Criação de Banco de Dados: Transações em Databases

No módulo 3, aprendemos uma série de comandos SQL envolvendo desde a criação de tabelas até operações de manipulação de dados, tais como inserções, atualizações e exclusões. Basicamente, um comando era codificado e submetido ao SGBD PostgreSQL, que em seguida o executava e devolvia algum resultado.

Perceba que, sob o contexto dos exemplos que estudamos, poderíamos concluir que, de certa maneira, havia somente um usuário acessando a totalidade dos recursos do SGBD.

No entanto, em um ambiente de produção, o SGBD gerencia centenas de requisições das aplicações. Com isso, concluímos que há acesso simultâneo a vários recursos que são gerenciados pelo SGBD.

É uma situação típica sobre a qual o SGBD precisa prover uma forma de realizar diversas operações como uma unidade lógica de processamento. Vamos aprender, então, que essa unidade de processamento é denominada transação.

Em sistemas de banco de dados, uma transação corresponde a um programa em execução que forma uma unidade de trabalho.

Os limites de uma transação são especificados por meio dos comandos **BEGIN** transaction (que indica o início de uma transação) e **END** transaction (que indica o término de uma transação) em um programa de aplicação.

Ainda se a transação não altera o banco de dados, é chamada de **somente leitura**; Caso contrário é chamada de **leitura-gravação**;

Em se tratando de SGBDs multiusuários, várias transações podem ser executadas simultaneamente. No entanto, caso essa execução ocorra de maneira descontrolada, poderão surgir problemas de inconsistências, tais como:

Atualização perdida:

Quando duas transações que acessam os mesmos itens de dados têm operações intercaladas de modo a tornar incorretos alguns itens do banco de dados.

Atualização temporária:

Quando uma transação atualiza um item do banco de dados e, depois, falha por algum motivo, enquanto, nesse meio tempo, o item é lido por outra transação antes de ser alterado de volta para seu valor original.

Resumo incorreto:

Quando uma transação calcula uma função de resumo de agregação em uma série de itens enquanto outras transações atualizam alguns desses itens.

Leitura não repetitiva:

Quando uma transação lê o mesmo item duas vezes e o item é alterado por outra transação entre as duas leituras.

Se uma transação for cancelada, deve ser executado um processo denominado ***rollback***, o qual força o SGBD a trazer de volta os valores antigos dos

registros antes da transação ter iniciado. Finalmente, caso a transação seja executada com sucesso, as atualizações devem ser confirmadas por meio do comando ***commit***.

Falhas que podem ocorrer durante o processo de transação do BDD:

Falha do computador

Erro de transação ou sistema

Condições de exceção detectadas pela transação

Falha de disco, problemas físicos e catástrofes

Propriedades de uma transação

Com objetivo de garantir a integridade dos dados contidos no banco de dados, o SGBD mantém um conjunto de integridade das transações. Que é denominado **ACID(*Atomicity, Consistency, Isolation, Durability*);**

Atomicidade: A transação precisa ser realizada completamente ou não realizada;

Consistência**:** A transação deve levar o banco de dados de um estado consistente para outro;

Integridade**:** A execução de uma transação não deve ser interferida por quaisquer outras transações;

Durabilidade As mudanças no banco de dados em função da confirmação da transação devem persistir;

Estados de uma transação

Ativo

Ocorre imediatamente após o início da transação, podendo executar operações de leitura e gravação.

Parcialmente confirmado

Quando a transação termina.

Confirmado

Após verificação bem-sucedida, as mudanças são gravadas permanentemente no banco de dados.

Falha

Se uma das verificações falhar ou se a transação for abortada durante seu estado ativo.

Confirmado

Transação sai do sistema.

Para poder recuperar-se de falhas que afetam transações, normalmente, o SGBD mantém um log para registrar todas as operações de transação que afetam os vários itens do banco de dados. As entradas em um registro de log de uma determinada transação possuem informações de valores antigos e novos do item de dados do

banco de dados, bem como se a transação foi concluída com sucesso ou se foi abortada.

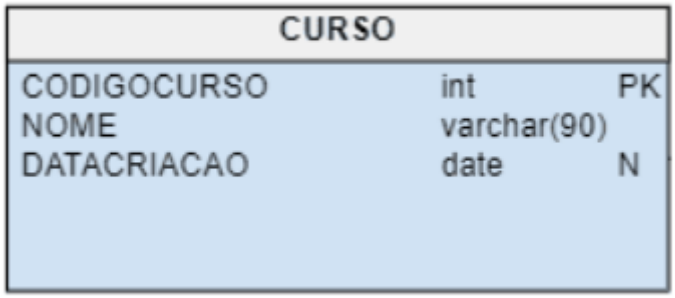
TRANSAÇÕES NO POSTGRESQL

De maneira geral, uma transação no PostegreSQL séria assim:

```
BEGIN-- início da transação
-- comandos
COMMIT-- transação confirmada
RO LLBACK-- transação cancelada
END-- mesma função do COMMIT
```

No entanto, nos SGBDs, a inicialização de uma transação ocorre implicitamente quando executamos alguns comandos.

