* **JOINS IN JPA CRITERIA BUILDER :**

1. **Multiple Joins :**

* **Get Vehicle Information table in Multiple join . Like Vehicle Information Primary key is VEH\_CODE but we are not Using VEH\_CODE in where n condition . We will use VEH\_CODe as Foreign key in Policy Information , Drivert\_Information table . Then we will Use Policy Information TABLE POL\_INFO\_ID And Driver Information LICENCE\_NO to search vehicle Data .**
* **if same VEH\_CODE is available in both the table it will Directly provide the Vehicle Information List .**
* **First We Want to Combine the Three Entity Together Using Entity Relationship. We will use @OneToMany @ManyToOne Annotation for that .**

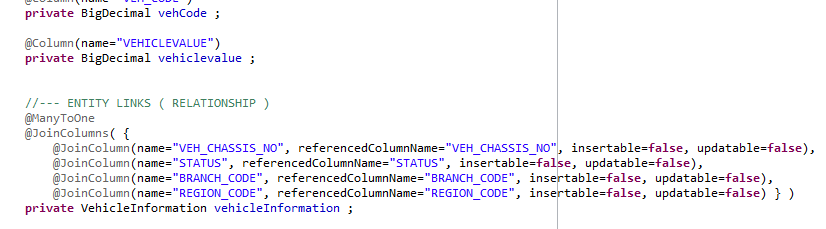
**VehicleInformation.java (Parent Relation – One To Many)**

****

**DriverInformation.java (Child Relation – Many To One )**

****

**PolicyInformation.java (Child Relation – Many to One )**

****

**Criteria Query For Multi Joins :**

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<VehicleInformation> query = cb.createQuery(VehicleInformation.**class**);

Root<VehicleInformation> veh = query.from(VehicleInformation.**class**);

Join<VehicleInformation, DriverInformation> dri = veh.join("listOfDriverInformation");

Join<VehicleInformation, PolicyInformation> pol = veh.join("listOfPolicyInformation");

query.select( veh );

Predicate n1 = cb.equal(dri.get("licenceNo"), "100045273");

Predicate n2 = cb.equal(pol.get("polInfoId"), "10277");

List<Order> orderList = **new** ArrayList<Order>();

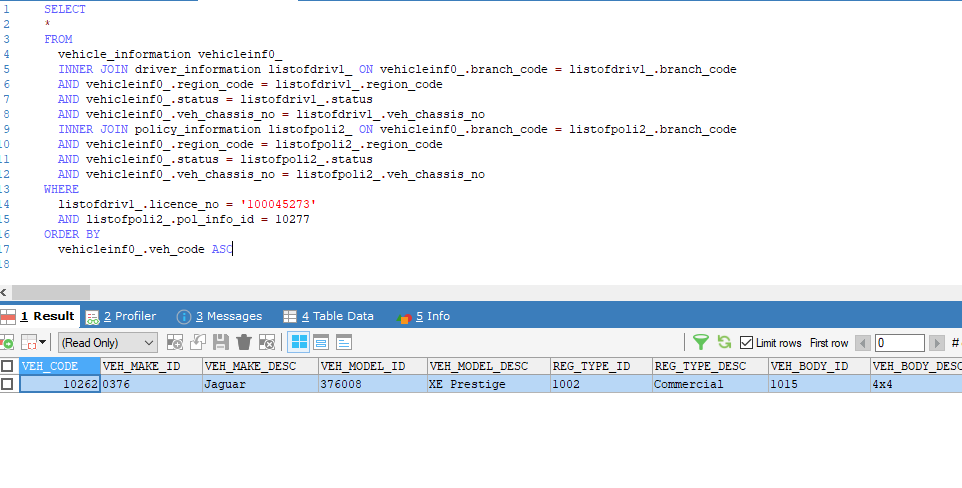
orderList.add(cb.asc(veh.get("vehCode")));

query.where(n1,n2).orderBy(orderList);

TypedQuery<VehicleInformation> res1 = em.createQuery(query);

List<VehicleInformation> list = res1.getResultList();

**Hibernate Running Query & Output :**



1. **Inner Join (Vehicle & Driver Having Without Empty Data will come )**

**Criteria Query**

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Tuple> query = cb.createQuery(Tuple.**class**);

Root<VehicleInformation> veh = query.from(VehicleInformation.**class**);

