CI218: Nível de Isolamento de Transações

Profa. Carmem Hara

Departamento de Informática/UFPR

17 de janeiro de 2023

Problema do Fantasma

Tabela cc	id	tipoCC	proprietario	saldo
	1	1	Jose	100
	2	2	Maria	200
	3	2	Pedro	150

```
          T1                                T2
-------------------------   ----------------------------
select max(saldo) from cc
where tipoCC=1;
                                  insert into cc
                                     values (4, 1, 'Ze', 200);

                                  delete from cc where id=2;
                                  commit;
select max(saldo) from cc
where tipoCC=2;
commit;
```

Problema do Fantasma

- T1: bloqueia páginas que contém CC com tipo 1 no primeiro select
- T2: A CC 4 é inserida em uma página diferente da que contém a CC 1: insere um "fantasma"
- Resultado de T1: 100, 150
- O protocolo 2PL estrito não resolveu o problema:
  - Escalonamento serial T1-T2: 100, 200
  - Escalonamento serial T2-T1: 200, 150

Possíveis Soluções - Problema do Fantasma

- Bloquear **todas** as páginas de CC e garantir que novas páginas não sejam adicionadas
- Se houver um índice sobre `tipoCC`, bloquear o índice. Assim, a inserção/remoção de entradas no índice por T2 esperam T1 ser efetivada

Isolamento

- Das propriedade ACID:
  - Isolamento: cada transação deve ser executada como se estivesse sozinha no sistema, ou seja, isolada das demais transações
- Fenômenos que podem acontecer sem o isolamento:
  - Leitura suja
  - Leitura não repetível
  - Fantasma

Leitura Suja

- Leitura de um item escrito por uma outra transação ainda não efetivada

```
   T1                    T2
-----------          -------------
read( X );
X:= X-10;
write( X );
                     read( X );
                     read( Y );
                     print( X + Y );
abort;
```

Leitura Não Repetível

- Uma transação lê valores diferentes para o mesmo item de dado (sem ter feito a atualização)

```
    T1                      T2
------------          ---------------
                       read( X )
                       print( X ) --> 15
read( X )
X:= X-10;
write( X );
commit
                       read( X );
                       print( X ) --> 5
```

Fenômeno Fantasma

- Novos itens de dados (fantasmas) inseridos por outras transações afetam o resultado de consultas

```
    T1                      T2
------------          ---------------
                       read( min(R) ) --> 10
insert 5 in R
commit
                       read( mint(R) ) --> 5
```

Níveis de Isolamento de Transações

Nível	Leitura Suja	Leitura Não Repetível	Fantasma
Read Uncommitted	Sim	Sim	Sim
Read committed	Não	Sim	Sim
Repeatable read	Não	Não	Sim
Serializable	Não	Não	Não

- **Serializable**: implementa o 2PL Estrito (todos os bloqueios são mantidos até o final da transação sobre o *conjunto* de dados utilizado na transação
- **Repeatable read**: bloqueia objetos individuais (não o conjunto de objetos)
- O comportamento pode ser diferente em cada SGBD

Exemplo no Postgres

- inicio de transação:
  `begin`
- mostrar o nível de isolamento:
  `show transaction isolation level`
- mudar o nível de isolamento:
  `set transaction isolation level`
  `[read uncommitted | read committed |`
  `repeatable read | serializable]`
- efetivar a transação:
  `commit`
- cancelar a transação:
  `rollback`

Postgres: Tabela CC

```
dbcc=# create table cc (
   id   serial primary key,
   prop char(8),
   saldo numeric(8,2));

dbcc=# insert into cc (prop, saldo) values ('um', 100 );
dbcc=# insert into cc (prop, saldo) values ('dois', 100 );
dbcc=# insert into cc (prop, saldo) values ('tres', 100 );

dbcc=# select * from cc;
 id |   prop   | saldo
----+----------+--------
  1 | um       | 100.00
  2 | dois     | 100.00
  3 | tres     | 100.00
```

