Introdução

Transações – postgres SQL

Introdução

- Muitas vezes, a realização de uma tarefa requer a execução de várias operações no banco de dados;
- Vamos pensar, por exemplo, na seguinte situação:
  - Germano quer transferir R$ 2.500,00 da sua conta para a conta de Fábio;
  - Para realizar esta tarefa, o SGBD deve:
    - Debitar o valor na conta de Germano ;
    - Creditar o valor na conta de Fábio;

Introdução

- Podemos resolver este problema facilmente, por exemplo, através de um procedimento armazenado;
- No entanto, o que acontecerá se, no meio da execução do procedimento, ocorrer uma falha no sistema;
  - Uma queda de energia, por exemplo;
  - E, pior, se a primeira operação tiver sido executada e a segunda não;
    - Onde vai parar o dinheiro debitado da conta de Germano?

Introdução

- Este tipo de situação pode acontecer em diversos tipos de aplicações;
  - Em um sistema de reserva de passagens;
  - Processamento de cartões de crédito;
  - Mercado de ações;
  - E muitos outros casos;

Para este tipo de situação, foi criado um novo recurso em banco de dados, chamado de **Transação**;

Introdução

Vamos agora pensar em outro tipo de situação:

- Os SGBDs mais novos são, em sua grande maioria, multiusuários;
  - Vários usuários podem se conectar e usar o SGBD simultaneamente;

Estes SGBDs usam um recurso chamado multiprogramação;

- Vários usuários executando programas "ao mesmo tempo";

