Razvoj Softvera

dr.sc. Emir Mešković

XI predavanje

Pregled

- JPA upiti
  - JPQL upitni jezik
  - Tipovi upita
  - Implementacija SQL koncepata u JPQL

JPQL upitni jezik

- JPA podržava više metodologija za pretragu baze podataka
  - JPQL, jezik portabilan za različite DBMS sisteme izveden iz SQL-a koji operira nad entitetima a ne na fizičkom modelu podataka
  - SQL koji se direktno prenosi DBMS-u
  - Kriterij API koji konstruiše pretragu korištenjem JAVA objekata
- JPQL pretrage izvršavaju se korištenjem dva interface-a
  - *Query* za povrat rezultata bez određenog tipa
  - *TypedQuery* za povrat rezultata tačno određenog tipa

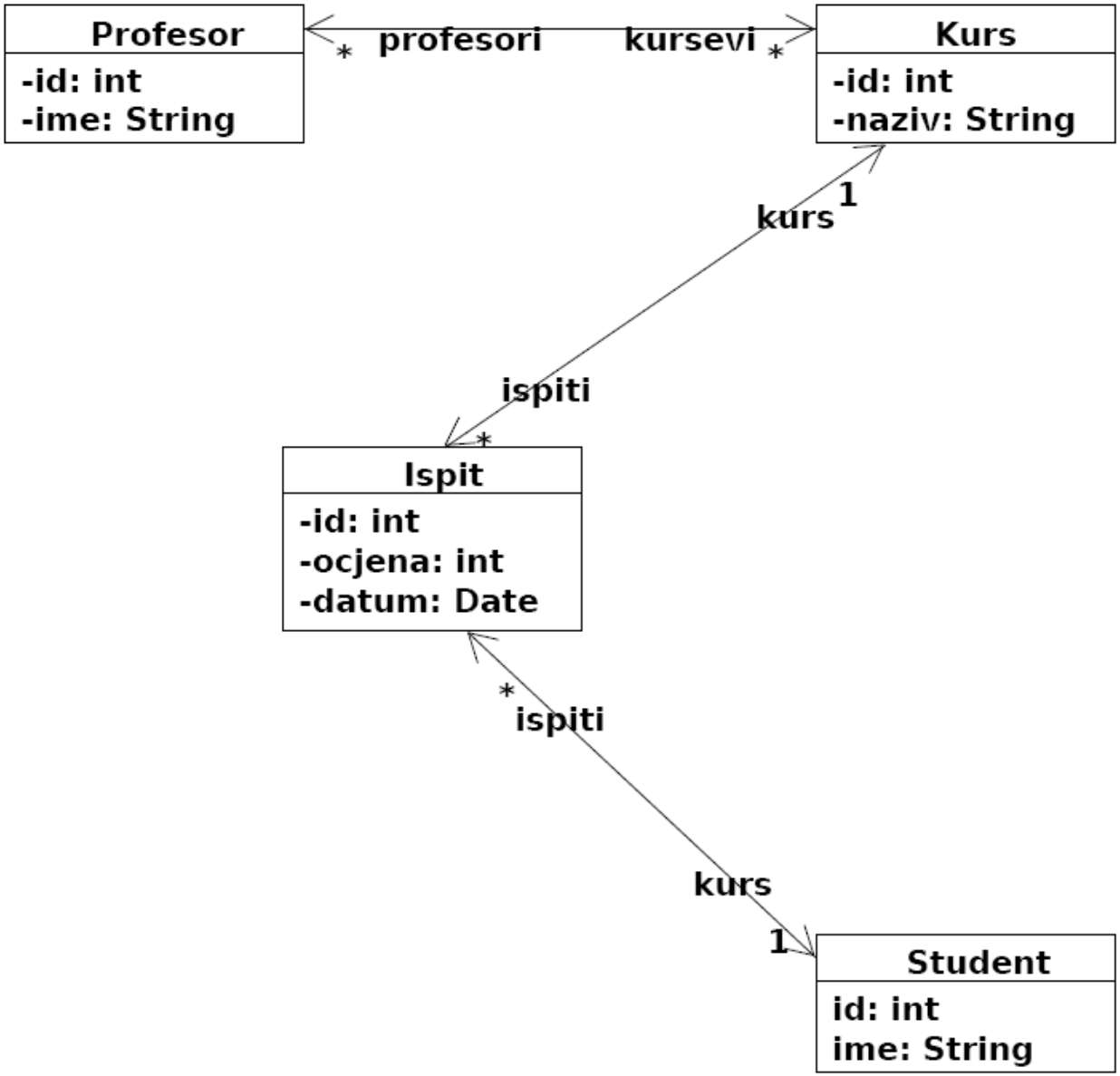
JPQL tipovi pretraga

- JPQL pretrage mogu biti:
  - Dinamički definisane putem stringova i to tokom izvršenja programa putem metoda *createQuery()* preko *EntityManager* objekta, a za šta postoje dvije varijante:
    - `em.createQuery(string, klasa) vraća TypedQuery`
    - `em.createQuery(string) vraća Query`
  - Statički definisane putem anotacije za entitet klase @NamedQuery i to u obliku:
    - `@NamedQuery(name = “nekoIme”, query = “nekiUpit”)`
    - `@NamedQueries({@NamedQuery(name = “i1”, query = “Q1”), @NamedQuery(name = “i2”, query = “Q2”)`

JPQL preuzimanje rezultata

- *Query* i *TypedQuery* podržavaju sljedeće često korištene metode za preuzimanje rezultata
  - *getSingleResult()*
    - Za slučaj kada se očekuje jedan rezultat,
    - generiše *NoResultException* iznimku ukoliko nema rezultata
    - generiše *NonUniqueResultException* iznimku za više rezultata
  - *getResultList()*
    - Za slučaj kada se vraća višestruki rezultat, vraća prazan kontejner ukoliko nema rezultata
- Rezultati u vraćenom kontejneru mogu biti tipa:
  - Osnovni Java tipovi (String, primitivni, JDBC tipovi)
  - Object[]
  - Entitet
  - Posebno definisani tipovi za tretman rezultata

