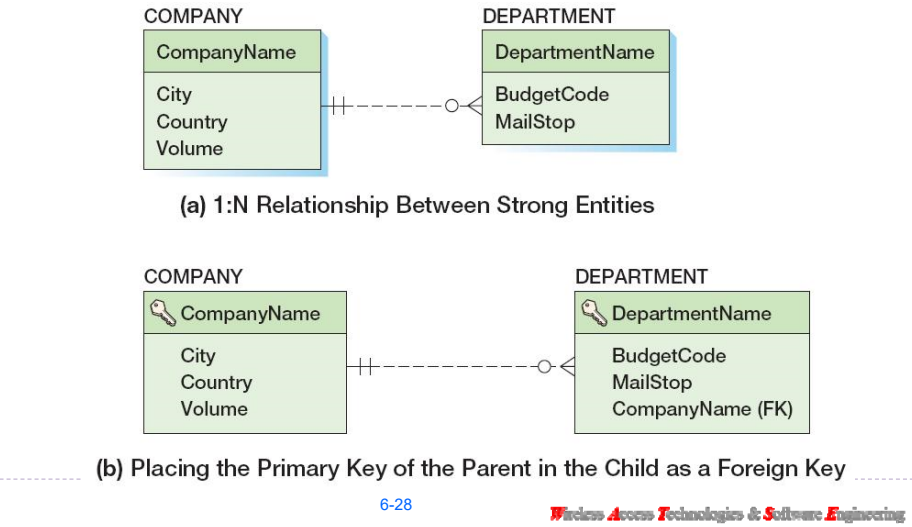
COMPANY_PART_INT (*CompanyName*, *PartNumber*)



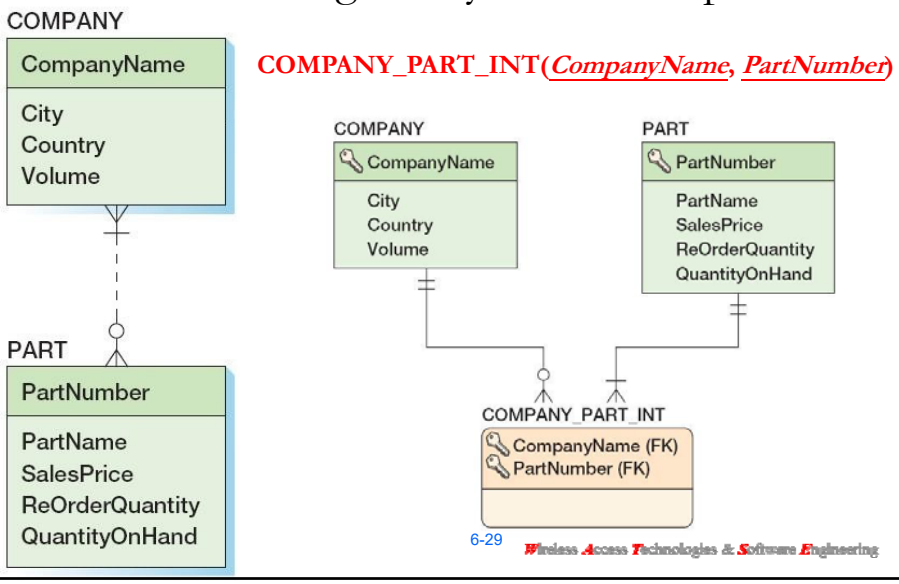
Create Relationships: 1:1 Strong Entity Relationships



