

# Banco de dados

## Linguagem SQL

### DQL – Data Query Language

Wagner Cesar Vieira  
wagner.vieira@sp.senai.br



# Agenda

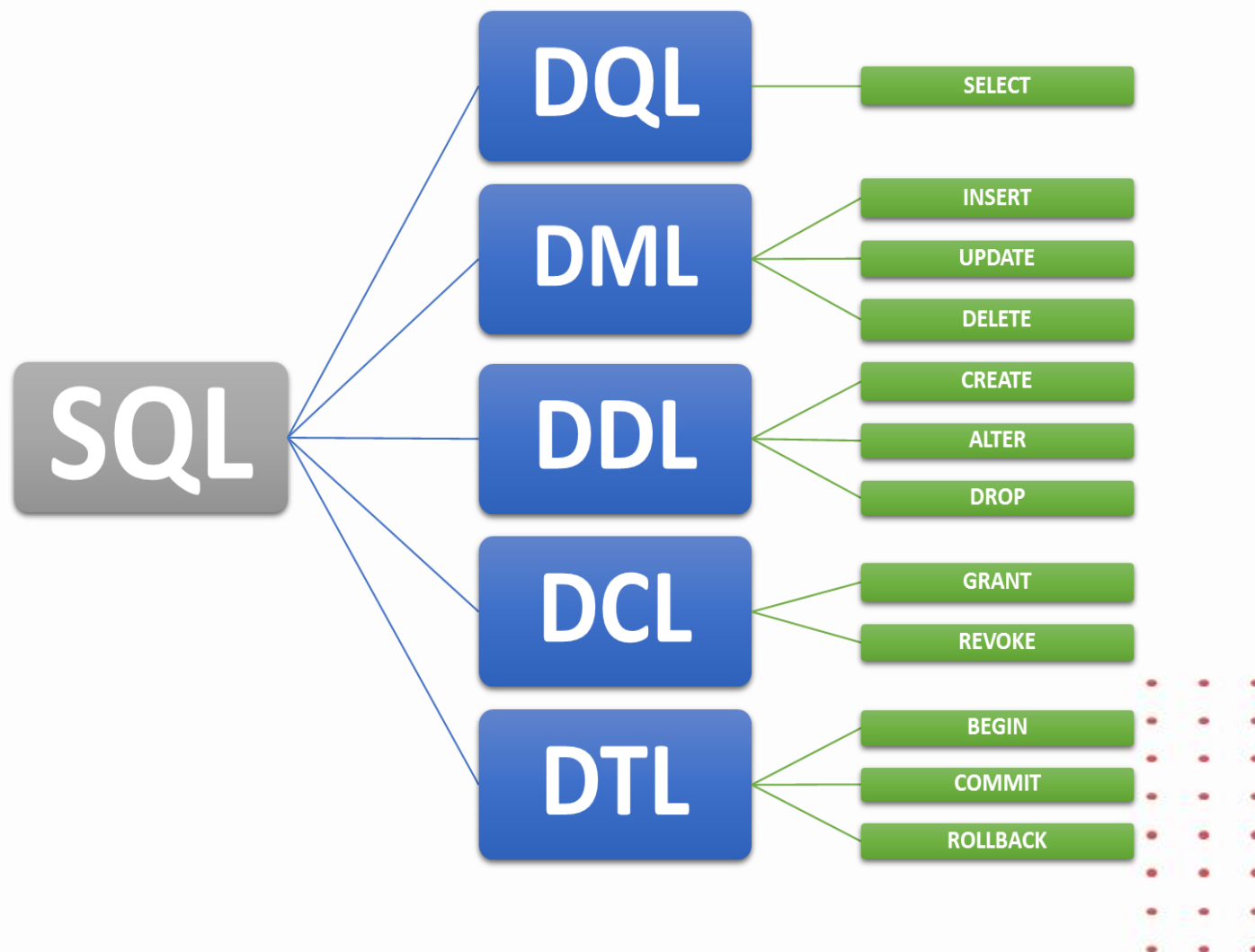


- ❑ Introdução
- ❑ Instrução **Select**
  - ❑ Cláusula Where
  - ❑ Cláusula Like
  - ❑ Operadores comparativos
  - ❑ Cláusula Between
  - ❑ Cláusula Order By
  - ❑ Lista de exercícios



# Introdução

A linguagem SQL é o recurso mais conhecido por DBAs e programadores para a execução de comandos em bancos de dados relacionais. É por meio dela que criamos tabelas, colunas, índices, atribuímos permissões a usuários, bem como realizamos consultas a dados. Enfim, é utilizando a SQL que “conversamos” com o banco de dados.



