Implementação de Banco de Dados

Aula 9: LINGUAGEM SQL – Transação

Apresentação

Acessos concorrentes em banco de dados podem gerar vários problemas, e, para mitigar esta situação, os SGBDs implementam o conceito de transação e o seu controle.

Nesta aula, vamos ver o que é uma transação, suas propriedades, estados e como o seu controle é realizado.

Bons estudos!

Objetivos

- Conceituar transação;
- Descrever as propriedades e os estados da transação;
- Realizar o controle de transações.

Banco de Dados de Exemplo

Continuaremos utilizando o Banco de Dados da Empresa para os exemplos.

Modelo Lógico

As tabelas possuem os seguintes dados:

	id_regiao numeric (7)	nome character varying (40)
1	1	Norte
2	2	Sul

Região

	id numeric (7)	nome character varying (40)	id_regiao numeric (7)
1	10	Administrativo	1
2	20	Vendas	1
3	30	Compras	2

Departamento

	id numeric (7)	ult_nome character varying (20)	prim_nome character varying (20)	cargo character varying (30)	salario numeric (7,2)	dt_admissao date	cpf character (11)	id_depto numeric (7)	id_gerente numeric (7)
1	1	Velasques	Carmen	Presidente	29500.00	2009-05-05	34567890125	10	[null]
2	2	Neves	Lauro	Diretor de Compras	19500.00	2009-03-03	23456789012	30	1
3	3	Nogueira	Ernane	Diretor de Vendas	18000.00	2010-04-07	34567890123	20	1
4	4	Queiroz	Mark	Gerente de Compras	8000.00	2010-11-11	12345432123	30	2
5	5	Rodrigues	Alberto	Vendedor	4000.00	2008-10-10	87965432123	20	3
6	6	Ugarte	Marlene	Vendedor	3500.00	2009-03-03	87654345678	20	3

Empregado

	id numeric (7)	nome character varying (40)	vendedor numeric (7)
1	110	Ponto Quente	5
2	120	Casa Supimpa	6
3	130	Coisas e Tralhas	5
4	140	Casa Desconto	[null]

Cliente

Tendo este Banco em mente, é altamente recomendável que você execute os comandos de exemplo no PostGreSql.

Comentário

Foi escolhido como base o PostgreSQL, por ser um SGBD mais leve e fácil de instalar; porém você pode usar o SQLServer ou o Oracle. Quando houver diferença entre os SGBDs, você receberá um alerta

Transação

Uma transação é uma unidade de execução de programa que acessa e, possivelmente, atualiza vários itens de dados. Uma transação, geralmente, é o resultado da execução de um programa de usuário escrito em uma linguagem de manipulação de dados, e é delimitada da seguinte forma:

begin transaction

.....

end transacion.

A transação consiste de todas as operações ali executadas, entre o começo e o final da transação. Durante a execução de uma transação, o banco de dados pode passar por estados de inconsistência por vários motivos, como: queda de energia, falha física etc.

Propriedades das Transações

Após uma transação, o banco de dados deve continuar consistente. Para assegurar a integridade dos dados, um sistema de banco de dados deve ter as seguintes propriedades das transações: atomicidade, consistência, isolamento e durabilidade, que são as chamadas propriedades **ACID** das transações.

Clique nos botões para ver as informações.

Atomicidade

Indivisibilidade - É a propriedade que garante que todas as operações de uma transação são refletidas corretamente no banco de dados ou nenhuma será. Uma transação é indivisível;

Consistência

A execução de uma transação isolada (ou seja, sem a execução de outra transação qualquer concorrentemente) preserva a consistência do banco de dados;

Isolamento

Embora diversas transações possam ser executadas concorrentemente, o SGBD deve garantir que, para todo par de transações Ti, Tj, Ti, o Tj tenha terminado suas operações antes de Ti começar com as suas. Assim, cada transação não toma conhecimento de outras transações concorrentes no sistema. O sistema pode executar um pedaço de Ti parar e executar um pedaço de Tj, mas para as transações elas são únicas no sistema;

Durabilidade

Depois de a transação completar-se com sucesso, as mudanças que ela faz no banco de dados persistem, até mesmo se houver falha no sistema.

