Ch. 4 Enhanced E-R Model

- Ternary Relationships
- Sub-classes, Super-classes, Inheritance
- Specialization and Generalization
- Constraints on Specializations and Generalizations
- Aggregation and Association
TERNARY RELATIONSHIPS

SUPPLIER —— SUPPLY —— PROJECT

SUPPLIER: SName

SUPPLY: Quantity

PROJECT: ProjName

PART: PartNo
Subclasses and Superclasses

• Entity type may have sub-grouping that need to be represented explicitly.
  – Example: Employee may grouped into secretary, engineer, manager, technician, exempt and non-exempt.
  – Sub-groups are called subclass and employee superclass
  – relationship can be described as class/subclass
  – presenting member of subclass as distinct object (related via a key attribute of its superclass)
  – entity that is a member of subclass inherits all attributes of superclass
  – It also inherits all relationship that superclass participate in
Specialization

• Top-down design process defines subgroupings within an entity type that are distinctive from other entities in the set.
  – Example: subclasses \{secretary, engineer, etc..\} is a specialization of superclass employee based on job type.
  – May have another specialization \{exempt, non-exempt\} based on method of pay

• These subclasses become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.

• Attached by lines to a circle connected to superclass (for superclass that have 2 or more subclasses)
Figure 4.1 EER diagram notation for representing specialization and subclasses.

Three specializations of EMPLOYEE: 
{SECRETARY, TECHNICIAN, ENGINEER} 
{MANAGER}
Figure 4.2 Some instances of the specialization of EMPLOYEE into the \{SECRETARY, ENGINEER, TECHNICIAN\} set of subclasses.
Generalization

• A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.

• Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.

• The terms specialization and generalization are used interchangeably.
Figure 4.3  Examples of generalization. (a) Two entity types CAR and TRUCK. (b) Generalizing car and TRUCK into VEHICLE.
Design Constraints on Specialization/Generalization

• Constraint on which entities can be members of a given lower-level entity set.
  – Predicate/condition-defined (superclass has attribute specifying the condition of subclass membership)
  – User-defined (no condition specified)

• Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  – Disjoint (entity can be member of at most one subclass in the specialization) denoted by \( d \) inside circle
  – Overlapping (subclasses are not constrained to be disjoint)
Figure 4.4  An attribute-defined specialization on the JobType attribute of EMPLOYEE.
Figure 4.5  Notation for specialization with overlapping (nondisjoint) subclasses.
Design Constraints on Specialization/Generalization

- Completeness constraint – specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a specialization.
  - Total (every entity in superclass is a member of some subclass in the specialization)
  - Total is defined by double lines connecting the circle to superclass
  - example: employee can be either exempt or non-exempt.
  - Partial (not every entity in superclass is a member of some subclass in the specialization)
  - defined by single line connecting the circle to superclass.

- Disjoint and Completeness are independent.
E-R Diagram With Redundant Relationships
Aggregation (Cont.)

• Relationship sets *works-on* and *manages* represent overlapping information

• Eliminate this redundancy via *aggregation*
  – Treat relationship as an abstract entity
  – Allows relationships between relationships
  – Abstraction of relationship into new entity

• Without introducing redundancy, the following diagram represents that:
  – An employee works on a particular job at a particular branch (and may work on different jobs at different branches)
  – An employee, branch, job combination may have an associated manager
E-R Diagram With Aggregation

- Employee
- Job
- Branch
- Works-on

- Manages
- Manager
E-R Design Decisions

- The use of an attribute or entity type to represent an object.
- Whether a real-world concept is best expressed by an entity type or a relationship type.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity type.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity type as a single unit without concern for the details of its internal structure.
Reduction of an E-R Schema to Tables

- Primary keys allow entity types and relationship types to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity type and relationship type there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.
Representing Entity Sets as Tables

- A strong entity set reduces to a table with the same attributes.