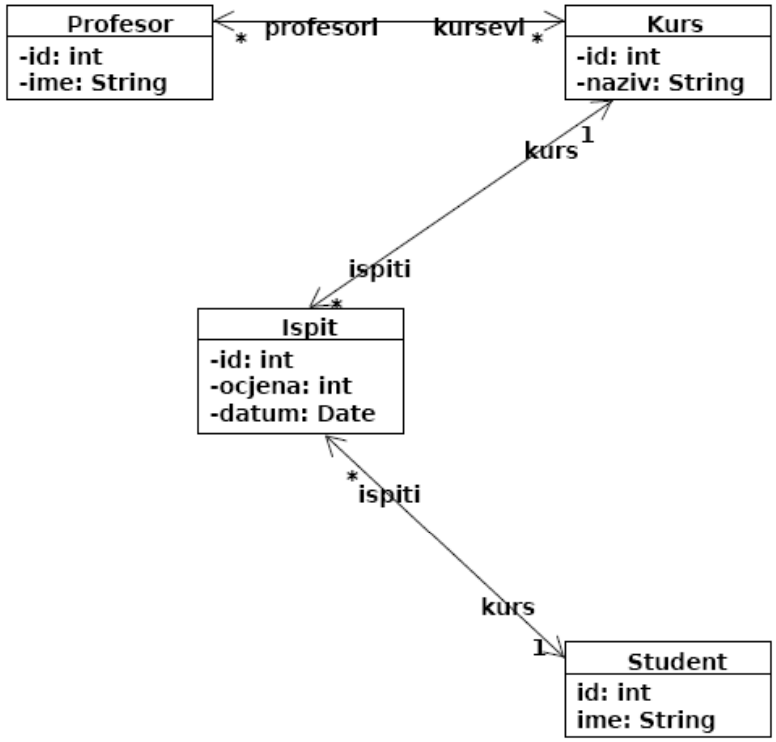
Model za primjere



Primjer dinamički i imenovani upit

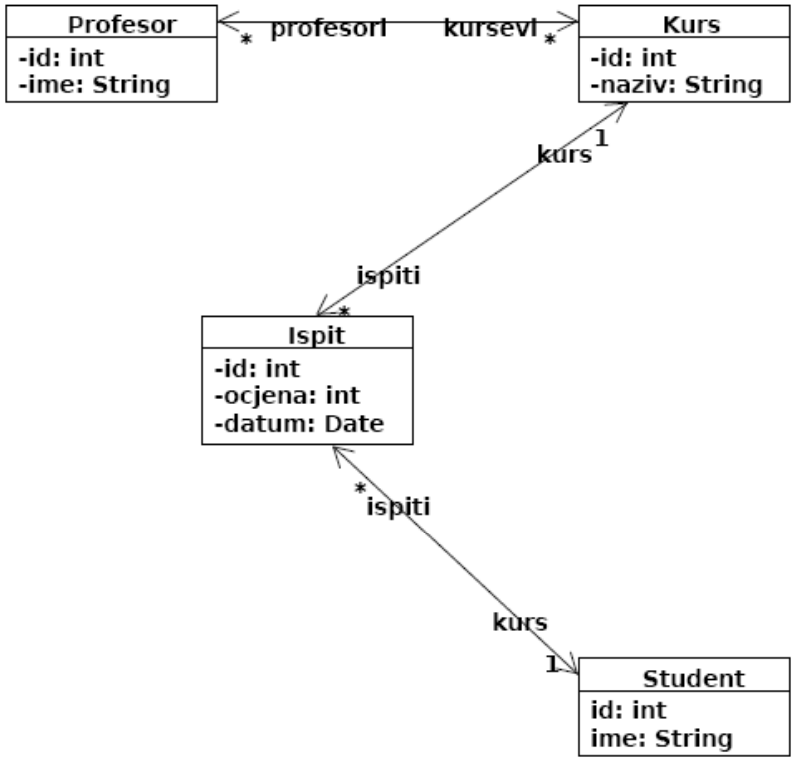

```
...
@Entity
@NamedQueries({
    @NamedQuery(name="Ispit.nadjiSve",query="SELECT i FROM Ispit i"),
    @NamedQuery(name="Ispit.pretraga",query="SELECT i, i.student.ime FROM Ispit i")
})
public class Ispit {
...
}
    public class Main {
       private static final String PERSISTENCE_UNIT_NAME = "TestPU";
       private static EntityManagerFactory emf;
       public static void main(String[] args) {
                  emf = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
                  EntityManager em = emf.createEntityManager();
                  Query q1 = em.createQuery("SELECT i FROM Ispit i");
                  Query q2 = em.createQuery("SELECT i, i.student.ime FROM Ispit i");
                  Query q3 = em.createNamedQuery("Ispit.nadjiSve",Ispit.class);
                  List ispiti1 = q1.getResultList();
                  for(Object o : ispiti1) {
                            System.out.println(o);
                  }
                  List<Object[]> rez = q2.getResultList();
                  for(Object[] niz : rez) {
                            System.out.println(niz[0].toString()+niz[1]);
                  }
                  List<Ispit> ispiti2 = q3.getResultList();
                  for(Ispit o : ispiti2 ) {
                            System.out.println(o);
                  }
                  em.close();
                  emf.close();
       }
```

