5) Consultanto dados em múltiplas tabelas

Cláusula JOIN

A cláusula JOIN é empregada para permitir que um mesmo select recupere informações de mais de uma fonte de dados (tabelas, views, etc.). Em geral, as tabelas referenciadas possuem algum tipo de relacionamento entre elas, através de um ou mais campos que definam a ligação entre uma tabela e a outra (integridade referencial).

Há duas maneiras de implementar um join:

- A primeira é chamada de **non-ANSI** ou estilo ***theta***, que utiliza a cláusula WHERE para efetuar a junção de tabelas;
- A segunda é chamada de **ANSI Join,** e é baseada no uso da cláusula JOIN propriamente dita.

Simples ligação

Um exemplo de JOIN em estilo *ANSI*:

```
SELECT p.uname, p.nome, a.qtde
from PESSOAS p
CROSS JOIN ACESSOS a;
```

Um exemplo de JOIN em estilo *theta*:

```
SELECT p.uname, p.nome, a.qtde
from PESSOAS p, ACESSOS a;
```

Note que na chamada ANSI utilizamos CROSS JOIN, que é a sintaxe utilizada para recuperar todos os registros das tabelas ligadas, formando um produto cartesiano. É basicamente um INNER JOIN (citado adiante) sem condições.

Tipos de junções
Inner Joins

Somente as linhas/registros que satisfaçam a ligação determinada pelo JOIN serão recuperados pelo select, sendo assim, os registros que **não** se enquadram no relacionamento definido pelo join **não serão recuperados**.

Um exemplo de INNER JOIN em estilo ANSI:

```
SELECT p.uname,
p.nome,
a.qtde
from PESSOAS p
INNER JOIN ACESSOS a on p.uname=a.pessoa order by p.uname;
```

O mesmo JOIN em estilo theta:

```
SELECT p.uname,
p.nome,
a.qtde
from PESSOAS p, ACESSOS a WHERE p.uname = a.pessoa order by p.uname;
```

Left Joins
Através do uso do **LEFT**, todos os registros na tabela à esquerda da query serão listados, independente de terem ou não registros relacionados na tabela à direita. Nesse caso, as colunas relacionadas com a tabela da direita voltam nulos (NULL).

Um exemplo de uso LEFT JOIN:

```
SELECT p.uname,
p.nome,
a.pessoa,
a.qtde
from PESSOAS p
LEFT JOIN ACESSOS a on p.uname=a.pessoa order by p.uname;
```

No exemplo acima, todos os registros da tabela PESSOAS serão listados, independente de terem ou não registros associados na tabela ACESSOS. Caso não existam registros associados na tabela ACESSOS, os campos *a.pessoa* e *a.qtde* retornarão NULL.

Right joins
É o inverso do Left Join, ou seja, todos os registros da tabela à direita serão listados, independente de terem ou não registros relacionados na tabela à esquerda.

Um exemplo de uso RIGHT JOIN:

```
SELECT p.uname,
p.nome,
a.pessoa,
a.qtde
from pessoas p
RIGHT JOIN acessos a on p.uname=a.pessoas order by p.uname;
```

Ou seja, todos os registros da tabela ACESSOS serão listados, e caso não haja correspondentes na tabela PESSOAS, a query devolve NULL para os campos p.uname e p.nome.

Matematicamente um Join provêm a operação fundamental em algebra relacional.

Tipos de junção

Junção cruzada

```
T1 CROSS JOIN T2
```

Para cada combinação de linhas de T1 e T2, a tabela derivada contém uma linha formada por todas as colunas de T1 seguidas por todas as colunas de T2. Se as tabelas possuírem N e M linhas, respectivamente, a tabela juntada terá N * M linhas.

FROM T1 CROSS JOIN T2 equivale a FROM T1, T2.

As palavras INNER e OUTER são opcionais em todas as formas. INNER é o padrão; LEFT, RIGHT e FULL implicam em junção externa.

A *condição de junção* é especificada na cláusula ON ou USING, ou implicitamente pela palavra NATURAL. A condição de junção determina quais linhas das duas tabelas de origem são consideradas "correspondentes", conforme explicado detalhadamente abaixo.

Os tipos possíveis de junção qualificada são:

INNER JOIN

Para cada linha L1 de T1, a tabela juntada possui uma linha para cada linha de T2 que satisfaz a condição de junção com L1.

LEFT OUTER JOIN

Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2. Portanto, a tabela juntada possui, incondicionalmente, no mínimo uma linha para cada linha de T1.

RIGHT OUTER JOIN

Primeiro, é realizada uma junção interna. Depois, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1. É o oposto da junção esquerda: a tabela resultante possui, incondicionalmente, uma linha para cada linha de T2.

FULL OUTER JOIN

Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2. Também, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1.

Tipos de junção no PostgreSQL, no SQL Server, no Oracle e no DB2

Tipo de junção	PostgreSQL 8.0.0	SQL Server 2000	Oracle 10g	DB2 8.1
INNER JOIN ON	sim	sim	sim	sim
LEFT OUTER JOIN ON	sim	sim	sim	sim
RIGHT OUTER JOIN ON	sim	sim	sim	sim
FULL OUTER JOIN ON	sim	sim	sim	sim

Tipo de junção	PostgreSQL 8.0.0	SQL Server 2000	Oracle 10g	DB2 8.1
INNER JOIN USING	sim	não	sim	não
CROSS JOIN	sim	sim	sim	não
NATURAL JOIN	sim	não	sim	não

Cláusula JOIN

A cláusula JOIN é empregada para permitir que um mesmo select recupere informações de mais de uma fonte de dados (tabelas, views, etc.). Em geral, as tabelas referenciadas possuem algum tipo de relacionamento entre elas, através de um ou mais campos que definam a ligação entre uma tabela e a outra (integridade referencial).