Create Relationships: 1:N Strong Entity Relationships



Create Relationships: N:M Strong Entity Relationships



Relationships Using ID-Dependent Entities: Four Uses for ID-Dependent Entities

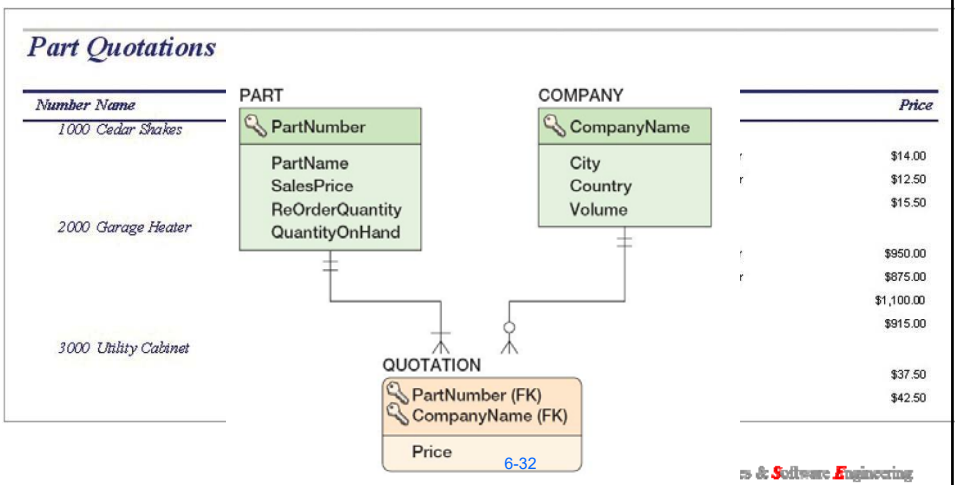
- Representing N:M Relationships
  - We just discussed this
- Association Relationships
- Multivalued Attributes
- Archetype/Instance Relationships

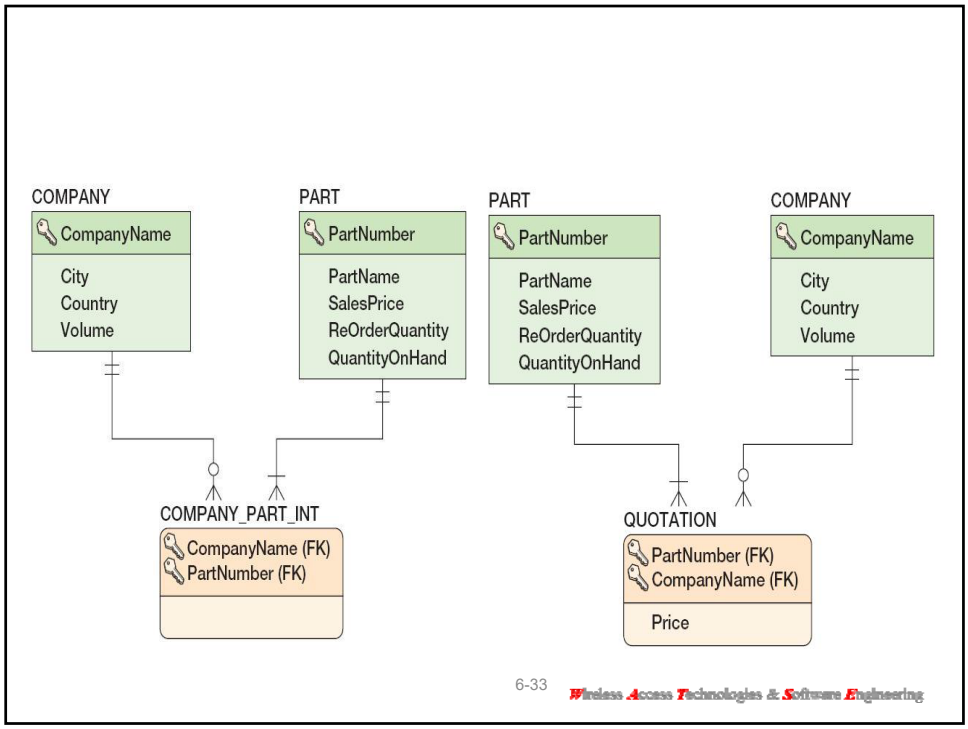
Relationships Using ID-Dependent Entities: Association Relationships

