Web Programlama II

Ders 02: JPA

Bu Sunum Hakkında

- Bu sunumlar sadece üst seviyeden kısa bir özettir
- Detaylar doğrudan kod içerisinde yorum satırı halinde verilmiştir

İlişkiler

Veritabanı (DB) tabloları arasında ilişkiler olabilir

- Bir category birden fazla subcategory sahibi olabilir
- Bir subcategory yalnızca bir parent category'e sahiptir
- Benzer türde bir ilişki SubCategory ve Quiz arasında da vardır
- "Bağlantılar" *foreignkey* kısıtlamalarıdır

Relationship Annotations

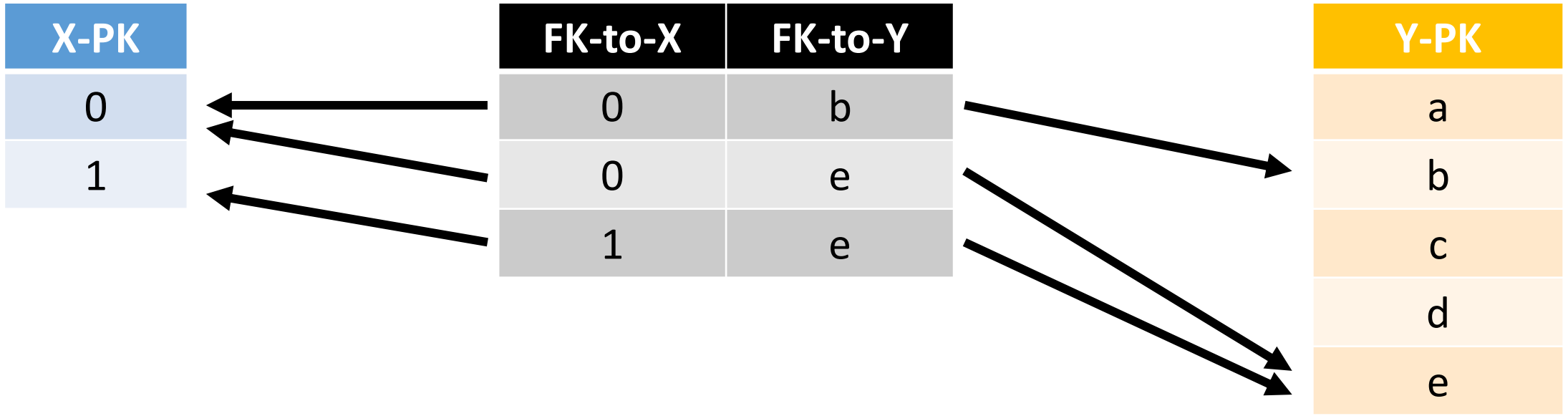
- 5 farklı türde annotation bulunmaktadır
- *@OneToOne*
- *@OneToMany*
- *@ManyToOne*
- *@ManyToMany*
- *@ElementCollection*

Bağlantılar (Links)

- “Links”, SQL *foreign-key* kısıtları ile temsil edilmektedir
  - ör, Bir tablodaki X alanı, bir başka tablodaki primary-key olan Y alanını işaret edebilir
- Eğer X ile Y arasında bağlantı varsa çift yönlü mü yoksa tek yönlü mü olduğu belirlenmelidir
  - ör, *unidirectional* veya *bidirectional* links
- @ annotasyonu *mappedBy* ile birlikte belirtilir
  - Eğer unutulursa bağımsız bağlantılar oluşabilir

