Hibernate Mapping

"Code with Passion!"

Topics

- Mapping cardinality relationship
  - One-To-Many
  - Many-To-Many
  - One-has-Map
- Mapping inheritance relationship
  - One table for each class hierarchy
  - One table for each subclass
  - One table per each concrete class implementation

Mapping Cardinality Relationship

Mapping Cardinality Relationships

- one-to-one
- one-to-many (or many-to-one)
- many-to-many

One to One Relationship

- Expresses a relationship between two classes where each instance of the first class is related to a single instance of the second class
- Can be expressed in the database tables in two ways
  - Using foreign key constraint from one table onto a unique identifier column of the other
  - Giving each of the respective tables the same primary key values

Mapping Cardinality Relationship: One-To-Many

One-To-Many Relationship Mapping

1. <set> - Set (no duplicate)
2. <list> - List (order/index)
3. <array> - Array (order/index)
4. <bag> - (duplicate allowed/no order)

Mapping Cardinality Relationship:

One-To-Many:

Using <set>

1. One-to-Many relationship: Using <set> in mapping file (1)

- An event has many speakers
- Mapping file of Parent class - *Event.hbm.xml*

```
<class name="Event" table="events">
        <id name="id" column="uid" type="long" unsaved-value="null">
            <generator class="increment"/>
        </id>
        <property name="name" type="string" length="100"/>

        <set name="speakers" cascade="all">
            <key column="event_id"/>
            <one-to-many class="Speaker"/>
        </set>
</class>
```

Foreign key in the child table

Child class

1. One-to-Many relationship: Using <set> in mapping file (2)

- An event has many speakers
- Mapping file of Child class – *Speaker.hbm.xml* – note there is no special setting

```
<class name="Speaker" table="speakers">
    <id name="id" column="uid" type="long">
        <generator class="increment"/>
    </id>
    <property name="firstName" type="string" length="20"/>
    <property name="lastName" type="string" length="20"/>
</class>
```

One to Many relationship: Using Set in Domain Class

- An event has many speakers and attendees

```
public class Event {

  private Long id;
  private String name;
  private Date startDate;
  private int duration;

  // Event has one-to-many relationship with Speaker
  private Set speakers;

  // ...
```

One to Many relationship: Creating Object Instance

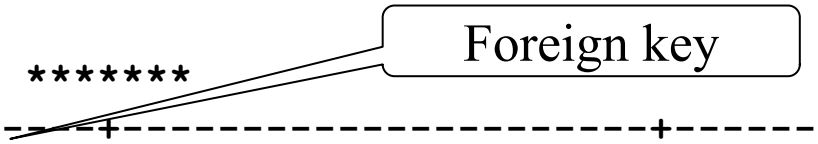
- An event has many speakers and attendees

```
// Create an Event object which has one to many relationship
// with Speaker objects.

Event event = new Event();
event.setName("Java Conference");
event.setSpeakers(new HashSet());
event.getSpeakers().add(new Speaker("Sang", "Shin"));
event.getSpeakers().add(new Speaker("Dave", "Smith"));
event.getSpeakers().add(new Speaker("Bill", "Clinton"));

session.saveOrUpdate(event);
```

One to Many relationship: Using <set> in the mapping file

- Tables

```
******** Table: events *******
+-------------+----------------------+------------+-------------+-------------+
|     UID     |         NAME         | START_DATE |   DURATION  | LOCATION_ID |
+-------------+----------------------+------------+-------------+-------------+
| 1           | Java Conference      |            0             |             |
| 2           | Passion Conference   |            0             |             |
+-------------+----------------------+------------+-------------+-------------+

******** Table: speakers *******
+-------------+-------------+----------------------+----------------------+
|     UID     |   EVENT_ID  |       FIRSTNAME      |       LASTNAME       |
+-------------+-------------+----------------------+----------------------+
| 1           | 1           | Sang                 | Shin                 |
| 2           | 1           | Dave                 | Smith                |
| 3           | 1           | Bill                 | Clinton              |
| 4           | 2           | Daniel               | Jones                |
| 5           | 2           | James                | Gosling              |
+-------------+-------------+----------------------+----------------------+
```



