Banco de dados II

Aula 6

Outubro de 2024

Transações e Stored Procedure

- Transições
- Estados da transição
- Princípios da transição
- PL/SQL - Stored procedures e Functions

01

Transições (nos SGBDs)

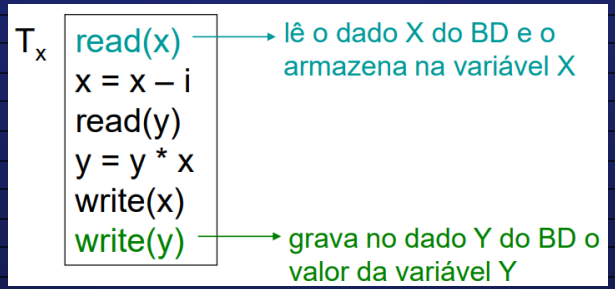
Banco de dados

Introdução (Transações)

- Sistema de processamento de operações de acesso ao BD
  - SGBDs são em geral multi-usuários
- Processam simultaneamente operações disparadas por vários usuários
  - deseja-se alta disponibilidade e tempo de resposta pequeno
    - execução intercalada de conjuntos de operações
  - exemplo: enquanto um processo i faz I/O, outro processo j é selecionado para execução
- Operações são chamadas transações

Introdução (Transações)

- Unidade lógica de processamento em um SGBD
- Composta de uma ou mais operações
  - seus limites podem ser determinados em SQL
- De forma abstrata e simplificada, uma transação pode ser encarada como um conjunto de operações de leitura e escrita de dados



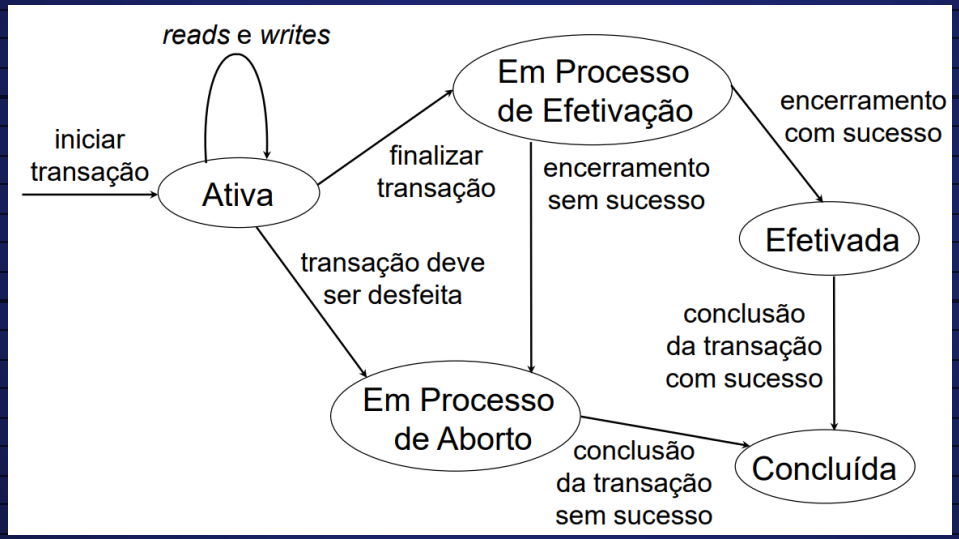
02

Estados da transição

Estados de uma TRANSIÇÃO

- Uma transação é sempre monitorada pelo SGBD quanto ao seu estado
  - que operações já fez? concluiu suas operações? deve abortar?
- Estados de uma transação
  - Ativa, Em processo de efetivação, Efetivada, Em processo de aborto, Concluída
  - Respeita um Grafo de Transição de Estados

Estados de uma TRANSIÇÃO



OBS.: Operação efetivada = COMMIT; Operação abortada = ROLLBACK; Em caso de rollback é possível realizar um recovery

Propriedades de uma Transição

- Requisitos que sempre devem ser atendidos por uma transação
- Chamadas de propriedades ACID
  - Atomicidade
  - Consistência
  - Isolamento
  - Durabilidade ou Persistência

03

Principios da Transição

Princípio Atomicidade

- Princípio do “Tudo ou Nada”
  - ou todas as operações da transação são efetivadas com sucesso no BD ou nenhuma delas se efetiva
    - preservar a integridade do BD
- Responsabilidade do subsistema de recuperação contra falhas (subsistema de recovery) do SGBD
  - desfazer as ações de transações parcialmente executadas
- Deve ser garantida, pois uma transação pode manter o BD em um estado inconsistente durante a sua execução

Princípio da Consistência

- Uma transação sempre conduz o BD de um estado consistente para outro estado também consistente
- Responsabilidade conjunta do
  - DBA
    - definir todas as RIs para garantir estados e transições de estado válidos para os dados
      - exemplos: salário > 0; salário novo > salário antigo
- subsistema de recovery
  - desfazer as ações da transação que violou a integridade

Princípio do Isolamento

- No contexto de um conjunto de transações concorrentes, a execução de uma transação Tx deve funcionar como se Tx executasse de forma isolada
  - Tx não deve sofrer interferências de outras transações executando concorrentemente
- Responsabilidade do subsistema de controle de concorrência (scheduler) do SGBD
  - garantir escalonamentos sem interferências

Princípio da Durabilidade ou Persistência

- Deve-se garantir que as modificações realizadas por uma transação que concluiu com sucesso persistam no BD
  - nenhuma falha posterior ocorrida no BD deve perder essas modificações
- Responsabilidade do subsistema de recovery
  - refazer transações que executaram com sucesso em caso de falha no BD

04

Stored Procedures

Um pouco de História

- Linguagem IBM Sequel desenvolvida como parte do projeto System R no IBM San Jose Research Laboratory
- Renomeada para Structured Query Language (SQL)
- Padrão SQL ANSI e ISO:
  - SQL-86
  - SQL-89
  - SQL-92

Um pouco de História

- SQL:1999 (nome da linguagem se tornou concordante com o Ano 2000!)
- SQL:2003

- Os sistemas comerciais oferecem a maioria, se não todos, os recursos da SQL-92, além de conjuntos de recursos variáveis dos últimos padrões e recursos proprietários especiais.
  - Nem todos os exemplos aqui podem funcionar em seu sistema específico.

Um pouco de História

O Modelo Relacional prevê, desde sua concepção, a existência de uma linguagem baseada em caracteres que suporte a definição do esquema físico (tabelas, restrições, etc.), e sua manipulação (inserção, consulta, atualização e remoção)

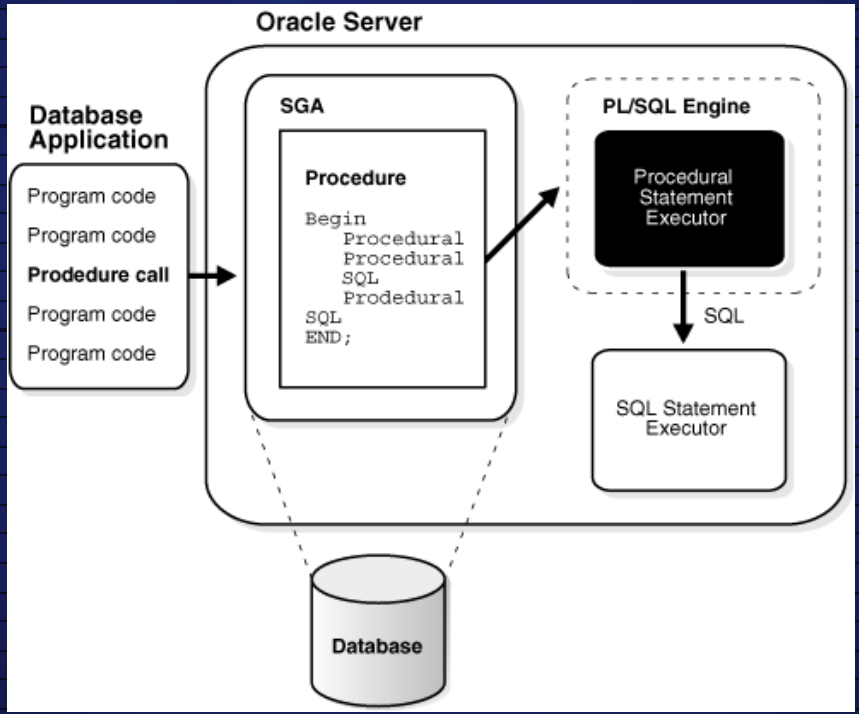
Um pouco de História

- A Linguagem SQL (Structured Query Language) é padrão para SGBDs Relacionais
  - padrão ANSI (American National Standards Institute)
    - ANSI X3.135-1986 = ISO/IEC 9075:1987
    - ANSI X3.135-1989 = ISO/IEC 9075:1989
    - ANSI X3.135-1992 = ISO/IEC 9075:1992 (SQL2)
    - ANSI X3.135.10-1998 substituído pelo SQL1999
    - ANSI X3.135-1999 = ISSO/IEC 9075:1999
- Embora seja capaz de prover acesso facilitado aos dados, a linguagem SQL possui certas limitações, como a impossibilidade de manipular uma tabela linha-a-linha, exigindo sua extensão, neste caso, através da definição de cursores

Um pouco de História

- O Modelo Relacional não previa, originalmente, a possibilidade de armazenar trechos de código no banco de dados. No entanto, foi adaptado para permitir a definição de
  - Stored Procedures: trechos de código escritos em linguagem SQL, armazenados no BD, e que podem ser ativados a partir de aplicações-cliente, comandos SQL, outras stored procedures, etc.
  - Triggers: trechos de código armazenados no BD ativados automaticamente após determinados eventos

Engenho do PL/SQL



Processamento de trabalhos em Batch

- A inicialização e terminação de programas envolvia manipulações de periféricos, pelo que se pensou deixá-las a cargo de especialistas, os operadores do sistema. O utilizador deixou de ter acesso à máquina, contrariamente ao regime de máquina dedicada, atrás referido.

PL/SQL

- A história do PL/SQL

Antes de 1991 a única forma deusar construções procedureis com o SQL era usar PRO*C. Foi onde as instruções SQL do Oracle foram embutidas em código C. O código C era pré-compilado para converter as instruções SQL em chamadas de bibliotecas.

Em 1991 o PL/SQL 1.0 foi lançado com o Oracle Versão 6.0. Ele era muito limitado nas suas capacidades.

PL/SQL

- O PL/SQL Versão 2.0 foi lançado com o Oracle Versão 7.0.
  - Esta era uma atualização maior. Ele suportava stored packages, procedures, funções, tabelas PL/SQL, registros definidos pelo programador e package extensions.

- O PL/SQL Versão 2.1 foi liberado com a Versão 7.1 do Oracle.
  - Isto permitiu o uso de stored functions dentro de instruções SQL e a criação de SQL dinâmico pelo uso do pacote DBMS_SQL. Foi também possível executar instruções de Linguagens de Definição de Dados de programas PL/SQL

PL/SQL

- A Versão 2.2 PL/SQJL foi lançada com a Versão 7.2 do Oracle.
  - Ele implementava um invólucro para programas PL/SQL para proteger o código de olhares curiosos. Também foi possível agendar trabalhos do banco de dados com o pacote DBMS_JOB.
- A Versão 2.3 do PL/SQL foi lançado com a Versão 7.3 do Oracle.
  - Esta versão aumentou as capacidades das tabelas PL/SQL e adicionou funcionalidades de E/S de arquivos.
- A Versão 2.4 do PL/SQL foi liberada com a Versão 8.0 do Oracle.
  - Esta versão suporta os melhoramentos do Oracle 8, incluindo Large Objects, projeto orientado a objetos, tabelas aninhadas e Oracle advanced queuing.

PL/SQL

A unidade básica em PL/SQL é um bloco. Todos os programas em PL/SQL são compostos por blocos, que podem estar localizados uns dentro dos outros. Geralmente, cada bloco efetua uma ação lógica no programa.

Um bloco tem basicamente a seguinte estrutura:

DECLARE - Secção para declaração de variáveis, tipos e subprogramas locais.

SELECTION - Secção para escolher linhas em uma tabela.

BEGIN - Secção Executável, nesta secção ficam as instruções procedimentais e SQL. Esta é a única secção do bloco que é indispensável e obrigatória.

EXCEPTION - Secção/Sector onde ficam as instruções de tratamento de erro.

END

PL/SQL

Os blocos PL/SQL são qualificados em:

Bloco anônimo

- Não tem nome
- Não está armazenado no SGDB
- Geralmente está armazenada na aplicação.

Stored SubProgramas

- Utiliza a estrutura do bloco anônimo com base.
- Estão armazenados no SGDB, a eles é atribuído um nome que poderá ser utilizado nas aplicações ou por outros objetos do banco de dados

05

PL / SQL

Procedural Language

Estrutura:

CREATE PROCEDURE [nome](parâmetros de entrada e saída)

BEGIN

DECLARE [variável] <tipo>;

[...]

END;

Procedural Language

Exemplo:

```
PROCEDURE `somar_numeros`(IN parametro1 INT, IN parametro2 INT )
BEGIN
    DECLARE total INT;
    SET total := parametro1 + parametro2;
    SELECT total;
END
```

Procedural Language

Exemplo(IF-THEM-ELSE):

```
CREATE PROCEDURE `verificar_par_impar` (IN numero INT, OUT resultado VARCHAR(5))
BEGIN
    IF MOD(numero, 2) = 0 THEN
        SET resultado := 'Par';
    ELSE
        SET resultado := 'Impar';
    END IF;
END
```

```
CALL verificar_par_impar(15, @resultado);

SELECT @resultado;
```

Procedural Language

Exemplo(WHILE):

```
CREATE PROCEDURE calcular_fatorial( IN numero INT, OUT resultado INT )
BEGIN
        DECLARE i INT;
        SET total := 1;
        SET i := 1;
        IF numero <= 0 THEN
                SET resultado := 0;
        ELSE
                WHILE i <= numero DO
                        SET total := total * i;
                        SET i := i + 1;
                END WHILE;
                SET resultado := total;
        END IF;
END;
```

```
CALL fatorial(5, @variavel);
```

```
SELECT @variavel;
```

Procedural Language

Exemplo texto:

```
PROCEDURE `remover_vogais`(
IN texto_entrada VARCHAR(255),
OUT texto_saida VARCHAR(255))
BEGIN
DECLARE i INT;
DECLARE letra VARCHAR(1);
DECLARE novo_texto VARCHAR(255);
SET i := 1;
```

OBS: SUBSTRING(string, posição inicial, tamanho)

```
SET letra := ‘’;
SET novo_texto := '';
WHILE i <= LENGTH(texto_entrada) DO
        SET letra := SUBSTRING(texto_entrada, i, 1);
        IF NOT (letra IN ('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U')) THEN
                SET novo_texto := CONCAT(novo_texto,
letra);
        END IF;
        SET i := i + 1;
END WHILE;
SET texto_saida := novo_texto;
END
```

Funções em SQL

String:

SUBSTRING(string, start, length): Extrai uma subcadeia da string especificada, definindo o ponto inicial e a quantidade de caracteres.

UPPER(string): Converte todos os caracteres da string para maiúsculas.

LOWER(string): Converte todos os caracteres da string para minúsculas.

TRIM(string): Remove espaços em branco desnecessários no início e no final da string.

LENGTH(string): Retorna o número de caracteres na string.

REPLACE(string, search, replace): Substitui todas as ocorrências de "search" por "replace" na string.

CONCAT(string1, string2): Combina duas strings em uma única string.

Funções em SQL

Numéricas:

ABS(number): Retorna o valor absoluto de um número (distância do zero).

SQRT(number): Calcula a raiz quadrada de um número.

POWER(number, power): Eleva um número a uma potência específica.

ROUND(number, precision): Arredonda um número para uma determinada precisão de casas decimais.

FLOOR(number): Arredonda um número para baixo para o inteiro mais próximo.

CEILING(number): Arredonda um número para cima para o inteiro mais próximo.

RAND(): Gera um número aleatório entre 0 e 1.

Funções em SQL

Data e hora:

NOW(): Retorna a data e hora atuais no formato completo.

CURDATE(): Retorna a data atual no formato YYYY-MM-DD.

CURTIME(): Retorna a hora atual no formato HH:MM:SS.

YEAR(date): Extrai o ano de uma data.

MONTH(date): Extrai o mês de uma data.

DAY(date): Extrai o dia do mês de uma data.

HOUR(time): Extrai a hora de um valor de tempo.

MINUTE(time): Extrai os minutos de um valor de tempo.

SECOND(time): Extrai os segundos de um valor de tempo.

Funções em SQL

Agregação:

COUNT(column): Conta o número de linhas em que a coluna não é NULL.

SUM(column): Retorna a soma de todos os valores na coluna.

AVG(column): Calcula a média de todos os valores na coluna.

MIN(column): Retorna o valor mínimo na coluna.

MAX(column): Retorna o valor máximo na coluna.

Funções em SQL

Funções de data:

DATEDIFF(interval, start_date, end_date): Calcula a diferença entre duas datas em unidades de intervalo especificado (ano, mês, dia, etc.).

STR_TO_DATE(string, format): Converte uma string no formato especificado para uma data.

DATE_FORMAT(date, format): Formata uma data no formato especificado.

Criando Funções

Também é possível criar funções a serem utilizadas nos stored procedures, segue a estrutura:

CREATE FUNCTION [nome da funcao](parâpetros, não é necessário IN ou OUT, todos são obrigatoriamente entrada)
RETURNS <tipo primitivo de retorno>
BEGIN
DECLARE <todas as declarações possíveis>;
<lógica da função>;
RETURN [variável];
END;

Criando Funções

Segue um exemplo:

```
CREATE FUNCTION calcular_area_triangulo(
base DECIMAL(10,2),  altura DECIMAL(10,2))
RETURNS DECIMAL(10,2)
BEGIN
DECLARE area DECIMAL(10,2);
SET area = base * altura;
RETURN area;
END;
```

Chamador:

```
SELECT calcular_area_retangulo(5.00, 3.00)
AS area_retangulo;
```

Diferenças entre FUNCTIONS e SP

Recurso	Funções	Procedimentos Armazenados
Propósito Principal	Retornar um único valor	Encapsular uma série de instruções SQL
Estrutura	Mais simples, com um tipo de retorno e parâmetros opcionais	Mais complexa, com blocos de código e parâmetros opcionais
Invocação	Chamada de dentro de instruções SQL, funções ou procedimentos armazenados	Chamada usando uma sintaxe semelhante às funções, mas também pode ser disparada por eventos ou execuções programadas
Manipulação de Dados	Pode modificar dados, mas normalmente não mantém estado nem gerencia transações	Pode realizar uma ampla variedade de tarefas de manipulação de dados, incluindo inserção, atualização, exclusão e recuperação de dados. Eles também podem gerenciar transações

Obrigado!

Alguma dúvida?

jheymesso.Cavalcanti@unicap.br