



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 correntes conhecidas

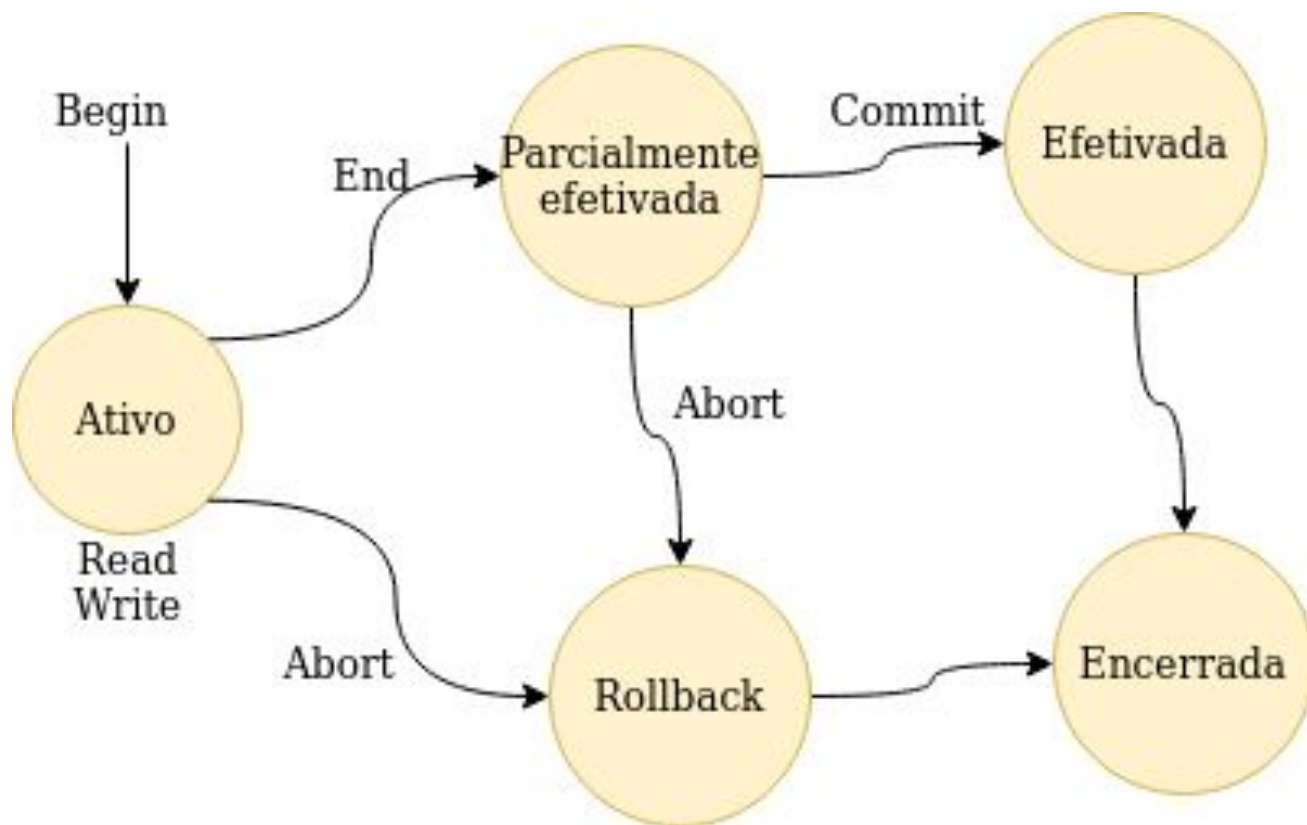
# Introdução

Transação

Fim normal = Commit

Fim anormal= abort (rollback)

# Transações- Estados



# Transações-ACID

- Atomacidade
- Consistência
- Isolamento
- Durabilidade

# Transações-ACID

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

# Transações-ACID

**Atomacidade:** 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

**Atomacidade:** 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

**Atomacidade:** 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 alterações permanecem no banco até que outras transações aconteçam

# Exemplo Transações

- 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 Transações

--Transação A

**begin;**

update trans set saldo = 1200 where id = 1;

**commit;**

--Transação B

**begin;**

select \* from trans where id = 1;

