

---

# SQL - Structured Query Language

## Unidade 2 – Manipulação Básica de Dados – Parte 4

Prof. Daniel Callegari  
Escola Politécnica – PUCRS

---

Até o momento, consultamos dados a partir de uma única tabela. Esta parte do material introduz o conceito de JUNÇÕES, que permitem a consulta de dados de mais de uma tabela, através da relação entre chaves estrangeiras e chaves primárias.

### 1. Consultas sobre múltiplas tabelas

A integridade de entidade define uma linha como entidade exclusiva de uma determinada tabela.

Para consultar dados de uma ou mais tabelas relacionadas devemos utilizar operações denominadas “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

A	B
1	2
3	4

Relação S

B	C	D
2	5	6
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 tuas 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:

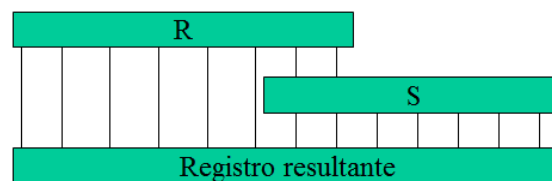
```
SELECT EST.uf, EST.nome, CID.uf, CID.nome
FROM ESTADOS EST CROSS JOIN CIDADES CID;
```

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 essa 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 os 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          ← repare na vírgula
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;
```

## INSTALAÇÃO DO ESTUDO DE CASO (somente Oracle)

✎ Antes de prosseguir, execute os *scripts* de criação das tabelas do banco de dados do nosso Estudo de Caso, disponível no Moodle.

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

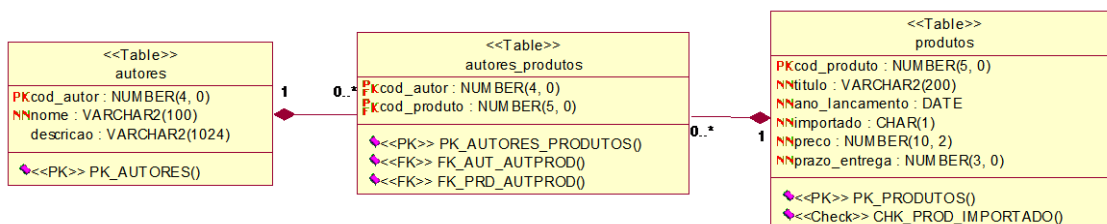
Há ocasiões em que precisamos buscar dados de mais de duas tabelas. Em outros casos, os dados de que precisamos encontram-se em tabelas mais “distantes” no esquema do banco de dados. Para poder obter esses dados, precisamos usar JOINS encadeados.

### 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
    JOIN AUTORES_PRODUTOS AP
        ON (AU.cod_autor = AP.cod_autor)
    JOIN PRODUTOS PROD
        ON (AP.cod_produto = PROD.cod_produto);
```

## 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
    INNER JOIN  TELEFONES TEL
        ON _____;      ← complete
```

Formas alternativas:

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

--- ou ---

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

 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 LEFT OUTER JOIN AUTORES_PRODUTOS
        ON _____)
    RIGHT OUTER JOIN AUTORES AUT
        ON _____
```

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.

-X-

## 6. Prática

### 1. Exercícios de junções (sobre o banco de dados do Estudo de Caso -- Oracle)

Para cada um dos itens abaixo, escreva um comando SELECT para buscar os dados:

1. O título e o nome do autor dos produtos importados, ordenados pelo título.
2. A quantidade de cidades cadastradas no estado de São Paulo.
3. Os nomes, e-mails e números de telefone dos clientes cujos telefones são do código de área (ddd) 51.
4. O código, o nome e a região de cada cidade dos estados do Rio Grande do Sul, São Paulo e Pernambuco cujo nome inicia por "Santa". Ordenar pelo nome da cidade de forma decrescente.
5. Os nomes dos autores (ordenados e sem repetir) cujos produtos já foram vendidos para algum cliente da região norte.



## 2. Scripts de exemplos sobre Produto Cartesiano e Junções

```
-- Criação das tabelas
CREATE TABLE TR
(
    A numeric(2),
    B numeric(2),
    CONSTRAINT PK_TRA PRIMARY KEY(A)
);

CREATE TABLE TS
(
    B numeric(2),
    C numeric(2),
    D numeric(2),
    CONSTRAINT PK_TSB PRIMARY KEY(B)
);
```

```
-- Dados
INSERT INTO TR (A,B) VALUES (1,2);
INSERT INTO TR (A,B) VALUES (3,4);

INSERT INTO TS (B,C,D) VALUES (2,5,6);
INSERT INTO TS (B,C,D) VALUES (4,7,8);
INSERT INTO TS (B,C,D) VALUES (9,10,11);

-- Verificação dos conteúdos
SELECT * FROM TR;
SELECT * FROM TS;
```

```
-- Acrescentar:
INSERT INTO TR (A,B) VALUES (9,NULL);
INSERT INTO TS (B,C,D) VALUES (15, NULL, NULL);
```

```
-- Depois remover:
DELETE FROM TR WHERE A=9;
DELETE FROM TS WHERE B=15;
```

```
-- Produto Cartesiano
SELECT * from TR,TS;
SELECT * from TR,TS WHERE TR.B IS NOT NULL;
```

```
-- Mais elaborado:
SELECT * FROM TR,TS WHERE TR.B = TS.B;
```

```
-- Junções
SELECT * from TR          JOIN TS ON (TR.B = TS.B);
SELECT * from TR    LEFT JOIN TS ON (TR.B = TS.B);
SELECT * from TR    RIGHT JOIN TS ON (TR.B = TS.B);
SELECT * from TR    FULL OUTER JOIN TS ON (TR.B = TS.B);
```

### 3. Exercícios

1. Como se faz para criar uma nova tabela?
2. Quais os tipos de dados que o Oracle suporta?
3. Como se consultam todos os dados de uma tabela?
4. Como se consultam quais tabelas tenho em minha área do Oracle?
5. Como se remove uma tabela do SGBD?
6. O que é uma chave-primária?
7. Como se especifica uma chave-primária em uma tabela?
8. Como se especifica se uma coluna é obrigatória ou opcional?
9. Como se acrescenta uma nova coluna a uma tabela existente?
10. Como se consultam TODOS os campos de uma tabela?
11. Como se consultam APENAS os campos desejados em uma tabela?
12. Como se altera temporariamente o nome de um campo ao se consultar dados de uma tabela?
13. Como se filtram determinados registros dentro de uma consulta?
14. Elabore um resumo da sintaxe mais comum dos comandos SQL vistos até agora.
15. Mostre um exemplo de cada comando sobre o banco de dados do Estudo de Caso.

-X-