Projeto de Banco de Dados

Transações em Banco de Dados: conceitos e propriedades

Professor: Alexandre Neves Louzada

Março 2025

Roteiro

FAETERJ-RIO
Faculdade de Educação Tecnológica do Estado do Rio de Janeiro

- Processamento de transações
  - Explicar o conceito de transações em Bancos de Dados.
  - Explicar as propriedades de uma transação.
  - Explicar os estados de uma transação.
  - Descrever os comandos básicos para aplicar uma transação.
  - Explicar os problemas gerados em execuções concorrentes.

Introdução

- Transação (conceitos)
  - É uma unidade lógica de processamento do banco de dados que inclui uma ou mais operações de acesso ao banco de dados (inclusão, exclusão, modificação, recuperação).
  - Sistemas de processamento de transações: sistemas com grandes bancos de dados e centenas de usuários concorrentes que estejam executando transações do banco de dados. Exemplo: sistema bancário, sistema de processamento de cartões de crédito, ...

Introdução

- Transação (conceitos)
  - Exigem alta disponibilidade e tempo rápido de resposta para centenas de usuários concorrentes.
  - Sempre que uma transação é submetida a um SGBD para execução, o sistema é responsável por certificar-se de que:
    - Ou todas as operações na transação se completam com sucesso e seu efeito registrado permanentemente no banco de dados;
    - Ou A transação não tem absolutamente nenhum efeito no banco de dados ou em quaisquer outras transações.

Propriedades

- Propriedades de uma transação
  - As transações devem ter diversas propriedades que são chamadas propriedades ACID e devem ser impostas pelos métodos de controle de concorrência e recuperação do SGBD. As propriedades ACID são as seguintes:
    - Atomicidade
    - Consistência
    - Isolamento
    - Durabilidade

Propriedades

- ACID
  - Atomicidade
    - **Uma transação é uma unidade atômica de processamento; é realizada integralmente ou não é realizada de modo algum.**
    - Observação: em certas situações é interessante se agregar vários comandos como sendo integrantes de uma mesma transação, como, por exemplo, em uma transferência bancária que envolve a retirada de dinheiro de uma conta e o acréscimo em outra como se fosse apenas uma única operação lógica.

Propriedades

- ACID
  - Consistência
    - **Uma transação é preservadora de consistência se a sua plena execução levar o banco de dados de um estado consistente para outro estado também consistente.**

Propriedades

- ACID
  - Isolamento
    - **Uma transação deve parecer como se estivesse sendo executada isoladamente de outras transações. Ou seja, a execução de uma transação não deve sofrer interferência de quaisquer outras transações que estejam sendo executadas concorrentemente.**

Propriedades

- ACID
  - Durabilidade
    - **As alterações aplicadas ao banco de dados por meio de uma transação confirmada devem persistir no banco de dados. Essas alterações não devem ser perdidas por falha alguma.**

Estados

- Estados de uma transação
  - Na ausência de falhas, todas as transações são completadas com sucesso. Porém, uma transação nem sempre pode completar sua execução com sucesso. Caso isso ocorra, essa transação é considerada abortada.

Estados

- Estados de uma transação
  - Se tivermos que garantir a propriedade de atomicidade, uma transação abortada não pode ter efeito sobre o estado do banco de dados. Assim, qualquer mudança que a transação abortada tenha feito no banco de dados deve ser desfeita. Quando as mudanças causadas por uma transação abortada tiverem sido desfeitas, dizemos que a transação foi revertida.

Estados

- Estados de uma transação
  - Uma transação que completa sua execução com sucesso é considerada confirmada. Uma transação confirmada que realizou atualizações transforma o banco de dados em um novo estado consistente, que precisa persistir mesmo que haja uma falha no sistema. Quando uma transação tiver sido confirmada, não podemos desfazer seus efeitos abortando-a. A única forma de desfazer os efeitos de uma transação confirmada é executar uma transação de compensação.

Estados

- Estados de uma transação
  - Em resumo, uma transação precisa estar em um dos seguintes estados:

    Ativa: é o seu estado inicial. A transação permanece nesse estado enquanto está sendo executada.

    Parcialmente confirmada: é o estado depois que a instrução final foi executada.

