

# Banco de Dados

[Jose.wellington@ceub.edu.br](mailto:Jose.wellington@ceub.edu.br)

# Planejamento

## 1 Dia

### a. Manhã

**Modelagem de dados**

### b. Tarde

**Normalização de dados**

## 2 Dia

### c. Manhã

**Comandos DDL**

**1 avaliação – Modelagem de dados e Normalização**

### d. Tarde

**Comandos DML – DQL Avançados**

## 3 Dia

### e. Manhã

**Banco de Dados NoSQL – Redis**

**2 avaliação – Comandos SQL**

### f. Tarde

**NoSQL – MongoDB**

**NoSQL – Neo4j**

# Agenda

- Exercício – Prático
- Backup
- Restore
- Comandos de Restrição em SQL
- Inner Join
- View
- Outer Join - Right e Left
- Manipulação de Data
- Cálculo
- Consulta de Agregação
- Transações – ACID Comandos do SQL
- SubQuery
- Índices
- Plano de execução
- Stored Procedure

# Exercício - Prático

## Exercício 01 – Fazer o MER e criar o banco de dados

### ■ Recursos Humanos

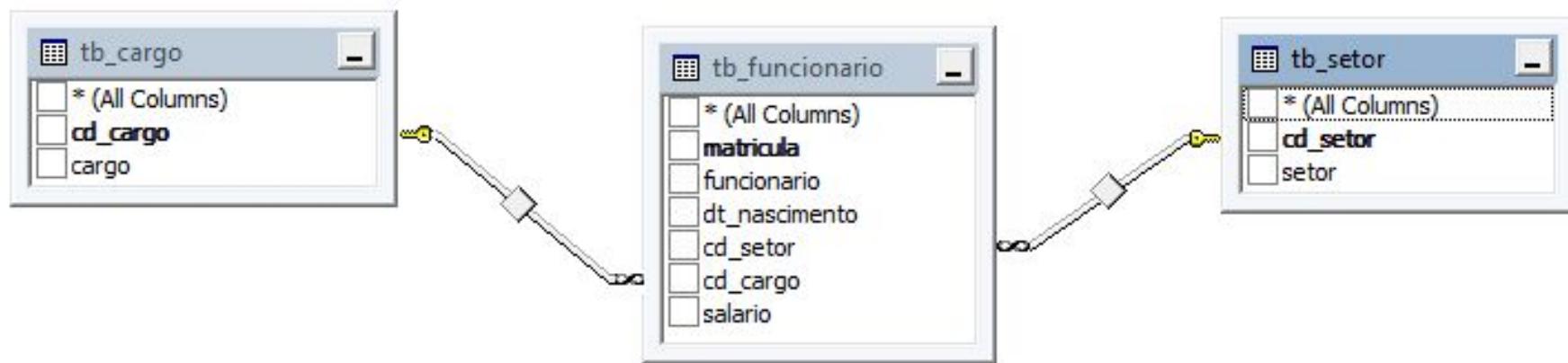
- A empresa de pesquisa agropecuária de referência no Brasil precisa controlar seus **colaboradores, as informações básicas.**

#### ■ Dados levantados:

- Funcionário
- Data de nascimento
- Cargo – 1 funcionário só pode ter 1 cargo
- setor – 1 funcionário só pode trabalhar em 1 setor
- Salário

# Exercício – Criar o Banco de Dados no SQL Server

# Criar o Banco de Dados



- Rodar o script no MySQL

# Exercício – Bd livro

SGBD - SQL - 2. parte

## ■ BD\_Livro\_01

- Uma empresa de venda de livros está querendo ter um cadastro (controle dos títulos) da sua empresa.

## ■ Dados levantados

- isbn int
- titulo char(50)
- editora char(30)
- genero char(30)
- preco money

## ■ Regra de negócio

- um titulo só tem um isbn
  - 1 (um) titulo só pode ter 1 (uma) editora e 1 (uma) editora por ter vários títulos
  - 1 (um) titulo só pode ter 1 (um) genero e 1 (um) genero por ter vários títulos
- Exemplo gênero medicina pode ter vários títulos.

# Gabarito

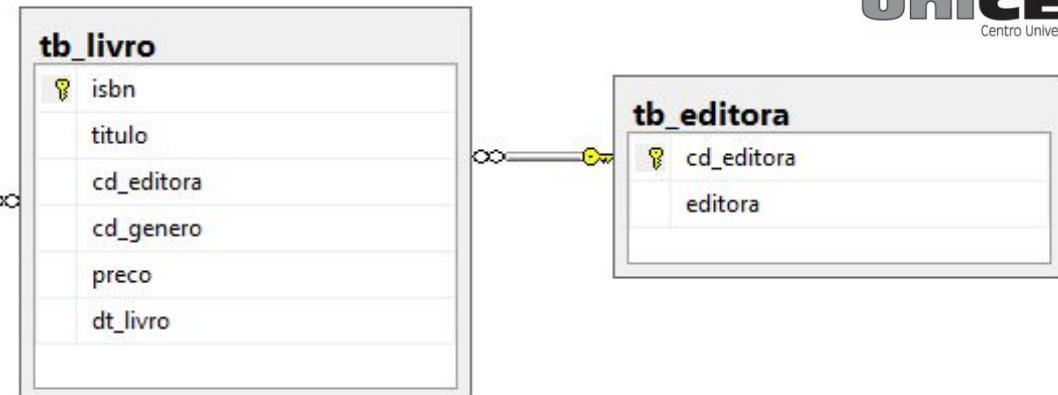
SGBD - SQL - 2. parte

## Exercício – Fazer o MER e criar o banco de dados



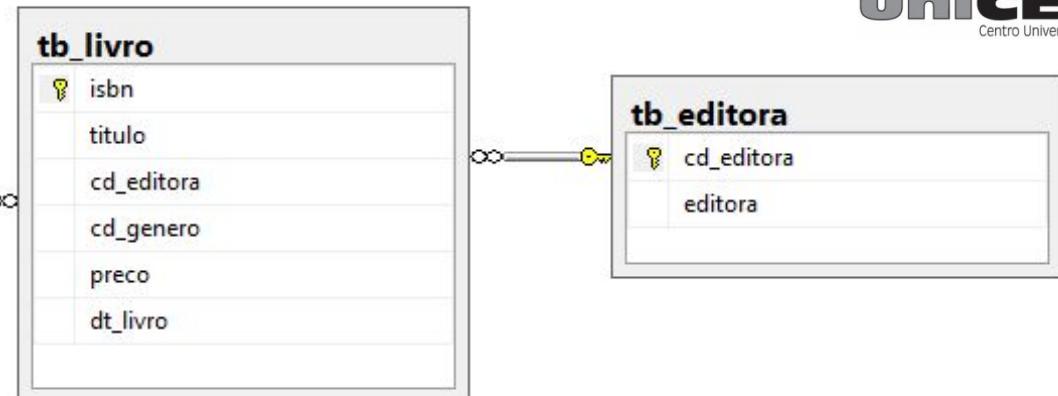
- Baixar o backup e Restaurar o Banco de dados

## Gabarito



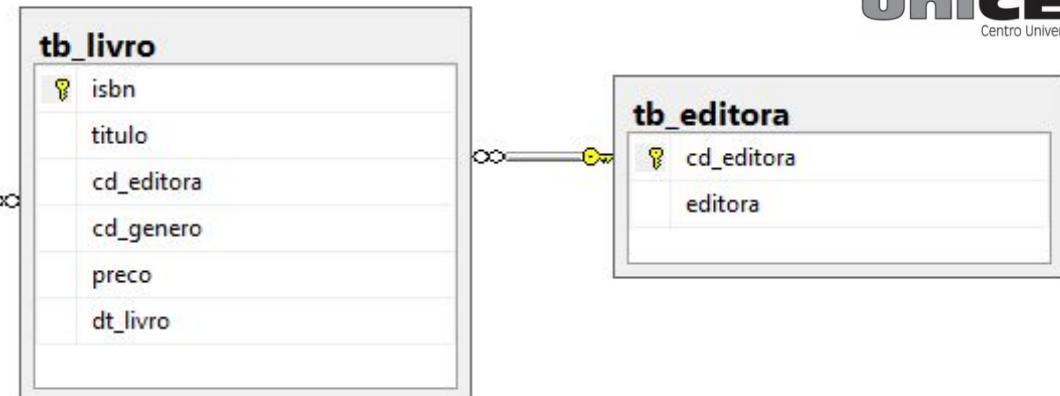
```
insert into tb_genero
(cd_genero, genero)
values
(1, 'Computacao'),
(2, 'Medicina'),
(3, 'Engenharia'),
(4, 'Juridico'),
(5, 'Arquitetura'),
(6, 'Biologia'),
(7, 'Mecatronica')
```

# Gabarito



```
insert into tb_editora
(cd_editora, editora)
values
(1, 'novatec'),
(2, 'amazon'),
(3, 'coopmed'),
(4, 'livraria florence'),
(5 , 'blucher'),
(6,'Mundial'),
(7,'saraiva'),
(8,'Editora Forum'),
(9, 'Dickens')
```

## Gabarito

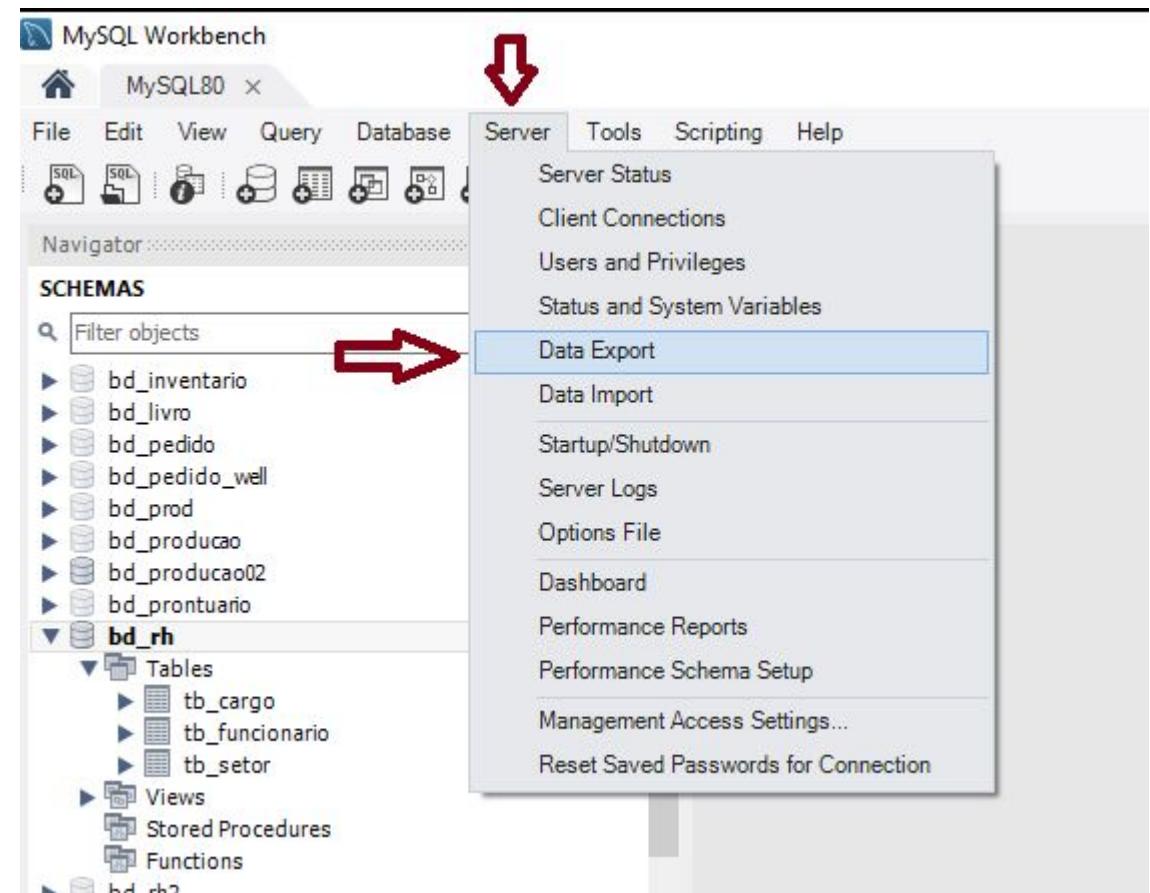


```
insert into tb_livro
(isbn, titulo, cd_editora, cd_genero, preco, dt_livro)
values
(1, 'banco de dados',1,1, 300, '2022-10-02'),
(2, 'Engenharia de Software',1,1, 350, '2023-09-02'),
(3, 'Ortopedia',3,2, 310, '2024-07-02'),
(4, 'Cardiologia',4,2, 320, '2022-10-02'),
(5, 'Estrutura Predial',5,3, 200, '2023-10-02'),
(6, 'Estrutura Hidraulica',6,3, 300, '2024-06-02'),
(7, 'Direito Penal',7,4, 150, '2022-07-02'),
(8, 'Direito Civil',8,4, 200, '2023-06-02'),
(9, 'Cores ',7,5, 200, '2024-08-02'),
(10, 'Paisagismo',8,5, 250, '2022-09-02'),
(11, 'Virus',9,6, 300, '2023-10-02'),
(12, 'Bacteria',9,6, 300, '2024-03-02');
```

# Backup



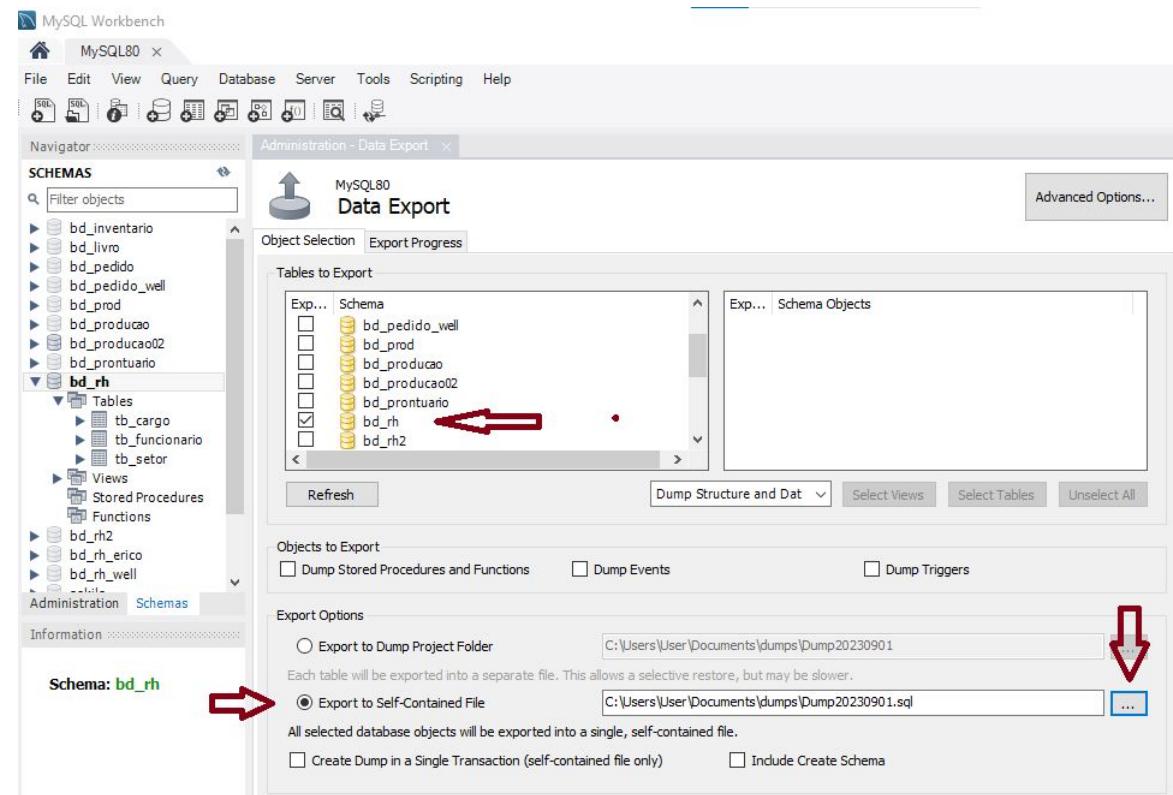
**Backup** é uma cópia segura de dados armazenada em local separado do original.





# Backup - MySQL

O **backup** protege dados contra perdas causadas por **falhas, erros, ataques ou desastres.**



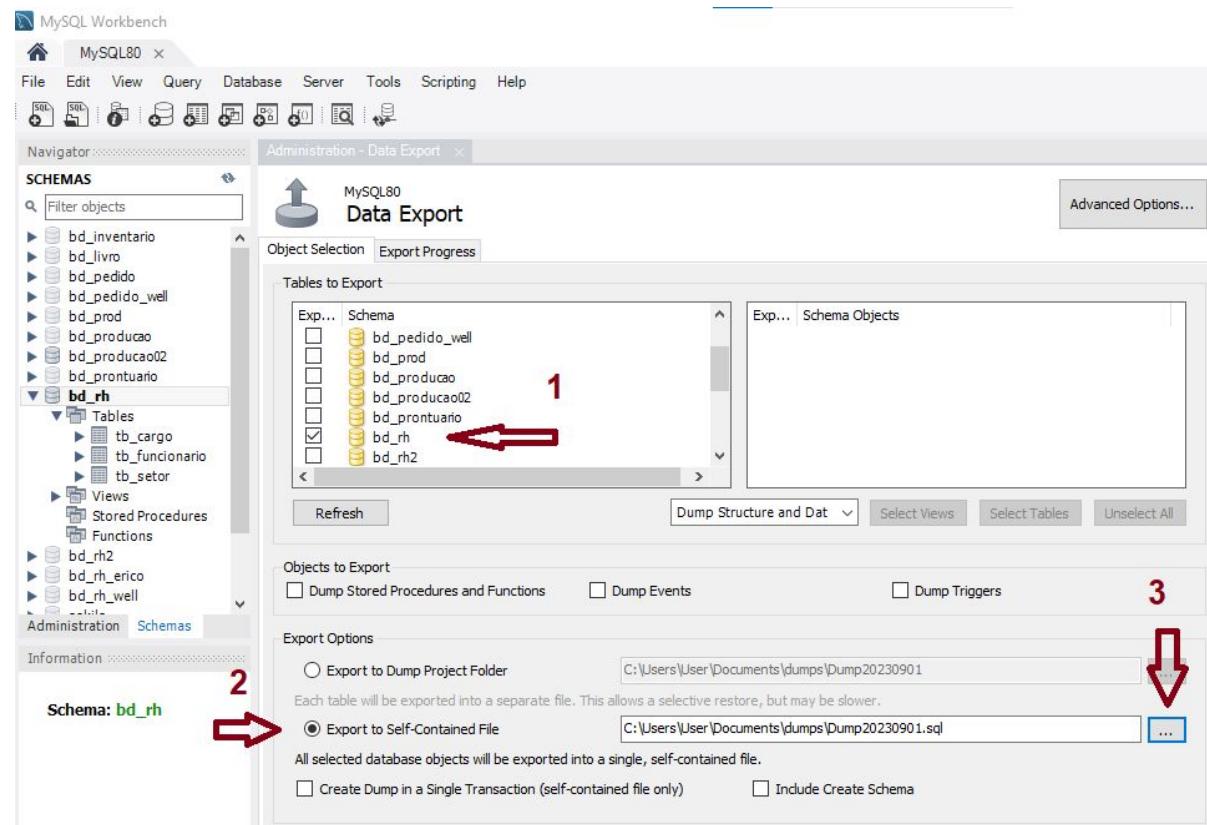


# Backup - MySQL

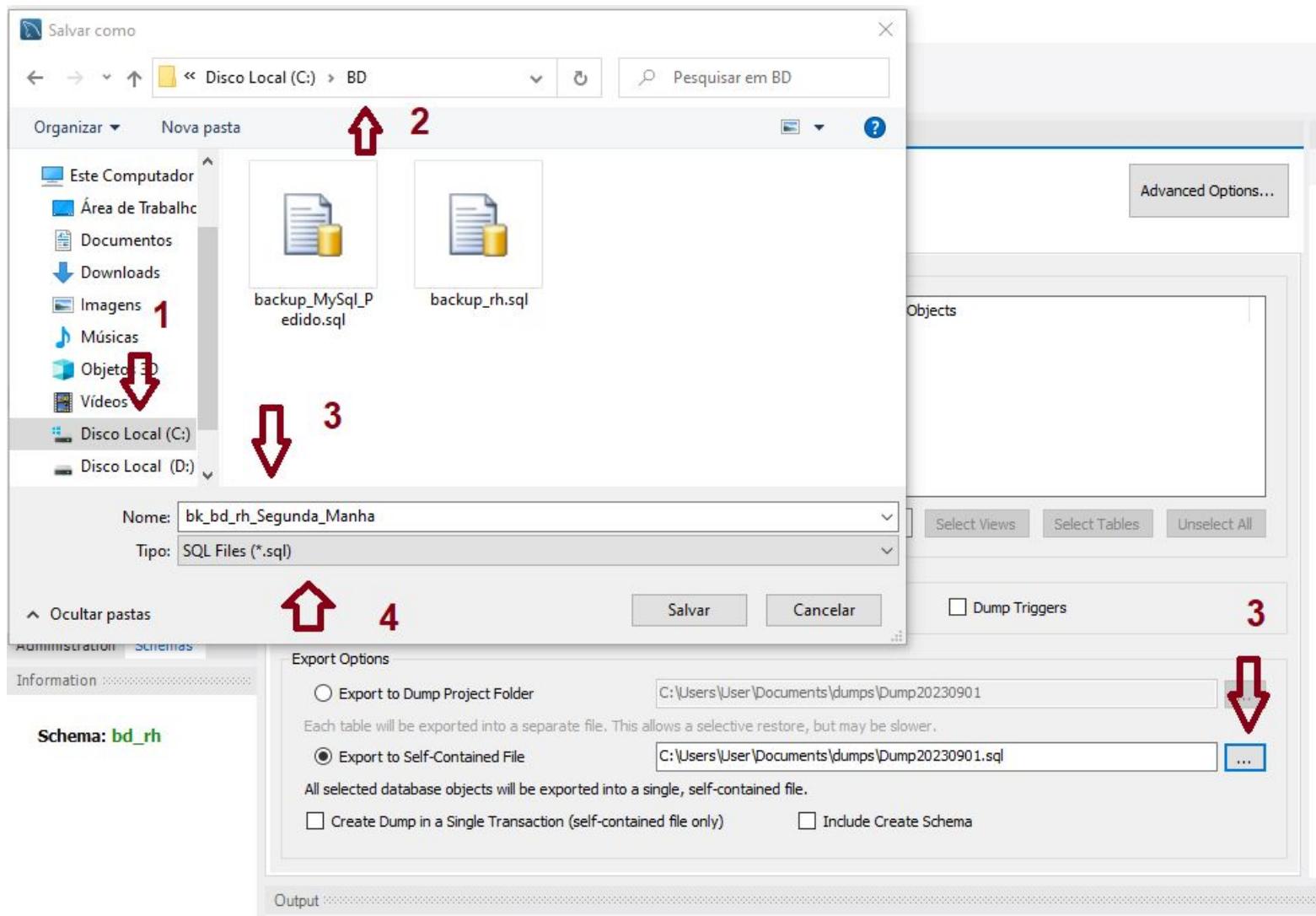
1 – Origem – Banco de dados  
para fazer backup.

2 – Destino – Contained File

3 – Destino – Selecionar  
Pasta e arquivo



# Backup - MySQL



# Backup - MySQL

Navigator :::::::::::::: Administration - Data Export x

SCHEMAS

MySQL80

Data Export

Object Selection Export Progress

Advanced Options...

Export Completed

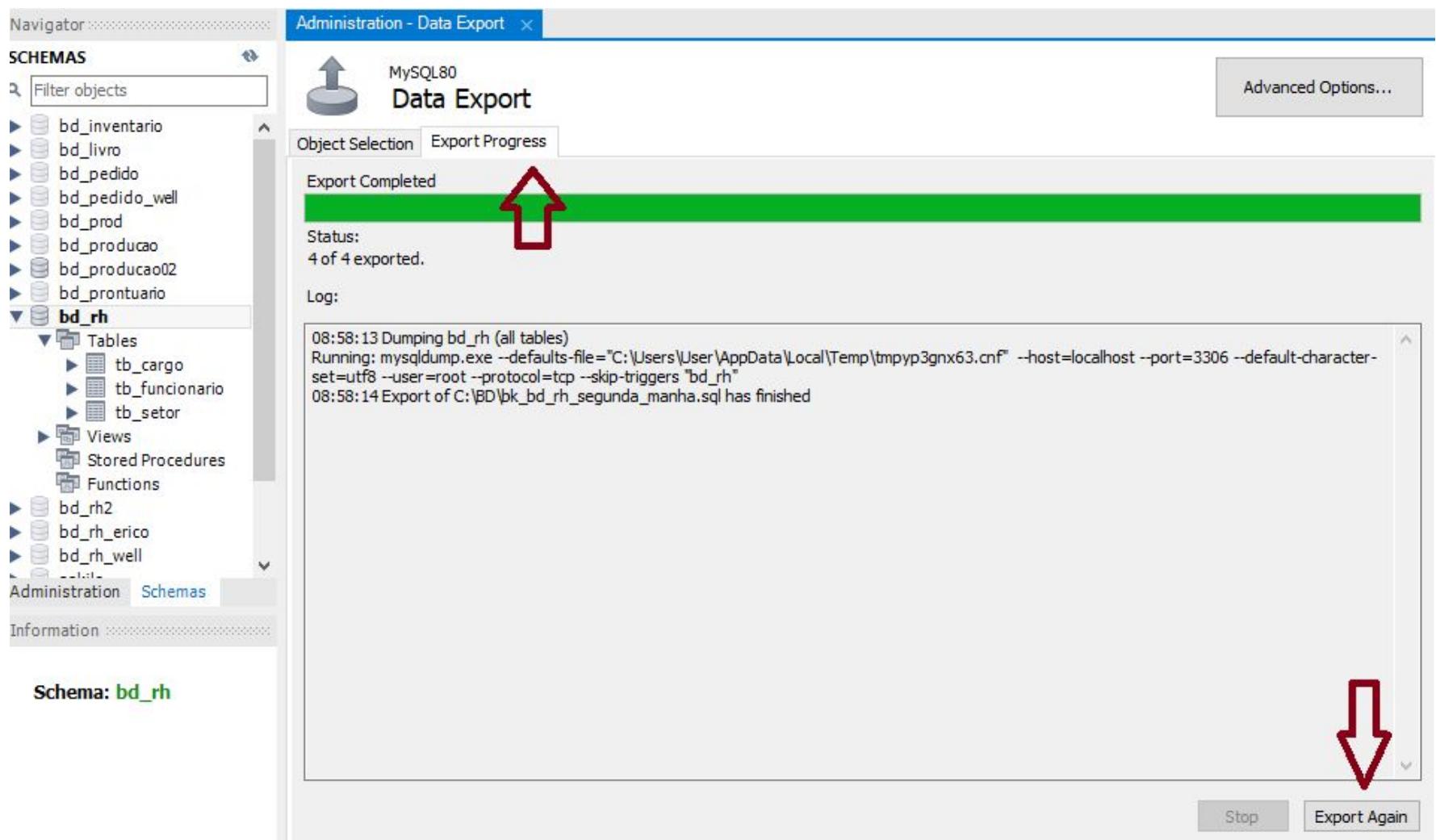
Status:  
4 of 4 exported.

