Transações

BDII

Guilherme Dal Bianco

Introdução

Atualizar o saldo de uma conta

```
UPDATE Contas SET Saldo = Saldo - 50 WHERE NoConta = 123
```

Transação com mais operações: transfere 50 reais da conta 123 para conta 456

```
begin;
UPDATE Contas SET Saldo = Saldo - 50 WHERE NoConta = 123;
UPDATE Contas SET Saldo = Saldo + 50 WHERE NoConta = 456;
end;
```

Introdução

Simplificando as operações:

```
Read(A)
A=A-50
Write(A)
Read(B)
B=B+50
Write(B)
```

Onde *A* e *B* representam os saldos das duas contas contas correntes conhecidas

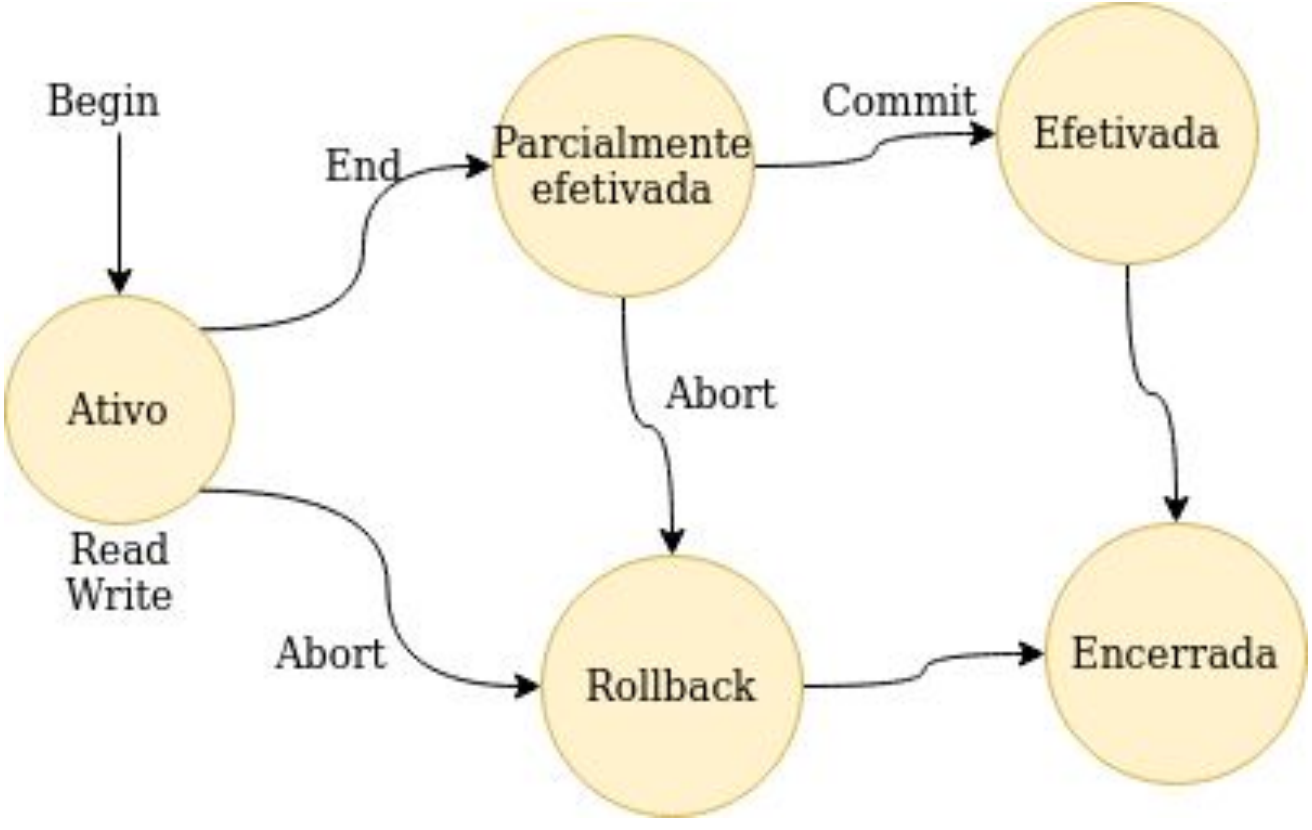
Introdução

Transação

Fim normal = Commit

Fim anormal= abort (rollback)