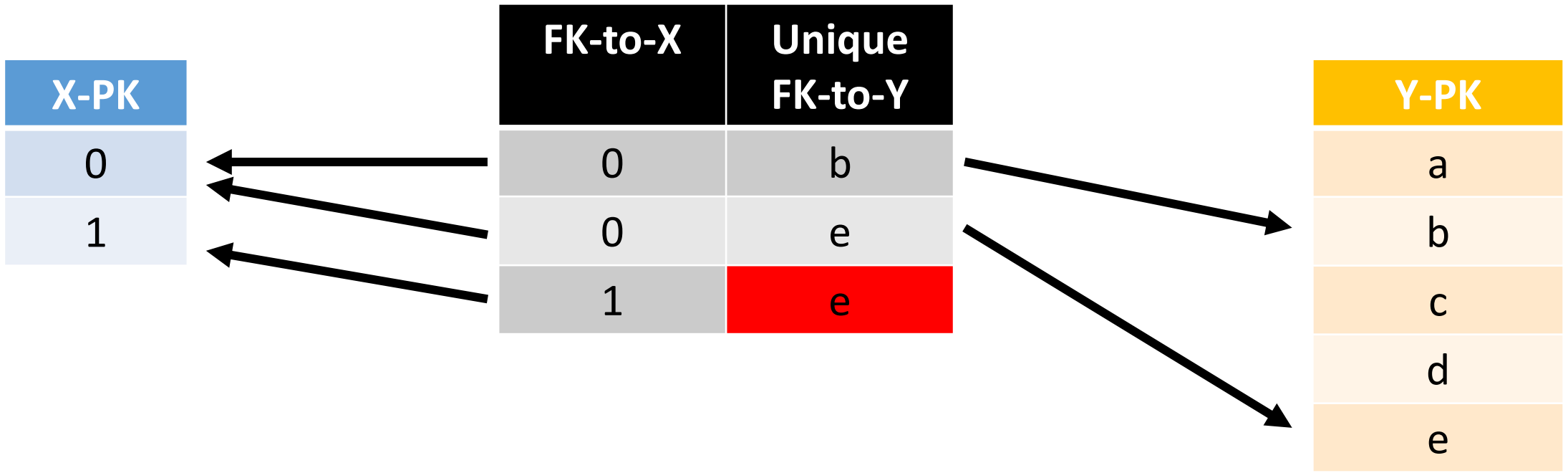
Many-to-Many

- Eğer bir X elementinin çoklu Y ile bağlantısı varsa ve Y'nin de çoklu X bağlantısı varsa 3. bir 2 FK barındıran tabloya ihtiyaç bulunur
- Ör, *0, b* ve *e* ile bağlı iken *e*'de *1* ile bağlı
- Unidirectional ve bidirectional aynı SQL tablosunu kullanır



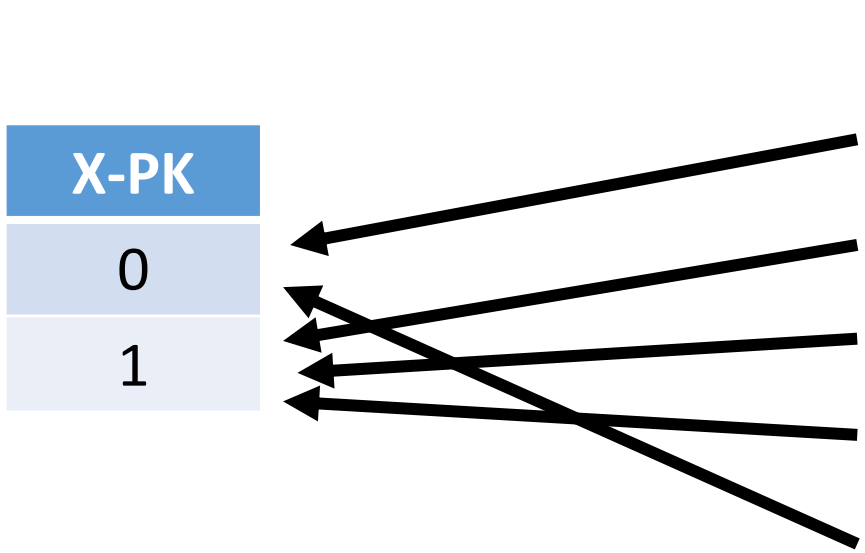
1-to-Many Unidirectional (Tek yönlü)

- Eğer bir X elemanının Y elemanına çoklu bağlantsı varsa ve Y'nin geri yönlü bir bağlantısı yoksa
- Many-to-Many'e benzer ancak FK-to-Y olacak şekilde **UNIQUE** bir kısıt bulunmaktadır.
- Ör, Eğer 0 elemanının *b* ve *e*'ye bağlantısı varsa *e* diğer ilişkilerde kullanılamaz



1-To-Many Bidirectional (çift yönlü)

- Bir X’in birden fazla Y bağlantısı varsa, FK Y’den X’e olacaktır
- X tablosunun Y hakkında bilgisi yoktur, ancak belirli bir X x'e işaret eden FK'ye sahip Y'deki tüm satırları bulmak için SQL kullanabilir
- *Tek yönlü olmasındansa çift yönlü olması daha iyidir.*



X-PK
0
1

Y-PK	Y-FK-to-X
a	0
b	1
c	1
d	1
e	0

EntityManager

Varlıkları DB'deki verilerle eşitlemek için kullanılan nesnedir

Farklı operasyonlar için kullanılabilir

- *persist()*
- *clear()*
- *find()*
- *contains()*
- *merge()*
- *remove()*
- vs.

