Spring Data - II

Teaching Faculty: Umur INAN

Prepared by Muhyieddin **AL-TARAWNEH**, Umur **INAN**

Parent-Child operations (Cascade Types)

- PERSIST
  - Cascading calls to EntityManager.persist() - persists children.
- MERGE
  - Cascading calls to EntityManager.merge() - updates children.
- ALL
  - Shortcut for cascade={PERSIST, MERGE, REMOVE, REFRESH}

Hibernate FETCH Strategies

- Select
- Join
- Subselect
- Batch

Hibernate FETCH Strategy -- SELECT

- Default
- N+1 Fetches
- a second SELECT [per parent N] is used to retrieve the associated collection.
- @Fetch(FetchMode.SELECT)
- To Be Avoided

Hibernate FETCH Strategy -- JOIN

- associated collections are retrieved in the same SELECT.
- uses an OUTER JOIN.
- 1 Fetch
- EAGER
- @Fetch(FetchMode.JOIN)
- Cartesian – need to watch collection sizes; can be useful strategy.

Hibernate FETCH Strategy -- JOIN

- ALWAYS Causes an EAGER fetch of the child collections.
- This is because the characteristic of a Join is ONE fetch Parent & Child TOGETHER
- This can only be accomplished by loading the child collection when the parent is fetched [~= EAGER fetch]
- It can be implemented Manually to use LAZY initialization.

Hibernate FETCH Strategy -- subselect

- a second SELECT is used to retrieve the associated collections for all entities retrieved in a previous query or fetch
- 2 Fetches
  - ORM will do ONE Fetch for All Parents
  - ORM will do ONE Fetch for All child collections

Hibernate FETCH Strategy -- SUBSELECT

- @Fetch(FetchMode.SUBSELECT)
- depends on the "parent" query. If parent Query is complex, it could have performance impacts.
- If fetch=FetchType.LAZY need to "hydrate" children

Hibernate FETCH Strategy -- batch

- Optimization of Select Fetching
- Associated collections are fetched according to declared Batch Size(N/Batch Size) + 1
- @BatchSize(size=n)
- Batch fetching is often called a blind-guess optimization
- # of Fetches “unknown“ UNLESS size of parent is constant

INHERITANCE

- Single Table
- Joined Tables [Table Per Subclass]
- Table per Class

Inheritance – single table

- Contains all columns for Super Class & ALL Sub Classes
- De-normalized schema
- Efficient queries
- Difficult to maintain as the number of columns increase.
- Fast polymorphic queries

Inheritance – joined table

- Table per Subclass
- Normalized schema
- Similar to OO classes
- Less efficient queries.
- Effective if hierarchy isn't too deep.
- Good if following GoF Patterns

Inheritance – table Per class

- Super Class is replicated in each subclass table.
- Uses UNION instead of JOIN.
- All needed columns in each table.

Embeddable and EmbeddedId

- To represent composite keys in JPA entities.

- Composite primary keys are keys that use more than one column to identify a row in the table uniquely.

Embeddable and EmbeddedId

```
@Embeddable
public class MyCompositeKey
      implements Serializable {

    private String key1;
    private String key2;

}
```

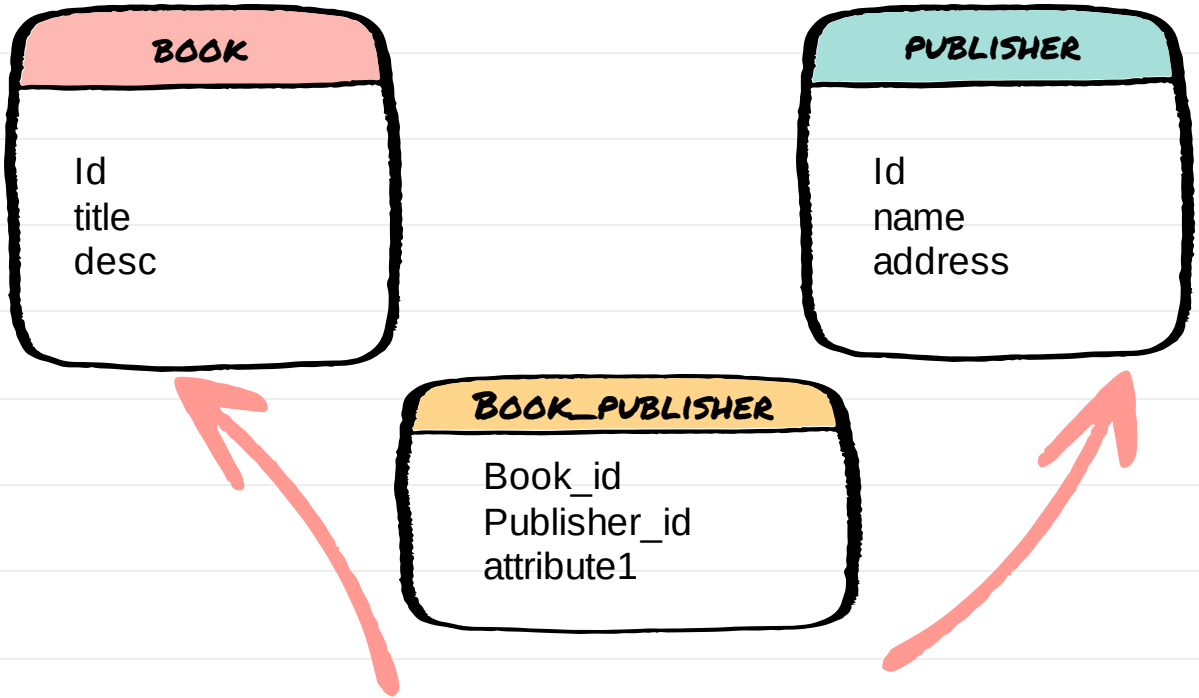
```
@Entity
public class MyEntity {

   @EmbeddedId
    private MyCompositeKey id;

// .... other attributes ...

}
```

Many-to-Many Association with additional Attributes



```java
@Entity
public class BookPublisher{

   @EmbeddedId
    private BookPublisherId id;

 @ManyToOne
   @MapsId("bookId")
   private Book book;

 @ManyToOne
   @MapsId("publisherId")
   private Publisher publisher;


Private String attribute1;


}
```

```java
@Embeddable
public class BookPublisherId
      implements Serializable {

   private Long bookId;
   private Long publisherId;


}
```

```
@Entity
public class Bookimplements
      Serializable {

   @Id
   @GeneratedValue
   private Long id;

   @OneToMany(mappedBy =
     "publisher")
   private Set<BookPublisher>
     bookPublishers = new HashSet<>();


}
```

```
@Entity
public class Publisher {

   @Id
   @GeneratedValue
   private Long id;

   @OneToMany(mappedBy =
     "publisher")
   private Set<BookPublisher>
     bookPublishers = new HashSet<>();

   ...
}
```

Customizing the Result With Spring Data Projection

- Define a Java interface composed of getter methods that match the projected attribute names.

Main points

- Spring provides a Transactional capability for ORM applications.

- The mechanism of transcending allows the individual to tap into Transcendental Consciousness and enlivens its qualities in activity.