Exemplo no Postgres

Iniciar uma transação e obter o nível de isolamento

```
dbcc=# show transaction isolation level;
 transaction_isolation
-----------------------
 read committed

dbcc=# begin;
BEGIN
dbcc=# set transaction isolation level read uncommitted;
SET
dbcc=# show transaction isolation level;
 transaction_isolation
-----------------------
 read uncommitted
```

Read Uncommitted

```
          T1                              T2
-------------------------   -----------------------------
select * from cc;
                                select * from cc where id=1;
                                 id |   prop    | saldo
                                ----+-----------+--------
                                  1 | um        | 100.00
update cc set saldo=saldo-10
where id=1 returning *;
                                select * from cc where id=1;
                                 id |   prop    | saldo
                                ----+-----------+--------
                                  1 | um        | 100.00
commit;
```

- Postgres: não implementa o read uncommitted
- MySQL: o saldo em T2 seria 90.00 (leitura suja)

Read Committed

```
          T1                                  T2
--------------------------   -------------------------------
select * from cc;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+----------+--------
                               1 | um        | 90.00
update cc set saldo=saldo-10
where id=1 returning *;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+----------+--------
                               1 | um        | 90.00
                             select * from cc where saldo >=90;
commit;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+----------+--------
                               1 | um        | 80.00
                             select * from cc where saldo >=90;
```

Repeatable Read

```
          T1                                  T2
--------------------------   -------------------------------
select * from cc;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+-----------+--------
                               1 | um        | 80.00
update cc set saldo=saldo-10
where id=1 returning *;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+-----------+--------
                               1 | um        | 80.00
                             select * from cc where saldo >=90;
commit;
                             select * from cc where id=1;
                              id |   prop    | saldo
                             ----+-----------+--------
                               1 | um        | 80.00
                             select * from cc where saldo >=90;
```

Repeatable Read

O problema de fantasma foi resolvido com este nível de isolamento.

```
          T1                              T2
--------------------------   -------------------------------
                              update cc set saldo=saldo-10
                              where id=1 returning *;
                              ERROR:  could not serialize access due to
                              concurrent update

                              rollback;
```

No MySQL o resultado em T2 seria:

```
 id |   prop    | saldo
----+-----------+-------
  1 | um        | 60.00
```

Embora no último SELECT o saldo aparecia como 80.00

Repeatable Read - Anomalia de Serialização

```
          T1                                T2
--------------------------   -------------------------------
select sum(saldo) from cc;
  sum
--------
 270.00

insert into cc (prop, saldo)
values ('soma', 270);
                                select sum(saldo) from cc;
                                  sum
                                --------
                                 270.00

                                insert into cc (prop, saldo)
                                values ('soma', 270);
commit;
                                commit;
```

Repeatable Read - Anomalia de Serialização

O conteúdo final da tabela não é equivalente a execução serial T1-T2 ou T2-T1:

```
 id |   prop    | saldo
----+-----------+--------
  2 | dois      | 100.00
  3 | tres      | 100.00
  1 | um        |  70.00
  4 | soma      | 270.00
  5 | soma      | 270.00
```

Serializable

```
          T1                                  T2
--------------------------   --------------------------------
select sum(saldo) from cc;
  sum
--------
 810.00

insert into cc (prop, saldo)
values ('soma', 810);
                                select sum(saldo) from cc;
                                  sum
                                --------
                                 810.00

                                insert into cc (prop, saldo)
                                values ('soma', 810);
commit;
                                commit;
```

Serializable

- O commit de T2 retorna ERRO:

```
ERROR: could not serialize access due to
read/write dependencies among transactions
DETAIL: Reason code:  Canceled on identification
as a pivot, during commit attempt.
HINT: The transaction might succeed if retried.
```

- O conteúdo final da tabela:

```
 id |   prop   | saldo
----+----------+--------
  2 | dois     | 100.00
  3 | tres     | 100.00
  1 | um       |  70.00
  4 | soma     | 270.00
  5 | soma     | 270.00
  6 | soma     | 810.00
```

- no MySQL: a transação T2 espera o commit de T1 no select