# Comandos DQL: Select



O comando select é bastante simples e, serve para você, mostrar os dados da tabela. Sua sintaxe pode ser utilizado de algumas formas, como:

**SELECT \* FROM** NOME\_DA\_TABELA;

Exemplo: **SELECT \* FROM** alunos;

**Nota 1:** o asterisco indica que você deseja mostrar todos os campos, ou seja, id, nome, email e cidade (conforme a tabela do MySQL que você deseja exibir os registros).

Exemplo: **SELECT** cpf, nome, telefone **FROM** alunos;

**Nota 2:** neste caso, você está indicando que deseja mostrar somente nome e e-mail. Isso é interessante, pois a consulta é executada de forma mais rápida, já que você está trazendo menos informações da tabela.



# Operadores comparativos



Os operadores de comparação são usados em condições que comparam uma expressão a outro valor ou expressão. A tabela abaixo mostra os operadores:

= Igual a  
> Maior que  
>= Maior ou igual a que  
< Menor que  
<= Menor ou igual a que  
<> Diferente de

## Cláusula Where e seus Operadores no SQL



```
SELECT codigo_empregado, nome, salario  
      FROM empregados  
      WHERE codigo_empregado = 8
```



# Operadores lógicos

O uso de um operador lógico faz com que duas condições tenham de produzir um resultado único. Uma linha só poderá ser retornada se o resultado global da condição for verdadeiro.

- **AND** Retorna TRUE se ambas as condições forem verdadeiras
- **OR** Retorna TRUE se uma das condições for verdadeira
- **NOT** Retorna TRUE se a condição seguinte for falsa

**SELECT** nome, salário, cidade, estado **FROM** empregados  
**WHERE** estado = 'SP' **AND** salario > 2200

OPERADOR	EXEMPLO
AND	CONDIÇÃO 1 AND CONDIÇÃO 2
OR	CONDIÇÃO 1 OR CONDIÇÃO 2
NOT	NOT CONDIÇÃO

# Select – cláusula WHERE



A cláusula Where permite ao comando SQL passar condições de filtragem. Exemplos:

❑ Selecione a cidade cuja descrição seja igual a 'Bebedouro'

```
SELECT * FROM cidade WHERE cidade_descricao = 'Bebedouro';
```

❑ Selecione todas as cidades cuja cidade\_código seja maior que 10.000

```
SELECT * FROM cidade WHERE cidade_codigo >= 10000;
```

❑ Selecione todos os bairros cuja cidade\_codigo seja menor ou igual a 26

```
SELECT * FROM bairro WHERE cidade_codigo <= 26;
```

❑ Selecione os campos numcode e nome da tabela de países

```
SELECT numcode, nome FROM paises;
```



# Cláusula WHERE – AND e OR

**WHERE** com **AND** e **OR** – Usamos os operadores **AND** e **OR** junto com Where quando é necessário usar mais de uma condição de comparação.

- ❑ O operador **AND** é usado quando desejamos atender as duas condições distintas.

```
SELECT endereco_codigo, endereco_cep, endereco_logradouro  
FROM endereco
```

```
WHERE endereco_cep >= '90000000' OR endereco_codigo <= 10;
```

- ❑ O operador **OR** é usado especificamente quando desejamos atender uma ou outra condição.

```
SELECT * FROM alunos
```

```
WHERE nome = 'Fulano de Tal' OR nome = 'Beltrano'
```





# Select – LIKE e NOT LIKE



Quando realizamos uma consulta SQL, utilizamos a cláusula **WHERE** para realizar um filtro dos registros a retornar.

