

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 supertypes/subtypes as tables
- To be able to represent recursive relationships as tables
- To be able 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)

EMPLOYEE



(a) EMPLOYEE Entity

EMPLOYEE



(b) EMPLOYEE Table

Primary key is designated by the key symbol

Select the Primary Key

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

EMPLOYEE

 EmployeeNumber
EmployeeName Phone Email HireDate ReviewDate EmpCode

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 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)
 - VARCHAR2(n)
 - Number(p,s)
 - DATE
 - TIMESTAMP
 - MONEY
 - INTEGER
 - DECIMAL

EMPLOYEE



EmployeeNumber: int

EmployeeName: char(50)

Phone: char(15)

Email: char(50) (AK1.1)

HireDate: datetime

ReviewDate: datetime

EmpCode: char(18)

MySQL

- Page 257-258
 - 1. varchar(m): (1-255)
 - 2. text: (1-65535)
 - 3. blob: 1-65535
 - 4. longblob
 - 5. Boolean: 0/1
 - 6. int:
 - 7. decimal(m,d)
 - 8. date (yyyy-mm-dd)
 - 9. datetime (yyyy-mm-dd hh:mm:ss)
 - 10. time

Specify Column Properties: Data Type + Null Status

EMPLOYEE



EmployeeNumber: int NOT NULL

EmployeeName: char(50) NOT NULL

Phone: char(15) NULL

Email: char(50) NULL (AK1.1)

HireDate: datetime NOT NULL

ReviewDate: datetime NULL

EmpCode: char(18) NULL

Specify Column Properties:

SQL Server 2008 R2 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(<i>n</i>)	Fixed length character field of length <i>n</i> . Maximum 2,000 characters.
DATE	7-byte field containing both date and time.
INTEGER	Whole number of length 38.
NUMBER(<i>n</i> , <i>d</i>)	Numeric field of length <i>n</i> , <i>d</i> places to the right of the decimal.
VARCHAR(<i>n</i>) or VARCHAR2(<i>n</i>)	Variable length character field up to <i>n</i> characters long. Maximum value of <i>n</i> = 4,000.

Specify Column Properties:

MySQL 5.5 Data Types I

Numeric Data 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 5.5 Data 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).

Verify Normalization

- The tables should be normalized based on the data model.
- Verify that all tables are:
 - BCNF
 - 4NF

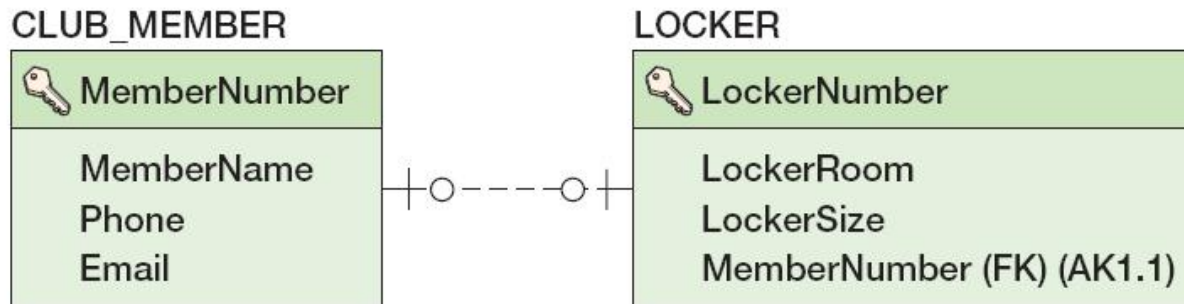
Create Relationships:

1:1 Strong Entity Relationships

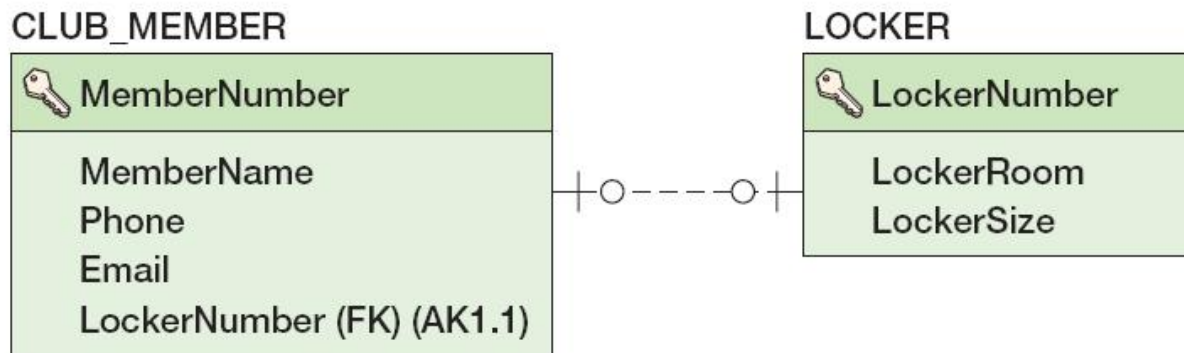
- Place the key of one entity in the other entity as a foreign key.
 - Either design will work—no parent, no child.
 - Minimum cardinality considerations may be important.
 - O-M will require a different design than M-O.
 - One design will be very preferable.

Create Relationships:

1:1 Strong Entity Relationships



(a) With Foreign Key in LOCKER



(b) With Foreign Key in CLUB_MEMBER

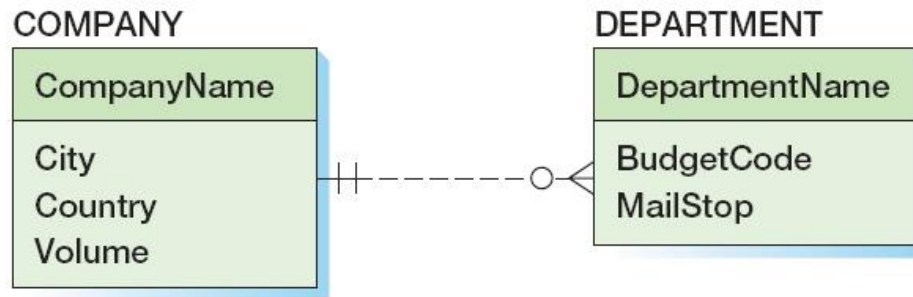
Create Relationships:

1:N Strong Entity Relationships

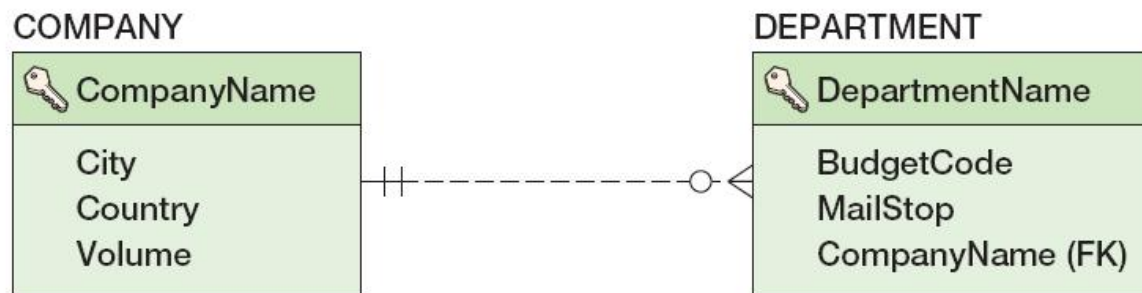
- Place the primary key of the table on the one side of the relationship into the table on the many side of the relationship as the foreign key.
- The *one* side is the **parent** table and the *many* side is the **child** table, so “place the key of the parent in the child.”

