Na aula anterior

- Views

Na aula de hoje

- Transações

Transações (ou transactions)

- Algumas queries precisam ser agrupadas de tal forma que o SGBD garanta que:
  - Ou todas elas sejam executadas ou nenhuma seja
  - As linhas afetadas não sejam modificadas por queries fora da transação enquanto a mesma está acontecendo

Exemplo

- Suponha um banco de dados onde temos uma tabela conta_corrente em que cada linha é a conta corrente de uma pessoa. Por exemplo

ID	Nome_correntista	Saldo
1	João	100,00
2	Maria	200,00

Exemplo

- Suponha um banco de dados onde temos uma tabela conta_corrente em que cada linha é a conta corrente de uma pessoa. Por exemplo

ID	Nome_correntista	Saldo
1	João	100,00
2	Maria	200,00

- Se João quiser transferir R$10,00 para Maria (dois UPDATE’s) podem acontecer dois problemas

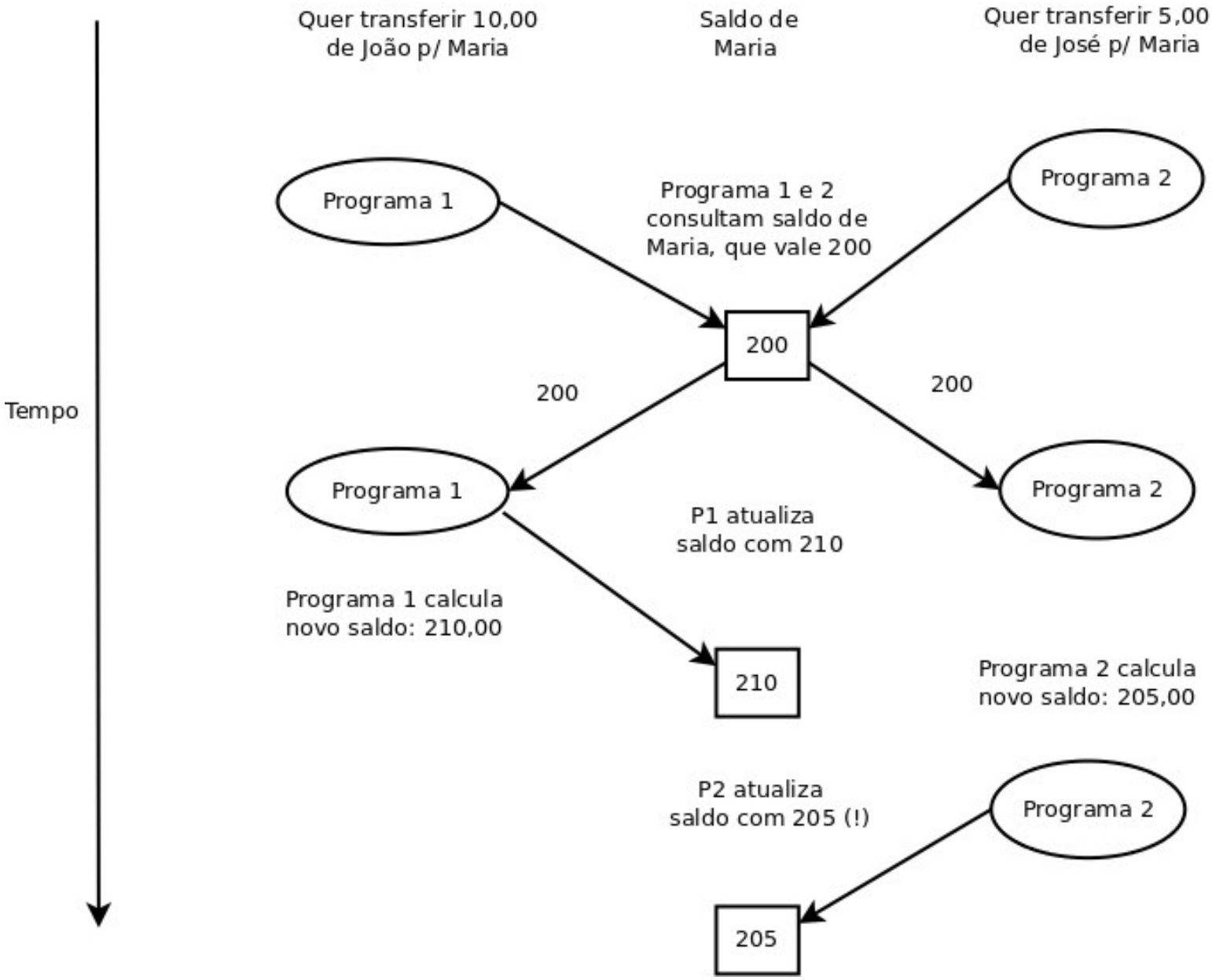
Exemplo

- Suponha um banco de dados onde temos uma tabela conta_corrente em que cada linha é a conta corrente de uma pessoa. Por exemplo

ID	Nome_correntista	Saldo
1	João	100,00
2	Maria	200,00

- Se João quiser transferir R$10,00 para Maria (dois UPDATE's) podem acontecer dois problemas
  - a) Pode haver algum erro (ex: falta de luz) antes do segundo UPDATE
  - b) Pode ser que dois programas estejam acessando os mesmos registros, ao mesmo tempo

Problema de concorrência



Transações

- Para evitar esse tipo de problema (que pode ser raro em bancos pequenos-médios mas tem um grande impacto) existe o conceito de transação
  - Com elas agrupamos comandos SQL em um único grupo (esse grupo é chamado de transação): ou essa transação é totalmente executada ou não é executada, e as linhas envolvidas ficam temporariamente indisponíveis

Sintaxe básica

```
-- Demarca o começo de uma transação
BEGIN [TRANSACTION|WORK];

COMANDOS SQL ...

-- Efetiva as modificações na base de dados
COMMIT [TRANSACTION|WORK];
```

Exemplo

```
BEGIN;

UPDATE conta_corrente SET saldo = 90
WHERE id = 1;

UPDATE conta_corrente SET saldo = 110
WHERE  id = 2;

COMMIT;
```

Rollbacks e Savepoints

- Se acontecer algo no meio da transação e quisermos cancelá-la, é possível
  - Para isso ao invés do comando COMMIT usamos o comando ROLLBACK
- É possível voltar para ‘checkpoints’ no meio da transação. Esses checkpoints são chamados de savepoints no PostgreSQL

Rollbacks e Savepoints

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
SAVEPOINT my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';
-- oops ... forget that and use Wally's account
ROLLBACK TO my_savepoint;
UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Wally';
COMMIT;
```

Observações finais

- Por padrão o PostgreSQL executa cada comando como se fosse uma transação
- Algumas bibliotecas também enviam sequências de comandos para o SGBD como se fosse uma única transação
- O comando BEGIN das transações (note o ‘;’ ao final) é diferente do BEGIN que inicia um bloco/função em plpgsql
- No PostgreSQL não é necessário usar transações dentro de uma função / trigger. Funções e triggers já se comportam como uma transação

Referências

- Transactions https://www.postgresql.org/docs/current/tutorial-transactions.html
- PostgreSQL transaction tutorial https://neon.tech/postgresql/postgresql-tutorial/postgresql-transaction
- Structure of PL/pgSQL https://www.postgresql.org/docs/17/plpgsql-structure.html
- Último acesso em dezembro de 2024