# **Advance mapping**

* **In the database , you most likely will have :-**
  + Multiple tables
  + Relationship between Tables
  + Need to model this With Hibernate
* **Types Of Advance Mapping**
  + - One-to-One
    - One-to-Many
    - Many-to-One
    - Many-to-Many

# **One-to-One Mapping**

* **An Instructor can have an “instructor detail” entity**
* **Similar to an “instructor Profile”**

Instructor Detail

**Instructor**

**Id int(11)**

**Youtube\_channel varchar(40)**

**Hobby varchar(30)**

**Id int(11)**

**First\_name varchar(20)**

**Last\_name varchar(20)**

**Email varchar(20)**

**Instructor\_Detail\_id int(11)**

# **One-to-Many Mapping**

* **An instructor can have many courses.**

**Course**

**Course**

**Course**

**Course**

**Course**

**Instructor**

# **Note:-**

**The Actual opposite or inverse of the “One-to-Many” is Many-to-One Mapping.**

# **Many-to-Many Mapping**

* **A course can have many Students.**
* **A Student can have many courses**.

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

# **Important Database Concept:-**

* **Primary key and Foreign Key**
* **Cascade**

**Primary key and Foreign Key**

* **Primary Key :**

**Identify a unique row in a table.**

* **Foreign Key:**
  + **Link tables together.**
  + **A field in one table that refers to primary key in another table.**

**Note: - (foreign key)**

* **Main purpose is to preserve between tables**
* **Referential Integrity**
* **Prevents operations that would destroy relationship.**
* **Ensure only valid data is inserted into the foreign key column.**
* **Can only contain valid reference to primary key in other table.**

**Foreign Key Example:-**

Foreign-Key

**Table: Instructor**

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **First\_name** | **Last\_name** | **Instructor\_Detail\_id** |
| **1** | **Raushan** | **Kumar** | **100** |
| **2** | **Moyn** | **Razvi** | **200** |

|  |  |  |
| --- | --- | --- |
| **Id** | **Youtube\_channel** | **hobby** |
| **100** | **Raushan\_Singh** | **Programming** |
| **200** | **Moyn\_Razvi** | **Debugging** |

**Table: Instructor detail**

# **Cascade**

* **You can cascade Operations.**
* **Apply the same Operation to related entities.**

Instructor detail

**Instructor**

**Note:-**

* **If I perform a save Operation on instructor it does the same operation on instructor detail.**
* **If we delete an instructor, we should also delete their instructor\_detail.**

**(If they no longer have an account, then why should we keep their detail around? )**

* **This is also known as “Cascade Delete”**

Foreign-Key

Column

# **Example: CASCADE DELETE**

**Table: Instructor**

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **First\_name** | **Last\_name** | **Instructor\_Detail\_id** |
| **1** | **Raushan** | **Kumar** | **100** |
| **2** | **Moyn** | **Razvi** | **200** |

|  |  |  |
| --- | --- | --- |
| **id** | **Youtube\_channel** | **hobby** |
| **100** | **Raushan\_Singh** | **Programming** |
| **200** | **Moyn\_Razvi** | **Debugging** |

**Table: Instructor detail**

# **Cascade Delete**

* **Cascade delete depends on the use case.**

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

**Student**

**Course**

* Should we do Cascade delete here?

Ans: - No Way (if we delete Student then we can’t delete the courses)

# **Fetch Types: Eager vs Lazy Loading**

* **When we fetch / Retrieve data, should we retrieve EVERYTHING?**
* **Eager** will Retrieve Everything.(in one Shot)
* **Lazy** Will Retrieve on Request.

**Course**

**Course**

**Course**

**Course**

**Course**

**Instructor**

# **Uni-Directional**

Instructor Detail

**Instructor**

Note:-

* You Start with Instructor object (load the instructor) and then from there, you can access the instructor detail.
* This is One-Way Relationship. (Uni-Directional)

# **Bi-Directional**

Instructor Detail

**Instructor**

Note:-

* Here we have instructor and then they have the relationship with the instructor detail, but then we can also go to the other way. So we can load the instructor detail and have reference to the given instructor.
* This is bi-Directional.(Two-Way –Relationship)