Mapping Cardinality Relationship:

One-To-Many:

Using <list>

2. One to Many relationship: Using <list> in mapping file

- A “Group” has many stories
- Mapping file of parent class - *Group.hbm.xml*

```
<class name="Group" table="grouptable">
    <id name="id" unsaved-value="0">
        <generator class="increment" />
    </id>

    <list name="stories" cascade="all">
        <key column="parent_id" />
        <!-- index in a single list -->
        <index column="idx" />
        <one-to-many class="Story" />
    </list>
    <property name="name" type="string" />
</class>
```

2. One to Many relationship: Using <list> in mapping file (2)

- Group has many stories
- Mapping file of child class – *Story.hbm.xml* – note there is no special setting

```
<class name="Story" table="story">
    <id name="id" unsaved-value="0">
        <generator class="increment" />
    </id>
    <property name="info" />
</class>
```

One to Many relationship: Using List in Domain Class

- Group has many stories

```
public class Group {

    private int id;
    private String name;

    private List stories;

    public void setStories(List l) {
        stories = l;
    }

    public List getStories() {
        return stories;
    }
    // ...
```

One to Many relationship: Creating Object Instances

- Group has many stories

```
ArrayList list = new ArrayList();
list.add(new Story("Tom Jones"));
list.add(new Story("Beatles"));
list.add(new Story("Elvis"));

Group sp = new Group("Singers");
sp.setStories(list);

ArrayList list2 = new ArrayList();
list2.add(new Story("Bill Clinton"));
list2.add(new Story("Ronald Reagan"));

Group sp2 = new Group("Politicians");
sp2.setStories(list2);
```

One to Many relationship: Using <list> in the mapping file

- Tables

```
******** Table: grouptable *******
+-------------+----------------------+
|      ID     |          NAME        |
+-------------+----------------------+
| 1           | Singers              |
| 2           | Politicians          |
+-------------+----------------------+

******** Table: story *******
+-------------+-----------------------------------+-------------+-------------+
|      ID     |                INFO               |     IDX     |  PARENT_ID  |
+-------------+-----------------------------------+-------------+-------------+
| 1           | Tom Jones                         | 0           | 1           |
| 2           | Beatles                           | 1           | 1           |
| 3           | Elvis                             | 2           | 1           |
| 4           | Bill Clinton                      | 0           | 2           |
| 5           | Ronald Reagan                     | 1           | 2           |
+-------------+-----------------------------------+-------------+-------------+
```

Mapping Cardinality Relationship:

One-To-Many:

Using <array>

3. One to Many relationship: Using <array> in mapping file

- Group has many stories
- Mapping file of parent class - *Group.hbm.xml*

```
<class name="Group" table="grouptable">
    <id name="id" unsaved-value="0">
        <generator class="increment"/>
    </id>

    <array name="stories" cascade="all">
        <key column="parent_id"/>
        <index column="idx"/>
        <one-to-many class="Story"/>
    </array>
    <property name="name" type="string"/>
</class>
```

3. One to Many relationship: Using <array> in mapping file (2)

- Group has many stories
- Mapping file of child class - *Story.hbm.xml*

```
<class name="Story" table="story">
    <id name="id" unsaved-value="0">
        <generator class="increment"/>
    </id>
    <property name="info"/>
</class>
```

One to Many relationship: Using an array in Domain Class

- Group has many stories

```
public class Group {

    private int id;
    private String name;

    // Group object has an array of Story objects
    private Story[] stories;

    public void setStories(Story[] l) {
        stories = l;
    }

    public Story[] getStories() {
        return stories;
    }

    // ...
```

One to Many relationship: Creating an Object Instance

- Group has many stories

```
// Create an Group object which has one to many relationship
// with Story objects.

Group sp = new Group("Group Name");
sp.setStories(new Story[]{new Story("Story Name 1"),
                          new Story("Story Name 2")});
```