Join<VehicleInformation, DriverInformation> dri = veh.join("listOfDriverInformation" , JoinType.***INNER***);

query.multiselect( veh.get("vehCode").alias("vehCode"), dri.get("licenceNo").alias("licenceNo") );

Predicate n1 = cb.equal(veh.get("status"), "Y");

List<Order> orderList = **new** ArrayList<Order>();

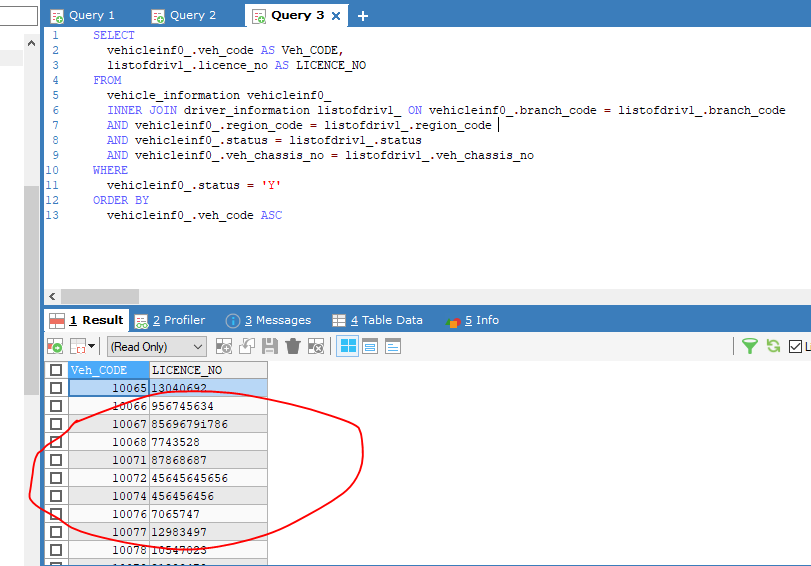
orderList.add(cb.asc(veh.get("vehCode")));

query.where(n1).orderBy(orderList);

TypedQuery<Tuple> res1 = em.createQuery(query);

List<Tuple> list = res1.getResultList();

**Hibernate Running Query & Output :**

****

1. **Left Join (Vehicle (Left Side ) having data will come . if Driver (Right Side ) table the data not available it will give columns as null )**
2. **Right Join (Same Query Driver ( Right Side) Having data will come . If Vehicle (Left Side ) not available it will give columns as null )**

**Criteria Query**

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Tuple> query = cb.createQuery(Tuple.**class**);

Root<VehicleInformation> veh = query.from(VehicleInformation.**class**);

Join<VehicleInformation, DriverInformation> dri = veh.join("listOfDriverInformation" , JoinType.***LEFT***);

query.multiselect( veh.get("vehCode").alias("vehCode"), dri.get("licenceNo").alias("licenceNo") );

Predicate n1 = cb.equal(veh.get("status"), "Y");

List<Order> orderList = **new** ArrayList<Order>();

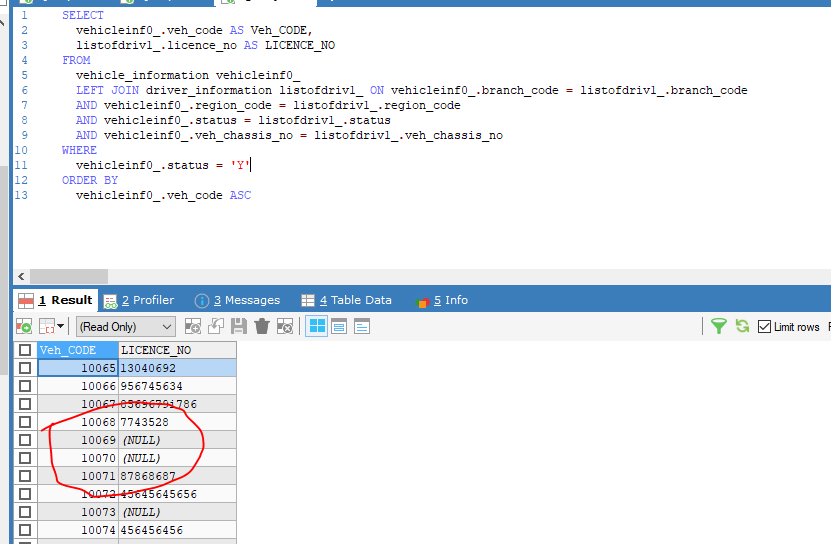
orderList.add(cb.asc(veh.get("vehCode")));

query.where(n1).orderBy(orderList);

TypedQuery<Tuple> res1 = em.createQuery(query);

List<Tuple> list = res1.getResultList();

**Hibernate Running Query & Output :**

****

* **Union & Union All in Criteria Builder**

1. **Union All ( Getting Same Column value from both table with duplicates . Its called Union All )**

**Criteria Query :**

We will use two Induvidual queries and add the two result in one list it will same as union all query .