Há duas maneiras de implementar um join:

- A primeira é chamada de **non-ANSI** ou estilo ***theta***, que utiliza a cláusula WHERE para efetuar a junção de tabelas;
- A segunda é chamada de **ANSI Join,** e é baseada no uso da cláusula JOIN propriamente dita.

Simples ligação

Um exemplo de JOIN em estilo *ANSI*:

```
SELECT p.uname, p.nome, a.qtde
from PESSOAS p
CROSS JOIN ACESSOS a;
```

Um exemplo de JOIN em estilo *theta*:

```
SELECT p.uname, p.nome, a.qtde
from PESSOAS p, ACESSOS a;
```

Note que na chamada ANSI utilizamos CROSS JOIN, que é a sintaxe utilizada para recuperar todos os registros das tabelas ligadas, formando um produto cartesiano. É basicamente um INNER JOIN (citado adiante) sem condições.

Tipos de junções
Inner Joins

Somente as linhas/registros que satisfaçam a ligação determinada pelo JOIN serão recuperados pelo select, sendo assim, os registros que **não** se enquadram no relacionamento definido pelo join **não serão recuperados**.

Um exemplo de INNER JOIN em estilo ANSI:

```
SELECT p.uname,
p.nome,
a.qtde
from PESSOAS p
INNER JOIN ACESSOS a on p.uname=a.pessoa order by p.uname;
```

O mesmo JOIN em estilo theta:

```
SELECT p.uname,
p.nome,
```

```
a.qtde
from PESSOAS p, ACESSOS a WHERE p.uname = a.pessoa order by p.uname;
```

Left Joins
Através do uso do **LEFT**, todos os registros na tabela à esquerda da query serão listados, independente de terem ou não registros relacionados na tabela à direita. Nesse caso, as colunas relacionadas com a tabela da direita voltam nulos (NULL).

Um exemplo de uso LEFT JOIN:

```
SELECT p.uname,
p.nome,
a.pessoa,
a.qtde
from PESSOAS p
LEFT JOIN ACESSOS a on p.uname=a.pessoa order by p.uname;
```

No exemplo acima, todos os registros da tabela PESSOAS serão listados, independente de terem ou não registros associados na tabela ACESSOS. Caso não existam registros associados na tabela ACESSOS, os campos *a.pessoa* e *a.qtde* retornarão NULL.

Right joins
É o inverso do Left Join, ou seja, todos os registros da tabela à direita serão listados, independente de terem ou não registros relacionados na tabela à esquerda.

Um exemplo de uso RIGHT JOIN:

```
SELECT p.uname,
p.nome,
a.pessoa,
a.qtde
from pessoas p
RIGHT JOIN acessos a on p.uname=a.pessoas order by p.uname;
```

Ou seja, todos os registros da tabela ACESSOS serão listados, e caso não haja correspondentes na tabela PESSOAS, a query devolve NULL para os campos p.uname e p.nome.

5.1) Utilizando Apelidos para as tabelas

Quando realizamos consultas em várias tabelas fica menor a consulta se adotarmos apelidos para as tabelas.

Por exemplo: vamos realizar uma consulta que envolve as tabelas: alunos e notas, então podemos usar os apelidos: a e n. Mas isso é algo opcional.

```
\c dml
create table alunos(codaluno int, nome varchar(45));
create table notas(codaluno int, nota1 numeric(4,2));

insert into alunos(codaluno, nome) values (1, 'João Pereira Brito');
insert into alunos(codaluno, nome) values (2, 'Roberto Pereira Brito');
insert into alunos(codaluno, nome) values (3, 'Manoel Pereira Brito');
insert into alunos(codaluno, nome) values (4, 'Pedro Pereira Brito');
insert into alunos(codaluno, nome) values (5, 'Francisco Pereira Brito');
```

```
insert into notas (codaluno, nota1) values (1, 7);
insert into notas (codaluno, nota1) values (2, 5);
insert into notas (codaluno, nota1) values (3, 8);
insert into notas (codaluno, nota1) values (4, 6);
insert into notas (codaluno, nota1) values (5, 9);
```

Quero trazer alunos e notas:

```
SELECT a.nome, n.nota1
FROM alunos a
INNER JOIN notas n ON a.codaluno = n.codaluno order by nota1;
```

5.2) Cruzando dados entre tabelas distintas

```
SELECT a.nome, n.nota1
FROM alunos a, notas n
WHERE a.codaluno = n.codaluno order by nota1;
```

As junções SQL são utilizadas quando precisamos selecionar dados de duas ou mais tabelas.

Existem as junções com estilo non-ANSI ou theta (junção com WHERE sem usar explicitamente a cláusula JOIN)

5.3) Entendendo os Tipos de Join disponíveis

As junções ANSI join (com JOIN explícito).
As junções ANSI podem ser de dois tipos, as INNER e as OUTER, que se subdividem em INNER, OUTER, LEFT e RIGHT.
A padrão é a INNER JOIN. INNER JOIN pode ser escrito com apenas JOIN.

Tipos de Junção Suportados pelo PostgreSQL:

INNER JOIN:
- NATURAL JOIN
- CROSS JOIN

OUTER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

5.4) Trabalhando com CROSS JOIN

Chamado também de cartesiano Join ou produto. Um cross join retorna o produto cartesiano do conjunto de registros das duas tabelas da junção.

Se A e B são dois conjuntos então cross join será AxB.

Cross Join Explícito:

```
SELECT *
FROM funcionarios CROSS JOIN departamentos;
```

Cross Join Implícito:

```
SELECT *
FROM funcionarios, departamentos;
```

Outros Exemplos:

```
SELECT p.maricula, p.senha, d.departamento FROM pessoal p CROSS JOIN departamento d;
```

INNER JOIN - Onde todos os registros que satisfazem à condição serão retornados.

Exemplo:

```
SELECT p.siape, p.nome, l.lotacao
FROM pessoal p INNER JOIN lotacoes l
ON p.siape = l.siape ORDER BY p.siape;
```

Exemplo no estilo theta (non-ANSI):

```
SELECT p.matricula, p.nome, d.departamento
FROM pessoal p, departamento d
WHERE p.matricula = d.matricula ORDER BY p.matricula;
```

OUTER JOIN que se divide em LEFT OUTER JOIN e RIGHT OUTER JOIN

LEFT OUTER JOIN ou simplesmente LEFT JOIN - Somente os registros da tabela da esquerda (left) serão retornados, tendo ou não registros relacionados na tabela da direita.

Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2. Portanto, a tabela juntada possui, incondicionalmente, no mínimo uma linha para cada linha de T1.

A tabela à esquerda do operador de junção exibirá cada um dos seus registros, enquanto que a da direita exibirá somente seus registros que tenham correspondentes aos da tabela da esquerda. Para os registros da direita que não tenham correspondentes na esquerda serão colocados valores NULL.

Exemplo (voltar todos somente de pessoal):

```
SELECT p.matricula, p.nome, d.departamentos
FROM pessoal p LEFT JOIN departamentos d
ON p.siape = d.matricula ORDER BY p.matricula ;
```

Veja que pessoal fica à esquerda em “FROM pessoal p LEFT JOIN departamentos d”.

RIGHT OUTER JOIN

Inverso do LEFT, este retorna todos os registros somente da tabela da direita (right).
Primeiro, é realizada uma junção interna. Depois, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1. É o oposto da junção esquerda: a tabela resultante possui, incondicionalmente, uma linha para cada linha de T2.

Exemplo (retornar somente os registros de lotacoes):

```
SELECT p.matricula, p.nome, d.departamentos
FROM pessoal p RIGHT JOIN departamentos d
ON p.siape = d.matricula ORDER BY p.nome;
```

FULL OUTER JOIN
Primeiro, é realizada uma junção interna. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, é adicionada uma linha juntada com valores nulos nas colunas de T2. Também, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, é adicionada uma linha juntada com valores nulos nas colunas de T1.
E também as:

5.5) Trabalhando com INNER e OUTER JOINs

INNER JOIN

Um exemplo de um inner join !
Vamos supor a seguinte estrutura de tabelas:
Temos as tabelas alunos, notas e frequencias
Alunos Notas Frequencias

CodAluno CodNotas CodFrequencia
Nome CodAluno CodAluno
Endereco Nota1 Freq1
Fone Nota2 Freq2

Para você selecionar vamos supor:

O nome do aluno com a nota 1 e frequencia 1; o SELECT seria assim:

```
SELECT A.Nome, N.Nota1, F.Freq1
FROM Alunos A
INNER JOIN Notas N ON A.CodAluno = N.CodAluno
INNER JOIN Frequencias F ON A.CodAluno = F.CodAluno
```

Isso buscaria de TODOS os Alunos sem excessão, o Nome, Nota1 e Freq1.
Agora se vc quisesse trazer de um determinado aluno, bastaria você acrescentar a seguinte linha:
WHERE A.CodAluno = ????

5.6) Trabalhando com NATURAL JOIN

É uma especialização do Equi-Join e NATURAL é uma forma abreviada de USING. Comparam-se ambas as tabelas do join e o resultado conterá somente uma coluna de cada par de colunas de mesmo nome.

Exemplo:

Tendo como base as duas tabelas:

Employee

LastName	DepartmentID
Rafferty	31
Jones	33
Steinberg	33
Robinson	34
Smith	34
Jasper	36

Department

DepartmentID	DepartmentName
31	Sales
33	Engineering
34	Clerical
35	Marketing

SELECT *
```
FROM employee NATURAL JOIN department;
```

Somente um campo DepartmentID aparece na tabela resultante.

DepartmentID	Employee.LastName	Department.DepartmentName
34	Smith	Clerical
33	Jones	Engineering
34	Robinson	Clerical
33	Steinberg	Engineering
31	Rafferty	Sales

Mais informações em:
http://www.w3schools.com/sql/sql_join.asp
http://en.wikipedia.org/wiki/Join_(SQL)
http://www.postgresql.org/docs/8.2/static/tutorial-join.html
http://www.postgresql.org/docs/current/static/queries-table-expressions.html
http://pgdocptbr.sourceforge.net/pg80/tutorial-join.html
http://pgdocptbr.sourceforge.net/pg80/queries-table-expressions.html

Consultas com JOINS - Thiago Caserta

Obs.: Faça os devidos ajustes, já que este artigo foi feito para o SQL Server.

Quando comecei na área de TI, sempre me deparava com situações no SQL em que precisava pegar campos de outras tabelas não vinculadas diretamente com a que estamos manipulando, e sentia certa dificuldade em amarrar as tabelas pelas suas respectivas chaves, fossem elas código, id, usuário, etc.

Bem, algo que é de extrema ajuda nesses casos são os nossos amigos JOINs (**LEFT JOIN, RIGHT JOIN, INNER JOIN e FULL JOIN**).

Hoje percebo que muitos daqueles que estão começando na área de TI sentem o mesmo.

Junções, tradução de JOINs, são utilizadas em duas cláusulas específicas: **FROM** e **WHERE**, eu particularmente prefiro usar na cláusula **FROM**, por questões de desempenho e organização.

Neste artigo vamos analisar algumas consultas possíveis com essas poderosas cláusulas. Então: Let s JOIN!

A princípio vamos entender o primeiro **JOIN** mencionado, o **LEFT JOIN**.

Como podemos observar, e a própria sintaxe indica, essa cláusula trabalha com os dados da tabela "Esqueda" como sendo os dados principais, ou seja, de acordo com o exemplo abaixo, o **LEFT JOIN** mostrará o que esta na Tabela1 (esquerda), podendo trabalhar também com qualquer outro dado da Tabela2 com a mesma chave encontrada na Tabela1.

Os dados principais que estaremos trabalhando serão os da Tabela1, já que esta, como já mencionado, é a nossa tabela " Esquerda ".

```
SELECT Tab1.* FROM Tabela1 Tab1

LEFT JOIN Tabela2 Tab2 ON Tab1.Cod = Tab2.Cod
```

Podemos fazer a mesma amarração junto à cláusula **WHERE** assim como no exemplo abaixo onde estamos pegando os mesmos valores do exemplo acima.

```
SELECT Tab1.* FROM Tabela1 Tab1, Tabela2 Tab2

WHERE Tab1.Cod *= Tab2.Cod
```

Nesse exemplo os sinais " *= " indicam a condição LEFT JOIN.

Partiremos agora para o irmão mais próximo do LEFT JOIN, o RIGHT JOIN.

O **RIGHT JOIN** retorna o que estiver na Tabela1 e Tabela2 com a mesma chave, e sendo o inverso do **LEFT JOIN** a tabela principal se torna a tabela da " Direita ", ou seja a Tabela2.

```
SELECT Tab2.* FROM @Tabela1 Tab1

RIGHT JOIN @Tabela2 T2 ON Tab1.Cod = Tab2.Cod
```

Do mesmo modo que podemos utilizar o **LEFT JOIN** na cláusula **WHERE** podemos fazer assim também com o **RIGHT JOIN.**

```
SELECT Tab2.* FROM @Tabela1 Tab1, @Tabela2 Tab2

WHERE T1.Cod =* T2.Cod
```

Observe que o sinal no **RIGHT JOIN** é diferente do **LEFT JOIN**, de "Asterisco ="mudamos para"= asterísco ".

Alteramos o asterisco da esquerda para a direita, o que se torna uma ajuda para não confundirmos as cláusulas.

LEFT JOIN - Asterisco à esquerda;

RIGHT JOIN - Asterisco à direita.

Desse ponto partiremos para os dois últimos **JOINS**, o **INNER JOIN** e o **FULL JOIN.**

O **INNER JOIN** nos retorna apenas o que esta na Tabela1 e Tabela2 com a mesma chave.

Exemplo:

```
SELECT * FROM @Tabela1 T1

INNER JOIN @Tabela2 T2 ON T1.Cod = T2.Cod
```

Assim como o **LEFT JOIN** e o **RIGHT JOIN**, podemos da mesma forma fazer essa amarração junto à cláusula **WHERE**, como segue o exemplo:

```
SELECT * from @Tabela T1, @Tabela2 T2

WHERE T1.Cod = T2.Cod
```

Nesse exemplo o sinal " **=** " indica a função **INNER JOIN.**

Já O **FULL JOIN** retorna o que estiver na Tabela1 e Tabela2 levando em conta o seu significado FULL, ou seja, completo. Portanto o **FULL JOIN** retorna tudo o que há nas Tabelas selecionadas

```
SELECT * FROM @Tabela1 T1

FULL JOIN @Tabela2 T2 ON T1.Cod = T2.Cod
```

Bem, essas são as 4 cláusulas mais utilizadas para fazermos amarrações entre tabelas ou pegarmos valores relacionados. Com certeza é de grande ajuda para todos os que fazem uso de banco de dados independentemente de qual seja.

Em anexo está um exemplo de todas as cláusulas tratadas aqui, porém com banco e tabelas reais.

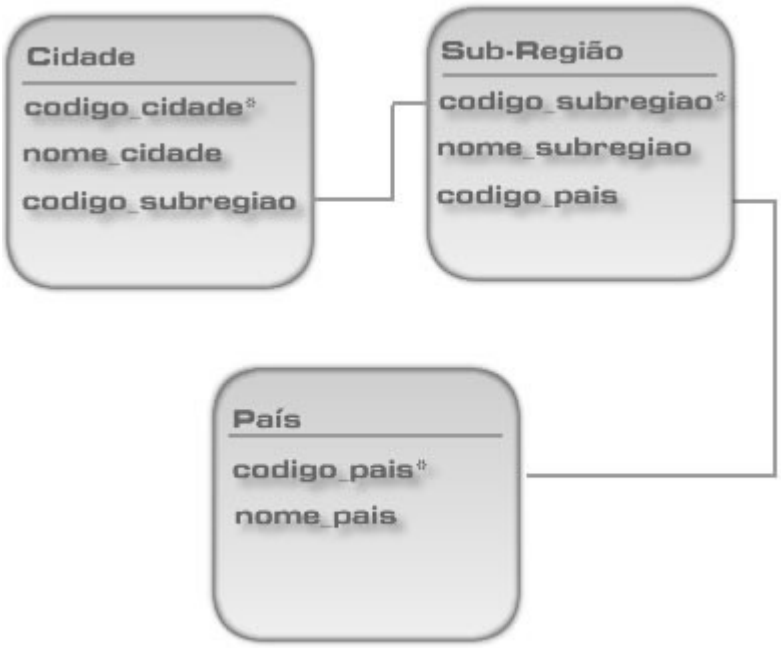
Este exemplo foi baseado no **Microsoft SQL Server**. As sintaxes das consultas não irão mudar, porém para àqueles que forem executar os exemplos em outros bancos terão de modificar os scripts de criação das tabelas.

Espero que esta matéria tenha sido de grande ajuda para àqueles que estão começando ou sentem dificuldades nesse assunto.

Junção entre tabelas no PostgreSQL - Daniel Oslei

A compreensão da real utilidade da junção de tabelas no estudo de banco de dados, e de que forma isto é feito, é um obstáculo para muitos estudantes. A dúvida mais constante a cerca do assunto é com o comando SQL conhecido como JOIN. Já recebi vários e-mails contendo dúvidas relacionadas a utilização correta dos JOINs. Por isso, o objetivo de hoje é esclarecer com uma seqüência de exemplos os tipos de junções de tabelas possíveis no PostgreSQL.

Para os nossos exemplos utilizaremos uma estrutura de três tabelas simples com alguns dados inseridos. O diagrama abaixo representa o relacionamento entre as tabelas:



Vamos partir para o povoamento das tabelas, em que serão inseridos alguns poucos dados, apenas para a efetuação de nossas consultas:

Tabela cidade

codigo	cidade	subregiao
1	Curitiba	1
2	Sao Paulo	2

3	Guarulhos	2
4	Buenos Aires	4
5	La Plata	4
6	Cordoba	5
7	Los Angeles	6
8	San Francisco	6
9	Orlando	7
10	Miami	7
11	Siena	8
12	Florenca	8
13	Milao	9
14	Yokohama	Null

Tabela subregiao

codigo	subregiao	pais
1	Parana	1
2	Sao Paulo	1
3	Rio Grande do Sul	1

4	Buenos Aires	2
5	Cordoba	2
6	California	3
7	Florida	3
8	Toscana	4
9	Lombardia	4
10	Aquitania	5
11	Borgonha	5
12	Calabria	5
13	Massachussetts	3
14	Chiapas	Null

Tabela País

codigo	pais
1	Brasil
2	Argentina
3	Estados Unidos
4	Italia

5	Franca
6	Noruega

Script SQL para criação das tabelas

Tabela cidade

```
CREATE TABLE "public"."cidade" (
"codigo_cidade" SERIAL,
"nome_cidade" VARCHAR(50),
"codigo_subregiao" INTEGER,
CONSTRAINT "cidade_pkey" PRIMARY
KEY("codigo_cidade")
) WITH OIDS;
```

Tabela subregiao

```
CREATE TABLE "public"."subregiao" (
"codigo_subregiao" SERIAL,
"nome_subregiao" VARCHAR(50),
"codigo_pais" INTEGER,
CONSTRAINT "subregiao_pkey" PRIMARY
KEY("codigo_subregiao")
) WITH OIDS;
```

Tabela país

```
CREATE TABLE "public"."pais" (
"codigo_pais" SERIAL,
"nome_pais" VARCHAR(50),
CONSTRAINT "pais_pkey" PRIMARY
KEY("codigo_pais")
) WITH OIDS;
```

Inserção de dados

```
INSERT INTO pais (nome_pais) VALUES ('Brasil');
INSERT INTO pais (nome_pais) VALUES ('Argentina');
INSERT INTO pais (nome_pais) VALUES ('Estados
```

```
Unidos');
INSERT INTO pais (nome_pais) VALUES ('Italia');
INSERT INTO pais (nome_pais) VALUES ('Franca');
INSERT INTO pais (nome_pais) VALUES ('Noruega');

INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Parana', 1);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Sao Paulo', 1);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Rio Grande do Sul', 1);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Buenos Aires', 2);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Cordoba', 2);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'California', 3);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Florida', 3);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Toscana', 4);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Lombardia', 4);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Aquitania', 5);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Borgonha', 5);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Calabria', 5);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Massachussetts', 3);
INSERT INTO subregiao ( nome_subregiao, codigo_pais)
VALUES ( 'Chiapas', NULL);

INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Curitiba', 1);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Sao Paulo', 2);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Guarulhos', 2);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Buenos Aires', 4);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('La Plata', 4);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Cordoba', 5);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
```

```
VALUES ('Los Angeles', 6);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('San Francisco', 6);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Orlando', 7);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Miami', 7);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Siena', 8);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Florenca', 8);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Milao', 9);
INSERT INTO cidade (nome_cidade, codigo_subregiao)
VALUES ('Yokohama', NULL);
```

A junção de tabelas ocasiona uma tabela derivada de outras duas tabelas (reais ou derivadas), de acordo com as regras do tipo de junção. No PostgreSQL as junções são classificadas como sendo qualificadas ou cruzadas.

Junções cruzadas

```
SELECT * FROM Tabela1 CROSS JOIN Tabela2
```

Cada linha de Tabela1 irá combinar-se com todas as linhas de Tabelas2. Para cada combinação de linhas de Tabela1 e Tabela2, a tabela derivada conterá uma linha com todas as colunas de Tabela1 seguidas por todas as colunas de Tabela2. O número de linhas retornadas por esta consulta sempre será o número de linhas de Tabela1 multiplicado pelo número de linha de Tabela2. Por exemplo, se Tabela1 possuir 20 linhas e Tabela2 possuir 10 linhas, será retornado 200 linhas. A consulta SELECT * FROM cidade CROSS JOIN subregiao de nosso exemplo retornará 196 linhas.

É óbvio que destas 196 linhas retornadas a maioria pode ser considerada inútil, portanto, devemos selecionar os nossos dados através de condições para nossa consulta. Essas condições são adicionadas através de cláusula WHERE.

```
SELECT * FROM cidade CROSS JOIN subregiao WHERE cidade.subregiao = subregiao.codigo
```

Como é perceptível, o uso de CROSS JOIN permite a junção de apenas duas tabelas. No entanto, nosso exemplo precisa juntar três tabelas, para isso, teremos que primeiro unir duas tabelas, para que o resultado desta junção seja utilizado com a terceira tabela.

```
SELECT * FROM cidade CROSS JOIN (subregiao CROSS JOIN pais).
```

Utilizar `SELECT * FROM cidade CROSS JOIN subregiao` equivale a utilizar `SELECT * FROM cidade, subregiao`, tanto uma como outra retornará as mesmas 196 linhas e utilizar `SELECT * FROM cidade CROSS JOIN (subregiao CROSS JOIN pais)` equivale a `SELECT * FROM cidade, subregiao, pais`, ambas retornarão as mesmas 1176 linhas.

Junções Qualificadas

As junções qualificadas trazem um pouquinho mais de complexidade e são divididas em junções internas e externas. Na utilização de junção qualificada, se não for especificado como junção interna

ou externa, por padrão o PostgreSQL considera como sendo interna.

Junções internas

A utilização da cláusula INNER é o que caracteriza o comando para uma junção interna, porém, ele não é obrigatório. Pode parecer à primeira vista que as junções internas se equiparam com as junções cruzadas vistas anteriormente, até por que as duas consultas a seguir são equivalentes:

```
SELECT * FROM cidade CROSS JOIN subregiao
```

```
SELECT * FROM cidade INNER JOIN subregiao ON TRUE
```

Mas nas junções internas é sempre obrigatória a especificação de condição de junção, ou seja, quais linhas de uma tabela têm alguma ligação com a linha de outra tabela. Para isso podemos utilizar uma das cláusulas ON ou USING ou utilizar a palavra NATURAL no nosso comando.

A cláusula ON é o mais comumente utilizado por se assemelhar com a cláusula WHERE, ou seja, um par de linhas de Tabela1 e Tabela2 são correspondentes, se a expressão da cláusula ON produz um resultado verdade (*true*) para este par de linhas.

```
SELECT * FROM cidade INNER JOIN subregiao ON
cidade.codigo_subregiao = subregiao.codigo_subregiao
```

codigo_cidade	nome_cidade	codigo_subregiao	codigo_subregiao_1	nome_subregiao	codigo_pais
1	Curitiba	1	1	Parana	1
2	Sao Paulo	2	2	Sao Paulo	1
3	Guarulhos	2	2	Sao Paulo	1
4	Buenos Aires	4	4	Buenos Aires	2
5	La Plata	4	4	Buenos Aires	2
6	Cordoba	5	5	Cordoba	2
8	San Francisco	6	6	California	3
7	Los Angeles	6	6	California	3

9	Orlando	7	7	Florida	3
10	Miami	7	7	Florida	3
11	Siena	8	8	Toscana	4
12	Florenca	8	8	Toscana	4
13	Milao	9	9	Lombardia	4

A cláusula USING traz alguma semelhança com o ON, por também retornar um valor verdadeiro ou falso para aquele conjunto de linhas, no entanto, ele é uma forma mais rápida e abreviada de criação da consulta. Passando um nome de coluna, a execução desta consulta irá procurar nas tabelas a coluna especificada e comparar as duas. Por exemplo, t1 INNER JOIN t2 USING (a, b, c) equivale a t1 INNER JOIN t2 ON (t1.a = t2.a AND t1.b = t2.b AND t1.c = t2.c). Portanto, a consulta anterior equivale à consulta abaixo:

```
SELECT * FROM subregiao INNER JOIN cidade USING
(codigo_subregiao)
```

Para facilitar mais, existe a utilização de NATURAL, que nada mais é abreviação de USING. Com NATURAL, a consulta encontrará todas as colunas que tem nomes iguais nas duas tabelas e fará a comparação de igualdade. O exemplo de USING acima equivale ao seguinte:

```
SELECT * FROM subregiao NATURAL INNER JOIN cidade
```

Mas cuidado com a utilização de NATURAL, pois, ele vai comparar todas as colunas com nomes iguais, o que pode trazer resultados inesperados quando houver duas colunas com o mesmo nome e estas não tenham nenhuma relação.

Junções Externas

Para representar uma junção externa utiliza-se a cláusula OUTER, no entanto, ela não é obrigatória. O que caracteriza realmente as junções externas são as cláusulas LEFT, RIGHT e FULL. As cláusulas ON, USING e NATURAL valem da mesma forma nas junções internas e externas.

LEFT OUTER JOIN

Primeiro, uma junção interna é realizada. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, uma linha juntada é adicionada com valores nulos nas colunas de T2. Portanto, a tabela juntada possui, incondicionalmente, no mínimo uma linha para cada linha de
T1.

```
SELECT * FROM subregiao LEFT OUTER JOIN cidade USING (codigo_subregiao)
```

codigo_subregiao	nome_subregiao	codigo_pais	codigo_cidade	nome_cidade

1	Parana	1	1	Curitiba
2	Sao Paulo	1	2	Sao Paulo
2	Sao Paulo	1	3	Guarulhos
3	Rio G. do Sul	1	Null	Null
4	Buenos Aires	2	4	Buenos Aires
4	Buenos Aires	2	5	La Plata
5	Cordoba	2	6	Cordoba
6	California	3	7	Los Angeles
6	California	3	8	San Francisco
7	Florida	3	9	Orlando
7	Florida	3	10	Miami
8	Toscana	4	11	Siena
8	Toscana	4	12	Florenca
9	Lombardia	4	13	Milao
10	Aquitania	5	Null	Null
11	Borgonha	5	Null	Null
12	Calabria	5	Null	Null

13	Massachussetts	3	Null	Null

Reparem nas linhas destacas acima. As sub-regiões Rio Grande do Sul, Aquitania, Borgonha, Calabria e Massachussets não possuem nenhuma cidade registrada. Em uma consulta normal eles seriam ignorados. Com o uso de LEFT todos as linhas das tabelas da esquerda que não possuem correspondentes na tabela da direita são acrescidas no resultado da consulta.

RIGHT OUTER JOIN

Primeiro, uma junção interna é realizada. Depois, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, uma linha juntada é adicionada com valores nulos nas colunas de T1. É o oposto da junção esquerda: a tabela resultante possui, incondicionalmente, uma linha para cada linha de T2.

SELECT * FROM subregiao RIGHT OUTER JOIN pais USING (codigo_pais)

codigo_pais	codigo_subregiao	nome_subregiao	nome_cidade
1	2	Sao Paulo	Brasil
1	3	Rio Grande do Sul	Brasil
1	1	Parana	Brasil
2	4	Buenos Aires	Argentina
2	5	Cordoba	Argentina
3	13	Massachussetts	Estados Unidos
3	6	California	Estados Unidos
3	7	Florida	Estados Unidos
4	9	Lombardia	Italia

4	8	Toscana	Italia
5	10	Aquitania	Franca
5	11	Borgonha	Franca
5	12	Calabria	Franca
6	Null	Null	Noruega

Basicamente, a diferença entre RIGHT e LEFT está na escolha da tabela em que os elementos que não possuem correspondentes serão escolhidos para ser acrescidos no resultado da consulta. Neste exemplo, Noruega não tem nenhuma sub-região cadastrada, mas mesmo assim ele entra no resultado final.

Continuaremos na próxima matéria, publicada ainda hoje, iniciando com o FULL OUTER JOIN.

Junção entre tabelas no PostgreSQL - Parte 02

Continuaremos falando sobre a compreensão da real utilidade da junção de tabelas no estudo de banco de dados. Para acessar a primeira parte da matéria, publicada hoje também, acesse o link http://www.imasters.com.br/artigo.php?cn=2867&cc=23

FULL OUTER JOIN

Primeiro, uma junção interna é realizada. Depois, para cada linha de T1 que não satisfaz a condição de junção com nenhuma linha de T2, uma linha juntada é adicionada com valores nulos nas colunas de T2. Também, para cada linha de T2 que não satisfaz a condição de junção com nenhuma linha de T1, uma linha juntada com valores nulos nas colunas de T1 é adicionada.

SELECT * FROM subregiao FULL OUTER JOIN cidade USING (codigo_subregiao)

codigo_subregiao	nome_subregiao	codigo_pais	codigo_cidade	nome_cidade
1	Parana	1	1	Curitiba
2	Sao Paulo	1	2	Sao Paulo
2	Sao Paulo	1	3	Guarulhos
3	Rio G. do Sul	1	Null	Null
4	Buenos Aires	2	4	Buenos Aires
4	Buenos Aires	2	5	La Plata
5	Cordoba	2	6	Cordoba
6	California	3	8	San Francisco
6	California	3	7	Los Angeles
7	Florida	3	9	Orlando

7	Florida	3	10	Miami
8	Toscana	4	11	Siena
8	Toscana	4	12	Florenca
9	Lombardia	4	13	Milao
10	Aquitania	5	Null	Null
11	Borgonha	5	Null	Null
12	Calabria	5	Null	Null
13	Massachussetts	3	Null	Null
Null	Null	Null	14	Yokohama

O uso de FULL não é nada mais que a utilização de RIGHT e LEFT juntos. Neste exemplo, foram acrescidos 5 sub-regiões que não posuem nenhum correspondente na tabela de cidade e nesta consulta apareceu a cidade de Yokohama que não possui uma sub-região.

Para buscarmos todos os dados de nossas tabelas utilizando JOIN podemos usar o seguinte comando:

```
SELECT * FROM cidade FULL JOIN (subregiao FULL JOIN pais USING (codigo_pais)) USING (codigo_subregiao)
```

ou

```
SELECT * FROM cidade FULL JOIN subregiao FULL JOIN pais USING (codigo_pais) USING (codigo_subregiao)
```

codigo_subregiao	codigo_cidade	nome_cidade	codigo_pais	nome_subregiao	nome_pais
1	1	Curitiba	1	Parana	Brasil
2	2	Sao Paulo	1	Sao Paulo	Brasil
2	3	Guarulhos	1	Sao Paulo	Brasil

3	Null	Null	1	Rio G. do Sul	Brasil
4	4	Buenos Aires	2	Buenos Aires	Argentina
4	5	La Plata	2	Buenos Aires	Argentina
5	6	Cordoba	2	Cordoba	Argentina
6	8	San Francisco	3	California	Estados Unidos
6	7	Los Angeles	3	California	Estados Unidos
7	9	Orlando	3	Florida	Estados Unidos
7	10	Miami	3	Florida	Estados Unidos
8	11	Siena	4	Toscana	Itália
8	12	Florenca	4	Toscana	Itália
9	13	Milao	4	Lombardia	Itália
Null	14	Yokohama	Null	Null	Null
10	Null	Null	5	Aquitania	Franca
11	Null	Null	5	Borgonha	Franca
12	Null	Null	5	Calabria	Franca
13	Null	Null	3	Massachussets	Estados Unidos

14	Null	Null	Null	Chiapas	Null
Null	Null	Null	6	Null	Noruega

Atenção nas condições da consulta

Quando se usa junção externa deve-se ter muito cuidado com as condições utilizadas na consulta, pois, lembre-se que nestas consultas mesmo que a condição não satisfaça uma linha em comparação com linhas da outra tabela, elas serão retornadas acompanhadas de valores nulos. Vejam um exemplo:

Se acaso quiser saber quais são as cidades registradas como sendo da região de Toscana, as consultas abaixo podem não ser as mais apropriadas:

```
SELECT cidade.descricao, subregiao.descricao FROM cidade LEFT OUTER JOIN subregiao ON cidade.codigo_subregiao = subregiao.codigo_subregiao AND subregiao.descricao = "Toscana"
```

descricao	descricao_1
Curitiba	Null
Sao Paulo	Null
Guarulhos	Null
Buenos Aires	Null
La Plata	Null
Cordoba	Null
San Francisco	Null
Los Angeles	Null
Orlando	Null
Miami	Null
Siena	Toscana

Florenca	Toscana
Milao	Null
Yokohama	Null

SELECT cidade.descricao, subregiao.descricao FROM cidade RIGHT OUTER JOIN subregiao ON cidade.codigo_subregiao = subregiao.codigo_subregiao AND subregiao.descricao = "Toscana"

descricao	descricao_1
Null	Parana
Null	Sao Paulo
Null	Rio Grande do Sul
Null	Buenos Aires
Null	Cordoba
Null	California
Null	Florida
Siena	Toscana
Florenca	Toscana
Null	Lombardia
Null	Aquitania
Null	Borgonha

Null	Calabria
Null	Massachussetts
Null	Chiapas

O mais correto para esta consulta é utilizar

SELECT cidade.descricao, subregiao.descricao FROM cidade INNER JOIN subregiao USING (codigo_subregiao) WHERE subregiao.descricao = "Toscana"

ou

SELECT cidade.descricao, subregiao.descricao FROM cidade INNER JOIN subregiao ON cidade.codigo_subregiao = subregiao.codigo_subregiao AND subregiao.descricao = "Toscana"

que retornarão o mesmo resultado:

nome_cidade	nome_subregiao
Siena	Toscana
Florenca	Toscana

A utilização de JOINs pode parecer complicada, no entanto, ele existe para tornar mais fácil a elaboração das consultas. Espero que tenham compreendido e qualquer dúvida que aparecer pode entrar em contato comigo pelo meu e-mail . Até a próxima semana.

EXEMPLOS DE JOINS SOFISTICADOS

```
SELECT event.name, comment.comment
    FROM event, comment
   WHERE event.id=comment.event_id;

SELECT event.name, comment.comment
    FROM event
    INNER JOIN comment
   ON event.id=comment.event_id;


SELECT event.name, comment.comment
    FROM event
    LEFT JOIN comment ON
```

```
   event.id=comment.event_id;


SELECT event.name, comment.comment
    FROM event
    RIGHT JOIN comment
   ON event.id=comment.event_id;


SELECT event.name, comment.comment
    FROM comment
    RIGHT JOIN event
   ON event.id=comment.event_id;
```

Join

Non-equijoin – função de unir tabelas sem campos em comun.

```
select a.nome, b.codigo
    from cd a, cd, cat b
    where a.preco between b.menor_preco and b.maior_preco;
```

União Regular (inner join ou equi-join)

São os join que tem a cláusula WHERE unindo a PK com a FK das tabelas afetadas.

```
select cd.cod, gravadora.nome
    from cd, gravadora
    where cd.cod_grav = gradora.cod_grav;
```

Sintaxe alternativa (quando a PK e a FK têm o mesmo nome):

```
select cd.cod, gravadora.nome
    from cd natural join gravadora;
```

Apelidos em Tabelas

```
select a.codigo, b.nome
    from cd a, gravadora b
    where a.codigo = b.codigo;
```

Unindo mais de duas Tabelas

```
select a.nome, b.numero, c.nome
```

```
from cd a, faixa b, musica c
where a.codigo in(1,2)
and a.codigo = b.codigo
and b.codigo = c.codigo;
```

Outer Join no PostgreSQL (com SQL padrão):

```
SELECT *
  FROM t1 LEFT OUTER JOIN t2 ON (t1.col = t2.col);
```

ou

```
SELECT *
FROM t1 LEFT OUTER JOIN t2 USING (col);
```

Mais detalhes em:
http://imasters.uol.com.br/artigo/6374/bancodedados/consultas_com_joins/imprimir/
http://www.imasters.com.br/artigo.php?cn=2867&cc=23
http://imasters.uol.com.br/artigo/2870/postgresql/juncao_entre_tabelas_no_postgresql_-_parte_02/