**commit;**

# Transações- exemplo

1. `create table dados (id serial, nome varchar(50));`  
#seleciona o número da transação atual
2. `select txid_current();`  
#acha o local dos dados
3. `select pg_relation_filepath('dados');`  
#mostra o dataset
4. `sudo hexdump -C /var/lib/postgresql/14/main/*`  
#insere dados
5. `insert into dados(nome) values ("fsfsdsdfsd")`  
#faz o salvamento
6. `vacuum full dados;`

# 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()
```

slido



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

① Start presenting to display the poll results on this slide.

slido



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

① Start presenting to display the poll results on this slide.



slido

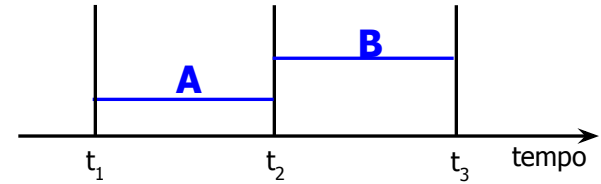


**Quando o hardware falha, a informação deve estar persistida. Qual propriedade ACID que garante isso?**

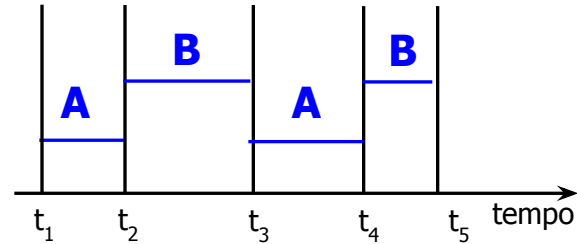
① Start presenting to display the poll results on this slide.

