Ch 06 Objectives

- In this chapter, you will learn:
  - About the extended entity relationship (EER) model’s main constructs
  - How entity clusters are used to represent multiple entities and relationships
  - The characteristics of good primary keys and how to select them
  - How to use flexible solutions for special data modeling cases
  - What issues to check for when developing data models based on EER diagrams

6.1 The Extended Entity Relationship Model

- Result of adding more semantic constructs to original entity relationship (ER) model
- Diagram using this model is called an EER diagram (EERD)

Entity Supertypes and Subtypes

- **Entity supertype** (實體超類型)
  - Generic (一般的) entity type related to one or more entity subtypes
  - Contains common characteristics
- **Entity subtypes** (實體子類型)
  - Contains unique characteristics of each entity subtype
- Example: benefits of various types of employee
  - Avoids unnecessary nulls
  - Enables relationship particular to an employee type

Figure 6.1 Nulls created by unique attributes

<table>
<thead>
<tr>
<th>EMP_NUM</th>
<th>EMP_LNAME</th>
<th>EMP_FNAME</th>
<th>EMP_INIT</th>
<th>EMP_LICENSE</th>
<th>EMP_RATING</th>
<th>EMP_MED_TYPE</th>
<th>EMP_MGR_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Kowalcz</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15-Mar-89</td>
</tr>
<tr>
<td>101</td>
<td>Leonov</td>
<td></td>
<td>ATP</td>
<td>SEL, MBS, HCM 1</td>
<td>1</td>
<td></td>
<td>15-Mar-89</td>
</tr>
<tr>
<td>102</td>
<td>Yordanov</td>
<td></td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td>20-Dec-89</td>
</tr>
<tr>
<td>103</td>
<td>Jones</td>
<td>Victoria</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td>20-Aug-89</td>
</tr>
<tr>
<td>104</td>
<td>Lange</td>
<td></td>
<td>ATP</td>
<td>SEL, MBS, HCM 1</td>
<td>1</td>
<td></td>
<td>20-Oct-89</td>
</tr>
<tr>
<td>105</td>
<td>Wilkerson</td>
<td></td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td>20-Nov-89</td>
</tr>
<tr>
<td>106</td>
<td>Zdzik</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td>20-Nov-89</td>
</tr>
<tr>
<td>107</td>
<td>Dusik</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td>20-Jul-89</td>
</tr>
<tr>
<td>108</td>
<td>Kliwenbach</td>
<td></td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td>20-Nov-89</td>
</tr>
<tr>
<td>109</td>
<td>Torras</td>
<td></td>
<td>T</td>
<td>SEL, MBS, HCM 1</td>
<td>1</td>
<td></td>
<td>14-Apr-91</td>
</tr>
<tr>
<td>110</td>
<td>Semlak</td>
<td></td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>01-Dec-91</td>
</tr>
</tbody>
</table>
Specialization Hierarchy (特殊化階層)

- Depicts arrangement of higher-level entity supertypes and lower-level entity subtypes
- Relationships described in terms of “IS-A” (是) relationships
- Subtype exists only within context of supertype
- Every subtype has only one supertype to which it is directly related
- Can have many levels of supertype/subtype relationships

Inheritance (繼承)

- Enables entity subtype to inherit attributes and relationships of supertype
- All entity subtypes inherit their primary key attribute from their supertype
- At implementation level, supertype and its subtype(s) maintain a 1:1 relationship
- Entity subtypes inherit all relationships in which supertype entity participates
- Lower-level subtypes inherit all attributes and relationships from all upper-level supertypes
**Subtype Discriminator** (子類型識別欄位)

- Attribute in **supertype** entity
  - Determines to which entity subtype each supertype occurrence is related
- **Default** (預設) comparison condition for subtype discriminator attribute is **equality** comparison
- Subtype discriminator may be based on **other** comparison condition

