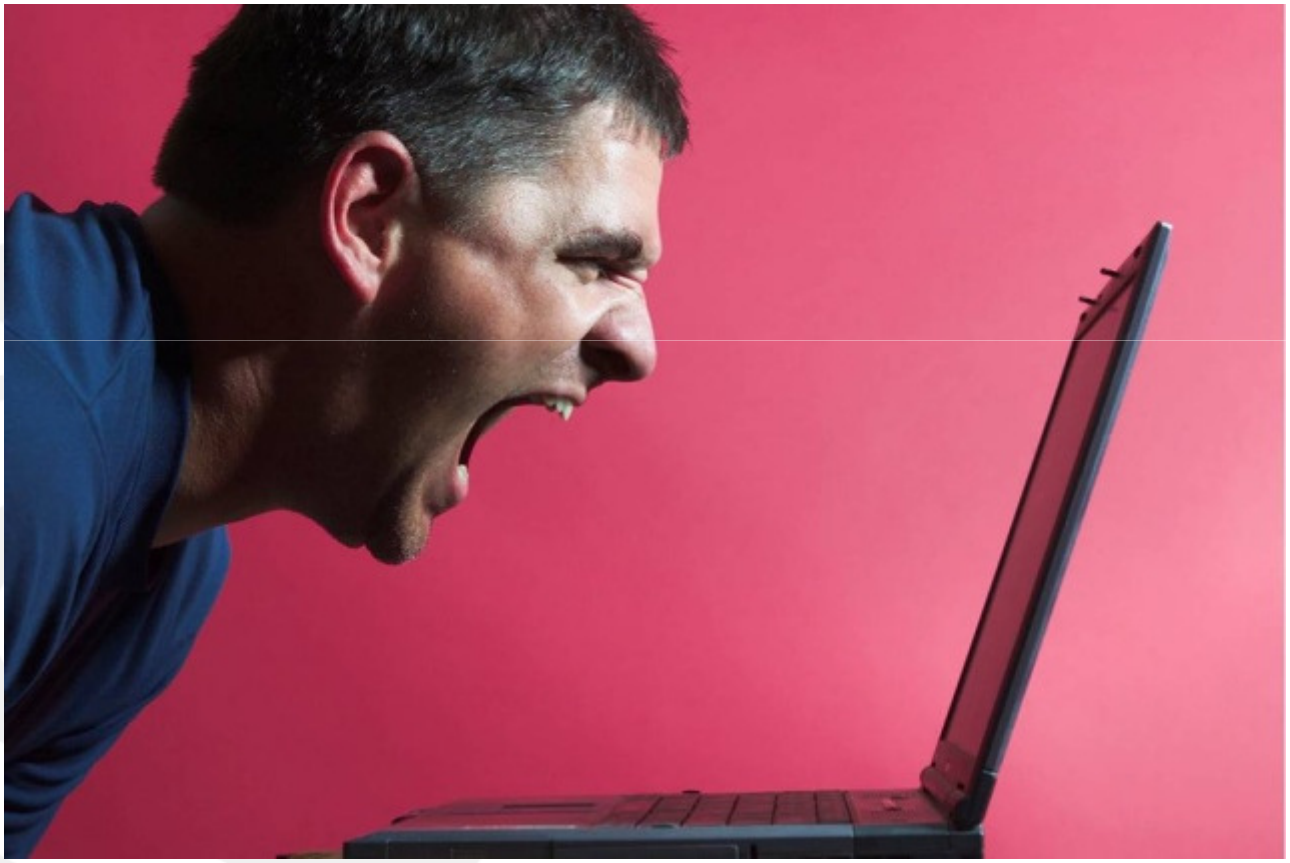
Java Persistence API (JPA) najbolje prakse

Slavko Žnidarić
Beta Tau Beta
slavko.znidaric@btb.hr

Vaš podatkovni sloj?

Vaš šef, DBA ili klijent?