Transações- Estados



Transações-ACID

- Atomicidade
- Consistência
- Isolamento
- Durabilidade

Transações-ACID

Atomicidade: todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

Transações-ACID

Atomicidade: todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

Consistência: transações preservam a consistência da base

Transações-ACID

Atomicidade: todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

Consistência: transações preservam a consistência da base

Isolamento: a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definida

Transações-ACID

Atomicidade: todas as operações de uma transação devem ser efetivadas; ou, na ocorrência de uma falha, nada deve ser efetivado

Consistência: transações preservam a consistência da base

Isolamento: a maneira como várias transações em paralelo interagem (o que pode ser lido e o que pode ser escrito por cada uma) deve ser bem definida

Durabilidade: uma vez consolidada (committed) a transação, suas as alterações permanecem no banco até que outras transações aconteçam

Exemplo Transaçōes

- create table trans (id int, nome varchar(50), saldo float);

begin;

insert into trans values (1,’joao’, 1000);

insert into trans values (2,’maria, 2000);

commit;

Exemplo Transaçōes

```
--Transação A
begin;
update trans set saldo = 1200 where id = 1;
commit;

--Transação B
begin;
 select * from trans where id = 1;
commit;
```

Atividade 1

- Abra um terminal no Postgres:
  - sudo -u postgres psql postgres
- Crie a seguinte tabela:
  - cliente(numero int, cpf int , nome varchar(50))
- Crie as seguintes operações e descreva o que acontece em cada caso

  1 Uma transação com “commit” (faça 3 inserts)

  2 Uma transação com “rollback” (faça 3 inserts)

  3 Uma transação com dois inserts com o mesmo valor do atributo “numero”, o que acontece?

  4 Uma transação tentando acessar dados de outra transação ainda não “comitados”. (abra dois terminais com o postgres). O que acontece?

Exemplo de transação em python

```
import psycopg2

conn = None
try:

    conn = psycopg2.connect(database="teste", user = "guilherme", password = "    ", host =
"127.0.0.1", port = "5432")
    cur = conn.cursor()
    cur.execute(“insert into employee values (1,’maria’))
    cur.execute(“insert into employee values (2,’jose’))
    conn.commit()
    cur.close()
except psycopg2.DatabaseError as error:
    print(error)
finally:
    if conn is not None:
        conn.close()
```

Atividade prática 2 - Homework

Construa uma aplicação em uma linguagem de programação capaz de executar N inserções usando uma **transação** na tabela “Cliente”. A aplicação deve ser capaz também de listar a tabela após as inserções. Trate a exceção no caso de uma inserção de uma chave já existente.

slido

Uma atualização, dentro de uma transação, tem seus dados persistidos em disco quando:

ⓘ Start presenting to display the poll results on this slide.

slido

Uma transação, altera do estado "parcialmente efetivada" para "efetivada" quando

ⓘ Start presenting to display the poll results on this slide.

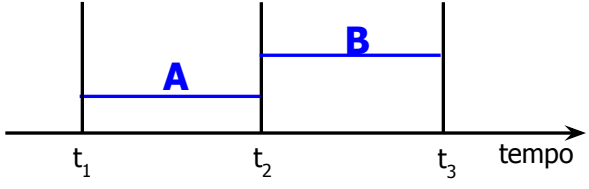
slido

When a hardware or software failure occurs, the information in the database must be accurate up to the last committed transaction before the failure.

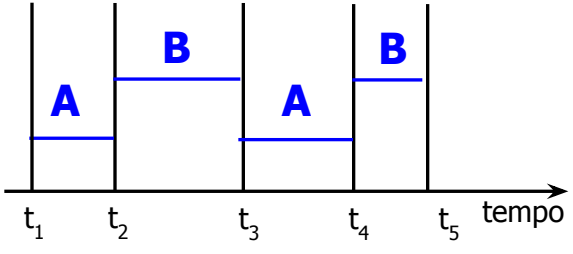
ⓘ Start presenting to display the poll results on this slide.