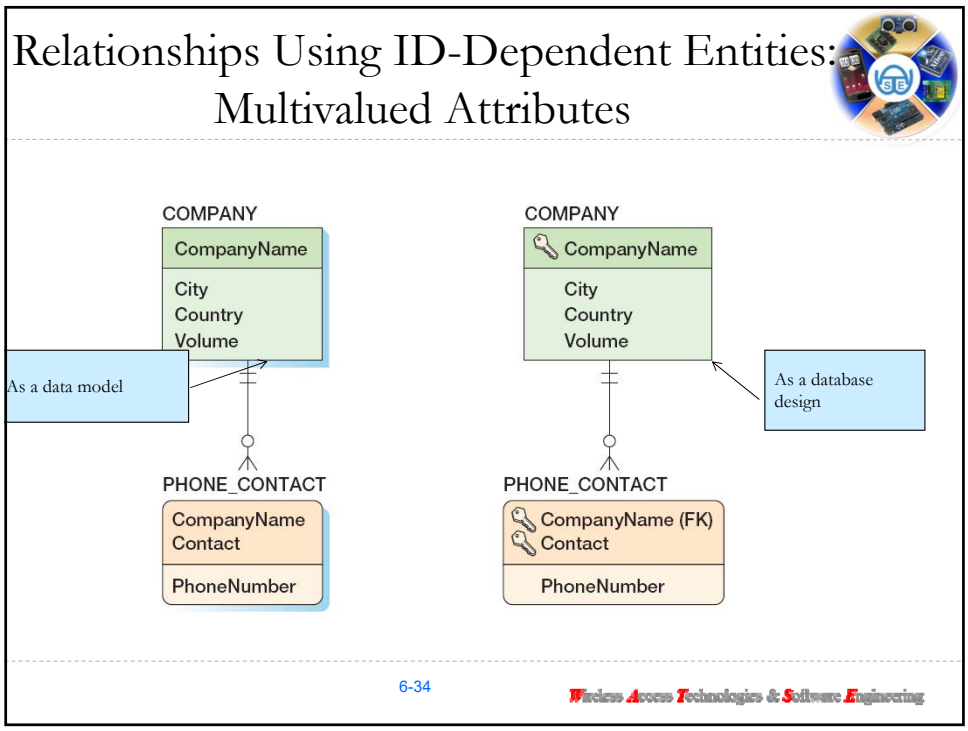
- An intersection table:
  - Holds the relationships between two strong entities in an N:M relationship
  - Contains *only* the primary keys of the two entities:
    - As a composite primary key
    - As foreign keys
- An association table
  - Has all the characteristics of an intersection table
  - PLUS it has one or more columns of attributes specific to the associations of the other two entities

Relationships Using ID-Dependent Entities: Association Relationships

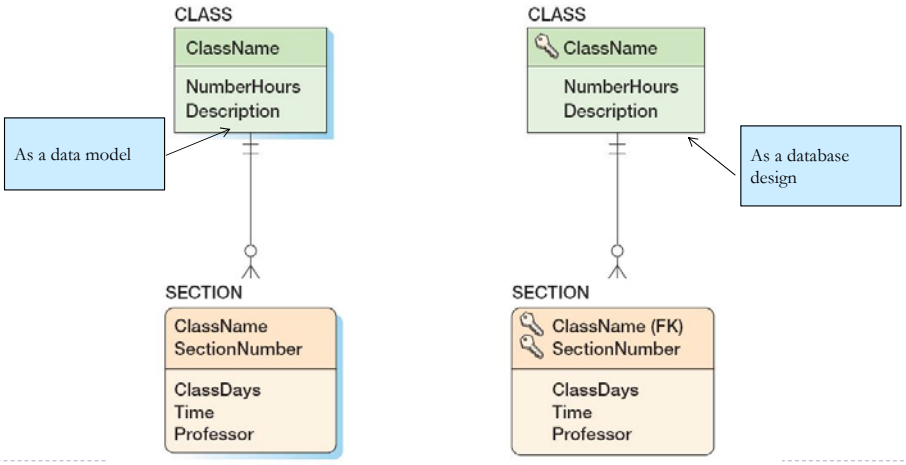
QUOTATION (*CompanyName*, *PartNumber*, Price)