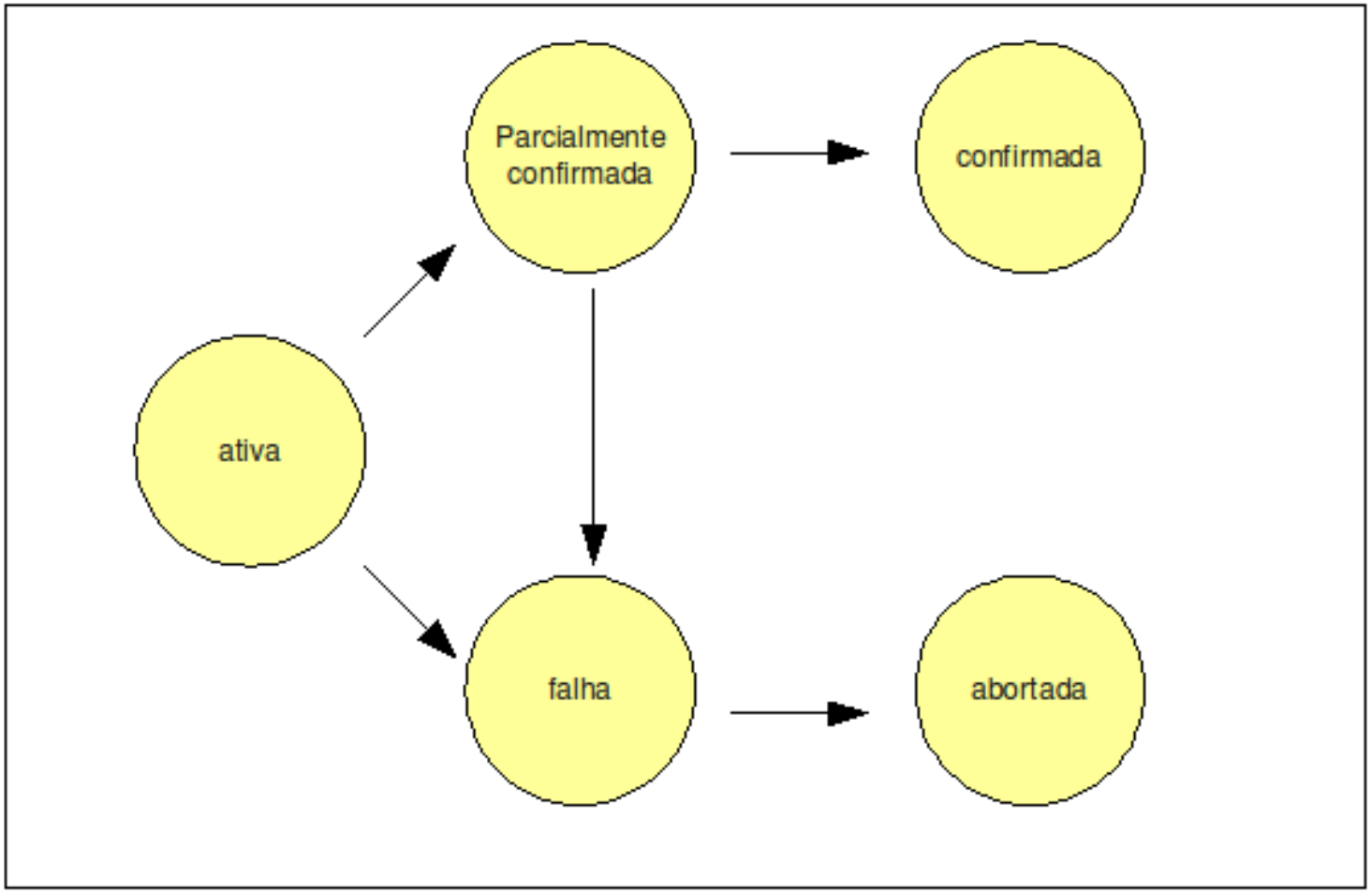
    Falha: é o estado depois da descoberta de que a execução normal não pode mais prosseguir.

    Abortada: estado depois que a transação foi revertida e o banco de dados foi restaurado ao seu estado anterior ao início da transação.

    Confirmada: estado após o término bem-sucedido.

Estados

- Estados de uma transação



Estados

- Diferentemente dos SGBD's tradicionais, que usam bloqueios para controlar a simultaneidade, o PostgreSQL mantém a consistência dos dados utilizando o modelo multi-versão (Multiversion Concurrency Control, MVCC).