Concorrência

Execução Serial (sequencial):



Execução Intercalada:



Controle de Concorrência

Execução Serial (sequencial): diversas transações executadas em sequência

- deixa a base de dados em estado correto e consistente

Execução Intercalada: comandos de diversas transações são intercalados

- pode levar a inconsistências

Controle de Concorrência

Execução Serial

- estado inicial correto e consistente $\Rightarrow$ estado final correto e consistente

Controle de Concorrência

Execução Intercalada

- Toda execução serial é consistente

- Mas uma execução intercalada só é consistente se for igual ao resultado de uma execução em sequência (em ordem conhecida)
  - esta execução é dita **serializável**

Problemas de Execução Intercalada

Ocorrência de anomalias

1. leitura inválida
2. leitura não repetível
3. leitura fantasma

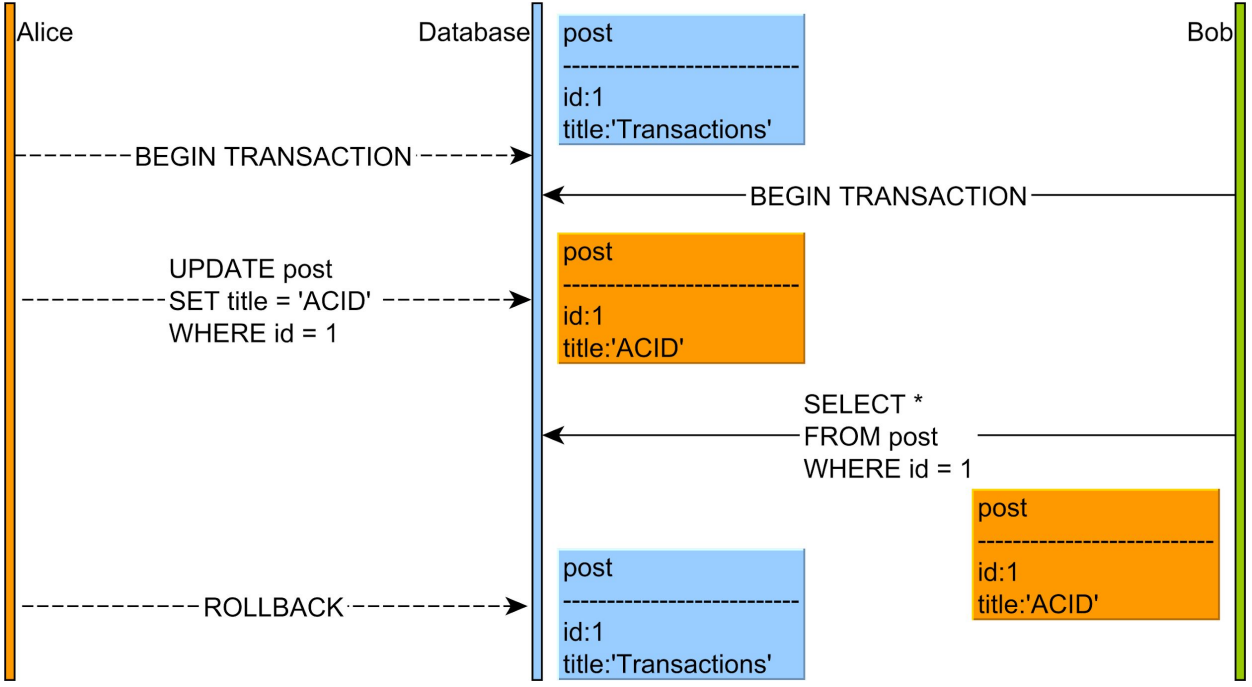
Problemas de Execução Intercalada

1) Leitura inválida (*Dirty Read*):

- transação T2 lê um dado modificado por uma transação T1 que ainda não terminou;

Problemas de Execução Intercalada

Leitura inválida (*Dirty Read*):



