MODELAGEM E DESENVOLVIMENTO DE BANCO DE DADOS

Pedro Henrique Chagas Freitas

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Linguagem de transação de dados (TCL)

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar a linguagem TCL.
- Exemplificar a linguagem TCL.
- Implementar a linguagem TCL.

Introdução

Neste capítulo, você vai estudar a utilização da linguagem TCL para permitir a realização de transações de dados em banco de dados via comunicação com os Sistemas Gerenciadores de Banco de Dados (SGBD). Logo, as instruções TCL são fundamentais para garantir e revogar comandos realizados no banco de dados.

Conceituando a utilização da linguagem TCL

A linguagem SQL (*Structured Query Language*) é uma linguagem para intercomunicação com bancos de dados que apresenta diversos recursos, o que significa que a linguagem SQL é muito rica em funcionalidades de intercomunicação entre o Sistema Gerenciador de Banco de Dados (SGBD) e o banco de dados. Por sua vez, para garantir a persistência dessa intercomunicação, a própria linguagem TCL (*Transaction Control Language*) — também conhecida como linguagem de controle de transações — que compõe a linguagem SQL garante instruções que vão confirmar a integridade do banco de dados após a realização de transações.

Por exemplo, quando uma tabela é criada e, posteriormente, dados são inseridos, atualizados ou deletados dessa tabela, estamos realizando manipulações via SGBD nos dados que estão no banco de dados; assim, precisamos garantir a integridade do banco validando as manipulações que estão sendo

realizadas. Para isso, utilizamos as instruções TCL (COMMIT, SAVEPOINT e ROLLBACK) para nivelar, junto ao banco, as transações que estão sendo realizadas. Assim, para iniciar as transações, usaremos sempre o BEGIN TRANSACTION.

Logo, a utilização da TCL é essencial para o bom e normal funcionamento de um banco de dados, principalmente se estivermos falando de um banco com várias instâncias, em que diferentes usuários estão constantemente "*commitando*" (COMMIT), salvando (SAVEPOINT) e revogando (ROLLBACK).

Temos, então, três instruções que vão conceituar e identificar a linguagem TCL:

- COMMIT;
- SAVEPOINT;
- ROLLBACK.

Esse é um exemplo puramente ilustrativo, mas demonstra bem o funcionamento das instruções TCL. Vamos supor que temos uma base de dados que é utilizada por uma fábrica de *software*. Logo, temos diversos desenvolvedores constantemente utilizando os dados do banco de dados – realizando consultas, alterações, inserções, atualizações, exclusões — e temos, também, os administradores de bancos de dados (DBAs — *database administrators*) dando acesso (GRANT) e retirando acesso (REVOKE) de vários desses desenvolvedores diariamente.

Pois bem, como vamos garantir a integridade do banco de dados com tantos usuários consultando, alterando, inserindo e excluindo dados, ao mesmo tempo, nesse banco? A resposta é simples: vamos mapear as transações realizadas a partir de uma linguagem que será responsável por gravar permanentemente no banco de dados (COMMIT), gravar temporariamente no banco de dados (SAVEPOINT) e revogar o que anteriormente havia sido gravado, retornando ao estado anterior (ROLLBACK).

Podemos perceber, então, que temos três variáveis dentro da TCL que, juntas, são responsáveis por manter a integridade do banco de dados a partir das instruções realizadas nesse banco. É comum, inclusive, que as instruções TCL acompanhem o final das linhas de código SQL, tendo em vista que, a partir dos comandos COMMIT, SAVEPOINT e ROLLBACK, é possível gravar e desfazer transações.

Além disso, podemos verificar a utilização das instruções TCL quando se comete um erro, por uma razão qualquer, e é necessário retornar uma coluna, um campo ou uma tabela a um estado anterior desejado (ROLLBACK) que

foi definido antes da alteração salva. Da mesma maneira, é importante salvar uma alteração realizada temporariamente (SAVEPOINT), assim como uma alteração permanente (COMMIT).

Assim sendo, quando o SGBD introduz uma instrução TCL para comunicação com o banco de dados, essa instrução (COMMIT, SAVEPOINT ou ROLLBACK) pode, ou não, acompanhar o parâmetro que se deseja salvar ou revogar. Logo, é normal que desenvolvedores utilizem a linguagem de controle de transação (TCL) para modificar suas linhas de código durante o desenvolvimento de uma funcionalidade de ou aplicações, por exemplo. Destaca-se que é fundamental garantir, também, o acesso (privilégio) ao que se deseja salvar (tabela, coluna, campo, etc.) ou revogar.

