Database Design

9-4
Subtype Mapping
Objectives

This lesson covers the following objectives:

• State and apply the table, column, identifiers, relationship, and integrity constraint rules for mapping:
  – supertype implementations
  – subtype implementations
  – supertype and subtype arc implementations
Purpose

• A carpenter who is building your dream house may know that you will use different types of light bulbs all around the house.

• However, if you do not provide information on where certain types of light bulbs should be installed, you could end up with an overly bright bedroom and a dimly lit kitchen!

• Mapping supertypes and subtypes makes sure that the right information gets stored with each type.
Supertype Implementation: Single Table

• This choice produces a single table for the implementation of the supertype entity and its subtypes.

• This is also called "single-table (or one-table) implementation."

• Rules:
  – Tables: Only one table is created, regardless of the number of subtypes.
  – Columns: The single table gets one column for each attribute of the supertype, along with the original optionality of the attribute.
Supertype Implementation: Single Table

• Rules (cont.):
  – The table also gets a column for each attribute belonging to the subtype, but the columns all become optional.
  – Additionally, a mandatory column should be created to act as a discriminator column to distinguish between the different subtypes of the entity.
  – The value it can take is from the set of all the subtype short names (FTE, PTE, OTR in the example).
  – This discriminator column is usually called \(<table\_short\_name>\_type\), which would be epe\_type in the example.
Supertype Implementation: Single Table

DEPARTMENTS (DPT)
pk * id

AGENCIES (AGY)
pk * id

EMPLOYEES (EPE)

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>first_name</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>last_name</td>
</tr>
<tr>
<td>0</td>
<td>salary</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>hourly_rate</td>
<td></td>
</tr>
<tr>
<td>fk1</td>
<td>*</td>
<td>dpt_id</td>
</tr>
<tr>
<td>fk2</td>
<td>o</td>
<td>agy_id</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>epe_type</td>
</tr>
<tr>
<td>fk3</td>
<td>o</td>
<td>mgr_id</td>
</tr>
</tbody>
</table>
Supertype Implementation: Single Table

• Rules:
  – Identifiers: Unique identifiers transform into primary and unique keys.
  – Relationships: Relationships at the supertype level transform as usual. Relationships at the subtype level are implemented as optional foreign-key columns.
  – Integrity constraints: A check constraint is needed to ensure that for each particular subtype, all columns that come from mandatory attributes are not null.
Supertype Implementation: Single Table

• In the conceptual model, salary is mandatory for full-time employees and hourly rate is mandatory for part-time employees.

• When the EMPLOYEE supertype is implemented as a single table in the physical model, these attributes become optional.

• A check constraint is needed to enforce the business rules modeled in the ERD.
Supertype Implementation: Single Table

• In the example, the code for the check constraint would look like this:
  – CHECK (epe_type = ‘FTE’ and salary is not null and hourly_rate is null and agy_id is null)
  – OR (epe_type = ‘PTE’ and salary is null and hourly_rate is not null and agy_id is not null)
Supertype Implementation: Single Table

• The code checks that if it is a full-time employee (epe_type = ‘FTE’), then a value must exist in the salary column and the hourly_rate and agy_id columns must be empty.

• Conversely, if it is a part-time employee (epe_type = ‘PTE’), then a value must exist in hourly_rate and agy_id, but salary must be left blank.
Supertype Implementation: Single Table

Sample Data for EMPLOYEES

<table>
<thead>
<tr>
<th>id</th>
<th>first_name</th>
<th>last_name</th>
<th>salary</th>
<th>hourly_rate</th>
<th>dpt_id</th>
<th>agy_id</th>
<th>epe_type</th>
<th>epe_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>Joan</td>
<td>Merrick</td>
<td>50000</td>
<td></td>
<td>10</td>
<td></td>
<td>FTE</td>
<td>111</td>
</tr>
<tr>
<td>111</td>
<td>Sylvia</td>
<td>Patakis</td>
<td>90000</td>
<td></td>
<td>10</td>
<td></td>
<td>FTE</td>
<td></td>
</tr>
<tr>
<td>2101</td>
<td>Marcus</td>
<td>Rivera</td>
<td>65.00</td>
<td></td>
<td>10</td>
<td>17</td>
<td>PTE</td>
<td>111</td>
</tr>
<tr>
<td>2102</td>
<td>Hector</td>
<td>Chen</td>
<td>75.00</td>
<td></td>
<td>25</td>
<td>17</td>
<td>PTE</td>
<td>45</td>
</tr>
<tr>
<td>45</td>
<td>Rajesh</td>
<td>Vishwan</td>
<td>90000</td>
<td></td>
<td>25</td>
<td></td>
<td>FTE</td>
<td></td>
</tr>
</tbody>
</table>
When Do You Choose the Single Table/Supertype Implementation?