**Disjoint and Overlapping Constraints**

- **Disjoint subtypes** (分離子類型)
  - Also known as **non-overlapping** subtypes
  - Subtypes that contain **unique** subset of supertype entity set
- **Overlapping subtypes** (重疊子類型)
  - Subtypes that contain **nonunique** subsets of supertype entity set
Completeness Constraint (完整性限制条件)

- Specifies whether entity supertype occurrence must be a member of at least one subtype
  - Partial completeness (部份完整性)
    - Symbolized by a circle over a single line
    - Some supertype occurrences that are not members of any subtype
  - Total completeness (整體完整性)
    - Symbolized by a circle over a double line
    - Every supertype occurrence must be member of at least one subtype

Specialization and Generalization

- Specialization (特殊化)
  - Identifies more specific entity subtypes from higher-level entity supertype
  - Top-down process
  - Based on grouping unique characteristics and relationships of the subtypes
- Generalization (一般化)
  - Identifies more generic entity supertype from lower-level entity subtypes
  - Bottom-up process
  - Based on grouping common characteristics and relationships of the subtypes

6.2 Entity Clustering (實體叢集)

- “Virtual” entity type used to represent multiple entities and relationships in ERD
- Considered “virtual” or “abstract” because it is not actually an entity in final ERD
- Temporary entity used to represent multiple entities and relationships
- Eliminate undesirable consequences
  - Avoid display of attributes when entity clusters are used
6.3 Entity Integrity: Selecting Primary Keys

- Primary key
  - most important characteristic of an entity
  - Single attribute or some combination of attributes
- Primary key’s function is to guarantee entity integrity
- Primary keys and foreign keys work together to implement relationships
- Properly selecting primary key has direct bearing on efficiency and effectiveness

Natural Keys and Primary Keys

- Natural key (自然索引鍵) is a real-world identifier used to uniquely identify real-world objects
  - Familiar to end users and forms part of their day-to-day business vocabulary
- Generally data modeler uses natural identifier as primary key of entity being modeled
- May instead use composite primary key (組合主索引鍵) or surrogate key (代理主索引鍵)

Primary Key Guidelines

- Attribute that uniquely identifies entity instances in an entity set
  - Could also be combination of attributes
- Main function is to uniquely identify an entity instance or row within a table
- Guarantee entity integrity, not to “describe” the entity
- Primary keys and foreign keys implement relationships among entities
  - Behind the scenes, hidden from user
When to Use Composite Primary Keys

- Composite primary keys useful in two cases:
  - As identifiers of composite entities
    - Where each primary key combination allowed once in M:N relationship
  - As identifiers of weak entities
    - Where weak entity has a strong identifying relationship with the parent entity
  - Automatically provides benefit of ensuring that there cannot be duplicate values
When to Use Composite Primary Keys

• When used as identifiers of weak entities normally used to represent:
  – Real-world object that is existent-dependent (存在相依) on another real-world object
  – Real-world object that is represented in data model as two separate entities in strong identifying relationship
• Dependent entity exists only when it is related to parent entity

When To Use Surrogate Primary Keys

• Especially helpful when there is:
  – No natural key
  – Selected candidate key has embedded semantic contents (包含語意內容)
  – Selected candidate key is too long or cumbersome

• If you use surrogate key
  – Ensure that candidate key of entity in question performs properly
  – Use "unique index" and "not null" constraints

Choose a primary key for this EVENT table

If later on, we need to model the resources needed for events by:

RESOURCE(RSC_ID, RSC_DESCRIPTION, RSC_TYPE, RSC_QTY, RSC_PRICE)

The M:N relationship between EVENT and RESOURCE??

