Native SQL İle Sorgulama

harezmi bilişim çözümleri

Native SQL Sorguları

- Session/EntityManager üzerinden **native SQL** de çalıştırılabilir
- Native SQL sorgu sonucunda istenirse **List of entity**, istenirse de **List of Object[]** dönebilir
- Sorgu sonucu dönen entity nesneleri HQL/JPQL'de olduğu gibi **Session/EntityManager ile ilişkili**dir
- Dolayısı ile bu entity nesneler üzerinde yapılan değişiklikler otomatik olarak DB'ye yansır

Native SQL Sorguları ve List of Object[] Dönülmesi

```
session.createSQLQuery("select * from T_PET")
        .list();

session.createSQLQuery("select ID,NAME from T_PET")
        .list();
```

Yukarıdaki native SQL sorguları sonucu List of Object[] dönülecektir. Her bir sütun değeri ResultSetMetadata'ya bakılarak SQL tipinden bir Java nesnesine dönüştürülecektir. Aşağıdaki yöntemle ResultSetMetadata'ya bakmadan her bir sütunun hangi Java tipine karşılık geldiği de belirtilebilir

```
session.createSQLQuery("select * from T_SPECIALTY")
        .addScalar("ID",Hibernate.LONG)
        .addScalar("NAME",Hibernate.STRING)
        .addScalar("BIRTH_DATE",Hibernate.DATE)
        .list();
```

Native SQL Sorguları ve Persistent Entity Dönülmesi

```
session.createSQLQuery("select * from T_PET")
        .addEntity(Pet.class)
        .list();
```

Sorgu sonucu dönen değerlerin entity nesnelere dönüştürülmesi de mümkündür. Dönüşüm sırasında sütun isimleri ile property'ler arasındaki eşleştirme metadata üzerinden sağlanır.

SQL sorgusu içerisinde JOIN yapmak ve parametreler tanımlamak da mümkündür. Sorgu sonucu dönen sütunlar {p.*} şeklinde placeholder alias ile yine spesifik entity nesneye dönüştürülebilir.

```
session.createSQLQuery("select {p.*} from T_PET p"
           + " join T_PET_TYPE t on p.TYPE_ID = t.ID"
           + " where t.NAME = :name")
        .addEntity("p",Pet.class)
        .setParameter("name", "Kedi")
        .list();
```

Native SQL Sorguları ve ResultTransformer

```
session.createSQLQuery("select {p.*}, {v.*} from T_PET p,
T_VISIT v where p.ID = v.PET_ID and p.NAME = ?")
        .addEntity("p",Pet.class)
        .addEntity("v",Visit.class)
        .setParameter(0,"Maviş")
        .list();
```

Sorgu sonucu birden fazla entity ile de eşleştirilebilir. Yukarıdaki sorgu da List of Object[] Dönülecek, Object[]'in ilk elemanı Pet, ikinci elemanı ise Visit nesneleri olacaktır.

İstenirse bunlardan herhangi birine ResultTransformer ile project edilebilir.

```
session.createSQLQuery("select {p.*}, {v.*} from T_PET p,
T_VISIT v where p.ID = v.PET_ID")
      .addEntity("v",Visit.class)
      .addEntity("p",Pet.class)
   .setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY)
      .list();
```

Native SQL Sorguları ve Eager Fetch

```
session.createSQLQuery("select {p.*}, {v.*} from T_PET p
left outer join T_VISIT v")
      .addEntity("v",Visit.class)
      .addEntity("p",Pet.class)
      .addJoin("v","p.visits")
      .addEntity("p",Pet.class)
      .setResultTransformer(Criteria.DISTINCT_ROOT_ENTITY)
      .list();
```

Lazy property veya collection ilişkileri EAGER FETCH de yapılabilir.
Burada dikkat edilmesi gereken nokta eğer ResultTransformer kullanılacak ise addJoin nedeni ile root entity olarak Visit dönülecektir!
Dönülecek entity'nin Pet olması isteniyorsa addEntity ile Pet.class tekrar belirtilmelidir

