

SQL

Comandos

Avançados



UNIQUE

- Define que uma coluna, ou um grupo de colunas, vai ser único na tabela.

```
CREATE TABLE funcionario(  
    codigo INTEGER PRIMARY KEY,  
    cpf VARCHAR(11) UNIQUE,  
    telefone VARCHAR(11),  
    email VARCHAR(70)  
);
```

UNIQUE

- Define que uma coluna, ou um grupo de colunas, vai ser único na tabela.

```
CREATE TABLE consulta(  
    codigo_consulta INTEGER PRIMARY KEY,  
    cpf_paciente VARCHAR(11),  
    crm_medico VARCHAR(6),  
    data_consulta date,  
    UNIQUE(cpf_paciente, crm_medico, data_consulta),  
    FOREIGN KEY (cpf_paciente) REFERENCES paciente (cpf),  
    FOREIGN KEY (crm_medico) REFERENCES medico (crm)  
);
```

Adicionando uma CONSTRAINT

- Constraint são todas as restrições que uma coluna pode ter: PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, etc.
- Quando preciso adicionar um CONSTRAINT em uma tabela já criada

```
ALTER TABLE funcionario  
ADD CONSTRAINT telefone_unico UNIQUE (telefone);
```

Removendo uma CONSTRAINT

- Quando é necessário remover uma restrição de uma coluna

```
ALTER TABLE funcionario  
DROP CONSTRAINT telefone_unico;
```

Adicionando uma CONSTRAINT

- Constraint são todas as restrições que uma coluna pode ter: PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, etc.
- Quando preciso adicionar um CONSTRAINT em uma tabela já criada

```
ALTER TABLE funcionario  
ALTER COLUMN email SET NOT NULL;
```

Removendo uma CONSTRAINT

- Quando é necessário remover uma restrição de uma coluna

```
ALTER TABLE funcionario  
ALTER COLUMN email DROP NOT NULL;
```

Modificando o valor DEFAULT

```
ALTER TABLE funcionario  
ALTER COLUMN email SET DEFAULT '--';
```


Removendo o valor DEFAULT

```
ALTER TABLE funcionario  
ALTER COLUMN email DROP DEFAULT;
```

Modificando o tipo de uma coluna

- Quando é necessário mudar o tipo de dados de uma determinada coluna.

```
ALTER TABLE funcionario  
ALTER COLUMN email TYPE VARCHAR(200);
```

Modificando o tipo de uma coluna

- CUIDADO: se já existir dados na tabela que não correspondem ao novo tipo, não será possível
- Outro ponto é que se a coluna modificada for chave estrangeira em outra tabela, é preciso primeiro mudar na tabela que possui chave estrangeira

Renomeando uma tabela

```
ALTER TABLE funcionario RENAME TO professor;
```

Criando tabelas em SCHEMAS

```
CREATE SCHEMA fapam;
```

```
CREATE TABLE fapam.aluno(  
    cpf VARCHAR(11),  
    nome VARCHAR(80)  
);
```

Retornando dados alterados

- Em alguns casos é necessário retornar dados que foram alterados (inseridos, atualizados, deletados)

Query

Query History

1

INSERT INTO professor

2

(codigo, cpf, telefone, email)

3

VALUES

4

(1, '11111111111', '37999999999', 'teste@fapam.edu.br')

5

RETURNING *;

Data Output

Messages

Notifications

≡+

▼

	<div><div>codigo</div><div>[PK] integer </div></div>	<div><div>cpf</div><div>character varying (11) </div></div>	<div><div>telefone</div><div>character varying (11) </div></div>	<div><div>email</div><div>character varying (200) </div></div>
1	1	11111111111	37999999999	teste@fapam.edu.br

Retornando dados alterados

Query

Query History

1

DELETE FROM professor RETURNING codigo;

Data Output

Messages

Notifications

codigo

[PK] integer

1

1

FULL OUTER JOIN

- Faz o RIGHT e o LEFT JOIN ao mesmo tempo, ou seja, vai retornar dados que não tem em comum nas duas tabelas, e não só na tabela da direita ou da esquerda

A	M
1	m
2	n
4	o

table_A

**SELECT * FROM table_A
FULL OUTER JOIN table_B
ON table_A.A=table_B.A;**

A	N
2	p
3	q
5	r

table_B

A	M	A	N
2	n	2	p
1	m	-	-
4	o	-	-
-	-	3	q
-	-	5	r

Output

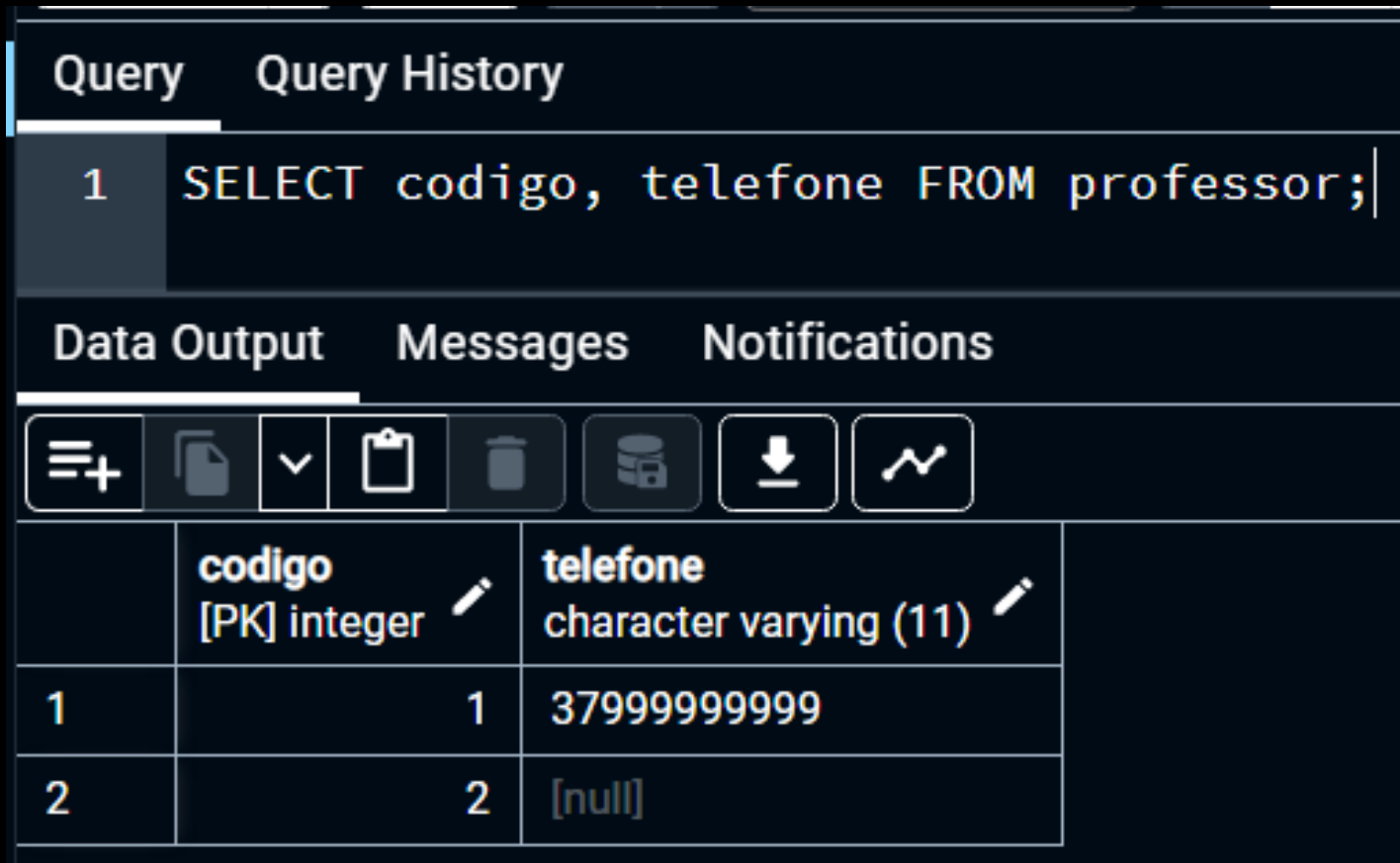
Limitando o número de linhas

- Em alguns casos, eu posso querer retornar um número específico de linhas, por exemplo, selecionar 5 alunos em uma tabela que possui 40 alunos.

```
SELECT *  
FROM aluno  
LIMIT 5;
```

Substituindo o valor Nulo

- Em alguns casos, podemos querer retornar um valor padrão, para caso alguma coluna tenha o valor nulo.



The screenshot shows a database query tool interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT codigo, telefone FROM professor;`. Below the query, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table of results. The table has two columns: 'codigo' (integer, primary key) and 'telefone' (character varying (11)). The first row shows 'codigo' 1 and 'telefone' 37999999999. The second row shows 'codigo' 2 and 'telefone' [null].

	codigo [PK] integer	telefone character varying (11)
1	1	37999999999
2	2	[null]

Substituindo o valor Nulo

- Em alguns casos, podemos querer retornar um valor padrão, para caso alguma coluna tenha o valor nulo.

Query

Query History

1

SELECT codigo, COALESCE(telefone, 'NAO CADASTRADO')









2



FROM professor;

Data Output

Messages

Notifications



	codigo [PK] integer 	coalesce character varying 
1	1	379999999999
2	2	NAO CADASTRADO

Pular as N primeiras linhas

Query

Query History

1

SELECT * FROM professor

2

ORDER BY codigo

3

OFFSET 1;

Data Output

Messages

Notifications

	<div>codigo</div> <div>[PK] integer</div>	<div>cpf</div> <div>character varying (11)</div>	<div>telefone</div> <div>character varying (11)</div>	<div>email</div> <div>character varying (200)</div>
1	2	22222222222	[null]	dois@fapam.edu.br

Selecionar um valor diferente

- Quando precisamos selecionar valores diferentes de algo

```
SELECT * FROM professor  
WHERE codigo != 1;
```

```
SELECT * FROM professor  
WHERE codigo <> 1;
```

Comentário

```
--Buscar professores  
SELECT * FROM professor;
```

Resto da divisão

Query

Query History

1

SELECT 10 % 3;

Data Output

Messages

Potência

Query		Query History	
1	SELECT 2 ^ 3;		
Data Output		Messages	
		<div><div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>	
	<div><div>?column?</div><div>double precision</div><div></div></div>		
1			8

Comando LIKE e NOT LIKE

- Quando desejamos pesquisar uma substring de um texto, utilizamos o comando LIKE ou o NOT LIKE. Utilizamos o % no início ou no final da string para representar QUALQUER CHARACTER.

Query

Query History

1

SELECT * FROM professor

2

WHERE email LIKE '%@fapam%';

3

Data Output

Messages

Notifications

≡+

	<div>codigo</div> <div>[PK] integer </div>	<div>cpf</div> <div>character varying (11) </div>	<div>telefone</div> <div>character varying (11) </div>	<div>email</div> <div>character varying (200) </div>
1	1	11111111111	37999999999	teste@fapam.edu.br
2	2	22222222222	[null]	dois@fapam.edu.br

CASE

- Utilizado para quando desejamos colocar uma condição no nosso SELECT.

```
CREATE TABLE cliente(  
    codigo INTEGER PRIMARY KEY,  
    tipo_cliente VARCHAR(2),  
    cpfcnpj VARCHAR(14)  
);
```

CASE

Query

Query History

1

SELECT * FROM cliente

Data Output

Messages

Notifications

	<div>codigo</div> <div>[PK] integer</div>	<div>tipo_cliente</div> <div>character varying (2)</div>	<div>cpfcnpj</div> <div>character varying (14)</div>
1	1	PF	1111111111
2	2	PJ	22222222222222

CASE

Query

Query History

1

SELECT cpfcnpj, CASE WHEN tipo_cliente = 'PF' THEN 'Pessoa Física'

2

WHEN tipo_cliente = 'PJ' THEN 'Pessoa Jurídica'

3

ELSE 'Cadastro inválido'

4

END

5

FROM cliente

Data Output

Messages

Notifications

</

Funções Matemáticas

- `pi()` – valor aproximado de π
- `factorial()` – valor do fatorial de n
- `log(n)` - valor logarítmico do número n
- `round(n)` – arredondar o valor para inteiro
- `trunc(n)` – ignora a parte decimal

Funções de String

- `character_length(texto)` – retorna número de caracteres
- `lower(texto)` – coloca em minúsculo
- `upper(texto)` – coloca em maiúsculo
- `position(substring IN texto)` – retorna a posição da uma substring dentro de um texto
- `Substring(substring for/from n)` – pega uma parte da string (começa no caracter 1)
 - From – caracter inicial
 - For – quantos caracteres pegar
 - `substring('Gabriel' from 3 for 3)` - bri

Concatenar Strings

Query

Query History

1

SELECT concat(tipo_cliente, '-', cpfcpnpj)

2

FROM cliente

Data Output

Messages

Notifications

concat

text

1

PF-11111111111

2

PJ-22222222222222

Conversão de Dados

Function
Description
Example(s)
<code>to_char(timestamp, text) → text</code> <code>to_char(timestamp with time zone, text) → text</code> Converts time stamp to string according to the given format. <code>to_char(timestamp '2002-04-20 17:31:12.66', 'HH12:MI:SS') → 05:31:12</code>
<code>to_char(interval, text) → text</code> Converts interval to string according to the given format. <code>to_char(interval '15h 2m 12s', 'HH24:MI:SS') → 15:02:12</code>
<code>to_char(numeric_type, text) → text</code> Converts number to string according to the given format; available for integer, bigint, numeric, real, double precision. <code>to_char(125, '999') → 125</code> <code>to_char(125.8::real, '999D9') → 125.8</code> <code>to_char(-125.8, '999D99S') → 125.80-</code>
<code>to_date(text, text) → date</code> Converts string to date according to the given format. <code>to_date('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05</code>
<code>to_number(text, text) → numeric</code> Converts string to numeric according to the given format. <code>to_number('12,454.8-', '99G999D9S') → -12454.8</code>
<code>to_timestamp(text, text) → timestamp with time zone</code> Converts string to time stamp according to the given format. (See also <code>to_timestamp(double precision)</code> in Table 9.33 .) <code>to_timestamp('05 Dec 2000', 'DD Mon YYYY') → 2000-12-05 00:00:00-05</code>

Conversão de Dados

Query	Query History	
1	SELECT '3.12' / 1;	
Data Output	Messages	Notifications
ERROR: invalid input syntax for type integer: "3.12" LINE 1: SELECT '3.12' / 1; ^ SQL state: 22P02 Character: 8		

Conversão de Dados

Query

Query History

1

SELECT CAST('3.12' AS NUMERIC) / 1;

Data Output

Messages

Notifications

?column?

numeric

1

3.1200000000000000

Conversão de Dados

Query

Query History

1

SELECT '3.12'::NUMERIC / 1;

Data Output

Messages

Notifications

?column?

numeric

1

3.120000000000000000

Funções de data/hora

- `age(timestamp, timestamp)` – calcular a idade
- <https://www.postgresql.org/docs/15/functions-datetime.html>