

---

# Laboratório de Banco de Dados I

## Parte 04 – Junções

Prof. Daniel Callegari  
Faculdade de Informática – FACIN – PUCRS

---

### 1. Consultas sobre múltiplas tabelas

Para consultar dados de uma ou mais tabelas relacionadas devemos utilizar operações de junções (JOINS). A junção de duas ou mais tabelas é equivalente – em termos de resultado final – à realização do produto cartesiano, comparando o valor de certos atributos, e aplicando uma projeção e uma seleção ao resultado.

Dica: a utilização de ALIASES (para nomes de tabelas) em junções facilita a sua leitura.

### 2. Produto Cartesiano

O produto cartesiano é uma operação da teoria de conjuntos. Executar um produto cartesiano entre duas tabelas resulta na combinação de todas as linhas (registros) da primeira tabela com todas as linhas da segunda tabela.

Relação R		Relação S		
A	B	B	C	D
1	2	2	5	6
3	4	4	7	8
		9	10	11

O produto cartesiano é representado pelo símbolo de operação “X” entre os conjuntos.

Em SQL, chamamos essa operação de CROSS JOIN.

Resultado de R x S (Prod. Cartesiano)

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Repare que as suas tabelas possuem uma coluna com o nome “B”, o que cria ambiguidade na tabela resultante. Usamos o nome da tabela e um ponto na frente de um nome de coluna para resolver a ambiguidade.

Exemplo:

```
SELECT EST.uf, EST.nome, CID.uf, CID.nome
FROM ESTADOS EST CROSS JOIN CIDADES CID;
```

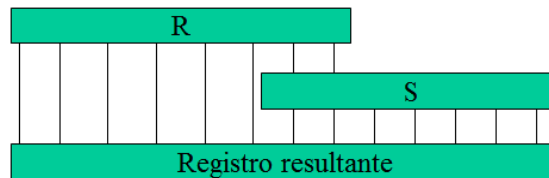
O total de linhas retornadas é a multiplicação do número de registros da primeira tabela pelo número de registros da segunda tabela (por quê?).

Na maioria das vezes, isso gera resultados sem utilidade.

Observe o resultado dessa consulta:

ESTADOS		CIDADES	
UF	NOME	UF	NOME
AC	Acre ???	MG	Central de Minas
AC	Acre	MA	Central do Maranhão
AC	Acre	MG	Centralina
AC	Acre	MA	Centro do Guilherme
AC	Acre	MA	Centro Novo do Maranhão
AC	Acre	RO	Cerejeiras
AC	Acre	GO	Ceres
...		...	

Logicamente, faz mais sentido obter apenas os registros que combinam entre as duas tabelas (ou seja, que se correspondem de alguma forma).



Seguindo nosso exemplo...

Relação R		Relação S		
A	B	B	C	D
1	2	2	5	6
3	4	4	7	8
		9	10	11

Resultado de R x S (Prod. Cartesiano)

A	R.B	S.B	C	D
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

→

A	B	C	D
1	2	5	6
3	4	7	8

Para tanto, precisamos entender o que são EQUI-JOINS


### 3. EQUI-JOINS

Um EQUI-JOIN relaciona linhas de uma tabela com as linhas de outra tabela, a partir de um critério de igualdade (ou equivalência). Normalmente esta igualdade ocorre entre uma PK (da tabela-pai) e uma FK (da tabela-filho). Em SQL usamos INNER JOIN:

```
SELECT EST.uf, EST.nome, CID.uf, CID.nome
FROM ESTADOS EST INNER JOIN CIDADES CID
ON EST.uf = CID.uf;
```

Compare o resultado do INNER JOIN com o resultado do CROSS JOIN. Repare que em todas as linhas retornadas, a UF da cidade corresponde à UF do estado:

ESTADOS		CIDADES	
UF	NOME	UF	NOME
MG	Minas Gerais	MG	Central de Minas
MA	Maranhão	MA	Central do Maranhão
MG	Minas Gerais	MG	Centralina
MA	Maranhão	MA	Centro do Guilherme
MA	Maranhão	MA	Centro Novo do Maranhão
RO	Rondônia	RO	Cerejeiras
...		...	

