Controle de Transações

Banco de Dados II

Banco de Dados II

O que é uma Transação em Bancos de Dados

Em sistemas de bancos de dados damos o nome de **Transação** a um conjunto de uma ou mais operações que compõem uma única tarefa ou unidade lógica de trabalho a ser executada.

As operações de dados podem ser classificadas em uma entre quatro categorias: Criação, Leitura, Atualização ou Exclusão (em inglês, Create, Read, Update, Delete – daí a sigla **CRUD**).

Banco de Dados II

O que é uma Transação em Bancos de Dados

O sistema de banco de dados precisa garantir a execução correta das transações, independente de ocorrerem falhas; a transação é executada por completo (todas as operações) ou nenhuma de suas operações é executada (transação é abortada).

Em termos de SQL, geralmente as transações são definidas por declarações no formato:

```
BEGIN TRANSACTION
.
.
.
END TRANSACTION
```

Banco de Dados II

O que é uma Transação em Bancos de Dados

E a transação corresponde ao conjunto de operações contidas entre essas duas declarações (que variam entre sistemas diferentes), sendo executadas como se fossem uma única instrução, do ponto de vista do usuário.

Par garantir a integridade dos dados, o sistema de BD deve manter as propriedades que chamamos de **ACID**.

Banco de Dados II

Commit e Rollback

Caso a transação tenha sucesso, o BD é alterado de forma permanente, tendo os dados alterados gravados em disco; chamamos a essa operação de **Commit**.

Caso haja falha em qualquer operação, o banco de dados será retornado ao estado anterior ao início da transação; essa operação é chamada de **Rollback**.

Banco de Dados II

Commit e Rollback

Um exemplo clássico é uma transferência de fundos entre duas contas correntes. Suponha uma transação T que consiste na transferência de R$ 100,00 de uma conta corrente X para um conta corrente Y. Podemos defini-la da seguinte forma

```
Iniciar Transação T
    ler(X);
    X = X - 100.00;
    gravar(X);
    ler(Y);
    Y = Y + 100.00;
    gravar(Y);
Fim Transação
```

A operação ler() transfere os dados do banco de dados físico para uma área de memória (buffer de dados) onde o processo é realizado, e a operação gravar() transfere os dados do buffer alocado na memória para o banco de dados armazenado em disco.

Caso algum problema ocorra com qualquer uma das operações que compõem essa transação, toda a transação é cancelada (processo de rollback). Por exemplo, se a conta corrente X não tiver os fundos necessários (R$ 100,00), não é possível executar a operação de incremento de Y, e, portanto, a transação deve ser abortada.

Banco de Dados II

Commit e Rollback

A operação ler() transfere os dados do banco de dados físico para uma área de memória (buffer de dados) onde o processo é realizado, e a operação gravar() transfere os dados do buffer alocado na memória para o banco de dados armazenado em disco.

Caso algum problema ocorra com qualquer uma das operações que compõem essa transação, toda a transação é cancelada (processo de rollback). Por exemplo, se a conta corrente X não tiver os fundos necessários (R$ 100,00), não é possível executar a operação de incremento de Y, e, portanto, a transação deve ser abortada.

Banco de Dados II

Transações em MySQL

Uma **transação em bancos de dados** configura um conjunto de declarações SQL que são combinadas em uma única unidade de trabalho, sendo executadas como se fossem uma única operação. Essa abordagem é empregada para evitar diversos problemas que podem ocorrer em um banco de dados, além de aplicar as propriedades ACID.

Banco de Dados II

Propósitos das transações

- Fornecer unidades de trabalho confiáveis e que permitam a recuperação correta de falhas
- Manter o banco de dados consistente em caso de falha do sistema ou de operações
- Fornecer isolamento entre os programas que acessam o banco de forma concorrente.

Banco de Dados II

Propósitos das transações

Uma transação deve ser atômica, o que significa que ela deve ser executada de forma completa ou não ter nenhum efeito nas tabela envolvidas – ou seja, ou ela acontece por inteiro, ou não acontece nada.

Na prática, as transações aplicam as quatro **propriedades ACID** às operações no banco de dados:

Banco de Dados II

Propósitos das transações

- **Atomicidade**: Se as operações forem confirmadas, elas acontecerão todas sem exceção. Não ocorre a execução de apenas parte das operações contidas na transação.

- **Consistência**: As alterações no banco de dados somente acontecem caso o novo estado do sistema for válido. Se uma mudança inválida for executada, ela irá falhar, e o sistema permanece no seu estado anterior, válido.

Banco de Dados II

Propósitos das transações

- **Isolamento**: A transação não é visível a outros processos até que ela seja confirmada e persistida no banco.

- **Durabilidade**: Após a transação ter sido confirmada, ou seja, que suas operações tenham todas sido executadas com sucesso, as alterações são automaticamente persistidas (gravadas no banco), e não é necessário se preocupar em gravar as mudanças.