Log:

```
08:58:13 Dumping bd_rh (all tables)
Running: mysqldump.exe --defaults-file="C:\Users\User\AppData\Local\Temp\tmpyp3gnx63.cnf" --host=localhost --port=3306 --default-character-set=utf8 --user=root --protocol=tcp --skip-triggers "bd_rh"
08:58:14 Export of C:\BD\bk_bd_rh_segunda_minha.sql has finished
```

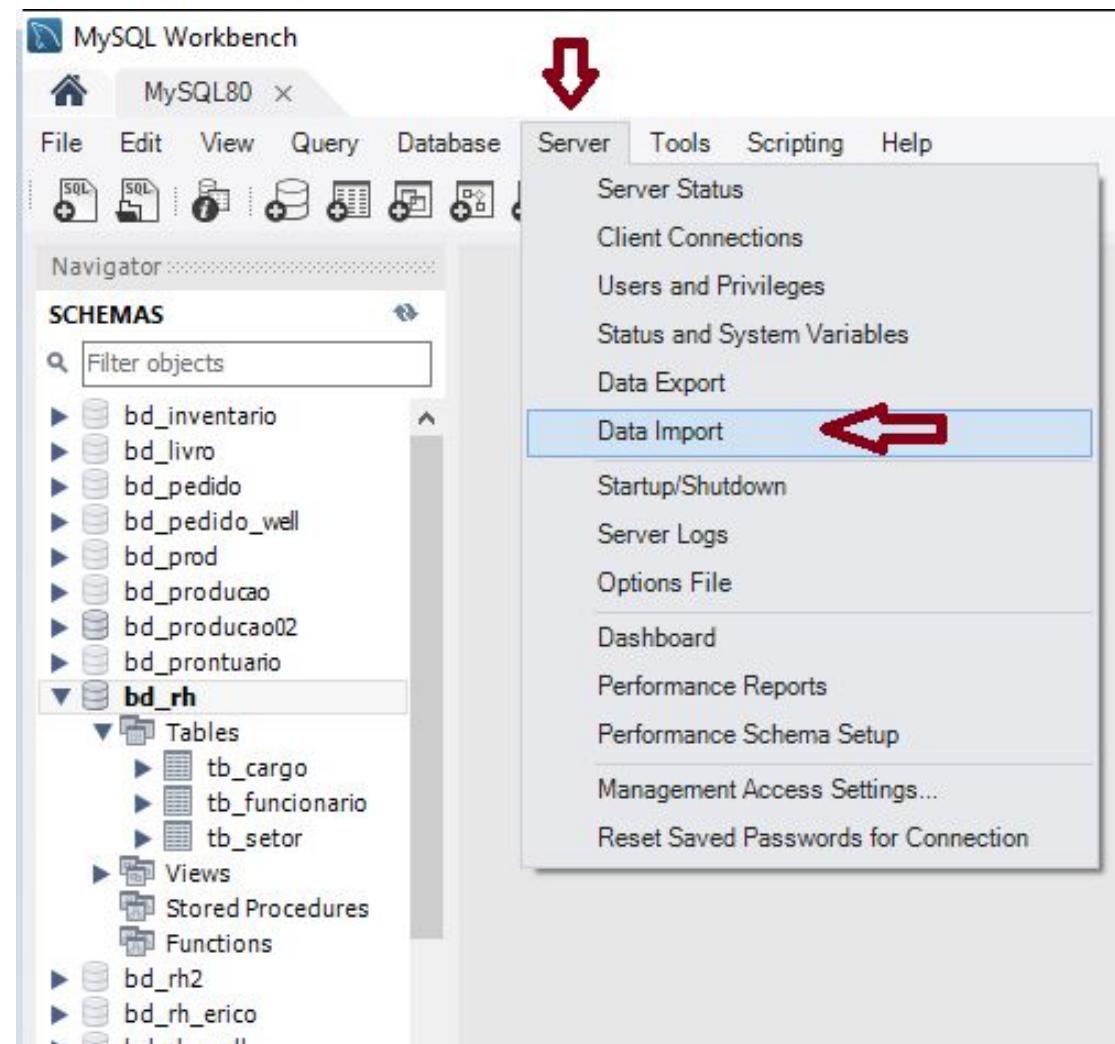
Schema: bd\_rh

Stop Export Again



# Restore

Restaurar um backup  
é recuperar dados  
para o local original ou  
um destino  
especificado.

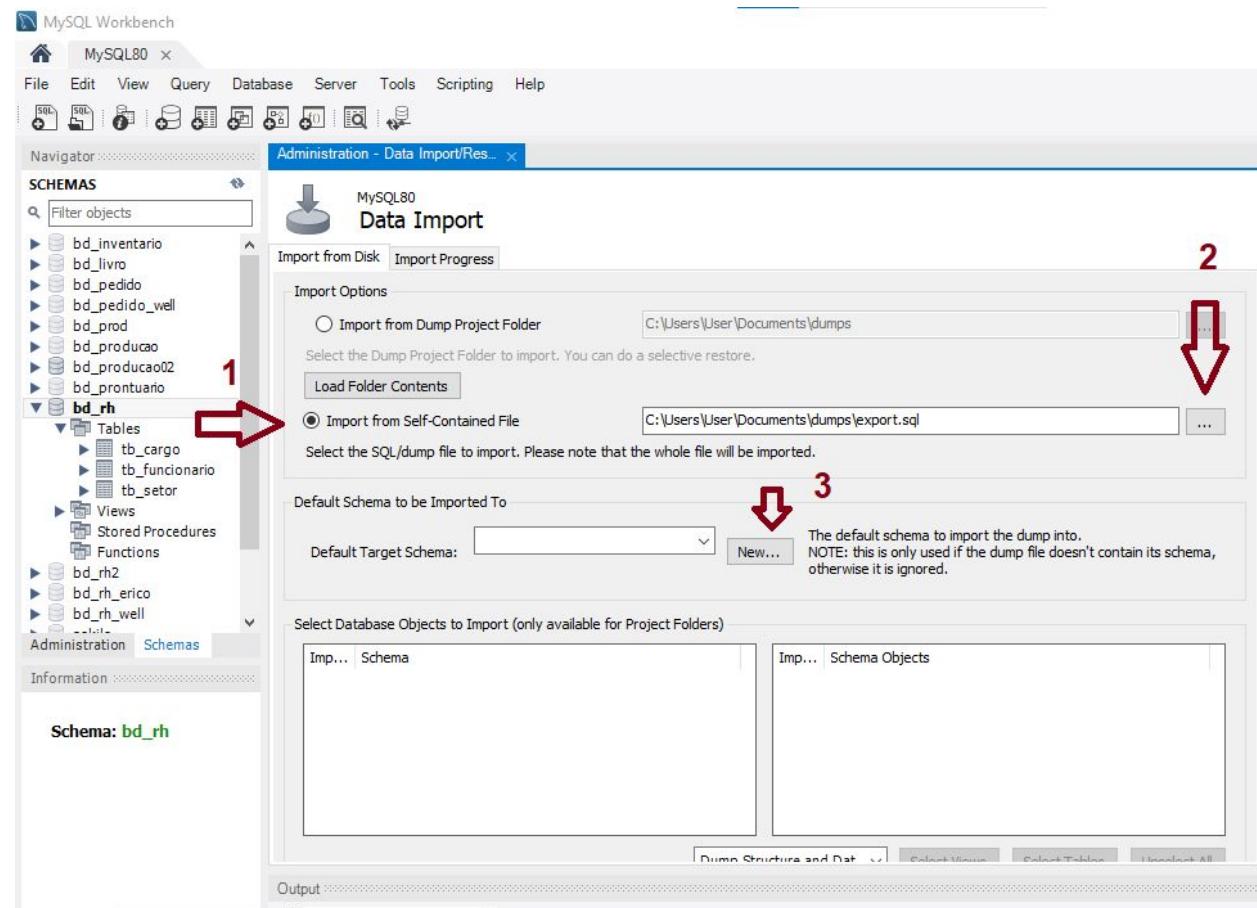


## Restore - MySQL

1 – Origem – Banco de dados fez o backup – file contained.

2 – Origem – Local onde foi armazenado o Contained File

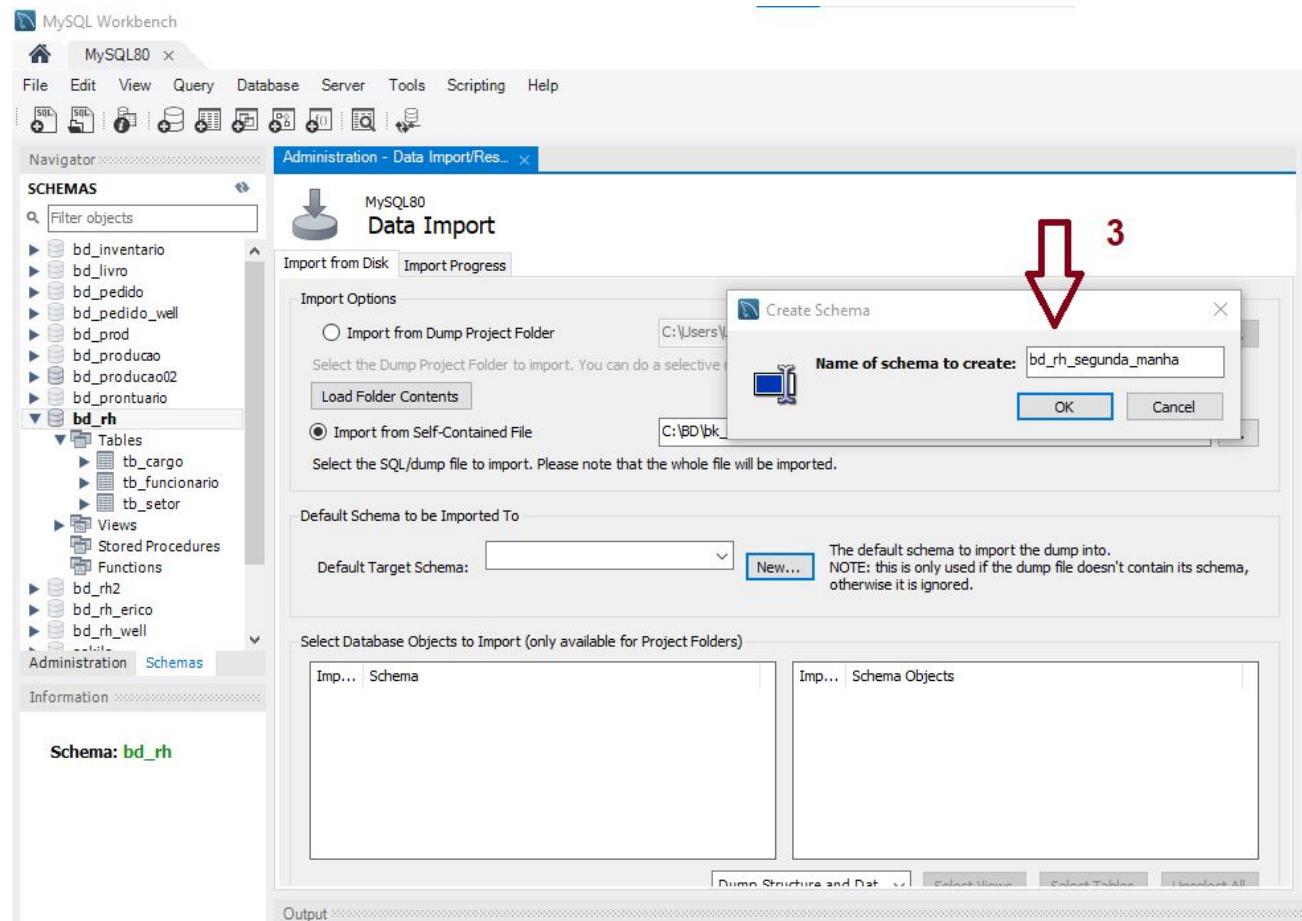
3 – Destino – Digitar o nome do novo banco de dados





## Restore - MySQL

3 – Destino – Digitar o nome do novo banco de dados



The screenshot shows the MySQL Workbench interface with the 'Administration - Data Import/Res...' tab selected. In the Navigator pane, the 'Schemas' section is open, showing various database schemas like 'bd\_inventario', 'bd\_livro', 'bd\_pedido', etc., and a schema named 'bd\_rh' which is expanded to show 'Tables', 'Views', and other objects. The main panel displays the 'Data Import' configuration window. Under 'Import Options', the 'Import from Self-Contained File' radio button is selected, and the file path 'C:\BD\bk' is specified. A 'Create Schema' dialog box is overlaid on the main window, with the schema name 'bd\_rh\_segunda\_manha' entered in its 'Name of schema to create:' field. A large red arrow points from the text above to this input field. The 'OK' button in the dialog box is also highlighted with a blue box.



# Restore - MySQL

MySQL Workbench

MySQL80 ×

File Edit View Query Database Server Tools Scripting Help

SQL SQL+ Database Editor Schema Browser

Navigator Administration - Data Import/Res... ×

**SCHEMAS**

Filter objects

- bd\_inventario
- bd\_livro
- bd\_pedido
- bd\_pedido\_well
- bd\_prod
- bd\_producao
- bd\_producao02
- bd\_prontuario
- **bd\_rh**
  - ▼ Tables
    - tb\_cargo
    - tb\_funcionario
    - tb\_setor
  - Views
  - Stored Procedures
  - Functions
- bd\_rh2
- bd\_rh\_erico
- bd\_rh\_well

Administration Schemas

Information

Schema: **bd\_rh**

**Data Import**

MySQL80

Import from Disk Import Progress

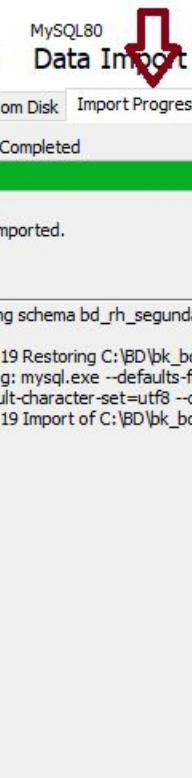
Import Completed

Status:  
1 of 1 imported.

Log:

```
Creating schema bd_rh_segunda_minha
09:06:19 Restoring C:\BD\bk_bd_rh_segunda_minha.sql
Running: mysql.exe --defaults-file="C:\Users\User\AppData\Local\Temp\tmp35_yrdty.cnf" --protocol=tcp --host=localhost --user=root --port=3306
--default-character-set=utf8 --comments --database=bd_rh_segunda_minha < "C:\BD\bk_bd_rh_segunda_minha.sql"
09:06:19 Import of C:\BD\bk_bd_rh_segunda_minha.sql has finished
```

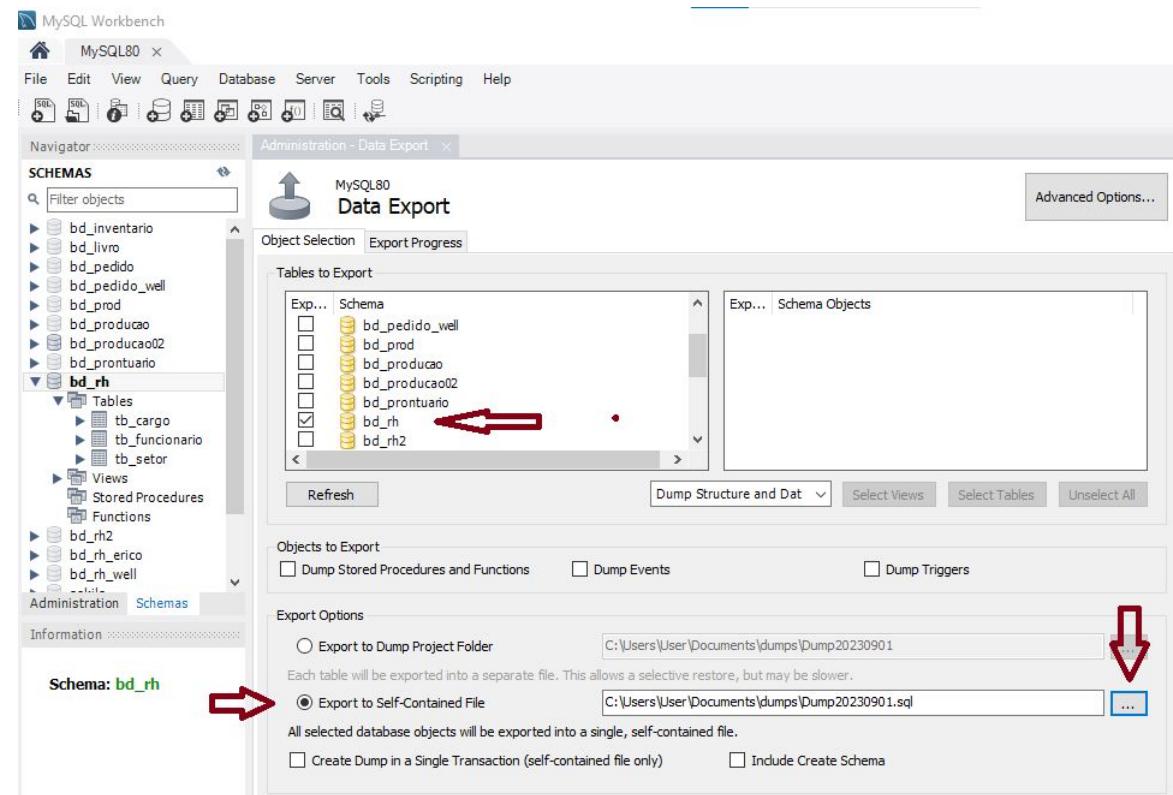
Stop Import Again





# Backup - MySQL

A finalidade do **backup** é proteger dados contra perdas causadas por **falhas, erros, ataques ou desastres.**



# Exercício

## exercício

- 1) Incluir na tb\_gênero o gênero publicidade
- 2) Alterar na tabela tb\_livro isbn = 1 preco (500)
- 3) Listar a tabela (tb\_livro) preco > 300

# Exercício 01

- **Bd\_pedido**

- Uma concessionária está querendo controlar os veiculos, vendas, vendedores. .

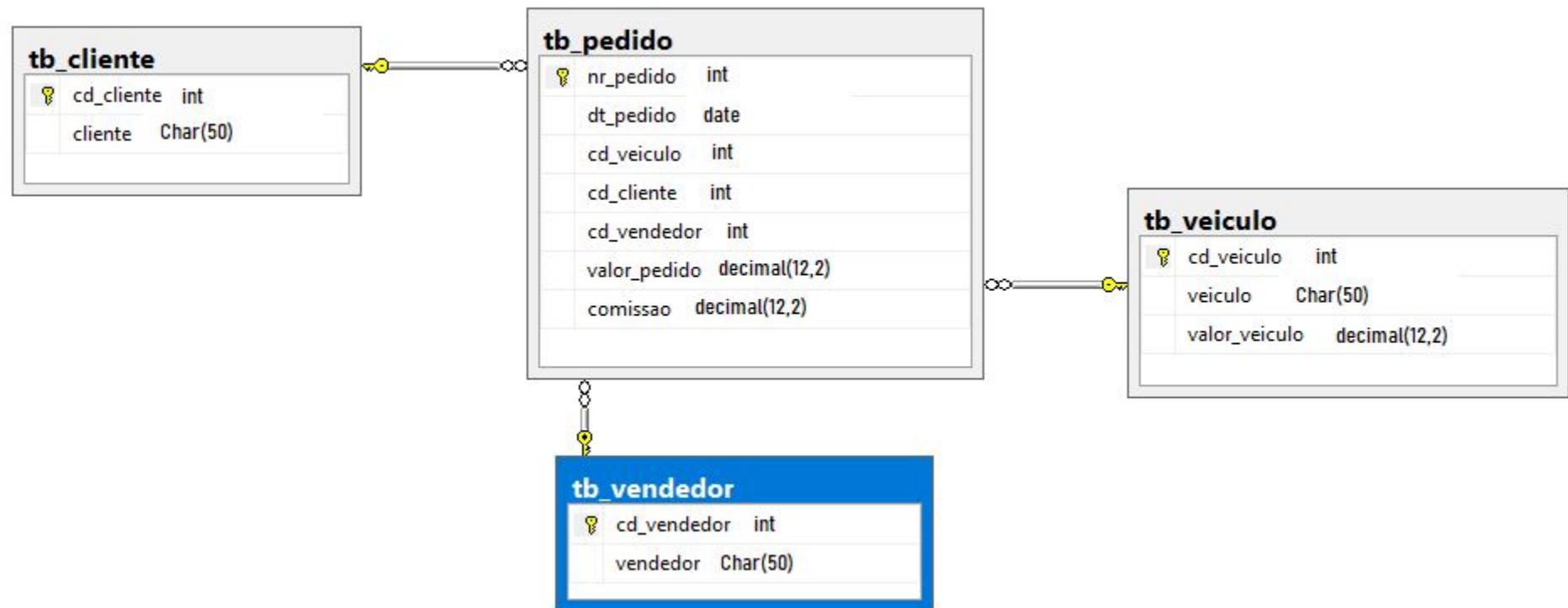
- **Dados levantados**

- Nr\_pedido int
  - Data\_pedido date
  - veiculo char(50)
  - Valor do veiculo money ou decimal(12,2)
  - cliente char(50)
  - Vendedor char(50)
  - Valor\_pedido money ou decimal(12,2)
  - Comissão money ou decimal(12,2)

- **Regra de negócio**

- 1 pedido só tem 1 cliente e 1 cliente pode fazer várias compras
  - 1 pedido só tem 1 veiculo e 1 veiculo ser vendido várias vezes (veiculo hb20 pode ser vendido várias vezes)
  - 1 pedido só tem 1 vendedor e 1 vendedor pode fazer várias vendas.

# Bd\_pedido



- Baixar o backup e Restaurar o Banco de dados

# Bd\_pedido



-- vendedor

```
select * from tb_vendedor
```

```
Insert into tb_vendedor
(cd_vendedor, vendedor)
Values
(1, 'Anibal'),
(2, 'Antonio de Moraes'),
(3, 'Barbara Alcantara'),
(4, 'Deise Castro'),
(5, 'Eider Nascimento');
```

# Bd\_pedido



```
-- cliente
select * from tb_cliente
```

```
Insert into tb_cliente
(cd_cliente, cliente)
Values
(1, 'Vallu Nascimento'),
(2, 'Rogeria Negreti'),
(3, 'Henrique Silva'),
(4, 'Wellington Alves'),
(5, 'Jose Pereira');
```

# Bd\_pedido



```
-- veiculo
select * from tb_veiculo
```

```
Insert into tb_veiculo
(cd_veiculo, veiculo, valor_veiculo)
Values
(1, 'Onix', 52000),
(2, 'Prisma', 49000),
(3, 'S10', 109000),
(4, 'Cruze', 101000),
(5, 'Spin', 69000),
(6, 'Cobalt', 63000);
```

# Bd\_pedido

-- Pedido

```
select * from tb_pedido

insert into tb_pedido
(nr_pedido, dt_pedido, cd_veiculo,
cd_cliente, cd_vendedor,
valor_pedido, comissao)
values
(1, '2019-01-10', 1, 2, 3, 52000, 0),
(2, '2019-02-20', 2, 3, 4, 49000, 0),
(3, '2019-03-30', 3, 4, 5, 109000, 0),
(4, '2019-04-10', 4, 5, 1, 101000, 0),
(5, '2019-05-20', 5, 5, 1, 69000, 0),
(6, '2019-06-30', 6, 1, 2, 63000, 0),
(7, '2019-07-10', 1, 4, 5, 52000, 0),
(8, '2019-08-20', 1, 4, 5, 52000, 0),
(9, '2019-09-30', 1, 4, 5, 52000, 0),
(10, '2019-10-10', 1, 4, 5, 52000, 0);
```



# Comandos de Restrição em SQL

## Restrição

```
select * from tb_funcionario
```

150 %

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario	
1	1	Ana Clara	1977-07-05	1	5	3000.00	
2	2	Patricia Azevedo	1944-07-04	1	1	4000.00	
3	3	Jose Maria	1971-05-10	1	3	6000.00	
4	4	Sonia Abrantes	1979-05-29	1	4	7000.00	
5	5	Valdir Reinaldo	1960-09-22	2	2	16000.00	
6	6	Jose Alberto	1955-01-13	2	2	15000.00	

## Clausula Where

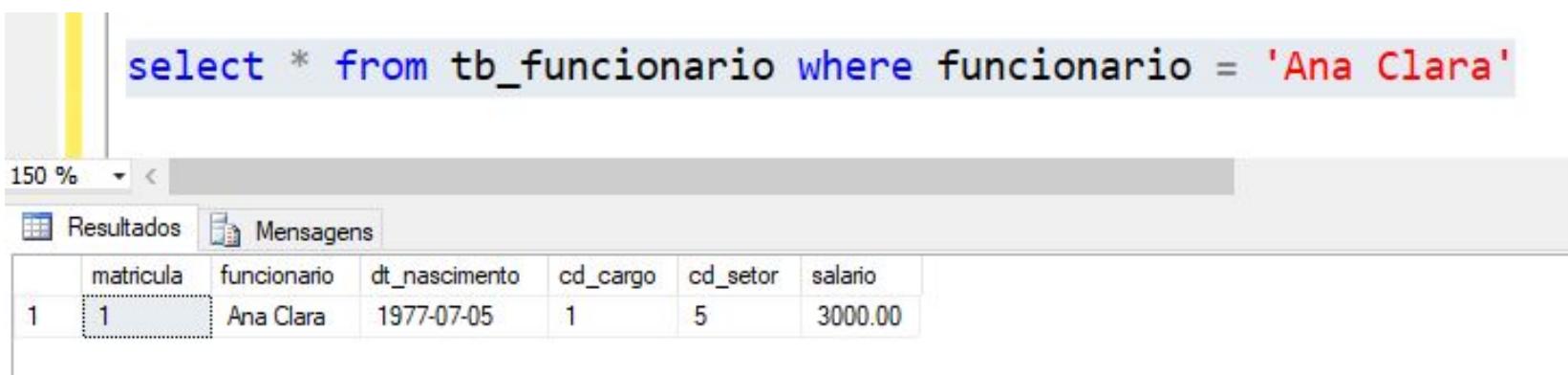
```
select * from tb_funcionario where salario < 5000
```

150 %

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario	
1	1	Ana Clara	1977-07-05	1	5	3000.00	
2	2	Patricia Azevedo	1944-07-04	1	1	4000.00	

## Predicados

Um predicado pode ser:



A screenshot of a SQL query execution interface. At the top, a code editor window displays the following SQL query:

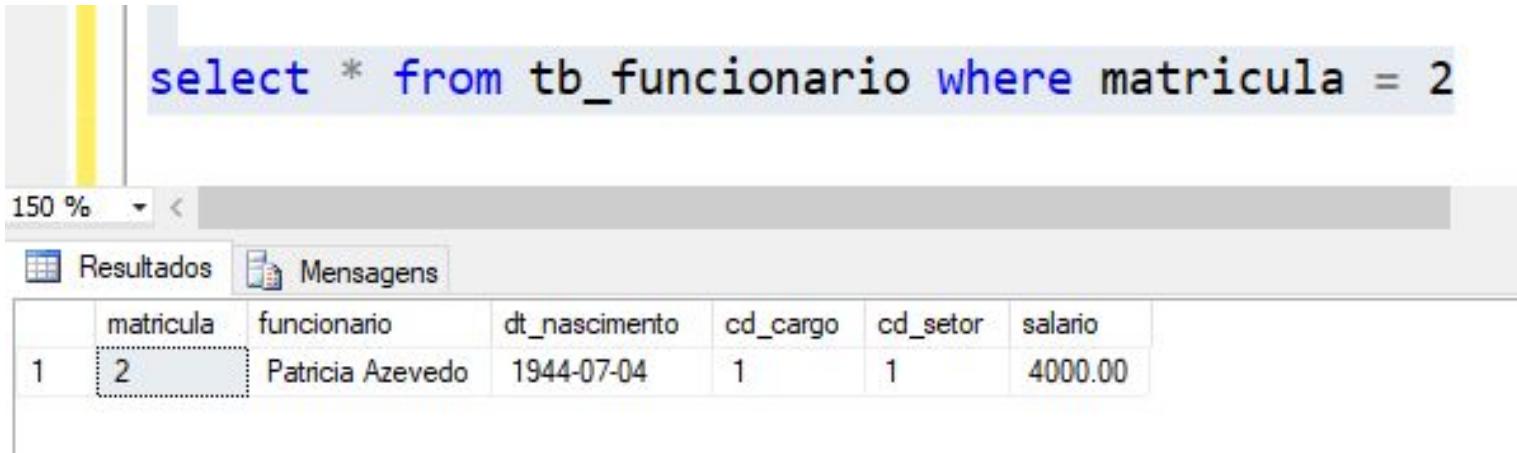
```
select * from tb_funcionario where funcionario = 'Ana Clara'
```

The interface includes a zoom level indicator (150%) and two tabs at the bottom: "Resultados" (selected) and "Mensagens". The results table shows one row of data:

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00

## Predicados

Um predicado pode ser:



A screenshot of a SQL query execution interface. The query entered is:

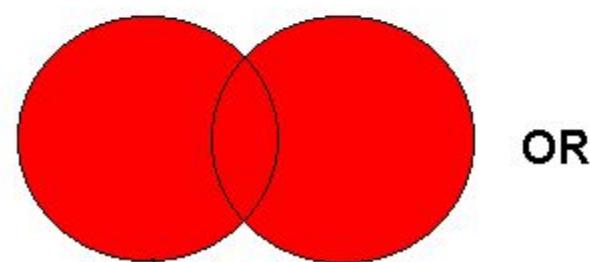
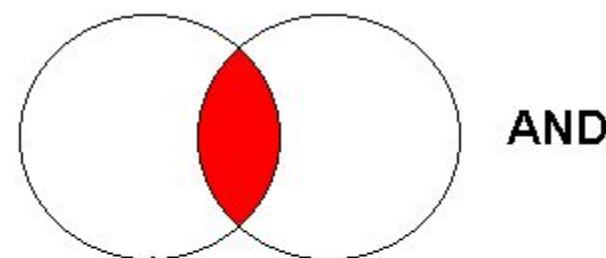
```
select * from tb_funcionario where matricula = 2
```

The results are displayed in a table:

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	2	Patricia Azevedo	1944-07-04	1	1	4000.00

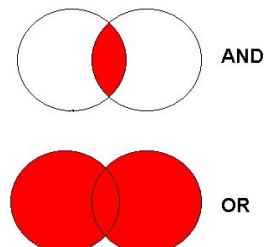
# Predicados

Um predicado pode ser: AND OR



## Predicados

Um predicado pode ser: AND OR



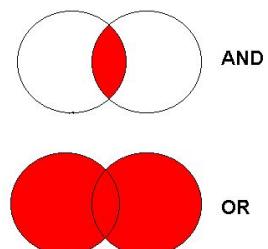
```
select * from tb_funcionario where cd_cargo = 1 and cd_setor = 1
```

150 % <

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	2	Patricia Azevedo	1944-07-04	1	1	4000.00

## Predicados

Um predicado pode ser: AND OR



```
select * from tb_funcionario where cd_cargo = 1 or cd_setor = 1
```

150 % < <

Resultados Mensagens

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00
2	2	Patricia Azevedo	1944-07-04	1	1	4000.00
3	3	Jose Maria	1971-05-10	1	3	6000.00
4	4	Sonia Abrantes	1979-05-29	1	4	7000.00

## Predicados

Um predicado pode ser: Between

```
select * from tb_funcionario where salario between 1000 and 6000
```

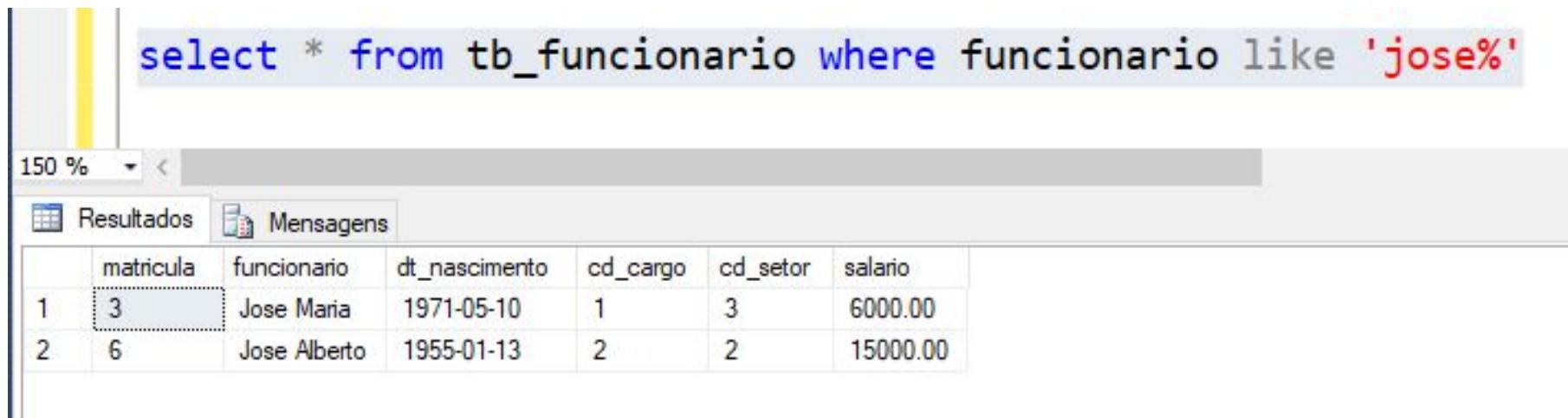
150 % <

Resultados Mensagens

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00
2	2	Patricia Azevedo	1944-07-04	1	1	4000.00
3	3	Jose Maria	1971-05-10	1	3	6000.00

## Predicados

Um predicado pode ser: LIKE



A screenshot of a SQL query results window. The query is:

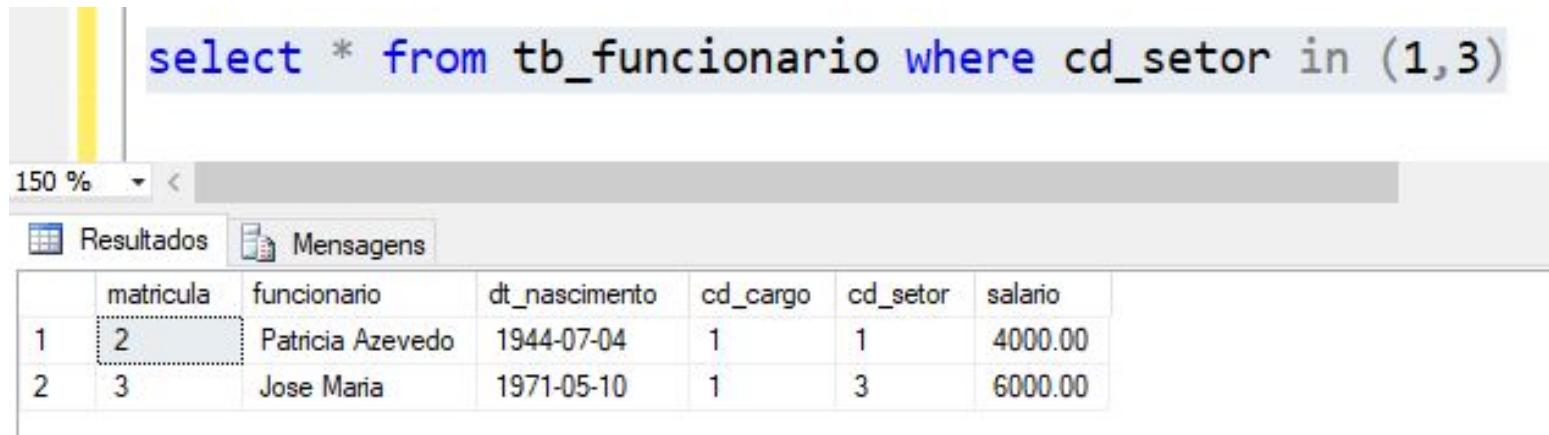
```
select * from tb_funcionario where funcionario like 'jose%'
```

The results show two rows of data:

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	3	Jose Maria	1971-05-10	1	3	6000.00
2	6	Jose Alberto	1955-01-13	2	2	15000.00

## Predicados

Um predicado pode ser: IN (1,3)



A screenshot of a SQL query execution interface. The query entered is:

```
select * from tb_funcionario where cd_setor in (1,3)
```

The results are displayed in a table:

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	2	Patricia Azevedo	1944-07-04	1	1	4000.00
2	3	Jose Maria	1971-05-10	1	3	6000.00

## Clausula ORDER BY

```
select * from tb_funcionario order by funcionario
```

150 %

Resultados Mensagens

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00
2	6	Jose Alberto	1955-01-13	2	2	15000.00
3	3	Jose Maria	1971-05-10	1	3	6000.00
4	2	Patricia Azevedo	1944-07-04	1	1	4000.00
5	4	Sonia Abrantes	1979-05-29	1	4	7000.00
6	5	Valdir Reinaldo	1960-09-22	2	2	16000.00

## Clausula ORDER BY DESC

```
select * from tb_funcionario order by funcionario desc
```

150 % < Resultados Mensagens

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario	
1	5	Valdir Reinaldo	1960-09-22	2	2	16000.00	
2	4	Sonia Abrantes	1979-05-29	1	4	7000.00	
3	2	Patricia Azevedo	1944-07-04	1	1	4000.00	
4	3	Jose Maria	1971-05-10	1	3	6000.00	
5	6	Jose Alberto	1955-01-13	2	2	15000.00	
6	1	Ana Clara	1977-07-05	1	5	3000.00	

## 2 Campos - Cláusula ORDER BY

```
select * from tb_funcionario order by cd_cargo, funcionario
```

150 %

Resultados

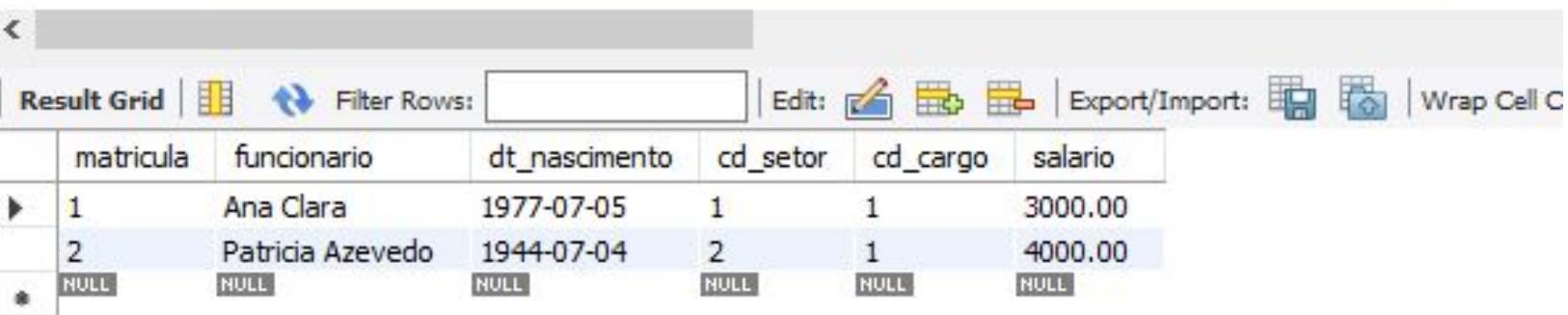
Mensagens

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00
2	3	Jose Maria	1971-05-10	1	3	6000.00
3	2	Patricia Azevedo	1944-07-04	1	1	4000.00
4	4	Sonia Abrantes	1979-05-29	1	4	7000.00
5	6	Jose Alberto	1955-01-13	2	2	15000.00
6	5	Valdir Reinaldo	1960-09-22	2	2	16000.00

**Limit** – limita as linhas retornadas em um conjunto de resultados de consulta a um número de linhas no SQL.

<

3 • `select * from tb_funcionario limit 2;`



	matricula	funcionario	dt_nascimento	cd_setor	cd_cargo	salario
▶	1	Ana Clara	1977-07-05	1	1	3000.00
▶	2	Patricia Azevedo	1944-07-04	2	1	4000.00
*	NULL	NULL	NULL	NULL	NULL	NULL

Top - Limita as linhas retornadas em um conjunto de resultados de consulta a um número ou percentual de linhas no SQL

## MS SQL Server – Oracle – DB2



A screenshot of a SQL query results window. The query entered is:

```
select top 2 * from tb_funcionario
```

The results show two rows of data from the tb\_funcionario table:

	matricula	funcionario	dt_nascimento	cd_cargo	cd_setor	salario
1	1	Ana Clara	1977-07-05	1	5	3000.00
2	2	Patricia Azevedo	1944-07-04	1	1	4000.00

**distinct** - Retorna todos os possíveis estados para a coluna selecionada no modelo.

Quais são os cargos que a minha empresa tem?

```
select cd_cargo from tb_funcionario
```

150 % <

Resultados Mensagens

cd_cargo
1
1
1
1
2
2

```
select distinct cd_cargo from tb_funcionario
```

150 % <

Resultados Mensagens

cd_cargo
1
2

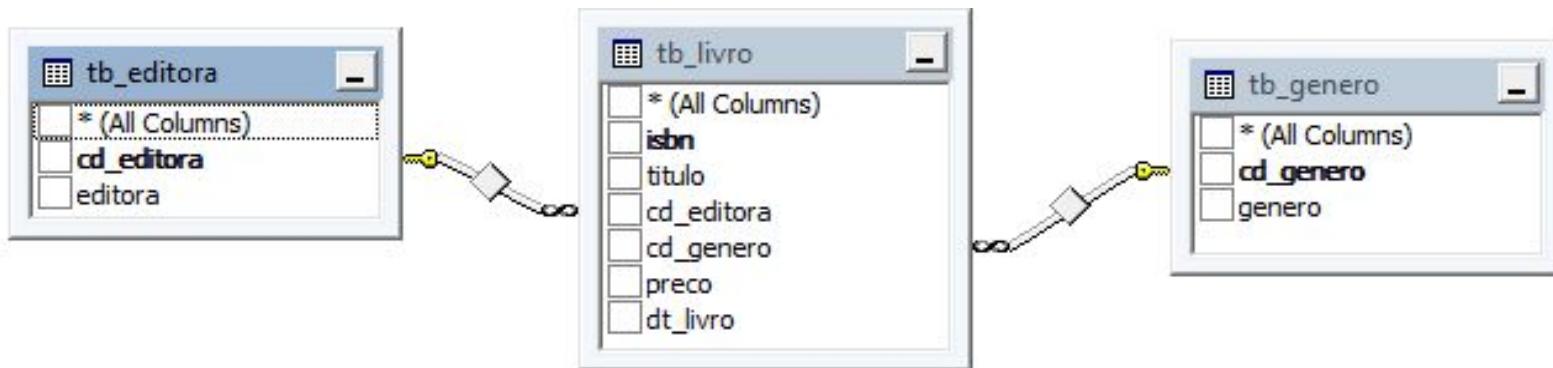
# Exercício

## exercício

- 1) Listar a cd\_setor = 2 na tabela tb\_funcionário
- 2) Alterar na tabela tb\_funcionario matricula = 1 salario (6000)
- 3) Lista os funcionários que termina com a letra a

# Inner Join

# Exemplo de Modelo de Dados



Uma cláusula **join da SQL** - correspondente a uma operação de **junção em álgebra relacional** - combina colunas de uma ou mais tabelas em um banco de dados relacional.

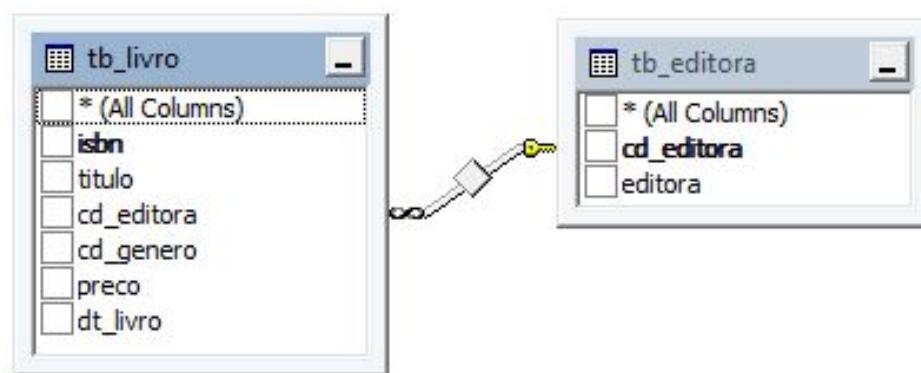
Um JOIN é um meio de combinar colunas de uma (auto-junção) ou mais tabelas, usando valores comuns a cada uma delas. O SQL padrão ANSI especifica cinco tipos de JOIN: INNER, LEFT OUTER, RIGHT OUTER.

# INNER JOIN

Com o INNER JOIN serão incluídas somente as linhas que satisfazem a condição do join.

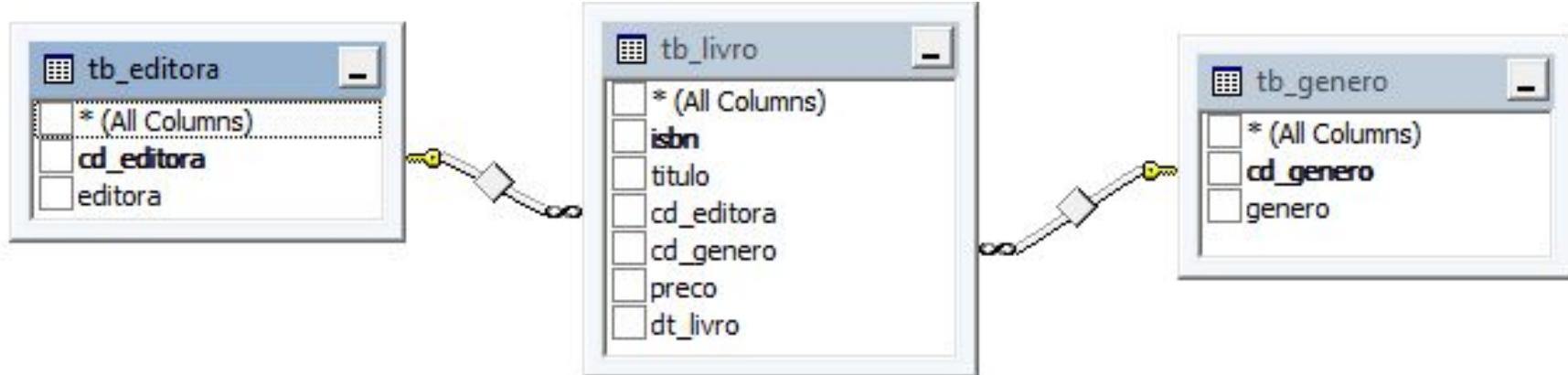
Problema: Ver os **Títulos** e **Editora** equivalente.

Diagrama gráfico:



	titulo	editora
1	engenharia de software	person
2	Algoritmos: Teoria e Prática	Pearson
3	Livro - Eletrônica Estudantes e Técnicos	Saraiva
4	Livro - Hardware: Versão Revisada e Atualizada	erica
5	Livro - Redes de Computadores	person
6	Livro - Guyton & Hall - Tratado de Fisiologia	erica
7	engenharia de Requisitos	person

# 1 - Sintaxe



**select L.titulo, E.editora from**

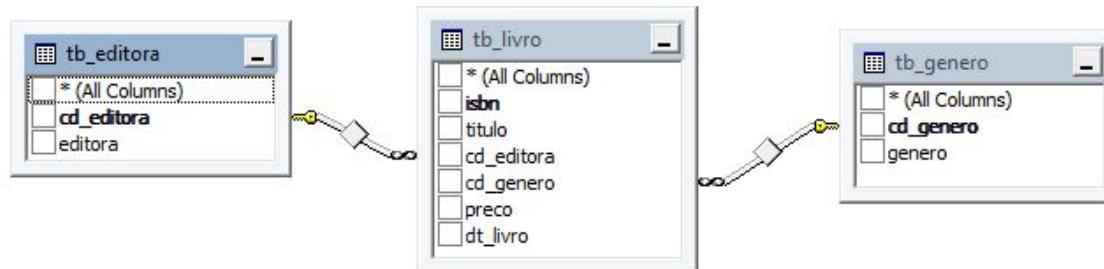
**tb\_livro L**

**inner join tb\_editora E**

**on L.cd\_editora = E.cd\_editora**

	titulo	editora
1	engenharia de software	person
2	Algoritmos: Teoria e Prática	Pearson
3	Livro - Eletrônica Estudantes e Técnicos	Saraiva
4	Livro - Hardware: Versão Revisada e Atualizada	erica
5	Livro - Redes de Computadores	person
6	Livro - Guyton & Hall - Tratado de Fisiologia	erica
7	engenharia de Requisitos	person

## 2 - Sintaxe



**select L.titulo, E.editora, G.genero      from tb\_livro L**

**inner join tb\_editora E**

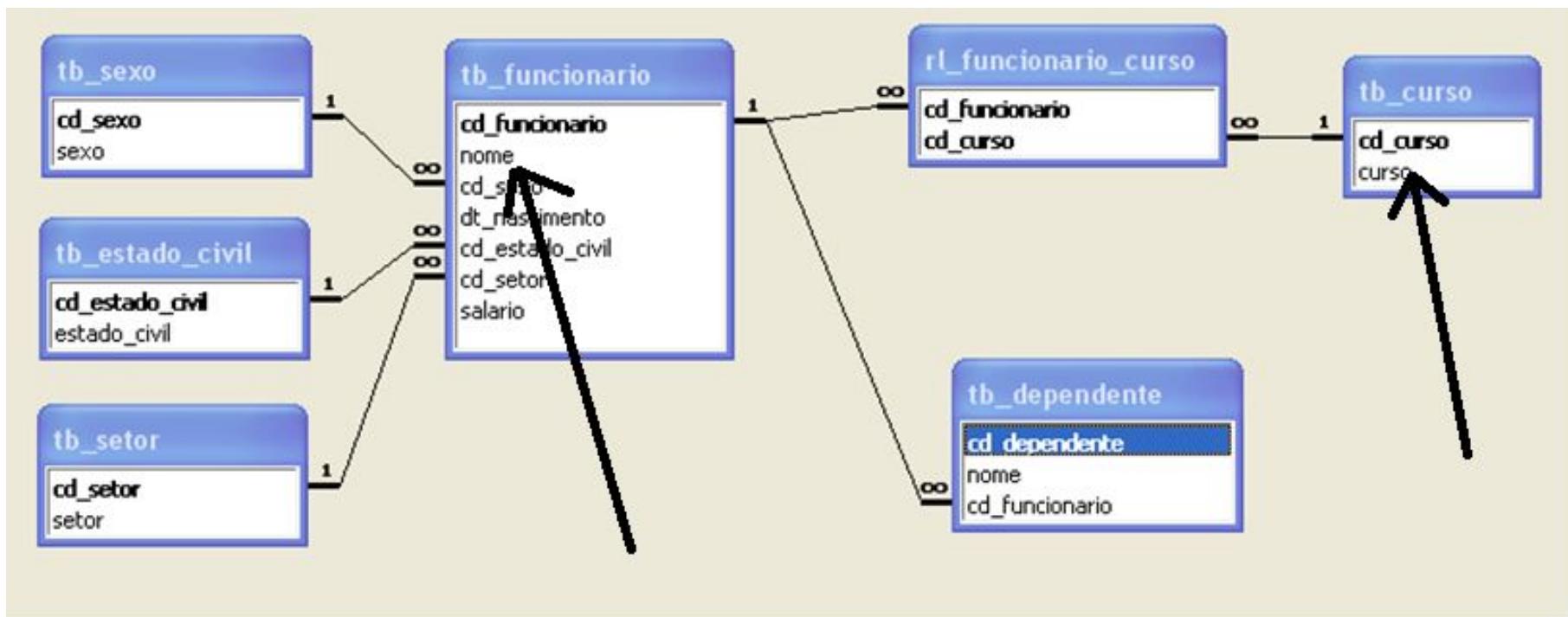
**on L.cd\_editora = E.cd\_editora**

**inner join tb\_genero G**

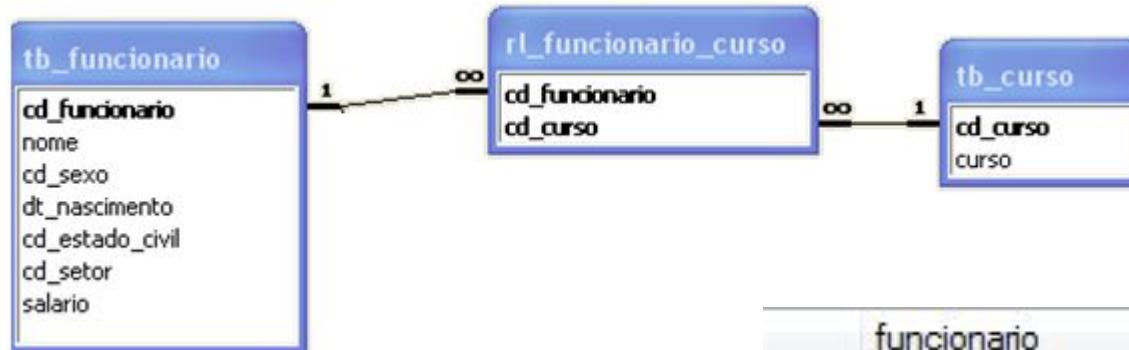
**on L.cd\_genero = G.cd\_genero**

	titulo	editora	genero
1	engenharia de software	person	Informatica
2	Algoritmos: Teoria e Prática	Pearson	Direito
3	Livro - Eletrônica Estudantes e Técnicos	Saraiva	Engenharia Civil
4	Livro - Hardware: Versão Revisada e Atualizada	erica	Engenharia Elétrica
5	Livro - Redes de Computadores	person	Enfermagem
6	Livro - Guyton & Hall - Tratado de Fisiologia	erica	Direito
7	engenharia de Requisitos	person	Enfermagem

