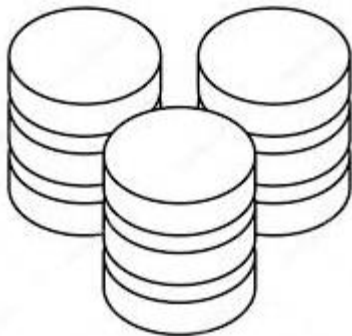


## Banco de Dados II

### Índices



Profa. Damires Souza  
damires@ifpb.edu.br



# Como fazemos uma busca em um livro??

---

**B**ancos de dados e sistemas de banco de dados são um componente essencial da vida na sociedade moderna; a maioria de nós encontra diariamente diversas atividades que envolvem alguma interação com um banco de dados. Por exemplo, quando vamos ao banco para depositar ou retirar fundos, fazemos uma reserva de hotel ou de voo, acessamos o catálogo de uma biblioteca virtual para procurar uma referência bibliográfica, ou compramos algo on-line — como um livro, um brinquedo ou um computador —, provavelmente essas atividades envolverão alguém ou algum programa de computador que acessa um banco de dados. Até mesmo a compra de produtos em um supermercado atualiza automaticamente o banco de dados que mantém o controle de estoque dos itens.

Essas interações são exemplos do que podemos chamar de **aplicações de banco de dados tradicionais**, em que a maior parte da informação armazenada e acessada é textual ou numérica. Nos últimos anos, os avanços na tecnologia levaram a interessantes novas aplicações dos sistemas de banco de dados. A nova tecnologia de mídia tornou possível armazenar imagens, cliques de áudio e streams de

nufatura. Além disso, técnicas de pesquisa de banco de dados estão sendo aplicadas à World Wide Web para melhorar a busca por informações necessárias feita pelos usuários que utilizam a Internet.

No entanto, para entender os fundamentos da tecnologia de banco de dados, devemos começar das aplicações básicas de banco de dados tradicional. Na Seção 1.1, começamos definindo um banco de dados, e depois explicamos outros termos básicos. Na Seção 1.2, oferecemos um simples exemplo de banco de dados UNIVERSIDADE para ilustrar nossa discussão. A Seção 1.3 descreve algumas das principais características dos sistemas de banco de dados, e as seções 1.4 e 1.5 classificam os tipos de pessoas cujas funções envolvem o uso e a interação com sistemas de banco de dados. As seções 1.6, 1.7 e 1.8 oferecem uma discussão mais profunda sobre as diversas capacidades oferecidas pelos sistemas de banco de dados e discutem algumas aplicações típicas. No final do capítulo é apresentado um resumo.

O leitor que quiser uma introdução rápida aos sistemas de banco de dados pode estudar as seções 1.1 a 1.5, depois pular ou folhear as seções 1.6 a 1.8 e seguir para o Capítulo 2.

# Como fazemos uma busca em um livro??

## ➤ ÍNDICE

- Identifica um determinado assunto e **localiza** a sua **posição** em uma **página**

Calça skinny ou justa.....	108
Calça pijama.....	109
Bolsos para calça feminina.....	110
Tipos de cós de calça.....	110
Short.....	112
Short saia.....	113
Modelagem masculina.....	114
Base do corpo masculino.....	115
Colete masculino.....	117
Camisa social masculina.....	118
Calça masculina simples.....	121
Calça social masculina com pregas.....	123
Bolsos para calça masculina.....	124
Cueca samba canção.....	126
Bermuda masculina.....	127
Blazer masculino.....	128
1- Manga alfaiate de blazer ou paletó masculino.....	129
2- Forro, revel e barra do blazer ou paletó masculino.....	130
3- Forro de manga e bolsos de blazer ou paletó masculino.....	131
Modelagem infantil.....	132
Vestido infantil evasé.....	133
Vestido infantil clássico.....	134
Base do corpo do vestido infantil.....	135
Saia infantil com babados sobrepostos.....	136
Calcinha para bebê.....	137
Camisa infantil unisex.....	138
Casaquinho de bebê.....	140
Bermuda Infantil para menino.....	141
Calça infantil unisex.....	142
Casaco infantil unisex.....	143
Dicas importantes.....	144
1- Para traçar a cava perfeita.....	144
2- Para traçar cava e manga para quem tem o braço muito grosso.....	145
3- Para traçar uma saia para mulheres com bumbum avantajado.....	146
4- Para traçar uma blusa para mulheres com muito seio.....	147
5- Procedimento para costurar em máquina doméstica.....	148
6- Para costurar malha em máquina de costura doméstica.....	149
7- Margem de costura.....	149
8- Tipos de máquina de costura.....	149
Ampliação e redução de moldes.....	150
Como colocar bolso em calça.....	153
1- Bolso faca.....	153
2- Bolso regular.....	154
3- Bolso embutido.....	155
4- Bolso embutido lateral.....	156
5- Bolso cargo.....	157