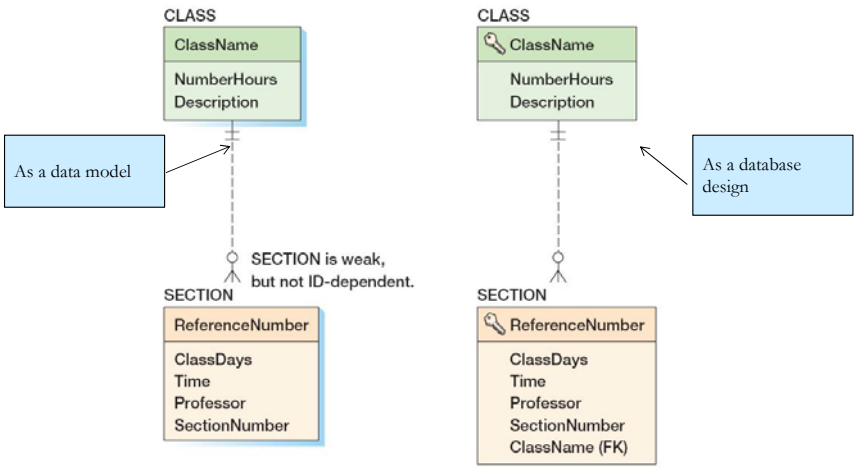

COMPANY
CompanyName
City
Country
Volume
PART
PartNumber
PartName
SalesPrice
ReOrderQuantity
QuantityOnHand
PART
PartNumber
PartName
SalesPrice
ReOrderQuantity
QuantityOnHand
COMPANY
CompanyName
City
Country
Volume
COMPANY_PART_INT
CompanyName (FK)
PartNumber (FK)
QUOTATION
PartNumber (FK)
CompanyName (FK)
Price
6-33
Wireless Access Technologies & Software Engineering


Relationships Using ID-Dependent Entities: Multivalued Attributes
COMPANY
CompanyName
City
Country
Volume
As a data model
PHONE_CONTACT
CompanyName
Contact
PhoneNumber
COMPANY
CompanyName
City
Country
Volume
As a database design
PHONE_CONTACT
CompanyName (FK)
Contact
PhoneNumber
6-34
Wireless Access Technologies & Software Engineering

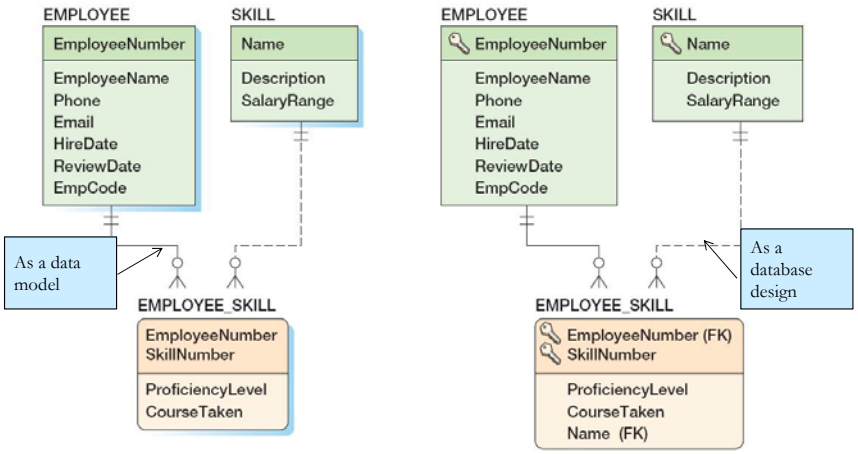
Relationships Using ID-Dependent Entities: Archetype/Instance Pattern



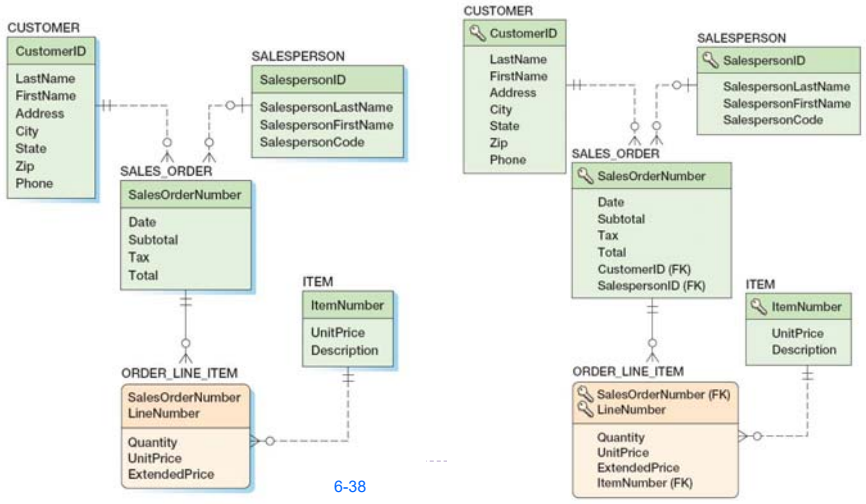
Relationships Using Weak Entities: Archetype/Instance Pattern



