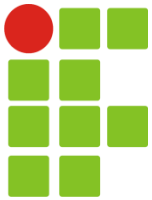


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

# BANCO DE DADOS
# LINGUAGEM SQL - JUNÇÕES

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

# Select



- Comando utilizado para selecionar tuplas de uma ou mais tabelas.

```
SELECT coluna1, coluna2, coluna3
FROM tabela_nome1, tabela_nome2
WHERE coluna1=valor1
AND    coluna2=valor2
OR     coluna2=valor3
GROUP BY coluna1
HAVING AVG(coluna1) > 100
ORDER BY coluna2;
```

# Junções



- Até o momento temos consultas acessando apenas uma tabela.
- E quando temos duas tabelas ligadas por uma chave estrangeira ?? Como realizar essa junção ??
- Utilizando o comando SELECT podemos acessar várias tabelas.

```
SELECT FUNC.NOME, DEP.NOME
FROM funcionario FUNC,
        dependente DEP
WHERE FUNC.cod = DEP.cod_func;
```

# Tipos de junções



- Existem alguns tipos de junção:
  - Junção de produto cartesiano
  - Junção Interna
  - Junção Externa

# Junção de produto cartesiano



- É uma junção entre duas tabelas que origina uma "terceira tabela" constituída por todos os elementos da primeira combinadas com todos os elementos da segunda.

# Junção de produto cartesiano



| COD | NOME | ESPECIALIDADE |
|---|---|---|
| 001 | JOSÉ | ENGENHEIRO |
| 002 | JOÃO | MESTRE DE OBRAS |
| 003 | MARIA | CONTABILISTA |

| COD | NOME | COD_FUNC |
|---|---|---|
| 001 | PEDRO | 001 |
| 002 | ALICE | 001 |
| 003 | LUANA | 003 |

```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC,
        dependente DEP
WHERE FUNC.cod = DEP.cod_func;
```

| NOME_FUNC | NOME_DEP |
|---|---|
| JOSÉ | PEDRO |
| JOSÉ | ALICE |
| MARIA | LUANA |

# Junção Interna



- Funciona de forma semelhante à junção de produto cartesiano.
- Porém, utiliza uma sintaxe diferente.

```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC INNER JOIN
        dependente DEP ON (FUNC.COD = DEP.COD_FUNC);
```

# Junção Interna



| COD | NOME | ESPECIALIDADE |
|---|---|---|
| 001 | JOSÉ | ENGENHEIRO |
| 002 | JOÃO | MESTRE DE OBRAS |
| 003 | MARIA | CONTABILISTA |

| COD | NOME | COD_FUNC |
|---|---|---|
| 001 | PEDRO | 001 |
| 002 | ALICE | 001 |
| 003 | LUANA | 003 |

```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC INNER JOIN
        dependente DEP ON (FUNC.COD = DEP.COD_FUNC);
```

| NOME_FUNC | NOME_DEP |
|---|---|
| JOSÉ | PEDRO |
| JOSÉ | ALICE |
| MARIA | LUANA |

# Junção Externa



- Retorna um valor nulo (null) para o correspondente que não encontrar.
- Existem vários padrões de junção externa, os principais são:
  - LEFT OUTER JOIN (Junção externa esquerda)
  - RIGHT OUTER JOIN (Junção externa direita)

```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC [LEFT OU RIGHT] OUTER JOIN
        dependente DEP ON (FUNC.COD = DEP.COD_FUNC);
```

# LEFT OUTER JOIN



| COD | NOME | ESPECIALIDADE |
|---|---|---|
| 001 | JOSÉ | ENGENHEIRO |
| 002 | JOÃO | MESTRE DE OBRAS |
| 003 | MARIA | CONTABILISTA |

| COD | NOME | COD_FUNC |
|---|---|---|
| 001 | PEDRO | 001 |
| 002 | ALICE | 001 |
| 003 | LUANA | 003 |

```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC LEFT OUTER JOIN
        dependente DEP ON (FUNC.COD = DEP.COD_FUNC);
```

| NOME_FUNC | NOME_DEP |
|---|---|
| JOSÉ | PEDRO |
| JOSÉ | ALICE |
| MARIA | LUANA |
| JOÃO | null |

# RIGHT OUTER JOIN



| COD | NOME | ESPECIALIDADE |
|---|---|---|
| 001 | JOSÉ | ENGENHEIRO |
| 002 | JOÃO | MESTRE DE OBRAS |
| 003 | MARIA | CONTABILISTA |

| COD | NOME | COD_FUNC |
|---|---|---|
| 001 | PEDRO | 001 |
| 002 | ALICE | 001 |
| 003 | LUANA | 003 |
| 004 | PAULO | null |