# 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 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

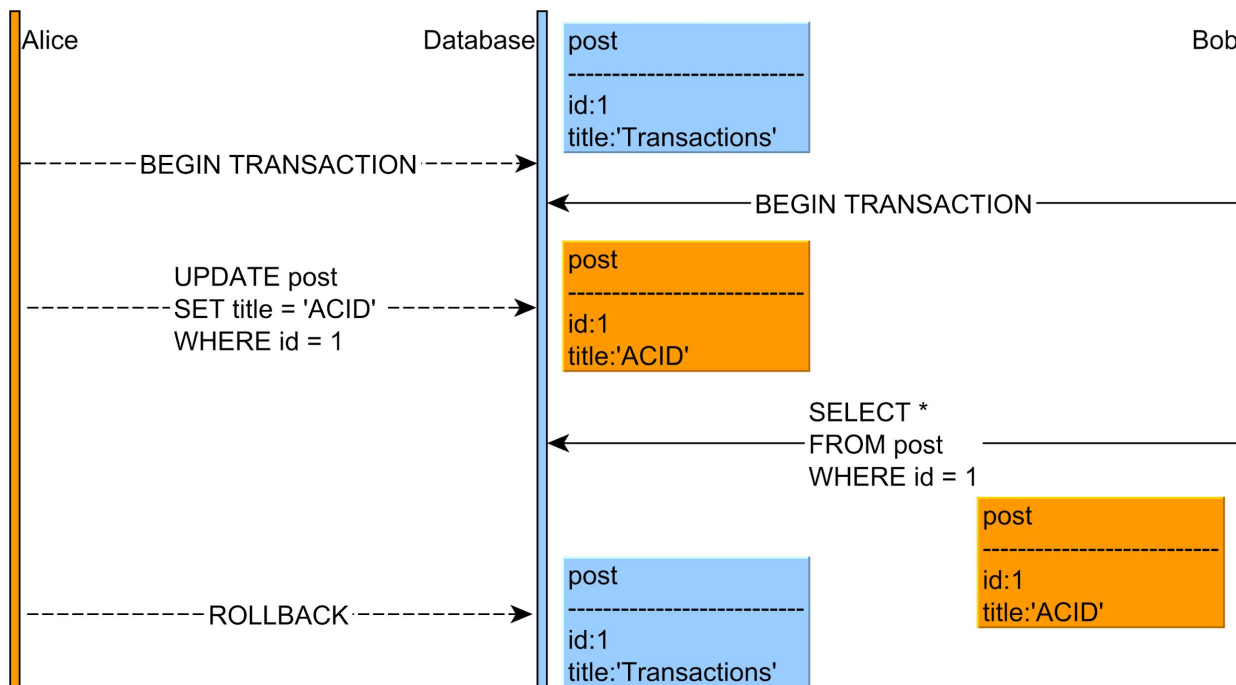
# Problemas de Execução Intercalada

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*):



# Problemas de Execução Intercalada

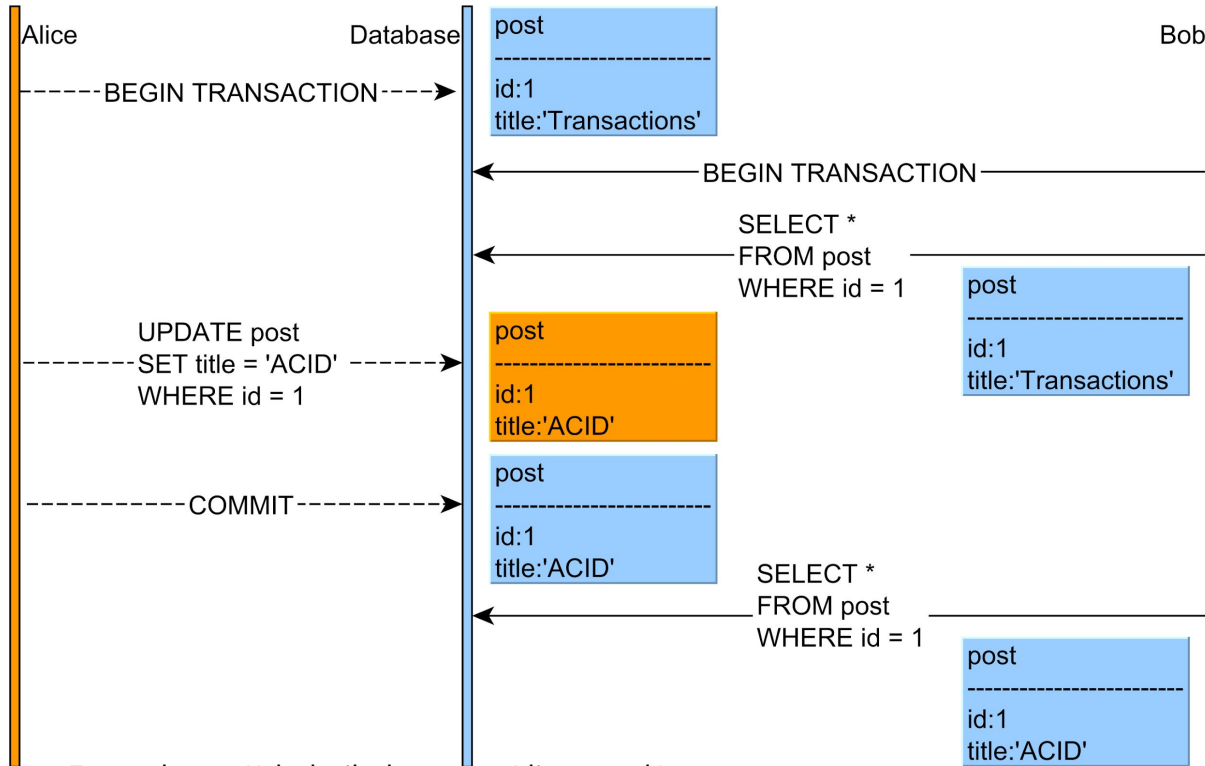
## 2) Leitura não repetível (*Nonrepeatable Read*):

- Ocorre quando uma transação lê duas vezes uma variável com valores diferentes;



# Problemas de Execução Intercalada

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

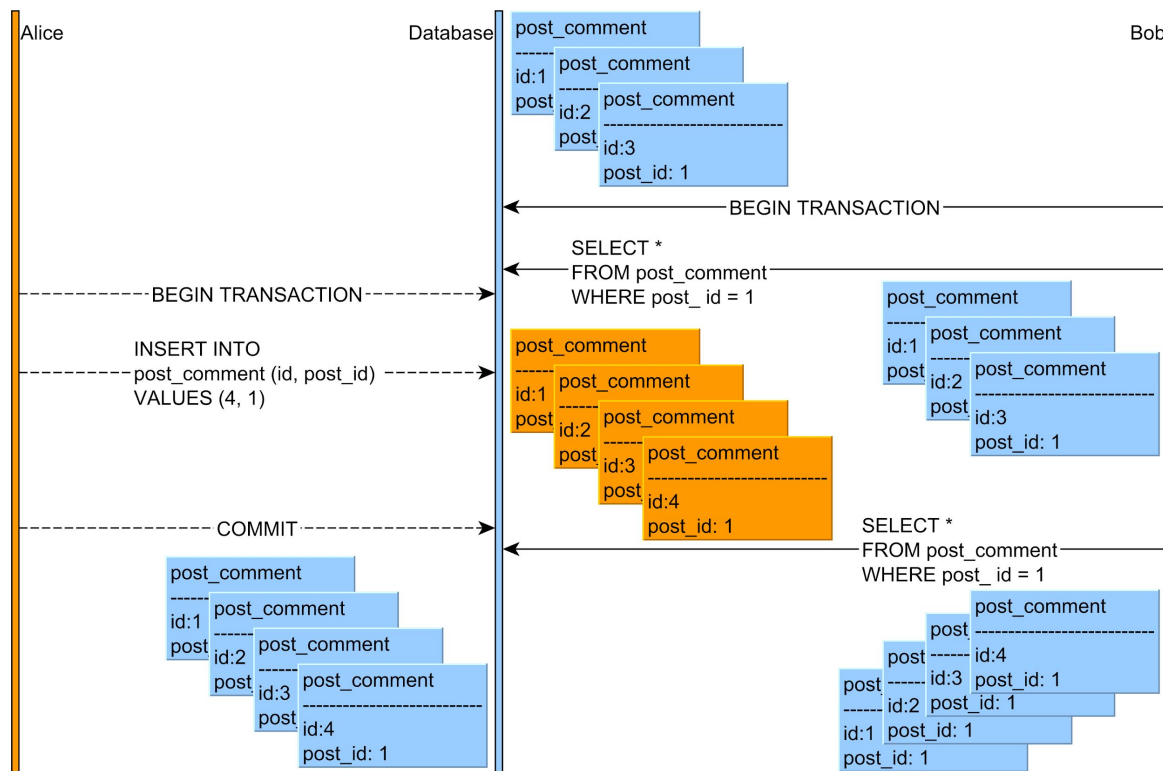


# Problemas de Execução Intercalada

## 3) Leitura fantasma (*Phantom Read*):

- Ocorre quando uma consulta transação quando executada mais de uma vez resulta em diferentes linhas;

# Problemas de Execução Intercalada



# 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
- *Phantom 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