A observância dessas propriedades evita os diversos problemas de execução concorrente que vimos na aula passada.

Você deve estar se perguntando o que vêm a ser essas propriedades. Não se preocupe: vamos explicá-las mais à frente nesta aula.

Estados da transação

Quando não ocorre falha alguma na transação, todas as suas operações completam-se com sucesso e ela é efetivada, seus efeitos não podem ser desfeitos ou abortados. De outro modo, se ocorrer algum problema (falha) durante a transação, neste caso a transação será abortada e as operações já realizadas serão desfeitas.

Para melhor entendermos o funcionamento desse sistema, descrevemos o modelo simples e abstrato de estados das transações.

Cinco são os estados possíveis de uma transação:

Ativa (ou estado inicial)	Em efetivação parcial	Em falha	Abortada	Em efetivação
A transação permanece neste estado enquanto está executando suas operações.	Após a execução da última operação.	Após a descoberta de que a execução normal da transação já não pode ser realizada.	Depois que a transação foi desfeita e o banco de dados foi restabelecido ao estado anterior do início da execução da transação.	Após a conclusão da transação com sucesso

Estados da transação. | **Fonte:** adaptado de Sistemas de Banco de Dados (Siberschatz, Korth e Sudarshan, 6ª edição).

Transações concorrentes

Os sistemas de processamento de transação de banco de dados normalmente permitem que diversas transações sejam executadas de modo concorrente. Permitir que essas transações concorram na atualização dos dados traz diversas complicações em relação à consistência dos bancos de dados.

Vantagens	• Uma transação consiste de diversos passos. Alguns envolvem atividades de E/S; outros, atividade de UCP. Logo, atividades de E/S podem ser feitas em paralelo com o processamento de UCP; assim, o paralelismo entre atividades de E/S e de UCP pode ser explorado para executar diversas transações em paralelo, aumentando o throughput do sistema (um maior número de transações podem ser executadas num determinado tempo); • Reduz o tempo médio de resposta para uma transação se completar após ser submetida, pois as transações curtas não precisam esperar que as longas terminem para serem iniciadas.
Desvantagens	• Assegurar a consistência de transações exige trabalho adicional. É mais fácil assegurar as propriedades ACID em transações com execução sequencial do que em execuções concorrentes.

Exemplo de transações concorrentes

Considere um sistema bancário simplificado com várias contas e um conjunto de transações que atualiza estas contas e as operações:

Read(X): transfere o item de dados X do banco de dados para o buffer alocado à transação;

Write(X): transfere o item de dados X do buffer da transação para o banco de dados.

- Sejam T1 e T2 duas transações que transferem fundos de uma conta para outra.
- O saldo inicial de A é 100, e de B, 200.
- A transação T1 transfere 50 reais da conta A para a B.
- A transação T2 transfere 10% do saldo da conta A para a B.

T1:
read(A);
A := A – 50;
write(A);
read(B);
B := B + 50;
write(B);

T2:
read(A);
temp := A * 0,1;
A := A – temp;
write(A);
read(B);
B := B + temp;
write(B);

Se executarmos as transações em sequência, primeiro T1 e depois T2, ou primeiro T2 e depois T1, os dados estarão consistentes, pois, apesar de os saldos das contas serem diferentes no final da execução, o valor total das duas continua sendo de 300.

Atenção

O fato de a execução das transações gerar dados consistentes não significa que produza o mesmo resultado. A mudança da ordem das transações pode provocar resultados diferentes. A execução sequencial de duas transações sempre produz um resultado consistente.

Se executarmos as transações de forma concorrente, ou seja, intercalando operações das duas transações, poderemos obter resultados inconsistentes. Repare que, na execução 2, o saldo total ficou em 310, devido ao fato de as duas transferências terem sido efetivadas a partir do saldo de 100 em A.

Atenção

Uma execução concorrente pode produzir ou não um resultado consistente.