- Isto significa que ao consultar o banco de dados, cada transação enxerga um estado do banco de dados, ou seja, como este era há um tempo atrás, sem levar em consideração o estado corrente dos dados subjacentes. Este modelo protege a transação para não enxergar dados inconsistentes, o que poderia ser causado por atualizações feitas por transações simultâneas nas mesmas linhas de dados, fornecendo um isolamento da transação para cada sessão do banco de dados.

Comandos

- BEGIN - -iniciar
- - - comandos
- COMMIT - -comitar/confirmar
- ROLLBACK - -parar/cancelar
- END - -mesma função do COMMIT

Comandos

```
 CREATE TABLE "CONTA"
(
  "ID" integer NOT NULL,
  "NOME" character varying(255) NOT NULL,
  "SALDO" numeric(15,2) DEFAULT 0,
  CONSTRAINT "CONTA_pkey" PRIMARY KEY ("ID")
)
```

Comandos

```
--INSERINDO DADOS NA TABELA
INSERT INTO "CONTA” ("ID", "NOME", "SALDO")
    VALUES (1,'JOAO',1000),
           (2,'PEDRO',1500),
           (3,'MARIO',450),
           (4,'JOAQUIM',40000);
```

Comandos


```
--BEGIN (implícito)
UPDATE conta SET saldo = 100 WHERE id = 1;
--COMMIT (implícito)
```

Ao realizar o comando acima implicitamente está sendo usado BEGIN e COMMIT, isso porque obrigatoriamente todos os comandos são executados dentro de uma transação, independente do seu tamanho (seja uma linha ou mil linhas).

Comandos

```
SELECT * FROM "CONTA" WHERE "ID" = 1;
BEGIN;
UPDATE "CONTA" SET "SALDO" = 120 WHERE "ID" = 1;
ROLLBACK;
SELECT * FROM "CONTA" WHERE "ID" = 1;
```

A primeira linha mostra os dados da consulta antes de iniciar a transação. Em seguida uma transação é iniciada e o saldo da conta “1” é atualizado. Em seguida é executado um rollback.

O resultado das execuções acima não tem nenhum efeito nos dados do SGBD, isso porque o rollback anulou todas as alterações e isso significa que o resultado do primeiro SELECT será igual ao resultado do último SELECT.

Comandos

```
BEGIN;
UPDATE "CONTA" SET "SALDO" = 150 WHERE "ID" = 1;
SAVEPOINT savepoint_1;
UPDATE "CONTA" SET "SALDO" =10000 WHERE "ID" = 2;
SELECT * FROM "CONTA" WHERE "ID" = 2;
ROLLBACK TO savepoint_1;
SELECT * FROM "CONTA" WHERE "ID" = 2;
COMMIT;
```

-- SAVEPOINT possibilita salvar um determinado ponto dentro de uma transação e voltar para ele quando achar necessário.

Comandos

A transação é iniciada e um update na conta 1 é executado, atualizando o saldo para 120;

Em seguida é criado um savepoint chamado savepoint_1, assim temos agora um ponto salvo em nossa transação e podemos voltar a ele quando necessário;

O saldo da conta 2 é atualizado para 10000 e depois executamos um SELECT para checar como ficou;

Comandos

Percebemos que o UPDATE que fizemos está errado, na conta 2, e queremos desfazer só ele, evitando a perda de toda a transação. Sendo assim, usamos o comando “rollback to savepoint_1”, que nos leva exatamente ao momento anterior ao update da conta 2;

Agora executamos o SELECT na conta 2 e verificamos que o valor do saldo já estava como antes;

Executamos o commit para confirma as alterações.

Isolamento

- Em cenários que trabalham com tarefas concorrentes precisamos pensar em como isolar determinadas transações, não permitindo que uma alteração de dados na transação "**A**" afete a execução da transação "**B**".
- Os níveis de isolamento só existem porque existem problemas ao trabalhar-se com transações concorrente, e antes de vermos quais são esses níveis e como aplicá-los, nós temos que entender que tipo de problemas podem ocorrer ao trabalhar em multithread.