<b>Read committed</b>	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 Read. O(s) comportamento(s) possível(eis) no nível de isolamento READ COMMITTED do padrão SQL-92 é(são):**

① 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).**

① Start presenting to display the poll results on this slide.



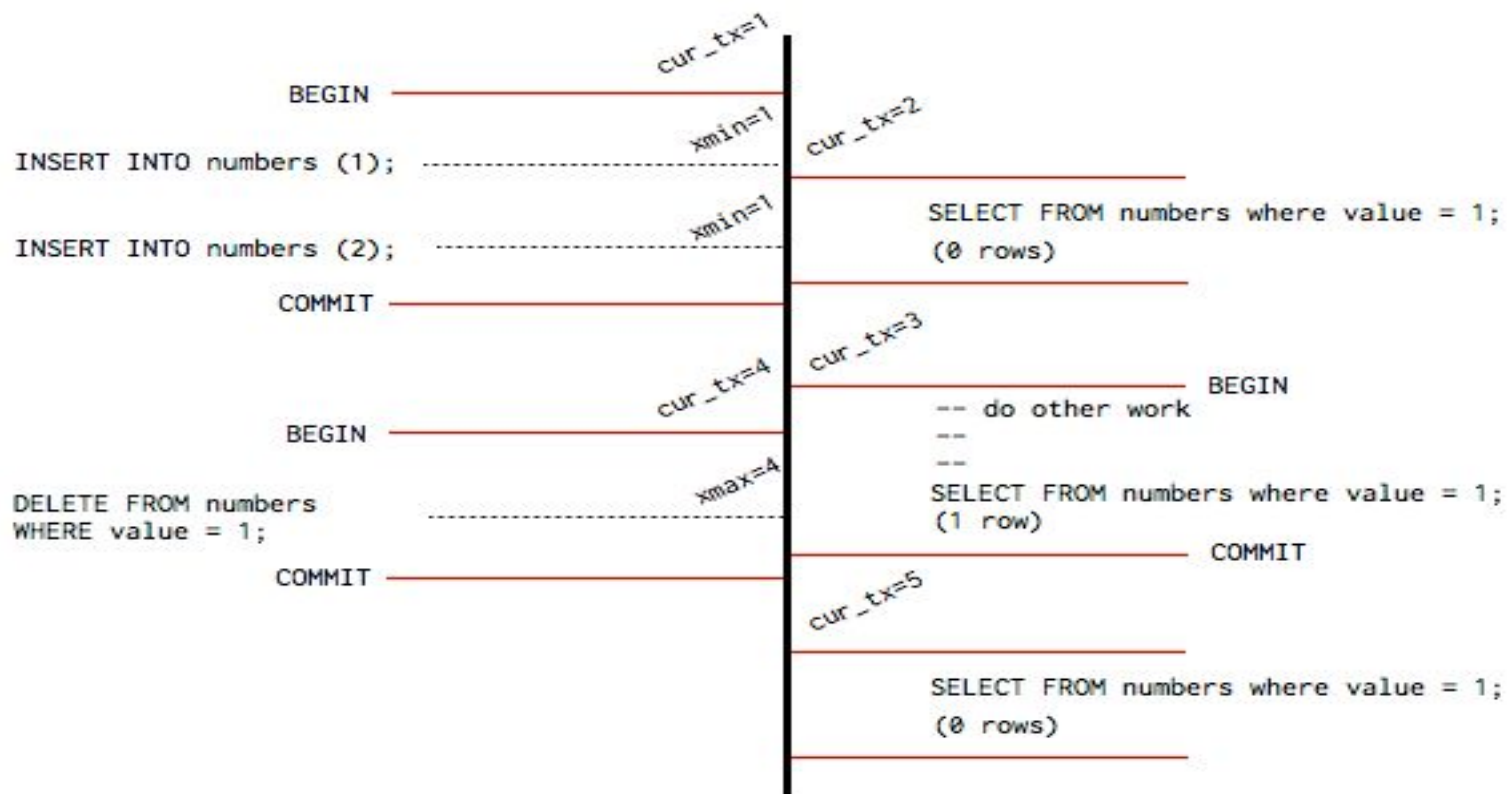
# Controle de concorrência no Postgres

- No Postgres, um Delete não apaga o dado “fisicamente”

# Multiversion Concurrency Control- MVCC

- Baseado no conceito que conflitos são infrequentes
- Deixa 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
- Ao iniciar uma transação é mantida uma lista de todas as outras em progresso.

# MVCC



# 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('teste') )
```

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 1

• Abra um terminal no Postgres:

- `sudo -u postgres psql postgres`
- Crie a seguinte tabela:
  - `cliente(numero serial primary key, 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 tentando acessar dados de outra transação ainda não “comitados”. (abra dois terminais com o postgres).  
O que acontece?

# Atividade 2

- A) Utilize uma transação para inserir o banco de dados Homework no Postgres. Compare os tempos de inserção. Explique o motivo da redução no tempo em relação à inserção sem uma transação.
  
- B) Construa uma aplicação em uma linguagem de programação capaz de executar 1000 inserções usando uma **transação** na tabela. 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 (**rollback**).

Linguagem Livre (menos Python).

Os dados podem ser gerados usando a ferramenta:

<https://www.mockaroo.com/>

# Atividade 3

A) Abra uma transação 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 4

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 tupla 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 com as transações já que os dados são operados em snap diferentes?**

# Atividade Extra

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?

# Código Python

```
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):
```