Fonte: https://vladmihalcea.com/dirty-read/

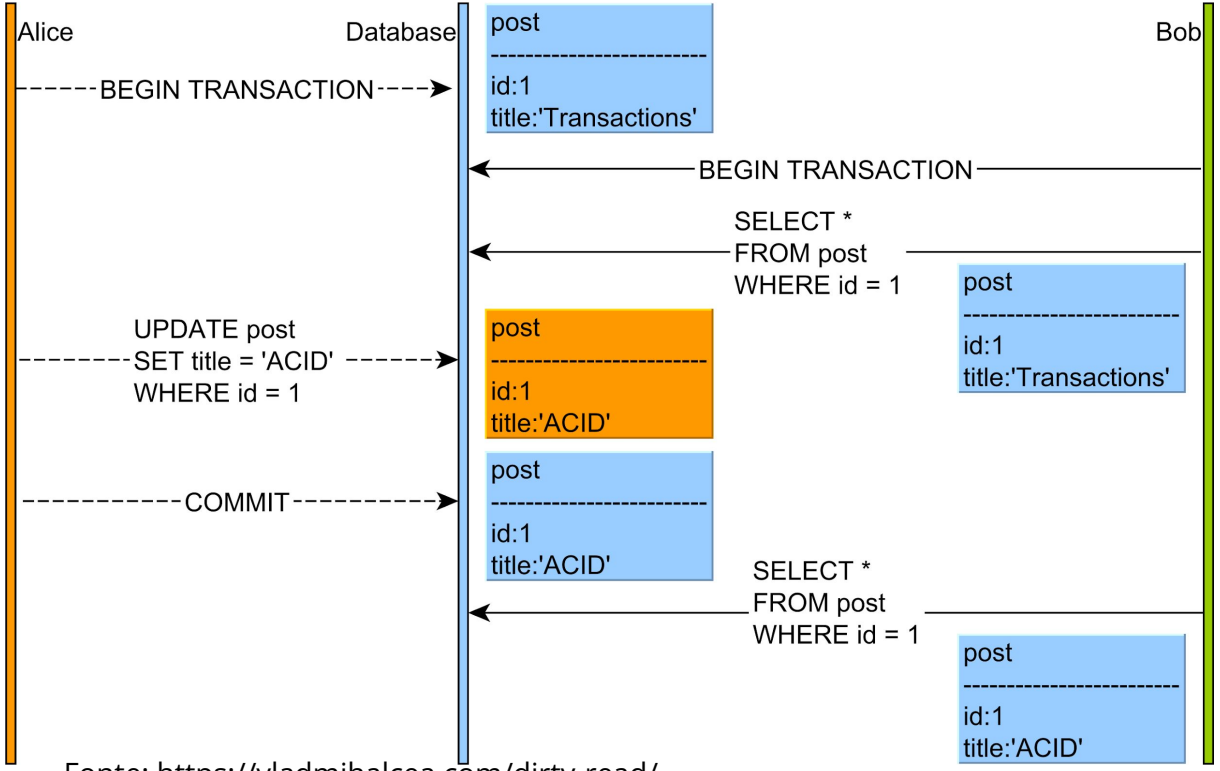
Problemas de Execução Intercalada

2) Leitura não repetível (*Nonrepeatable Read*):

- transação T1 lê um dado
- esse dado é modificado por uma transação T2 que começou depois de T1
- T1 é efetivada
- se T2 tentar reler o mesmo dado, obterá valores diferentes (*nonrepeatable read*)

Problemas de Execução Intercalada

Leitura não repetível (*Nonrepeatable Read*):