JPA ve Native SQL

- JPA'da **native SQL** ile çalışmak da mümkündür
- JPQL veya criteria API'nin yetersiz kaldığı, DB'ye özel kabiliyetlerin kullanılması veya stored procedure'lerin çağrılması gereken durumlarda native SQL ile çalışmak gerekebilir
- Native SQL sorguları normalde sorgu sonucu olarak **List of Object[]** döner
- Dönen sonuç içerisinde entity'ler, scalar değerler veya her ikisi birlikte de olabilir

JPA ve Native SQL

```
javax.persistence.Query query =
                    entityManager.createNativeQuery(
        "select ID,NAME,BIRTH_DATE from T_PET where NAME like :name");

query.setParameter("name", "%e");

List<Object[]> resultList = query.getResultList();

for (Object[] arr: resultList) {
    System.out.println(ArrayUtils.toString(arr));
}
```

JPA'da Native SQL ile Entity Dönmek

JAVA Eğitimleri

- **Entity nesne** dönülebilmesi için sorguda entity ile ilgili **bütün sütunlar mevcut olmalı**dır ve ayrıca Entity'nin tipi de belirtilmelidir
- Dönülen entity nesneler **persistence context ile ilişkili**dir

```
Query query = entityManager.createNativeQuery(
                "select * from T_PET where NAME like :name",Pet.class);

query.setParameter("name", "%e");

List<Pet> resultList = query.getResultList();

for (Pet pet : resultList) {
    System.out.println(pet);
}
```

JPA'da Native SQL ile Birden Fazla Entity Dönmek

- Eğer native SQL sorgusu birden fazla tipte entity tipi veya scalar değeri birlikte dönüyorsa explicit **resultset mapping metadata**'sına ihtiyaç vardır

```
Query query = entityManager.createNativeQuery(
"select p.ID,p.NAME,p.BIRTH_DATE,p.OWNER_ID,p.TYPE_ID,p.OPT_LOCK_VERSION,"
+ "v.ID,v.VISIT_DATE,v.DESCRIPTION,v.CHECKUP,v.PET_ID,v.OPT_LOCK_VERSION "
+ "from T_PET p left outer join T_VISIT v on p.ID = v.PET_ID where NAME like
:name",
"petsWithVisits");

query.setParameter("name", "%e");

List<Object[]> resultList = query.getResultList();

for (Object[] arr : resultList) {
    Pet p = (Pet) arr[0];
    Visit v = (Visit) arr[1];
    System.out.println(p.getName());
    System.out.println(v.getDescription());
}
```

@SqlResultSetMapping anotasyonu ile ayrıca tanımlanmış olmalıdır

JPA'da Native SQL ve @SqlResultSetMapping

```
@SqlResultSetMapping(name="petsWithVisits",entities={
    @EntityResult(entityClass=Pet.class,fields={
        @FieldResult(column="ID",name="id"),
        @FieldResult(column="NAME",name="name"),
        @FieldResult(column="BIRTH_DATE",name="birthDate"),
        @FieldResult(column="OPT_LOCK_VERSION",name="version"),
        @FieldResult(column="OWNER_ID",name="owner"),
        @FieldResult(column="TYPE_ID",name="type")
    }),
    @EntityResult(entityClass=Visit.class,fields={
        @FieldResult(column="ID",name="id"),
        @FieldResult(column="VISIT_DATE",name="date"),
        @FieldResult(column="DESCRIPTION",name="description"),
        @FieldResult(column="OPT_LOCK_VERSION",name="version"),
        @FieldResult(column="CHECKUP",name="checkup"),
        @FieldResult(column="PET_ID",name="pet")
    })
})
```

Anotasyon yerine **META-INF/orm.xml** içerisinde **<sql-result-set-mapping>** elemanı ile de tanımlanabilir