• The single-table implementation is a common and flexible implementation.

• It is the one you are likely to consider first and is especially appropriate where:
  – Most of the attributes are at the supertype level.
  – Most of the relationships are at the supertype level.
  – Business rules are globally the same for the subtypes.
Subtype Implementation: Two Table

• This is also called "two-table implementation."
• You create a table for each of the subtypes.
• So, in reality, you could have more than two tables, if you had more than two subtypes.
Subtype Implementation: Two Table

• Rules:
  – Tables: One table per first-level subtype.
  – Columns: Each table gets one column for each attribute of the supertype along with its original optionality.
  – Each table also gets one column for each attribute belonging to the subtype along with its original optionality.
Subtype Implementation: Two Table

• Rules (cont.):
  – Identifiers: The primary UID at the supertype level creates a primary key for each table. Secondary UIDs of the supertype become unique keys in each table.
  – Relationships: All tables get a foreign key for a relationship at the supertype level, with the original optionality.
    • For relationships at the subtype levels, the foreign key is implemented in the table it is mapped to.
    • Original optionality is retained.
Subtype Implementation: Two Table

**CLOTHING**
- # id
- * material

**SHIRT**
- * sleeve length
- * neck size
- o collar style

**SHOE**
- * size
- * buckle style
- o heel height

**SHIRTS (SHT)**

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>material</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>sleeve_length</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>neck_size</td>
</tr>
<tr>
<td></td>
<td>o</td>
<td>collar_style</td>
</tr>
<tr>
<td>fk1</td>
<td>o</td>
<td>tlr_id</td>
</tr>
<tr>
<td>fk2</td>
<td>*</td>
<td>mnr_id</td>
</tr>
</tbody>
</table>

**SHOES (SHE)**

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>material</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>size</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>buckle_style</td>
</tr>
<tr>
<td></td>
<td>o</td>
<td>heel_height</td>
</tr>
<tr>
<td>fk1</td>
<td>o</td>
<td>clr_id</td>
</tr>
<tr>
<td>fk2</td>
<td>*</td>
<td>mnr_id</td>
</tr>
</tbody>
</table>

- produced by
- altered by
- repaired by

**MANUFACTURER**
- # id

**TAILOR**
- # id

**COBLLER**
- # id

**SHIRT**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**SHOE**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**SHIRTS (SHT)** refers to manufacturers

**SHOES (SHE)** refers to manufacturers

**SHIRTS (SHT)**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**SHOES (SHE)**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**SHIRTS (SHT)** refers to manufacturers

**SHOES (SHE)** refers to tailors

**CLOTHING**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**CLOTHING** refers to manufacturers

**CLOTHING** refers to tailors

**CLOTHING** refers to cobblers

**CLOTHING**
- produced by MANUFACTURER
- altered by TAILOR
- repaired by COBLLER

**CLOTHING** refers to manufacturers

**CLOTHING** refers to tailors

**CLOTHING** refers to cobblers
Subtype Implementation: Two Table

- In the example, a separate table would be created for SHIRTS and SHOES.

**Sample Data for SHIRTS**

<table>
<thead>
<tr>
<th>id</th>
<th>material</th>
<th>sleeve_length</th>
<th>neck_size</th>
<th>collar_style</th>
<th>mnr_id</th>
<th>tlr_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>linen</td>
<td>33</td>
<td>16</td>
<td>button down</td>
<td>65</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>wool</td>
<td>32</td>
<td>15.5</td>
<td>nehru</td>
<td>65</td>
<td>22</td>
</tr>
<tr>
<td>14</td>
<td>cotton</td>
<td>33</td>
<td>15.5</td>
<td></td>
<td>60</td>
<td>22</td>
</tr>
</tbody>
</table>

**Sample Data for SHOES**

<table>
<thead>
<tr>
<th>id</th>
<th>material</th>
<th>size</th>
<th>buckle_style</th>
<th>heel_height</th>
<th>mnr_id</th>
<th>clr_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>leather</td>
<td>7.5</td>
<td>monkstrap</td>
<td>1.5</td>
<td>75</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>canvas</td>
<td>8</td>
<td>velcro</td>
<td>1</td>
<td>70</td>
<td>44</td>
</tr>
</tbody>
</table>
When to Consider Subtype Implementation

Subtype implementation may be appropriate when:

• Subtypes have very little in common. There are few attributes at the supertype level and several at the subtype level.