Create Relationships:

1:N Strong Entity Relationships



(a) 1:N Relationship Between Strong Entities

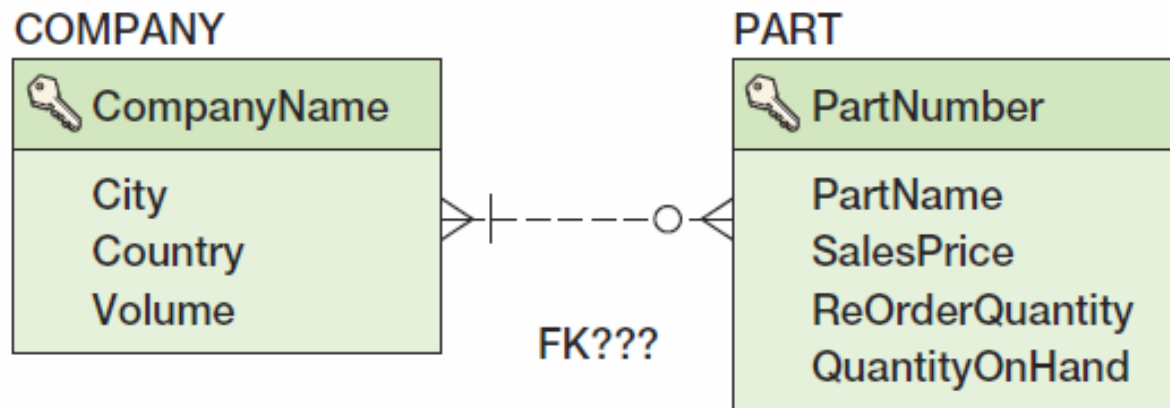


(b) Placing the Primary Key of the Parent in the Child as a Foreign Key

Create Relationships:

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:

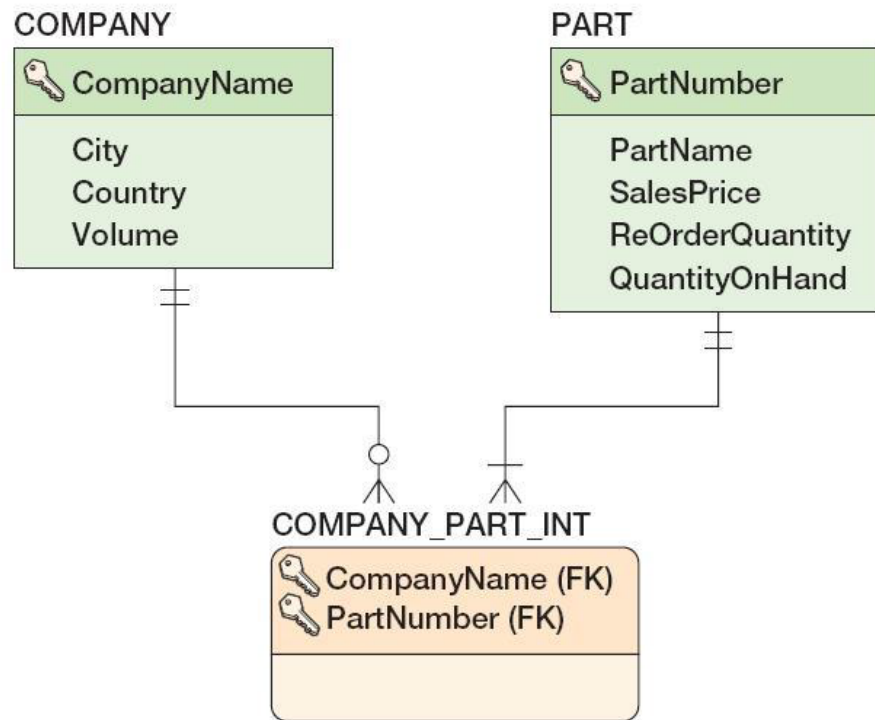
N:M Strong Entity Relationships

- 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)