Spajanja i parametrizirani upit



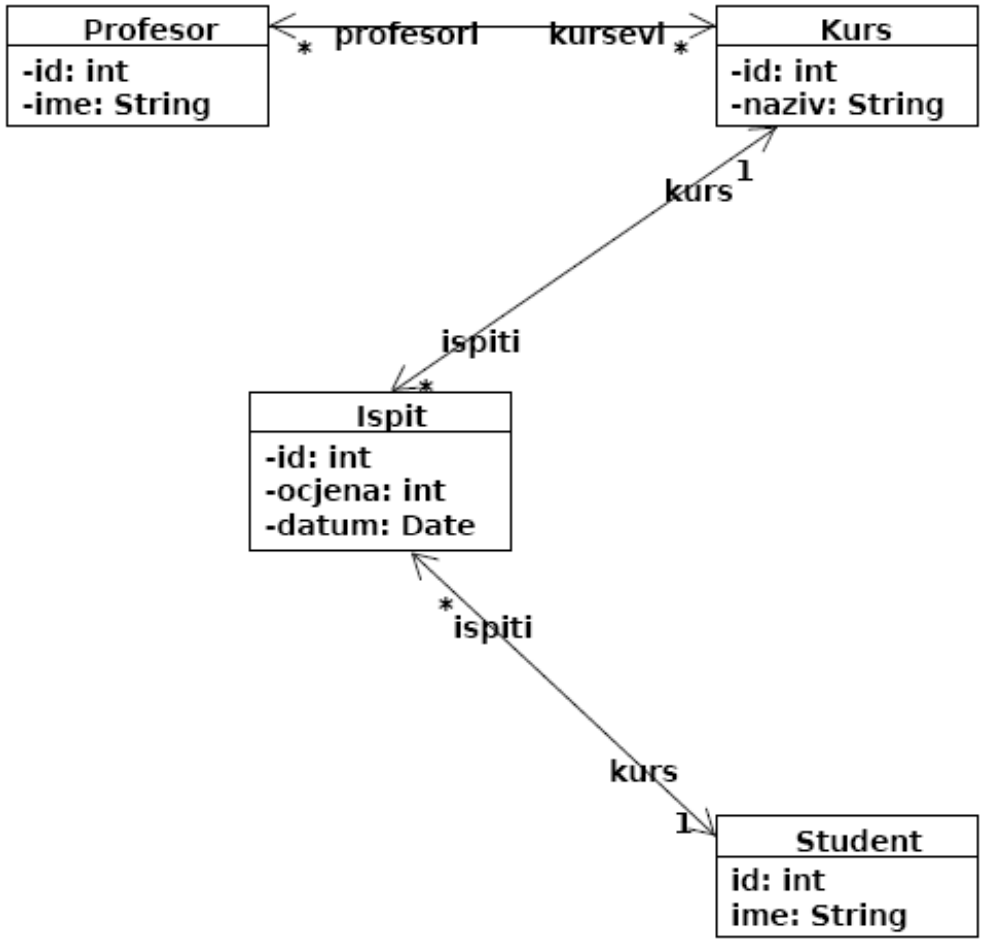

```
String primjer1 = "SELECT i FROM Ispit i WHERE i.student.ime = :stName";
Query q = em.createQuery(primjer1,Ispit.class).setParameter("stName","John");
String primjer2 =
    " SELECT s.ime " +
    " FROM Student s, Ispit i, Kurs k " +
    " WHERE i.kurs = k AND i.student = s AND k.naziv= 'OOP'";
String primjer3 =
    " SELECT s.ime " +
    " FROM Student s JOIN s.ispiti i JOIN i.kurs k" +
    " WHERE k.naziv= 'OOP'";
```

Grupisanje i agregatne funkcije




```
String primjer4 =
    " SELECT s, COUNT(i), MAX(i.ocjena), AVG(i.ocjena) " +
    " FROM Student s JOIN s.ispiti i " +
    " WHERE i.datum > {d '2008-12-31'} AND i.datum <={d '2010-12-31'}" +
    " GROUP BY s " +
    " ORDER BY s.ime DESC";
String primjer5 =
    " SELECT s, p, AVG(i.ocjena), COUNT(i.ocjena) " +
    " FROM Student s JOIN s.ispiti i JOIN i.kurs k JOIN k.profesori p" +
    " GROUP BY s,p";
```

Podupiti




```
String primjer6 =
    " SELECT p " +
    " FROM Profesor p " +
    " WHERE ( SELECT COUNT(k) FROM p.kursevi k) > 1";
```

UPDATE i DELETE naredbe

- Koriste se kada je potrebno ažurirati ili obrisati veći broj entiteta, umjesto da se svaki dohvaća i modificira sa menadžerom entiteta

```
em.createQuery("DELETE FROM Kurs k " +
                    "WHERE k.profesori IS EMPTY").executeUpdate();
```

- Pažnja: persistence context se ne ažurira kako bi se reflektovao rezultat operacije
  - Operacija se izvodi kao SQL naredba nad bazom podataka
  - Ažuriranje vrijednosti neće promijeniti trenutne vrijednosti u entitetima trenutno upravljanim menadžerom entiteta
  - Treba se izvesti u zasebnoj transakciji ili kao prva naredba u transakciji
- DELETE naredba se ne primjenjuje kaskadno na vezane entitete čak i ukoliko je postavljena REMOVE kaskadna opcija