One to Many relationship: Using <array> in the mapping file

- Tables

```
******** Table: grouptable *******
+-------------+----------------------+
|      ID     |          NAME        |
+-------------+----------------------+
| 1           | Singers              |
| 2           | Politicians          |
+-------------+----------------------+

******** Table: story *******
+-------------+-----------------------------------+-------------+-------------+
|      ID     |                INFO               |     IDX     |  PARENT_ID  |
+-------------+-----------------------------------+-------------+-------------+
| 1           | Tom Jones                         | 0           | 1           |
| 2           | Beatles                           | 1           | 1           |
| 3           | Bill Clinton                      | 0           | 2           |
| 4           | Ronald Reagan                     | 1           | 2           |
+-------------+-----------------------------------+-------------+-------------+
```

Mapping Cardinality Relationship:

One-To-Many:

Using <bag>

4. One to Many relationship: Using <bag> in mapping file

- Using bag for association mapping, the domain objects in the collection could be duplicate (non-set) and without order (non-list)
- *Group.hbm.xml*

```
<class name="Group" table="grouptable">

    <id name="id" unsaved-value="0">
        <generator class="increment"/>
    </id>

    <bag name="stories" cascade="all">
        <key column="parent_id"/>
        <one-to-many class="Story"/>
    </bag>

    <property name="name" type="string"/>

</class>
```

One to Many relationship: Using an List in Domain Class

- Group has many stories

```
public class Group {

    private int id;
    private String name;
    private List stories;

    public void setStories(List l) {
        stories = l;
    }

    public List getStories() {
        return stories;
    }

    // ...
```

One to Many relationship: Creating an Object Instance

- Group has many stories

```
// Create an Group object which has one to many relationship
// with Story objects.

ArrayList list = new ArrayList();
list.add(new Story("Story Name 1"));
list.add(new Story("Story Name 2"));
Group sp = new Group("Group Name");
sp.setStories(list);
```

One to Many relationship: Using <bag> in the mapping file

- Tables

```
******** Table: grouptable *******
+-------------+----------------------+
|      ID     |          NAME        |
+-------------+----------------------+
| 1           | Singers              |
| 2           | Polticians           |
+-------------+----------------------+

******** Table: story *******
+-------------+-----------------------------------+-------------+
|      ID     |                INFO               |  PARENT_ID  |
+-------------+-----------------------------------+-------------+
| 1           | Tom Jones                         | 1           |
| 2           | Beatles                           | 1           |
| 3           | Ronald Reagan                     | 2           |
+-------------+-----------------------------------+-------------+
```

Lab:

Exercise 1: One to Many Relationship
3516_hibernate_mapping.zip

Mapping Cardinality Relationship: Many-To-Many

Many to Many relationship

- *Speakers and Events relationship:* A Speaker speaks in many events and an Event has many speakers
- *EventManyToMany.hbm.xml*

```
<class name="EventManyToMany" table="m_events">
    <id name="id" column="uid" type="long" unsaved-value="null">
         <generator class="increment"/>
    </id>
    <property name="name" type="string" length="100"/>
    <property name="startDate" column="start_date" type="date"/>
    <property name="duration" type="integer"/>

    <!-- events_speakers is a join table -->
    <set name="speakers" table="events_speakers" cascade="all">
        <key column="event_id"/>
        <many-to-many column="speaker_id" class="SpeakerManyToMany"/>
    </set>
</class>
```

Many to Many relationship

- *SpeakerManyToMany.hbm.xml*

```
<class name="SpeakerManyToMany" table="m_speakers">
    <id name="id" column="uid" type="long">
        <generator class="increment"/>
    </id>
    <property name="firstName" type="string" length="20"/>
    <property name="lastName" type="string" length="20"/>

     <!-- events_speakers is a join table -->
    <set name="events" table="events_speakers" cascade="all">
        <key column="speaker_id"/>
        <many-to-many column="event_id" class="EventManyToMany"/>
    </set>

</class>
```

