



MINISTÉRIO DAS TELECOMUNICAÇÕES, TECNOLOGIAS
DE INFORMAÇÃO E COMUNICAÇÃO SOCIAL
MINISTÉRIO DA EDUCAÇÃO

PROGRAMABILIDADE: VIEWS E PROCEDIMENTOS ARMAZENADOS



INSTITUTO DE TELECOMUNICAÇÕES



ÍNDICE

1. Views;
2. Criar VIEWS;
3. Consultas sobre VIEWS;
4. Editar e apagar VIEWS;
5. Exercícios;



ÍNDICE

- 6. Procedimentos armazenados;
- 7. Características;
- 8. Vantagens;
- 9. Criar procedimentos armazenados;
- 10. Criar procedimentos armazenados – MYSQL;



ÍNDICE

- 11. Estrutura Condicionais – IF;
- 12. Estrutura condicionais - CASE;
- 13. Estrutura de repetição - WHILE;
- 14. Declaração de variáveis;
- 15. Exercícios;

Uma view funciona como uma janela. Dá diferentes perspectivas da base de dados.

- ❑ View: é uma forma diferente de se ter acesso a uma tabela ou conjunto de tabelas.

(Damas,2005)

- ❑ A View não existe fisicamente. Funciona como uma tabela virtual. Os dados continuam a existir nas tabelas, logo qualquer alteração nos dados é reflectida na view que a eles estão associados.

As views não são mais do que um comando SELECT armazenado numa base de dados com um nome associado.

- ❑ A view na verdade é uma tabela definida sobre a forma de uma consulta SQL pré-definida

Fonte: <https://www.tutorialspoint.com/sql/sql-using-views.htm>

❑ Pode-se implementar views para resolver as seguintes situações:

- Segurança – evitar que determinados utilizadores consultem determinados dados;

- Confidencialidade – evitar que utilizadores consultem dados de acesso reservado;

- Simplicidade: a forma como se acede a uma view é exactamente como se acede a uma tabela;

❑ Pode-se condensar em uma única view dados de várias tabelas;

❑ Um comando simples feito na view pode equivaler a um comando SELECT bastante complexo.



Criar VIEWS

- ❑ Para criar uma view utiliza-se o comando CREATE VIEW

Sintaxe:

CREATE VIEW Nome_da_view AS Comando_SELECT

Uma vez, criada pode-se fazer consultas sobre a view como se de uma tabela se tratasse.

Exemplo: Considere a tabela de encomendas, criar uma view de encomendas processadas.

ID_Encomenda	ID_Cliente	Estado	Valor
100	69	Pendente	15000
101	18	Em processamento	30000
102	142	Cancelada	12500
103	18	Processada	5000
104	18	Processada	18500
105	142	Processada	20000
106	142	Em processamento	55000

CREATE VIEW

Encomenda_Processadas AS

SELECT * FROM Encomendas WHERE Estado LIKE 'Processada';



Consultas sobre VIEWS

☐ Consultas sobre VIEWS

As consultas sobre as views são feitas da mesma forma que a consultas as tabelas.

```
SELECT * FROM Encomenda_Processadas
```

Qual é o resultado deste comando?



Editar e apagar VIEWS

❑ Editar e apagar VIEWS

As vies são editadas através do comando ALTER VIEW

Sintaxe:

```
ALTER VIEW Encomenda_Processadas AS
```

```
SELECT * FROM Encomendas WHERE Estado LIKE
```

```
'Processada' AND Valor > 1000
```

As views são eliminadas através do comando DROP VIEW

Sintaxe:

```
DROP VIEW Encomenda_Processadas
```



Exercícios

1. Use a base de dados GCBD:
2. Crie uma view de produtos e suas respectivas categorias. Na view deve constar o ID, nome do produto, nome da categoria e preço.
3. Crie uma view de nome “EncomendasCarlos” que mostre todas as encomendas (e seus respectivos detalhes) processadas pelo funcionário Carlos. Use subconsultas sempre que necessário.
4. Crie uma view de nome “Diário_de_Caixa” que mostre o total de facturamento por dia.