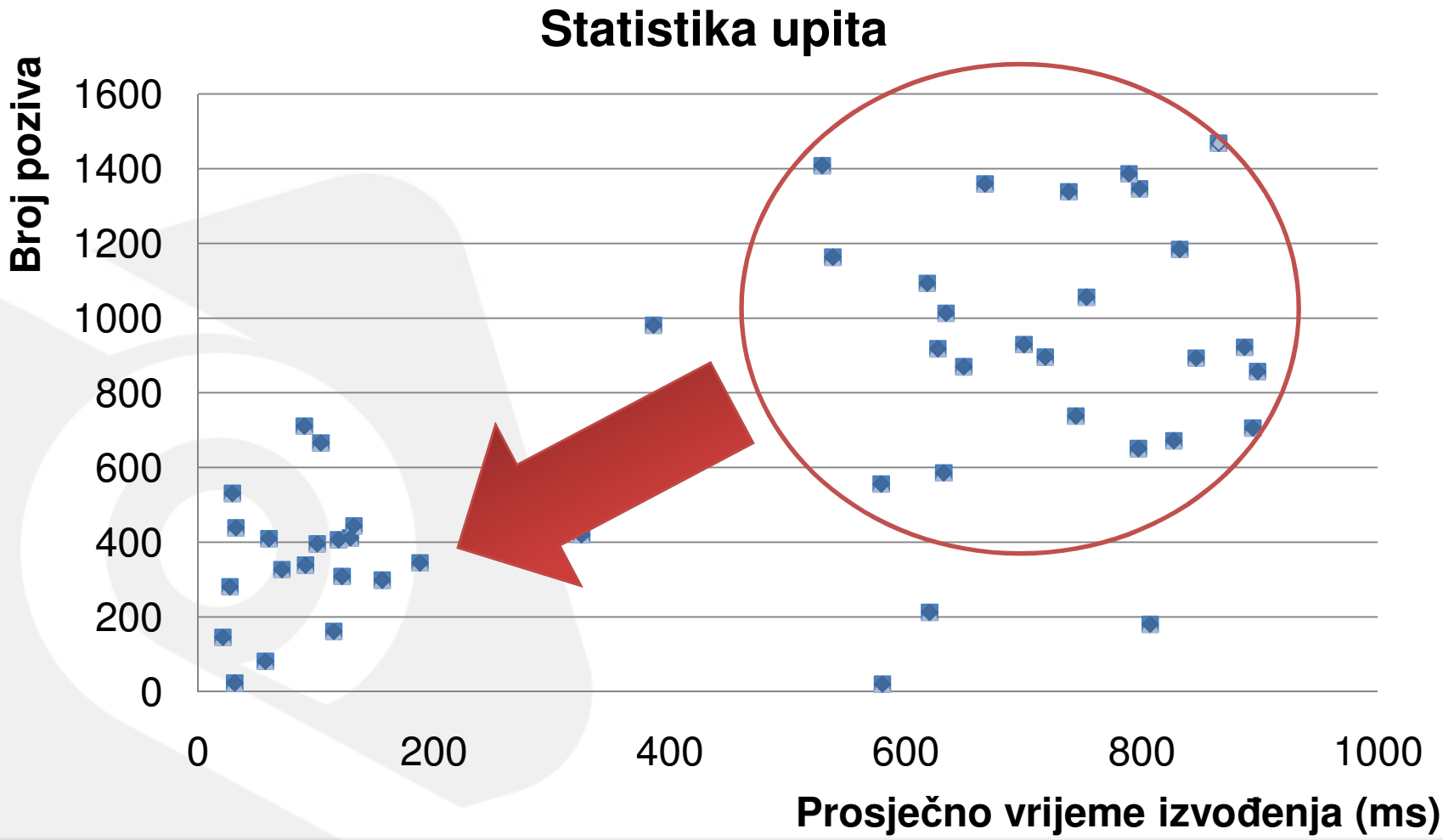
Vi?



