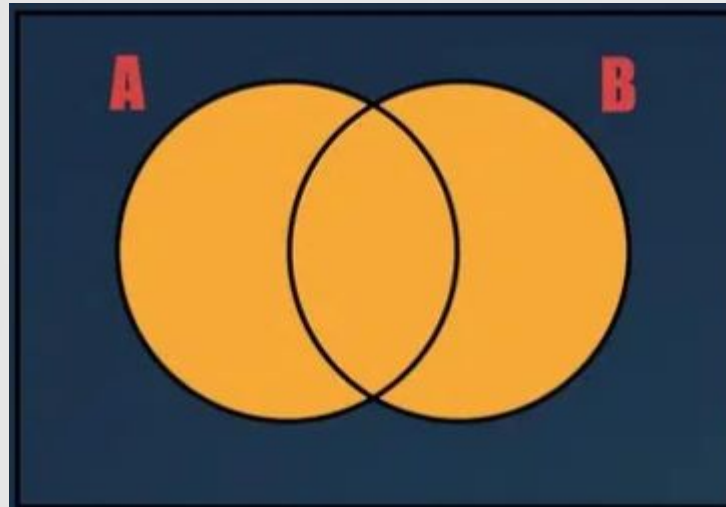


# Aula 07 e TDE 07 - 21/02

## aula 7

## <Join>

INNER JOIN - A cláusula JOIN é uma forma de relacionar duas tabelas e gerar uma tabela resultante