Porém, com o **WHERE**, só podemos aplicar filtros de correspondência exata de palavras. E se precisarmos aplicar um filtro que verifique palavras de forma parcial, como palavras que iniciem ou terminem com determinados caracteres, ou que possuam sequências de caracteres específicas? Neste caso, usamos a cláusula **LIKE**:

```
SELECT cidade_descricao, cidade_cep FROM cidade  
WHERE cidade_descricao LIKE '%Be%';
```



# Select – LIKE e NOT LIKE



A cláusula **LIKE** determina se uma cadeia de caracteres (string) corresponde a um padrão especificado. Um padrão pode incluir caracteres normais e curingas.

```
SELECT * FROM paises
WHERE iso3 LIKE 'B%'
OR iso3 LIKE 'F%';
```

	iso	iso3	numcode	nome
1	BA	BIH	70	Bósnia-Herzegovi...
2	BB	BRB	52	Barbados
3	BD	BGD	50	Bangladesh
4	BE	BEL	56	Bélgica
5	BF	BFA	854	Burkina Faso
6	BG	BGR	100	Bulgária
7	BH	BHR	48	Bahrain
8	BI	BDI	108	Burundi
9	BJ	BEN	204	Benin
10	BM	BMU	60	Bermuda
11	BN	BRN	96	Brunei
12	BO	BOL	68	Bolívia
13	BR	BRA	76	Brasil
14	BS	BHS	44	Bahamas
15	BT	BTN	64	Butão
16	BV	BVT	74	Bouvet, Ilha
17	BW	BWA	72	Botswana
18	BY	BLR	112	Bielo-Rússia
19	BZ	BLZ	84	Belize
20	FI	FIN	246	Finlândia
21	FJ	FJI	242	Fiji
22	FK	FLK	238	Malvinas, Ilhas (F...
23	FM	FSM	583	Micronésia, Estad...
24	FO	FRO	234	Faroe, Ilhas
25	FR	FRA	250	França



## Select – LIKE e NOT LIKE



A cláusula **NOT LIKE** inverte a comparação, verificando se a cadeia de caracteres NÃO corresponde ao padrão especificado.

```
SELECT * FROM bairro  
WHERE bairro_descricao LIKE 'VILA%';
```

→ Retorna 4197 rows (linhas / registros)

```
SELECT * FROM bairro  
WHERE bairro_descricao NOT LIKE 'VILA%';
```

→ Retorna 23954 rows (linhas / registros)



# Select – WHERE com IN



Podemos usar o operador **IN** no lugar do operador **OR** em determinadas situações. O **IN** permite verificar se o valor de uma coluna está presente em uma lista de elementos.

-- **WHERE** com **IN**

```
SELECT endereco_codigo, endereco_logradouro FROM endereco  
WHERE endereco_codigo IN (10, 15, 20, 25, 30, 35, 40);
```



# Select – WHERE com NOT IN



Este operador, ao contrário do **IN**, permite obter como resultado o valor de uma coluna que não pertence a uma determinada lista de elementos.

## -- WHERE com AND

```
SELECT endereco_codigo, endereco_logradouro FROM endereco  
WHERE endereco_codigo IN (10, 15, 20, 25, 30, 35, 40);
```

## -- WHERE com NOT IN

```
SELECT endereco_codigo, endereco_logradouro FROM endereco  
WHERE endereco_codigo NOT IN (10, 15, 20, 25, 30, 35, 40);
```



# WHERE com BETWEEN



O operador BETWEEN tem a finalidade de permitir a consulta entre uma determinada faixa de valores.

-- WHERE com AND

```
SELECT uf_codigo, cidade_descricao FROM cidade  
WHERE cidade_codigo >=20 AND cidade_codigo <=1000;
```

-- WHERE com BETWEEN

```
SELECT uf_codigo, cidade_descricao FROM cidade  
WHERE cidade_codigo BETWEEN 20 AND 1000;
```



# WHERE com NOT BETWEEN



O operador NOT BETWEEN, ao contrário do anteriormente descrito, permite consultar os valores que não se encontram em uma determinada faixa.