Uzroci loših performansi

- Previše upita
- Spori upiti
- Loše podešena baza podataka
- Loša mrežna inrastruktura

Klasifikacija upita



Razlozi

Previše upita

- Poslovna logika
- Mapiranja
- Nedostatak keširanja
- N+1 select problem

Spori upiti

- Niska selektivnost
- Nedostatak veznih varijabli
- Nedostatak indeksa
- Kartezijev produkt
- DB Lock-ovi
- Dizajn baze podataka

Odakle krenuti?

DOHVAT

- Lazy
- Eager
- Join
- Batch
- Subselect

CACHING

- 1st Level Cache
- 2nd Level Cache

UPITI

- Selektivnost
- Bind varijable
- Query Cache

Dobar tuning je pitanje…

Lazy fetch

- Dafault za **one-to-many i many-to-many** relacije
- Dohvat objekta ***on-demand***
- Nema mreže objekata
- Persistence framework kreira ***Proxy u*** Persistence Context-u (PC) ili u memoriji
- Lazy fetch za **CLOB** i **BLOB** tipove je poželjan
  - Izdvajanje u zasebnu tablicu

Lazy fetch

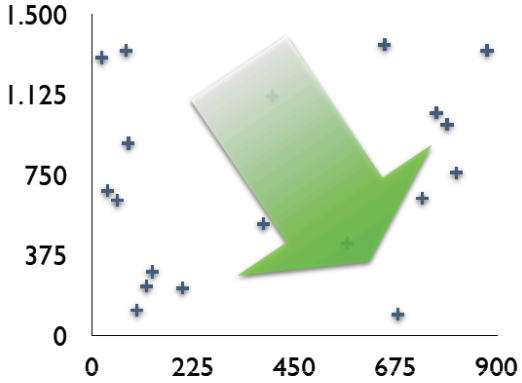

```
@Entity
public class Account {
...
@ManyToOne(fetch=FetchType.LAZY)
public Currency getCurrency() {...}
...
}
List accountList = session.createCriteria(Account.class).list();
    for (Account account : accountList) {
account.getCurrency().getCurrencyFullName();
}
```

N+1 select problem

```
SELECT * FROM ACCOUNTS
SELECT * FROM CURRENCY WHERE CURR_ID = ?
SELECT * FROM CURRENCY WHERE CURR_ID = ?
SELECT * FROM CURRENCY WHERE CURR_ID = ?
...
```

Eager fetch

- Obavezno konfigurirati **dubinu** dohvata:
  - `hibernate.max_fetch_depth`
  - razuman raspon **1 - 5**
- Koristi se **OUTER JOIN**
- **NE KORISTITI** sa 2 ili više kolekcije
- **Loš kandidat** za globalni plan dohvata!



Eager fetch

```
@Entity
public class Account {
...
@OneToMany(fetch=Fe        EAGER)
public Set<Transa        Transaction        ..}
@OneToMany(fetch        ER)
public Set<Payme        er> g        ntOrders        ..}
...
}
select account.*,        action.*, p        .*
from ACCOUNTS acco
left outer join TRANS
on account.ID = transactio
left outer join PAYMENTORDER paymentOrder
on
```

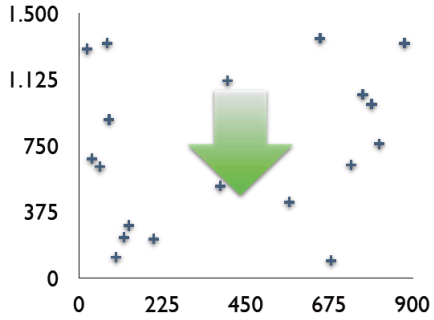
KARTEZIJEV PRODUKT

Join Fetch

- **CILJ**: izbjegnuti lazy fetch kod podataka koje često dohvaćamo
- Izbjegava se **N+1 select problem**
- Izbjegava se nepotreban eager dohvat za ostale slučajeve
- EAGER fetch alternativa

```
Query query = entityManager.createQuery(
"SELECT acc " +
"FROM Account acc JOIN FETCH acc.currency" +
"WHERE acc.accountNo = :accountNo");
query.setParameter("accountNo", accountNo);
Account acc = (Account) query.getSingleResult();
```