Procedimentos

- ☐ Procedimentos armazenados ou stored procedures são um conjunto de comandos SQL identificados por um nome que podem ser armazenados em um servidor de base de dados, e são usados para eventuais processamentos. (Márcio Bueno)
- ☐ Permite encapsular um conjunto de tarefas repetitivas, evitando que se reedite as instruções SQL. Podendo apenas ser chamado um procedimento.

Exemplos de utilização:

- Inserção conjunta de dados em mais de uma tabela;
- Actualização massiva de dados (Ex: actualização de preços de produtos);
- Implementação de rotinas de limpeza de dados que não são mais necessários

❑ Características

- 1-Têm um nome;
- 2- Têm que ser criados e depois “chamados” - executados;
- 3- Podem ser chamados pelo SGBD ou por uma aplicação;
- 4- Suportam parâmetros (variáveis definidas pelo utilizador) de entrada e saída;
- 5- Suportam estruturas condicionais e de repetição.



Vantagens

Úteis para efetuar tarefas de processamento periódico e complexo;

- Fácil gestão – permitem encapsular regras de negócio, caso as regras mudarem, são trocadas num único lugar;
- Segurança – os utilizadores não precisam de saber dos detalhes da sua implementação;

Performance :

- Permite mover grande parte da manipulação de dados para o servidor de base de dados, em vez de ser na aplicação;
- Quando executado pela primeira vez os comandos são analisados e otimizados, o que evita que estas operações se repitam a cada execução;
- Após a primeira execução é armazenado em memória, o que garante melhores velocidades de processamento;
- Permite executar operações complexas enviando apenas um comando, o que reduz a quantidade de requisições entre aplicações e o servidor melhorando o desempenho das aplicações.



Criar procedimentos armazenados

Sintaxe:

```
CREATE PROCEDURE nome_do_procedimento [{@parâmetros  
tipo_de_dados}]
```

```
AS
```

```
[BEGIN]
```

```
<Bloco de instruções_SQL>
```

```
[END]
```



Criar procedimentos armazenados

Os parâmetros são opcionais, i.e., um procedimento armazenado pode ou não conter parâmetros.

Para que o procedimento receba parâmetros deve respeitar uma sintaxe própria.

(Modo Nome_parâmetro domínio)

Quanto ao modo os parâmetros podem ser :

- IN – Indica que é um parâmetro de entrada de dados;
- OUT – Usado para a saída de dados. Deve ser passada um variável por referência.

Quando são múltiplos parâmetros, estes devem estar separados por vírgulas.

Nota: Se não se associar nenhum tipo ao parâmetro, por defeito será de entrada.



Criar procedimentos armazenados

Após serem criados os procedimentos armazenados são guardados na base de dados.

No SGBD MySQL ficam armazenados no objecto Stored Procedure.

Sempre que for necessária a sua utilização devem ser apenas executados/chamados.

Sintaxe de execução geral EXEC Nome_procedimento [parâmetros];