Dessa forma, é comum que sejam utilizados GRANTs (permissões de acesso) e REVOKEs (revogações de acesso) anteriormente à utilização das instruções TCL, tendo em vista que não é qualquer usuário em qualquer contexto que pode salvar permanentemente (COMMIT) ou temporariamente (SAVEPOINT) ou revogar uma instrução realizada (ROLLBACK). Assim sendo, delimita-se uma tabela, por exemplo, em que podem ser realizados os comandos COMMIT, SAVEPOINT e ROLLBACK.

Também é possível que haja o que se conhece com ambiente de desenvolvimento, em que são realizadas alterações diversas antes da subida para produção e interligação com o banco de dados. Logo, temos dois ambientes (desenvolvimento e produção), sendo que, no desenvolvimento, as instruções TCL são utilizadas para validar as modificações que vão ocorrendo até que elas sejam migradas para a produção. É importante observar que também podemos ter modificações diretas no ambiente de produção e, por sua vez, ter a utilização das instruções TCL nesse ambiente.

Fique atento

Os comandos TCL são utilizados para gerenciar as mudanças lógicas feitas pelas instruções DML, fazendo com que haja um agrupamento de instruções de forma lógica, seja para salvar no banco de dados ou para revogar o que outrora foi salvo.

Assim, as instruções TCL têm como responsabilidade:

- Salvar alterações realizadas (COMMIT);
- Salvar alterações temporárias (SAVEPOINT);
- Restaurar campos, colunas ou tabelas ao estado anterior a um COMMIT ou SAVEPOINT (ROLLBACK).

Saiba mais

Uma transação é uma unidade lógica de processamento que tem por objetivo preservar a integridade e a consistência dos dados. Esse processamento pode ser executado em sua totalidade, ou não, garantindo a atomicidade das informações.

Exemplificando a utilização da linguagem TCL

As instruções TCL permitem controlar e gerenciar as transações para manter a integridade dos dados dentro de instruções SQL. Vamos exemplificar a utilização das instruções TCL a partir de código SQL. A sintaxe básica de uma transação é:

BEGIN TRANSACTION (iniciamos nossa transação)

Corpo de comando: inserimos algum comando, seja uma inserção, atualização, exclusão, etc.; logo, temos, aqui, o conjunto de comandos a serem executados dentro de uma transação.

COMMIT ou SAVEPOINT ou ROLLBACK

Realizamos o controle da transação por meio de alguma instrução TCL para garantir a integridade daquilo que estamos implementando.

Nesse caso, então, temos que os comandos TCL vão finalizar as transações: a transação que realiza um COMMIT confirma o conjunto de comandos permanentemente, o SAVEPOINT fará o mesmo de forma temporal e o ROLLBACK desfaz todo o processo executado pelo corpo de comandos caso tenha ocorrido algum evento contrário ao desejado. Podemos concluir, então, que é comum a utilização do ROLLBACK quando temos erros, por exemplo, no corpo dos comandos de uma transação.

Exemplo

Por exemplo: temos um conjunto de instruções SQL que faz a identificação de um erro em uma transação chamada de '*ERROR*'. Essa função por padrão recebe o valor 0 caso não tenhamos erro; caso tenhamos algum erro, a função assume o valor 1. Temos, então, o valor 0 para não ocorreu erro e o valor 1 caso ocorra algum erro.

Faremos, então:

```
BEGIN TRANSACTION (Inicializamos nossa transação)
UPDATE FROM TbContas
SET NuSaldo = 1.000
WHERE NuSaldo < 50
IF '*ERROR*' = 0
COMMIT
ELSE
ROLLBACK
END
```

Nesse caso, estamos fazendo o seguinte: com o BEGIN TRANSACTION, iniciamos a transação e, com o UPDATE FROM TbContas, estamos dizendo que essa transação consistirá em uma atualização (UPDATE) na tabela TbContas. Essa atualização ocorrerá da seguinte forma:

```
SET NuSaldo = 1.000
WHERE NuSaldo < 50
IF '*ERROR*'= 0
COMMIT
```

