

**STORED PROCEDURE**

Stored procedures são conjuntos de instruções SQL armazenadas no banco de dados para serem executadas posteriormente. Elas são úteis para agrupar várias operações em uma unidade lógica e reutilizável. As stored procedures podem receber parâmetros de entrada e retornar valores ou conjuntos de resultados, tornando-as flexíveis e capazes de lidar com uma variedade de tarefas.

Aqui está um exemplo de criação de uma stored procedure que seleciona os funcionários de um determinado departamento:

DELIMITER $$

CREATE PROCEDURE funcionarios\_de\_depto\_3()

BEGIN

SELECT \* FROM cadfun WHERE DEPTO = 3;

END $$

DELIMITER ;

Agora você pode chamar a stored procedure da seguinte maneira:

**CALL** funcionarios\_de\_depto\_3();

Esse código irá executar a stored procedure e retornar os resultados da consulta.



**As finalidades das stored procedures são as seguintes:**

**Reutilização de código**: As stored procedures permitem que você escreva o código SQL uma vez e o reutilize várias vezes em diferentes partes do aplicativo. Isso reduz a duplicação de código e facilita a manutenção.

**Modularidade**: Ao usar stored procedures, você pode dividir a lógica de negócios em partes menores e mais gerenciáveis. Isso torna o código mais modular, facilitando a leitura, o entendimento e as alterações futuras.

**Melhoria no desempenho**: Stored procedures são compiladas e armazenadas em cache pelo banco de dados, o que pode resultar em melhor desempenho. A execução repetida da mesma stored procedure não requer análise e otimização do plano de execução toda vez, o que pode reduzir o tempo de processamento.

**Controle de acesso e segurança**: As stored procedures permitem que você defina permissões de acesso granulares, controlando quem pode executá-las e quais tabelas ou dados podem ser acessados. Isso ajuda a fortalecer a segurança do banco de dados.

A diferença entre stored procedures e consultas SQL regulares (ou ad hoc) está na forma como são usadas e executadas:

**Consultas SQL regulares**: São consultas SQL individuais que são executadas diretamente a partir de um aplicativo ou interface de banco de dados. Elas são escritas e executadas em tempo real, sem a necessidade de armazenamento prévio no banco de dados. São usadas para recuperar, inserir, atualizar ou excluir dados de forma imediata.

**Stored procedures**: São blocos de código SQL armazenados no banco de dados e que podem ser chamados posteriormente. São criadas e armazenadas previamente no banco de dados e podem ser executadas repetidamente sem a necessidade de reescrever o código SQL. Permitem encapsular lógica de negócios complexa, manipular dados e executar várias operações em um único chamado. São usadas para automatizar tarefas comuns, melhorar a reutilização de código e fornecer uma camada de abstração para o aplicativo.

**Exemplo de proceder com entrada de dados.**

**Criação de uma stored procedure:**

DELIMITER $$

**CREATE** **PROCEDURE** salario(**in** idEnpregado **int**)

**BEGIN**

**select** SALARIO **FROM** CADFUN **WHERE** CODFUN = IdEnpregado;

**END** $$

DELIMITER ;

A função dessa stored procedure é retornar o salário de um determinado empregado com base no código de funcionário fornecido como parâmetro.

Agora você pode chamar a stored procedure da seguinte maneira:

Atenção como esse stored procedure tem entrada de dados é necessário que neste caso específico inserido um valor inteiro para o seu funcionamento correto.

**call** salario(2);

Esse código irá executar a stored procedure e retornar os resultados da consulta.



Essa stored procedure pode ser útil em situações em que você precise obter rapidamente o salário de um empregado específico sem ter que escrever a consulta SQL completa toda vez que precisar dessa informação.

Dessa forma, a stored procedure "salario" oferece uma maneira conveniente e eficiente de obter o salário de um empregado.

**Exemplo de proceder com entrada e saida de dados.**

Em uma determinada empresa, o salário máximo para um funcionário é de 10.000 BRL. Para verificar cotidianamente quanto falta para um funcionário alcançar esse valor, foi criado um stored procedure que exibe o valor atual do salário do funcionário e a diferença para atingir o salário máximo.