Batch fetch




```
@Entity
public class Account {
@ManyToOne(...)
public Currency getCurrency()
{...}
...
}
@Entity
@BatchSize(size=5)
public class Currency{
...
}
```

```
List accountList =
s.createCriteria(Account.class).list();
for (Account account: accountList) {
account.getCurrency().getCurrencyFullName(
);
}
```

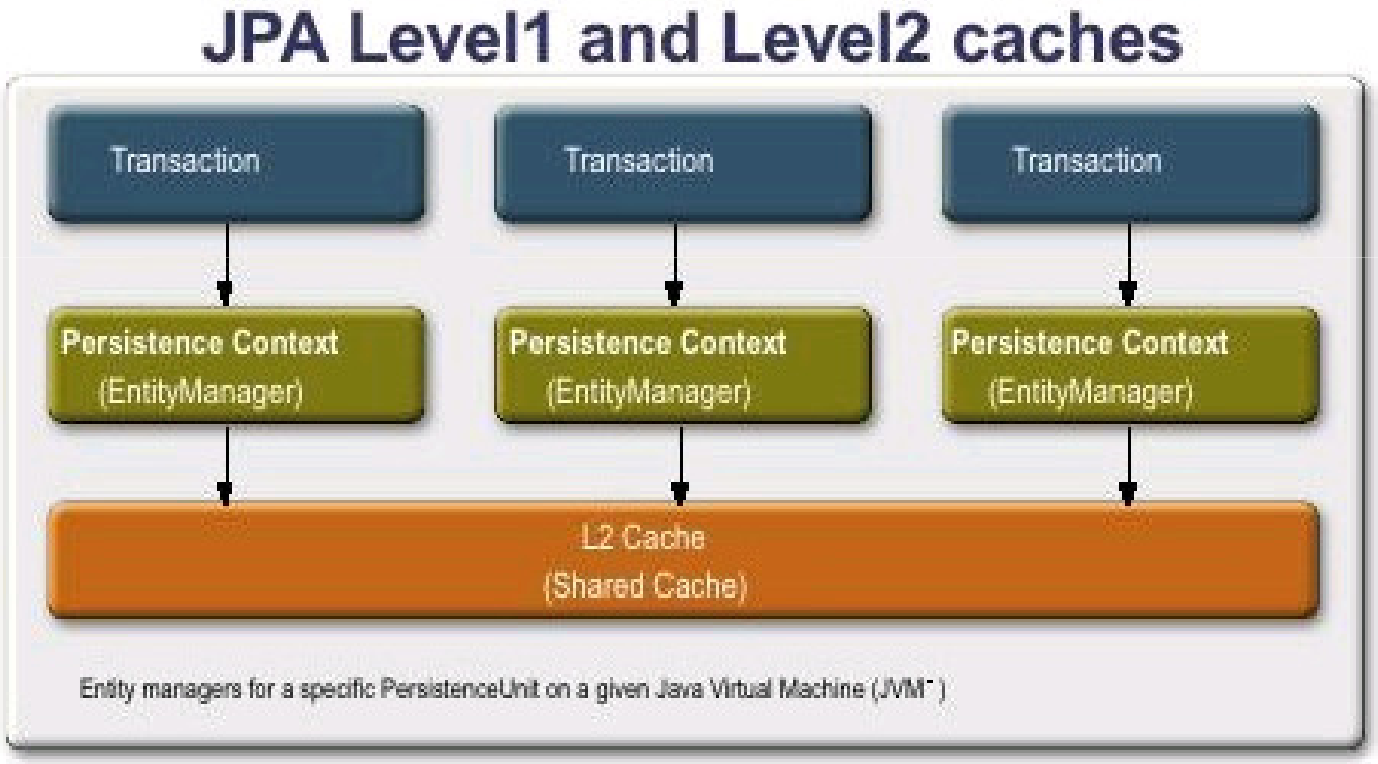
```
SELECT * FROM ACCOUNTS

SELECT * FROM CURRENCY WHERE
CURR_ID IN ( .....)
SELECT * FROM CURRENCY WHERE
CURR_ID IN ( .....)
```