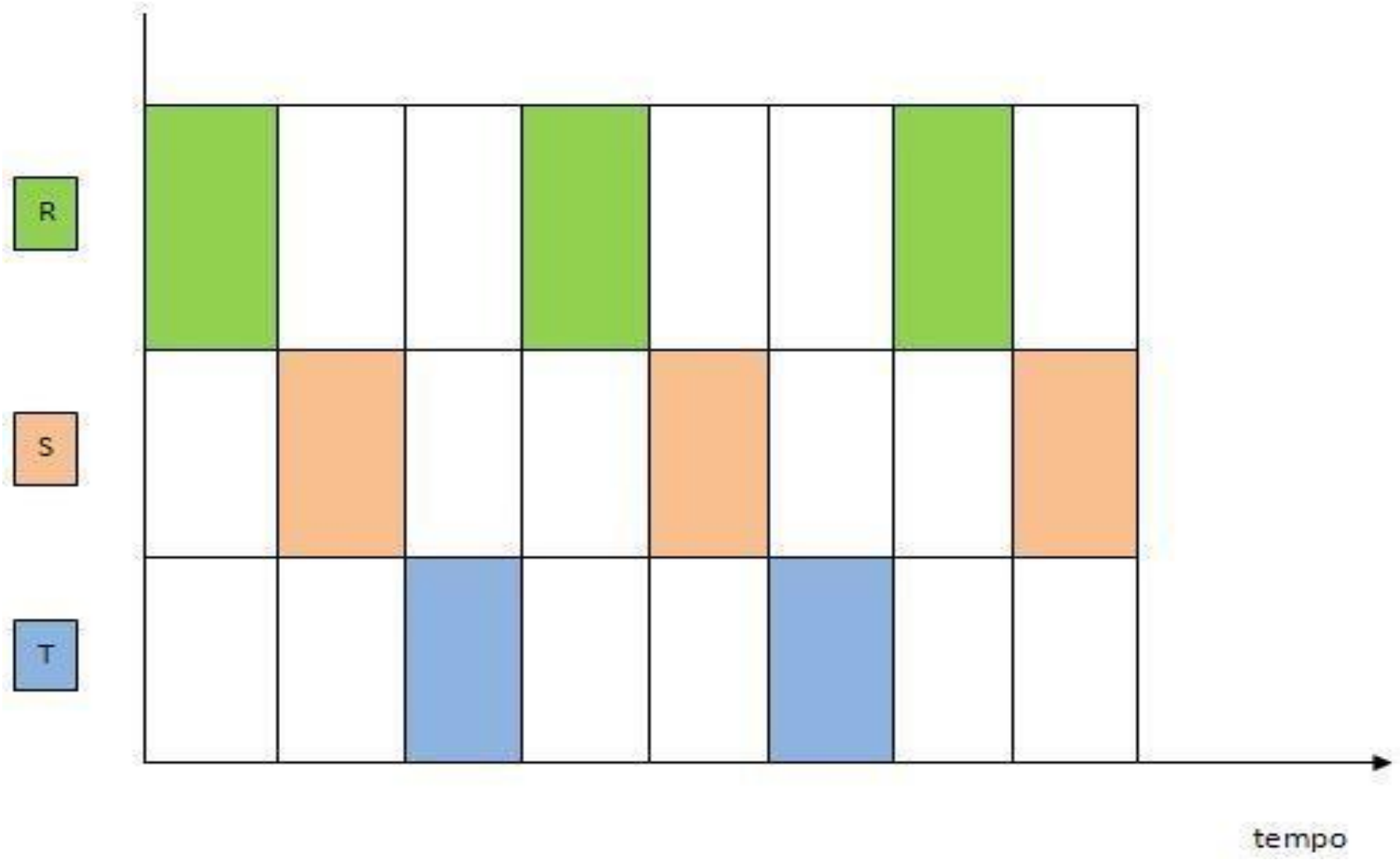
Introdução

- No entanto, uma máquina que possui apenas uma CPU só pode executar uma tarefa de cada vez;
- As tarefas são executadas “simultaneamente” através de um recurso chamado de **compartilhamento de tempo**;
- Ela executa as tarefas de parte em parte;
  - Uma parte de uma tarefa, depois uma parte de outra, depois outra parte de uma outra, [...] ;
  - E depois volta pra primeira tarefa e repete o processo;

Introdução

- Quando ela volta para uma tarefa, ela continua a execução a partir do ponto em que ela parou;
  - Assim, cada tarefa pode passar diversas vezes pela CPU até que a sua execução seja concluída;
- A mudança de uma tarefa para a outra acontece numa velocidade muito grande;
  - O usuário tem a impressão de que todas as tarefas são executadas ao mesmo tempo;
- O próximo slide mostra a execução concorrente de três subprogramas armazenados R, S e T;

Introdução



Introdução

- A multiprogramação é um recurso interessante, mas também pode trazer problemas:
- Vamos supor que um procedimento para reserva de passagens é composta por duas operações:
  - Obter uma poltrona disponível;
  - Reservar a poltrona para o cliente;