Many to Many relationship:

- Event has many speakers

```
public class EventManyToMany {

    private Long id;
    private String name;
    private Date startDate;
    private int duration;
    private Set speakers;
    private Set attendees;

    public void setSpeakers(Set speakers) {
        this.speakers = speakers;
    }

    public Set getSpeakers() {
        return speakers;
    }

    //
```

Many to Many relationship:

- A speaker speaks in many events

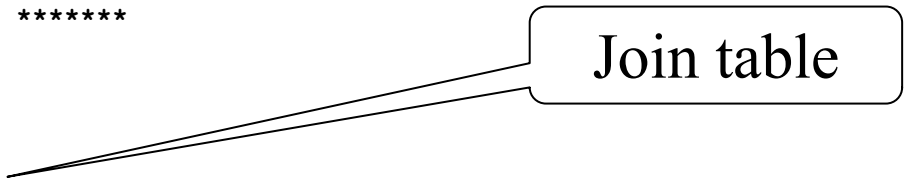
```
public class SpeakerManyToMany {

    private Long id;
    private String firstName;
    private String lastName;
    private Set events;

    public Set getEvents() {
        return this.events;
    }

    public void setEvents(Set events) {
        this.events = events;
    }

    // ...
```

Many to Many relationship: Creating object instances

```
// Event has Many-To-Many relationship with Speaker
EventManyToMany event = new EventManyToMany();
event.setName("JavaOne conference");
event.setSpeakers(new HashSet());
event.getSpeakers().add(new SpeakerManyToMany("Sang", "Shin", event));
event.getSpeakers().add(new SpeakerManyToMany("Joe", "Smith", event));
event.getSpeakers().add(new SpeakerManyToMany("x", "Man", event));

session.save(event);

// Create the second event with multiple speakers
EventManyToMany event2 = new EventManyToMany();
event2.setName("Passion Conference");
event2.setSpeakers(new HashSet());
event2.getSpeakers().add(new SpeakerManyToMany("Sang", "Shin", event2));
event2.getSpeakers().add(new SpeakerManyToMany("Shelly", "Lumm",
   event2));
event2.getSpeakers().add(new SpeakerManyToMany("Diane", "Woon", event2))

session.save(event2);
```

Many to Many relationship

```
******** Table: m_events *******
+-------------+----------------------+------------+-------------+-------------+
|     UID     |         NAME         | START_DATE |   DURATION  | LOCATION_ID |
+-------------+----------------------+------------+-------------+-------------+
| 1           | JavaOne conference   |              0           |
| 2           | Passion Conference   |              0           |
+-------------+----------------------+------------+-------------+-------------+

******** Table: m_speakers *******
+-------------+----------------------+----------------------+
|     UID     |       FIRSTNAME      |       LASTNAME       |
+-------------+----------------------+----------------------+
| 1           | Joe                  | Smith                |
| 2           | John                 | Smith                |
| 3           | Sang                 | Shin                 |
| 4           | Sang                 | Shin                 |
| 5           | Diane                | Woon                 |
| 6           | Shelly               | Lumm                 |
+-------------+----------------------+----------------------+

******** Table: events_speakers *******
+-------------+-------------+
|   EVENT_ID  |  SPEAKER_ID |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 4           |
| 2           | 5           |
| 2           | 6           |
| 2           | 4           |
| 2           | 5           |
```



Lab:

Exercise 2: Many to Many Relationship
3516_hibernate_mapping.zip

Mapping Cardinality Relationship: Using <map>

One-Has-Collection relationship: Using <map> in mapping file

- SupportProperty class has a field whose type is Map
- *SupportProperty.hbm.xml*

```
<class name="SupportProperty" table="supportproperty">
    <id name="id">
        <generator class="increment"/>
    </id>

    <map name="properties">
        <key column="id"/>
        <index column="property_name" type="string"/>
        <element column="property_value" type="string"/>
    </map>

    <property name="name" type="string"/>
</class>
```