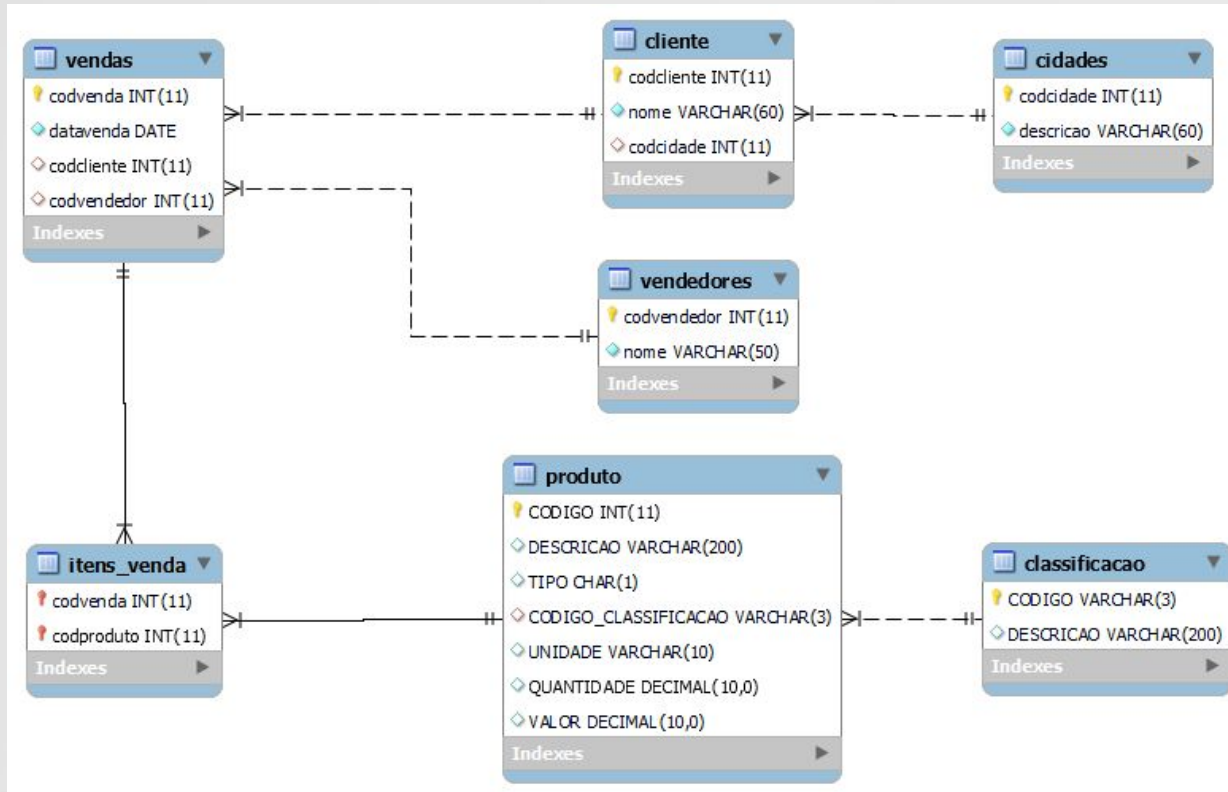


## <Join>

```
SELECT *  
FROM produto p  
  join classificacao c on p.CODIGO_CLASSIFICACAO = c.CODIGO
```

# <DER>

## Diagrama de entidade relacionamento





## <Join>



- 1) Qual Cliente foi a venda ?
  - 2) Qual Cidade é o cliente ?
  - 3) Quais produtos ele comprou ?
  - 4) Qual quantidade de cada produto ele comprou ?
  - 5) Qual é Classificação do produto ?
  - 6) Qual nome vendedor ?
  - 7) Qual valor total de cada produto ?
-

## <Join>

```
SELECT
    c.nome as 'cliente',
    ci.descricao as 'cidade',
    p.CODIGO as 'cod.prod',
    p.DESCRICAO as 'desc.prod',
    p.QUANTIDADE as 'qt.prod',
    P.VALOR AS 'vl.prod',
    cla.DESCRICAO as 'desc.classificacao',
    vd.nome as 'vendedor',
    p.QUANTIDADE * p.VALOR as 'TOTAL'
FROM vendas v
    JOIN cliente c on v.codcliente = c.codcliente
    JOIN cidades ci on ci.codcidade = c.codcidade
    JOIN itens venda iv on iv.codvenda = v.codvenda
    JOIN produto p on iv.codproduto = p.CODIGO
    join classificacao cla on cla.CODIGO = p.CODIGO CLASSIFICACAO
    join vendedores vd on vd.codvendedor = v.codvendedor;
```

## <Join>

- 1) Quantidade de produtos vendidos ?
- 2) Quantidade de vendas por vendedor ?
- 3) Melhor vendedor nos 3 primeiros meses do ano ?
- 4) A melhor venda ( valor mais alto) ?
- 5) Qual é o valor da Média de vendas por mês ?
- 6) Qual cidade compra mais ?
- 7) Qual é o melhor cliente ?
- 8) O pior vendedor ?
- 9) Comissão de 10% para o vendedor que seu total de vendas que for acima da média do mês de todos vendedores ?

# <FUNCTION>

No Mysql uma FUNCTION é uma função armazenada onde é possível passar parâmetros para ela e então retornar um valor.

Utilize o modelo de código abaixo para realizar a criação de um FUNCTION:

```
DELIMITER $$  
    CREATE FUNCTION nome_function (parâmetros)  
    RETURNS tipo_dado  
    DETERMINISTIC  
    BEGIN  
  
        /*CORPO DA FUNÇÃO*/  
  
        RETURN retorna_valor  
    END $$  
DELIMITER ;
```



# <FUNCTION>

- **nome\_função:** É o nome pelo qual a função armazenada é chamada.
- **parametro\_da\_função:** É o argumento cujo valor é usado pela função dentro do seu corpo(de **Begin a End;**).
- tipo de dados: é o tipo de dados do valor retornado pela função.
- [NOT] DETERMINISTIC: uma função armazenada pode ser **DETERMINISTIC** ou **NOT DETERMINISTIC**, na qual é especificada em sua declaração.
- Uma função é considerada “determinística” se sempre produzir o mesmo resultado para os mesmos parâmetros de entrada. Caso contrário, será “não determinística”. Se você não usar DETERMINISTIC ou NOT DETERMINISTIC, o MySQL usará por padrão a opção NOT DETERMINISTIC.
- O **DELIMITER** serve para indicar o final do conjunto de instruções. Por muito tempo, o delimitador sempre foi um ponto e vírgula. Isso causa um problema, porque em uma função armazenada, pode-se ter muitas declarações, e cada uma deve terminar com um ponto e vírgula.