-- **WHERE** com **BETWEEN**

```
SELECT * FROM uf  
WHERE uf_codigo NOT BETWEEN 1 AND 15;
```

```
SELECT endereco_codigo, endereco_cep, endereco_logradouro  
FROM endereco  
WHERE endereco_cep NOT BETWEEN '10000000' AND '90000000';
```



# ORDER BY



**ORDER BY** organiza os resultados de acordo com uma ou mais colunas da tabela, podendo definir a ordem dos resultados como crescente ou decrescente.

**Caso a ordem não seja declarada, será crescente (ASC), por padrão.**

**Exemplo:**

```
SELECT * FROM uf
WHERE uf_codigo NOT BETWEEN 1 AND 15
ORDER BY uf_codigo DESC;
```


```
SELECT * FROM cidade
WHERE cidade_descricao LIKE 'A%'
ORDER BY cidade_cep;
```





# Fonte de dados – download da base






 **cep** Public

Pin Unwatch 1 Fork 0 Star 0


main 1 Branch 0 Tags

Go to file t Add file <> Code

 **wcrvieira** Adicionando modelo base d321d03 · yesterday 2 Commits

 cep.bak	Backup - modelo CEP	yesterday
 modelo_cep.png	Adicionando modelo base	yesterday

README



## Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Fonte: <https://github.com/wcrvieira/cep>

# Criando o banco cep vazio



Solution1 - Microsoft SQL Server Management Studio

File Edit View Project Tools Window Help

Object Explorer

Connect

NB-SARAIVA (SQL Server 15.0.2104.1 - sa)

Data

New Database...

Attach...

Restore Database...

Restore Files and Filegroups...

Filter

Deploy Data-tier Application...

Import Data-tier Application...

Start PowerShell

Reports

Refresh

PolyBase

Always On High Availability

Management

Integration Services Catalogs

SQL Server Agent (Agent XPs disabled)

XEvent Profiler

New Database

Select a page

General

Options

Filegroups

Database name: cep

Owner: <default>

☒ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize	Path
cep	ROWS...	PRIMARY	8	By 64 MB, Unlimited	C:\...
cep_log	LOG	Not Applicable	8	By 64 MB, Unlimited	C:\...

Connection

Server: NB-SARAIVA

Connection: sa

[View connection properties](#)