E = (N / Batch Size) + 1

Caching

- Uključite 2nd level cache

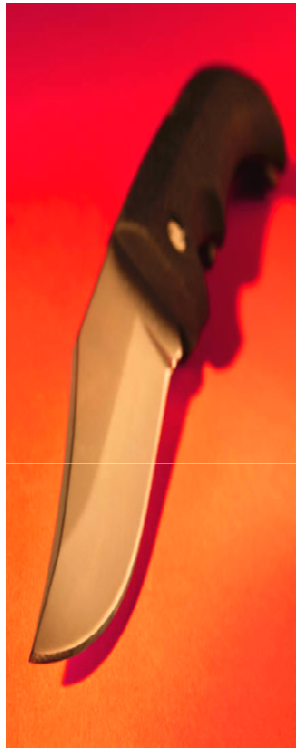


2nd level Cache

- KONZERVATIVAN odabir kandidata za keširanje
- Malo insert/update
- Mnogo dohvata
- Nekritični podaci
- Podaci korišteni kod puno sesija
- Podaci potrebni mnogim korisnicima

Bind varijable

```
"from User u where u.name=" + name
```



- SQL Injection
- Performance killer
- Tricky
- **UVIJEK** koristite bind varijable:

```
Query query =
session.createQuery("…from User u where u.name=
:name");
q.setString("name", "Slavko");
```

Transakcije

- Ne koristiti transkacije “read-only” upite
- Reducira se korištenje entity manager-a i izbjegava locking/isolation overhead.
- Istina za iznenađujuće velik broj upita u sustavima

```
@Stateless
public class PaymentOrderService implements BasicService {
@PersistenceContext
private EntityManager entityManager;
@TransactionAttribute(TransactionAttributeType.NOT_SUPPORTED)
public List<PaymentOrder> getPaymentOrders() {
    List<PaymentOrder> paymentOrders =
    entityManager.createQuery("SELECT p FROM PaymentOrder
    p").getResultList();
    return paymentOrders;
    }
}
```

Zapisivanje

- JPA nije Batch Tool!
- redovito koristite *flush* i *clear* kod procesiranja velike količine podataka

```
for ( int i=0; i<100000; i++ ) {
        Account acc = new Account(...);
        session.save(acc);
        if ( i % 50 == 0 ) {
                session.flush();
                session.clear();
        }
}
```

Analiza upita

- Uključite SQL output:

```
hibernate.show_sql=true
hibernate.format_sql=true
hibernate.use_sql_comments=true
toplink.logging.level=FINE
```

- Koristite alate za nadgledanje:
  - Oracle Enterpriser Manager, SQL Profiler, MySQL Enterprise Monitor
- Koristite alate za analizu upita
  - Oracle explain plan, SQL Query Analyzer, MySQL Query Analyzer
- Budite friendly sa vašim DBA

Analiza upita

- **Hibernate statistika**
- Izuzetno korisne informacije
- Potrebno ju je aktivirati:
  - Konfiguracijom:
    - hibernate.generate_statistics
  - Programski:
    - sessionFactory.getStatistics().setStatisticsEnabled(true)
- Pristup statisticici:
  - sessionFactory.getStatistics()