JPA'da Native SQL ve <sql-result-set-mapping>

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings ...>
    <sql-result-set-mapping name="petsWithVisits">
        <entity-result entity-class="com.javaegitimleri.petclinic.model.Pet">
            <field-result name="id" column="ID" />
            <field-result name="name" column="NAME" />
            <field-result name="birthDate" column="BIRTH_DATE" />
            <field-result name="owner" column="OWNER_ID" />
            <field-result name="type" column="TYPE_ID" />
            <field-result name="version" column="OPT_LOCK_VERSION" />
        </entity-result>
        <entity-result entity-class="com.javaegitimleri.petclinic.model.Visit">
            <field-result name="id" column="ID" />
            <field-result name="date" column="VISIT_DATE" />
            <field-result name="description" column="DESCRIPTION" />
            <field-result name="checkup" column="CHECKUP" />
            <field-result name="pet" column="PET_ID" />
            <field-result name="version" column="OPT_LOCK_VERSION" />
        </entity-result>
    </sql-result-set-mapping>
</entity-mappings>
```

JPA'da Native SQL ile Scalar/Sütun Değerler Dönmek

- Entity nesnelerinin yanı sıra **scalar değer**ler veya tek tek sütun değerleri de döndürülebilir

```
Query query = entityManager.createNativeQuery("select p.ID,p.NAME,p.BIRTH_DATE,"
    + "p.OWNER_ID,p.TYPE_ID,p.OPT_LOCK_VERSION, count(v.ID) as VISIT_COUNT "
    + "from T_PET p left join T_VISIT v on p.ID = v.PET_ID "
    + "group by p.ID,p.NAME,p.BIRTH_DATE,p.OWNER_ID,p.TYPE_ID,p.OPT_LOCK_VERSION",
"petsWithVisitCounts");

List<Object[]> resultList = query.getResultList();
for (Object[] arr : resultList) {
    Pet p = (Pet) arr[0];Long visitCount = (Long) arr[1];
    System.out.println(p.getName() + ":" + visitCount);
}
```

```
@SqlResultSetMapping(name = "petsWithVisitCounts", entities = {
    @EntityResult(entityClass = Pet.class, fields = {
    @FieldResult(column = "ID", name = "id"),
    @FieldResult(column = "NAME", name = "name"),
    @FieldResult(column = "BIRTH_DATE", name = "birthDate"),
    @FieldResult(column = "OPT_LOCK_VERSION", name = "version"),
    @FieldResult(column = "OWNER_ID", name = "owner"),
    @FieldResult(column = "TYPE_ID", name = "type")}
)}, columns = { @ColumnResult(type = Long.class, name = "VISIT_COUNT") })
```

JPA'da Native SQL ve Constructor Result Mapping

- Native SQL sorgu sonucu persistence context ile ilişkili entity dönmek yerine **DTO (value object) dönmek** de mümkündür
- JPQL'deki **constructor expression**'lara benzer

```
Query query = entityManager.createNativeQuery(
    "select p.ID,p.NAME,count(v.ID) as VISIT_COUNT "
    + "from T_PET p left join T_VISIT v on p.ID = v.PET_ID "
    + "group by p.ID,p.NAME", "petVisitInfoList");

List<PetVisitInfo> resultList = query.getResultList();

for (PetVisitInfo petVisitInfo : resultList) {
    System.out.println(
        petVisitInfo.getName() + " :" + petVisitInfo.getVisitCount());
}
```

@SqlResultSetMapping anotasyonu ile ayrıca tanımlanmış olmalıdır

JPA'da Native SQL ve Constructor Result Mapping

JAVA Eğitimleri

```
public class PetVisitInfo {
    private Long id;
    private String name;
    private Long visitCount;

    public PetVisitInfo(Long id, String name, Long visitCount) {
        this.id = id;
        this.name = name;
        this.visitCount = visitCount;
    }
    //getters...
}
```

```
@SqlResultSetMapping(name = "petVisitInfoList", classes={
    @ConstructorResult(targetClass=PetVisitInfo.class,
        columns={
            @ColumnResult(name="ID",type=Long.class),
            @ColumnResult(name="NAME",type=String.class),
            @ColumnResult(name="VISIT_COUNT",type=Long.class)
        })
})
@Entity
@Table(name = "T_PET")
public class Pet {
...
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com