# Tabela (*Relação*)

Query Editor Query History

```
1 select * from artista;
```

**Tabela = conjunto de dados -> não ordenado**

```
select * from artista order by codart;
```

Data Output Explain Messages Notifications

	<b>codart</b> [PK] integer	<b>nomeart</b> character varying (25)	<b>cidade</b> character varying (20)	<b>pais</b> character varying (20)	<b>datanasc</b> date	<b>indicaoscar</b> integer
1	7	Rodrigo Santoro	Rio de Janeiro	Brasil	1978-10-12	0
2	1	Cameron Diaz	[null]	Brasil	1975-07-15	0
3	2	Julia Roberts	[null]	Brasil	2067-08-20	0
4	4	Joaquin Phoenix	[null]	Brasil	1972-04-06	0
5	8	Rosamund Pike	[null]	USA	1979-01-27	0
6	9	Rosamund Pike	[null]	USA	1979-01-27	0
7	10	Rosamund Pike	[null]	USA	1979-01-27	0
8	11	Rosamund Pike	[null]	USA	1979-01-27	0

# Tabela x Consulta

---

Create table **Teste** (  
id integer,  
nome varchar(40));

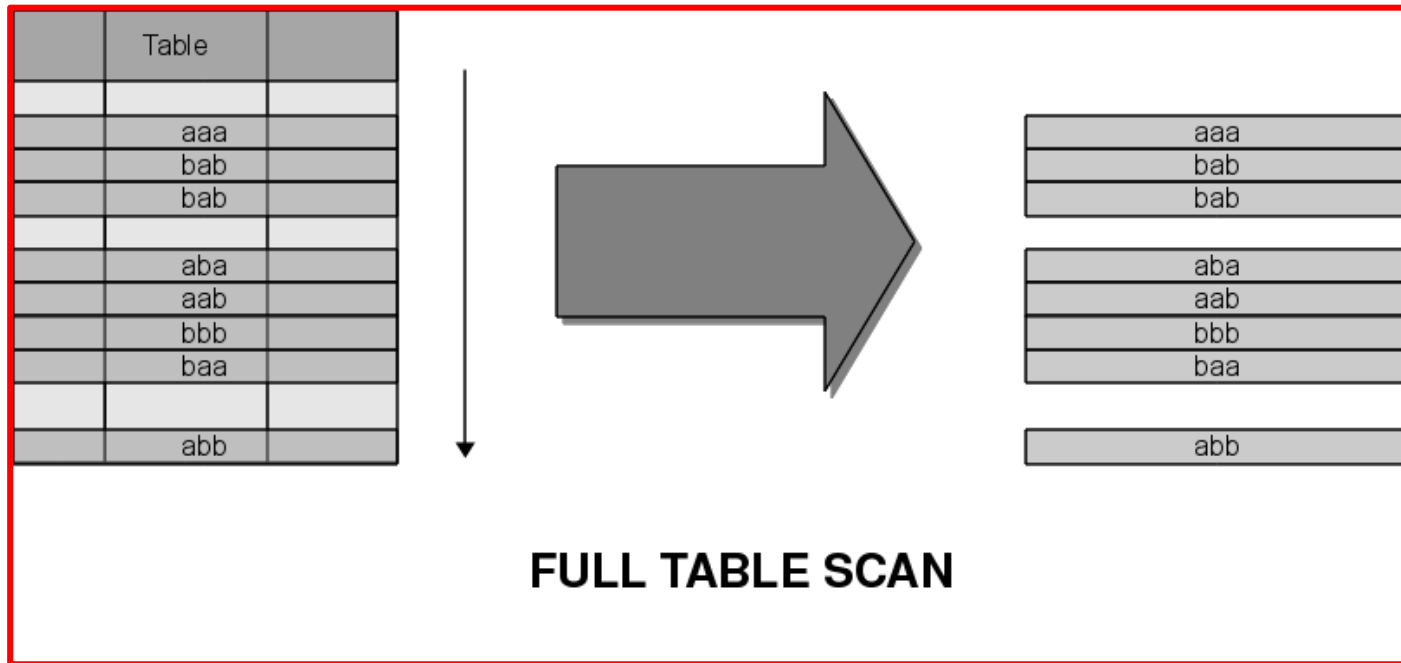
Teste => 100.000 linhas....