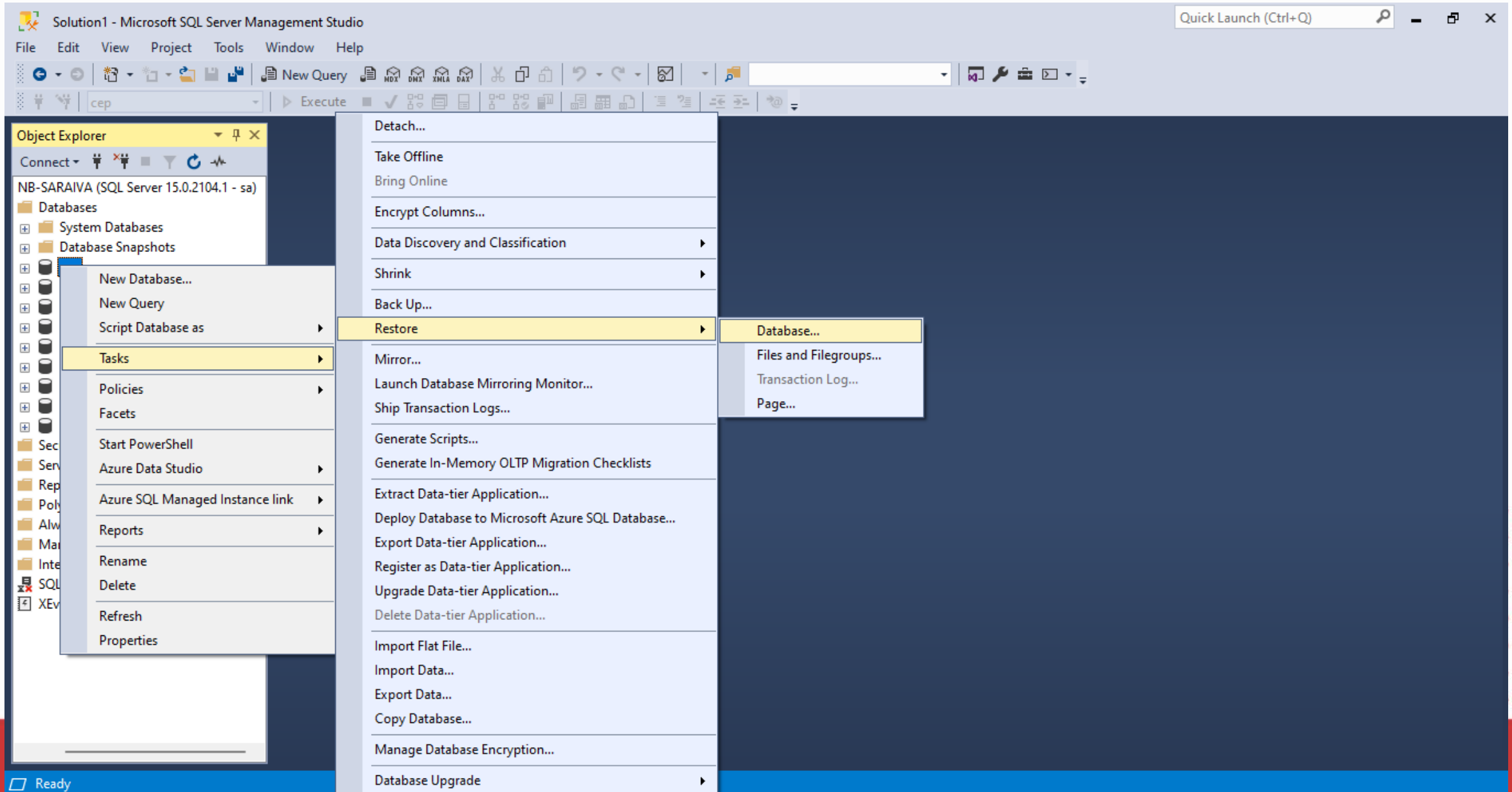
Progress

Ready

Add Remove

OK Cancel

# Restaurando o backup do banco CEP



# Restaurando o backup do banco CEP

Para restaurar o backup do banco, escolha a opção Device e aponte para o arquivo .bak baixado no passo anterior e, em seguida, clique em OK.

Restore Database - cep

⚠ A tail-log backup of the source database will be taken. View this setting on the Options page.

Select a page

- General
- Files
- Options

1

Script | ? Help

Source

☐ Database:

☒ Device:

2

Database: cep

Destination

Database: cep

Restore to: The last backup taken (te

Restore plan

Backup sets to restore:

Restore	Name	Component
<input checked="" type="checkbox"/>	cep-Full Database Backup	Database

Connection

NB-SARAIVA [sa]

[View connection properties](#)

Progress

Done

3

OK Cancel Help

Verify Backup Media

Select backup devices

Specify the backup media and its location for your restore operation.