Banco de Dados II

Exemplo de transação corriqueira

Um exemplo clássico de transação é a transferência de dinheiro de uma conta bancária para outra, por TED, DOC ou PIX, por exemplo. Para executar a transferência, primeiro deve ser debitado o valor da conta de origem e, logo após, o dinheiro deve ser depositado na conta de destino.

Essa operação deve ser totalmente realizada (bem-sucedida). Se a operação for interrompida no meio do caminho, por exemplo após debitar o dinheiro da conta de origem mas antes de depositar na conta de destino, o valor transferido será perdido, o que evidentemente é um grande problema para os usuários e para a instituição financeira.

Banco de Dados II

Exemplo de transação corriqueira

No MySQL, as transações são suportadas pelo motor de banco de dados InnoDB, mas não o são pelo motor MyISAM. Desta forma, este tutorial tratará exclusivamente de bancos que utilizem o InnoDB.

Banco de Dados II

COMMIT e ROLLBACK de transações MySQL

Quando uma transação é efetuada com sucesso (todas as suas operações são bem-sucedidas), o banco de dados é alterado de forma permanente, com os dados envolvidos na transação persistidos (ou seja, salvos em disco); a essa operação damos o nome de **COMMIT**.

Porém, caso haja falha em qualquer uma das operações constituintes da transação, o banco de dados deverá ser retornado ao estado em que se encontrava anteriormente à execução da transação; essa operação é denominada **ROLLBACK**.

Banco de Dados II

COMMIT e ROLLBACK de transações MySQL

Desta forma, caracterizamos as transações da seguinte maneira:

- Todas as transações possuem início e fim
- As transações podem ser salvas (consolidadas no banco de dados – commit) ou desfeitas (rollback)
- Se transação falhar durante a execução de qualquer uma de suas operações constituintes, nenhuma destas operações será salva no banco de dados – nenhuma é comitada e todas são desfeitas.

Banco de Dados II

Autocommit

Por padrão, o MySQL utiliza o modo autocommit, que significa que declarações INSERT, UPDATE ou DELETE sofrem commit automaticamente quando executadas. Vamos usar transações para que o commit (ou rollback) sejam executados quando nós o quisermos, de modo a ter maior controle sobre a execução das declarações DML.

Podemos verificar se o autocommit está ativado no sistema, executando o comando a seguir:

```
SELECT @@autocommit;
```

Banco de Dados II

Autocommit

Se for retornado um valor igual a 0 (zero), isso significa que o autocommit está desativado; um valor igual a 1 indica status de autocommit ativo.

Para trabalhar com transações devemos desativar esse comportamento padrão, para que os commits venham a ser executados somente quando for indicado na declaração SQL.

Para forçar MySQL a não executar commit automaticamente, usamos a seguinte declaração:

```
SET autocommit = 0;
# ou
SET autocommit = OFF;
```

Banco de Dados II

Autocommit

Para habilitar novamente o autocommit usamos a seguinte declaração:

```
SET autocommit = 1;
# ou
SET autocommit = ON;
```

Após desativar o commit automático, podemos começar a trabalhar com transações no MySQL Workbench.

Banco de Dados II

Declaração START TRANSACTION

A declaração START TRANSACTION identifica o início de uma transação. Essa declaração desabilita temporariamente o modo autocommit, de modo que você deverá programar o momento de efetivar a transação. O modo autocommit é reativado logo após o término da transação.

No geral, criamos transações em procedimentos armazenados que alterem o estado do banco de dados. Vejamos um exemplo, criando uma transação.

Banco de Dados II

Exemplos de Transações

Antes de testarmos transações no MySQL, vamos desativar o commit automático:

```
SET autocommit = 0;
# ou
SET autocommit = OFF;
```

Conferimos o valor do autocommit atual:

```
SELECT @@autocommit;
```

Banco de Dados II

Exemplos de Transações

Agora, vamos **criar um tabela** de testes para realizarmos transações envolvendo commits e rollbacks:

```
# Criar tabela para testes
CREATE TABLE Dados_Livro
SELECT NomeLivro,ISBN13, PrecoLivro
FROM tbl_livros;

# Visualizar conteúdo da tabela
SELECT * FROM Dados_Livro
```

Banco de Dados II

Exemplos de Transações

Agora, vamos **criar um tabela** de testes para realizarmos transações envolvendo commits e rollbacks:

```
# Criar tabela para testes
CREATE TABLE Dados_Livro
SELECT NomeLivro,ISBN13, PrecoLivro
FROM tbl_livros;

# Visualizar conteúdo da tabela
SELECT * FROM Dados_Livro
```

Estamos prontos para testar nossos scripts.

