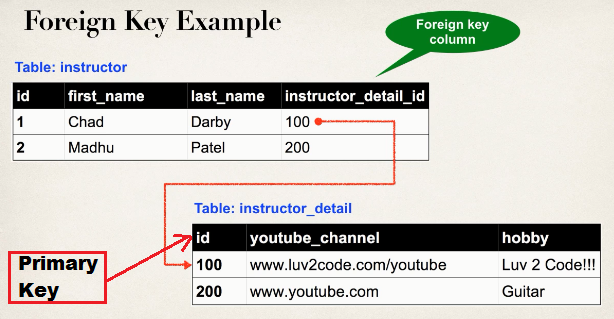
**22.2. Important Database Concepts**

**Primary key**:

A primary key is basically is an identifier. It identifies a unique row in a table. In a work primary key is a unique identifier for each row in a table.

**Foreign Key**:

We use foreign key to link tables together. It’s a field on one table that refers to the primary key in another table.



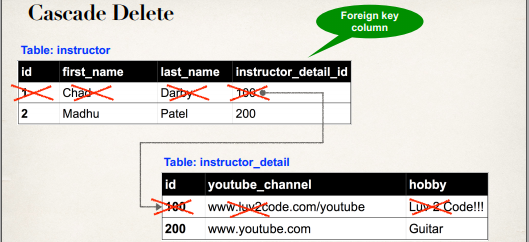
**Cascading in hibernate**:

Cascade basically means we can apply the same operation to related entities.

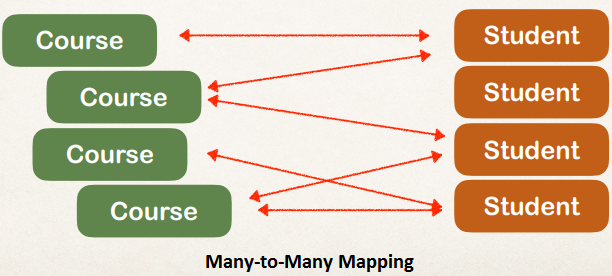
**Example**:

Suppose we have two tables, **instructor** and **instructor\_details**. If we save the **instructor** table then it will cascade the operation and apply the same operation to **instructor\_details** table. This is cascading.

If we delete an **instructor**, we should also delete their **instructor\_detail**. This is known as “**CASCADE DELETE**”.



Cascade delete depends on use case. We have to be careful here when we run the operation CASCADE\_DELETE. Look up the example bellow where we have a relation many-to-many between two table **student** and **course**.



If we delete the student, we also delete the course. But in this case when we delete the student, we should not delete the course.

**Cascade Types**:

|  |  |
| --- | --- |
| **Cascade Operations** | **Description** |
| **CascadeType.PERSIST** | cascade type persist means that save() or persist() operations cascade to related entities. |
| **CascadeType.MERGE** | cascade type merge means that related entities are merged when the owning entity is merged. |
| **CascadeType.REFRESH** | cascade type refresh does the same thing for the refresh() operation. |
| **CascadeType.REMOVE** | cascade type remove removes all related entities association with this setting when the owning entity is deleted. |
| **CascadeType.DETACH** | cascade type detach detaches all related entities if a “manual detach” occurs. |
| **CascadeType.ALL** | cascade type all is shorthand for all of the above cascade operations. |
| **orphanRemoval = true** | removes an owned object from the database when it’s removed from its owning relationship. |

**Face types Eager vs Lazy Loading**:

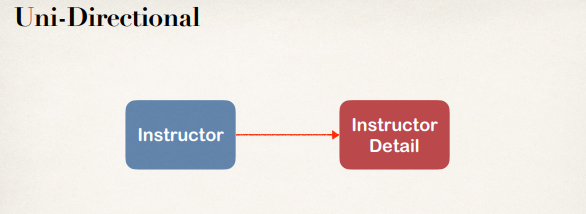
When we face/retrieve data should we retrieve everything?

* Eager will retrieve everything in one short
* Lazy will retrieve the data on request

Link: <https://howtodoinjava.com/hibernate/hibernate-jpa-cascade-types/>

**Uni-Directional**:

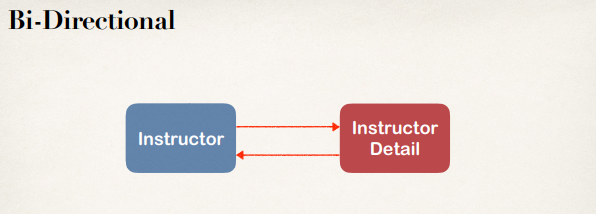
Here is example of Uni-directional relationship. Suppose we have an **instructor table** and relationship with **instructor\_detail** table. We start with the instructor object, we load the **instructor**, and then from there we can access the **instructor\_detail**. So, it’s really a one-way relationship. So that’s uni-directional.



**Bi-Directional**:

Here is example of Bi-directional relationship. Suppose we have an **instructor** table and relationship with **instructor\_detail** table. We start with the instructor object, we load the **instructor**, and then from there we can access the **instructor\_detail**.

We can also go the other way. We can load the **instructor\_detail** and have a reference to the given **instructor**. So, it’s really a two-way relationship. So that’s Bi-directional.



22.2. Important Database Concepts