# Sintaxe



## 6- Sintaxe



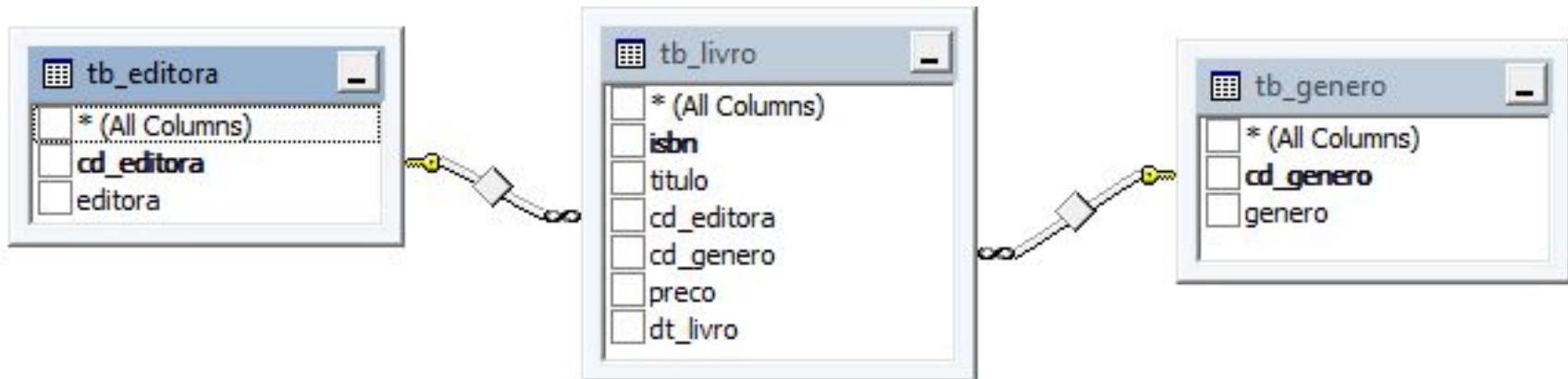
```

select F.nome, C.CURSO
from tb_funcionario F
inner join
rl_funcionario_curso RL
on F.cd_funcionario = RL.cd_funcionario
inner join tb_curso C
on RL.cd_curso = C.cd_curso
    
```

	funcionario	CURSO
1	Ana Clara	Medicina
2	Ana Clara	Direito
3	Patricia Azevedo	Medicina
4	Jose Maria	Computacao
5	Sonia Abrantes	Engenharia
6	Valdir Reinaldo	Computacao
7	Valdir Reinaldo	Contabilidade
8	Jose Alberto	Administracao
9	Jose Pereira	Computacao
10	Edson Nogueira	Computacao
11	Paula Nobrega	Engenharia

# Exercício

# Exercício

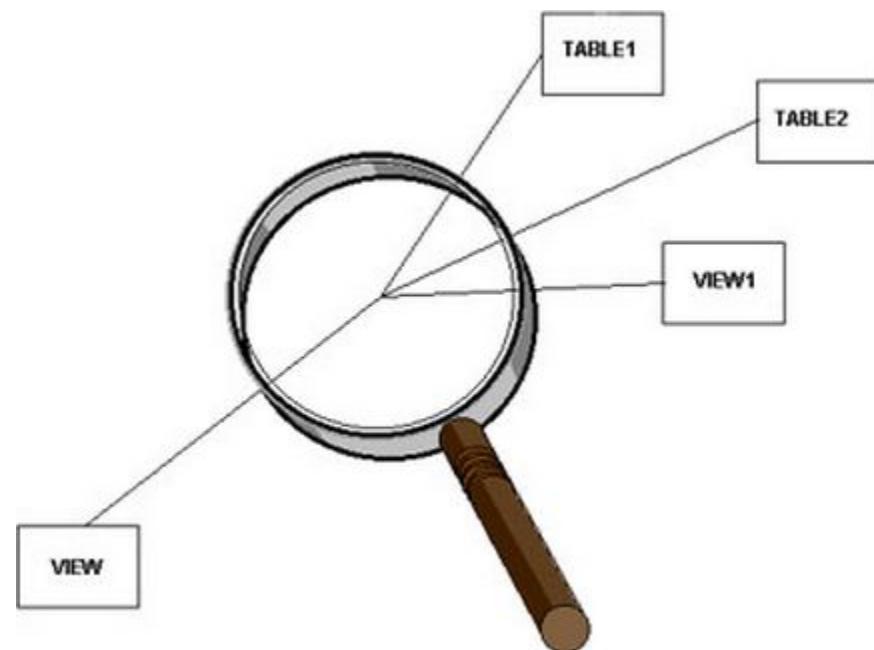


- 1 Lista titulo, editora quando o preco maior que R\$200,00
- 2 Lista titulo, gênero quando o preco entre R\$200,00 e R\$300,00

# View

# O que é uma view ?

- Uma **view** é uma maneira alternativa de observação de dados de uma ou mais **entidades (tabelas)**, que compõem uma base de dados.
  
- Pode ser considerada como uma **tabela virtual** ou uma consulta armazenada.



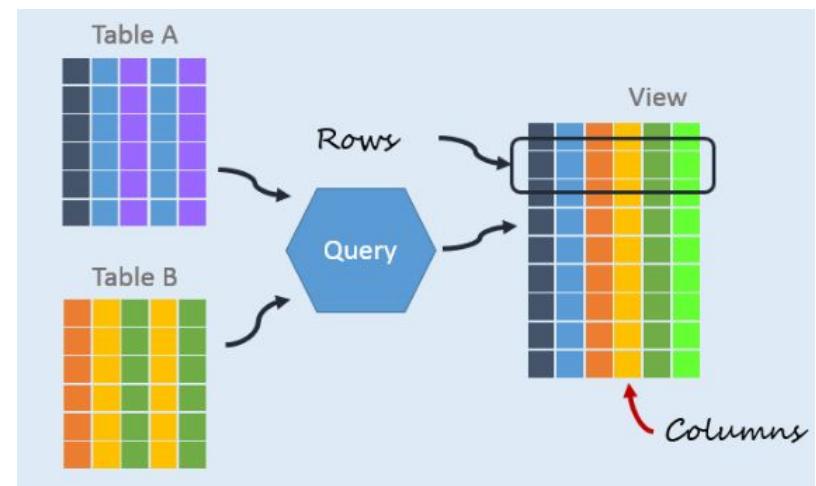
# Onde se aplicam as views?

# Onde se aplicam as views?

Geralmente e **recomendável**, uma view, implementada encapsulando uma instrução **SELECT** (busca de dados para exposição).

Uma **tabela virtual**.

Por este motivo, pode ser mais rápido ter uma consulta armazenada em forma de view, em vez de ter que retrabalhar uma instrução.



# Onde aplicamos?

**Restrição usuário x dados;**

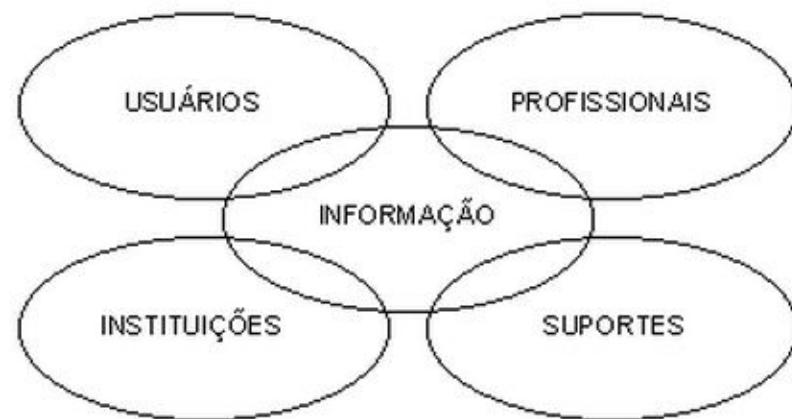
**Ex.:** Seu departamento de vendas **não precisa saber** ou ter **acesso** a uma coluna que contém valores (dados) referentes aos **salários** dos desenvolvedores.



# Onde aplicamos?

## - Restrição usuário x domínio;

**Ex.:** Podemos restringir o acesso de um **usuário** específico a **colunas** (domínios) específicas (os) de uma tabela.



# Onde aplicamos?



- Associar vários domínios formando uma única entidade;

Ex.: Podemos ter várias "**JOIN**" encapsuladas em uma view, formando somente uma tabela arbitrariamente.

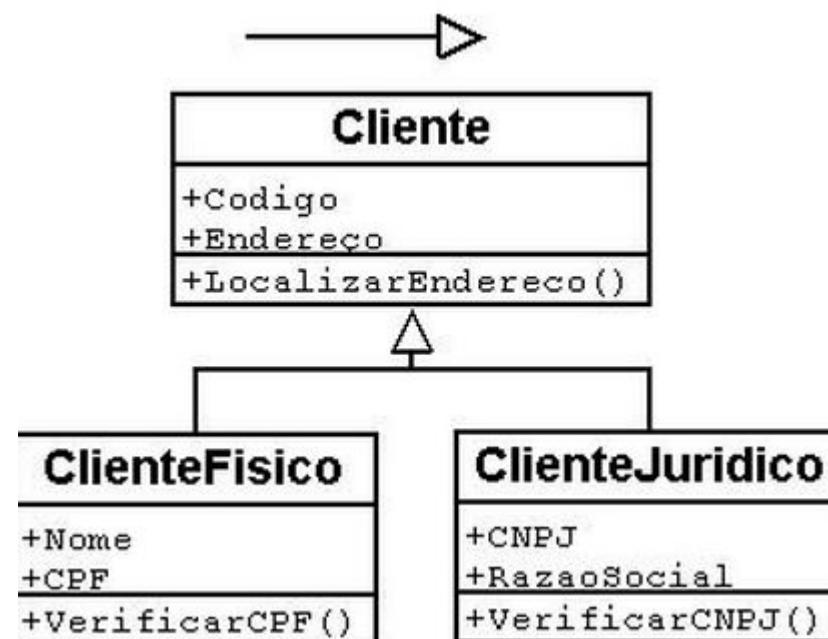


**Qual as vantagens de se usar views ?**

# Qual as vantagens de se usar views ?

**Economizar tempo com retrabalho;**

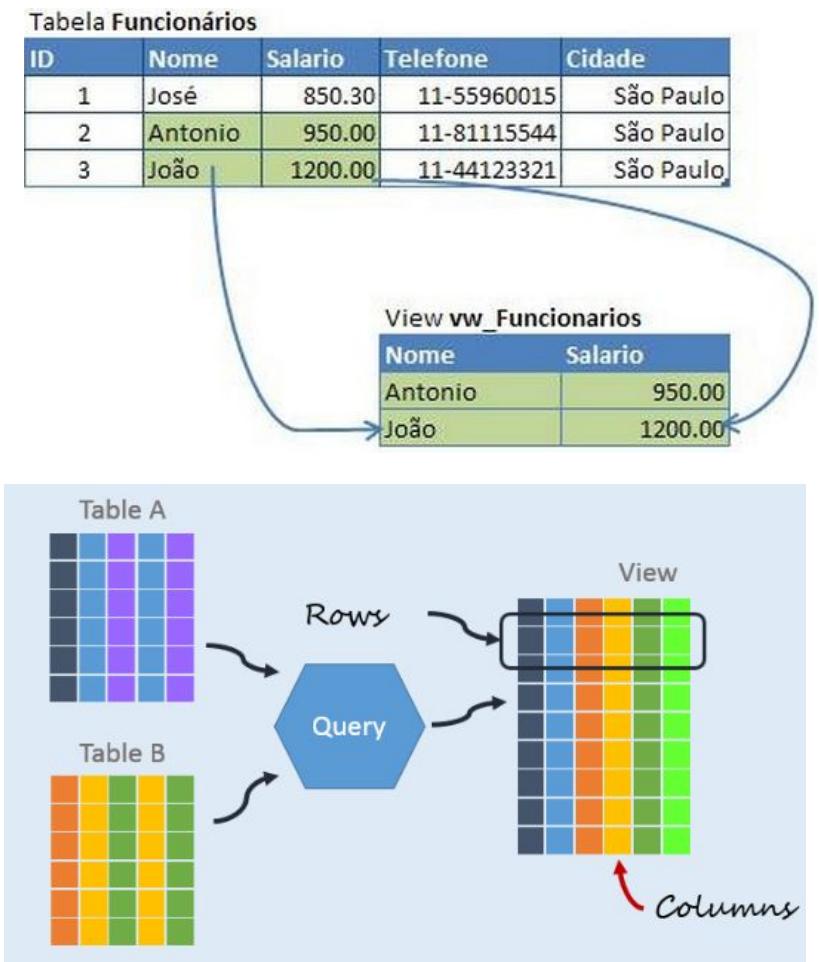
**Ex.: Você não precisar escrever** aquela instrução enorme.  
Escreva uma vez e armazene!



# Qual as vantagens de se usar views ?

- Velocidade de acesso às informações;

**Ex.:** Uma vez **compilada**, o seu **recordset** (conjunto de dados) é armazenado em uma tabela temporária (virtual).



# Qual as vantagens de se usar views ?

Mascarar complexidade do banco de dados;

Ex.: As **views isolam do usuário a complexidade do banco de dados.**

Nomes de domínios podem ser referenciados com **literais** e outros recursos.

Isso proporciona aos desenvolvedores a capacidade de alterar a estrutura sem afetar a interação do usuário com o banco de dados.

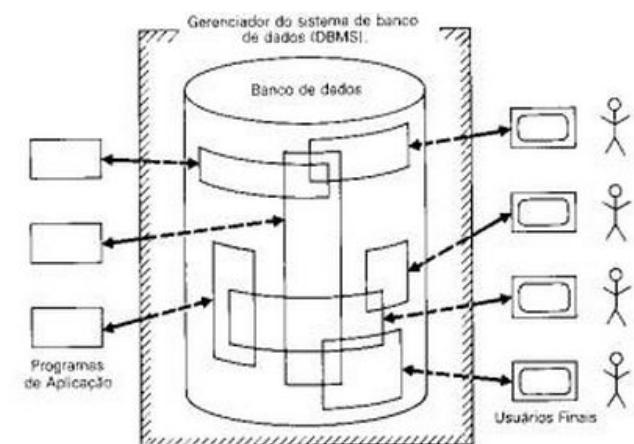


# Qual as vantagens de se usar views ?

Simplifica o gerenciamento de permissão de usuários;

**Ex.:** Em vez de conceder **permissão** para que os usuários contem tabelas base, os proprietários de bancos de dados podem conceder permissões para que os usuários consultem dados somente através de views.

Isso também **protege as alterações** na estrutura das tabelas base subjacentes. Os usuários não serão interrompidos durante uma visualização de dados.



# Criando views

# Criando views



3

4 • `create view vw_01 as`  
 5   `select titulo, preco, dt_livro from tb_livro;`

6

7 • `select * from vw_01;`

Result Grid | Filter Rows: [ ] | Export

	titulo	preco	dt_livro
▶	banco de dados	360.00	2019-10-02
	Engenharia de Software	420.00	2019-10-03
	Ortopedia	372.00	2019-10-04
	Cardiologia	384.00	2019-10-05
	Estrutura Predial	240.00	2019-10-06
	Estrutura Hidráulica	360.00	2019-10-07
	Direito Penal	180.00	2019-10-08
	Direito Civil	240.00	2019-10-09
	Cores	240.00	2019-10-10
	Paisagismo	300.00	2019-10-11
	Virus	360.00	2019-10-12
	Bacteria	360.00	2019-10-12

# Criando views



```

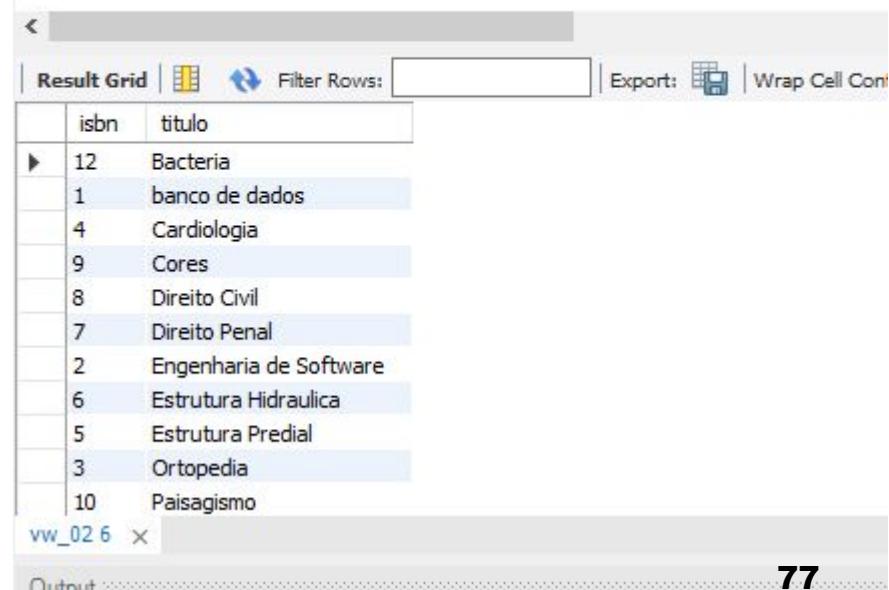
8
9 • create view vw_02 as
10 select isbn, titulo from tb_livro;
11

```

```

11
12 • select * from vw_02 order by titulo;
13
14

```



	isbn	titulo
▶	12	Bacteriologia
	1	banco de dados
	4	Cardiologia
	9	Cores
	8	Direito Civil
	7	Direito Penal
	2	Engenharia de Software
	6	Estrutura Hidráulica
	5	Estrutura Predial
	3	Ortopedia
	10	Paisagismo

vw\_02 6 x

Output ::::

# Criando views



```

13
14 • create view vw_03 as
15   select titulo, editora from tb_livro as L
16   inner join tb_editora as E
17   on L.cd_editora = E.cd_editora;
  
```

```

18
19 • select * from vw_03 order by titulo;
  
```

Result Grid | Filter Rows:  | Export: | Wrap Cell Content

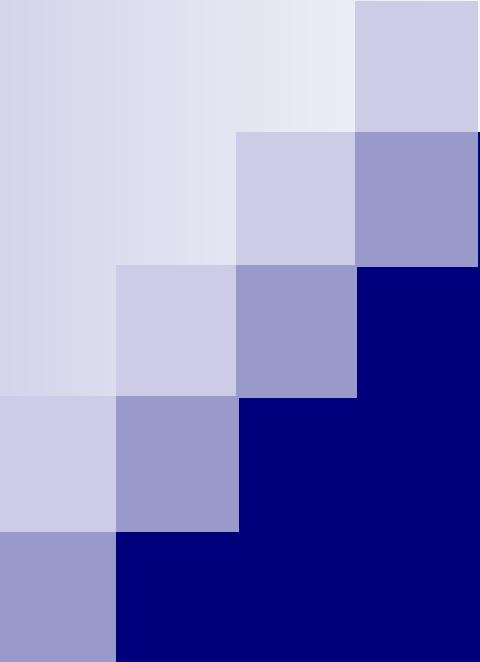
	titulo	editora
▶	Bateria	Dickens
	banco de dados	novatec
	Cardiologia	livraria florence
	Cores	saraiva
	Direito Civil	Editora Forum
	Direito Penal	saraiva
	Engenharia de Software	novatec
	Estrutura Hidráulica	Mundial
	Estrutura Predial	blucher
	Ortopedia	coopmed
	Paisagismo	Editora Forum

vw\_03 7 x

# Exercício

# Exercício

- 1 Criar uma View – liste isbn, titulo
- 2 Criar uma View – list isbn, titulo, preco, preco com 10%



Outer Join

Right e Left

# Outer join



--Lista as Editora que não livro na biblioteca

```
select e.editora, l.titulo from
tb_livro l inner join tb_editora e
on e.cd_editora = l.cd_editora
```

	editora	titulo
1	person	engenharia de software
2	Pearson	Algoritmos: Teoria e Prática
3	Saraiva	Livro - Eletrônica Estudantes e Técnicos
4	erica	Livro - Hardware: Versão Revisada e Atualizada
5	person	Livro - Redes de Computadores
6	erica	Livro - Guyton & Hall - Tratado de Fisiologia
7	person	engenharia de Requisitos

# Outer join



■--Lista as Editora que não livro na biblioteca

```

select e.editora, l.titulo from
tb_livro l right outer join tb_editora e
on e.cd_editora = l.cd_editora
where l.titulo is null
  
```

	editora	titulo
1	Editora FTD	NULL
2	ABRIL	NULL
3	ATICA	NULL
4	NOVA FRONTEIRA	NULL

# Outer join



----- Lista os Generos que não tem na biblioteca

```
select g.genero, l.titulo from
tb_genero g inner join tb_livro l
on l.cd_genero = g.cd_genero
```

	genero	titulo
1	Informatica	engenharia de software
2	Direito	Algoritmos: Teoria e Prática
3	Engenharia Civil	Livro - Eletrônica Estudantes e Técnicos
4	Engenharia Elétrica	Livro - Hardware: Versão Revisada e Atualizada
5	Enfermagem	Livro - Redes de Computadores
6	Direito	Livro - Guyton & Hall - Tratado de Fisiologia
7	Enfermagem	engenharia de Requisitos

# Outer join



----- Lista os Generos que não tem na biblioteca

```

select g.genero, l.titulo from
tb_genero g left outer join tb_livro l
on l.cd_genero = g.cd_genero
where l.titulo is null
    
```

	genero	titulo
1	Direito	NULL

# Exercício

# Exercício

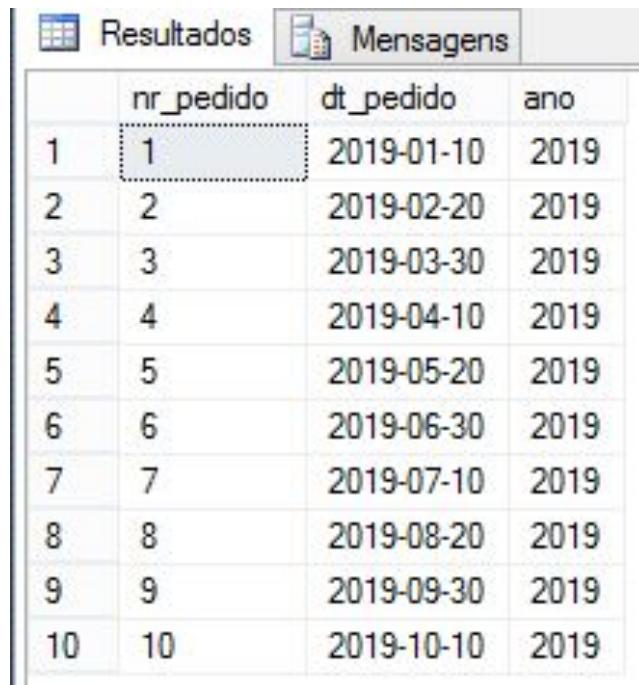
- 1 Lista os cargos que não tem funcionário
- 2 Lista o setor que não tem funcionário

# Manipulação de Data

# Manipulação Data

- Year - recebe o ano      `year(dt_nascimento)`

```
select nr_pedido, dt_pedido, year(dt_pedido) ano from tb_pedido
```



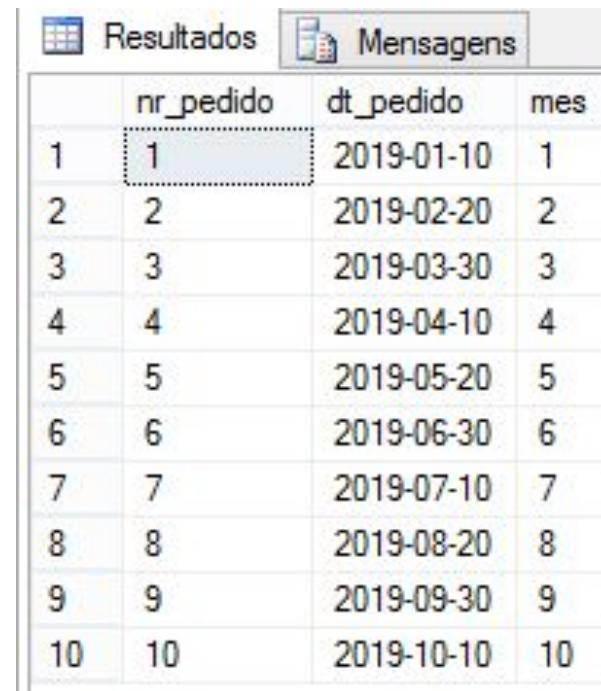
The screenshot shows a database query results window with two tabs: "Resultados" and "Mensagens". The "Resultados" tab is selected, displaying a table with four columns: nr\_pedido, dt\_pedido, and ano. The table contains 10 rows, each with a value for nr\_pedido (1 to 10), a date value for dt\_pedido (from 2019-01-10 to 2019-10-10), and the year 2019 for the ano column.

	nr_pedido	dt_pedido	ano
1	1	2019-01-10	2019
2	2	2019-02-20	2019
3	3	2019-03-30	2019
4	4	2019-04-10	2019
5	5	2019-05-20	2019
6	6	2019-06-30	2019
7	7	2019-07-10	2019
8	8	2019-08-20	2019
9	9	2019-09-30	2019
10	10	2019-10-10	2019

# Manipulação Data

- Month - recebe o mês      Month(dt\_nascimento)

```
select nr_pedido, dt_pedido, month(dt_pedido) mes from tb_pedido
```

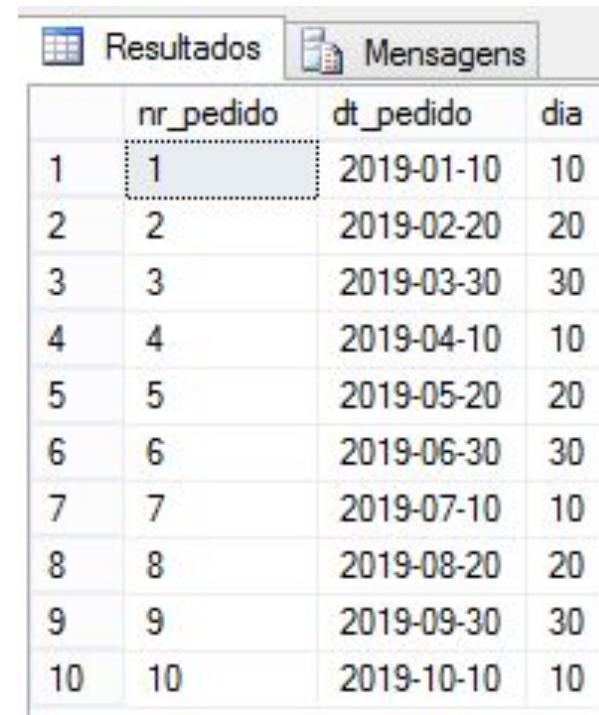


	nr_pedido	dt_pedido	mes
1	1	2019-01-10	1
2	2	2019-02-20	2
3	3	2019-03-30	3
4	4	2019-04-10	4
5	5	2019-05-20	5
6	6	2019-06-30	6
7	7	2019-07-10	7
8	8	2019-08-20	8
9	9	2019-09-30	9
10	10	2019-10-10	10

# Manipulação Data

- Day - recebe o dia      `day(dt_nascimento)`

```
select nr_pedido, dt_pedido, day(dt_pedido) dia from tb_pedido
```



The screenshot shows the 'Resultados' tab of a SQL Server Management Studio window. It displays a table with four columns: 'nr\_pedido', 'dt\_pedido', 'dia', and a primary key column which is not explicitly labeled but corresponds to the 'nr\_pedido' column. The data consists of 10 rows, each representing a different date in the 'dt\_pedido' column and its corresponding day value in the 'dia' column.