Exemplo

Exemplo adaptado de Sistemas de Banco de Dados (Siberschatz, Korth e Sudarshan, 6ª edição):
É tarefa do sistema de banco de dados garantir que qualquer escala executada deixe o banco consistente. O componente do sistema de banco de dados que executa esta tarefa é chamado de componente de controle de concorrência, e nele, as únicas operações significativas executadas por uma transação, do ponto de vista da escala de execução, são as instruções de leitura e de gravação.
A melhor forma de assegurar a consistência do banco de dados, sob execução concorrente, é garantir que qualquer escala executada tenha o mesmo efeito de outra que estivesse sendo executada sem qualquer concorrência.

Controle de transações

O SGBD assegura a consistência dos dados baseado nas transações. As transações dão mais flexibilidade e controle quando da mudança do conteúdo das tabelas e asseguram a consistência dos dados em caso de falhas nos processos do usuário ou no sistema.

Uma transação consiste de comandos DML (insert, update, delete, commit, rollback) que fazem uma mudança consistente nos dados. Por exemplo: uma transferência de valores entre duas contas bancárias implica no débito em uma conta e no crédito em outra no mesmo montante. Ambas as ações são realizadas ou são anuladas. O crédito não pode ser concretizado sem o correspondente débito.

Saiba mais

Na maioria dos SGBDs, uma transação começa com o comando **begin transaction**.

Uma exceção é o Oracle, no qual, quando damos qualquer comando, uma transação é aberta automaticamente pelo SGBD.

Uma transação termina quando aparece um dos seguintes comandos:

Commit

Encerra a transação corrente, fazendo que todas as modificações pendentes passem a ser definitivas.

Rollback

Encerra a transação corrente, desprezando todas as modificações pendentes.

Todas as modificações feitas durante a transação são temporárias, até que a transação seja commited, ou seja, concretizada.

Situação do dado antes do commit ou do rollback:

- As operações de manipulação de dados primeiramente afetam o buffer do banco de dados;
- O usuário corrente pode rever os resultados das operações de manipulação de dados usando o comando **select**;
- Outros usuários não podem ver os resultados das operações de manipulação de dados do usuário **corrente**.
- As linhas afetadas ficam bloqueadas; outros usuários não podem modificar os dados existentes nas linhas afetadas.

Situação do dado depois do rollback:

- As mudanças são desfeitas. O conteúdo anterior do dado é restabelecido;
- Os bloqueios nas linhas afetadas são desfeitos; as linhas ficam disponíveis para que outros usuários possam executar novas alterações.

  Além do commit e do rollback, alguns SGBDs têm o SAVEPOINT, que serve para criar um ponto intermediário para fazermos o rollback, de forma que não precisemos desfazer a transação toda.

  Vamos agora ver o controle de transação sendo realizado passo a passo nos três SGBDs que estamos estudando. Sugestão: faça o passo a passo junto com a aula no SGBD que você escolheu para estudar.

PostgreSQL

PASSO 1: Abrir a transação.

PASSO 2: Criar a tabela teste com duas colunas.

PASSO 3: Confirmar a criação da tabela.

PASSO 4: Inserir uma linha na tabela.

PASSO 5: Confirmar a inclusão.

PASSO 6: Criar um savepoint.

PASSO 7: Inserir uma segunda linha.

PASSO 8: Criar outro savepoint

PASSO 9: Inserir uma terceira linha.

PASSO 10: Ver o conteúdo da tabela.

PASSO 11: Fazer rollback para savepoint.

Foram criados dois pontos de salvamento:

SAVEPOINT A	**Após a inclusão da linha 1**
SAVEPOINT B	**Após a inclusão da linha 2**

Portanto, se comandar um rollback para o ponto de salvamento B, a inclusão da linha 3 será desfeita.

Comando:

Verificando o resultado:

Note que agora somente aparecem as linhas 1 e 2.

PASSO 12: fazer rollback para savepoint A.

Comando:

Verificando o resultado:

Como esperado, agora só existe a linha 1. Será que você consegue voltar ao savepoint B e desta forma ter novamente a linha 2?

Não consegue, pois, uma vez realizado o rollback até um savepoint, ele é desmarcado e não fica mais disponível para uso.