Exemplo:



A execução de um INSERT na tabela ocorre dentro do contexto implícito de uma transação:

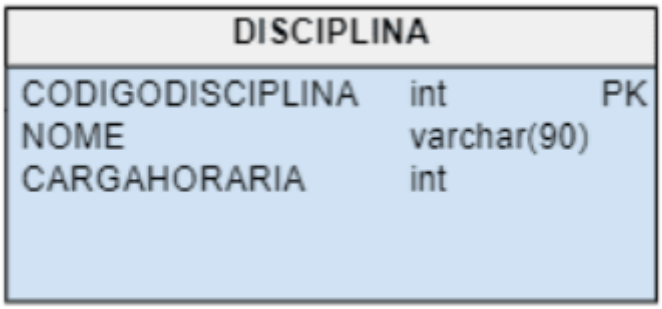
```
-- BEGIN implícito
INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO) VALUES (5,'Engenharia de
Computação',NULL);
-- COMMIT implícito
```

Estudamos que, quando uma transação é desfeita, qualquer operação que faz parte da transação deve ser cancelada. Vamos, então, ver como podemos fazer isso no PostgreSQL. Veja o exemplo a seguir, construído com objetivo de inserir um registro na tabela CURSO e, em seguida, indicar que a operação de inserção ser desfeita pelo SGBD:

```
SELECT * FROM CURSO; --Dados atuais;

BEGIN;

INSERT INTO CURSO (CODIGOCURSO,NOME,DATACRIACAO)
VALUES(6,'PSICOLOGIA', NULL);

SELECT * FROM CURSO; --Dados após a inserção;
```

```
ROLLBACK;

END;

SELECT * FROM CURSO;--Dados apos desfazer a transação;
```

Essa operação consiste um somente uma inserção no banco, vamos programar uma transação que consiste em mais de uma operação:



Vamos listar o conteúdo da tabela diciplina:

```
SELECT * FROM DISCIPLINA;
```

Agora, nossa intenção é alterar a carga horária das disciplinas de acordo com os critérios a seguir: gora, nossa intenção é alterar a carga horária das disciplinas de acordo com os critérios a seguir:

```
BEGIN;

UPDATE DISCIPLINA SET CARGAHORARIA = CARGAHORARIA*1.2 WHERE CARGAHORARIA
= 60;

UPDATE DISCIPLINA SET CARGAHORARIA = CARGAHORARIA*1.1 WHERE CARGAHORARIA
= 40;

SELECT*FROM DISCIPLINA;

COMMIT;
```

Outro ponto interessante no projeto de transações é a utilização de pontos de salvamento (SAVEPOINT). Observe o exemplo a seguir:

```
BEGIN;

UPDATE DISCIPLINA SET CARGAHORARIA = CARGAHORARIA*1.2 WHERE CARGAHORARIA
= 60;

SAVEPOINT CARGA60;

UPDATE DISCIPLINA SET CARGAHORARIA = CARGAHORARIA*1.1 WHERE CARGAHORARIA
= 40;

ROLLBACK TO CARGA60;
```

```
SELECT*FROM DISCIPLINA;

COMMIT;
```

O **SAVEPOINT** denominado **CARGA60**. Quando `ROLLBACK TO CARGA60;`

for executada, o SGBD vai desfazer a operação de UPDATE com carga horaria 40;

UM POUCO MAIS SOBRE ATUALIZAÇÃO TEMPORÁRIA

Estudamos que uma transação não deve atrapalhar o andamento de outra. Pense na execução da nossa última transação, que envolveu dois comandos de atualização na tabela DISCIPLINA.

o que aconteceria se tivéssemos em paralelo outra aplicação que submetesse consulta para acessar os registros da tabela DISCIPLINA no mesmo momento da execução da linha 3 da transação?

A consulta em questão enxergaria os dados “originais”, sem quaisquer alterações. Por qual razão? Para não haver o problema da atualização temporária. Queremos dizer que, se a transação fosse desfeita por qualquer motivo, o UPDATE da linha 2 seria também desfeito

UM POUCO MAIS SOBRE TRANSAÇÃO DE LEITURA

Vimos que uma transação que não modifica dados é denominada transação somente de leitura (READ ONLY). Caso contrário, é denominada leitura-gravação (READ WRITE).

Para especificar o tipo de transação, usaremos o comando SET TRANSACTION <TIPOTRANSAÇÃO>. No PostgreSQL, quando iniciamos uma transação, o padrão é READ WRITE.

```
BEGIN;

SET TRANSACTION READ ONLY;

UPDATE DISCIPLINA SET CARGAHORARIA = 60;

ROLLBACK;
```

O SGBD irá retornar uma mensagem de erro, informando que um comando de atualização não pode acontecer em uma transação de somente leitura(read only);