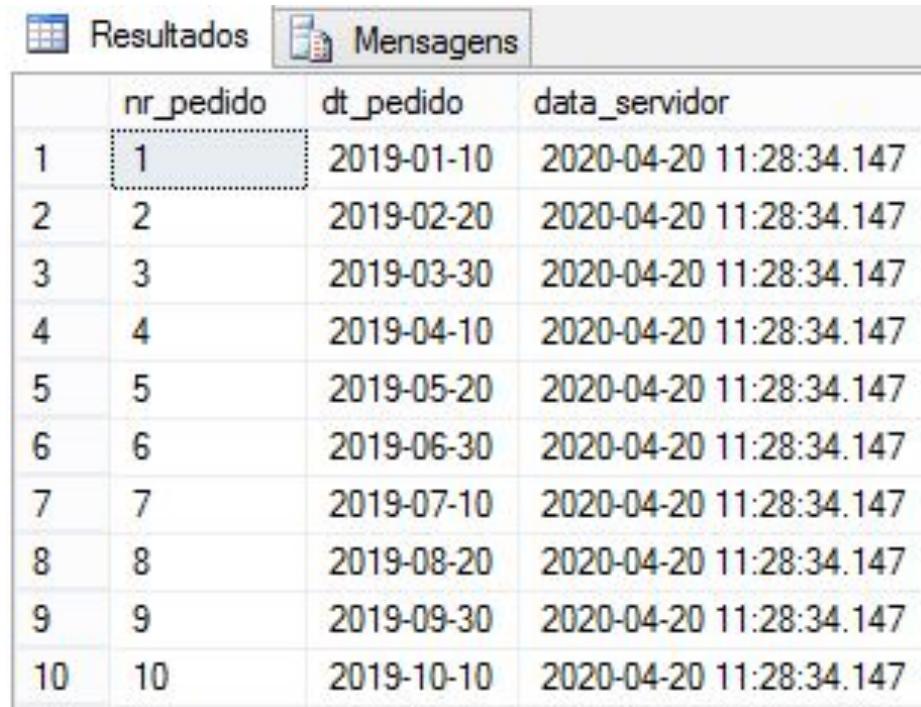
	nr_pedido	dt_pedido	dia
1	1	2019-01-10	10
2	2	2019-02-20	20
3	3	2019-03-30	30
4	4	2019-04-10	10
5	5	2019-05-20	20
6	6	2019-06-30	30
7	7	2019-07-10	10
8	8	2019-08-20	20
9	9	2019-09-30	30
10	10	2019-10-10	10

# Manipulação Data

- curdate() – recebe a data do Servidor

```
select nr_pedido, dt_pedido, curdate() data_servidor from tb_pedido
```

```
select nr_pedido, dt_pedido, now() data_servidor from tb_pedido
```



	nr_pedido	dt_pedido	data_servidor
1	1	2019-01-10	2020-04-20 11:28:34.147
2	2	2019-02-20	2020-04-20 11:28:34.147
3	3	2019-03-30	2020-04-20 11:28:34.147
4	4	2019-04-10	2020-04-20 11:28:34.147
5	5	2019-05-20	2020-04-20 11:28:34.147
6	6	2019-06-30	2020-04-20 11:28:34.147
7	7	2019-07-10	2020-04-20 11:28:34.147
8	8	2019-08-20	2020-04-20 11:28:34.147
9	9	2019-09-30	2020-04-20 11:28:34.147
10	10	2019-10-10	2020-04-20 11:28:34.147

# Manipulação Data

- Datediff - verifica diferença de data
- Datediff(now() , dt\_nascimento)

```
select nr_pedido, dt_pedido, now() data_servidor,
datediff(now(), dt_pedido,) dia_compra from tb_pedido
```

Resultados

	nr_pedido	dt_pedido	data_servidor	dia_compra
1	1	2019-01-10	2020-04-20 11:37:53.360	466
2	2	2019-02-20	2020-04-20 11:37:53.360	425
3	3	2019-03-30	2020-04-20 11:37:53.360	387
4	4	2019-04-10	2020-04-20 11:37:53.360	376
5	5	2019-05-20	2020-04-20 11:37:53.360	336
6	6	2019-06-30	2020-04-20 11:37:53.360	295
7	7	2019-07-10	2020-04-20 11:37:53.360	285
8	8	2019-08-20	2020-04-20 11:37:53.360	244
9	9	2019-09-30	2020-04-20 11:37:53.360	203
10	10	2019-10-10	2020-04-20 11:37:53.360	193

# Manipulação Data

- Datediff - verifica diferença de data
- Datediff(now() , dt\_nascimento) / 365

```
select nr_pedido, dt_pedido, now() data_servidor,  
datediff(now() ,dt_pedido) / 365 Qtd_anos_passaram from tb_pedido
```

	nr_pedido	dt_pedido	data_servidor	Qtd_anos_passaram
▶	1	2019-01-10	2021-05-16 19:29:26	2.3479
	2	2019-02-20	2021-05-16 19:29:26	2.2356
	3	2019-03-30	2021-05-16 19:29:26	2.1315
	4	2019-04-10	2021-05-16 19:29:26	2.1014
	5	2019-05-20	2021-05-16 19:29:26	1.9918
	6	2019-06-30	2021-05-16 19:29:26	1.8795
	7	2019-07-10	2021-05-16 19:29:26	1.8521
	8	2019-08-20	2021-05-16 19:29:26	1.7397
	9	2019-09-30	2021-05-16 19:29:26	1.6274
	10	2019-10-10	2021-05-16 19:29:26	1.6000

# Manipulação Data

- Datediff - verifica diferença de data

```
round(datediff(now(), dt_pedido) / 365, 0)
```

```
select nr_pedido, dt_pedido, now() data_servidor,  
round(datediff(now(), dt_pedido) / 365, 0) Qtd_anos_passaram from  
tb_pedido;
```

	nr_pedido	dt_pedido	data_servidor	Qtd_anos_passaram
▶	1	2019-01-10	2021-05-16 19:31:29	2
	2	2019-02-20	2021-05-16 19:31:29	2
	3	2019-03-30	2021-05-16 19:31:29	2
	4	2019-04-10	2021-05-16 19:31:29	2
	5	2019-05-20	2021-05-16 19:31:29	2
	6	2019-06-30	2021-05-16 19:31:29	2
	7	2019-07-10	2021-05-16 19:31:29	2
	8	2019-08-20	2021-05-16 19:31:29	2
	9	2019-09-30	2021-05-16 19:31:29	2
	10	2019-10-10	2021-05-16 19:31:29	2

# Cálculo

# Expressões Aritméticas

Se a sua expressão aritmética conter mais de um operador a prioridade é “\*” e “/”, posteriormente a “+” e “-”.

```
select comissao, comissao+100 '+100', comissao*1.1 '10%',  
comissao*0.5 '-50%' from tb_pedido
```

	comissao	+100	10%	-50%
1	3744,00	3844,00	4118.40000	1872.00000
2	3528,00	3628,00	3880.80000	1764.00000
3	7848,00	7948,00	8632.80000	3924.00000
4	7272,00	7372,00	7999.20000	3636.00000
5	4968,00	5068,00	5464.80000	2484.00000
6	4536,00	4636,00	4989.60000	2268.00000
7	3744,00	3844,00	4118.40000	1872.00000
8	3744,00	3844,00	4118.40000	1872.00000
9	3744,00	3844,00	4118.40000	1872.00000
10	3744,00	3844,00	4118.40000	1872.00000

# Funções Numéricas

## SUM

SUM (N)

Retorna a soma de todos os campos N

```
select sum(comissao) soma_comissao from tb_pedido
```

	nr_pedido	comissao
1	1	3744,00
2	2	3528,00
3	3	7848,00
4	4	7272,00
5	5	4968,00
6	6	4536,00
7	7	3744,00
8	8	3744,00
9	9	3744,00
10	10	3744,00

	soma_comissao
1	46872,00

# Funções Numéricas

## AVG

AVG (N)

Retorna a média aritmética de todos os campos N

```
select avg(comissao) media_das_comissoes from tb_pedido
```

	nr_pedido	comissao
1	1	3744,00
2	2	3528,00
3	3	7848,00
4	4	7272,00
5	5	4968,00
6	6	4536,00
7	7	3744,00
8	8	3744,00
9	9	3744,00
10	10	3744,00

	media_das_comissoes
1	4687,20

# Funções Numéricas

## COUNT

COUNT (\*)

Retorna o número de linhas da consulta.

```
select count(*) Qtd_Reg from tb_pedido
```

Resultados		Mensagens
		Qtd_Reg
1	10	

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
1	1	2019-01-10	2	2	3	74880,00	3744,00
2	2	2019-02-20	2	3	4	70560,00	3528,00
3	3	2019-03-30	3	4	5	156960,00	7848,00
4	4	2019-04-10	4	5	1	145440,00	7272,00
5	5	2019-05-20	5	5	1	99360,00	4968,00
6	6	2019-06-30	6	2	2	90720,00	4536,00
7	7	2019-07-10	2	4	5	74880,00	3744,00
8	8	2019-08-20	2	4	5	74880,00	3744,00
9	9	2019-09-30	2	4	5	74880,00	3744,00
10	10	2019-10-10	2	4	5	74880,00	3744,00

# Funções Numéricas

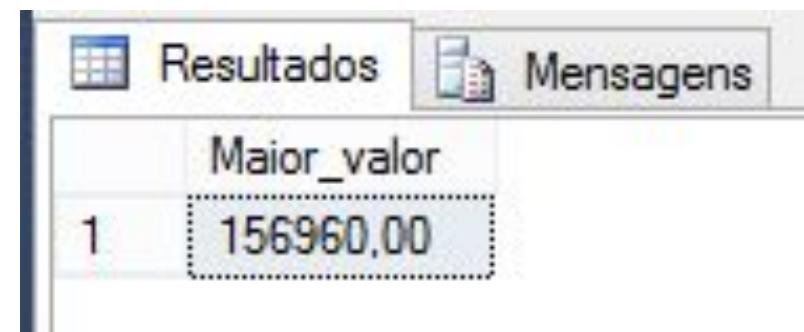
## MAX

MAX(preco)

Retorna o valor máximo de expr.

```
select max(valor_pedido) Maior_valor from tb_pedido
```

	nr_pedido	dt_pedido	valor_pedido
1	1	2019-01-10	74880,00
2	2	2019-02-20	70560,00
3	3	2019-03-30	156960,00
4	4	2019-04-10	145440,00
5	5	2019-05-20	99360,00
6	6	2019-06-30	90720,00
7	7	2019-07-10	74880,00
8	8	2019-08-20	74880,00
9	9	2019-09-30	74880,00
10	10	2019-10-10	74880,00



# Funções Numéricas

## MIN

MIN(preco)

Retorna o valor mínimo de expr

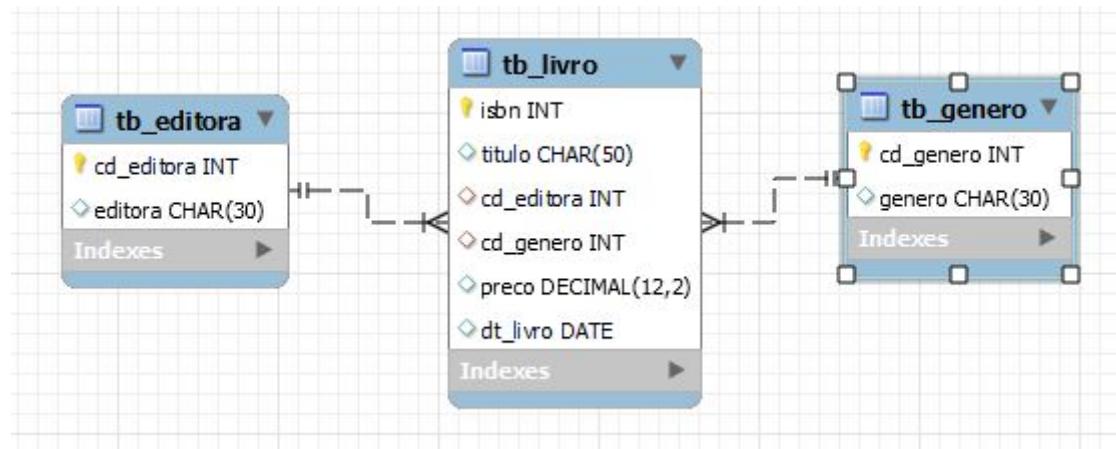
```
select min(valor_pedido) Menor_valor from tb_pedido
```

	nr_pedido	dt_pedido	valor_pedido
1	1	2019-01-10	74880,00
2	2	2019-02-20	70560,00
3	3	2019-03-30	156960,00
4	4	2019-04-10	145440,00
5	5	2019-05-20	99360,00
6	6	2019-06-30	90720,00
7	7	2019-07-10	74880,00
8	8	2019-08-20	74880,00
9	9	2019-09-30	74880,00
10	10	2019-10-10	74880,00

Resultados		Mensagens
	Menor_valor	
1	70560,00	

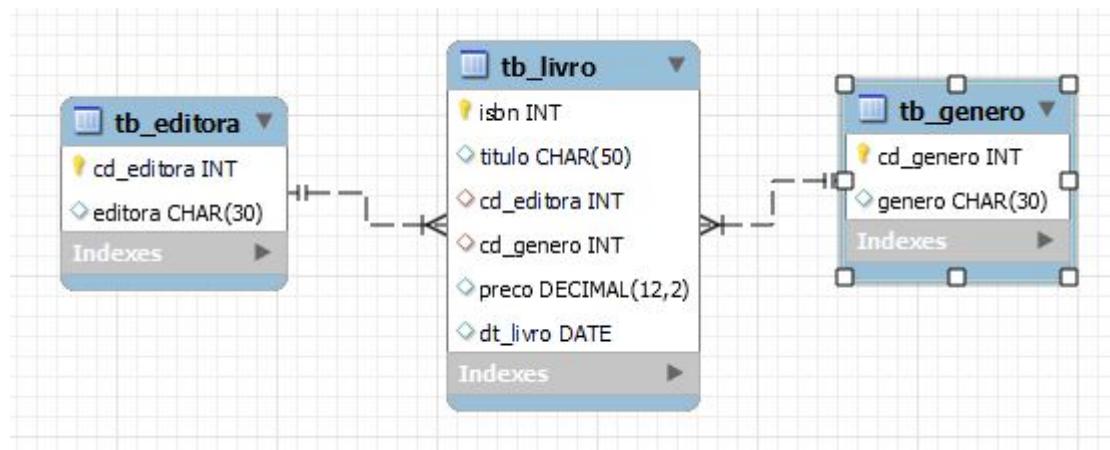
# Consulta de Agregação

# Consulta de Agregação

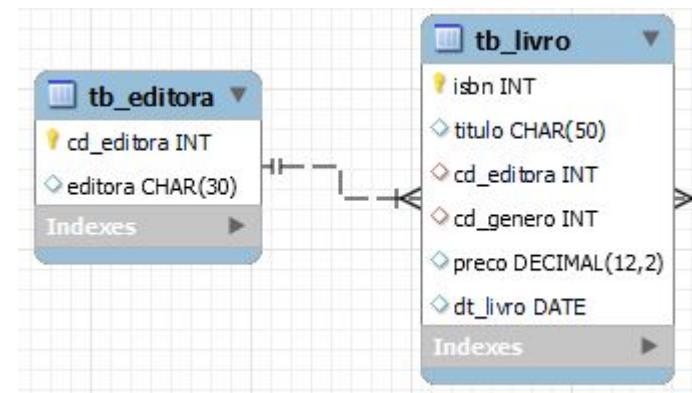


- As funções de agregação executam um cálculo **count** em um conjunto de valores e retornam um único valor.
- As funções de agregação utilizam a cláusula **GROUP BY** da instrução **SELECT**.

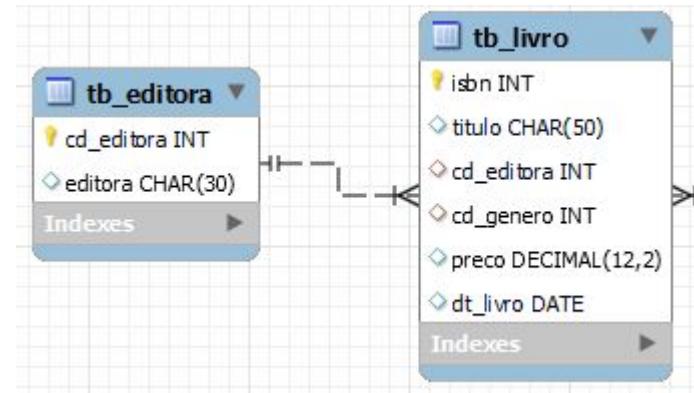
# Consulta de Agregação



- 1.A) Soma os valores dos livros – Qual a editora que a soma é maior?



# Consulta de Agregação

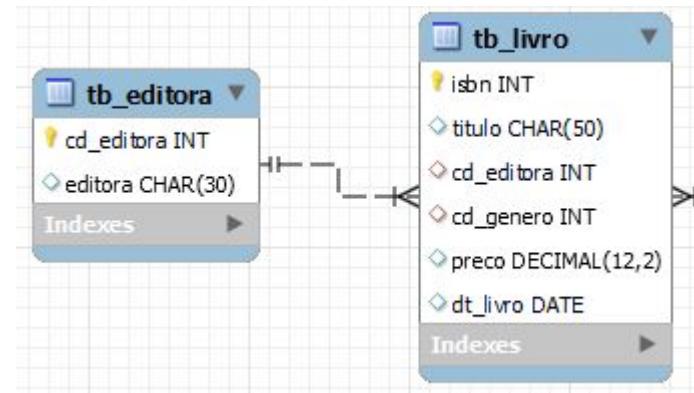


- 1.B) **Soma os valores dos livros – Qual a editora que a soma é maior?**

```
select E.editora, L.preco from tb_livro L
inner join tb_editora E
on L.cd_editora = E.cd_editora;
```

	editora	preco
▶	novatec	360.00
	novatec	420.00
	coopmed	372.00
	livraria florence	384.00
	blucher	240.00
	Mundial	360.00
	saraiva	180.00
	Editora Forum	240.00
	saraiva	240.00
	Editora Forum	300.00
	Dickens	360.00
	Dickens	360.00

# Consulta de Agregação



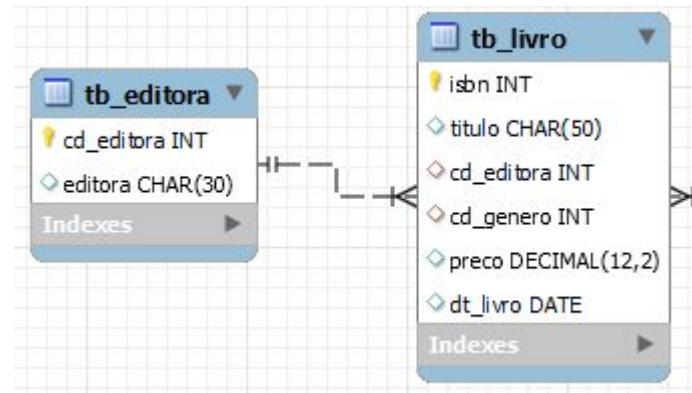
- 1.B) Soma os valores dos livros – Qual a editora que a soma é maior?

```

select E.editora, sum(L.preco) preco_total
from tb_livro L
inner join tb_editora E
on L.cd_editora = E.cd_editora
group by E.editora
order by preco_total desc;
  
```

	editora	preco_total
▶	novatec	780.00
	Dickens	720.00
	Editora Forum	540.00
	saraiva	420.00
	livraria florence	384.00
	coopmed	372.00
	Mundial	360.00
	blucher	240.00

# Consulta de Agregação



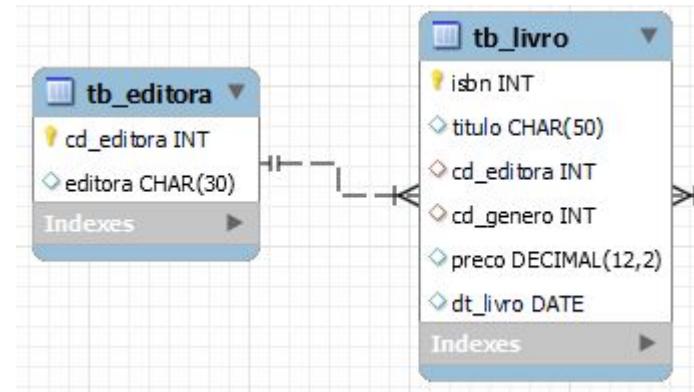
- 1.C) **Lista a quantidade de livros por editora?**

```

select E.editora, L.titulo from
tb_livro L
inner join tb_editora E
on L.cd_editora = E.cd_editora;
    
```

editora	titulo
novatec	banco de dados
novatec	Engenharia de Software
coopmed	Ortopedia
livraria florence	Cardiologia
blucher	Estrutura Predial
Mundial	Estrutura Hidráulica
saraiva	Direito Penal
Editora Forum	Direito Civil
saraiva	Cores
Editora Forum	Paisagismo
Dickens	Virus
Dickens	Bacteria

# Consulta de Agregação

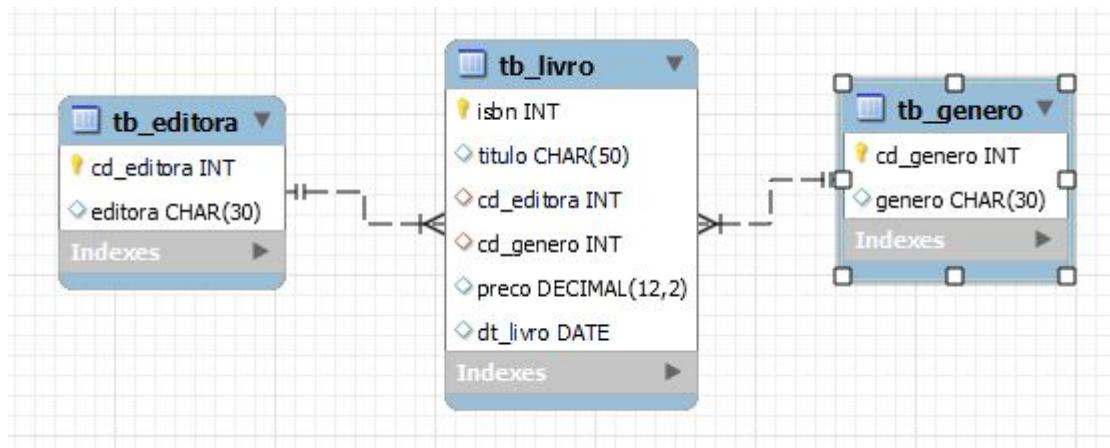


- 1.C) **Lista a quantidade de livros por editora?**

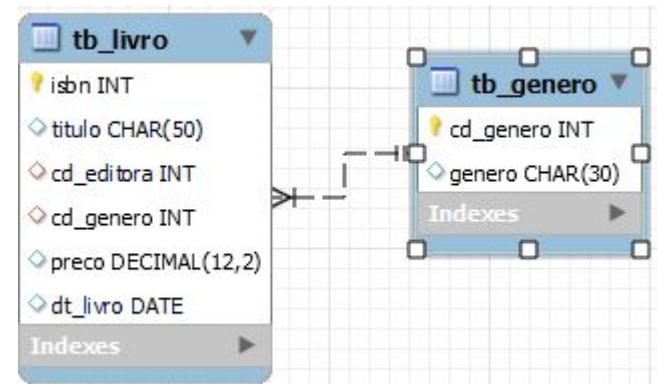
```
select E.editora, count(L.titulo) qtd from
tb_livro L
inner join tb_editora E
on L.cd_editora = E.cd_editora
group by E.editora
order by qtd desc;
```

editora	qtd
novatec	2
saraiva	2
Editora Forum	2
Dickens	2
coopmed	1
livraria florence	1
blucher	1
Mundial	1

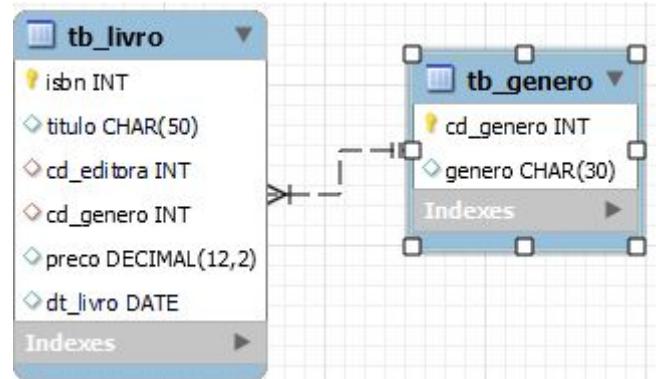
# Consulta de Agregação



- 2.A) **Lista a quantidade de livros por gênero?**



# Consulta de Agregação

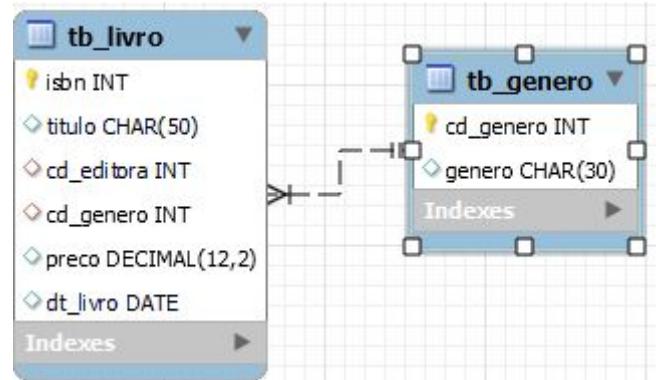


## ■ 2.B) Lista a quantidade de livros por gênero?

```
select G.Genero, L.titulo from tb_livro L
inner join tb_genero G
on L.cd_genero = G.cd_genero;
```

Genero	titulo
Computacao	banco de dados
Computacao	Engenharia de Software
Medicina	Ortopedia
Medicina	Cardiologia
Engenharia	Estrutura Predial
Engenharia	Estrutura Hidraulica
Juridico	Direito Penal
Juridico	Direito Civil
Arquitetura	Cores
Arquitetura	Paisagismo
Biologia	Virus
Biologia	Bacteria

# Consulta de Agregação

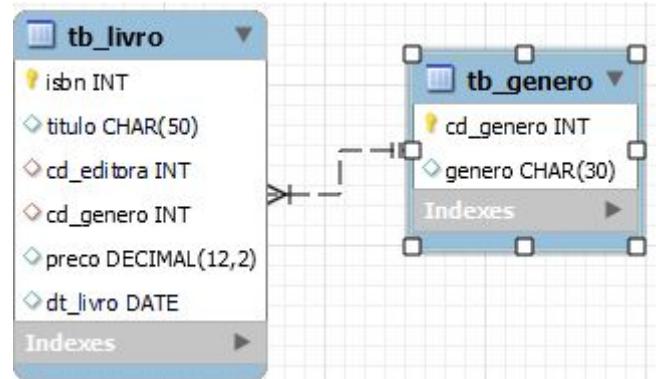


## ■ 2.B) Lista a quantidade de livros por gênero?

```
select G.Genero, L.titulo from tb_livro L
inner join tb_genero G
on L.cd_genero = G.cd_genero;
```

Genero	titulo
Computacao	banco de dados
Computacao	Engenharia de Software
Medicina	Ortopedia
Medicina	Cardiologia
Engenharia	Estrutura Predial
Engenharia	Estrutura Hidraulica
Juridico	Direito Penal
Juridico	Direito Civil
Arquitetura	Cores
Arquitetura	Paisagismo
Biologia	Virus
Biologia	Bacteria

# Consulta de Agregação



- 2.B) **Lista a quantidade de livros por gênero?**

```
select G.Genero, count(L.titulo) qtd from tb_livro L
inner join tb_genero G
on L.cd_genero = G.cd_genero
group by G.genero
order by qtd desc;
```

Genero	qtd
Computacao	2
Medicina	2
Engenharia	2
Juridico	2
Arquitetura	2
Biologia	2

# Exercício

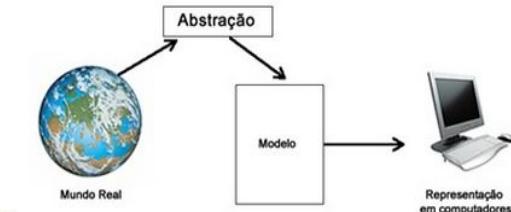
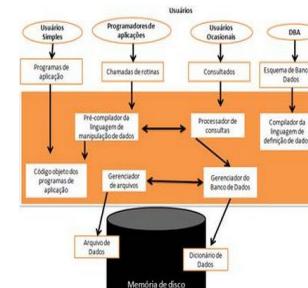
# Exercício

- 1 Lista a quantidade de funcionários por setor.
- 2 Lista o valor total do salario por setor.