6.4 Design Cases:
Learning Flexible Database Design

• Data modeling and design requires skills acquired through experience
• Experience acquired through practice
• Four special design cases that highlight:
  – Importance of flexible design
  – Proper identification of primary keys
  – Placement of foreign keys
Implementing 1:M Relationships

- Foreign keys work with primary keys to properly implement 1:M relationships in relational model
- Put primary key of the “one” side on the “many” side as foreign key
  - Primary key: parent entity
  - Foreign key: dependent entity

Design Case #1:
Implementing 1:1 Relationships

- In 1:1 relationship two options:
  - Place a foreign key in both entities (not recommended)
  - Place a foreign key in one of the entities
    - Primary key of one of the two entities appears as foreign key of the other

<table>
<thead>
<tr>
<th>CASE</th>
<th>ER RELATIONSHIP CONSTRAINTS</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>One side is mandatory and the other side is optional.</td>
<td>Place the PK of the entity on the mandatory side in the entity on the optional side as a FK, and make the FK mandatory.</td>
</tr>
<tr>
<td>II</td>
<td>Both sides are optional.</td>
<td>Select the FK that causes the fewest number of nulls, or place the FK in the entity in which the relationship role is played.</td>
</tr>
<tr>
<td>III</td>
<td>Both sides are mandatory.</td>
<td>See Case II, or consider revising your model to ensure that the two entities do not belong together in a single entity.</td>
</tr>
</tbody>
</table>

Design Case #2:
Maintaining History of Time-Variant Data

- Normally, existing attribute values replaced with new value without regard to previous value
- Time-variant data:
  - Values change over time
  - Must keep a history of data changes
- Keeping history of time-variant data equivalent to having a multi-valued attribute in your entity
- Must create new entity in 1:M relationships with original entity
- New entity contains new value, date of change
Design Case #3: Fan Traps (扇形陷阱)

- Design trap (陷阱) occurs when relationship is improperly or incompletely identified
  - Represented in a way not consistent with the real world
- Most common design trap is known as fan trap
- Fan trap occurs when one entity is in two 1:M relationships to other entities
  - Produces an association among other entities not expressed in the model
Design Case #4: Redundant Relationships

- Redundancy is seldom a good thing in a database environment
- Occur when there are multiple relationship paths between related entities
- Main concern is that redundant relationships remain consistent across the model
- Some designs use redundant relationships to simplify the design

6.5 Data Modeling Checklist

- Data modeling translates specific real-world environment into data model
  - Represents real-world data, users, processes, interactions
- EERM enables the designer to add more semantic content to the model
- Data modeling checklist helps ensure data modeling tasks successfully performed
- Based on concepts and tools learned since Chapter 3
Summary

- Extended entity relationship (EER) model adds semantics to ER model
  - Adds semantics via entity supertypes, subtypes, and clusters
  - Entity supertype is a generic entity type related to one or more entity subtypes
- Specialization hierarchy
  - Depicts arrangement and relationships between entity supertypes and entity subtypes
- Inheritance means an entity subtype inherits attributes and relationships of supertype

Summary (continued)

- Subtype discriminator determines which entity subtype the supertype occurrence is related to:
  - Partial or total completeness
  - Specialization vs. generalization
- Entity cluster is “virtual” entity type
  - Represents multiple entities and relationships in ERD
  - Formed by combining multiple interrelated entities and relationships into a single object
Summary (continued)

- Natural keys are identifiers that exist in real world
  - Sometimes make good primary keys
- Characteristics of primary keys:
  - Must have unique values
  - Should be nonintelligent
  - Must not change over time
  - Preferably numeric or composed of single attribute

Summary (continued)

- Composite keys are useful to represent
  - M:N relationships
  - Weak (strong-identifying) entities
- Surrogate primary keys are useful when no suitable natural key makes primary key
- In a 1:1 relationship, place the PK of mandatory entity
  - As FK in optional entity
  - As FK in entity that causes least number of nulls
  - As FK where the role is played

Summary (continued)

- Time-variant data
  - Data whose values change over time
  - Requires keeping a history of changes
- To maintain history of time-variant data:
  - Create entity containing the new value, date of change, other time-relevant data
  - Entity maintains 1:M relationship with entity for which history maintained

Summary (continued)

- Fan trap:
  - One entity in two 1:M relationships to other entities
  - Association among the other entities not expressed in model
- Redundant relationships occur when multiple relationship paths between related entities
  - Main concern is that they remain consistent across the model
- Data modeling checklist provides way to check that the ERD meets minimum requirements