# Banco de Dados II UNIRIO



# Técnicas de Programação em Banco de Dados Procedures – Parte II

Prof. Tadeu Classe tadeu.classe@uniriotec.br



- Até o momento vimos procedimentos se comportando como Views, porém os procedimentos vão além do comportamento de exibição como as viwes;
- É possível realizar dentro de procedimentos operações como exclusão, inclusão, alterações e etc.



- Exercitando:
  - Procedimento para incluir um novo departamento:
    - 1. Crie o SQL para isso:

```
INSERT into departments (dept_no, dept_name)
values ('d010', 'Informática')
```



- Exercitando:
  - Procedimento para incluir um novo departamento:
    - 2. Crie o procedimento:

```
DELIMITER $$
CREATE procedure addDept
(deptNum char(4), depNome varchar(50))
BEGIN
    INSERT into departments (dept_no, dept_name)
    values (deptNum, depNome);
END$$
```



- Exercitando:
  - Procedimento para incluir um novo departamento:
    - 3. Uso do procedimento:

CALL addDept('d011', 'Redes de Computadores');



- Exercitando:
  - Procedimento para incluir um novo departamento:
    - 4. Verifique se realmente funcionou:

### Select \* from departments

	dept_no	dept_name
<b>)</b>	d004	Production
	d005	Development
	d006	Quality Management
	d007	Sales
	d008	Research
	d009	Customer Service
	d010	Informática
	d011	Redes de Computadores
*	NULL	NULL



- Exercitando:
  - Crie um procedimento para incluir novos empregados;



- Declarar variáveis no MySQL é simples:
- Basta utilizar a Sintaxe:

```
SET @<nome variavel> := <valor variavel>
```

Exemplo:

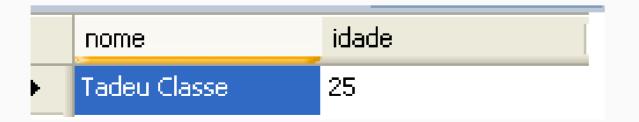


- Em procedimentos as variáveis devem ser "declaradas" através do comando DECLARE.
- Exemplo de um procedimento com variáveis:

```
DELIMITER $$
CREATE PROCEDURE nome_idade()
BEGTN
   DECLARE nome varchar(50);
   DECLARE idade int;
   SET nome = 'Tadeu Classe ';
   SET idade = 25;
   Select nome, idade;
END$$
```



- Em procedimentos as variáveis devem ser "declaradas" através do comando DECLARE.
- Exemplo de um procedimento com variáveis:





```
DELIMITER $$
CREATE procedure addDept
(depNome varchar(50))
BEGIN
    DECLARE qtdeDept int;
    SELECT count(*) + 1
    INTO qtdeDept
    FROM departments;
    INSERT INTO departments(dept_no, dept_name)
   VALUES (concat('d', qtdeDept), depNome);
END$$
```



```
DELIMITER $$
CREATE procedure addDept
                                    Removido o parâmetro
(depNome varchar(50))
                                    de número do
BEGIN
                                    departamento;
    DECLARE qtdeDept int;
    SELECT count(*) + 1
    INTO qtdeDept
    FROM departments;
    INSERT INTO departments(dept_no, dept_name)
    VALUES (concat('d', qtdeDept), depNome);
END$$
```



```
DELIMITER $$
CREATE procedure addDept
(depNome varchar(50))
BEGIN
                                     Declaração da variável
    DECLARE qtdeDept int: _____
                                     de quantidade de
                                     departamentos;
    SELECT count(*) + 1
    INTO qtdeDept
    FROM departments;
    INSERT INTO departments(dept_no, dept_name)
    VALUES (concat('d', qtdeDept), depNome);
END$$
```



```
DELIMITER $$
CREATE procedure addDept
(depNome varchar(50))
BEGIN
    DECLARE qtdeDept int:
                                   Busca da quantidade de
    SELECT count(*) + 1
                                   departamentos + 1.
    INTO qtdeDept
                                   Sendo colocado o valor
    FROM departments;
                                   na variável criada.
    INSERT INTO departments(dept_no, dept_name)
    VALUES (concat('d', qtdeDept), depNome);
```



```
DELIMITER $$
CREATE procedure addDept
(depNome varchar(50))
BEGIN
    DECLARE qtdeDept int;
                                  Inclusão do código do
    SELECT count(*) + 1
                                  departamento
    INTO qtdeDept
                                  concatenado ao seu
    FROM departments;
                                  prefixo.
    INSERT INTO departments(dept_no, dept_name)
    VALUES (concat('d', qtdeDept), depNome);
```



- Exercícios (Utilizando variáveis):
  - Crie um procedimento que passe por parâmetros os dados para a inclusão de um novo funcionário. Crie variável para controlar o auto incremento do seu campo primário. Inclua no banco de dados um novo registro;
  - Crie um procedimento que passe por parâmetro o nome do departamento e do funcionário. Crie variáveis que receba o código deste funcionário e deste departamento. Inclua um novo registro na tabela dept\_emp;



- Exercícios (Utilizando variáveis):
  - Crie um procedimento que passe o nome de um funcionário, se seu título por parâmetros.
     Recupere em uma variável seu código. Inclua o registro na tabela titles.
  - Crie um procedimento que passe o nome de um funcionário por parâmetro. Recupere em uma variável seu código e inclua o registro na tabela salaries.