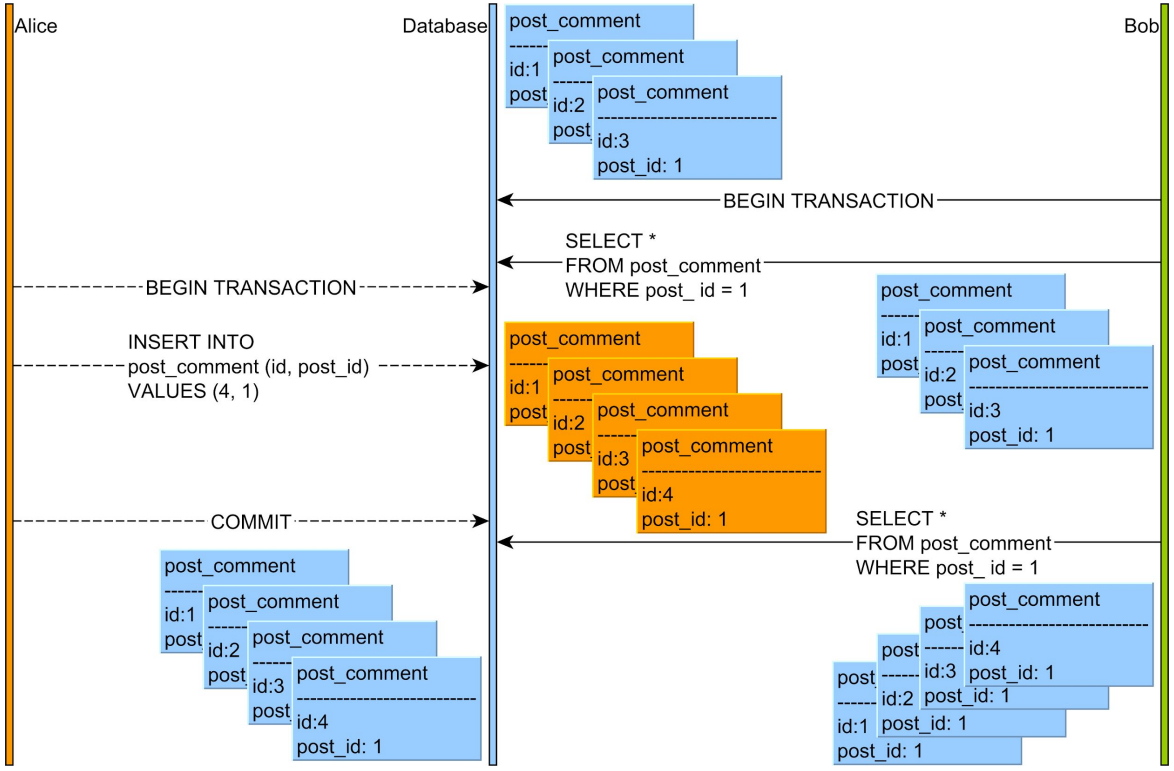
Fonte: https://vladmihalcea.com/dirty-read/

Problemas de Execução Intercalada

3) Leitura fantasma (*Phantom Read*):

- transação T1 lê um conjunto de tuplas que atendam a uma condição de consulta
- transação T2 **insere/remove/atualiza** uma tupla que atenderia a essa condição e é efetivada
- se T1 refizer a mesma consulta, obterá um conjunto diferente de tuplas (*phantom read*)

Problemas de Execução Intercalada



Fonte: https://vladmihalcea.com/dirty-read/

Problemas de Execução Intercalada

Repeatable read vs Phantom read

→ *Repeatable read:* lê valores diferentes de **um** mesmo dado que ainda está lá, mas foi alterado

→ *Phanton read*: lê **conjuntos** de dados diferentes, sendo que um dos conjuntos possui dados que não existem no(s) outro(s) conjunto(s) – fantasmas.

Transações Postgres

```
SET TRANSACTION ISOLATION LEVEL { SERIALIZABLE |
 REPEATABLE READ | READ COMMITTED | READ
 UNCOMMITTED }
 READ WRITE | READ ONLY
```

Níveis de isolamento

	1) Leitura inválida	2) Leitura não repetível	3) Leitura fantasma
Read uncommitted	Sim	Sim	Sim
Read committed	Não	Sim	Sim
Repeatable read	Não	Não	Sim
Serializable	Não	Não	Não

slido

Considere os seguintes comportamentos em transações de banco de dados:Dirty Read | Nonrepeatable Read | Phantom ReadO(s) comportamento(s) possível(eis) no nível de isolamento READ COMMITTED do padrão SQL-92 é(são):

ⓘ Start presenting to display the poll results on this slide.

slido

Assinale a alternativa que contém o nível de isolamento de transação que impede a ocorrência de leituras sujas (dirty reads), leituras fuzzy (nonrepeatable reads), e leituras fantasma (phantom reads).

ⓘ Start presenting to display the poll results on this slide.

Dentre os níveis permitidos, encontra-se o nível de Leitura Confirmada (read committed). Em um SGBD operando em tal nível de isolamento, tentam-se executar duas transações (T1 e T2). Observe na tabela abaixo

T1	Tempo	T2
Begin Transaction	t1	
Read(A)	t2	
...	t3	
...	t4	Begin Transaction
...	t5	Read(A)
...	t6	A = A + 30
...	t7	Write(A)
...	t8	Commit
Read(A)	t9	
A = A – 50	t10	
Write(A)	t11	
Rollback	t12	

A a transação T1 não poderá executar o comando Rollback, pois a transação T2 executou o comando Commit.