Isolamento

- São três os possíveis problemas:
  - Dirty Read (Leitura Suja)
  - Nonrepeatable Read (Leitura que não pode ser repetida)
  - Phantom Read (Leitura Fantasma)

Isolamento

- Dirty Read (Leitura Suja):
  - Ocorre quando uma transação ”**A**” altera um determinado valor e uma transação ”**B**” lê esse valor alterado.
  - A transação ”**A**” executa um rollback, ou seja, não confirma a persistência dos dados alterados, porém a transação ”**B**” lê esse valor alterado.
  - Em outras palavras, uma transação lê dados escritos por uma transação simultânea não efetivada (uncommitted).
  - Observação: a maioria dos SGBD, incluindo o PostgreSQL ,não permitem, em nenhuma circunstância, que isso ocorra.

Isolamento

- Exemplo

No PgAdmin abra duas janelas de consulta e execute cada uma em uma janela separada.

```
--TRANSAÇÃO A
begin;
UPDATE "CONTA" SET "NOME" = 'JOAO II' WHERE "ID" = 1;
SELECT * FROM "CONTA";
Rollback;
```

Isolamento

- Exemplo

```
--TRANSAÇÃO B
begin;
SELECT "NOME" FROM "CONTA" WHERE "SALDO" >= 2000;
SELECT "ID" FROM "CONTA" WHERE "SALDO" < 2000;
SELECT “NOME”  FROM "CONTA“ WHERE  “ID” = 2;
SELECT “SALDO”  FROM "CONTA“ WHERE  “ID” = 3;
UPDATE "CONTA" SET "SALDO" = 300 WHERE "NOME" = 'JOAO';
commit;
```

Isolamento

- Exemplo (Imaginando que o PostgreSQL aceite Dirty Read)

1. Ambas as transações **A** e **B** são iniciadas.

2. A transação **A** executa primeiramente um UPDATE no cliente 'JOAO' mudando seu nome para 'JOAO II'.

3. Como a transação **B** executa vários comandos antes de chegar no seu UPDATE, quando ela chegar neste ponto o UPDATE da transação **A** já foi feito e quando a transação **B** tentar fazer um UPDATE pelo cliente 'JOAO' nada será feito pois o nome do cliente agora é 'JOAO II'.

4. Mesmo a transação **A** executando um rollback a transação **B** já está com o dado errado e a inconsistência de dados ocorre.

Observação: como o PostgreSQL não permite que isso ocorra então ao realizar o UPDATE na transação **B** tudo funcionará normalmente visto que houve um rollback na transação **A**.

Isolamento

- Exemplo (Imaginando que o PostgreSQL aceite Dirty Read)

1. Ambas as transações **A** e **B** são iniciadas.

2. A transação **A** executa primeiramente um UPDATE no cliente 'JOAO' mudando seu nome para 'JOAO II'.

3. Como a transação **B** executa vários comandos antes de chegar no seu UPDATE, quando ela chegar neste ponto o UPDATE da transação **A** já foi feito e quando a transação **B** tentar fazer um UPDATE pelo cliente 'JOAO' nada será feito pois o nome do cliente agora é 'JOAO II'.

4. Mesmo a transação **A** executando um rollback a transação **B** já está com o dado errado e a inconsistência de dados ocorre.

Observação: como o PostgreSQL não permite que isso ocorra então ao realizar o UPDATE na transação **B** tudo funcionará normalmente visto que houve um rollback na transação **A**.

Isolamento

- Nonrepeatable Read:
  - A transação lê novamente dados lidos anteriormente, e descobre que os dados foram alterados por outra transação (que os efetivou após ter sido feita a leitura anterior).
  - No Dirty Read tínhamos um dado que foi alterado e depois não foi confirmado causando uma inconsistência a uma outra transação que o lê, porém neste caso os dados alterados são confirmados, através do commit. No Nonrepeatable read dados alterados e confirmados são “sentidos” pela outra transação que ainda não foi finalizada.

Isolamento

- Nonrepeatable Read (exemplo):

--Execute em instâncias diferentes!

--TRANSAÇÃO A

```
begin;
UPDATE "CONTA" SET "NOME" = 'JOAO II' WHERE "ID" = 1;
commit;
```

--Transação B

```
begin;
UPDATE "CONTA" SET "SALDO" = 350 WHERE "NOME" = 'JOAO II';
commit;
```

Isolamento

- Nonrepeatable Read (exemplo):

1. Execute o begin de ambas as transações para torná-las concorrentes;
2. Execute o UPDATE da transação **A** para mudar o nome de JOAO para JOAO II;
3. Agora execute o UPDATE da transação **B** para tentar mudar o saldo de JOAO II para 350. Você verá a seguinte mensagem: 0 rows affected. Isso ocorreu porque a transação **A** ainda não foi comitada e até o momento “JOAO” ainda não é “JOAO II”, se a atualização ocorresse então teríamos um caso de Dirty Read;
4. Agora execute o COMMIT da transação **A**;
5. Depois execute novamente o UPDATE da transação **B**, agora você verá que 1 registro foi alterado (1 rows affected), isso porque a transação **A** já foi comitada/confirmada.