• Most of the relationships are at the subtype level.

• Business rules and functionality are quite different between subtypes.

• How tables are used is different -- for example, one table is being queried while the other is being updated.
Modeling the Supertype as an Arc

- A supertype entity and its subtypes can be modeled as an arc relationship.
- Here again is the original ERD with the supertype and subtypes.

```
CLOTHING
  # id
  * material

SHIRT
  * sleeve length
  * neck size
  o collar style

SHOE
  * size
  * buckle style
  o heel height

MANUFACTURER
  # id

TAILOR
  # id

COBBLER
  # id
```

Produced by:
- CLOTHING
- SHIRT
- SHOE

The producer of:
- MANUFACTURER

Altered by:
- CLOTHING
- SHIRT
- SHOE

The alterer of:
- TAILOR

Repaired by:
- CLOTHING
- SHIRT
- SHOE

The repairer of:
- COBBLER
Model An Arc Illustrated

- In this ERD, we have redrawn the CLOTHING supertype and its subtypes of SHIRT and SHOE as standalone entities...
Model An Arc Illustrated

• ...with each one having mandatory 1:1 relationships with the supertype. The relationships are in an arc.
Supertype and Subtype (Arc) Implementation

• This choice produces one table for every entity.
• The supertype table has a foreign key for each subtype table.
• These foreign keys represent exclusive relationships.
• They are optional because only one of them can have a value for each row in the table.
Supertype and Subtype (Arc) Implementation

• Rules
  – Tables: As many tables are created as there are subtypes, as well as one for the supertype.
  – Columns: Each table gets a column for all attributes of the entity it is based on, with the original optionality.

• Identifiers: The primary UID of the supertype level creates a primary key for each of the tables.
  – All other unique identifiers become unique keys in their corresponding tables.
Supertype and Subtype (Arc) Implementation

• Relationships: All tables get a foreign key for a relevant relationship at the entity level, with the original optionality.

• Integrity constraints: Two additional columns are created in the table based on the supertype.

• They are foreign-key columns referring to the tables that implement the subtypes.
Supertype and Subtype (Arc) Implementation

• The columns are optional because the foreign keys are in an arc.
• An additional check constraint is needed to implement the arc.
• The foreign-key columns are also unique keys because they implement a mandatory 1:1 relationship.
Supertype and Subtype (Arc) Implementation

CLOTHING (CTG)

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>material</td>
</tr>
<tr>
<td>fk1, uk1</td>
<td>o</td>
<td>sht_id</td>
</tr>
<tr>
<td>fk2, uk2</td>
<td>o</td>
<td>she_id</td>
</tr>
<tr>
<td>fk3</td>
<td>*</td>
<td>mnr_id</td>
</tr>
</tbody>
</table>

SHIRTS (SHT)

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sleeve_length</td>
</tr>
<tr>
<td></td>
<td></td>
<td>neck_size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>collar_style</td>
</tr>
<tr>
<td>fk1</td>
<td>o</td>
<td>tlr_id</td>
</tr>
</tbody>
</table>

refers to tailors
refers to shirts
refers to manufacturers

SHOES (SHE)

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Optionality</th>
<th>Column Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk</td>
<td>*</td>
<td>id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>buckle_style</td>
</tr>
<tr>
<td></td>
<td></td>
<td>heel_height</td>
</tr>
<tr>
<td>fk1</td>
<td>o</td>
<td>clr_id</td>
</tr>
</tbody>
</table>

refers to cobblers
When to Consider Both a Supertype and Subtype (Arc) Implementation

• This implementation is rarely used, but it could be appropriate when:
  – Subtypes have very little in common and each table represents information that can be used independently.
  – For example, when the CLOTHING table gives all global information, and both SHOES and SHIRTS give specific information, and the combination of global and specific information is hardly ever needed.
  – Business rules and functionality are quite different between all types.
  – How tables are used is different.
Terminology

Key terms used in this lesson included:

• Arc implementations
• Subtype implementations
• Supertype implementations
Summary

In this lesson, you should have learned how to:

• State and apply the table, column, identifiers, relationship, and integrity constraint rules for mapping:
  – supertype implementations
  – subtype implementations
  – supertype and subtype arc implementations