Banco de Dados II

Exemplos de Transações

Exemplo 01: Transação com rollback

```
START TRANSACTION;
      DELETE FROM Dados_Livro;
      # apaga todos registros da tabela, "sem querer"
      INSERT INTO Dados_Livro(NomeLivro,ISBN13,
PrecoLivro) VALUES ('Ciência de Dados com
Python','9876532145632',69.88);
      SELECT * FROM Dados_Livro;
      # mostra tabela totalmente vazia
ROLLBACK;
# desfaz a transação

SELECT * FROM Dados_Livro;
# mostra os dados normalmente, pois nada foi comitado.
```

Banco de Dados II

Exemplos de Transações

Exemplo 02: Transação com commit

```
START TRANSACTION;
      DELETE FROM Dados_Livro;
      # apaga todos registros da tabela
      INSERT INTO Dados_Livro(NomeLivro, ISBN13,
PrecoLivro) VALUES('Ciência de Dados com
Python','9876532145632',69.88);
      SELECT * FROM Dados_Livro;
      # mostra tabela vazia
COMMIT;
# confirma a transação
SELECT * FROM Dados_Livro;
# agora os dados foram apagados e apenas o registro do
livro "Ciência de Dados com Python" é mostrado, pois a
transação foi confirmada.
```

Banco de Dados II

Exemplos de Transações

Exemplo 02: Transação em Stored Procedure

```
DELIMITER //
CREATE PROCEDURE insere_dados()
BEGIN
DECLARE erro_sql TINYINT DEFAULT FALSE;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro_sql = TRUE; START TRANSACTION;
        INSERT INTO Dados_Livro (NomeLivro, ISBN13, PrecoLivro) VALUES ('História da
Numismática', '9789865321465', 78.60);     INSERT INTO Dados_Livro (NomeLivro, ISBN13,
PrecoLivro) VALUES ('Biologia Marinha', '9784233876972', 177.50);
        INSERT INTO Dados_Livro (NomeLivro, ISBN13, PrecoLivro) VALUES ('Química
Experimental', '9789563210970', 165.32);   INSERT INTO Dados_Livro (NomeLivro, ISBN13,
PrecoLivro) VALUES ('Artes Plásticas', '9784523415974', 98,00);
        # Aqui há um erro que impedirá o COMMIT e provocará o
ROLLBACK.
        IF erro_sql = FALSE THEN COMMIT;
                SELECT 'Transação efetivada com sucesso.' AS Resultado;
        ELSE
                ROLLBACK;
                SELECT 'Erro na transação' AS Resultado;
        END IF; END//
DELIMITER ;
```

Banco de Dados II

Exemplos de Transações

Exemplo 02: Transação em Stored Procedure

```
DELIMITER //
CREATE PROCEDURE insere_dados()
BEGIN
DECLARE erro_sql TINYINT DEFAULT FALSE;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro_sql = TRUE; START TRANSACTION;
        INSERT INTO Dados_Livro (NomeLivro, ISBN13, PrecoLivro) VALUES ('História da
Numismática', '9789865321465', 78.60);     INSERT INTO Dados_Livro (NomeLivro, ISBN13,
PrecoLivro) VALUES ('Biologia Marinha', '9784233876972', 177.50);
        INSERT INTO Dados_Livro (NomeLivro, ISBN13, PrecoLivro) VALUES ('Química
Experimental', '9789563210970', 165.32);   INSERT INTO Dados_Livro (NomeLivro, ISBN13,
PrecoLivro) VALUES ('Artes Plásticas', '9784523415974', 98,00);
        # Aqui há um erro que impedirá o COMMIT e provocará o
ROLLBACK.
        IF erro_sql = FALSE THEN COMMIT;
                SELECT 'Transação efetivada com sucesso.' AS Resultado;
        ELSE
                ROLLBACK;
                SELECT 'Erro na transação' AS Resultado;
        END IF; END//
DELIMITER ;
```

Banco de Dados II

Exemplos de Transações

Para testar, basta chamar o **procedimento armazenado**:

```
CALL insere_dados();
```

Verificamos o conteúdo da tabela Dados_Livro após sua execução:

```
SELECT * FROM Dados_Livro;
```

Nenhum dado foi inserido, pois há um erro no último insert: foi usada vírgula para determinar o preço do livro “Artes Plásticas", mas o correto seria usar ponto. Conserte esse erro, altere ou recrie o procedimento armazenado, e o execute novamente, verificando se os dados são inseridos corretamente desta vez.

Banco de Dados II

Aplicações das Transações

Use transações sempre que:

- Existirem declarações DML INSERT, UPDATE ou DELETE que afetem dados que são relacionados
- Ao mover dados de uma tabela para outra
- E SEMPRE que a falha em uma das declarações DML puder violar a integridade dos dados e incorrer em sua perda.

Observação: Após a execução de declarações DDL, como CREATE ou DROP, o commit é executado automaticamente pelo MySQL, independente de se usar transações com START TRANSACTION.

Referências

- **Bóson Treinamentos**

  www.bosontreinamentos.com.br/mysql/como-trabalhar-com-transacoes-em-mysql/