**Aplicação A com consulta:**

```
SELECT nome  
FROM teste  
WHERE id = 11001;
```

**O que aconteceria??**

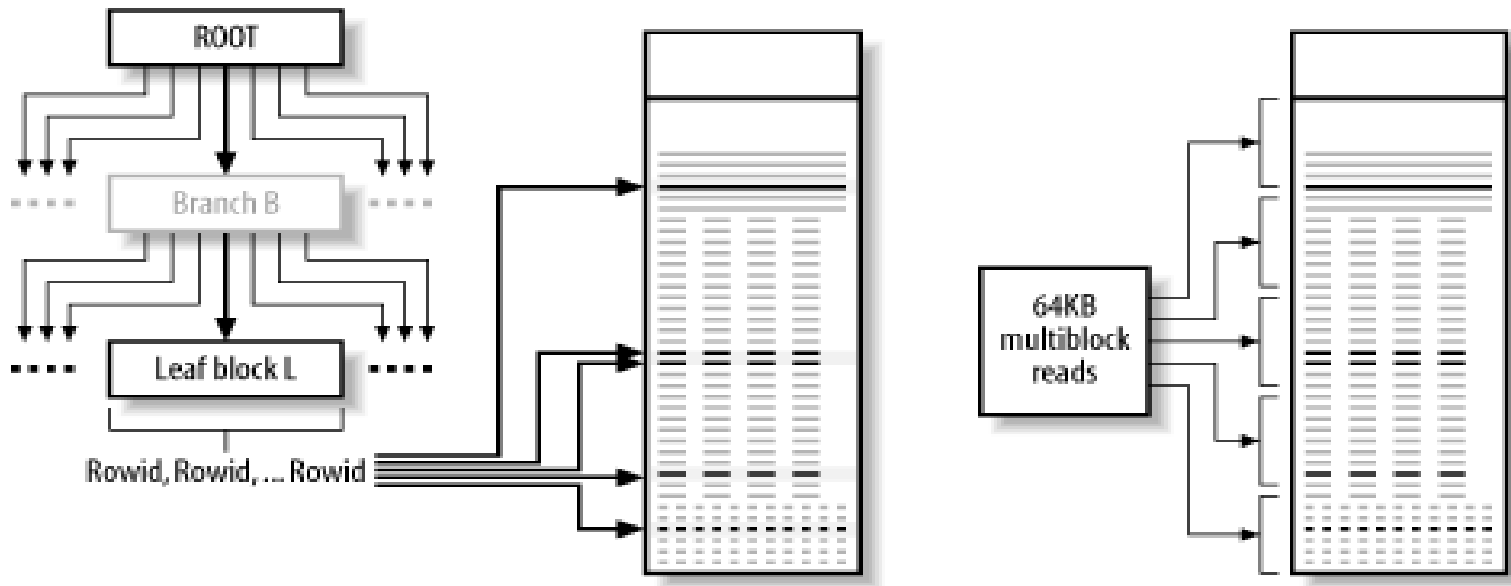
# Tabela x Consulta



- Examina todas as páginas de dados da tabela, começando do seu início, **passando por todos os registros**, página a página e **extraindo aqueles que satisfazem aos critérios da consulta**.

