Esempio 2
(esame 09/2012)

Start Transaction	
Read(x)	
x=x+100	
Write(x)	
	Start Transaction
	Read(x)
	x=x+1
	Write(x)
Commit	
	Commit

Esempio 2 con 2PL

Start Transaction	
Read(x)	
x=x+100	
Write(x)	
	Start Transaction
	Read(x)
	x=x+1
	Write(x)
Commit	
	Commit

Esempio 2 con 2PL (repeatable read)

Esempio 2 con 2PL (read committed)

Esempio 2 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;
                                     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
                                     commit


drop table if exists app1            drop table if exists app2
```

Esempio 2

- 2PL funziona bene in modo efficace ed efficiente
- MV evita l'anomalia, ma con l'abort

Esempio 2 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;
                                     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
                                     commit


drop table if exists app1            drop table if exists app2
```

Esempio 3: dirty read

Start Transaction	
Read(x)	
x=x+100	
Write(x)	
	Start Transaction
	Read(x)
	x=x+1
	Write(x)
Abort	
	Commit

Esempio 3

- Con repeatable read o serializable
  - OK entrambi
    - Con 2PL la seconda si ferma prima della lettura
    - Con MV la seconda si ferma prima della scrittura (e quindi è un "un po' più veloce")

Esempio 3 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;
                                     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;
abort
                                     commit


drop table if exists app1            drop table if exists app2
```

Esempio 3 su Postgres

```
start transaction isolation level
                read uncommitted;
select saldo into app1
from conti where num=1;
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
                                     start transaction isolation level
                                                     read uncommitted;
                                     select saldo into app2
                                     from conti where num=1;
                                     update conti set saldo =
                                       (select saldo + 1 from app2)
                                          where num = 1;
abort
                                     commit


drop table if exists app1            drop table if exists app2
```

Esempio 3 con 2PL (es DB2 o SQLServer)

```
start transaction isolation level
                read uncommitted;
select saldo into app1
from conti where num=1;
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
                                     start transaction isolation level
                                                   read uncommitted;
                                     select saldo into app2
                                     from conti where num=1;
                                     update conti set saldo =
                                       (select saldo + 1 from app2)
                                          where num = 1;
abort

                                     commit



drop table if exists app1            drop table if exists app2
```

Che cosa succede?

Esempio 3 con 2PL (es DB2 o SQLServer)

```
start transaction isolation level
                read uncommitted;
select saldo into app1
from conti where num=1;
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
                                     start transaction isolation level
                                                     read uncommitted;
                                     select saldo into app2
                                     from conti where num=1;
                                     update conti set saldo =
                                       (select saldo + 1 from app2)
                                          where num = 1;
abort
                                     commit


drop table if exists app1            drop table if exists app2
```

Esempio 3 con 2PL (es DB2 o SQLServer)

```
start transaction isolation level
                read uncommitted;
select saldo into app1
from conti where num=1;
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
```

```
start transaction isolation level
                read uncommitted;
select saldo into app2
from conti where num=1;
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;
```

.

Esempio 3 con 2PL (es DB2 o SQLServer)

```
start transaction isolation level
                read uncommitted;
select saldo into app1
from conti where num=1;
update conti set saldo =
  (select saldo + 100 from app1)
     where num = 1;
```

```
start transaction isolation level
              read uncommitted;
select saldo into app2
from conti where num=1;
update conti set saldo =
  (select saldo + 1 from app2)
     where num = 1;
```

LEGGE IL DATO SPORCO

.

.

Esempio 4, lettura inconsistente

Start Transaction	
Read(x)	
	Start Transaction
	Read(x)
	x=x+20
	Write(x)
	Commit
Read(x)	
Commit	

Esempio 4, con Postgres

```
start transaction
  isolation level serializable;
select saldo
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 + 20 from app2)
                                        where num = 1;
                                   commit
select saldo
from conti where num=1;
commit

                                   drop table if exists app2
```

Esempio 4, con Postgres

```
start transaction isolation level
                  read committed;
select saldo
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 + 20 from app2)
                                          where num = 1;
                                     commit

select saldo
from conti where num=1;
commit

                                     drop table if exists app2
```

Repeatable read vs serializable

```
start transaction  isolation level …
                                          start transaction isolation level …
select count(*)
from conti
                                          select count(*)
                                          from conti;
                                          insert into conti  values ( … );
insert into conti values (…)
commit
                                          commit
```

```
delete from conti;
insert into conti values (1,201);
```

Repeatable read vs serializable con Postgres

```
start transaction isolation level
                  repeatable read;
select count(*) as n into app1
from conti;
```

```
start transaction isolation level
                  repeatable read;
select count(*) as n into app2
from conti;
insert into conti
  select n+3, 3000
  from app2;
```

```
insert into conti
  select n+1, 2000
  from app1;
commit
```

```
commit;
```

```
drop table if exists app1
```

```
drop table if exists app2
```

Con una esecuzione seriale

- Prima 1 e poi 2
  - Inseriamo 2,2000 e poi 5,3000
- Prima 2 e poi 1
  - Inseriamo 4,3000 e poi 3,2000

```
delete from conti;
insert into conti values (1,1000);
```

Repeatable read vs serializable con Postgres

```
start transaction isolation level
                  serializable;
select count(*) as n into app1
from conti;




insert into conti
  select n+1, 2000
  from app1;
commit

drop table if exists app1
```

```
start transaction isolation level
                     serializable;
select count(*) as n into app2
from conti;
insert into conti
  select n+3, 3000
  from app2;




commit;

drop table if exists app2
```