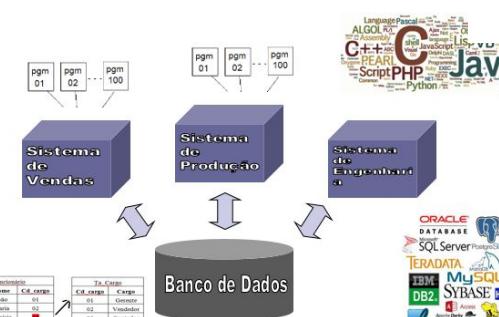
# Regras SGBD

# SGBD - Regras

**Regra 1: Auto-Contenção**



**Regra 3: Abstração dos Dados**



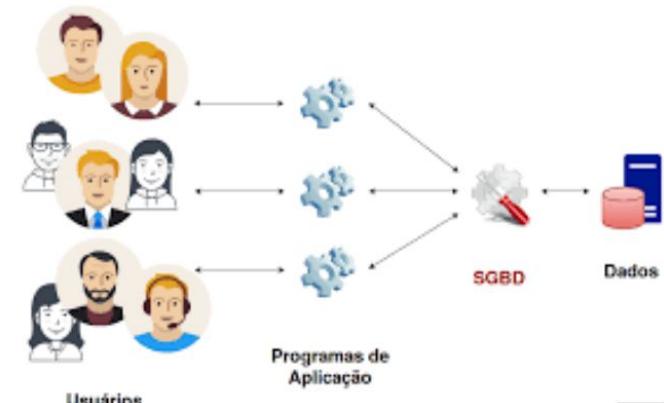
**Regra 4: Visões**

**Regra 5: Integridade referencial**

**Regra 6: Acesso Automático**

Tb Funcionário		
Código	Nome	Cd cargo
001	João	01
002	Maria	02
003	Patrícia	05

Ta Cargo	
Cd cargo	Cargo
01	Gerente
02	Vendedor
03	Contador



# SGBD - Regras

## Regra 7: Transações

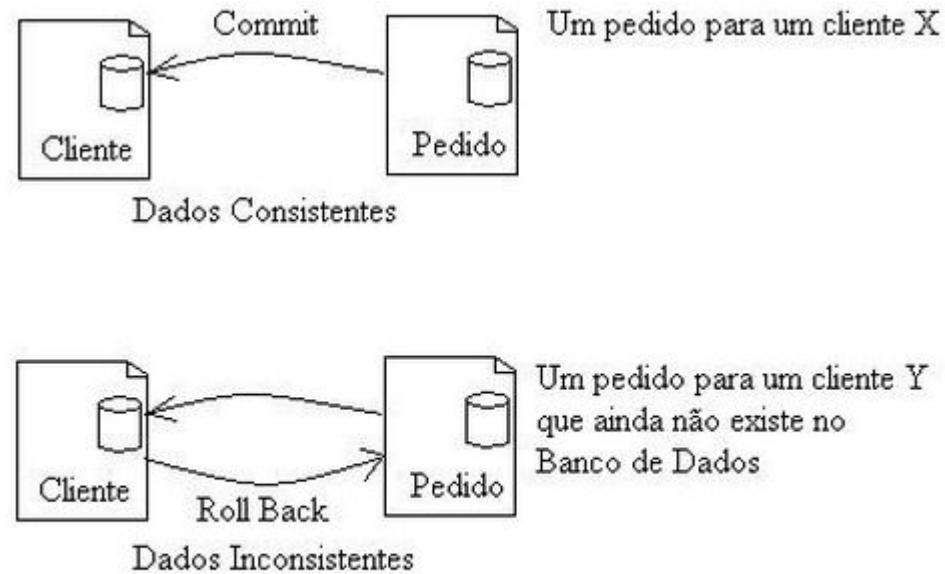
- Desta forma exige-se que o banco de dados tenha ao menos uma instrução que permita a gravação de uma série modificações simultâneas e uma instrução capaz de cancelar um série modificações.



# SGBD - Regras

## Regra 7: Transações

- Desta forma exige-se que o banco de dados tenha ao menos uma instrução que permita a gravação de uma série de modificações simultâneas e uma instrução capaz de cancelar um conjunto de modificações.



## Transações – ACID Comandos do SQL

# SGBD - Regras

## Transações

```
SQLQuery1.sql - HP\...SS.master (sa (53))* X
-- Acessa ambiente
Select nr_conta from tb_correntista where nr_conta = 100

-- Acessa saldo
Select * from tb_saldo where nr_conta = 100

-- retirar dinheiro
update tb_saldo
Set saldo = saldo - 1000
Where nr_conta = 100
```



# Transação

- Requisitos que sempre devem ser atendidos por uma transação
- Chamadas de **propriedades ACID**
  - **Atomicidade**
  - **Consistência**
  - **Isolamento**
  - **Durabilidade ou Persistência**

**Transação para transferir \$ 50  
da conta A para a conta B:**

- 1.  $\text{read}(A)$
- 2.  $A := A - 50$
- 3.  $\text{write}(A)$
- 4.  $\text{read}(B)$
- 5.  $B := B + 50$
- 6.  $\text{write}(B)$

## Exemplo – 1 – atualiza valor\_pedido em 10% - rollback

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2021-06-21	2	2	3	57200.00	5200.00
	3	2019-03-30	3	4	5	119900.00	10900.00
	4	2019-04-10	4	5	1	111100.00	10100.00
	5	2019-05-20	5	5	1	75900.00	6900.00
	6	2019-06-30	6	1	2	69300.00	6300.00
	7	2019-07-10	1	4	5	57200.00	5200.00
	8	2019-08-20	1	4	5	57200.00	5200.00
	9	2019-09-30	1	4	5	57200.00	5200.00
	10	2019-10-10	1	4	5	57200.00	5200.00

## Exemplo – 1 – atualiza valor\_pedido em 10%

```

start transaction;

select * from tb_pedido;

update tb_pedido

set valor_pedido = valor_pedido * 1.1 where nr_pedido < 200;

select * from tb_pedido;

commit;

rollback;

select * from tb_pedido;
  
```

01

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2021-06-21	2	2	3	62920.00	5200.00
	3	2019-03-30	3	4	5	131890.00	10900.00
	4	2019-04-10	4	5	1	122210.00	10100.00
	5	2019-05-20	5	5	1	83490.00	6900.00
	6	2019-06-30	6	1	2	76230.00	6300.00
	7	2019-07-10	1	4	5	62920.00	5200.00
	8	2019-08-20	1	4	5	62920.00	5200.00
...	...	...	...	...	...	...	...

02

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2021-06-21	2	2	3	57200.00	5200.00
	3	2019-03-30	3	4	5	119900.00	10900.00
	4	2019-04-10	4	5	1	111100.00	10100.00
	5	2019-05-20	5	5	1	75900.00	6900.00
	6	2019-06-30	6	1	2	69300.00	6300.00
	7	2019-07-10	1	4	5	57200.00	5200.00
	8	2019-08-20	1	4	5	57200.00	5200.00
	9	2019-09-30	1	4	5	57200.00	5200.00
	10	2019-10-10	1	4	5	57200.00	5200.00

03

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2021-06-21	2	2	3	57200.00	5200.00
	3	2019-03-30	3	4	5	119900.00	10900.00
	4	2019-04-10	4	5	1	111100.00	10100.00
	5	2019-05-20	5	5	1	75900.00	6900.00
	6	2019-06-30	6	1	2	69300.00	6300.00
	7	2019-07-10	1	4	5	57200.00	5200.00
	8	2019-08-20	1	4	5	57200.00	5200.00
	9	2019-09-30	1	4	5	57200.00	5200.00
	10	2019-10-10	1	4	5	57200.00	5200.00

# MySql

## Exemplo – 2 – excluir os pedidos - rollback

01

nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
1	2021-06-21	2	2	3	57200.00	5200.00
3	2019-03-30	3	4	5	119900.00	10900.00
4	2019-04-10	4	5	1	111100.00	10100.00
5	2019-05-20	5	5	1	75900.00	6900.00
6	2019-06-30	6	1	2	69300.00	6300.00
7	2019-07-10	1	4	5	57200.00	5200.00
8	2019-08-20	1	4	5	57200.00	5200.00
9	2019-09-30	1	4	5	57200.00	5200.00
10	2019-10-10	1	4	5	57200.00	5200.00

02

*	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

03

nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
1	2021-06-21	2	2	3	57200.00	5200.00
3	2019-03-30	3	4	5	119900.00	10900.00
4	2019-04-10	4	5	1	111100.00	10100.00
5	2019-05-20	5	5	1	75900.00	6900.00
6	2019-06-30	6	1	2	69300.00	6300.00
7	2019-07-10	1	4	5	57200.00	5200.00
8	2019-08-20	1	4	5	57200.00	5200.00
9	2019-09-30	1	4	5	57200.00	5200.00
10	2019-10-10	1	4	5	57200.00	5200.00

## Exemplo – 2 – excluir os pedidos - rollback

```

start transaction;

select * from tb_pedido;

delete from tb_pedido where nr_pedido < 200;

select * from tb_pedido;

commit;

rollback;

select * from tb_pedido;
  
```

01

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

02

Result Grid | Filter Rows:  | Edit: | Export/Import: | V

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2021-06-21	2	2	3	57200.00	5200.00
	3	2019-03-30	3	4	5	119900.00	10900.00
	4	2019-04-10	4	5	1	111100.00	10100.00
	5	2019-05-20	5	5	1	75900.00	6900.00
	6	2019-06-30	6	1	2	69300.00	6300.00
	7	2019-07-10	1	4	5	57200.00	5200.00
	8	2019-08-20	1	4	5	57200.00	5200.00
	9	2019-09-30	1	4	5	57200.00	5200.00
	10	2019-10-10	1	4	5	57200.00	5200.00

126

# MySQL - Oracle

## Exemplo – 1 - Rollback

```
91 • start transaction;  
92 • update tb_livro  
93 set preco = preco *1.1;  
94 • select * from tb_livro;  
95  
96 • rollback;  
97 • commit;  
98
```

Result Grid							
		isbn	titulo	dt_livro	cd_editora	cd_genero	preco
1	banco de dados			2015-08-10	1	1	110.00
2	engenharia de software			2014-08-20	1	2	220.00
3	Algoritmos: Teoria e Prática			2015-01-10	3	2	242.00
4	Introdução à programação			2014-01-10	4	2	121.00

# Exemplo – 1 - Rollback

```
91 • start transaction;  
92 • update tb_livro  
93 set preco = preco *1.1;  
94 • select * from tb_livro;  
95  
96 • rollback;  
97 • commit;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

isbn	titulo	dt_livro	cd_editora	cd_genero	preco
1	banco de dados	2015-08-10	1	1	110.00
2	engenharia de software	2014-08-20	1	2	220.00
3	Algoritmos: Teoria e Prática	2015-01-10	3	2	242.00

```
96 • rollback;  
97 • select * from tb_livro;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

isbn	titulo	dt_livro	cd_editora	cd_genero	preco
1	banco de dados	2015-08-10	1	1	100.00
2	engenharia de software	2014-08-20	1	2	200.00
3	Algoritmos: Teoria e Prática	2015-01-10	3	2	220.00

# Exemplo – 1 - commit

```
91 • start transaction;  
92 • update tb_livro  
93 set preco = preco *1.1;  
94 • select * from tb_livro;  
95  
96 • rollback;  
97 • commit;  
98
```

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content:					
isbn	titulo	dt_livro	cd_editora	cd_genero	preco
1	banco de dados	2015-08-10	1	1	110.00
2	engenharia de software	2014-08-20	1	2	220.00
3	Algoritmos: Teoria e Prática	2015-01-10	3	2	242.00

```
96 • rollback;  
97 • select * from tb_livro;  
98 • commit;  
99
```

Result Grid   Filter Rows:   Edit:   Export/Import:   Wrap Cell Content:					
isbn	titulo	dt_livro	cd_editora	cd_genero	preco
1	banco de dados	2015-08-10	1	1	110.00
2	engenharia de software	2014-08-20	1	2	220.00
3	Algoritmos: Teoria e Prática	2015-01-10	3	2	242.00

# SQL Server – PostgreSQL – Db2

## Exemplo – 1 – Rollback - Commit

```
select * from tb_genero
```

	cd_genero	genero
1	1	Informatica
2	2	Direito
3	3	Engenharia Civil
4	4	Engenharia Elétrica
5	5	Enfermagem

```
BEGIN TRANSACTION;
    INSERT INTO tb_genero (cd_genero, genero) VALUES(6, 'Jornalismo')
    INSERT INTO tb_genero (cd_genero, genero) VALUES(7, 'Psicologia')

--commit;
ROLLBACK;
```

	cd_genero	genero
1	1	Informatica
2	2	Direito
3	3	Engenharia Civil
4	4	Engenharia Elétrica
5	5	Enfermagem

# Exercício

# Exercício

01 – Atualiza em 10% a mais o salario dos funcionários

E desfaz a atualização.

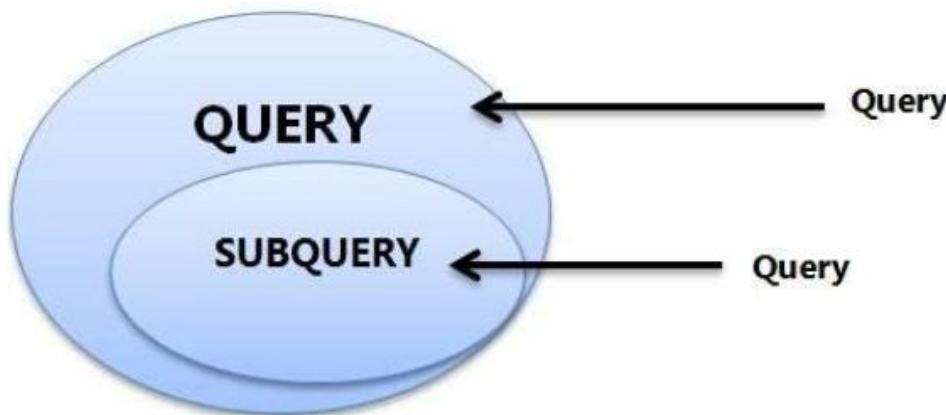
02 – Exclui todos os funcionários

E desfaz a atualização.

# SubQuery

# subquery

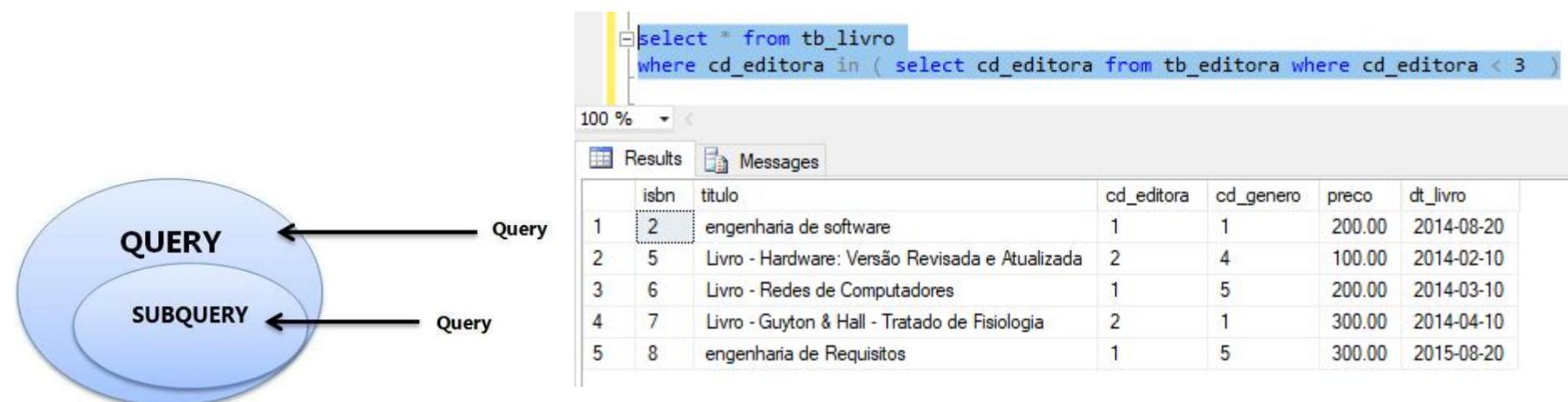
Uma **subconsulta** (ou mais conhecida, subquery) é uma instrução SELECT que está condicionada dentro de outra instrução SQL.



```
1. SELECT column-names  
2.   FROM table-name1  
3. WHERE value IN (SELECT column-name  
4.                   FROM table-name2  
5.                   WHERE condition)
```

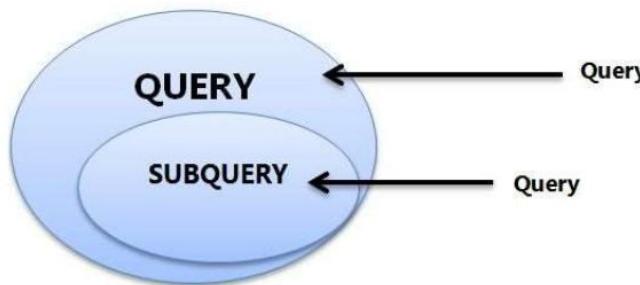
# subquery

Uma **subconsulta** (ou mais conhecida, subquery) é uma instrução SELECT que está condicionada dentro de outra instrução SQL.



# subquery

Determina se um determinado valor corresponde a qualquer valor em uma subconsulta.



Operador	
=	Igual
>	Maior que
<	Menor que
>=	Maior igual
<=	Menor igual
<> ou !=	Diferente
IN	Entre os valores
EXISTS	Existe resultado

# Ex 01



Lista os livros que estão acima da média?

	isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1		banco de dados	1	1	360.00	2019-10-02
2		Engenharia de Software	1	1	420.00	2019-10-03
3		Ortopedia	3	2	372.00	2019-10-04
4		Cardiologia	4	2	384.00	2019-10-05
5		Estrutura Predial	5	3	240.00	2019-10-06
6		Estrutura Hidráulica	6	3	360.00	2019-10-07
7		Direito Penal	7	4	180.00	2019-10-08
8		Direito Civil	8	4	240.00	2019-10-09
9		Cores	7	5	240.00	2019-10-10
10		Paisagismo	8	5	300.00	2019-10-11
11		Virus	9	6	360.00	2019-10-12
12		Bacteria	9	6	360.00	2019-10-12

# Ex 01



Lista os livros que estão acima da média?

01

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1	banco de dados	1	1	360.00	2019-10-02
2	Engenharia de Software	1	1	420.00	2019-10-03
3	Ortopedia	3	2	372.00	2019-10-04
4	Cardiologia	4	2	384.00	2019-10-05
5	Estrutura Predial	5	3	240.00	2019-10-06
6	Estrutura Hidráulica	6	3	360.00	2019-10-07
7	Direito Penal	7	4	180.00	2019-10-08
8	Direito Civil	8	4	240.00	2019-10-09
9	Cores	7	5	240.00	2019-10-10
10	Paisagismo	8	5	300.00	2019-10-11
11	Virus	9	6	360.00	2019-10-12
12	Bacteria	9	6	360.00	2019-10-12

02

`select avg(preco) media from tb_livro;`

media  
318.000000

# Ex 01



Lista os livros que estão acima da média?

01

```
select avg(preco) media from tb_livro;
```

media

318.000000

02

```
select * from tb_livro
```

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1	banco de dados	1	1	360.00	2019-10-02
2	Engenharia de Software	1	1	420.00	2019-10-03
3	Ortopedia	3	2	372.00	2019-10-04
4	Cardiologia	4	2	384.00	2019-10-05
5	Estrutura Predial	5	3	240.00	2019-10-06
6	Estrutura Hidráulica	6	3	360.00	2019-10-07
7	Direito Penal	7	4	180.00	2019-10-08
8	Direito Civil	8	4	240.00	2019-10-09
9	Cores	7	5	240.00	2019-10-10
10	Paisagismo	8	5	300.00	2019-10-11
11	Virus	9	6	360.00	2019-10-12
12	Bacteria	9	6	360.00	2019-10-12

```
where preco >= (select avg(preco) media from tb_livro);
```

03

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1	banco de dados	1	1	360.00	2019-10-02
2	Engenharia de Software	1	1	420.00	2019-10-03
3	Ortopedia	3	2	372.00	2019-10-04
4	Cardiologia	4	2	384.00	2019-10-05
6	Estrutura Hidráulica	6	3	360.00	2019-10-07
11	Virus	9	6	360.00	2019-10-12
12	Bacteria	9	6	360.00	2019-10-12
NULL	NULL	NULL	NULL	NULL	NULL

# Ex 02



Lista os livros que estão acima da média da editora novatec?

	isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1		banco de dados	1	1	360.00	2019-10-02
2		Engenharia de Software	1	1	420.00	2019-10-03
3		Ortopedia	3	2	372.00	2019-10-04
4		Cardiologia	4	2	384.00	2019-10-05
5		Estrutura Predial	5	3	240.00	2019-10-06
6		Estrutura Hidráulica	6	3	360.00	2019-10-07
7		Direito Penal	7	4	180.00	2019-10-08
8		Direito Civil	8	4	240.00	2019-10-09
9		Cores	7	5	240.00	2019-10-10
10		Paisagismo	8	5	300.00	2019-10-11
11		Virus	9	6	360.00	2019-10-12
12		Bacteria	9	6	360.00	2019-10-12

01

`select * from tb_editora;`

cd_editora	editora
1	novatec
2	amazon
3	coopmed
4	livraria florence
5	blucher
6	Mundial
7	saraiva
8	Editora Forum
9	Dickens

## Ex 02



Lista os livros que estão acima da média da editora novatec?

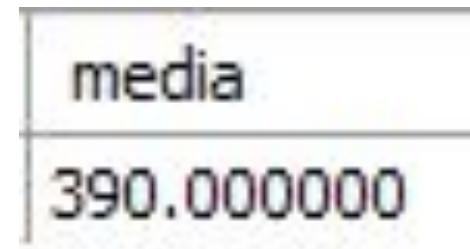
01

```
select * from tb_editora;
```

cd_editora	editora
1	novatec
2	amazon
3	coopmed
4	livraria florence
5	blucher
6	Mundial
7	saraiva
8	Editora Forum
9	Dickens

02

```
select avg(preco) media from tb_livro where cd_editora = 1;
```



# Ex 02



Lista os livros que estão acima da média da editora novatec?

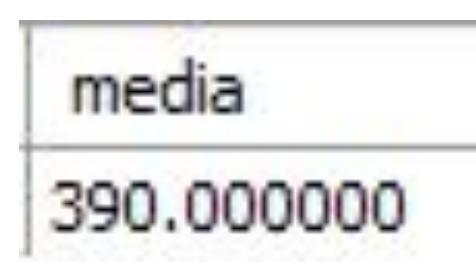
01

```
select * from tb_editora;
```

cd_editora	editora
1	novatec
2	amazon
3	coopmed
4	livraria florence
5	blucher
6	Mundial
7	saraiva
8	Editora Forum
9	Dickens

02

```
select avg(preco) media from tb_livro where cd_editora = 1;
```



## Ex 02



Lista os livros que estão acima da média da editora novatec?

01

```
select avg(preco) media from tb_livro where cd_editora = 1;
```

media

390.000000

02

```
select * from tb_livro where cd_editora = 1;
```

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1	banco de dados	1	1	360.00	2019-10-02
2	Engenharia de Software	1	1	420.00	2019-10-03
NULL	NULL	NULL	NULL	NULL	NULL

# Ex 02



Lista os livros que estão acima da média da editora novatec?

```
select avg(preco) media from tb_livro where cd_editora = 1;
```

01

media

390.000000

02

```
select * from tb_livro where cd_editora = 1;
```

03

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
1	banco de dados	1	1	360.00	2019-10-02
2	Engenharia de Software	1	1	420.00	2019-10-03

```
select * from tb_livro
```

```
where preco >= (select avg(preco) media from tb_livro where cd_editora = 1) and
cd_editora = 1;
```

isbn	titulo	cd_editora	cd_genero	preco	dt_livro
2	Engenharia de Software	1	1	420.00	2019-10-03

# Exercício

# Exercício

01 – Lista o total de funcionários por cargo

02 - Lista os funcionários, cargo com salario acima da média

# 4 - Arquivo sequencial

# Arquivo sequencial

- Registros são distribuídos em uma ordem, um após o outro.
- Ordem pode ser a sequencia de geração dos registros.
- **Vantagem** – Simplicidade
- **Desvantagem** – Busca de registro através de acesso sequencial.

Área de Dados

CD-CLIENTE	NM-CLIENTE
000515	FARID ELETRO MOVEIS
000901	SUPERMERCADO ABC
000988	PAPELARIA OURO PRETO
001101	SUPERMERCADO ITAMARKET
001333	CONFECCAO TERESOPOLIS
001514	CONDOMINIO GREEN VILLAGE
001700	PROMOLIGHT EVENTOS
001240	IMOBILIARIA DEDO DE DEUS
002005	CASA COLORIDA MOVEIS
002201	TEREPOINT PEÇAS E AUTOPEÇAS
002304	OLIVEIRA SANTOS CONTABILIDADE
002800	CONSTRUTORA JJR
000789	POLICLINICA IMBUI
003501	ALTO DROGARIAS

# Índices

# Índices

## Objetivo

Permitir um rápido acesso aleatório aos registros num arquivo.

Banco de dados utilizam índices para recuperação de informação.

Arquivos de índices



## O que é um Índice?

Uma estrutura de dados adicional associada ao arquivo (tabela).