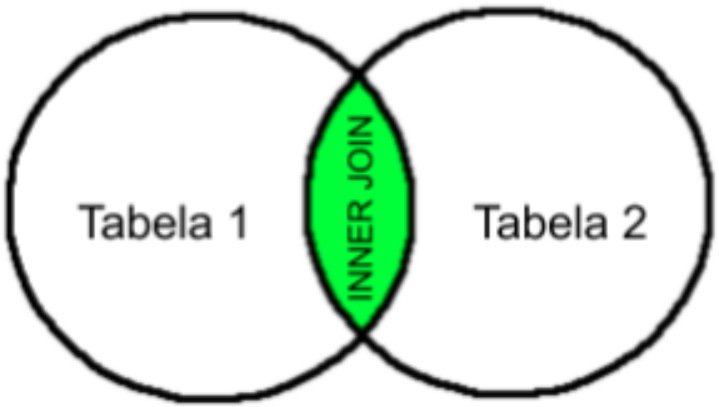
```
SELECT FUNC.NOME NOME_FUNC, DEP.NOME NOME_DEP
FROM funcionario FUNC RIGHT OUTER JOIN
        dependente DEP ON (FUNC.COD = DEP.COD_FUNC);
```

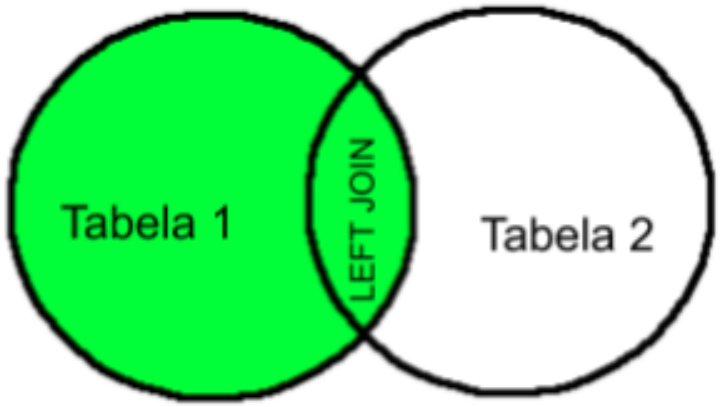
| NOME_FUNC | NOME_DEP |
|---|---|
| JOSÉ | PEDRO |
| JOSÉ | ALICE |
| MARIA | LUANA |
| null | PAULO |

# INNER JOIN LEFT JOIN



## Diferença entre INNER JOIN e LEFT JOIN




Enquanto o INNER JOIN combina todos os valores das duas tabelas e retorna no resultado somente as linhas presentes em ambas, o LEFT JOIN traz todas as linhas presentes na tabela 1 (ou tabela da esquerda) com os valores correspondentes da tabela 2.

# VIEWS



## O que é uma view?

- Uma view é uma maneira alternativa de observação de dados de uma ou mais entidades (tabelas), que compõem uma base de dados. Pode ser considerada como uma tabela virtual ou uma consulta armazenada.

# VIEWS



**Onde se aplicam as views?**

- Geralmente e recomendável, uma view, implementada encapsulando uma instrução SELECT (busca de dados para exposição), guarda os dados em uma tabela virtual, armazenando também em cache, pois todas as consultas ao banco, encapsuladas ou não, ao serem executadas, são armazenadas em cache. Por este motivo, pode ser mais rápido ter uma consulta armazenada em forma de view, em vez de ter que retrabalhar uma instrução.

# VIEWS



**As vantagens de se usar views são:**

**Economizar tempo com retrabalho;**

Ex.: Você não precisar escrever aquela instrução enorme. Escreva uma vez e armazene!

**Velocidade de acesso às informações;**

Ex.: Uma vez compilada, o seu recordset (conjunto de dados) é armazenado em uma tabela temporária (virtual).

# VIEWS



**Mascarar complexidade do banco de dados;**

Ex.: As views isolam do usuário a complexidade do banco de dados. Nomes de domínios podem ser referenciados com literais e outros recursos. Isso proporciona aos desenvolvedores a capacidade de alterar a estrutura sem afetar a interação do usuário com o banco de dados.

# VIEWS



**Simplifica o gerenciamento de permissão de usuários;**

Ex.: Em vez de conceder permissão para que os usuários contem tabelas base, os proprietários de bancos de dados podem conceder permissões para que os usuários consultem dados somente através de views. Isso também protege as alterações na estrutura das tabelas base subjacentes. Os usuários não serão interrompidos durante uma visualização de dados.

# VIEWS



**Organizar dados a serem exportados para outros aplicativos;**

Ex.: Você pode criar uma view baseada em uma consulta complexa, que associe até 32 tabelas e depois exportar dados para outro aplicativo para análise adicional. Pode ser gerado um arquivo de DUMP* automaticamente.

# VIEWS



**Criando Views**

Para definir Views em um banco de dados, utilize a declaração CREATE VIEW, a qual tem a seguinte sintaxe:

CREATE [OR REPLACE]VIEW name view [(column_list)]
AS select_statement
FROM tabelas
WHERE condições;

# VIEWS



**Criando Views**

Ex.:

```
CREATE [OR REPLACE]VIEW vw_dadosCliente
AS select c.nome AS nomeCliente d.nome AS nomeDep
FROM cliente c, departamento d
WHERE c.cod_dep = d.codigo_dep;
```