Sintaxe de execução MySQL

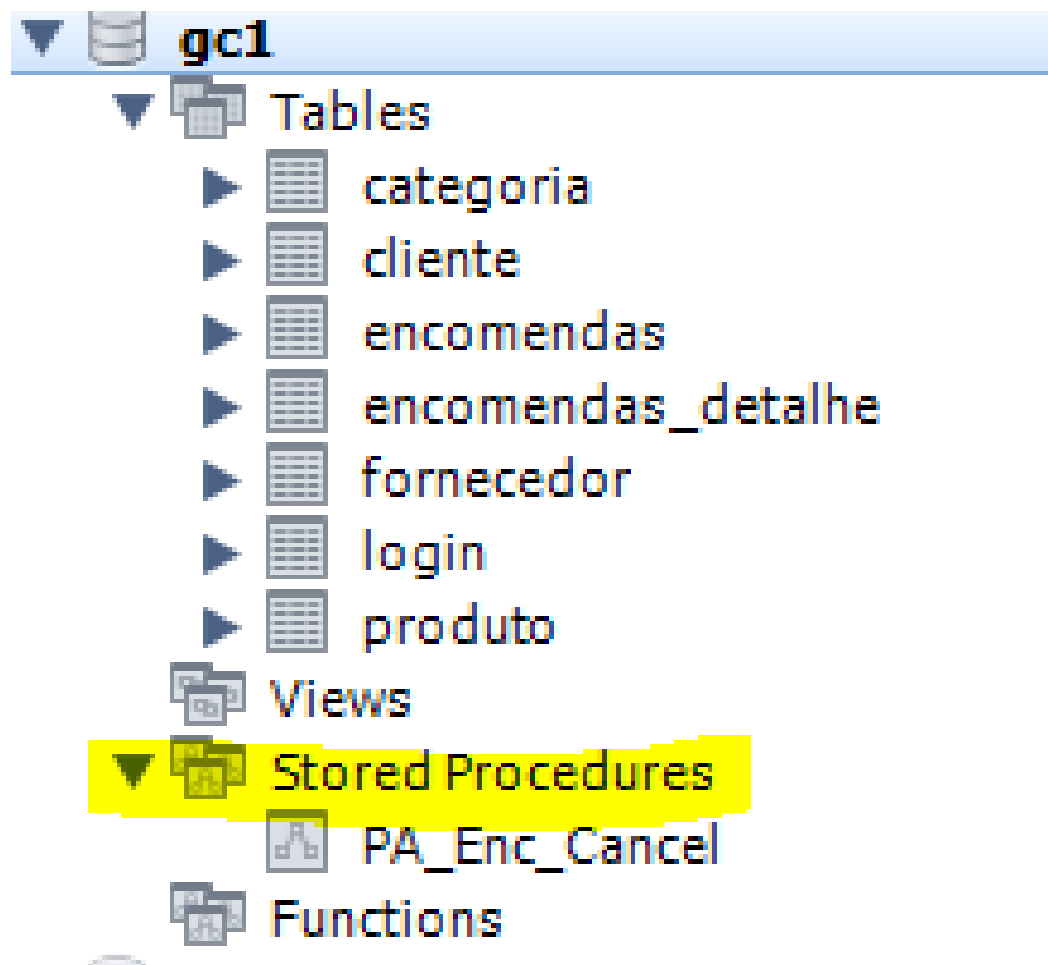
CALL Nome_procedimento [parâmetros];

Ainda poderão ser apagados.

DROP PROCEDURE Nome_procedimento;



Criar procedimentos armazenados





Criar procedimentos armazenados – MYSQL

O SGBD Mysql, apresenta uma sintaxe ligeiramente diferente da sintaxe padrão dos procedimentos armazenados:

- É obrigatório colocar os parâmetros entre parêntesis, e caso um procedimento não tiver parâmetros, os parêntesis devem estar vazios;
- A declaração tem que ser demilitada pelas cláusulas DELIMITER seguida de caracteres delimitadores. Um para delimitar o fim do comando SQL e outro para delimitar o bloco de declaração;



Criar procedimentos armazenados – MYSQL

Exemplo: Criar um procedimento armazenado que liste todas as encomendas canceladas.

```
DELIMITER //
```

```
CREATE PROCEDURE PA_Enc_Cancel()
```

```
BEGIN
```

```
SELECT * FROM Encomendas WHERE Estado
```

```
LIKE 'Cancelada';
```

```
END //
```

```
DELIMITER ;
```