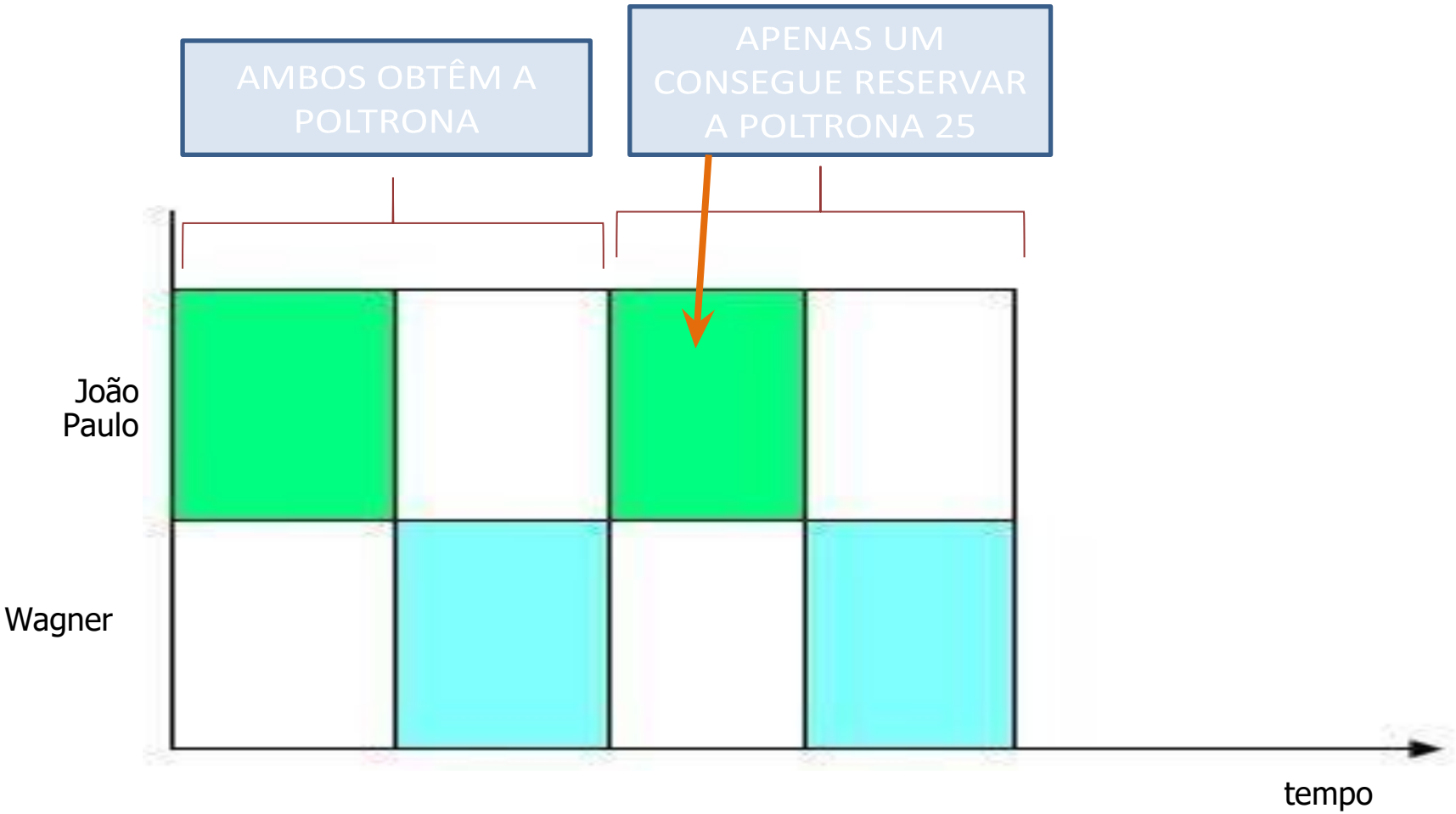
Introdução

Vamos agora supor que:

- João Paulo e Wagner executam este procedimento ao mesmo tempo;
- Só existe uma única poltrona livre;
  - Digamos, a poltrona 25;

Vamos agora analisar o que acontecerá se, devido à multiprogramação, os procedimentos forem executados da forma mostrada no slide a seguir;

Introdução


AMBOS OBTÊM A POLTRONA
APENAS UM CONSEGUE RESERVAR A POLTRONA 25
João Paulo
Wagner
tempo

Introdução

- A execução acontecerá da seguinte forma:
  - João Paulo vai obter a poltrona 25;
  - Wagner também vai obter a poltrona 25;
    - Observando que a operação que reserva a poltrona para João Paulo ainda não foi executada;
  - A poltrona 25 é alocada para João Paulo;
  - Na hora de alocar a poltrona para Wagner:
    - A operação pode sobrescrever a operação de João Paulo e gerar um *overbooking*;
    - A operação pode travar e gerar um erro;

Introdução

Este problema, assim como o da conta bancária, poderia ser evitado se os procedimentos pudessem ser tratados logicamente como se fossem uma única operação;

- Embora eles possam ter várias;

Para solucionar este tipo de problema e oferecer esta visão foi criado o conceito de transações;

n o

- Uma transação é um programa em execução que forma uma unidade lógica de processamento;
- Um transação contém uma série de comandos SQL que realizam alguma operação no banco de dados;
  - É como se fosse um procedimento armazenado com algumas propriedades especiais;
- Uma transação que não atualiza o banco de dados é chamada de transação de leitura;

