İlişkilerin Lazy/Eager Yüklenmesi

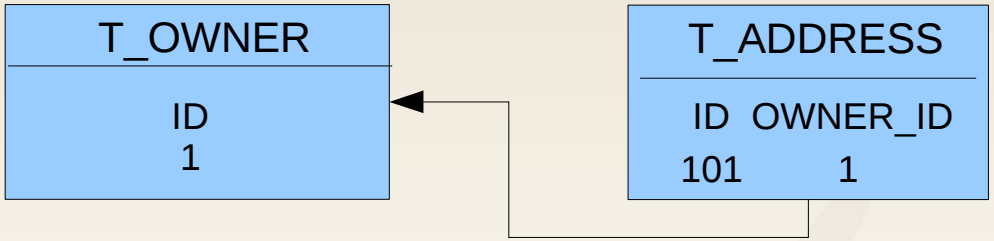
1:1 İlişkiler ve Lazy

```java
@Entity
public class Owner {
    @Id @GeneratedValue
    private Long id;

    @OneToOne(mappedBy="owner",fetch=FetchType.LAZY)
    private Address address;
}

@Entity
public class Address {
    @Id
    @GeneratedValue
    private Long id;

    @OneToOne
    @JoinColumn(name = "OWNER_ID")
    private Owner owner;
}
```



Owner nesnesi yüklenirken address property'si NULL'mı bırakılacak, yoksa bir proxy address nesnesi mi set edilecek bunun kararına varabilmek için T_ADDRESS tablosuna da bakılması gerekir. Dolayısı ile Address bilgisine bu aşamada erişilmiş olacaktır. **optional=false** set edilir ise bu durumda Hibernate ilişkinin zorunlu olduğunu kabul edip bir proxy nesne set edecektir. **Hibernate 5.x JPA uyumluluğu nedeni ile optional=false yapılsa bile LAZY yapmamaktadır.**

1:1 İlişkiler ve Lazy

```
@Entity
public class Owner {
    @Id @GeneratedValue
    private Long id;

    @OneToOne(mappedBy="owner",fetch=FetchType.LAZY)
    @LazyToOne(LazyToOneOption.NO_PROXY)
    private Address address;
}

@Entity
public class Address {
    @Id
    @GeneratedValue
    private Long id;

    @OneToOne
    @JoinColumn(name = "OWNER_ID")
    private Owner owner;
}
```

Çalışması için ayrıca **bytecode enhancement'ın aktive edilmesi** gerekir bytecode enhancement maven ile build zamanında veya JPA managed mod ile çalışırken runtime'da aktive edilebilir

İki Tane Eager Bag Collection Problemi

- Bir entity içerisinde **iki tane EAGER** 1:M veya M:N Bag tanımı yapılamaz

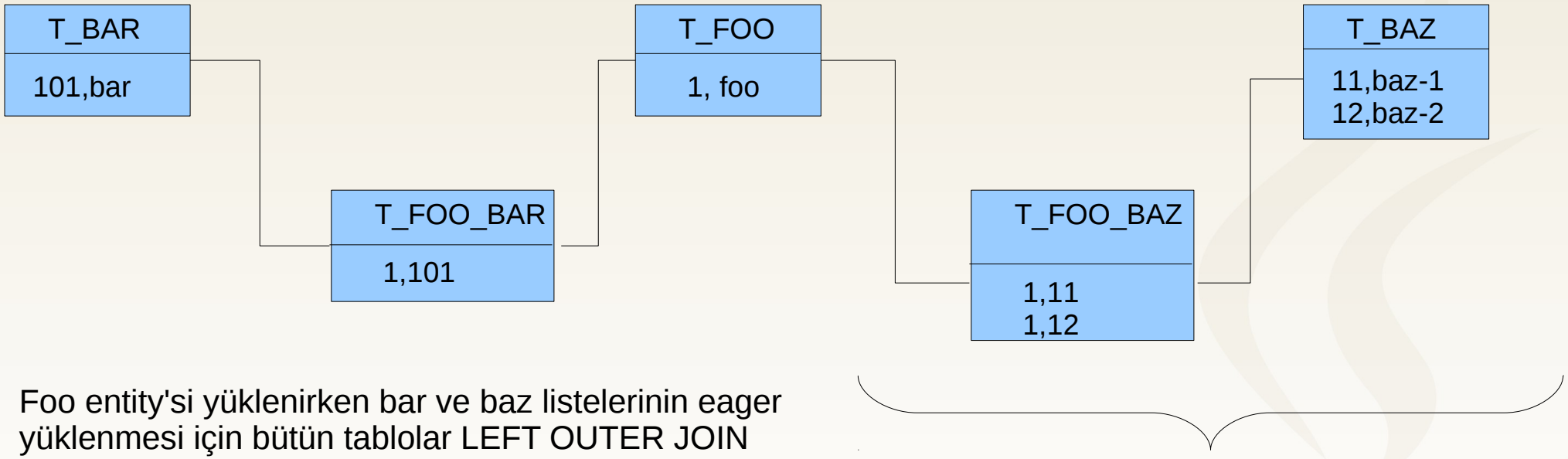
İlişkilerden bir tanesi LAZY tanımlanmak zorundadır!

```
@Entity
public class Foo {

    @OneToMany(fetch = FetchType.EAGER)
    private List<Bar> bars = new ArrayList<Bar>();


    @OneToMany(fetch = FetchType.EAGER)
    private List<Baz> baz = new ArrayList<Baz>();
}
```