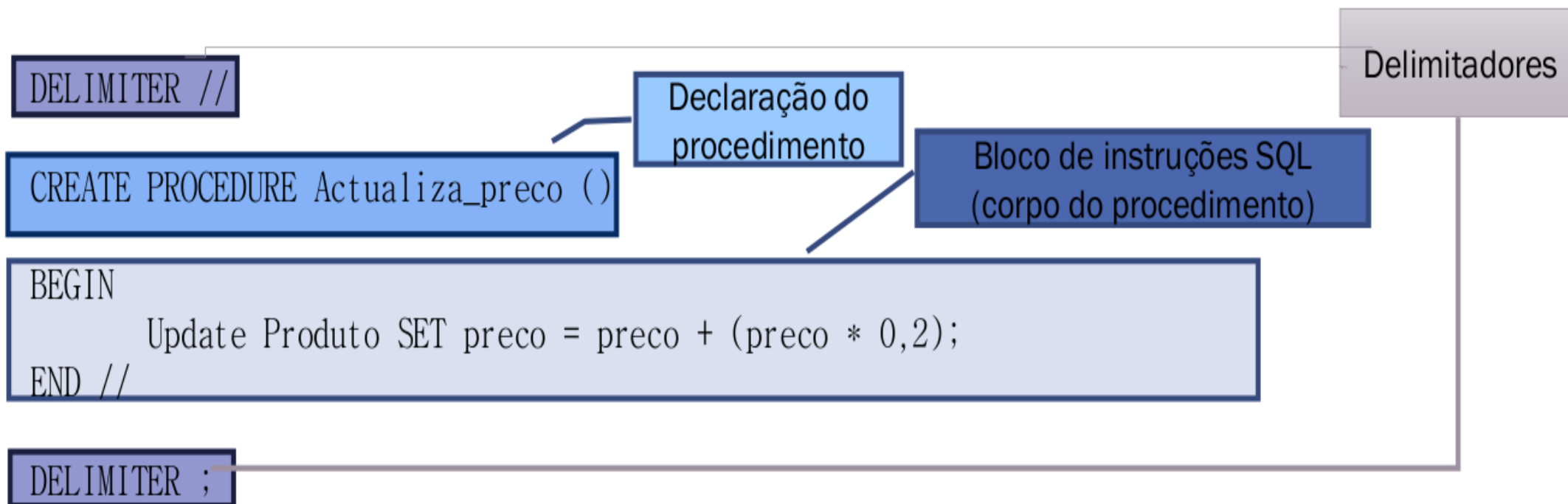
Para executar:

```
CALL PA_Enc_Cancel();
```



Criar procedimentos armazenados – MYSQL

Crie um procedimento armazenado de nome Actualiza_preco que, actualize os preços de todos os produtos, de maneira que fiquem 20% mais caros.





Criar procedimentos armazenados – MYSQL

Após criar o procedimento o mesmo deve ser executado ou chamado.