B.a transação T1 terá dois resultados distintos (nos tempos t2 e t9) para o mesmo comando de consulta ao registro A..

C.o valor final do registro A, após a execução de ambas as transações T1 e T2, será igual ao valor inicial reduzido de 20.

D.o registro A terá o mesmo valor que tinha antes do início de ambas as transações, devido ao comando Rollback executado por T1, ao final da execução das transações T1 e T2.

E.esse escalonamento não pode ocorrer, pois o nível de isolamento utilizado impede a execução de duas transações em simultâneo, sendo que a transação T2 somente será executada pelo SGBD após o término da transação T1.

Atividade Prática 1 - Deadline 30/11

Usar o dataset reddit_vm para responder às seguintes perguntas:

1- Qual o tempo de execução na inserção de 10000 tuplas com o autocommit True e False? Explique o que aconteceu. OBS: rodar 5 vezes e fazer a média e desvio padrão dos tempos de execução

2- Abra dois terminais e execute, ao mesmo tempo, o código da questão anterior com o autocommit False. Além disso, setar o nível de isolamento SERIALIZABLE. Reportem o tempo (5 execuções com o desvio padrão). Explique o que acontece na prática neste caso?

Entregar em forma de relatório

```
import psycopg2

import time

from psycopg2 import extensions, connect

print ("Opened database successfully")

conn=None

try:

    conn = psycopg2.connect(database="teste", user = "guilherme", password = "1761791", host = "127.0.0.1", port = "5432")

    conn.autocommit = False

    cur = conn.cursor()

    starttime = time.time()

    for i in range(10000):

        cur.execute("insert into employee values (1,'joao',100000)")

        if(i%5000==0):
```