Analiza upita

iBank database statistics

REFRESH

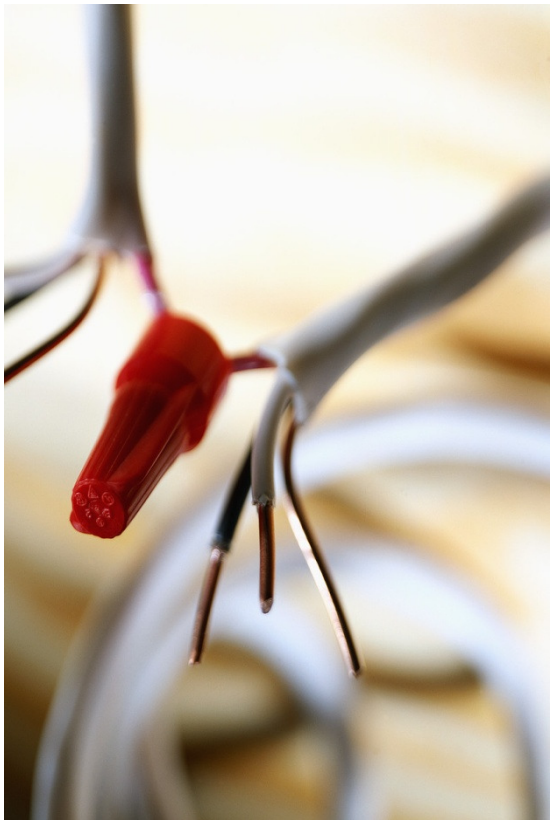
ENABLE STATISTICS | DISABLE STATISTICS

IBANK - HIBERNATE STATISTICS

Query	Avg time ▼	Max time	Min time	Row count	Execution Count	Cache Miss Count	Cache Put Count
select client from Client as client left join fetch client.clientStatus cs left join fetch client.person person left join fetch client.locationByCliePrimaryAddressLocaId prloc left join fetch client.locationByCliePrimaryAddressLocaId.place prlocpl order by client.clieName asc	1594		1594	2252	1		0
select charge from Charge as charge where (charge.charId> :filterIdStart and charge.userAction.uactId = :filterUactId) order by charge.charId asc	703		703	3195	1		0
select client from Client client left join client.account order by client.clieName asc	547		547	2252	1		0
select form from ApplicationForm as form	156		156	127	1		0
select distinct user.userId,person.persFirstname,person.persLastname,person.persJmbgId, status.usstName, businessUnit.buunName, businessUnit.buunCode,status.usstId, person.persId, person.persTaxNo from IbankUser user left join user.userRoles as userRoles left join userRoles.role as role left join user.person as person left join user.businessUnit as businessUnit left join user.userS ...	125		125	5	1		0
select count(distinct user.userId) from IbankUser user left join user.userRoles as userRoles left join userRoles.role as role left join user.person as person left join user.businessUnit as businessUnit left join user.userStatus as status where (0 = :userStatusIdInFilter or status.usstId = :userStatusId) and (0 = :userIdInFiler or user.userId = :userId) and (0 = :userJmbgInFi ...	62		62	1	1		0
select distinct businessEvent from BusinessEvent as businessEvent order by businessEvent.buevCode asc	31		31	73	1		0
select * from MODULE order by MODU_ID	16		16	13	1		0
select form from ApplicationFormStatus as form	16		16	4	1		0
select distinct chargeStatus from ChargeStatus as chargeStatus order by chargeStatus.chstId asc	16		16	2	1		0
select role from IbankUser user left join user.userRoles as userRoles left join userRoles.role as role where user.userId = :userId	2		0	5	5		0
select role from Role as role where role.moduleByRoleModuId.moduId=:moduleId	0		0	18	1		0

Baza i infrastruktura

- Uvijek **nadgledajte** bazu podataka!
- Da li su **indeksi** postavljeni **korektno**?
- U kakvom je stanju **DB runtime**?
- U kakvom je stanju **infrastruktura**?
  - ➡ Connection Pool
  - ➡ Transaction Monitor
  - ➡ Application Server



Pitanja?

Hvala!

slavko.znidaric@btb.hr