ÍNDICE

Número	Endereço
100	1
150	2
200	3
250	4
300	5

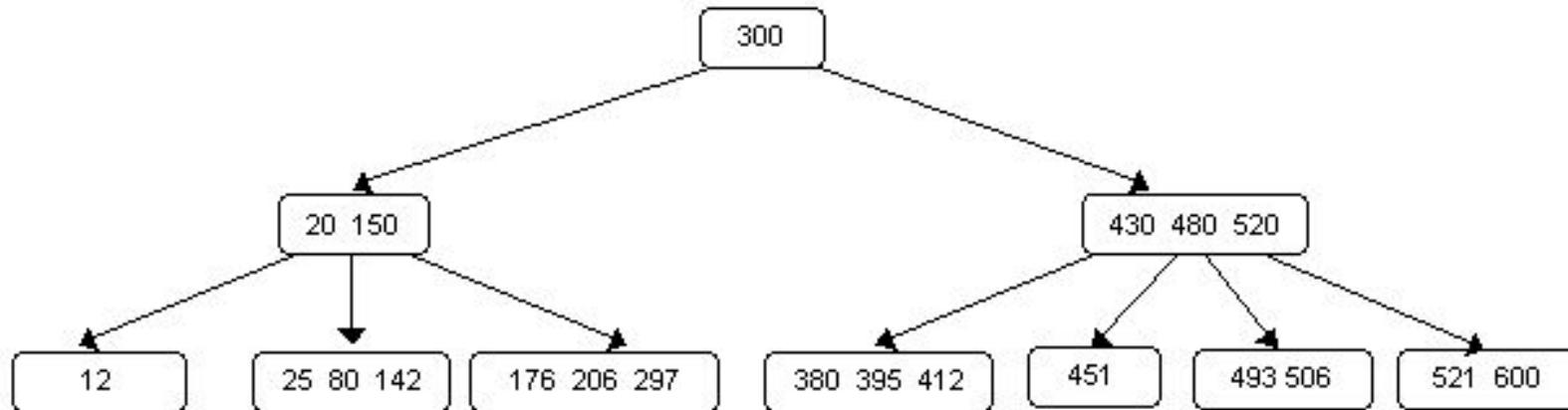
ÁREA DE DADOS NO DISCO

	Número	Nome	Elo
1	100	Pedro	-
2	150	João	10
3	200	Maria	-
4	250	Carla	20
5	300	Max	-

ÁREA DE EXTENSÃO

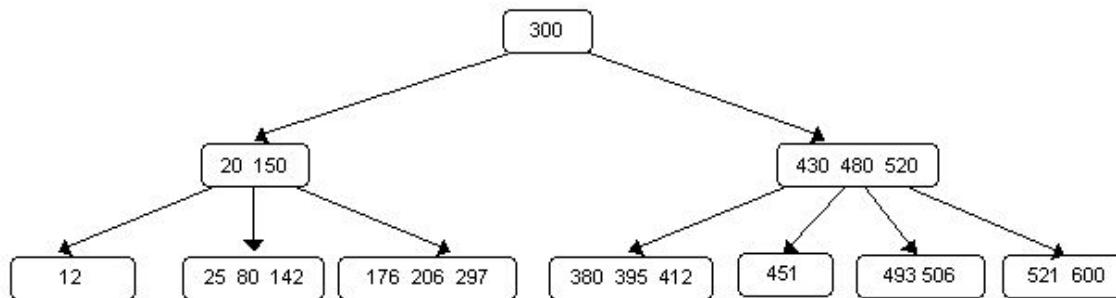
	Número	Nome	Elo
10	175	Bill	3
20	275	Nara	5

# Índice - Árvore B Tree

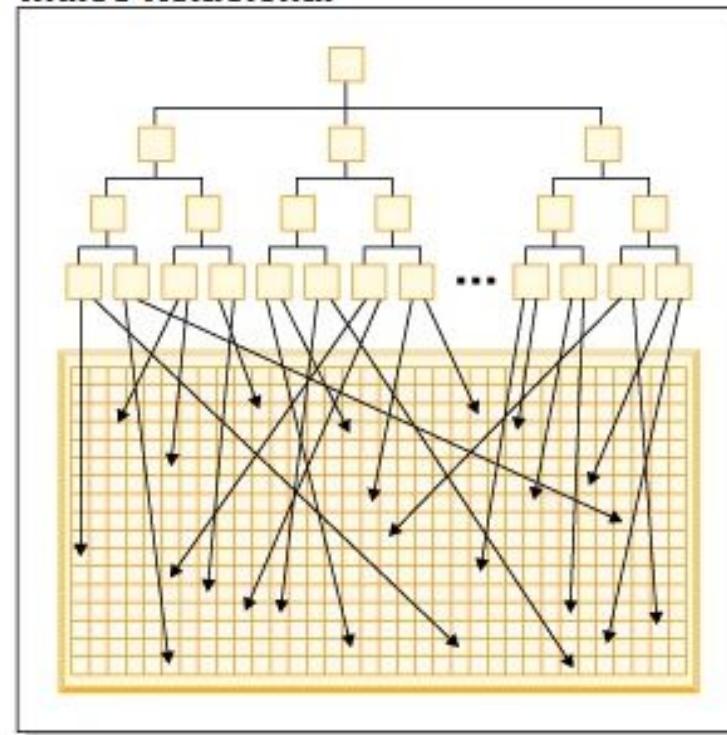


- Uma árvore B é uma estrutura de dados auto-balanceada que armazena dados ordenados e permite **buscas, inserções, remoções e acessos sequenciais em tempo logarítmico**.

# Índice - Árvore B Tree



**Índice Relacional**



# Índice - Árvore B Tree

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

**SCHEMAS**

Filter objects

- bd\_livro
- bd\_rh
  - Tables
    - tb\_cargo
    - tb\_funcionario
      - Columns
        - matricula
        - funcionario
        - dt\_nascimento
        - cd\_setor
        - cd\_cargo
        - salario
    - Indexes
      - PRIMARY
      - fk\_cargo
      - fk\_setor

Administration Schemas

Information

**Index: PRIMARY**

**Definition:**

Type BTREE  
Unique Yes  
Visible Yes  
Columns matricula

Object Info Session

```

59
60
61 • insert into tb_funcionario
62   (matricula, funcionario, dt_nascimento, cd_setor, cd_cargo, salario)
63   Values
64   (1,'Ana Clara', '1977-07-05', 5, 1, 3000),
65   (2,'Patricia Azevedo', '1944-07-04', 1, 1, 4000),
66   (3,'Jose Maria', '1971-05-10', 3, 1, 6000),
67   (4,'Sonia Abrantes', '1979-05-29', 4, 1, 7000),
68   (5,'Valdir Reinaldo', '1960-09-22', 2, 2, 16000),
69   (6,'Jose Alberto', '1955-01-13', 2, 2, 15000);
70
71 • select * from tb_funcionario;
72
73
74
75
76
77
78
79
80
81
82

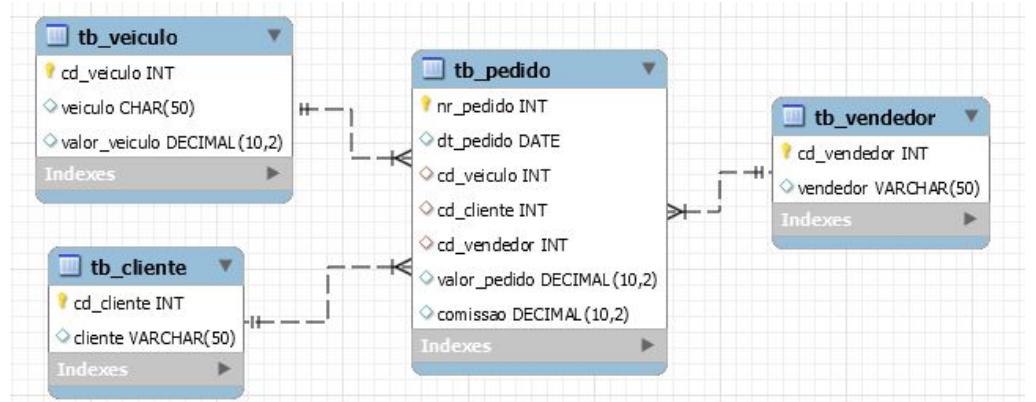
```

# Índices

Por padrão, o SGBD cria índices automaticamente para campos de:

- Chave primária
- Chave estrangeira
- Constraint UNIQUE

Além disso, podemos criar índices para outras colunas usadas com frequência em buscas ou junções.



```
44
45 • use titanicdb;
46 • select * from titanic;
47 • show index from titanic;
48 • create index idx_name on titanic (name);
49 • explain select * from titanic where name = 'Anderson, Mr. Harry';
50 • drop index idx_name on titanic;
```

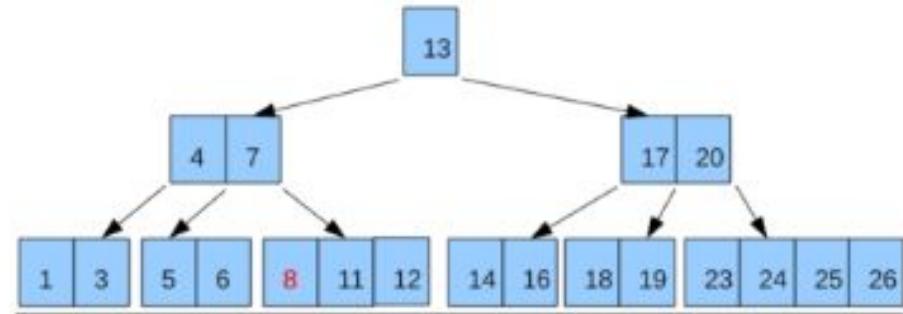
Result Grid | Filter Rows: | Export: | Wrap Cell Contents: | Fetch rows: |

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embark
1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.5500	C22 C26	S
1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.5500	C22 C26	S
1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.5500	E12	S
1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S
1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0.0000	A36	S
1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S
1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	C	
1	0	Astor, Col. John Jacob	male	47	1	0	PC 17757	227.5250	C62 C64	C
1	1	Astor, Mrs. John Jacob (Madeleine Talmadge Fo...	female	18	1	0	PC 17757	227.5250	C62 C64	C
1	1	Aubart, Mme. Leontine Pauline	female	24	0	0	PC 17477	69.3000	B35	C
1	1	Barber, Miss. Ellen "Nellie"	female	26	0	0	19877	78.8500	S	
1	1	Barkworth, Mr. Algernon Henry Wilson	male	80	0	0	27042	30.0000	A23	S
1	0	Baumann, Mr. John D	male	0	0	0	PC 17318	25.9250	S	
1	0	Baxter, Mr. Quigg Edmond	male	24	0	1	PC 17558	247.5208	B58 B60	C
1	1	Baxter, Mrs. James (Helene DeLaudeniere Chap...	female	50	0	1	PC 17558	247.5208	B58 B60	C
1	1	Bazzani, Miss. Albina	female	32	0	0	11813	76.2917	D15	C

# Árvore B (B tree)

# Árvore B (B tree)

Uma árvore B (B tree) é uma estrutura de dados em árvore, **auto-balanceada**, que armazena dados classificados e permite pesquisas, acesso sequencial, inserções e remoções em **tempo logarítmico**.



```

3 • use bd_pedido_100;
4 • select * from tb_pedido;
5 • show index from tb_pedido;
6 • create index idx_dt_pedido on tb_pedido (dt_pedido);
  
```

<

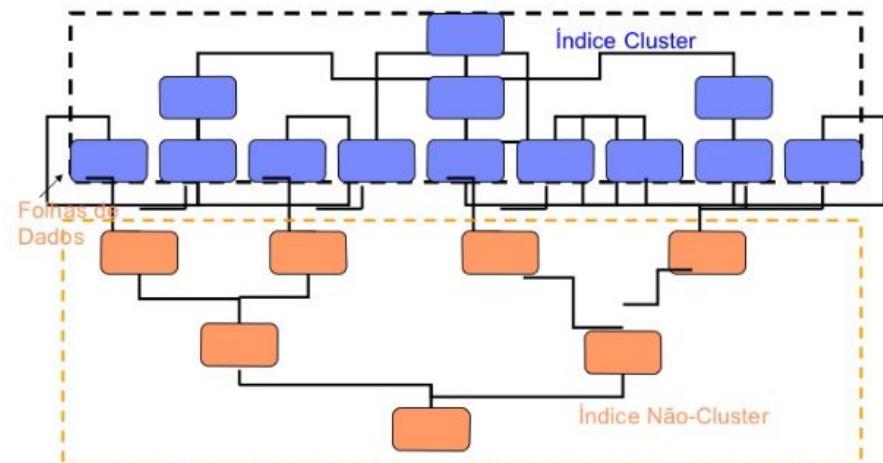
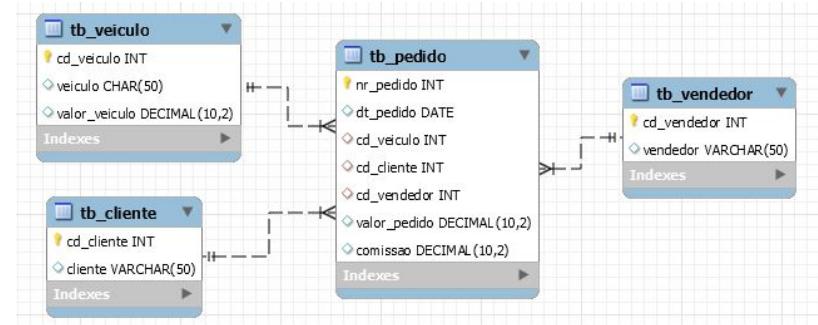
Result Grid											
	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
▶	tb_pedido	0	PRIMARY	1	nr_pedido	A	10	NULL	NULL		BTREE
	tb_pedido	1	fk_veiculo	1	cd_veiculo	A	6	NULL	NULL	YES	BTREE
	tb_pedido	1	fk_cliente	1	cd_cliente	A	5	NULL	NULL	YES	BTREE
	tb_pedido	1	fk_vendedor	1	cd_vendedor	A	5	NULL	NULL	YES	BTREE
	tb_pedido	1	idx_dt_pedido	1	dt_pedido	A	10	NULL	NULL	YES	BTREE

# Índice Clusterizado

## Índice Clusterizado

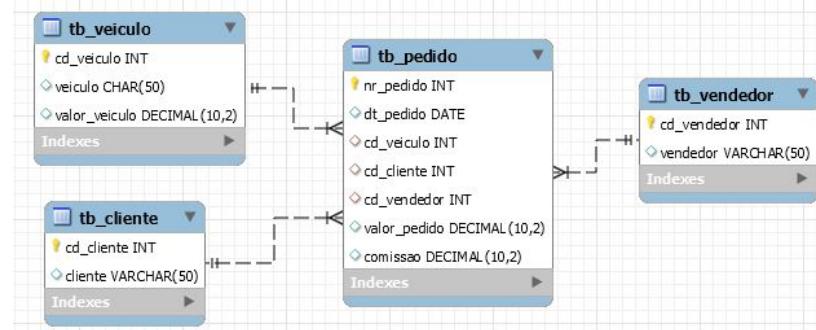
- Índices clusterizados alteram a forma como os dados são armazenados em um banco de dados, pois ele classifica as linhas de acordo com a coluna que possui o índice.

- Uma tabela **só pode ter um índice clusterizado.**



## Índice Clusterizado

- Geralmente está na coluna que é chave primária da tabela.
- Se uma tabela não possuir índice clusterizado, suas linhas são armazenadas em uma estrutura *não-ordenada* chamada de **heap**.



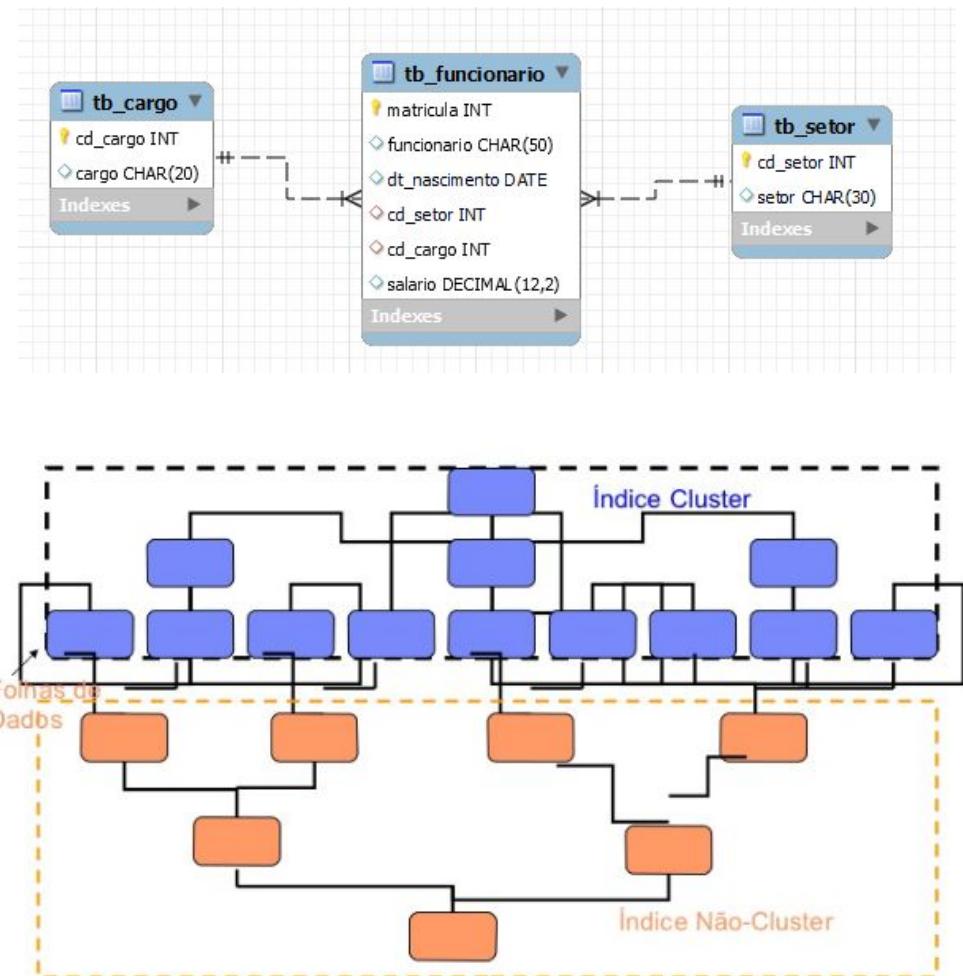
```
3 • use bd_pedido_100;
4
5 • select * from tb_pedido;
6
```

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2019-01-10	1	2	3	52000.00	0.00
	2	2019-02-20	2	3	4	49000.00	0.00
	3	2019-03-30	3	4	5	109000.00	0.00
	4	2019-04-10	4	5	1	101000.00	0.00
	5	2019-05-20	5	5	1	69000.00	0.00
	6	2019-06-30	6	1	2	63000.00	0.00
	7	2019-07-10	1	4	5	52000.00	0.00
	8	2019-08-20	1	4	5	52000.00	0.00
	9	2019-09-30	1	4	5	52000.00	0.00
	10	2019-10-10	1	4	5	52000.00	0.00
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

# Índice Não-Clusterizado

## Índice Não-Clusterizado

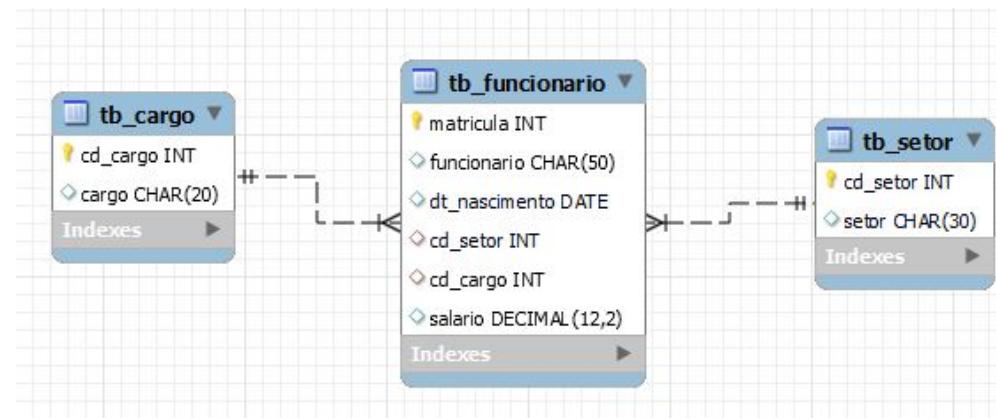
Em um índice não-clusterizado a forma como os dados são armazenados não é alterada, em um objeto separado é criado na tabela, apontando para as linhas da tabela original após a busca.



# Índices

## Índice Não-Clusterizado

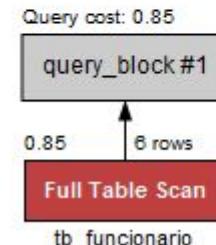
- Baseia-se em valores-chave
- Uma tabela pode ter vários índices não-clusterizados.



```

21
22 • select * from tb_funcionario;
  
```

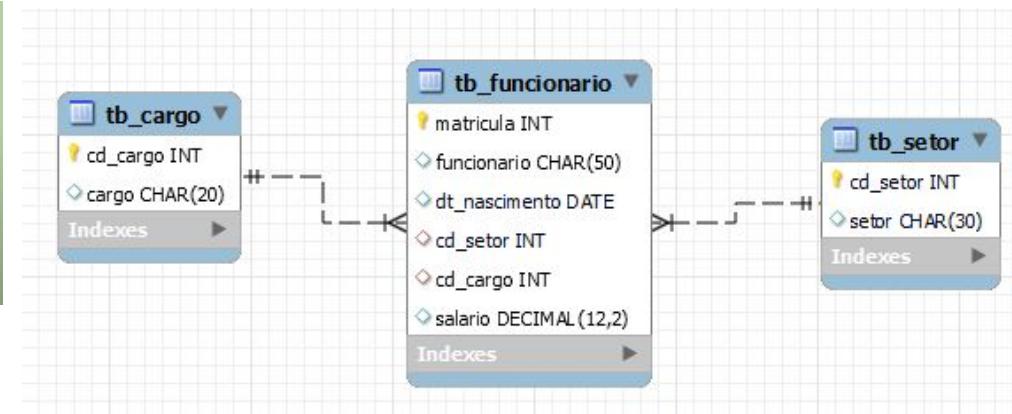
Visual Explain | Display Info: Read + Eval cost | ? | Overview: | View Source: |



# Criando o Índice

# Criando o Índice

```
CREATE [UNIQUE] INDEX nome_índice
ON nome_tabela (
    coluna1 [ASC | DESC],
    [coluna2 [ASC | DESC]]...
);
```



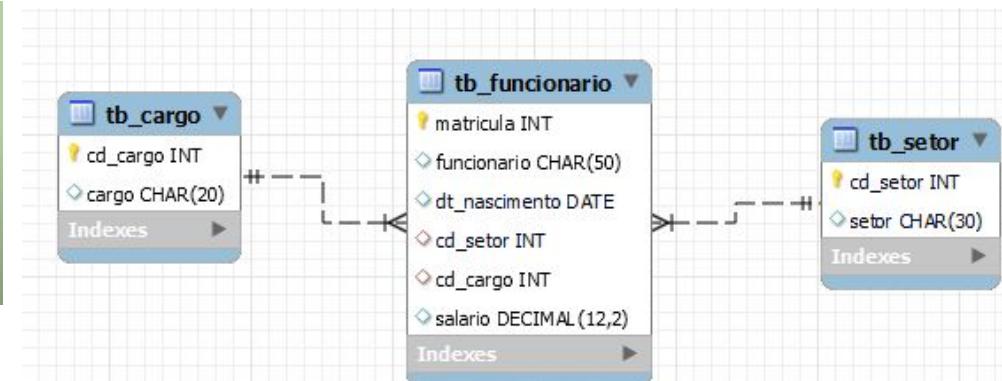
`show index from tb_funcionario;`

```
24 • show index from tb_funcionario;
25
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Co
tb_funcionario	0	PRIMARY	1	matricula	A	6	NULL	NULL		BTREE	
tb_funcionario	1	fk_setor	1	cd_setor	A	4	NULL	NULL	YES	BTREE	
tb_funcionario	1	fk_cargo	1	cd_cargo	A	2	NULL	NULL	YES	BTREE	

# Criando o Índice

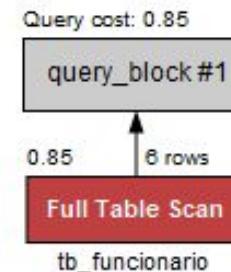
```
CREATE [UNIQUE] INDEX nome_índice
ON nome_tabela (
    coluna1 [ASC | DESC],
    [coluna2 [ASC | DESC]]...
);
```



select \* from tb\_funcionario where funcionario like 'Ana%';

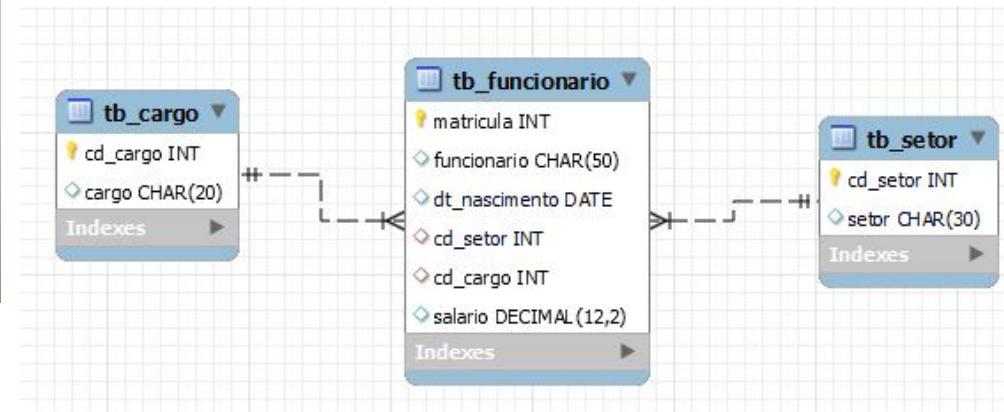
```
28 •   select * from tb_funcionario where funcionario like 'Ana%';
29
```

Visual Explain    Display Info: Read + Eval cost    Overview: View Source:



# Criando o Índice

```
CREATE [UNIQUE] INDEX nome_índice
ON nome_tabela (
    coluna1 [ASC | DESC],
    [coluna2 [ASC | DESC]]...
);
```



```
create index idx_funcionario on tb_funcionario (funcionario);
show index from tb_funcionario; ;
```

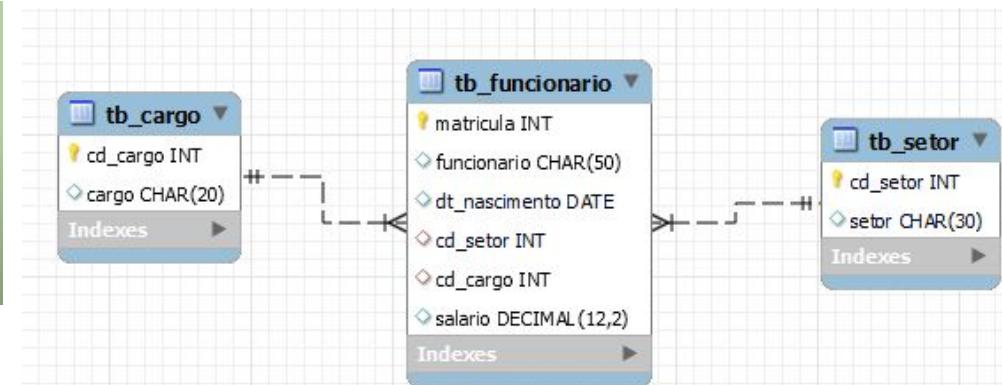
```
33 •  create index idx_funcionario on tb_funcionario (funcionario);
34 •  show index from tb_funcionario;
```