One-Has-Collection relationship: Domain Class

- SupportProperty class has a field whose type is Map

```
public class SupportProperty {

    private int id;
    private String name;
    private Map properties;

    public void setProperties(Map m) {
        properties = m;
    }

    public Map getProperties() {
        return properties;
    }

    // ...
```

One-Has-Collection relationship: Creating an Object Instance

- SupportProperty class has a field whose type is Map

```
// Create Domain object, SupportProperty object has a Map object.
SupportProperty sp = new SupportProperty();
sp.setName("MyProperties");

HashMap h = new HashMap();
h.put("car", "ford");
h.put("house", "lexington");
sp.setProperties(h);

session.save(sp);

// Create another object instance
SupportProperty sp2 = new SupportProperty();
sp2.setName("YourProperties");
HashMap h2 = new HashMap();
h2.put("tv", "samsung");
h2.put("house", "lexington");
sp2.setProperties(h2);

session.save(sp2);
```

One to Many relationship: Using <map> in the mapping file

- Tables

```
******** Table: supportproperty *******
+-------------+----------------------+
|      ID     |          NAME        |
+-------------+----------------------+
| 1           | MyProperties         |
| 2           | YourProperties       |
+-------------+----------------------+

******** Table: properties *******
+-------------+----------------------+----------------------+
|      ID     |     PROPERTY_NAME    |    PROPERTY_VALUE    |
+-------------+----------------------+----------------------+
| 1           | car                  | ford                 |
| 1           | house                | lexington            |
| 2           | tv                   | samsung              |
| 2           | house                | lexington            |
+-------------+----------------------+----------------------+
```

Lab:

Exercise 3: Hashmap Relationship
3516_hibernate_mapping.zip

Mapping Inheritance:
3 Different Ways

Inheritance Relationship Table Representations

- 3 different ways
  - One table for each class hierarchy
  - One table for each subclass
  - One table per each concrete class implementation
- Each of these techniques has different costs and benefits

Example Class Hierarchy

- Book class is parent class

```
public class Book {
   int id;
   String title;
   String artist;
   Date purchaseDate;
   double cost;
   ..
}
```

- SpecialEditionBook is a child class of Book class

```
public class SpecialEditionBook extends Book {
   private String newfeatures;
   ..
}
```

- InternationalBook is a child class of Book class

```
public class InternationalBook extends Book {
   private String languages;
   private int region;
   ..
}
```

Mapping Inheritance: 1 Table for the Class Hierarchy

One Table per Class Hierarchy

- A single table for the whole class hierarchy
  - The table contains fields of all classes
- Discriminator column contains key to identify the base type
  - The column indicates which type/subtype a row belongs
- Advantages
  - Offers best performance even for in the deep hierarchy since single select may suffice
- Disadvantages
  - Change to any class in the hierarchy requires a change of the table
  - Wasted space

One Table per Class Hierarchy

- How to define the mapping
  - Use <*subclass*> element with *extends* and *discriminator-value* attributes

One Table per Class Hierarchy: Parent (Book.hbm.xml)

```
<class name="Book" table="Book"
       discriminator-value="Book">
    <id name="id" type="integer"  unsaved-value="0">
        <generator class="increment"/>
    </id>

    <!-- Parent class mapping file specifies the
         discriminator column -->
    <discriminator column="Book_type" type= "string"/>

    <property name="title"/>
    <property name="artist"/>
    <property name="purchasedate" type="date"/>
    <property name="cost" type="double"/>

</class>
```