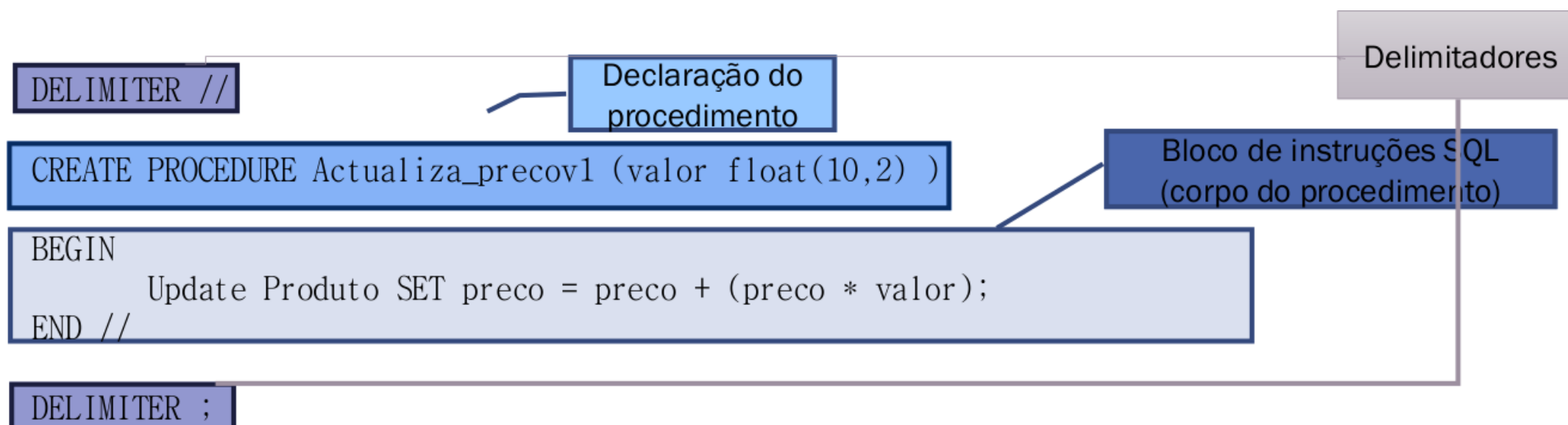
```
CALL Actualiza_preco ();
```

Bloco de instruções SQL
(corpo do procedimento)



Criar procedimentos armazenados – MYSQL

Crie um procedimento armazenado de nome Actualiza_preco que, actualize os preços de todos os produtos, de maneira que fiquem mais caros com base num valor passado pelo utilizador (parâmetro de entrada).





Criar procedimentos armazenados – MYSQL

Após criar o procedimento o mesmo deve ser executado ou chamado.

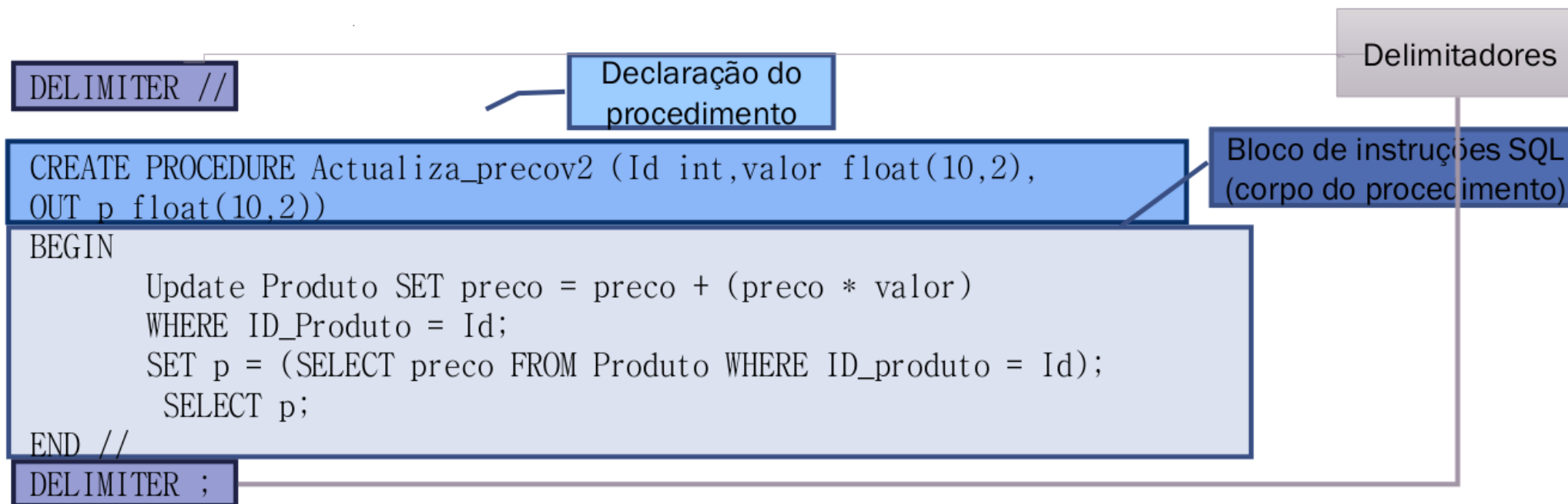
```
CALL Actualiza_precov1 (0.1);
```

Valor do parâmetro de entrada



Criar procedimentos armazenados – MYSQL

Crie um procedimento armazenado de nome Actualiza_preco que, actualize os preços de um produto (parâmetro de entrada), de maneira que fique mais caro com base num valor passado pelo utilizador (parâmetro de entrada). O preço actualizado deve ser devolvido (parâmetro de saída)