Isolamento

- Phantom Read (leitura Fantasma):

Quando um dado é inserido em uma transação **A** e esta transação é comitada/confirmada, então este dado pode ser lido por uma transação **B** que ainda não foi confirmada.

A lógica do Phantom Read é quase a mesma do Nonrepeatable read, tendo apenas uma única diferença: as inserções de novos registros.

Isolamento

- Phantom Read (exemplo):

```
--Execute em instâncias diferentes!
--Transação A
begin;
INSERT INTO "CONTA"("ID", "NOME", "SALDO")
VALUES (5,'Cristiano fernandes', 503);
commit;

--Transação B
begin;
SELECT * FROM "CONTA";
commit;
```

Isolamento

- Phantom Read (exemplo):

1. Inicie ambas as transações com o “begin”.

2. Execute o insert do novo registro na transação **A.**

3. Execute o SELECT na transação **B**. teremos o seguinte retorno: 0 rows retrieved. Isso ocorre porque a transação A ainda não foi confirmada e voltamos a dizer que Dirty Read não são permitidas, seja para alterações, deleções ou inserções.

4. Execute o commit da transação **A**.

5. Agora execute o SELECT da transação **B**. Seu retorno será 1 linha contendo os valores do novo registro inserido na transação **A**: 5;“Cristiano fernandes";503.00

Isolamento

- Níveis de Isolamento
  - Os níveis de isolamento configuram quais “problemas” podem e não podem ocorrer, é como dizer ao SGBD que você deseja que ele permita que ocorra um Dirty Read, ou um Nonrepeatable read ou um Phantom Read.
  - Como já vimos não é possível permitir o Dirty Read no PostgresSQL, mas é possível controlar os outros “problemas”, permitindo ou não. Usamos aspas na palavra problemas pois neste caso vale ao DBA considerar se este é um problema ou não.

Isolamento

- Níveis de Isolamento
  - Existem quatro níveis de isolamento segundo o padrão SQL
    - Read Uncommited,
    - Read Commited,
    - Repeatable Read,
    - Serializable.

Isolamento

- Níveis de Isolamento – Read Uncommited

  Este é o nível menos isolado e o como o próprio nome já sugere, ele permite a leitura antes da confirmação. É exatamente o caso do Dirty Read que estudamos logo no início. Neste nível de isolamento todos os problemas podem ocorrer sem restrição.

  É muito difícil que esse nível seja aplicado na prática, pois poderíamos ter sérios problemas de consistência, por isso ele é considerado mais acadêmico, apenas para fins de estudos. O PostgreSQL não possui esse nível de isolação, evitando assim que este seja configurado.

Isolamento

- Níveis de Isolamento – Read Commited

Neste nível de isolamento não podem ocorrer Dirty Reads mas são permitidos Nonrepeatable reads e Phantom Reads. Este é o nível padrão do PostgresSQL.

Isolamento

- Níveis de Isolamento – Repeatable Read

Aqui apenas ocorrem Phantom Reads. O SGBD bloqueia o conjunto de dados lidos de uma transação, não permitindo a leitura de dados alterados ou deletados mesmo que comitados pela transação concorrente.

É permitida a leitura de novos registros comitados por outras transações.

Isolamento

- Níveis de Isolamento – Serializable

  Este é o nível mais isolado que não permite nenhum tipo de problema (Dirty Read, Nonrepeatable read e Phantom Read).

  Este nível emula a execução serial das transações, como se todas as transações fossem executadas uma após a outra, em série, em vez de simultaneamente. Entretanto, os aplicativos que utilizam este nível de isolamento devem estar preparados para tentar executar novamente as transações, devido a falhas de serialização.

Isolamento

- Isolamento X Problemas

Nível de isolamento	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possível	Possível	Possível
Read committed	Impossível	Possível	Possível
Repeatable read	Impossível	Impossível	Possível
Serializable	Impossível	Impossível	Impossível