PASSO 13: terminar a transação.

Uma transação pode terminar com commit ou rollback. Se você der commit, ela será efetivada e depois não adianta dar rollback, pois já fechou, sendo que o contrário também é verdadeiro: se der rollback, não adianta tentar dar commit depois.

Comando:

Se você der select na tabela teste agora, dirá que ela não existe; o rollback inclusive cancelou a criação da tabela.

ORACLE

Vejamos agora uma sequência de passos similar no Oracle.

Passo 1: abrir a transação e criar a tabela teste.

Lembre-se que no Oracle não existe comando de abertura de transação; a abertura é implícita.

PASSO 2: confirmar a criação da tabela.

PASSO 3: inserir uma linha na tabela.

PASSO 4: Confirmar a inclusão.

PASSO 5: Criar um savepoint.

PASSO 6: Inserir uma segunda linha.

PASSO 7: Ver o conteúdo da tabela.

PASSO 8: Fazer rollback para savepoint.

Foi criado um savepoint denominado a após a inclusão da linha 1. Portanto, se for comandado rollback para o ponto de salvamento A, a inclusão da linha 2 será desfeita.

Comando:

Verificando o resultado:

Note que agora somente aparece a linha 1.

PASSO 8: terminar a transação.

Vamos fazer rollback da transação.

Comando:

Se você der select na tabela teste agora, dirá que ela não existe, correto?

Ela ainda existe; diferentemente do PostgreSQL e do SQLServer, os comandos de DDL (create table, drop table, alter table etc.) são uma transação autônoma, o que isso significa que possuem commit automático. Então você deve ter muito cuidado com isso.

Se, por exemplo, existir a seguinte sequência de comando:

Insert into X values (1);

Create table Y (C1 integer);

Insert into Y values (1);

Rollback

Somente a inserção na tabela Y será desfeita; já ao se criar a tabela Y, será realizado um commit automático incluindo todos os comandos antes da criação da tabela.

SQLServer

Vejamos agora no SQLServer.

Passo 1: Abrir a transação e criar a tabela teste.

Passo 2: Confirmar a criação da tabela.

Passo 3: Inserir uma linha na tabela.

Passo 4: Confirmar a inclusão.

Passo 5: criar um savepoint.

No SQLServer, o ponto de salvamento é criado com o comando **save transaction**.

Passo 6: inserir uma segunda linha.

Passo 7: ver o conteúdo da tabela.

Passo 8: fazer rollback para savepoint.

Foi criado um savepoint denominado A. Vamos fazer rollback.

Comando:

Note que, no SQLServer, o comando é rollback transaction <ponto de restauração>.

Verificando o resultado:

Note que agora somente aparece a linha 1.

Passo 8: Terminar a transação.

Vamos fazer rollback da transação.

Comando:

Se você der select, a tabela não irá mais existir, como aconteceu no PostgreSQL.

Atividade

1. Quais são as propriedades das transações?

2. Quando desejamos que uma transação seja totalmente executada, estamos no referindo a que propriedade das transações?

3. Qual o estado da transação que terminou com o comando de rollback?

4. Qual o comando que passa uma transação para o estado de efetivada?

Notas

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Referências

DATE, C. J. **Introdução a sistemas de banco de dados**. 7. ed. Rio de Janeiro: Campus, 2000.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7. ed. São Paulo: Pearson Addison Wesley, 2015.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistemas de banco de dados**. 5. ed. Rio de Janeiro: Campus, 2006.

Próxima aula

Próxima data

- Outros objetos de banco de dados.

Explore mais

Leia os seguintes artigos:

- IBM PureData System for Analitics, Version 7.1. Acesso em 5 mar. 2020
- Controle de concorrência entre transações em bancos de dados. Acesso em 5 mar. 2020
- Transações no Oracle: Commit, Rollback e Savepoint. Acesso em 5 mar. 2020
- PostgreSQL Prático/ Transações. Acesso em 5 mar. 2020
- Begin Transaction (Transact-SQL). Acesso em 5 mar. 2020