Livelli di isolamento
Esercizi

Esempio 1 (perdita di aggiornamento)

```
Start Transaction
        isolation level XXX
Read(x)

                                Start Transaction
                                        isolation level XXX
                                Read(x)

x=x+100
Write(x)
Commit

                                x=x+1
                                Write(x)
                                Commit
```

Su Postgres

- Supponiamo che x sia il saldo del conto con `num=1` della relazione conti

```
create schema "20200511livelliIsolamento";
set search_path to "20200511livelliIsolamento";
create table conti (num integer primary key,
                    saldo integer);
insert into conti values (1,1000);
```

- Leggiamo con

```
select saldo into app1 from conti where num=1;
```

- E scriviamo con

```
update conti set saldo =
     (select saldo + 100 from app)
      where num = 1
```

Esempio 1 su Postgres

```
start transaction
  isolation level serializable;
select saldo into app1
from conti where num=1;
```

```
start transaction
  isolation level serializable;
select saldo into app2
from conti where num=1;
```

```
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
commit
```

```
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;
commit
```

```
drop table if exists app1
```

```
drop table if exists app2
```

Esempio 1 con dati diversi su Postgres: OK

```
start transaction
  isolation level serializable;
select saldo into app1
from conti where num=1;
```

```
start transaction
  isolation level serializable;
select saldo into app2
from conti where num=2;
```

```
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
commit
```

```
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 2;
commit
```

```
drop table if exists app1
```

```
drop table if exists app2
```

Vediamo meglio il comportamento

- Lock
- Verifica dell'ammissibiltà

Esempio 1 bis su Postgres

```
start transaction
  isolation level serializable;
select saldo into app1
from conti where num=1;
```

```
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
```

```
commit
```

```
drop table if exists app1
```

```
start transaction
  isolation level serializable;
select saldo into app2
from conti where num=1;
```

```
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;
```

```
commit
```

```
drop table if exists app2
```

Esempio 1 bis su Postgres (per conto vostro)

```
start transaction isolation level
                    repeatable read;
select saldo into app1
from conti where num=1;



update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;



commit



drop table if exists app1
```

```
start transaction isolation level
                    repeatable read;
select saldo into app2
from conti where num=1;



update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;

commit



drop table if exists app2
```

Esempio 1 bis su Postgres

```
start transaction
  isolation level read committed;
select saldo into app1
from conti where num=1;



update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;



commit



drop table if exists app1
```

```
start transaction
  isolation level read committed;
select saldo into app2
from conti where num=1;



update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;

commit



drop table if exists app2
```

Esempio 1 con 2PL

Start Transaction
isolation level XXX
Read(x)

x=x+100
Write(x)
Commit

Start Transaction
isolation level XXX
Read(x)

x=x+1
Write(x)
Commit

Esempio 1 con 2PL

```
start transaction
        isolation level serializable
rlock (x)
read(x)
                                        start transaction
                                                isolation level serializable
                                        rlock (x)
                                        read(x)
wlock(x)
                                        wlock(x)
abort (timeout)
                                        write(x)
                                        commit
```

Esempio 1ter con 2PL

Start Transaction
isolation level XXX
Read(x)
x=x+100
Write(x)

Commit

Start Transaction
isolation level XXX

Read(x)

x=x+1
Write(x)
Commit

Esempio 1ter con 2PL

```
start transaction
        isolation level serializable
rlock (x)
read(x)
write(x)
                                        start transaction
                                                isolation level serializable
                                        rlock (x)
commit
                                        read(x)
                                        write(x)
                                        commit
```

Esempio 1 ter su Postgres

```
start transaction
  isolation level serializable;
select saldo into app1
from conti where num=1;
```

```
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
```

```
commit
```

```
drop table if exists app1
```

```
start transaction
  isolation level serializable;
```

```
select saldo into app2
from conti where num=1;
```

```
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;
commit
```

```
drop table if exists app2
```

Confronto

- Esempio 1ter
  - 2PL (DB2):
    - con i lock, ha governato la situazione e ha portato a compimento entrambe le transazioni
  - MVCC (Postgres):
    - troppo ottimista, ha fatto avviare entrambe le transazioni e non può portarle a termine (almeno non entrambe)
- Esempio 1bis
  - 2PL
    - va in stallo
  - MVCC (Postgres):
    - gestisce la situazione, abortendo una transazione

Esercizi

- Continuate per conto vostro, con gli altri esercizi nella presentazione che è stata inviata, o altri (ad esempio compiti d'esame)