**Criação de uma stored procedure:**

DELIMITER $$

**CREATE** **PROCEDURE** SALARIO\_MAXIMO(**in** idEnpregado **int**, **out** salario\_Limite **decimal**)

**BEGIN**

**DECLARE** FF **DECIMAL**;

**SET** FF = (**select** SALARIO **as** SALARIO\_ATUAL **FROM** CADFUN **WHERE** CODFUN = IdEnpregado);

**SET** salario\_Limite = 10000 - FF;

**select** SALARIO **as** SALARIO\_ATUAL **FROM** CADFUN **WHERE** CODFUN = IdEnpregado;

**END** $$

DELIMITER ;

Agora você pode chamar a stored procedure da seguinte maneira:

Atenção como esse stored procedure tem entrada e saida de dados é necessário que neste caso específico inserido um valor inteiro e crie uma variável que armazene o valor de saída para o seu funcionamento correto.

**CALL** SALARIO\_MAXIMO(1, @META);

**SELECT** @META;

Esse código irá executar a stored procedure e retornar os resultados da consulta.



Essa stored procedure pode ser útil para monitorar o progresso dos funcionários em relação ao salário máximo. Ao utilizar essa função regularmente, é possível verificar quanto falta para cada funcionário alcançar o salário máximo estabelecido. Isso pode ser útil para fins de gerenciamento de recursos humanos, planejamento financeiro e motivação dos funcionários.

**Foi usado o seguinte banco de dados:**

Comentário: Este código cria um novo banco de dados chamado "banco\_teste":

create database banco\_teste;

Comentário: Este código cria uma tabela chamada "CadFun" com as colunas CODFUN, NOME, DEPTO, FUNCAO e SALARIO. A coluna CODFUN é definida como chave primária (primary key) e as outras colunas têm seus tipos de dados especificados:

create table CadFun (

CODFUN integer not null primary key,

NOME varchar(40) not null,

DEPTO char( 2),

FUNCAO char(20),

SALARIO decimal(10, 2)

);

Comentário: Este código insere um novo registro na tabela CadFun com os valores fornecidos:

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (12, 'CARLOS ALBERTO', '3', 'VENDEDOR', 1530.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, SALARIO) VALUES (44, 'JACIBA DA SILVA', '3', 1500.00);

INSERT INTO cadfun VALUES (2, 'WILSON DE MACEDO', '3', 'PROGRAMADOR', 1050.00);

INSERT INTO cadfun VALUES (2, 'WILSON DE MACEDO', '3', 'PROGRAMADOR', 1050.00);

INSERT INTO cadfun VALUES (5, 'AUGUSTO SOUZA', '3', 'PROGRAMADOR', 1050.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (4, 'CARLOS BASTOS', '5', 'VENDEDOR', 1530.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (25, 'PEDRO SILVA', '3', 'SUPERVISOR', 1599.51);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (3, 'ANA BASTOS', '5', 'VENDEDORA', 1530.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (10, 'PAULO DA SILVA', '2', 'VENDEDOR', 1530.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (9, 'SOLANGE PACHECO', '5', 'SUPERVISORA', 1599.51);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (6, 'MARCELO SOUZA', '3', 'ANALISTA', 2250.00);

INSERT INTO cadfun (CODFUN, NOME, DEPTO, FUNCAO, SALARIO)

VALUES (1, 'CELIA NASCIMENTO', '2', 'SECRETARIA', 1200.50);

**Referência Bibliográfica**

Referência:

COMO USAR parâmetros de entrada, saída e o comando IF em Stored Procedures - Curso de SQL - Aula 38. [*S. l.*], 23 fev. 2021. Disponível em: <https://www.youtube.com/watch?v=Hz_ozlrA9Z8>. Acesso em: 27 maio 2023.

Referência:

VAMOS aprender sobre Stored Procedures #P1 - Curso de SQL - Aula 37. [*S. l.*], 19 fev. 2021. Disponível em: <https://www.youtube.com/watch?v=lHE5i7sbhhU>. Acesso em: 27 maio 2023.