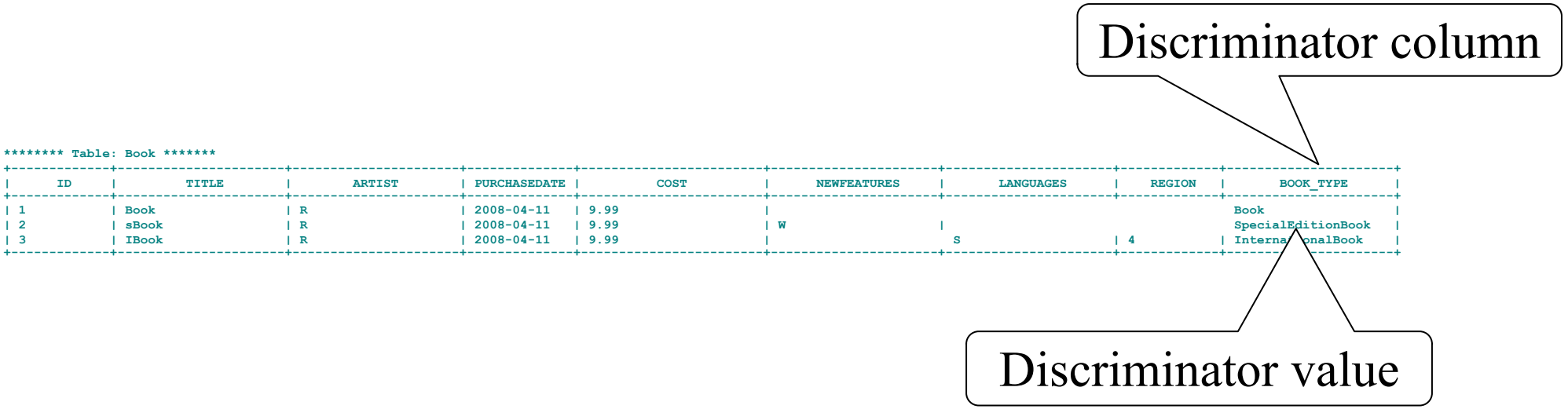
One Table per Class Hierarchy: Child (SpecialEditionBook.hbm.xml)

```
<hibernate-mapping>

    <subclass name="SpecialEditionBook"
            extends="Book"
            discriminator-value="SpecialEditionBook">
        <property name="newfeatures" type="string" />
    </subclass>

</hibernate-mapping>
```

One Table per Class Hierarchy: Child (InternationalBook.hbm.xml)

```
<hibernate-mapping>

    <subclass name="InternationalBook"
            extends="Book"
            discriminator-value="InternationalBook">
        <property name="languages"/>
        <property name="region" />
    </subclass>

</hibernate-mapping>
```

One Table per Class Hierarchy



Mapping Inheritance: 1 Table for Subclass

One Table per Subclass

- One table for each class in the hierarchy
  - Common fields (fields of the parent class) are maintained in the parent table
  - Subclass table maintain only the subclass specific fields
  - Foreign key relationship exists between common table and subclass tables
- Advantages
  - Does not require complex changes to the schema when a class is modified
  - Works well with shallow hierarchy
- Disadvantages
  - Can result in poor performance with deep hierarchy – as hierarchy grows, the number of joins required to construct a leaf class also grows

One Table per Subclass

- How to define the mapping
  - Use *<joined-subclass>* element with *extends* attribute in the mapping file of the subclass

One Table per Subclass: Parent (Book.hbm.xml)

```
<!-- Mapping file of Parent class has no hierarchical
     relationship specific element -->
<class name="Book" table="Book">
    <id name="id" type="integer" unsaved-value="0">
        <generator class="increment" />
    </id>

    <property name="title" />
    <property name="artist" />
    <property name="purchasedate" type="date" />
    <property name="cost" type="double" />

</class>
```

One Table per Subclass - Child (SpecialEditionBook.hbm.xml)

```
<joined-subclass name="SpecialEditionBook"
                 extends="Book"
                 table="SpecialEditionBook">
    <key column="id" />
    <property name="newfeatures" type="string" />
</joined-subclass>
```

