



INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Colatina

BANCO DE DADOS I

Joins
Union

Prof. Gustavo Ludovico Guidoni

Obtendo Dados de Várias Tabelas

cod_fita	nom_fita	cod_genero
210	Os Normais	3
220	Titanic	1
230	Guerra nas Estrelas	4
290	Top Gang	3

cod_genero	dsc_genero
1	Romance
2	Drama
3	Comédia
4	Aventura



cod_fita	nom_fita	cod_genero	dsc_genero
210	Os Normais	3	Comédia
220	Titanic	1	Romance
230	Guerra nas Estrelas	4	Aventura
290	Top Gang	3	Drama

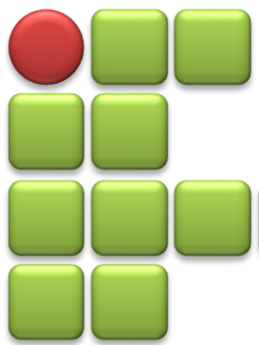


Junção (*Join*)

- É utilizada quando se deseja consultar dados a partir de uma ou mais tabelas

```
SELECT tabela1.coluna, tabela2.coluna
FROM tabela1
[INNER | { {LEFT | RIGHT | FULL} [OUTER]}]
    JOIN tabela2
    ON tabela1.coluna = tabela2.coluna
```

- A palavra-chave **JOIN** especifica quais tabelas serão associadas e como associá-las
- A palavra-chave **ON** especifica as colunas que as tabelas têm em comum



Junção (*Join*)

```
SELECT tabela1.coluna, tabela2.coluna  
FROM   tabela1, tabela2  
WHERE  tabela1.coluna = tabela2.coluna
```

- Na cláusula **FROM** são especificadas quais tabelas serão associadas
- Na cláusula **WHERE** são especificadas as colunas que as tabelas têm em comum e quais as condições de associação



Produto Cartesiano

- Um produto cartesiano é formado quando:
 - Uma condição de junção estiver omitida
 - Uma condição de junção estiver inválida
 - Todas as linhas na primeira tabela estão associadas a todas as linhas da segunda tabela

```
SELECT *  
FROM   FITA, GENERO
```

- Para evitar um produto cartesiano, sempre inclua uma condição de junção válida na cláusula **ON** ou **WHERE**



Tipos de Junções



- Junção interna (**INNER JOIN**)
- Junção externa (**OUTER JOIN**)
- Auto-junção (**SELF JOIN**)
- Junção cruzada (**CROSS JOIN**)

Junção Interna (INNER JOIN)

- Junção típica, que utiliza um operador de comparação (como =, <, >) para comparar colunas de duas tabelas baseado nos valores comuns que as mesmas possuem

cod_fita	nom_fita	cod_genero
210	Os Normais	3
220	Titanic	1
230	Guerra nas Estrelas	4
290	Top Gang	3



Chave estrangeira

cod_genero	dsc_genero
1	Romance
2	Drama
3	Comédia
4	Aventura



Chave primária

Junção Interna (INNER JOIN)

```
SELECT fita.cod_fita, fita.nom_fita,  
       fita.cod_genero, genero.dsc_genero  
FROM   fita  
INNER JOIN genero  
       ON fita.cod_genero = genero.cod_genero
```

OU

```
SELECT fita.cod_fita, fita.nom_fita,  
       fita.cod_genero, genero.dsc_genero  
FROM   fita, genero  
WHERE  fita.cod_genero = genero.cod_genero
```

COD_FITA	NOM_FITA	COD_GENERO	DESCRICAO
210	Os Normais	3	Comédia
220	Titanic	1	Romance
230	Guerra nas Estrelas	4	Aventura
290	Top Gang	3	Comédia



Qualificando Nomes de Colunas Ambíguos

- Use os prefixos de tabela para qualificar nomes de coluna que estiverem em várias tabelas
- Melhore o desempenho usando os prefixos de tabela
- Diferencie colunas que possuem nomes idênticos, mas que estejam em tabelas diferentes, utilizando apelidos de coluna



Utilizando Apelidos para Tabelas

- Simplifique as consultas utilizando apelidos para as tabelas

```
SELECT f.cod_fita  
      , f.nom_fita  
      , f.cod_cor  
      , c.val_fita  
FROM   fita f  
INNER JOIN cor c  
      ON f.cod_cor      = c.cod_cor
```

Incluindo outras Condições de Pesquisa

```
SELECT f.cod_fita
      , f.nom_fita
      , f.cod_genero
      , g.dsc_genero
FROM   fita f
INNER JOIN genero g
      ON f.cod_genero = g.cod_genero
WHERE  f.cod_genero = 3
```

COD_FITA	NOM_FITA	COD_GENERO	DSC_GENERO
210	Os Normais	3	Comédia
290	Top Gang	3	Comédia



Omitindo a palavra-chave INNER

- A palavra-chave **INNER** pode ser omitida, pois a junção interna é o tipo de junção padrão

```
SELECT f.cod_fita
      , f.nom_fita
      , f.cod_genero
      , g.dsc_genero
FROM   fita f
JOIN   genero g
      ON f.cod_genero      = g.cod_genero
WHERE  f.cod_genero      = 3
```

Associando mais de Duas Tabelas

cod_fita	cod_cliente	dat_locacao
250	100	2003-01-20
230	110	2003-03-22
240	110	2003-10-20

cod_fita	nom_fita
230	Guerra nas Estrelas
240	À espera de um milagre
250	O sexto sentido

cod_cliente	nom_cliente
100	João da Silva
110	Maria José de Souza
120	Antônio Carlos Ferreira



Associando mais de Duas Tabelas

```
SELECT f.cod_fita, f.nom_fita, c.cod_cliente,  
       c.nom_cliente, l.dat_locacao  
FROM   locacao l  
JOIN   fita f  
       ON l.cod_fita = f.cod_fita  
JOIN   cliente c  
       ON l.cod_cliente = c.cod_cliente
```

OU

```
SELECT f.cod_fita, f.nom_fita, c.cod_cliente,  
       c.nome, l.data  
FROM   locacao l, fita f, cliente c  
WHERE  l.cod_fita = f.cod_fita  
AND    l.cod_cliente = c.cod_cliente
```



Junção Externa (*Outer Join*)

- Utilizada para retornar linhas que em geral não atendem às condições de junção

```
SELECT tabela1.coluna, tabela2.coluna  
FROM tabela1  
{LEFT | RIGHT | FULL} OUTER JOIN tabela2  
    ON tabela1.coluna = tabela2.coluna
```

Junção Externa (*Outer Join*)

- Qual a quantidade de empregados por departamento? Departamento que não possuem empregados devem ser exibidos

cod_empregado	nom_empregado	cod_departamento
7369	SMITH	20
7499	ALLEN	30
7521	WARD	30
7566	JONES	20
7698	BLAKE	30
7782	CLARK	10
7788	SCOTT	20
7839	KING	10
7844	TURNER	30

cod_departame nto	nom_departame nto
10	CONTABILIDADE
20	PESQUISA
30	VENDA
40	OPERACIONAL

Nenhum funcionário no departamento **OPERACIONAL**



Usando OUTER JOIN

```
SELECT d.nom_departamento, e.nom_empregado
FROM   departamento d
LEFT OUTER JOIN empregado e
      ON d.cod_departamento = e.cod_departamento
```

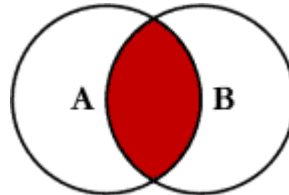
OU

```
SELECT d.nom_departamento, e.nom_empregado
FROM   empregado e
RIGHT OUTER JOIN departamento d
      ON d.cod_departamento = e.cod_departamento
```

A palavra
OUTER é
opcional

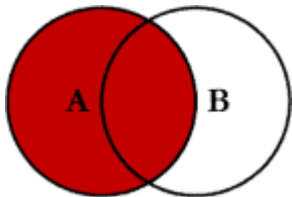
INNER, LEFT, RIGHT, CROSS E FULL JOINS

INNER JOIN



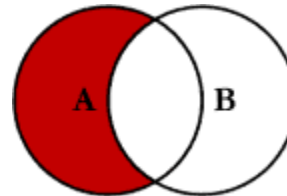
```
SELECT *
FROM A
INNER JOIN B
ON A.chave = B.chave
```

LEFT JOIN



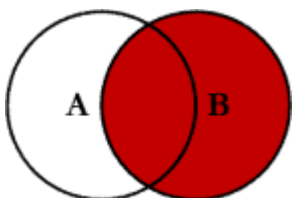
```
SELECT *
FROM A
LEFT JOIN B
ON A.chave = B.chave
```

LEFT Excluindo JOIN



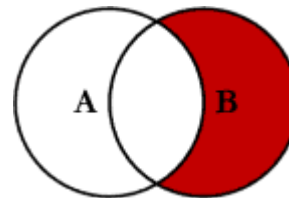
```
SELECT *
FROM A
LEFT JOIN B
ON A.chave = B.chave
WHERE B.chave IS NULL
```

RIGHT JOIN



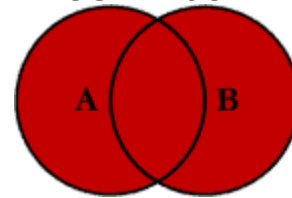
```
SELECT *
FROM A
RIGHT JOIN B
ON A.chave = B.chave
```

RIGHT Excluindo JOIN



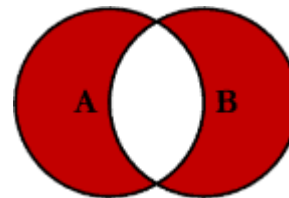
```
SELECT *
FROM A
RIGHT JOIN B
ON A.chave = B.chave
WHERE A.chave IS NULL
```

OUTER JOIN



```
SELECT *
FROM A
FULL OUTER JOIN B
ON A.chave = B.chave
```

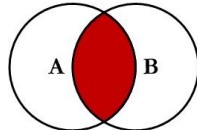
OUTER Excluindo JOIN



```
SELECT *
FROM A
FULL OUTER JOIN B
ON A.chave = B.chave
WHERE A.chave IS NULL OR B.chave
IS NULL
```

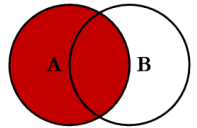
JOINS

Inner Join



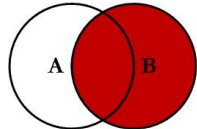
c, d

Left Join



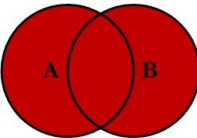
a, b, c, d

Right Join



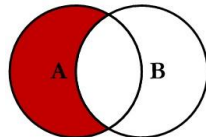
e, f, c, d

Outer Join



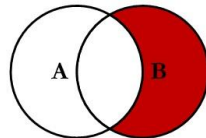
a, b, c, d, e, f

Left Excluding Join



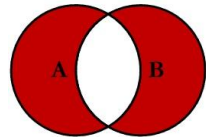
a, b

Right Excluding Join



e, f

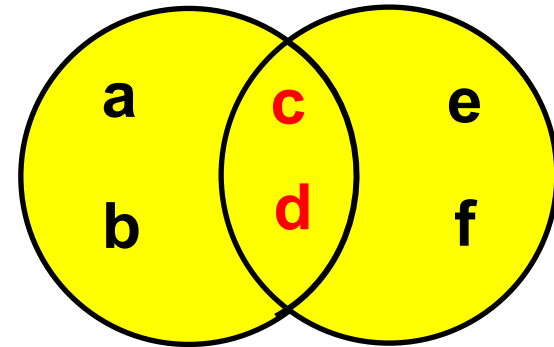
Outer Excluding Join



a, b, e, f

TabelaA

TabelaB



```

Drop table TabelaB;
CREATE TABLE TabelaA
(
  chave varchar(5) not null,
);
CREATE TABLE TabelaB
(
  chave varchar(5) not null,
);
INSERT INTO TabelaA VALUES('a');
INSERT INTO TabelaA VALUES('b');
INSERT INTO TabelaA VALUES('c');
INSERT INTO TabelaA VALUES('d');
INSERT INTO TabelaA VALUES('e');
INSERT INTO TabelaA VALUES('f');
INSERT INTO TabelaB VALUES('c');
INSERT INTO TabelaB VALUES('d');
INSERT INTO TabelaB VALUES('e');
INSERT INTO TabelaB VALUES('f');

--Inner join- Retorna os registros que são comuns às duas tabelas (Implicito e Explicito)
--Explicito
SELECT a.chave, b.chave
FROM TabelaA as A
INNER JOIN TabelaB as B on a.chave = b.chave
--Implicito
SELECT a.chave, b.chave
FROM TabelaA as A, TabelaB as B where a.chave = b.chave

--Left join- Retorna os registros que estão em A (inclusive que não estejam em B) e os registros de B que são comuns à A
SELECT a.chave, b.chave
FROM TabelaA as A
LEFT JOIN TabelaB as B on a.chave = b.chave
--Opcional TabelaB
SELECT a.chave, b.chave
FROM TabelaA as A,
LEFT OUTER JOIN TabelaB as B on a.chave = b.chave

--Right join- Retorna os registros que estão em B (inclusive que não estejam em A) e os registros de A que são comuns à B
SELECT a.chave, b.chave
FROM TabelaA as A
RIGHT JOIN TabelaB as B on a.chave = b.chave
--Opcional
SELECT a.chave, b.chave
FROM TabelaA as A,
RIGHT OUTER JOIN TabelaB as B on a.chave = b.chave

--Outer join- Retorna os registros de A e B, também conhecido por Full Outer join ou Full join
SELECT a.chave, b.chave
FROM TabelaA as A
FULL OUTER JOIN TabelaB as B on a.chave = b.chave

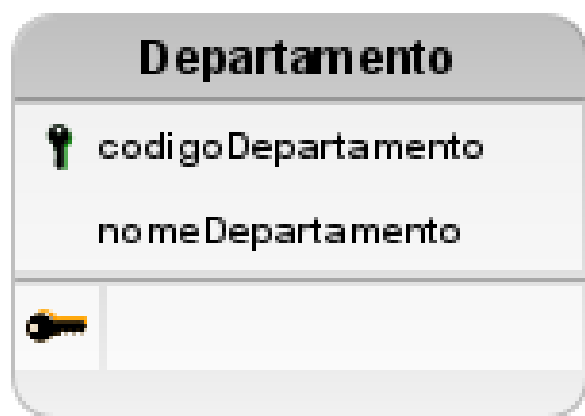
--Left Excluding join- Retorna os registros que estão em A e que não estejam em B
SELECT a.chave, b.chave
FROM TabelaA as A
LEFT JOIN TabelaB as B on a.chave = b.chave
WHERE b.chave is null

--Right Excluding join- Retorna os registros que estão em B e que não estejam em A
SELECT a.chave, b.chave
FROM TabelaA as A
RIGHT JOIN TabelaB as B on a.chave = b.chave
WHERE a.chave is null

--Outer Excluding join- Retorna os registros que estão em A, e que não estejam em B, e os registros que estão em B, e que não estejam em A
SELECT a.chave, b.chave
FROM TabelaA as A
FULL OUTER JOIN TabelaB as B on a.chave = b.chave
WHERE a.chave is not null or b.chave is not null

--CROSS JOIN- Retorna todas as linhas das tabelas por cruzamento, ou seja, para cada linha de tabela A quarentos todas as linhas de B
SELECT
FROM TabelaA
CROSS JOIN TabelaB
SELECT
FROM TabelaA, TabelaB
    
```

Exemplo



(0,1)

(0,n)



```

drop table Professor;
drop table Departamento;

create table Departamento
(
  codigoDepartamento serial primary key,
  nomeDepartamento varchar(255) not null
);

create table Professor
(
  codigoProfessor serial primary key,
  nomeProfessor varchar(255) not null,
  codigodepartamento integer references Departamento(codigoDepartamento)
);

insert into Departamento (nomeDepartamento) values ('Informática');
insert into Departamento (nomeDepartamento) values ('Química');
insert into Departamento (nomeDepartamento) values ('Física');
insert into Departamento (nomeDepartamento) values ('Matemática');
insert into Departamento (nomeDepartamento) values ('Língua');
insert into Departamento (nomeDepartamento) values ('Português');

insert into Professor (nomeProfessor, codigodepartamento) values ('Carlos Pedro', 1);
insert into Professor (nomeProfessor, codigodepartamento) values ('João Pedro Meirel', 2);
insert into Professor (nomeProfessor, codigodepartamento) values ('Rafael Diego', 3);
insert into Professor (nomeProfessor, codigodepartamento) values ('Fábio Luiz', 4);
insert into Professor (nomeProfessor, codigodepartamento) values ('Rita de Cassia', null);
insert into Professor (nomeProfessor, codigodepartamento) values ('João Roberto', null);
insert into Professor (nomeProfessor, codigodepartamento) values ('Dora Delgadete', null);

select 'sem professor'
where 'sem departamento';

--Join Join - Retorna os registros que são comuns de duas tabelas (implícito e explícito)
--Explícito
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
INNER JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento
--Implícito
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A, Professor as B where a.codigoDepartamento = b.codigodepartamento

--Left Join - Retorna os registros que estão em A (mesmo que não estejam em B) e os registros de B que não estejam em A
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
LEFT JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento
--Or
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
LEFT OUTER JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento

--Right Join - Retorna os registros que estão em B (mesmo que não estejam em A) e os registros de A que não estejam em B
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
RIGHT JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento

--Full Join - Retorna os registros que estão em A e B, também conhecido por Full Outer Join ou Full Join
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
FULL OUTER JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento

--Left Excluding Join - Retorna os registros que estão em A e que não estejam em B
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
LEFT JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento
WHERE b.codigodepartamento is null

--Right Excluding Join - Retorna os registros que estão em B e que não estejam em A
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
RIGHT JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento
WHERE a.codigoDepartamento is null

--Outer Excluding Join - Retorna os registros que estão em B, que não estejam em A, e os registros de A, que não estejam em B
SELECT a.nomeDepartamento, b.nomeProfessor
FROM Departamento as A
FULL OUTER JOIN Professor as B on a.codigoDepartamento = b.codigodepartamento
WHERE a.codigoDepartamento is null or b.codigodepartamento is null

--Cross Join - Retorna todos os dados das tabelas por cruzamento, ou seja, para cada linha de tabela A, apresenta todas as linhas de B
SELECT *
FROM Departamento
CROSS JOIN Professor

Select 'sem Departamento, Professor'

```



Autojunção (*Self Join*)

- Utilizada para associar uma tabela a ela mesma

```
SELECT tit.cod_cliente 'cod tit', tit.nom_cliente 'nom tit',  
       dep.cod_cliente 'cod dep', dep.nom_cliente 'nom dep'  
FROM   cliente tit  
JOIN   cliente dep  
ON     tit.cod_cliente = dep.cod_cliente_titular
```

OU

```
SELECT tit.cod_cliente 'cod tit', tit.nom_cliente 'nom tit',  
       dep.cod_cliente 'cod dep', dep.nom_cliente 'nom dep'  
FROM   cliente dep, cliente tit  
WHERE  tit.cod_cliente = dep.cod_cliente_titular
```



Autojunção (*Self Join*)

- Resultado da query anterior

COD TIT	NOM TIT	COD DEP	NOM DEP
110	Maria José de Souza	210	Jussara de Souza
110	Maria José de Souza	220	Geraldo José de Souza
120	Antônio Carlos Ferr	230	Bruno Garzon Ferreira
120	Antônio Carlos Ferr	240	Carla Oliveira Ferrei



Junção Cruzada (*Cross Join*)

- Utilizada para gerar um produto cartesiano entre uma ou mais tabelas

```
SELECT tabela1.coluna, tabela2.coluna  
FROM tabela1 CROSS JOIN tabela2
```

OU

```
SELECT tabela1.coluna, tabela2.coluna  
FROM tabela1, tabela2
```



Junção Cruzada (*Cross Join*)

```
SELECT f.cod_fita, p.cod_cor, p.dsc_cor, p.val_fita  
FROM fita f CROSS JOIN cor p
```

OU

```
SELECT f.cod_fita, p.cod_cor, p.dsc_cor, p.val_fita  
FROM fita f, cor p
```

cod_fita	cod_cor	dsc_cor	val_fita
-----	-----	-----	-----
200	VM	AM	4.0
210	AM	AM	4.0
220	VD	AM	4.0
230	BR	AM	4.0
240	PR	AM	4.0
...

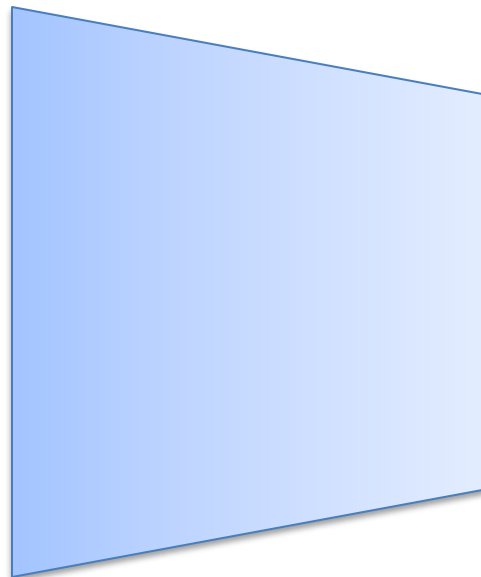
Comando UNION

- Realiza a união de dois resultados
 - Une o resultado de dois **selects**
 - O **tipo** do dado exibido em cada coluna deve ser o mesmo para todos os **selects**

col1	col2
AM	4
AZ	2
BR	2

UNION

col5	col6
TESTE	4
TST	2



col1	col2
AM	4
AZ	2
BR	2
TESTE	4
TST	2



Comando UNION



- Exiba todos os clientes cujos nomes iniciam com a letra F e nunca alugaram fitas e os clientes que alugaram mais de 2 fitas.

Comando UNION

```
select  nom_cliente, 'Nunca alugou fita' as tipo
from    cliente c
where   c.nom_cliente    like 'F%'
and     not exists (      select  1
                           from    locacao l
                           where   c.cod_cliente    = l.cod_cliente
                           )

union

select  nom_cliente, 'Alugou mais de 2 fitas'
from    cliente c
join    locacao l
        on      c.cod_cliente    = l.cod_cliente
group by nom_cliente
having count(*) > 2
order by 1
```



Comando UNION



nom_cliente	tipo
-----	-----
Daniela marinho	Alugou mais de 2 fitas
Fabírcia passos	Alugou mais de 2 fitas
Flavio Medeiros dos Santos	Nunca alugou fita
Francisco de Assis Silva	Nunca alugou fita
Maria josé de souza	Alugou mais de 2 fitas