 Quantos registros no total são retornados neste caso? É possível calcular antecipadamente esse valor?

Dica: Os dois comandos a seguir são equivalentes (tanto faz usar um ou o outro). O primeiro comando está escrito na forma mais antiga, enquanto que o segundo comando está em uma forma mais moderna e preferida, porque não confunde o critério do JOIN com demais critérios de seleção das linhas (na cláusula WHERE):

```
-- Forma mais antiga
SELECT EST.uf, EST.nome, CID.uf, CID.nome
FROM ESTADOS EST , CIDADES CID
WHERE EST.uf = CID.uf;

-- Forma mais moderna (preferida)
SELECT EST.uf, EST.nome, CID.uf, CID.nome
FROM ESTADOS EST INNER JOIN CIDADES CID
ON EST.uf = CID.uf;
```

### 3. NATURAL JOIN

Um NATURAL JOIN é um EQUI-JOIN cuja condição é estabelecida pela igualdade entre as colunas de mesmo nome da tabela-pai e da tabela-filho (tantas quantas forem). Das colunas de mesmo nome das tabelas apenas uma comporá o resultado:

```
SELECT
    END.rua, END.numero, END.complemento, CID.nome
FROM
    ENDERECOS END NATURAL JOIN CIDADES CID;
```

Observação: Nem todo SGBD suporta a operação NATURAL JOIN.

## 4. Consultando dados a partir de mais de duas tabelas

Há ocasiões em que precisamos buscar dados de mais de duas tabelas. Em outros casos, os dados de que precisamos encontram-se em tabelas mais “distantes” no esquema do banco de dados. Para poder obter esses dados, precisamos usar JOINS encadeados.

✎ Antes de prosseguir, execute os *scripts* de criação das demais tabelas do banco de dados do nosso Estudo de Caso.

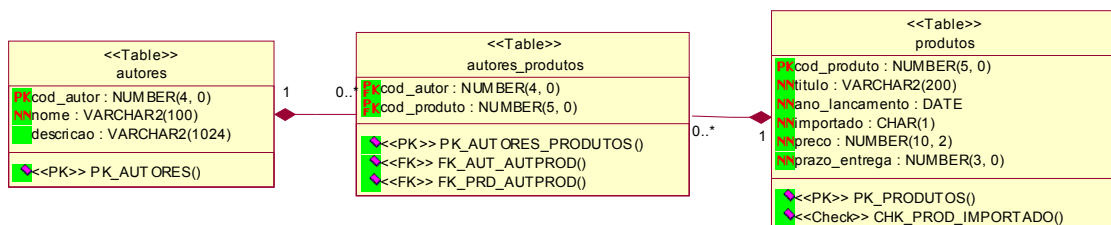
✎ Reserve também um tempo para executar os scripts que “populam” as tabelas com dados de exemplo. Se necessário, peça ajuda ao professor.

### 4.1. JOINS encadeados

Observação: os exemplos a partir desse ponto assumem que você já rodou os *scripts* do Estudo de Caso da disciplina.

✎ Antes de prosseguir, examine com cuidado o esquema do banco de dados do Estudo de Caso.

Para exemplificar o uso de JOINS encadeados, vamos selecionar o nome dos autores e o título dos livros (produtos) que eles escreveram.



```
SELECT AU.nome, PROD.titulo
FROM
    AUTORES AU    NATURAL JOIN    AUTORES_PRODUTOS AP
                NATURAL JOIN    PRODUTOS PROD;
```

## 4.2. Precedência de JOINS

A fim de melhor controlar a precedência de joins encadeados, devem ser utilizados parênteses. Por exemplo: vamos selecionar o nome dos clientes e o número com DDD de seus telefones.


```
SELECT USU.nome, TEL.ddd, TEL.numero
FROM
    USUARIOS USU INNER JOIN CLIENTES CLI
        ON USU.COD_USUARIO = CLI.COD_CLIENTE
    NATURAL JOIN TELEFONES TEL
```