One Table per Subclass - Child (InternationalBook.hbm.xml)

```
<joined-subclass name="InternationalBook"
                 extends="Book"
                 table="InternationalBook">
    <key column="id" />
    <property name="languages"/>
    <property name="region"/>
</joined-subclass>
```

One Table per Subclass

```
******** Table: Book *******
+-------------+-----------------------+-----------------------+--------------+--------------------------+
|      ID     |         TITLE         |        ARTIST         | PURCHASEDATE |           COST           |
+-------------+-----------------------+-----------------------+--------------+--------------------------+
| 1           | Book                  | R                     | 2008-04-11   | 9.99                     |
| 2           | sBook                 | R                     | 2008-04-11   | 9.99                     |
| 3           | IBook                 | R                     | 2008-04-11   | 9.99                     |
+-------------+-----------------------+-----------------------+--------------+--------------------------+

******** Table: SpecialEditionBook *******
+-------------+---------------------------------+
|      ID     |           NEWFEATURES           |
+-------------+---------------------------------+
| 2           | W                               |
+-------------+---------------------------------+

******** Table: InternationalBook *******
+-------------+---------------------------------+-------------+
|      ID     |            LANGUAGES            |    REGION   |
+-------------+---------------------------------+-------------+
| 3           | S                               | 4           |
+-------------+---------------------------------+-------------+
```

Common fields are maintained in the parent table

ID of the child table points to the row of the the parent table.

Mapping Inheritance: 1 Table for Concrete Class

One Table per Concrete Class

- Map each of the concrete classes as normal persistent class
- Pros
  - Easiest to implement
- Cons
  - Data belonging to a parent class is scattered across a number of different tables, which represent concrete classes
  - A query couched in terms of parent class is likely to cause a large number of select operations
  - Changes to a parent class can touch large number of tables
  - This scheme is not recommended for most cases

One Table per Concrete Class

- How to define the mapping
  - The mapping of the subclass repeats the properties of the parent class

One Table per Concrete Class

```
<hibernate-mapping>
    <class name="Book" table="cd" discriminator-value="cd">
        <id name="id" type="integer" unsaved-value="0">
            <generator  class="increment"/>
        </id>
        <property name="title"/>
        <property name="artist"/>
        <property name="purchasedate" type="date"/>
        <property name="cost" type="double"/>
    </class>

    <class  name="SpecialEditionBook" table="secd">
        <id name="id" type="integer" unsaved-value="0">
            <generator  class="increment"/>
        </id>
        <property name="title"/>
        <property name="artist"/>
        <property name="purchasedate" type="date"/>
        <property name="cost" type="double"/>
        <property name="newfeatures" type="string"/>
    </class>


</hibernate-mapping>
```

The mapping of the subclass repeats the properties of the parent class

One Table per Concrete Class

```
******** Table: Book *******
+-------------+----------------------+----------------------+--------------+-------------------------+
|      ID     |         TITLE        |        ARTIST        | PURCHASEDATE |          COST           |
+-------------+----------------------+----------------------+--------------+-------------------------+
| 1           | Book                 | R                    | 2008-04-11   | 9.99                    |
+-------------+----------------------+----------------------+--------------+-------------------------+

******** Table: SpecialEditionBook *******
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+
|      ID     |         TITLE        |        ARTIST        | PURCHASEDATE |          COST           |      NEWFEATURES     |
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+
| 1           | sBook                | R                    | 2008-04-11   | 9.99                    | W                    |
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+

******** Table: InternationalBook *******
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+-------------+
|      ID     |         TITLE        |        ARTIST        | PURCHASEDATE |          COST           |       LANGUAGES      |    REGION   |
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+-------------+
| 1           | IBook                | R                    | 2008-04-11   | 9.99                    | S                    | 4           |
| 2           | IBook                | R                    | 2008-04-11   | 100.9                   | T                    | 3           |
+-------------+----------------------+----------------------+--------------+-------------------------+----------------------+-------------+
```

Lab:

Exercise 4: Object Inheritance Relationship
3516_hibernate_mapping.zip

Code with Passion!