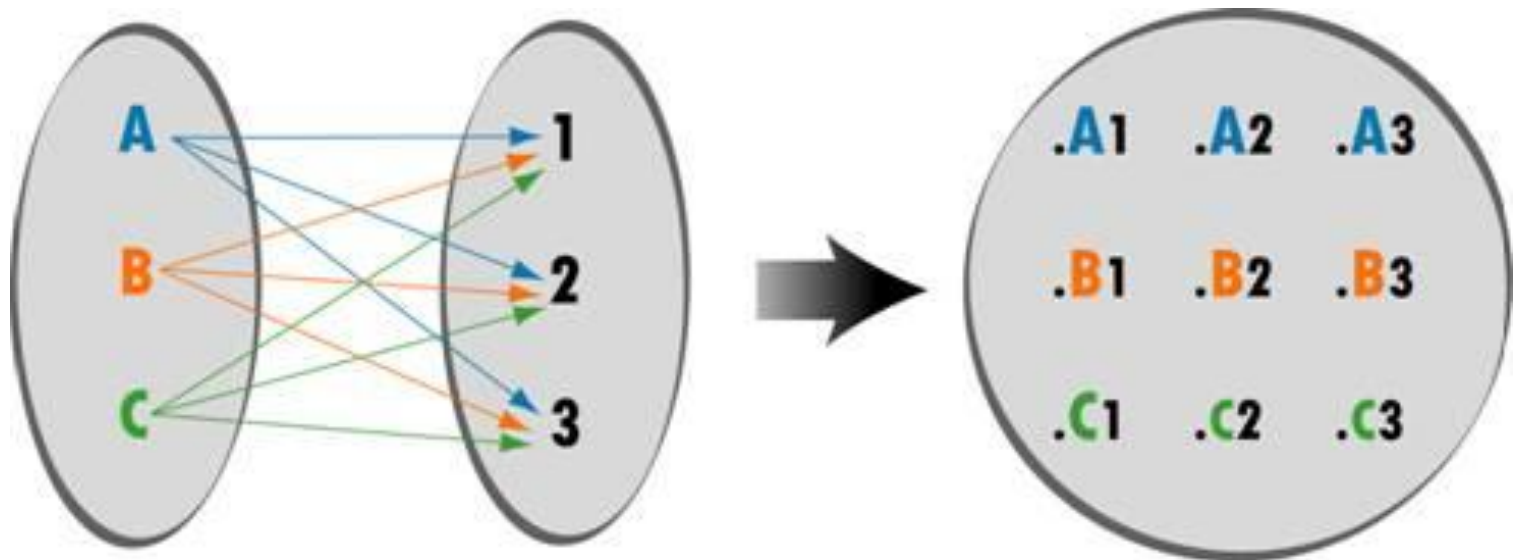


✓ Cláusula FROM

- Deve se informar qual(is) tabela(s) são necessárias para se realizar a consulta.
- **É um produto cartesiano**

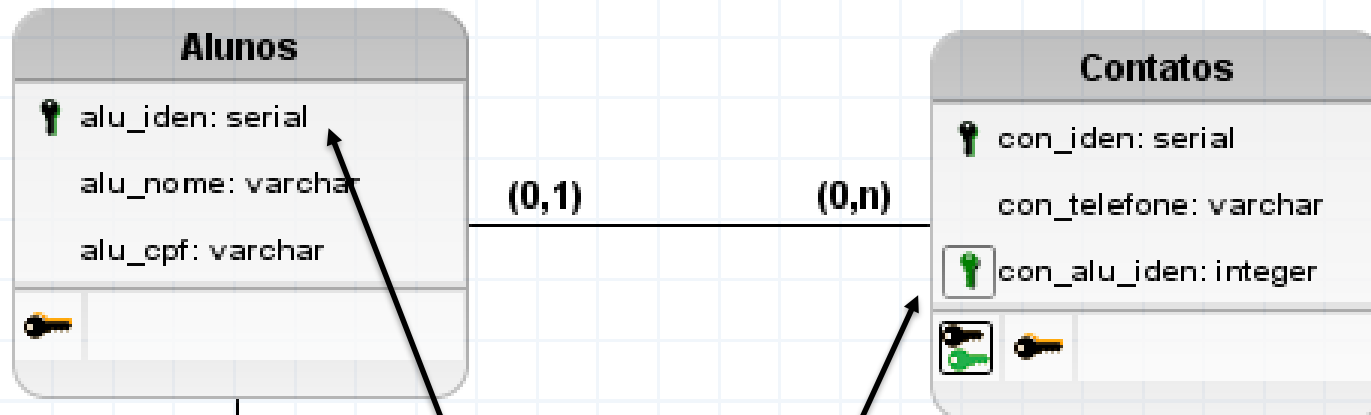


✓ Cláusula FROM

- Note que SQL usa a notação ***nome_relação.nome_atributo*** para evitar ambigüidades.
- Usou no from mais de uma tabela, deve-se realizar essa comparação de chave estrangeira com chave primária da outra tabela.

```
SELECT atributos  
FROM nome_tabela1, nome_tabela2  
WHERE chave_primaria_tab1 = chave_estrangeira_tab2
```

SQL – DML

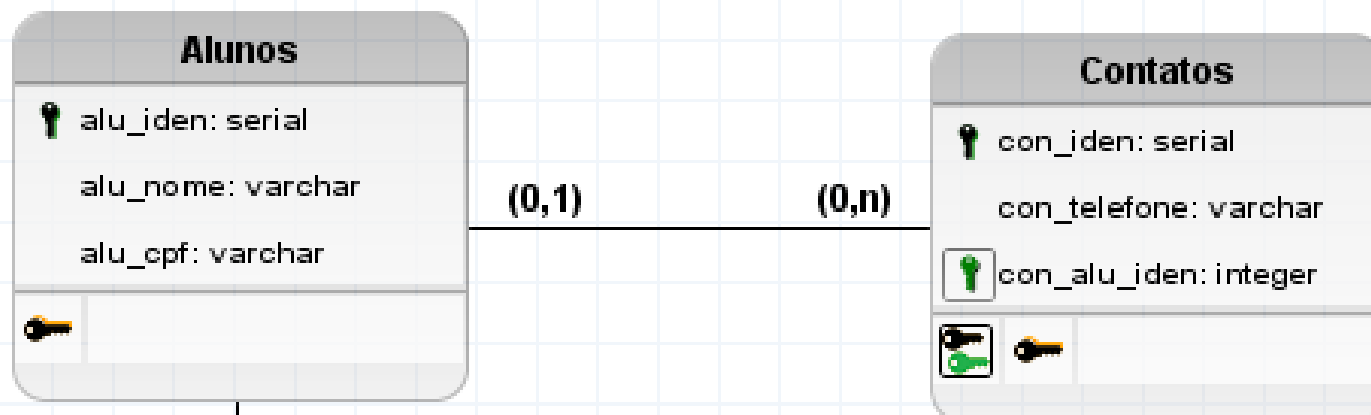


```
1  SELECT alu_nome, contatos.*
2  FROM alunos, contatos
3  WHERE alunos.alu_iden = contatos.con_alu_iden
```

[Data Output](#) [Explain](#) [Messages](#) [Notifications](#)

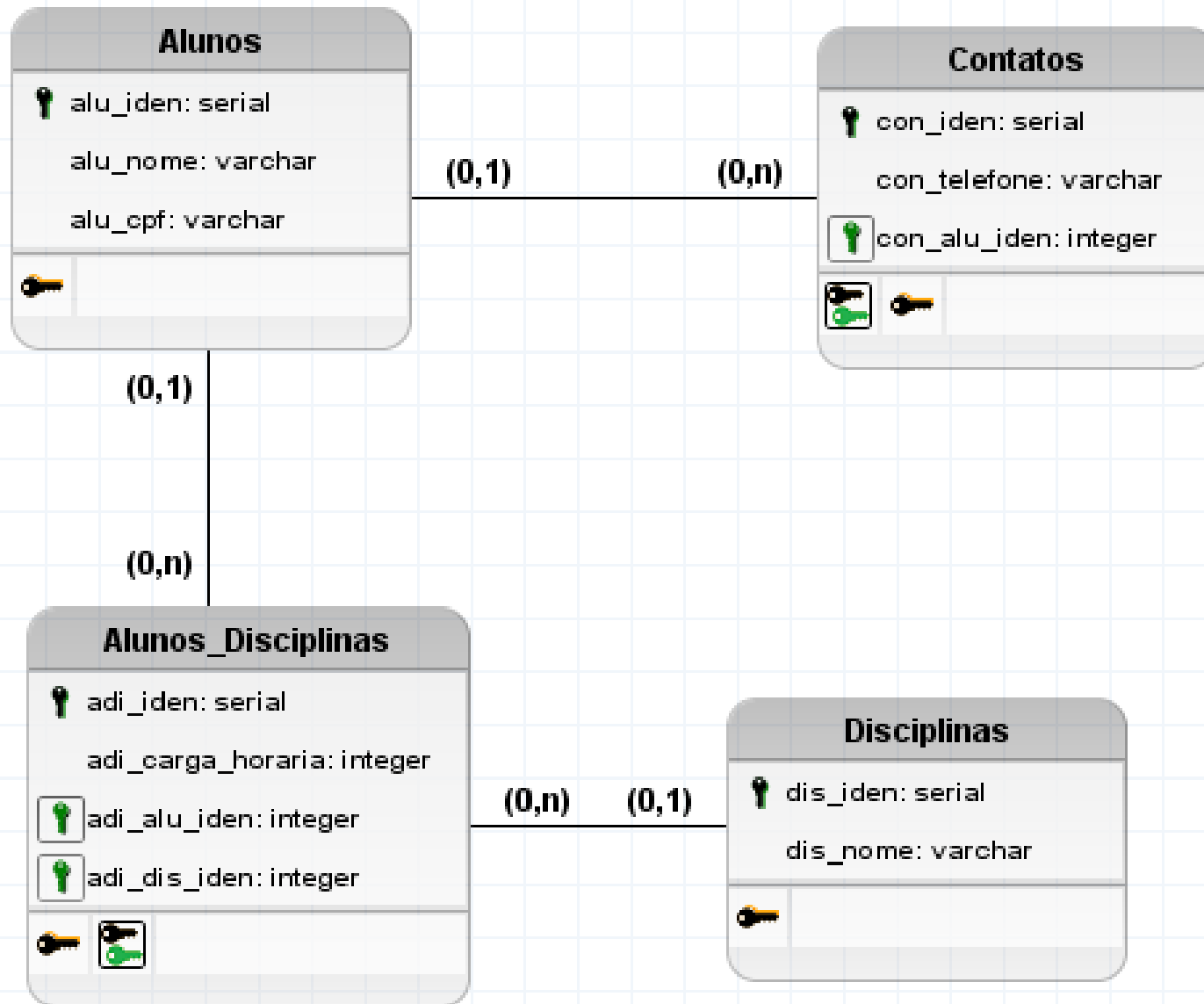
	alu_nome character varying	con_iden integer	con_telefone character varying	con_alu_iden integer
1	MARIA JOAQUINA	7	5555-5555	1
2	MARIA JOAQUINA	8	5444-5555	1
3	RAMBO	9	5333-5555	3

SQL – DML



```
1  /**JUNÇÃO SEM COMANDO JOIN*/
2  SELECT *
3  FROM alunos a, contatos c
4  WHERE a.alu_iden = c.con_alu_iden
5
6  /**USANDO JOIN*/
7  SELECT *
8  FROM alunos a JOIN contatos c
9       ON a.alu_iden = c.con_alu_iden
10
```

SQL – DML



SQL – DML

```
1  /*JUNÇÃO NA UNHA*/
2  SELECT *
3  FROM alunos a, contatos c, alunos_disciplinas ad, disciplinas d
4  WHERE a.alu_iden = c.com_alu_iden AND
5         a.alu_iden = ad.adi_alu_iden AND
6         d.dis_iden = ad.adi_dis_iden
7
8  /*USO DO JOIN*/
9  SELECT *
10 FROM alunos a JOIN contatos c ON c.com_alu_iden = a.alu_iden
11      JOIN alunos_disciplinas ad ON a.alu_iden = ad.adi_alu_iden
12      JOIN disciplinas d ON d.dis_iden = ad.adi_dis_iden
13
```

SQL – DML

`SELECT * FROM a
INNER JOIN b ON a.key = b.key`



`SELECT * FROM a
LEFT JOIN b ON a.key = b.key`



`SELECT * FROM a
RIGHT JOIN b ON a.key = b.key`



POSTGRES JOINS



`SELECT * FROM a
LEFT JOIN b ON a.key = b.key
WHERE b.key IS NULL`



`SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
WHERE a.key IS NULL`



`SELECT * FROM a
FULL JOIN b ON a.key = b.key`



`SELECT * FROM a
FULL JOIN b ON a.key = b.key
WHERE a.key IS NULL OR b.key IS NULL`



SQL – Operação Rename

- ✓ Proporciona um mecanismo para rebatizar tanto relações quanto atributos, usando a cláusula **AS**.

Nome_antigo **AS** nome_novo

- ✓ Pode estar tanto cláusula **SELECT** como **FROM**

SQL – Operação Rename

Query Editor Query History

```
1  /*COM USO DO 'AS'*/
2  SELECT usu_nome as nome, usu_renda as renda
3  FROM usuarios
4
5  /*SEM USO DO 'AS'*/
6  SELECT usu_nome nome, usu_renda renda
7  FROM usuarios
8
9
10
```

Data Output Explain Messages Notifications

	nome	renda
	character varying (80)	numeric (7,2)
1	MARIA	8455.66
2	JOAO	1000.50

SQL – Variáveis Tuplas

- ✓ Uma variável tupla em SQL precisa estar associada a uma relação em particular (palavra chave **AS** opcional)

```
1  /*APELIDO PARA TABELA, OPCIONAL USAR 'AS'*/  
2  SELECT u.usu_nome, c.con_telefone  
3  FROM usuarios u, contatos c  
4  WHERE u.usu_iden = c.con_usu_iden
```

The diagram illustrates the association of tuple variables with tables in an SQL query. Red arrows point from the table names 'usuarios' and 'contatos' in the FROM clause to the tuple variables 'u' and 'c' respectively. Another set of red arrows points from 'u.usu_iden' and 'c.con_usu_iden' in the WHERE clause back to the tuple variables 'u' and 'c', showing the join condition.

SQL – Operações em Strings

- ✓ Operações sobre Strings mais usadas são as verificações de coincidências de pares
 - Operador **LIKE**
 - **(%)**: Compara qualquer substring
 - **(_)** : Compara qualquer caracter
 - Sensíveis(Maiúsculas / Minúsculas)
 - SQL <> sql

SQL – Operações em Strings

```
1  /*TENHA A EM QUALQUER PARTE */
2  SELECT * FROM usuarios u
3  WHERE u.usu_nome LIKE '%A%'
4
5  /*COMECE COM J */
6  SELECT * FROM usuarios u
7  WHERE u.usu_nome LIKE 'J%'
8
9  /*TERMINE COM IA */
10 SELECT * FROM usuarios u
11 WHERE u.usu_nome LIKE '%IA'
12
13 /*COMECE COM QUALQUER CARACTERE
14 TENHA O SEGUNDO COMO 'A' E
15 QUALQUER FINAL */
16 SELECT * FROM usuarios u
17 WHERE u.usu_nome LIKE '_A%'
```

Funções e operadores disponíveis para examinar e manipular valores cadeia de caracteres

Veja mais...

<https://www.postgresql.org/docs/11/functions-string.html>

SQL – Ordenação e apresentação de Tuplas


- ✓ SQL oferece ao usuário algum controle sobre a ordenação.

ORDER BY nome_atributo


- ✓ A cláusula ORDER BY relaciona os itens em ordem ascendente. Para especificar a forma de ordenação, devemos indicar:
 - ☐ **DESC** (Ordem Descendente)
 - ☐ **ASC** (Ordem Ascendente)

SQL – Ordenação e apresentação de Tuplas

```
1  /*ORDENAR NOME DE MÉDICO
2  ORDEM CRESCENTE - ASC É OPCIONAL*/
3  SELECT *
4  FROM medicos m
5  ORDER BY m.med_nome ASC
6
7  /*ORDENAR NOME DE MÉDICO
8  ORDEM DECRESCENTE*/
9  SELECT *
10 FROM medicos m
11 ORDER BY m.med_nome DESC
```



1	Aarão Câmara	[null]	
2	Aarão Ribeiro	[null]	
3	Abílio Camacho	[null]	
4	Abílio Novalles	[null]	



	med_nome character varying	med_crm character varyir
1	Zuriel Fragoso	[null]
2	Zuriel Carrasqueira	[null]
3	Zuleide Robalinho	[null]

SQL – Modificações no BD

✓ **REMOÇÃO**

Podemos remover somente tuplas inteiras; mas não podemos excluir valores de uma célula em particular

```
DELETE FROM r  
WHERE P
```



SQL – Modificações no BD

✓ REMOÇÃO



```
1  /*REMOVER O MÉDICO ALCIDES VILLANUEVA*/  
2  DELETE FROM medicos m  
3  WHERE m.med_nome = 'Alcides Villanueva'  
4
```

```
1  /*REMOVER FILMES ENTRE 2015 E 2019  
2  QUE O TÍTULO COMECAM COM A*/  
3  DELETE FROM filmes f  
4  WHERE f.fil_ano BETWEEN 2015 AND 2019  
5  AND f.fil_titulo LIKE 'A%'
```


SQL – Modificações no BD

✓ **Não confunda!**

✓ **DROP TABLE** r

✓ ***DELETE FROM** r

✓ ***TRUNCATE FROM** r

Delete mantém a tabela r, mas remove todas as suas tuplas. **Drop** não remove apenas todas as tuplas de r, mas também seu esquema.