Backup media type: File

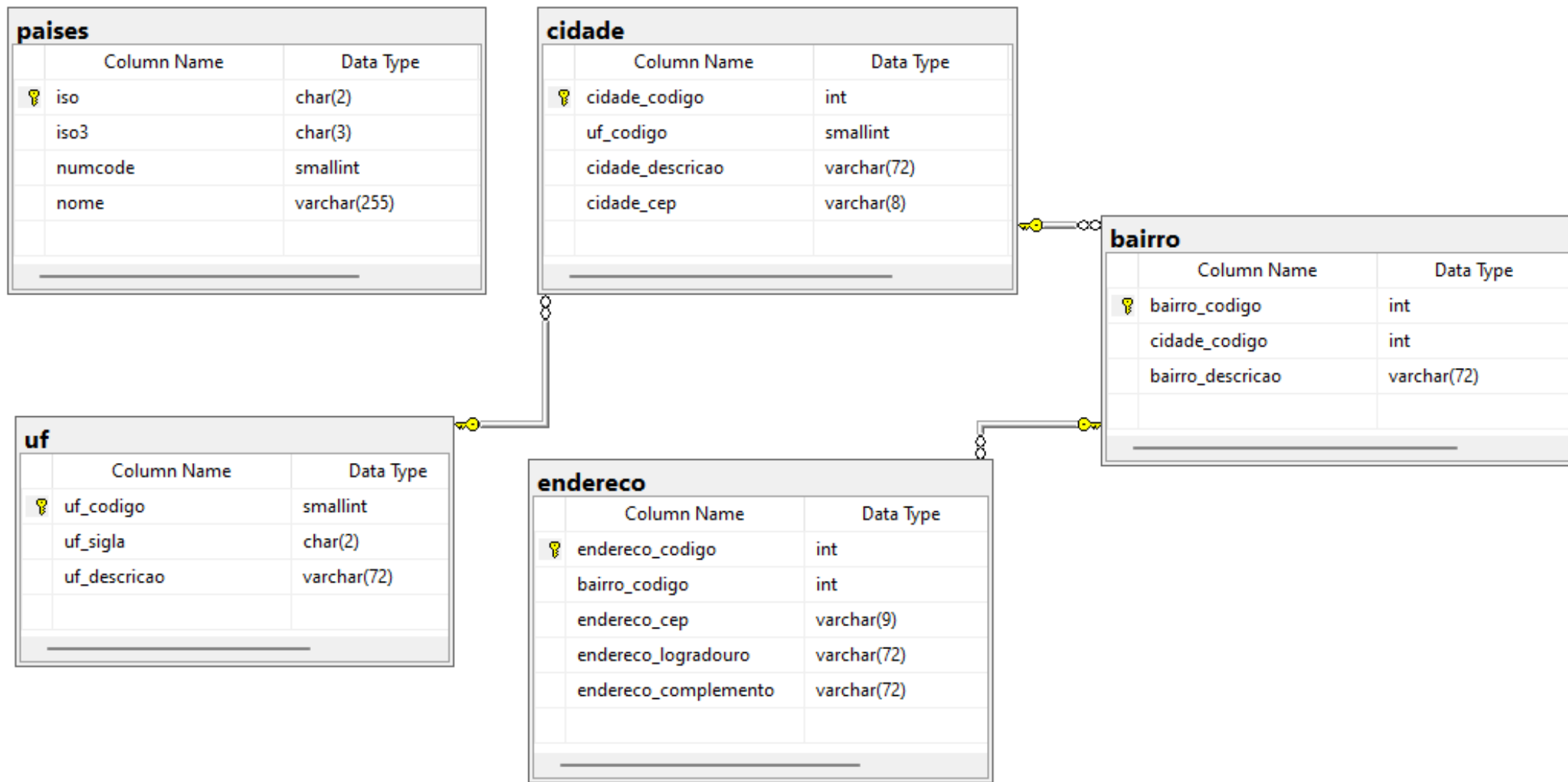
Backup media:

D:\Sesi\1. Banco de Dados\Backup\cep.bak

Add Remove Contents

OK Cancel Help

# Modelagem - CEP



# Script - CEP



```
CREATE DATABASE cep;
```

```
USE cep;
```

```
CREATE TABLE uf (  
    uf_codigo smallint NOT NULL default '0',  
    uf_sigla char(2) default NULL,  
    uf_descricao varchar(72) default NULL,  
    PRIMARY KEY (uf_codigo)  
);
```

```
INSERT INTO uf VALUES (1, 'AC', 'ACRE');  
INSERT INTO uf VALUES (2, 'AL', 'ALAGOAS');  
INSERT INTO uf VALUES (3, 'AP', 'AMAPÁ');  
INSERT INTO uf VALUES (4, 'AM', 'AMAZONAS');  
INSERT INTO uf VALUES (5, 'BA', 'BAHIA');  
INSERT INTO uf VALUES (6, 'CE', 'CEARÁ');  
INSERT INTO uf VALUES (7, 'DF', 'DISTRITO FEDERAL');  
INSERT INTO uf VALUES (8, 'ES', 'ESPÍRITO SANTO');  
INSERT INTO uf VALUES (9, 'RR', 'RORAIMA');  
INSERT INTO uf VALUES (10, 'GO', 'GOIÁS');  
INSERT INTO uf VALUES (11, 'MA', 'MARANHÃO');  
INSERT INTO uf VALUES (12, 'MT', 'MATO GROSSO');  
INSERT INTO uf VALUES (13, 'MS', 'MATO GROSSO DO SUL');  
INSERT INTO uf VALUES (14, 'MG', 'MINAS GERAIS');  
INSERT INTO uf VALUES (15, 'PA', 'PARÁ');  
INSERT INTO uf VALUES (16, 'PB', 'PARAÍBA');  
INSERT INTO uf VALUES (17, 'PR', 'PARANÁ');  
INSERT INTO uf VALUES (18, 'PE', 'PERNAMBUCO');
```