As propriedades ACID

- Para que transações sejam executadas de forma eficiente, elas devem ter quatro propriedades;
  - Atomicidade;
  - Consistência;
  - Isolamento;
  - Durabilidade;

Os subsistemas de controle de concorrência e de restauração do SGBD são responsáveis por garantir estas propriedades;

As propriedades ACID

Atomicidade:

- Uma transação é uma unidade lógica de processamento;

  Isto implica que, ou ela é totalmente executada, ou não deve ser executada de modo algum;

  - Não existe uma transação parcialmente executada;

  Caso ocorra uma falha durante a execução da mesma, o banco de dados deve ser restaurado para o estado em que ele estava antes da transação ser iniciada;

As propriedades ACID

- Consistência:
  - A execução completa de uma transação deve levar o banco de dados a um estado consistente;
  - As restrições de integridade dos dados devem ser sempre preservadas pela transação;
    - Restrições de domínio, integridade de entidade, integridade referencial, asserções, etc;
  - Caso uma transação leve o banco de dados a um estado inconsistente ela deve ser totalmente desfeita;

As propriedades ACID

- Isolamento:
  - Uma transação deve ser executada de forma isolada das demais;
    - Mesmo que, na maioria dos casos, várias transações sejam executadas “ao mesmo tempo”;
    - Uma transação não pode ser afetada pelos efeitos de uma transação simultânea;
    - É como se todas as transações fossem executadas de forma seqüencial;
    - Isto é garantido pelo sistema de controle de concorrência;

As propriedades ACID

Durabilidade:

- Uma vez que a transação é concluída, os seus resultados devem ser armazenados de forma permanente no banco de dados;

  - Isto porque os dados são armazenados em buffer antes de serem gravados fisicamente no disco;

  - Caso ocorra uma falha na gravação dos dados, o sistema deve ser restaurado para o seu estado anterior ao início da transação;

Ciclo de vida

- Uma transação passa por vários estados durante a sua execução;
- Estes estados são:
  - Ativa;
  - Parcialmente Efetivada;
  - Efetivada;
  - Falha;
  - Encerrada;

Ciclo de vida

- Ativa:
  - Indica que a transação começou a ser executada;
  - É neste estado onde as operações da transação são executadas;
  - Ela permanece neste estado até que todos os seus comandos sejam executados;
    - Ou até que aconteça uma falha;
  - Após este estado, ela pode ir para o estado *Parcialmente Efetivada* ou *Falha*;

Ciclo de vida

- Parcialmente Efetivada:
  - Significa que todos os comandos da transação foram executados, mas seus efeitos ainda não foram assegurados;
  - Neste estado, o SGBD vai trabalhar para garantir a durabilidade dos efeitos da transação;
    - Ou seja, a transação ainda pode falhar mesmo após a execução de todos os seus comandos;
- Efetivada:
  - Indica que todos os efeitos da transação estão garantidos;
    - Ou seja, neste estado, a durabilidade é alcançada;
  - A partir deste ponto, os resultados da transação estão garantidos mesmo se houver uma falha;
  - O ponto de efetivação da transação é chamado de *commit point*;

Ciclo de vida

- Falha:
  - Indica que uma falha aconteceu durante a execução da transação antes dela ser efetivada;
  - Uma falha pode acontecer por diversos motivos;
    - Erros de hardware, como queda de energia, falha de memória ou de acesso ao disco;
    - Erros de software, como uma divisão por zero;
    - Condições de exceção, como um saldo insuficiente ou um cliente inadimplente;
    - Etc;

Ciclo de vida

Falha:

- Quando a transação chega a este estado, o subsitema de restauração atua e desfaz os efeitos das operações executadas antes da falha;

Encerrada:

- Indica que a transação terminou;
  - De forma bem sucedida ou por causa de uma falha;