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

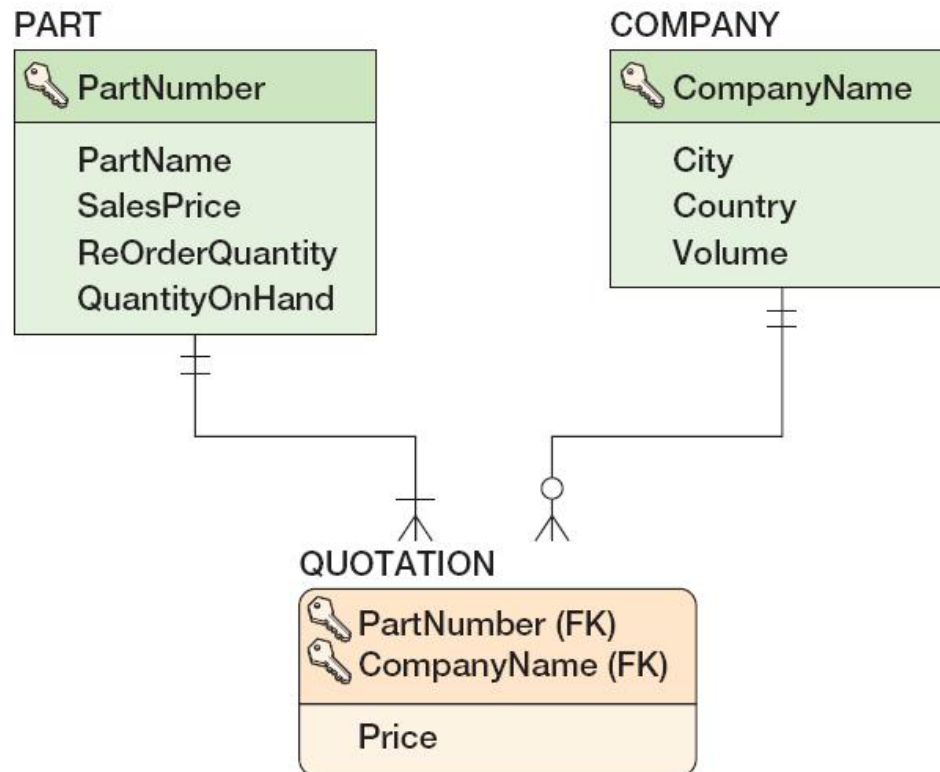
- Representing N:M Relationships
 - We just discussed this
- Association Relationships
- Multivalued Attributes
- Archetype/Instance Relationships

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)



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*.

Cascading Updates and Deletes

- A **cascading update** occurs when a change to the parent's primary key is applied to the child's foreign key.
 - *Surrogate keys never change and there is no need for cascading updates when using them.*
- A **cascading delete** occurs when associated child rows are deleted along with the deletion of a parent row.
 - For strong entities, generally do *not* cascade deletes.
 - For weak entities, generally do cascade deletes.

MySQL

- In the child table: insert or update a child
 - By default, MySQL rejects any INSERT or UPDATE operation that attempts to create a new foreign key value in a child table if there is no a matching key value in the parent table.
- In the parent table: delete or update a parent with children
 - We define one of five referential actions on the child table:
 - cascade, set null, restrict, no action, set default

SQL Sample (Ch. 7)

```
CREATE TABLE parent (  
    id INT NOT NULL,  
    PRIMARY KEY (id)  
) ENGINE=INNODB;
```

```
CREATE TABLE child (  
    id INT,  
    parent_id INT,  
    INDEX par_ind (parent_id),  
    FOREIGN KEY (parent_id)  
        REFERENCES parent(id)  
        ON DELETE CASCADE  
) ENGINE=INNODB;
```

Question

- When a parent row is deleted, if we don't want to cascade the delete action, i.e., removing the associated child rows, what change can you make to the data model?

Actions When the Parent Is Required

[Figure 6-28(a)]

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.

Actions When the Child Is Required [Figure 6-28(b)]

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.

Application Programming: Triggers

- Application programming uses SQL embedded in triggers, stored procedures, and other program code to accomplish a specific task.
- A **trigger** is a stored program that is executed by the DBMS whenever a specified event occurs on a specified table or view (defined in Chapter Seven).
- Triggers are used to enforce specific minimum cardinality enforcement actions not otherwise programmed into the DBMS.
- Triggers will be discussed in detail in Chapters 7, 10 (SQL Server), 10A (Oracle Database), and 10B (MySQL 5.5).

Actions To Apply to Enforce Minimum Cardinality

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!