Java Persistence query language (JPQL)

- *EntityManager#find(id)* ile verilen *id* değerine ait *@Entity'i* getiren sorgu oluşturulur
- Ancak belli bir kategorideki quiz'leri getiren bir sorguya ihtiyaç duysaydınız ne olacaktı?
- Tabii ki de SQL JPQL ile SQL de kullanabilirsiniz
- JPQL: SQL syntax'ına benzerdir ancak doğrudan veritabanı ile değil *@Entity* nesnesini işaret eder
- JPA, JPQL'i çalışma zamanında SQL'e çevirir

JPQL Örnek

```
select u from User u where u.address.country = ‘Turkey'
```

- SQL’e oldukça benzerdir, ör, SELECT/FROM/WHERE kullanırken
- Ancak, bir *@Entity*’i işaret eder
- Entity’e bir isim verilmektedir, ör “*u*” (ancak herhangi bir şey de olabilir)
  - Bu sayede nesnenin alanlarına erişmeye çalıştığımız bir @Entity örneği olacaktır

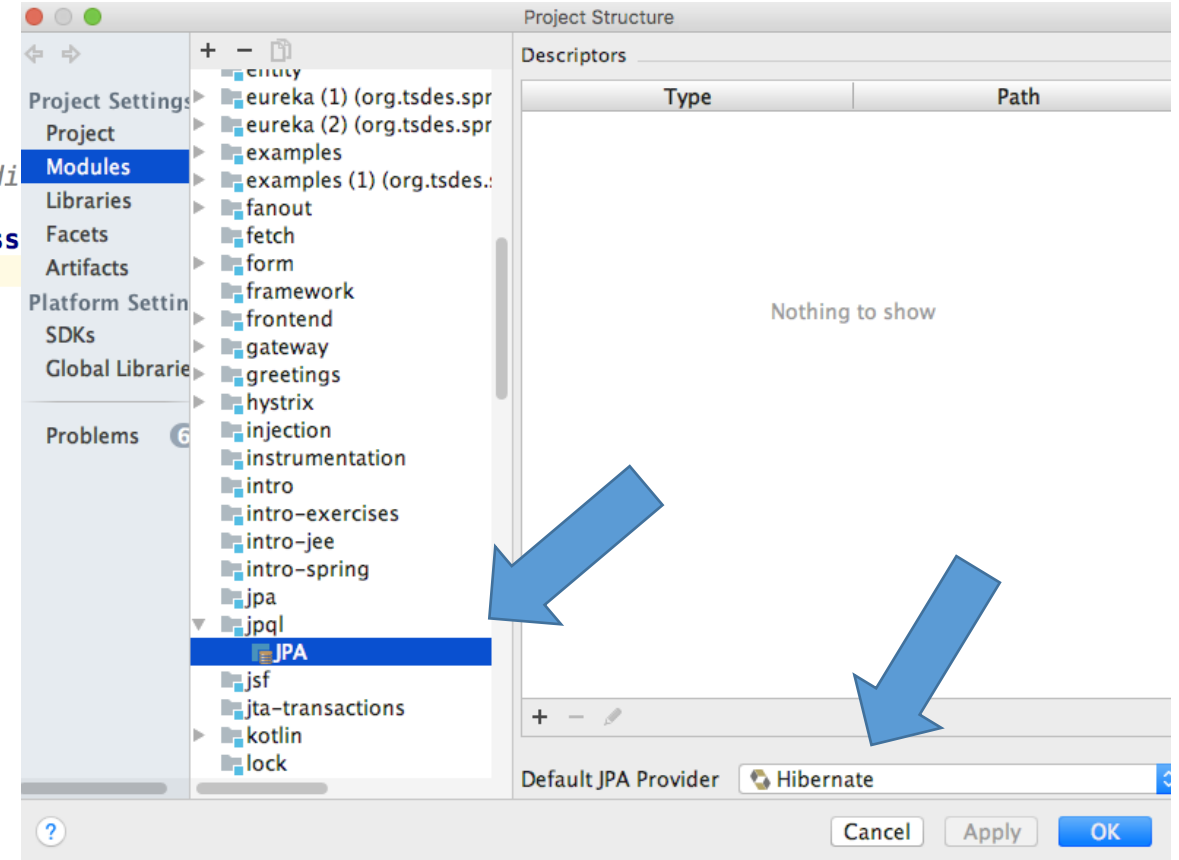
JPQL ve IntelliJ

IntelliJ otomatik olarak syntax'ı analiz eder ve JPQL ifadelerini tamamlayabilir ancak ayar yapmak gerekmektedir

```
@Test
public void testGetAllWithOnTheFlyQuery() {

    //you can create queries on the fly. but if a query is used in a lot of di
    //places, it might be best to use a named one
    TypedQuery<User> query = em.createQuery("select u from User u", User.class
    List<User> users = query.getResultList();

    assertEquals(4, users.size());
}
```



Git Repository Modülleri

- *Not: açıklamaların çoğu slaytlarda değil kod içerisinde yorum satırı olarak bulunmaktadır*
- **intro/jee/jpa/relationship**
- **intro/jee/jpa/relationship-sql**
- **intro/jee/jpa/manager**
- **intro/jee/jpa/jpql**
- **intro/jee/jpa/fetch**
- Ders 02 alıştırması (dokümantasyona bakınız)