İki Tane Eager Bag Collection Problemi



Foo entity'si yüklenirken bar ve baz listelerinin eager yüklenmesi için bütün tablolar LEFT OUTER JOIN yapılmaktadır.

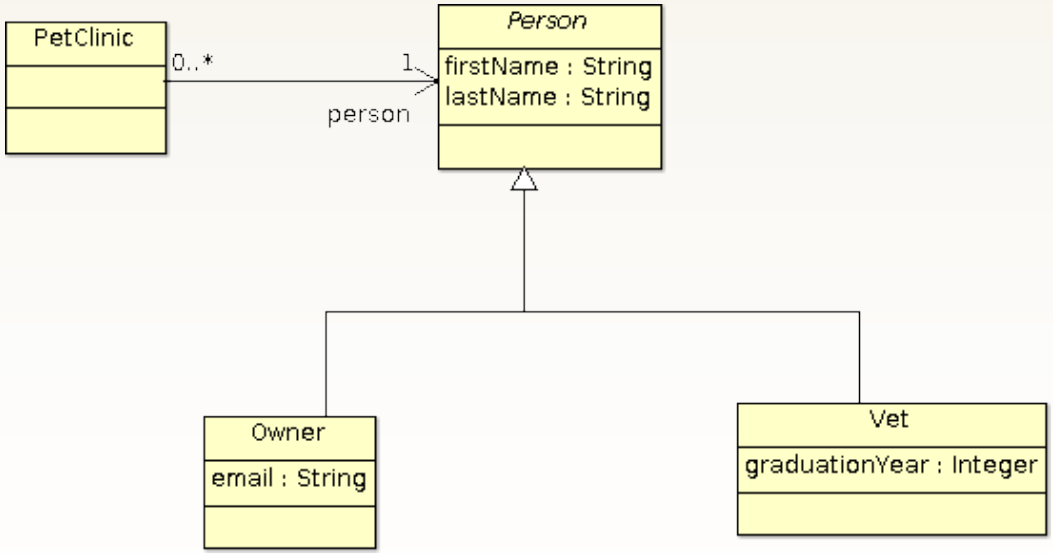
Tanımı gereği bag collection ilişkisinde duplike kayıtlar olabilir. Örnekte de görüldüğü üzere duplikasyon join sırasında da oluşabilir.

Dolayısı ile duplikasyonun, join işlemi sonucu mu, yoksa bag collection içerisindeki duplikasyondan mı kaynaklandığı bilinemediği için Hibernate iki eager bag collection mapping'e izin vermemektedir

1,foo,1,101,bar,1,11,baz-1
1,foo,1,101,bar,1,12,baz-2

Polymorphic M:1/1:1 Lazy İlişkiler

- **M:1** veya **1:1** ilişkiler **lazy** tanımlanırsa, **proxy dönülebileceği için** polymorphic ilişkinin hedef nesne tipi **instanceof** operatörü ile tespit edilemez




```
@Entity
public class PetClinic extends BaseEntity {

    @ManyToOne(fetch=FetchType.LAZY)
    private Person person;

    public Person getPerson() {
        return person;
    }

    public void setPerson(Person person) {
        this.person = person;
    }
}
```

Polymorphic M:1/1:1 Lazy İlişkiler

JAVA Eğitimleri

```
Petclinic petclinic = session.get(Petclinic.class,1L);
Person personProxy = petclinic.getPerson();
```

Hata!

```
System.out.println(personProxy instanceof Owner);
System.out.println(personProxy instanceof Vet);
```

→ Her ikiside **false** dönecektir.

```
System.out.println(Hibernate.getClass(personProxy).isAssignableFrom(Owner.class));
System.out.println(Hibernate.getClass(personProxy).isAssignableFrom(Vet.class));
```

Çalışır!

instanceof yerine Hibernate'in yardımcı sınıfı kullanılabilir. Bu işlemin yan etkisi olarak proxy initialize edilecektir

İletişim

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