<table>
<thead>
<tr>
<th>customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>Turner</td>
<td>Putnam</td>
<td>Stamford</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>244-66-8800</td>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>335-57-7991</td>
<td>Adams</td>
<td>Spring</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>963-96-3963</td>
<td>Williams</td>
<td>Nassau</td>
<td>Princeton</td>
</tr>
</tbody>
</table>
Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set customer with composite attribute name with component attributes first-name and last-name the table corresponding to the entity set has two attributes
    
    name.first-name and name.last-name

- A multivalued attribute M of an entity E is represented by a separate table EM
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
  - E.g. Multivalued attribute dependent-names of employee is represented by a table
    
    employee-dependent-names(employee-id, dname)
  - Each value of the multivalued attribute maps to a separate row of the table EM
    - E.g., an entity with primary key John and dependents Johnson and Peter maps to two rows: (John, Johnson) and (John, Peter)
A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

<table>
<thead>
<tr>
<th>loan-number</th>
<th>payment-number</th>
<th>payment-date</th>
<th>payment-amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-11</td>
<td>53</td>
<td>7 June 2001</td>
<td>125</td>
</tr>
<tr>
<td>L-14</td>
<td>69</td>
<td>28 May 2001</td>
<td>500</td>
</tr>
<tr>
<td>L-15</td>
<td>22</td>
<td>23 May 2001</td>
<td>300</td>
</tr>
<tr>
<td>L-16</td>
<td>58</td>
<td>18 June 2001</td>
<td>135</td>
</tr>
<tr>
<td>L-17</td>
<td>5</td>
<td>10 May 2001</td>
<td>50</td>
</tr>
<tr>
<td>L-17</td>
<td>6</td>
<td>7 June 2001</td>
<td>50</td>
</tr>
<tr>
<td>L-17</td>
<td>7</td>
<td>17 June 2001</td>
<td>100</td>
</tr>
<tr>
<td>L-23</td>
<td>11</td>
<td>17 May 2001</td>
<td>75</td>
</tr>
<tr>
<td>L-93</td>
<td>103</td>
<td>3 June 2001</td>
<td>900</td>
</tr>
<tr>
<td>L-93</td>
<td>104</td>
<td>13 June 2001</td>
<td>200</td>
</tr>
</tbody>
</table>
Representing Relationship Sets as Tables

• A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

• E.g.: table for relationship set *borrower*

<table>
<thead>
<tr>
<th>customer-id</th>
<th>loan-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>019-28-3746</td>
<td>L-11</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>L-23</td>
</tr>
<tr>
<td>244-66-8800</td>
<td>L-93</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>L-17</td>
</tr>
<tr>
<td>335-57-7991</td>
<td>L-16</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>L-14</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>L-15</td>
</tr>
<tr>
<td>963-96-3963</td>
<td>L-17</td>
</tr>
</tbody>
</table>
Redundancy of Tables

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side.

- E.g.: Instead of creating a table for relationship account-branch, add an attribute branch to the entity set account.
Redundancy of Tables (Cont.)

• For one-to-one relationship sets, either side can be chosen to act as the “many” side
  – That is, extra attribute can be added to either of the tables

• If participation is partial on the many side, replacing a table by an extra attribute in the relation corresponding to the “many” side could result in null values

• The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  – E.g. The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-number).
Representing Specialization as Tables

- Form a table for the higher level entity
- Form a table for each lower level entity set, include primary key of higher level entity set and local attributes

<table>
<thead>
<tr>
<th>table</th>
<th>table attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, credit-rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, salary</td>
</tr>
</tbody>
</table>

Drawback: getting information about, e.g., employee requires accessing two tables

Form a table for each entity set with all local and inherited attributes

<table>
<thead>
<tr>
<th>table</th>
<th>table attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>name, street, city, credit-rating</td>
</tr>
<tr>
<td>employee</td>
<td>name, street, city, salary</td>
</tr>
</tbody>
</table>

If specialization is total, no need to create table for generalized entity

Drawback: street and city may be stored redundantly for persons who are both customers and employees
Relations Corresponding to Aggregation

- To represent aggregation, create a table containing primary key of the aggregated relationship and the primary key of the associated entity set.

- E.g. to represent aggregation `manages` between relationship `works-on` and entity set `manager`, create a table:

  `manages(employee-id, branch-name, title, manager-name)`

- Table `works-on` is redundant **provided** we are willing to store null values for attribute `manager-name` in table `manages`