```
35
```

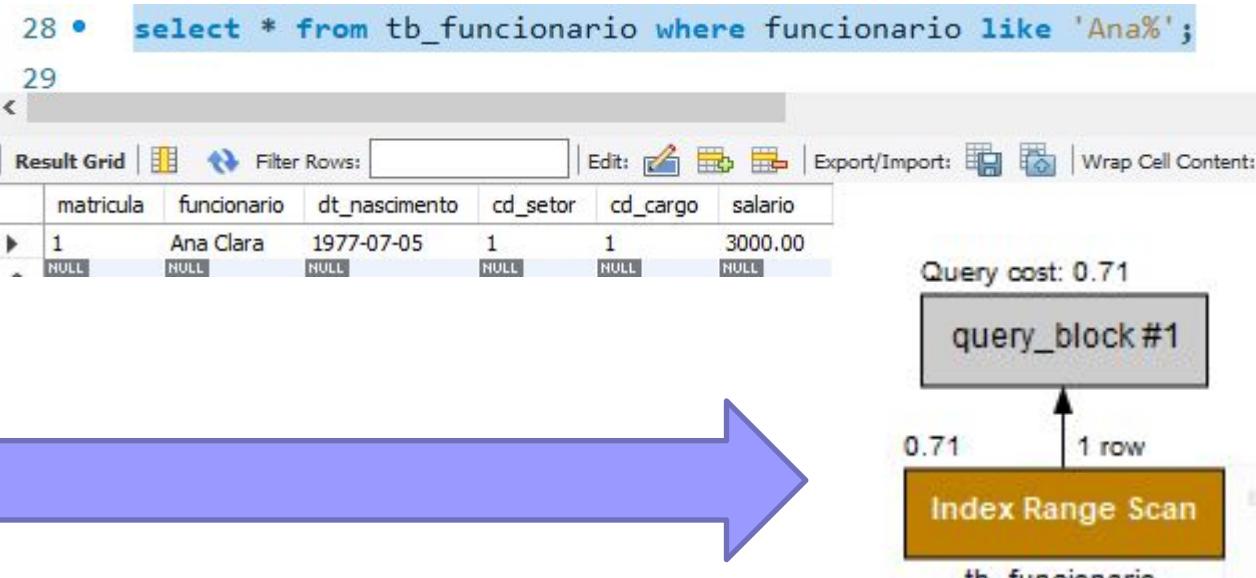
Result Grid										
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
tb_funcionario	0	PRIMARY	1	matricula	A	6	NULL	NULL		BTREE
tb_funcionario	1	fk_setor	1	cd_setor	A	4	NULL	NULL	YES	BTREE
tb_funcionario	1	fk_cargo	1	cd_cargo	A	2	NULL	NULL	YES	BTREE
tb_funcionario	1	idx_funcionario	1	funcionario	A	6	NULL	NULL	YES	BTREE

# Criando o Índice

```
CREATE [UNIQUE] INDEX nome_índice
ON nome_tabela (
  coluna1 [ASC | DESC],
  [coluna2 [ASC | DESC]]...
);
```



```
select * from tb_funcionario where funcionario like 'Ana%';
```



The screenshot shows the MySQL Workbench interface. At the top, the query is displayed:

```
28 • select * from tb_funcionario where funcionario like 'Ana%';
29
```

The results grid shows one row:

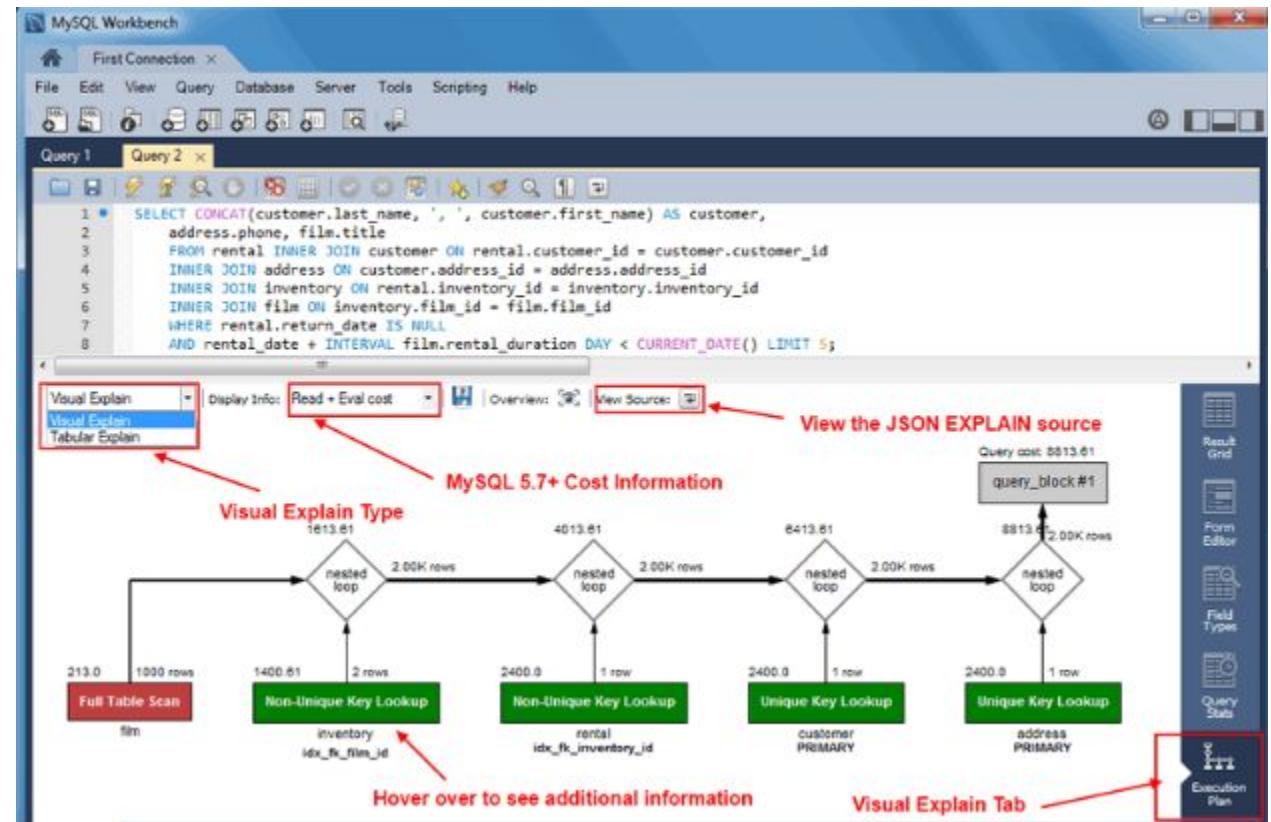
	matricula	funcionario	dt_nascimento	cd_setor	cd_cargo	salario
▶	1	Ana Clara	1977-07-05	1	1	3000.00
◀	NULL	NULL	NULL	NULL	NULL	NULL

Below the results, the execution plan is shown:

- Full Table Scan** (tb\_funcionario): Query cost: 0.85, 6 rows.
- Index Range Scan** (tb\_funcionario idx\_funcionario): 0.71, 1 row.
- query\_block #1**: 0.71, 1 row.

# Plano de execução

# Índices –plano de execução



## 1 – O que é um plano de execução

Um plano de execução exibe de maneira gráfica os métodos de recuperação de dados que o Otimizador de Consulta do SQL Server escolheu. Além disso, mostra também o custo (em termos de uso de recursos) da execução das instruções.

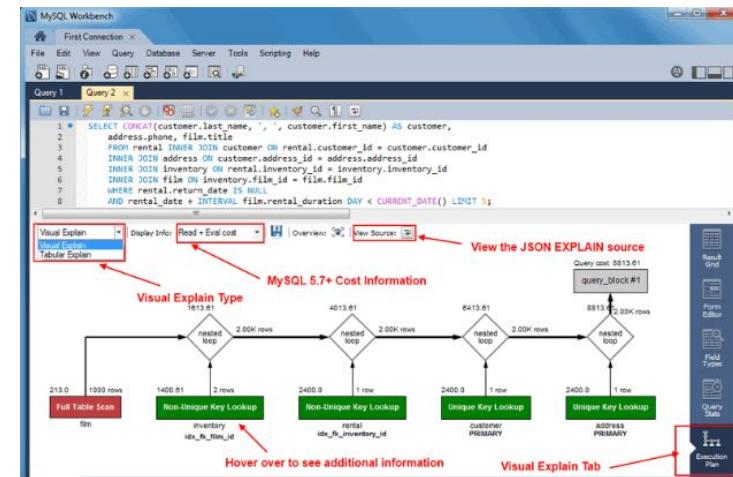
# Índices –plano de execução

## 2 – Para que serve um plano de execução

Com esse recurso, você consegue ver as etapas de execução da sua query e assim verificar o que pode ser melhorado.

Uma outra utilidade é comparar duas queries.

Imagine que você fez uma melhoria no seu sistema e alterou uma consulta.



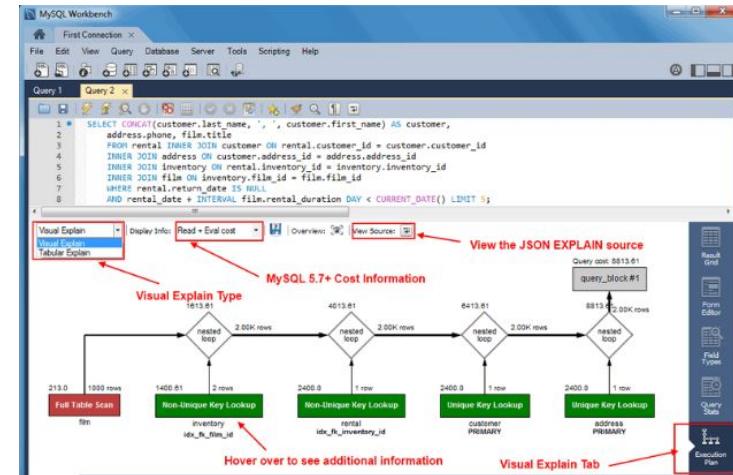
# Índices –plano de execução

## 3- Quando usar um plano de execução

use o plano de execução sempre que escrever

uma consulta! Assim você sempre entregará a

melhor query (naquele momento).



# Exercício

# Exercício

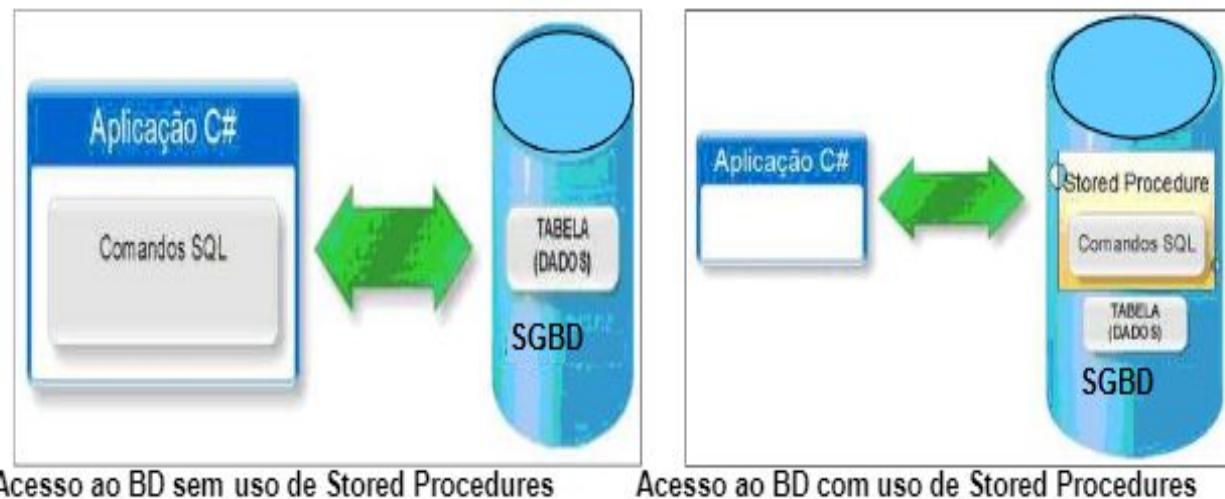
- 01 – Lista o funcionário com o nome = “Jose Alberto”
- 02 – verifique o plano de execução e melhore a performance

# Stored Procedure

# O que é Stored Procedure?

**Stored Procedure**, que traduzido significa **Procedimento Armazenado**, é uma conjunto de comandos em SQL que podem ser executados de uma só vez.

Ele armazena tarefas repetitivas e aceita parâmetros de entrada para que a tarefa seja efetuada de acordo com a necessidade individual.



# O que é Stored Procedure?

## *SQL Statement*

x

## *Stored Procedure*

### Primeira execução

- Verifica sintaxe
- Compila
- Executa
- Retorna dados

### Segunda execução

- Verifica sintaxe
- Compila
- Executa
- Retorna dados

### Criação

- Verifica sintaxe
- Compila

### Primeira execução

- Execute
- Retorna dados

### Segunda execução

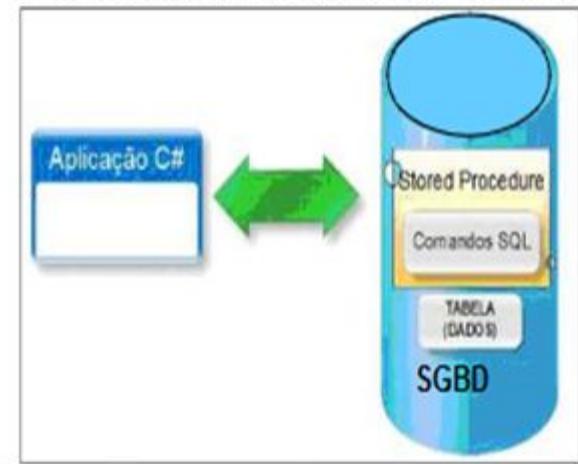
- Executa
- Retorna dados

# Vantagens da Stored Procedure

1. Facilitar a manutenção e a alteração das aplicações
2. Ocultar a complexidade de acesso ao BD
3. Poder receber parâmetros de entrada e retornar resultados
4. Reduzir o tráfego de rede gerado pela aplicação
5. Facilitar e centralizar o gerenciamento de permissões
6. Melhora a velocidade de execução.



Acesso ao BD sem uso de Stored Procedures



Acesso ao BD com uso de Stored Procedures

# Delimitador Stored Procedure

# Delimitador Stored Procedure

```
use bd_pedido_02;
DELIMITER $$

CREATE PROCEDURE sp_primeiro()
BEGIN
    select * from tb_cliente limit 1;
    select * from tb_pedido limit 1;
    select * from tb_veiculo limit 1;
    select * from tb_vendedor limit 1;
END$$

DELIMITER ;
call sp_primeiro();
```

## **Begin...End;**

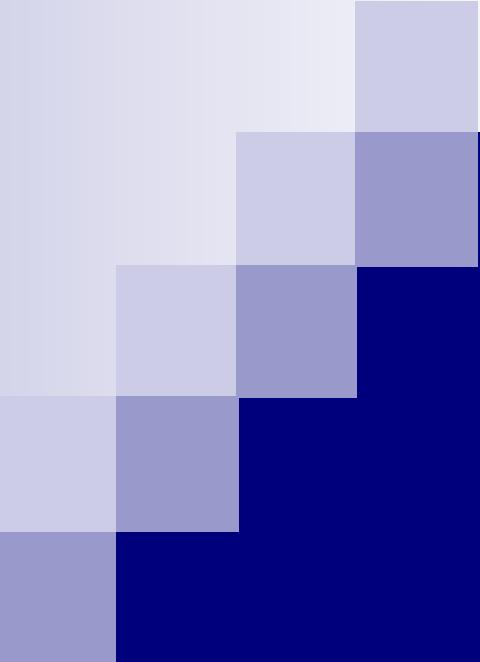
- São (contêineres) usados para delimitar blocos de comandos a serem executados pela stored procedure.
- Cada declaração aninhada possui um delimitador (;
- Um bloco de Begin pode ser aninhado dentro de outros blocos.

# Delimitador Stored Procedure

```
use bd_pedido_01;  
  
DELIMITER $$  
  
CREATE PROCEDURE sp_primeiro()  
BEGIN  
    select * from tb_cliente limit 1;  
    select * from tb_pedido limit 1;  
    select * from tb_veiculo limit 1;  
    select * from tb_vendedor limit 1;  
  
END$$  
  
DELIMITER ;  
  
call sp_primeiro();
```

**DELIMITER** para criar rotinas com vários comandos.

- Porém o **delimitador** **(;)** pode ser problemático pois, ao ser encontrado em um procedimento, finaliza imediatamente.



# Stored Procedure sem Parâmetros

# Stored Procedure sem parâmetros

```

DELIMITER $$

CREATE PROCEDURE sp_02()
BEGIN
    select P.nr_pedido, P.dt_pedido, V.veiculo, P.valor_pedido from
    tb_pedido P inner join tb_veiculo V
    on P.cd_veiculo = V.cd_veiculo;
END$$

DELIMITER ;

```

`call sp_02();`

nr_pedido	dt_pedido	veiculo	valor_pedido
1	2019-01-10	Onix	52000.00
7	2019-07-10	Onix	52000.00
8	2019-08-20	Onix	52000.00
9	2019-09-30	Onix	52000.00
10	2019-10-10	Onix	52000.00
2	2019-02-20	Prisma	49000.00
3	2019-03-30	S10	109000.00
4	2019-04-10	Cruze	101000.00
5	2019-05-20	Spin	69000.00
6	2019-06-30	Cobalt	63000.00

# Stored Procedure sem parâmetros

Criar uma sp\_02A para listar (nr\_pedido, dt\_pedido, valor\_pedido, comissao)

nr_pedido	dt_pedido	valor_pedido	comissao
1	2019-01-10	52000.00	5200.00
2	2019-02-20	49000.00	4900.00
3	2019-03-30	109000.00	10900.00
4	2019-04-10	101000.00	10100.00
5	2019-05-20	69000.00	6900.00
6	2019-06-30	63000.00	6300.00
7	2019-07-10	52000.00	5200.00
8	2019-08-20	52000.00	5200.00
9	2019-09-30	52000.00	5200.00
10	2019-10-10	52000.00	5200.00

# Stored Procedure Parâmetros (in)

# Stored Procedure parâmetros (in)

DELIMITER \$\$

```

CREATE PROCEDURE sp_03(in
vcd_veiculo int)
BEGIN
  select P.nr_pedido, P.dt_pedido, V.veiculo, C.cliente, P.valor_pedido from
  tb_pedido P inner join tb_veiculo V
  on P.cd_veiculo = V.cd_veiculo
  inner join tb_cliente C
  on P.cd_cliente = C.cd_cliente
  where P.cd_veiculo = vcd_veiculo;

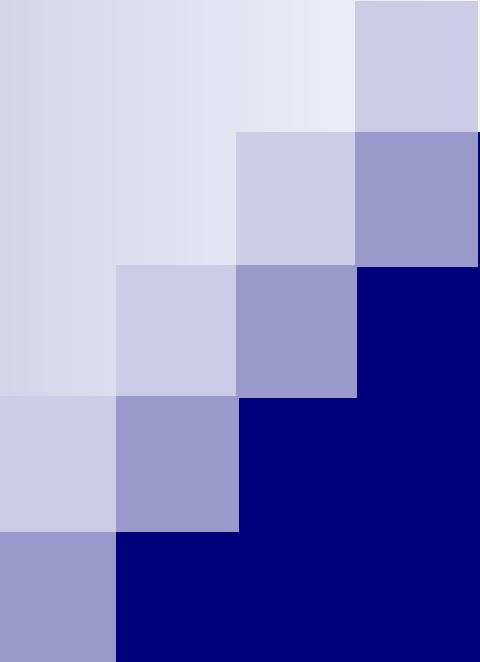
```

END\$\$

DELIMITER ;

call sp\_03(1);

nr_pedido	dt_pedido	veiculo	cliente	valor_pedido
1	2019-01-10	Onix	Rogeria Negreti	52000.00
7	2019-07-10	Onix	Wellington Alves	52000.00
8	2019-08-20	Onix	Wellington Alves	52000.00
9	2019-09-30	Onix	Wellington Alves	52000.00
10	2019-10-10	Onix	Wellington Alves	52000.00



# Stored Procedure múltiplos Parâmetros

# Stored Procedure parâmetros (in)

DELIMITER \$\$

```

CREATE PROCEDURE sp_04(in
vcd_veiculo int,
vcd_cliente int)
BEGIN
  select * from tb_pedido
  where cd_veiculo = vcd_veiculo or cd_cliente
= vcd_cliente;
  
```

END\$\$

DELIMITER ;

	nr_pedido	dt_pedido	cd_veiculo	cd_cliente	cd_vendedor	valor_pedido	comissao
▶	1	2019-01-10	1	2	3	52000.00	5200.00
	6	2019-06-30	6	1	2	63000.00	6300.00
	7	2019-07-10	1	4	5	52000.00	5200.00
	8	2019-08-20	1	4	5	52000.00	5200.00
	9	2019-09-30	1	4	5	52000.00	5200.00
	10	2019-10-10	1	4	5	52000.00	5200.00

call sp\_04(1,1);

# Exercício

# Exercício

- 1) Criar uma stored procedure - **sp\_07A** para listar  
(nr\_pedido, dt\_pedido, cliente, valor\_pedido, comissao) tb\_pedido e tb\_cliente
  
- 2) Criar uma stored procedure - **sp\_07B** para Incluir Novos vendedores e depois listar

# Exercicio

```

use bd_pedido_02;

-- 01 - Criar uma stored procedure - sp_07A para listar
-- (nr_pedido, dt_pedido, cliente, valor_pedido, comissao) tb_pedido e
tb_cliente

delimiter $$

create procedure sp_07A ()
begin

  select P.nr_pedido, P.dt_pedido, C.cliente, P.valor_pedido,
P.comissao
    from tb_pedido P inner join tb_cliente C
  on P.cd_cliente = C.cd_cliente;

end$$

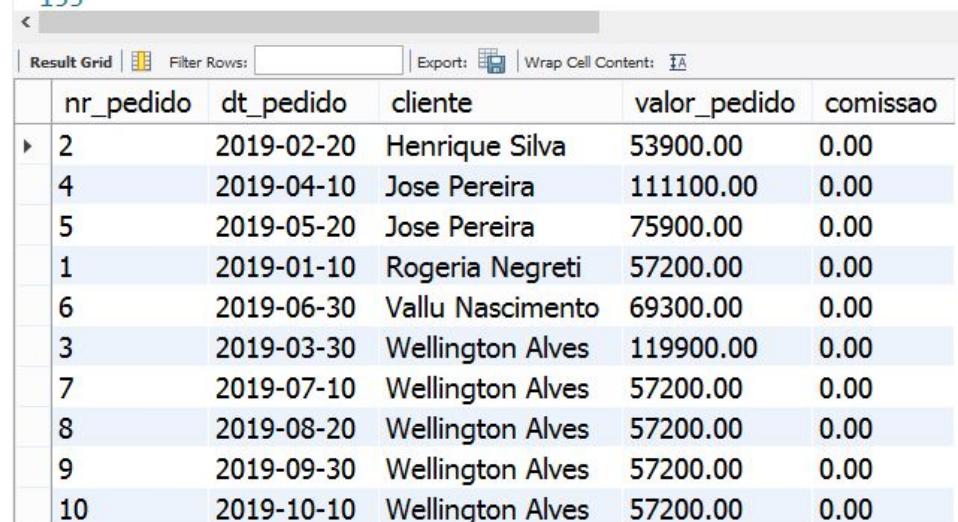
delimiter ;

call sp_07A;
  
```

151 • call sp\_07A;

152

153



nr_pedido	dt_pedido	cliente	valor_pedido	comissao
2	2019-02-20	Henrique Silva	53900.00	0.00
4	2019-04-10	Jose Pereira	111100.00	0.00
5	2019-05-20	Jose Pereira	75900.00	0.00
1	2019-01-10	Rogeria Negreti	57200.00	0.00
6	2019-06-30	Vallu Nascimento	69300.00	0.00
3	2019-03-30	Wellington Alves	119900.00	0.00
7	2019-07-10	Wellington Alves	57200.00	0.00
8	2019-08-20	Wellington Alves	57200.00	0.00
9	2019-09-30	Wellington Alves	57200.00	0.00
10	2019-10-10	Wellington Alves	57200.00	0.00

# Exercicio

```
-- 02 Criar uma stored procedure - sp_07B para
-- Incluir Novos vendedores e depois listar

delimiter $$

create procedure sp_07B (in vcd_vendedor int , vvendedor varchar(50))
begin

    insert into tb_vendedor
    (cd_vendedor, vendedor)
    values
    (vcd_vendedor, vvendedor);

    select * from tb_vendedor;

end$$

delimiter ;
select * from tb_vendedor;
call sp_07B(7,'Edson Arantes');
```

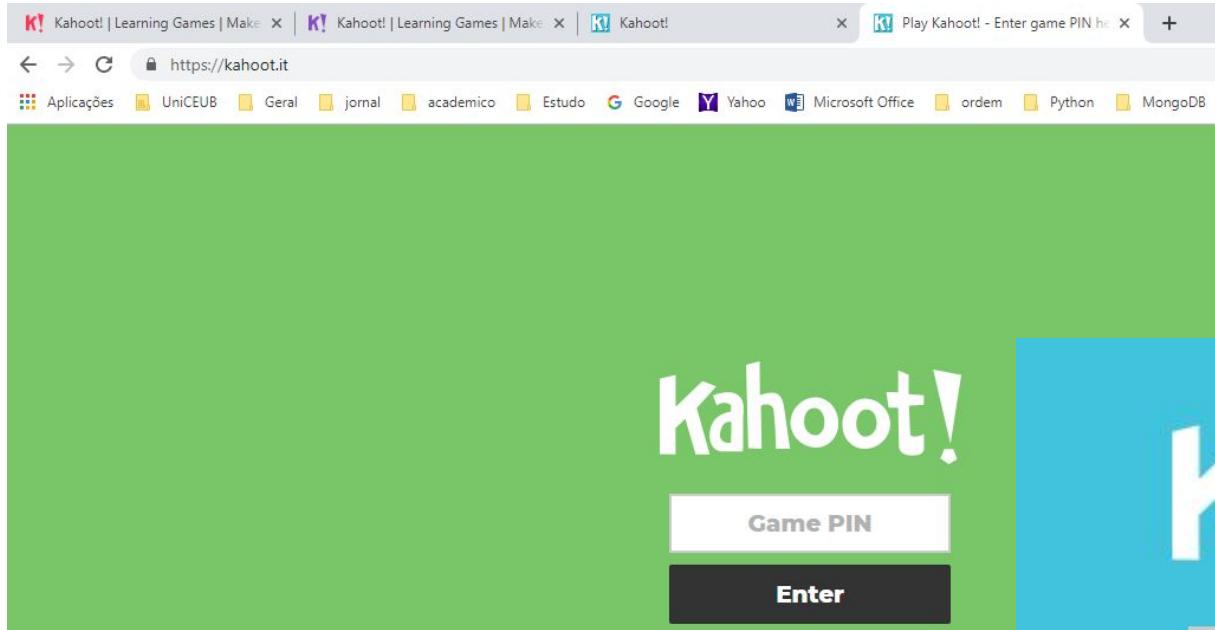
131 • call sp\_07B(7,'Edson Arantes');

132

	cd_vendedor	vendedor
▶	1	Anibal
	2	Antonio de Moraes
	3	Barbara Alcantara
	4	Deise Castro
	5	Eider Nascimento
	6	Eduardo Castro
	7	Edson Arantes
	NULL	NULL

# Questionário

# Questionário



<https://kahoot.it/>

Nome + RA



# Agenda

- Exercício – Prático
- Backup
- Restore
- Comandos de Restrição em SQL
- Inner Join
- View
- Outer Join - Right e Left
- Manipulação de Data
- Cálculo
- Consulta de Agregação
- Transações – ACID Comandos do SQL
- SubQuery
- Índices
- Plano de execução
- Stored Procedure

# Dúvidas

[Jose.wellington@uniceub.br](mailto:Jose.wellington@uniceub.br)