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Tuple> query = cb.createQuery(Tuple.**class**);

Root<VehicleInformation> veh = query.from(VehicleInformation.**class**);

// Result 1 Query

Predicate n1 = cb.equal(veh.get("status"), "Y");

List<Order> orderList = **new** ArrayList<Order>();

orderList.add(cb.asc(veh.get("vehCode")));

query.multiselect( veh.get("vehCode").alias("vehCode"), veh.get("vehChassisNo").alias("vehChassisNo")) ;

query.where(n1).orderBy(orderList);

TypedQuery<Tuple> res1 = em.createQuery(query);

List<Tuple> result1 = res1.getResultList();

// Result 2 Quey

Root<DriverInformation> dri = query.from(DriverInformation.**class**);

Predicate n2 = cb.equal(dri.get("status"), "Y");

List<Order> orderList2 = **new** ArrayList<Order>();

orderList2.add(cb.asc(dri.get("vehCode")));

query.multiselect( dri.get("vehCode").alias("vehCode"), dri.get("vehChassisNo").alias("vehChassisNo")) ;

query.where(n2).orderBy(orderList);

TypedQuery<Tuple> res2 = em.createQuery(query);

List<Tuple> result2 = res2.getResultList();

// Union All (Combining Two Result )

result1.addAll(result2);

List<Tuple> unionAll = result1 ;

1. **Union ( Getting Same Column value from both table with Distinct . Its called Union )**

**Criteria Query :**

We will use same Union All Criteria Query For this. after that we will filter remove the Duplicate values using distinct by method

// Which are the duplicate key you don’t want you want to mention here

List<Tuple> union = unionAll.stream().filter(*distinctByKey*(o -> Arrays.*asList*( o.get("vehCode").toString() , o.get("vehChassisNo").toString() ) )).collect(Collectors.*toList*());

// Distinct By Method

**private** **static** <T> java.util.function.Predicate<T> distinctByKey(java.util.function.Function<? **super** T, ?> keyExtractor) {

Map<Object, Boolean> seen = **new** ConcurrentHashMap<>();

**return** t -> seen.putIfAbsent(keyExtractor.apply(t), Boolean.***TRUE***) == **null**;

}

* **Group By in Criteria Builder ( It will Group all column values with distinct by row count )**

**Criteria Query :**

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Tuple> query = cb.createQuery(Tuple.**class**);

// Find All

Root<EserviceCustomerDetails> c = query.from(EserviceCustomerDetails.**class**);

Root<EserviceMotorDetails> m = query.from(EserviceMotorDetails.**class**);

// Select

query.multiselect(cb.count(m).alias("idsCount"),

// Customer Info

c.get("customerReferenceNo").alias("customerReferenceNo"), c.get("idNumber").alias("idNumber"),

c.get("clientName").alias("clientName"),

// Vehicle Info

m.get("companyId").alias("companyId"), m.get("productId").alias("productId"),

m.get("branchCode").alias("branchCode"), m.get("requestReferenceNo").alias("requestReferenceNo"));

// Order By

List<Order> orderList = **new** ArrayList<Order>();

orderList.add(cb.desc(m.get("updatedDate")));

// Where

Predicate n1 = cb.equal(c.get("customerReferenceNo"), m.get("customerReferenceNo"));

Predicate n2 = cb.equal(m.get("companyId"), req.getInsuranceId());

Predicate n3 = cb.equal(m.get("productId"), req.getProductId());

query.where(n1, n2, n3)

.groupBy(c.get("customerReferenceNo"), c.get("idNumber"), c.get("clientName"), m.get("companyId"), m.get("productId"), m.get("branchCode"), m.get("requestReferenceNo"))

.orderBy(orderList);

// Get Result

TypedQuery<Tuple> result = em.createQuery(query);

List<Tuple> list = result.getResultList();