Implementing Actions for M-O Relationships

- See Figure 6-28(a)
- Make sure that:
 - Every child has a parent.
 - Operations never create orphans.
- The DBMS will enforce the action as long as:
 - Referential integrity constraints are properly defined.
 - The foreign key column is NOT NULL.

Implementing Actions for O-M Relationships

- See Figure 6-28(b)
- The DBMS does not provide much help.
- Triggers or other application codes will need to be written.

Implementing Actions for M-M Relationships

- The worst of all possible worlds:
 - Especially in strong entity relationships.
 - In relationships between strong and weak entities the problem is often easier when all transactions are initiated from the strong entity side.
- All actions in both Figure 6-28(a) and Figure 6-28(b) must be applied simultaneously.
- Complicated and careful application programming will be needed.

Implementing Actions for M-O Relationships: DEPARTMENT and EMPLOYEE

- DEPARTMENT is parent—EMPLOYEE is child.
- Actions on parent:
 - DEPARTMENT rows can be created.
 - DEPARTMENT primary key—cascade updates if not surrogate key.
 - IF a DEPARTMENT is deleted, do we delete the associate EMPLOYEEs?
 - IF YES—cascade deletes.
 - IF NO—prohibit associate employees.

Implementing Actions for M-O Relationships: DEPARTMENT and EMPLOYEE

- Actions on child
 - Set referential integrity constraint and set foreign key to NOT NULL.
 - A new EMPLOYEE must have a valid DEPARTMENT or disallow the insert.
 - EMPLOYEEs can be reassigned to a different DEPARTMENT if a valid DEPARTMENT or disallow the update.
 - EMPLOYEEs can be deleted.

Implementing Actions for O-M Relationships: DEPARTMENT and EMPLOYEE

- DEPARTMENT is parent—EMPLOYEE is child.
- There must be at least one child row for each parent at all time.
- Actions on parent:
 - DEPARTMENT rows can only be created when a relationship is created to a child row—REQUIRES A TRIGGER.
 - DEPARTMENT primary keys can only be updated if at least one EMPLOYEE foreign key is also updated —REQUIRES A TRIGGER.
 - Can a DEPARTMENT be deleted?
 - YES—it is the EMPLOYEE who is required.

Implementing Actions for O-M Relationships: DEPARTMENT and EMPLOYEE

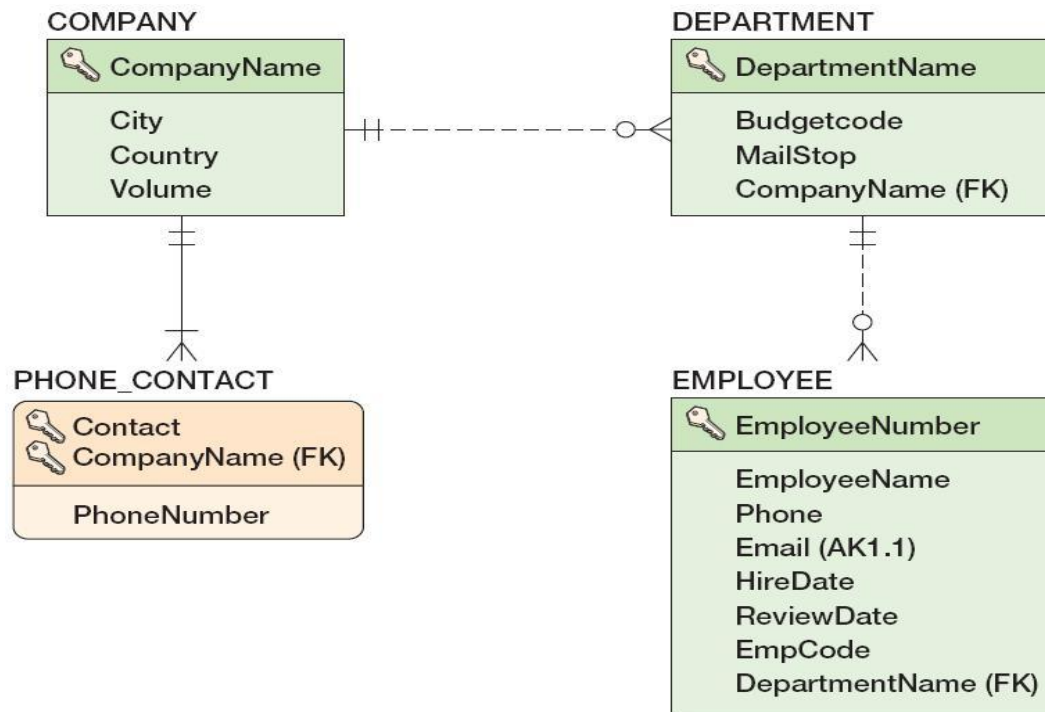
- Actions on child
 - OK to insert a new EMPLOYEE.
 - There must be one EMPLOYEE for each department.
 - Cannot change EMPLOYEE foreign key (DEPARTMENT) if last EMPLOYEE in the DEPARTMENT.
 - Cannot delete an EMPLOYEE if last EMPLOYEE in the DEPARTMENT.