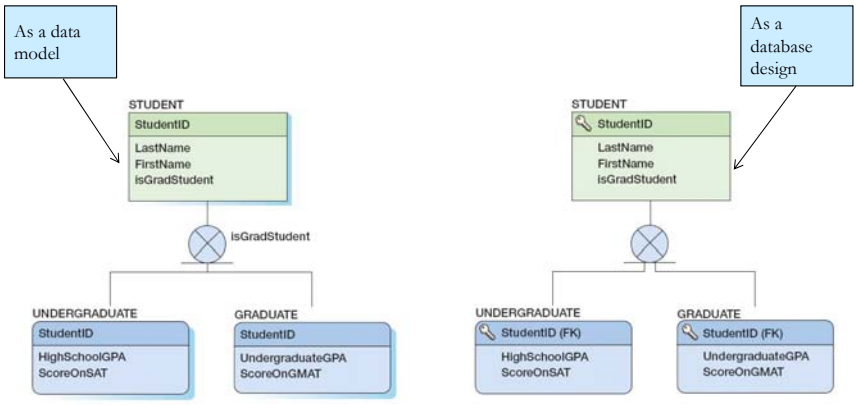
Mixed Entity Relationships



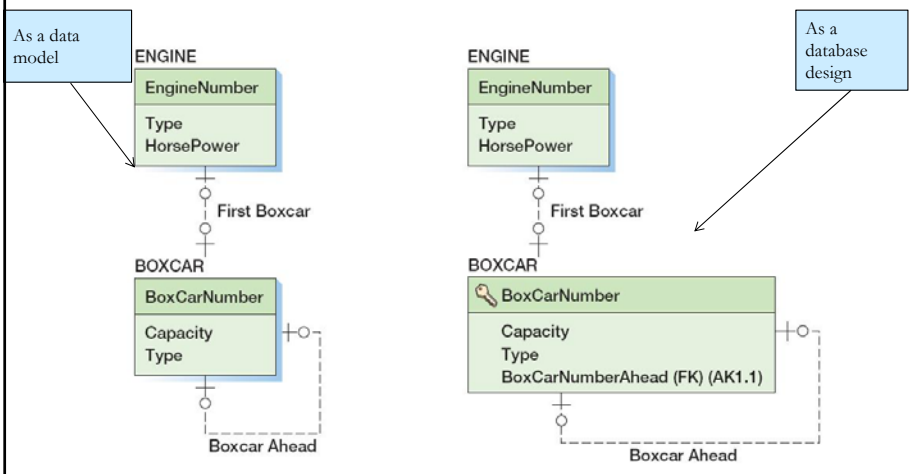
Mixed Entity Relationships: The SALES_ORDER Pattern



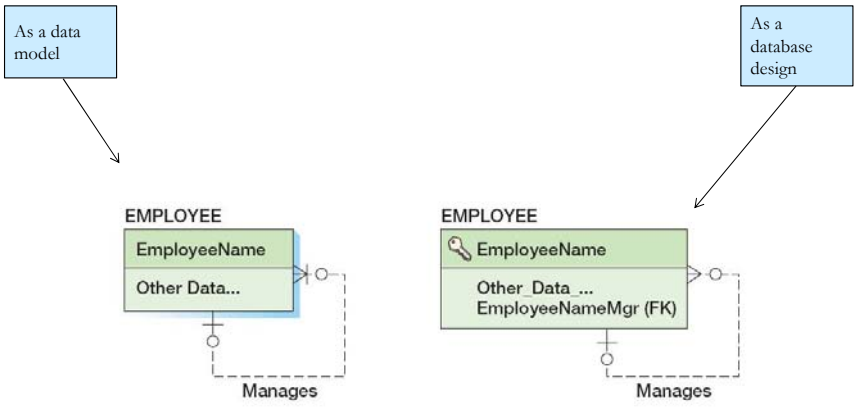
Subtype Relationships



Recursive Relationships: 1:1 Recursive Relationships



Recursive Relationships: 1:N Recursive Relationships



Recursive Relationships: N:M Recursive Relationships