Criar procedimentos armazenados – MYSQL

Após criar o procedimento o mesmo deve ser executado ou chamado.

```
CALL Actualiza_precov2 (1, 0.1, @n);
```

```
SELECT @n AS 'Novo Preço' ;
```

Variável por referência para o
parâmetro de saída

Valores dos parâmetros de
entrada

Necessário para visualizar o
novo preço



Estrutura Condicionais – IF

Sintaxe:

IF (condição) THEN

Instruções;

END IF;

Ou ainda:

IF (condição) THEN

Instruções;

ELSE

Instruções;

END IF;



Estrutura Condicionais – IF

Exemplo:

IF (ID<100) THEN

Update fatura set total = total – 10;

ELSE

Update fatura set total = total – 5;

END IF;



Sintaxe:

CASE variável

WHEN valor THEN

Instruções;

END CASE;



Exemplo:

CASE OP

WHEN 1 THEN

Update fatura set total = total – 10;

WHEN 2 THEN

Update fatura set total = total – 5;

END CASE;

Nota: A estrutura CASE pode ser usada directamente no comando SELECT.



Exemplo:

```
SELECT preco, CASE ID_categoria WHEN 1 THEN 'Baixo'  
ELSE 'ALTO' END AS  
'Nive'FROM produto
```



Estrutura de Repetição - WHILE

Sintaxe:

WHILE (condição) DO

Instruções;

Incremento/decremento;

END WHILE; • Exemplo:

WHILE (ID<100) DO

Update fatura set total=total-10;

ID=ID+1;

END WHILE;



Declaração de Variáveis

Sintaxe:

DECLARE variável domínio;

SET variável = valor;

Exemplo:

DECLARE i int;

SET i = 0;



Exercícios

1. Crie um base de dados login. Crie uma tabela utilizador com três atributos: Nome, Utilizador e senha.
 - a) Escreva um PA que verifique se os dados inseridos para o login correspondem com os dados armazenados. O PA deve ter dois parametros de entrada, um para utilizador e outra para a senha. E deve verificar se os dados coincidem com os dados armazenados na base de dados. Caso coincidirem, deve ser enviada uma mensagem de sucesso. Caso não coincidirem deve ser enviada uma mensagem de erro.

2. Use a BD_ Escola:

a) Escreva um PA que faça a inscrição/anulação de um aluno numa determinada disciplina.

Este PA deve receber como parâmetros a operação (aceita somente dois valores 1 e 0) o número do aluno e número da disciplina e deve utilizar a estrutura IF. Se a opção escolhida for 1 então, deve-se fazer a inscrição do aluno na disciplina e consequentemente actualizar o número total de alunos nessa mesma disciplina. Se a opção escolhida for 0, então deve-se eliminar o registo de inscrição desse aluno na respectiva disciplina e actualizar o total de alunos.

b) Reescreva o mesmo PA de modo a funcionar com o CASE.

3. Use a BD_RH:

a) Escreva um PA que deia uma bonificação aos funcionários. Este PA deve ter como parametros um valor base de bonificação (inteiro), um limite de bonificação (inteiro) e o ID do departamento dos funcionarios que sofreram a bonificação. Deve utilizar a estrutura WHILE. Os funcionarios abrangidos deverão ter actualizações consecutivas do salário (salário + valor base) enquanto o limite de bonificação não for atingido.

Exemplo: suponha que o funcionario X actualmente ganha 10.000 Kz.

Se o valor base de bonificação for 100 Kz, e definirmos o limite como 3. O funcionário deverá ter as seguintes actualizações:

10.000 + 100;

10.100 + 100;

10.200 + 100;



INSTITUTO DE TELECOMUNICAÇÕES

MUITO OBRIGADO!