# Script - CEP



```
CREATE TABLE paises (  
    iso char(2) NOT NULL,  
    iso3 char(3) NOT NULL,  
    numcode smallint default NULL,  
    nome varchar(255) NOT NULL,  
    PRIMARY KEY (iso)  
);
```

```
INSERT INTO paises VALUES ('AD','AND',20,'Andorra');  
INSERT INTO paises VALUES ('AE','ARE',784,'Emiratos Árabes Unidos');  
INSERT INTO paises VALUES ('AF','AFG',4,'Afeganistão');  
INSERT INTO paises VALUES ('AG','ATG',28,'Antigua e Barbuda');  
INSERT INTO paises VALUES ('AI','AIA',660,'Anguilla');  
INSERT INTO paises VALUES ('AL','ALB',8,'Albânia');  
INSERT INTO paises VALUES ('AM','ARM',51,'Armênia');  
INSERT INTO paises VALUES ('AN','ANT',530,'Antilhas Holandesas');  
INSERT INTO paises VALUES ('AO','AGO',24,'Angola');  
INSERT INTO paises VALUES ('AQ','ATA',10,'Antártida');  
INSERT INTO paises VALUES ('AR','ARG',32,'Argentina');  
INSERT INTO paises VALUES ('AS','ASM',16,'Samoa Americana');  
INSERT INTO paises VALUES ('AT','AUT',40,'Áustria');  
INSERT INTO paises VALUES ('AU','AUS',36,'Austrália');  
INSERT INTO paises VALUES ('AW','ABW',533,'Aruba');
```



# Script - CEP



```
CREATE TABLE cidade (  
  cidade_codigo int NOT NULL default '0',  
  uf_codigo int default NULL,  
  cidade_descricao varchar(72) default NULL,  
  cidade_cep varchar(8) default NULL,  
  PRIMARY KEY (cidade_codigo),  
  FOREIGN KEY uf_codigo REFERENCES uf(uf_codigo)  
);
```

```
INSERT INTO cidade VALUES (1,1,'ACRELÂNDIA','69945000');  
INSERT INTO cidade VALUES (2,1,'ASSIS BRASIL','69935000');  
INSERT INTO cidade VALUES (3,1,'BRASILIA','69932000');  
INSERT INTO cidade VALUES (4,1,'BUJARI','69923000');  
INSERT INTO cidade VALUES (5,1,'CAPIXABA','69922000');  
INSERT INTO cidade VALUES (6,1,'CRUZEIRO DO SUL','69980000');  
INSERT INTO cidade VALUES (7,1,'EPITACIOLÂNDIA','69934000');  
INSERT INTO cidade VALUES (8,1,'FEIJÓ','69960000');  
INSERT INTO cidade VALUES (9,1,'JORDÃO','69975000');  
INSERT INTO cidade VALUES (10,1,'MÂNCIO LIMA','69990000');  
INSERT INTO cidade VALUES (11,1,'MANOEL URBANO','69950000');  
INSERT INTO cidade VALUES (12,1,'MARECHAL THAUMATURGO','69983000');
```



# Script - CEP



```
CREATE TABLE bairro (  
    bairro_codigo int NOT NULL default '0',  
    cidade_codigo int default NULL,  
    bairro_descricao varchar(72) default NULL,  
    PRIMARY KEY (bairro_codigo),  
    FOREIGN KEY cidade_codigo REFERENCES (cidade_codigo)  
);
```

```
INSERT INTO bairro VALUES (1,16,'ABRAHÃO ALAB');  
INSERT INTO bairro VALUES (2,16,'ADALBERTO ARAGÃO');  
INSERT INTO bairro VALUES (3,16,'AEROPORTO VELHO');  
INSERT INTO bairro VALUES (4,16,'ALEGRIA');  
INSERT INTO bairro VALUES (5,16,'AREAL');  
INSERT INTO bairro VALUES (6,16,'AVIÁRIO');  
INSERT INTO bairro VALUES (7,16,'BAHIA');  
INSERT INTO bairro VALUES (8,16,'BAHIA NOVA');  
INSERT INTO bairro VALUES (9,16,'BAIXA DA COLINA');  
INSERT INTO bairro VALUES (10,16,'BASE');  
INSERT INTO bairro VALUES (11,16,'BELA VISTA');  
INSERT INTO bairro VALUES (12,16,'BOSQUE');  
INSERT INTO bairro VALUES (13,16,'CADEIA NOVA');
```

# Script - CEP



```
CREATE TABLE endereco (  
    endereco_codigo int NOT NULL default '0',  
    bairro_codigo int default NULL,  
    endereco_cep varchar(9) default NULL,  
    endereco_logradouro varchar(72) default NULL,  
    endereco_complemento varchar(72) default NULL,  
    PRIMARY KEY (endereco_codigo),  
    KEY bairro_codigo (bairro_codigo)  
);
```

```
INSERT INTO endereco VALUES (1,101,'57051901','AVENIDA DOM ANTÔNIO BRANDÃO, 333','');  
INSERT INTO endereco VALUES (2,92,'57045900','AVENIDA PRESIDENTE ROOSEVELT, 206','');  
INSERT INTO endereco VALUES (3,105,'57052901','AVENIDA FERNANDES LIMA, 3700','');  
INSERT INTO endereco VALUES (4,98,'57020905','RUA ENGENHEIRO ROBERTO GONÇALVES MENEZES, 149','');  
INSERT INTO endereco VALUES (5,121,'57035900','AVENIDA ELVARO OTACILIO, 2991','');  
INSERT INTO endereco VALUES (6,110,'57036900','AVENIDA ELVARO OTACILIO, 4065','');  
INSERT INTO endereco VALUES (7,98,'57020901','PRAÇA MARECHAL FLORIANO PEIXOTO, S/N','');  
INSERT INTO endereco VALUES (8,101,'57051900','AVENIDA ARISTEU DE ANDRADE, 355','');  
INSERT INTO endereco VALUES (9,96,'57017900','RUA GENERAL HERMES, 80','');  
INSERT INTO endereco VALUES (10,112,'57032901','AVENIDA COMENDADOR GUSTAVO PAIVA, 2990','');  
INSERT INTO endereco VALUES (11,101,'57055904','RUA GOIAS, S/N','');
```

# Exercícios de consulta SQL



**Construa e execute as consultas abaixo e cole o resultado obtido em cada uma delas.**

1. Consulte e retorne todos os registros da tabela **cidade**;
2. Selecione todos os registros das **cidades** que começam com a letra B e informe a quantidade encontrada;
3. Selecione **todas** as cidades do estado do **PR**;



# Exercícios de consulta SQL



Execute as consultas abaixo e cole o resultado obtido em cada uma delas.

4. Selecione todas as **idades** cujo **uf\_codigo** esteja entre 25 e 27;

5. Consulte e retorne todos os  **Bairros** cujo **cidade\_codigo** seja igual a 8962;



# Exercícios de consulta



6. Consulte e retorne todos os **países** que começam com a letra **A** e a letra **Z**;

7. Consulte e retorne todos os **bairros** da cidade de **Barretos** - SP da tabela **bairro**;

8. Consulte e retorne todos os ceps da cidade de **Pitangueiras** - **SP** da tabela **cidade** ;



## Exercícios de consulta



9. Selecione todos os endereços cujo o campo **endereço\_complemento** seja igual a '**Comercio**' ou '**Com?rcio**' (falha na codificação da acentuação no banco de dados);

Para corrigir este problema, utilize o seguinte comando:

```
UPDATE endereco SET endereco_complemento = 'Comércio'  
WHERE endereco_complemento = 'Com?rcio';
```

10. Selecione todas as **faixas de cep** do estado do **GO** e que esteja entre os valores '72880000' e '76820000'.





**Wagner Cesar Vieira**  
[wagner.vieira@sp.senai.br](mailto:wagner.vieira@sp.senai.br)