- Os procedimentos são estruturas extremamente poderosas que facilitam a interação dos usuários com o servidor de banco de dados, além de ajudar no controle de acesso aos dados, aumentando a segurança do sistema, como mencionado anteriormente.
- Um dos requisitos que contribuem para essas características dos procedimentos está na possibilidade de uso de estruturas de controle de fluxo de dados, como estruturas de decisão e de repetição.



- As estruturas de controle de fluxo definidas no MySQL são: IF, CASE, LOOP, LEAVE, ITERATE, REPEAT e WHILE.
- Vamos discutir as duas mais usadas, o IF, que define estruturas de decisão, e o WHILE, que define estruturas de repetição.
- Abaixo apresenta a estrutura do IF em MySQL:

```
IF condição 1 THEN
comandos em SQL
Else
comandos em SQL
END IF;
```



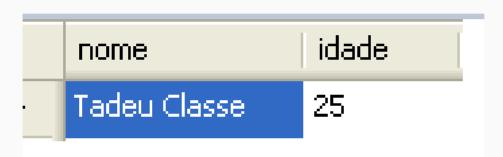
Exemplo:

```
DELIMITER $$
CREATE PROCEDURE nome_idade
(nome varchar(50), idade int)
BEGIN
    if((nome != '') and (idade > 0)) then
        Select nome, idade;
    else
        Select 'Nome ou idade Inválidos!';
    end if;
END$$
```



Exemplo:

Call nome\_idade('Tadeu Classe', 25)



Call nome\_idade('', 25)

Nome ou idade Inválidos!

Nome ou idade Inválidos!



#### Exercícios:

 Altere os procedimentos dos exercícios anteriores para verificar se o registros retornados para as variáveis são válidos ou não. Caso sejam válidos as inclusões podem ocorrer normalmente. Se não forem válidos, o banco de dados deve retornar uma mensagem de erro.



WHILE, Exemplo:

```
DELIMITER $$
CREATE PROCEDURE nome_idade
(nome varchar(50), idade int)
BEGIN
    declare cont int;
    if((nome != '') and (idade > 0)) then
        set cont = 1;
        while (cont < idade) DO
                Select nome, cont, 'Crescendo...' periodo;
                set cont = cont +1;
        End while;
        Select nome, idade, 'Cresceu!' periodo;
    else
        Select 'Nome ou idade Inválidos!';
    end if;
END$$
```



#### Exercícios:

- Crie um procedimento que altere os salário de funcionários que tenham o título de desenvolvedor em 10% do valor;
- Crie um procedimento para alterar o salário de todos os funcionários em 5% de acréscimo;
- Crie um procedimento que altere todos os títulos de quem é Desenvolvedor para Engenheiro;



- Exercícios:
  - Crie um procedimento para despedir (excluir) todos os funcionários que tenha mais de 55 anos;
  - Crie um procedimento para alterar o salário de um determinado funcionário.
  - Em todos os exercícios devem ser realizadas verificações de informações válidas;

#### Conclusão



- Nesta unidade, estudamos a criação e utilização dos procedimentos armazenados.
- Aprendemos a usar os comandos CREATE PROCEDURE, CALL e DROP PROCEDURE para criar, executar e apagar um procedimento.
- Aprendemos como utilizar o recurso de lista de parâmetros para definir parâmetros de entrada e saída para os procedimentos.
- Vimos também o uso dos procedimentos em conjunto com as estruturas de controle de fluxo de dados, IF ... END IF e WHILE ... END WHILE.

# Obrigado e Até a Próxima Aula!

#### Contato:

Prof. Tadeu Classe
Bacharelado em Sistemas de
Informação (BSI)
Universidade Federal do Estado do
Rio de Janeiro (UNIRIO)



O trabalho Banco de Dados II - Universidade Federal do Estado do Rio de Janeiro (UNIRIO) de Tadeu Moreira de Classe está licenciado com uma Licença Creative Commons - Atribuição-NãoComercial-Compartilhalgual 4.0 Internacional.