Implementing Actions for M-M Relationships: DEPARTMENT and EMPLOYEE

- DEPARTMENT is parent—EMPLOYEE is child.
- All of the previous (M-O and O-M) apply at the same time!
- This creates conflicts that require careful programming to avoid or fix problems such as:
 - A new DEPARTMENT insert will run a trigger that tries to create a new EMPLOYEE, but the EMPLOYEE row is checked by the DBMS for a valid DEPARTMENT before the transaction is completed.
 - If we try to delete a DEPARTMENT with any EMPLOYEEs we will find the trigger on EMPLOYEE delete will not let us delete the last EMPLOYEE, so we can't delete the DEPARTMENT.

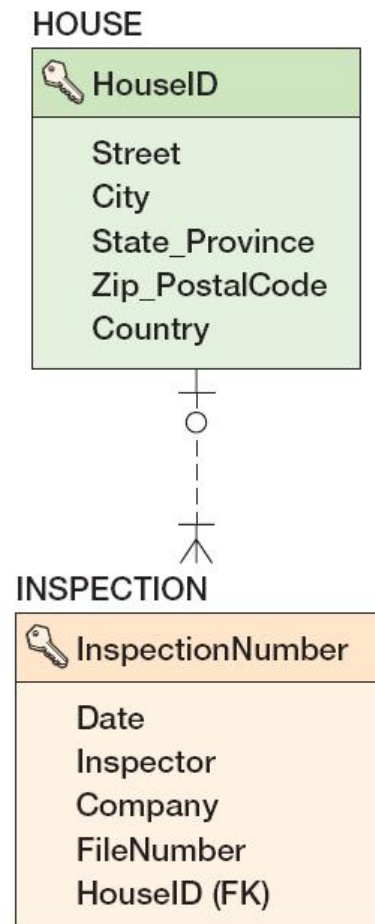
Documenting the Minimum Cardinality Design: Documenting Required Parents

- COMPANY is parent, DEPARTMENT is child.
- The relationship is M-O.
- This can often be done in the database design tools.



Documenting the Minimum Cardinality Design: Documenting Required Children

- HOUSE is parent, INSPECTION is child.
- The relationship is O-M.
- Use documentation based on Figure 6-28(b)—see the next slide.



Documenting the Minimum Cardinality Design: Documenting Required Children

INSPECTION Is Required	Action on HOUSE	Action on INSPECTION
Insert	Trigger to create row in INSPECTION when inserting HOUSE. Disallow HOUSE insert if INSPECTION data are not available.	None.
Modify key or foreign key	Not possible, surrogate key.	Prohibit. HOUSE has surrogate key and inspections never change to a different house.
Delete	None.	Trigger to prohibit if sole INSPECTION report.

Summary of Minimum Cardinality Design

Relationship Minimum Cardinality	Design Decisions to Be Made	Design Documentation
M-O	<ul style="list-style-type: none">• 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	<ul style="list-style-type: none">• 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.