Formas alternativas:

```
SELECT USU.nome, TEL.ddd, TEL.numero
FROM
    (USUARIOS USU INNER JOIN CLIENTES CLI
        ON USU.COD_USUARIO = CLI.COD_CLIENTE)
    NATURAL JOIN TELEFONES TEL

--- ou ---

SELECT USU.nome, TEL.ddd, TEL.numero
FROM
    USUARIOS USU INNER JOIN
        (CLIENTES CLI NATURAL JOIN TELEFONES TEL)
        ON USU.COD_USUARIO = CLI.COD_CLIENTE
```

 Para pensar: o que ocorre quando há valores nulos em campos de tabelas e executamos JOINS entre elas?

## 5. Junções externas

Em um EQUI-JOIN somente compõem o resultado as linhas da tabela-pai e da tabela-filha que tenham a condição atendida.

No entanto, podem existir linhas não relacionadas por possuírem valor NULL na coluna FK. Em uma junção externa, ou OUTER JOIN, é possível definir que todas as linhas de determinada tabela (pai ou filha) farão parte do resultado, inclusive as que forem NULL nas colunas da condição.

Exemplo: selecionar o nome de todos os clientes e, quando tiverem, seus números de telefone:

```
SELECT      USU.nome,  
            DECODE (TEL.NUMERO,  
                    NULL, 'SEM TELEFONE',  
                    TEL.NUMERO)  
FROM  
    USUARIOS USU INNER JOIN CLIENTES CLI  
        ON USU.COD_USUARIO = CLI.COD_CLIENTE  
    LEFT OUTER JOIN TELEFONES TEL  
        ON CLI.COD_CLIENTE = TEL.COD_CLIENTE
```

Dica: A palavra OUTER é opcional.

Dica: A função DECODE é uma função proprietária da Oracle. Usamos aqui apenas como complemento ao exemplo. A função compara o primeiro argumento com o segundo. Se forem iguais, a função retorna o terceiro argumento. Caso contrário, a função retorna o valor do seu quarto argumento. Como alternativa, você pode usar expressões CASE (não abordadas neste material).

Vamos agora ver outro exemplo: selecionar os títulos de todos os produtos e o nome de todos os autores, relacionando-os quando possível:

```
SELECT PROD.titulo, AUT.nome  
FROM  
    (PRODUTOS PROD NATURAL LEFT OUTER JOIN  
        AUTORES_PRODUTOS)  
    NATURAL RIGHT OUTER JOIN AUTORES AUT
```

Para incluir todas as linhas relacionadas ou não na consulta, pode-se empregar um FULL OUTER JOIN. Exemplo: se houvesse estados sem cidades e cidades sem estado:

```
SELECT EST.nome, CID.nome  
FROM ESTADOS EST FULL OUTER JOIN CIDADES CID  
    ON EST.UF = CID.UF
```

## Dicas finais desta aula

- Lembre-se de usar apelidos (alias) para tabelas ao usar junções para facilitar a redação do comando.
- O tipo de JOIN mais usado é o EQUI-JOIN.
- Para buscar dados de N tabelas relacionadas, usamos N-1 JOINS.
- As junções externas também incluem resultados quando não há correspondência entre os dados das tabelas.

## Fechamento

Parabéns! Você aprendeu como trabalhar com o conceito de Junções em SQL.

1. INNER JOINS são mais comuns do que as outras formas.
2. Há outras maneiras de combinar resultados de tabelas distintas usando operações da teoria de conjuntos (união, interseção, subtração). Pesquise a respeito.

### *Registro de alterações deste documento*

<i>Data</i>	<i>Autor</i>	<i>Alterações</i>
27/08/2012	Daniel Callegari	Primeira versão.
26/09/2012	Daniel Callegari	Correção: alias para a tabela telefones nos exemplos (TEL).

-X-