# <FUNCTION>

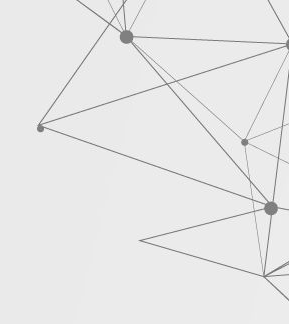

```
CREATE FUNCTION olar (nome CHAR(20)) RETURNS CHAR(50)  
RETURN CONCAT('Olaaaaaaa, ', nome, ' !');
```

```
select olar ("johnny");
```



# <FUNCTION>

O functions retornam valores e as procedures podem ou não retornar um valor. As functions tem duas características que diferem das procedures, as quais não podemos deixar de tratar:

- As functions sempre retornam valores
  - Functions são usadas como parte de uma expressão.
- 
- 

# <Procedure>

- Stored procedures são rotinas definidas no banco de dados, identificadas por um nome pelo qual podem ser invocadas. Um procedimento desses pode executar uma série de instruções, receber parâmetros e retornar valores.

# <Procedure>

## Pontos positivos:

- Simplificação da execução de instruções **SQL** pela aplicação;
- Transferência de parte da responsabilidade de processamento para o servidor.
- Facilidade na manutenção, reduzindo a quantidade de alterações na aplicação.

## Pontos negativos:

- Necessidade de maior conhecimento da sintaxe do banco de dados para escrita de rotinas em SQL;
- As rotinas ficam mais facilmente acessíveis. Alguém que tenha acesso ao banco poderá visualizar e alterar o código.

# <Procedure>

```
DELIMITER $$
```

```
CREATE PROCEDURE nome_procedimento (parâmetros)
```

```
BEGIN
```

```
/*CORPO DO PROCEDIMENTO*/
```

```
END $$
```

```
DELIMITER ;
```

# <Procedure>

PARAMETROS:

```
CREATE PROCEDURE Selecionar_Produtos(IN quantidade INT, IN  
categoria varchar(3), out saldo decimal)
```



# <Procedure>



O “nome” dos parâmetros também segue as mesmas regras de definição de variáveis.

O “TIPO” nada mais é que do tipo de dado do parâmetro (int, varchar, decimal, etc).

O “MODO” indica a forma como o parâmetro será tratado no procedimento, se será apenas um dado de entrada, apenas de saída ou se terá ambas as funções. Os valores possíveis para o modo são:

- IN: indica que o parâmetro é apenas para entrada/recebimento de dados, não podendo ser usado para retorno;
- OUT: usado para parâmetros de saída. Para esse tipo não pode ser informado um valor direto (como ‘teste’, 1 ou 2.3), deve ser passada uma variável “por referência”;
- INOUT: como é possível imaginar, este tipo de parâmetro pode ser usado para os dois fins (entrada e saída de dados). Nesse caso também deve ser informada uma variável e não um valor direto.



# <Procedure>

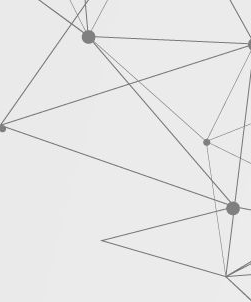
Para executar uma PROCEDURE se usa a palavra reservada CALL ex:

```
CALL Selecionar_Produtos (10 , '003' , @SALDO) ;
```



## <View>

Bem como o próprio nome diz, **view** é uma visão de dados. Muito utilizado quando queremos simplificar a maneira como acessamos um result set complexo ou até mesmo para dar acesso somente leituras em determinadas tabelas.



## <View>

```
CREATE VIEW nome_da_view AS SELECT * FROM nome_tabela;
```

## <View>

View que aplica 10% de reajuste na categoria 003

```
create view vw_ajuste as select p.*,  
    (p.VALOR * 10)/100 + p.VALOR as reajuste from produto p  
where p.CODIGO_CLASSIFICACAO = '003';
```

```
select * from vw_ajuste;
```