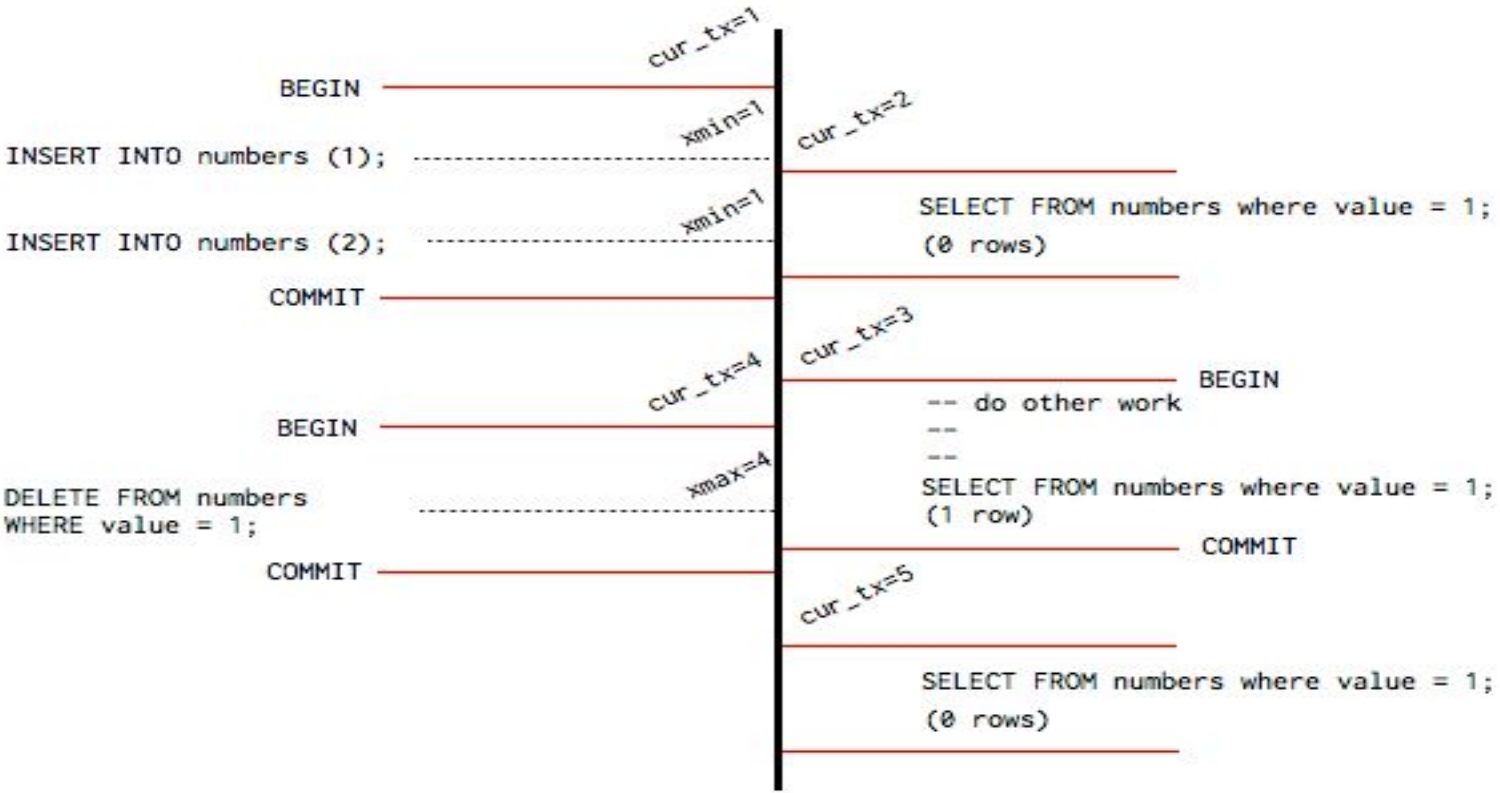
Controle de concorrência Postgres

-No Postgres, um Delete nao apaga o dado de fato

Multiversion Concurrency Control- MVCC

- Baseado no conceito que conflitos são infrequentes
- Deixar executar as transações concorrentemente
- Se ocorrer um conflito, uma das transações é abortada
- Cada transação enxerga sobre uma cópia dos dados e não “lê” alterações de outras transações não comitadas
- Dados são salvos em cópias
- Ao iniciar uma transação é mantido uma lista de todas as outras em progresso.

MVCC


cur_tx=1
BEGIN
xmin=1
INSERT INTO numbers (1);
cur_tx=2
xmin=1
INSERT INTO numbers (2);
SELECT FROM numbers where value = 1;
(0 rows)
COMMIT
cur_tx=3
cur_tx=4
BEGIN
BEGIN
-- do other work
--
--
xmax=4
DELETE FROM numbers
WHERE value = 1;
SELECT FROM numbers where value = 1;
(1 row)
COMMIT
COMMIT
cur_tx=5
SELECT FROM numbers where value = 1;
(0 rows)

MVCC

Variáveis:

- Xmin: armazena a transação que fez a última alteração
- Xmax: reporta se o registro está em processo de remoção

Exemplo 1

A)Crie a tabela teste (id integer, value char(500))

B)Adicione 10 linhas

C) Verifique o tamanho da tabela

```
SELECT pg_size_pretty( pg_total_relation_size('table name') )
```

D) Verifique a posição de cada registro com o comando

```
SELECT ctid,* from teste
```

E) Atualize todas as linhas para o id receber +1

```
Update teste set id=id+1;
```

F) Verifique o tamanho da tabela novamente e a posição de cada registro. **Descreva o que aconteceu.**

Atividade

A) Abra uma transacao A:

   Rode SELECT txid_current()

   Rode SELECT xmin, xmax, ctid, * FROM teste

B) Apague uma tupla em outra transação B (outro terminal)

   SELECT txid_current()

C) Rode SELECT xmin, xmax, ctid, * FROM teste
D) Rode novamente o SELECT xmin, xmax, ctid, * FROM teste. **Explique o que aconteceu?**
E) Faça o mesmo com o comando update, atividade A-D.. **Explique novamente o que aconteceu?**

G) Explique o motivo do Postgres não apagar um dado quando solicitado.

Atividade 3 - para entregar em dupla

A)Crie a tabela teste (id integer primary key, value char(500))

B)Adicione 10 linhas

C) Verifique o tamanho da tabela

D) Insira uma com id =10 em uma transação A (sem comitar)

E) Insira tupla com id =10 em uma transação B (sem comitar)

Explique o que aconteceu já que os dados são operados em cópias diferentes?