Atribuímos o saldo (NuSaldo) = 1.000, onde o saldo (NuSaldo) < 50, se (IF) não ocorrer erro, ou seja, '*ERROR*' = 0, podemos "*commitar*" ou salvar permanentemente. Poderíamos usar, também, um SAVEPOINT; no caso, ficaria:

```
SET NuSaldo = 1.000
WHERE NuSaldo < 50
IF '*ERROR*'= 0
SAVEPOINT
```

Caso contrário, isto é, ELSE (se não), ou seja, IF '*ERROR*'= 1 — então ROLLBACK revoga a transação realizada. Lembre-se de que o valor 0 é utilizado para quando não ocorrer erro e o valor 1 é utilizado para quando ocorrer um erro. Nesse exemplo, então, iniciamos uma transação para executar uma mudança de saldo em contas onde, em caso de um erro, executamos o comando ROLLBACK para finalizar a transação e retornar os valores dos saldos. Caso tudo ocorra sem erro, a transação será executada e a alteração será gravada (Figura 1).

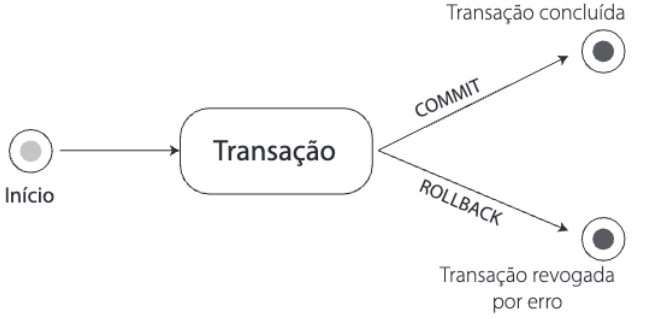

Figura 1. Transação.

Fique atento

A utilização da linguagem de controle de transações (TCL) pressupõe a utilização da linguagem DML. É possível utilizar a linguagem TCL com DDL, DCL e DML. A utilização da linguagem TCL não requer, necessariamente, uma manipulação dos dados (DML), mas, de modo geral, ocorre uma manipulação. Para dar seguimento a essa manipulação, ocorre uma transação por meio do TCL.

Implementação da linguagem TCL

Nessa implementação, utilizaremos a linguagem TCL com instruções SQL para implementar controles de transações no banco de dados com a finalidade de criar uma transação que irar mudar todos os salários dos jogadores de futebol

que jogam no Brasil, mas que agora estão indo jogar na Europa. Logo, o jogador que possui um salário menor que 10.000, terá um salário, agora, de 100.000.

Para isso, vamos criar um SAVEPOINT seguido de um COMMIT, para garantir que as informações sejam inseridas na tabela, salvando essa informações temporariamente, tendo em vista que, durante a migração para um determinado time europeu, o salário ficará em negociação. Logo, utilizamos um SAVEPOINT e, posteriormente, para garantir a integridade das informações salariais, vamos salvar permanentemente com o COMMIT. Teremos, então:

```
BEGIN TRANSACTION
UPDATE FROM TbBanco
SET ValorSalario = 100.000
WHERE ValorSalario < 10.000

SAVEPOINT
INSERT INTO TbBanco SELECT ValorSalario
IF '*Contrato_Assinado*'= 1
COMMIT
ELSE
ROLLBACK
```

Incluímos, então, um (IF) para designar que, após salvar temporariamente o salário = 100.000, caso tenhamos o contrato assinado com o jogador, isto é, valor 1, então COMMIT, salvamos permanentemente. Caso contrário, se não (ELSE), retornamos ao salário anterior de 10.000 por meio do ROLLBACK.

Leituras recomendadas

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Pearson, 2010.

HEUSER, C. A. *Projeto de banco de dados*. 6. ed. Porto Alegre: Bookman, 2010. (Série Livros Didáticos Informática UFRGS, 4).

KORTH, H. F.; SILBERSHATZ, A.; SUDARSHAN, S. *Sistemas de banco de dados*. 6. ed. Rio de Janeiro: Campus, 2012.

RAMAKRISHNAN, R. *Sistemas de gerenciamento de bancos de dados*. 3. ed. Porto Alegre: Penso, 2009.

SETZER, V. W. *Banco de dados*: conceitos, modelos, gerenciadores, projeto lógico, projeto físico. 3. ed. São Paulo: Blucher, 2002.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS