

C07 - BD

Arthur Openheimer

Procedures

São blocos de comando SQL armazenados de modo persistente e executados pelo SGBD no próprio servidor do banco de dados

Procedures

Quando utilizar?

1 - Quando o banco se conecta a várias aplicações escritas em diferentes linguagens, ou rodam em diferentes plataformas, porém devem executar o mesmo procedimento num banco de dados

Procedures

Quando utilizar?

- 1 - Quando o banco se conecta a várias aplicações escritas em diferentes linguagens, ou rodam em diferentes plataformas, porém devem executar o mesmo procedimento num banco de dados
- 2 - Reduzir o tráfego de informações, aumentar o desempenho e reduzir custos de comunicação entre clientes e servidor

Procedures

Quando utilizar?

- 1 - Quando o banco se conecta a várias aplicações escritas em diferentes linguagens, ou rodam em diferentes plataformas, porém devem executar o mesmo procedimento num banco de dados
- 2 - Reduzir o tráfego de informações, aumentar o desempenho e reduzir custos de comunicação entre clientes e servidor
- 3 - Priorizar consistência e segurança, mesmo que uma funcionalidade seja utilizada em diferentes lugares, sua centralização evita a chance de erros

Procedures

Quando utilizar?

- 1 - Quando o banco se conecta a várias aplicações escritas em diferentes linguagens, ou rodam em diferentes plataformas, porém devem executar o mesmo procedimento num banco de dados
- 2 - Reduzir o tráfego de informações, aumentar o desempenho e reduzir custos de comunicação entre clientes e servidor
- 3 - Priorizar consistência e segurança, mesmo que uma funcionalidade seja utilizada em diferentes lugares, sua centralização evita a chance de erros
- 4 - Para garantir que toda operação seja executada, ou que nada seja salvo no BD caso algum erro ocorra

Procedures

Vantagens x Desvantagens

Vantagens - Reduzir tráfego na rede, melhorar a performance de um BD, criar tarefas agendadas ou rotinas de processamento, diminuir riscos

Desvantagens - Exige muito mais processamento do lado do servidor

Procedures

Criando a
procedure

```
4 DELIMITER $$
5 • DROP PROCEDURE IF EXISTS soma $$
6 • CREATE PROCEDURE soma(IN n1 INT, IN n2 INT)
7   BEGIN
8     SELECT (n1+n2) AS Soma;
9   END $$
10 DELIMITER ;
11
12 • CALL soma(4,5);
13 • CALL soma(6,6);
14
15 • DROP PROCEDURE soma;
```

Parâmetros

Chamando
a procedure

Apagando
procedure

Result Grid		Filter Rows:		Export:		Wrap Cell
	Soma					
▶	12					

Procedures

IF-ELSE

Criando e usando variáveis

```
5 DELIMITER $$
6 DROP PROCEDURE IF EXISTS testeif $$
7 CREATE PROCEDURE testeif(IN n1 INT, IN n2 INT)
8 BEGIN
9     DECLARE soma INT;
10    DECLARE resposta VARCHAR(15);
11
12    SET soma = n1+n2;
13
14    IF soma > 0
15    THEN SET resposta = 'Positivo';
16    ELSE
17        SET resposta = 'Negativo';
18    END IF;
19
20    SELECT concat('Resultado da soma resulta em um numero: ', resposta) AS Resposta;
21 END $$
22 DELIMITER ;
23
24 CALL testeif(4,-10);
```

Utilizando o bloco IF

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Resposta
Resultado da soma resulta em um numero: Negativo

Procedures

WHILE

```
5 DELIMITER $$
6 ● DROP PROCEDURE IF EXISTS testewhile $$
7 ● CREATE PROCEDURE testewhile(IN num INT)
8   BEGIN
9     DECLARE res INT;
10    SET res = 1;
11
12    WHILE num > 0 DO
13      SET res = res*num;
14      SET num = num - 1;
15    END WHILE;
16
17    SELECT res as FATORIAL;
18  END $$
19  DELIMITER ;
20
21 ● CALL testewhile(3);
```

Utilizando o
bloco WHILE

Result Grid	Filter Rows:	Export:
FATORIAL		
6		

Functions

Também são blocos de comando SQL armazenados de modo persistente e executados pelo SGBD no próprio servidor do banco de dados

Functions

Quando utilizar?

Realizam pequenas operações, normalmente auxiliares, que podem ser solicitadas durante um processo de transação, diferente das Procedures

São responsáveis por tratamento de variáveis, que podem ser compartilhadas para as Procedures ou aplicações

Functions também sempre retornam valores e parâmetros, Procedures não possuem retorno, mas podem gerar conjuntos de resultados e retornar via parâmetros

Functions

Criando a
function

```
5 DELIMITER $$
6 DROP FUNCTION IF EXISTS mult $$
7 CREATE FUNCTION mult(a FLOAT, b FLOAT) RETURNS FLOAT
8 DETERMINISTIC
9 BEGIN
10     RETURN a*b;
11 END $$
12 DELIMITER ;
```

Parâmetros e
tipo do retorno

Chamando
a function

```
3
4 SELECT mult(2.5, 7.5) AS Resultado;
5
6 DROP FUNCTION mult;
```

Apagando a
function

Result Grid	Filter Rows:	Export:	Wrap
Resultado			
18.75			

Exercício

```
DROP DATABASE IF EXISTS exercicio_procedure_function;  
CREATE DATABASE exercicio_procedure_function;  
USE exercicio_procedure_function;
```

```
CREATE TABLE Aluno(  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(50),  
    idade INT,  
    curso VARCHAR(3),  
    email VARCHAR(100)  
);
```

- Crie uma Function **criaEmailAluno** para criar um email para cada aluno (nome@curso.inatel.br)
- Depois crie uma Procedure **insereAluno** para fazer a inserção dos seguintes alunos: (as inserções devem ser feitas dentro da procedure)

Dica: Na Function utilize a função CONCAT() para gerar o email, basta colocar a string e os atributos separados por vírgula

id	nome	idade	curso	email
1	Joao	20	ges	Joao@ges.inatel.br
2	Maria	21	gec	Maria@gec.inatel.br
3	Jose	18	geb	Jose@geb.inatel.br
4	Ana	19	get	Ana@get.inatel.br