# Voltando ao exemplo...

Com índice



- Percorre-se a **estrutura do índice para localizar os registros**, extraíndo somente aqueles registros necessários para satisfazerem os critérios da consulta

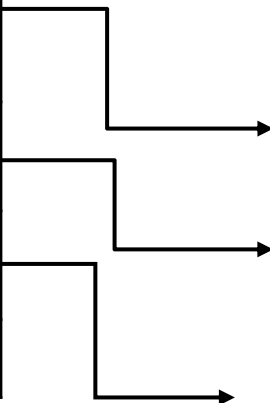
# Visão mais geral

Tabela

1233	Joana Maria	João Pessoa
1654	Priscila Marques	Recife
1321	Vagner Gonçalves	Natal

Índice

1233				
1654				
1321				



Armacenados...

<b>1233 – Joana...</b>	<b>1334 – Mariana...</b>
<b>1654 – Priscila...</b>	<b>1678 – Pedro...</b>
<b>1321 – Vagner...</b>	<b>1345 – Carla...</b>
<b>1667 – Patricio...</b>	<b>1688 – Poliana...</b>



# Índices

---

- Um **índice** é uma estrutura de BD usada pelo servidor para **localizar rapidamente** uma linha ou registro de uma tabela
- Índices são **estruturas auxiliares** às tabelas com o objetivo de melhorar o desempenho no seu acesso
- Como criar?

```
CREATE INDEX idx_teste_id ON Teste(id);
```



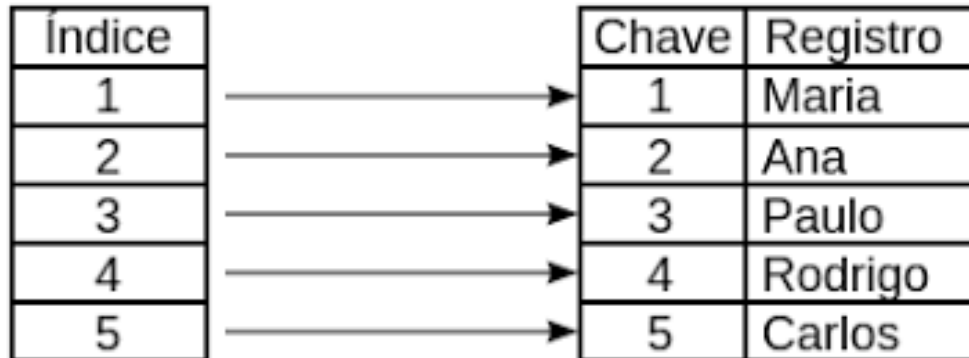
Índice

Tabela

# Índices

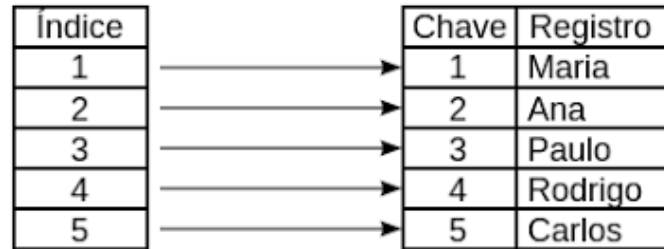
---

- O índice possui **tamanho muito menor** do que o tamanho do arquivo de dados (tabela)
  - Pode ser mantido em **memória principal** caso seja de tamanho suficiente



# Índices

---



- Índices são lógica e fisicamente independentes de uma tabela
- A existência de índices não afeta a localização física dos registros dos arquivos de dados
- Índices podem ser **únicos** ou não
- Índices podem ser definidos sobre **uma ou mais colunas**

# Exemplo

ANG3795	167
COL31809	353
COL38358	211
DG139201	396
DG18807	256
FF245	442
LON2312	32
MER75016	300
RCA2626	77
WAR23699	132

*índice*

32	LON   2312   Romeo and Juliet   Prokofiev ...
77	RCA   2626   Quartet in C Sharp Minor ...
132	WAR   23699   Touchstone   Corea ...
167	ANG   3795   Symphony No. 9   Beethoven ...
211	COL   38358   Nebraska   Springsteen ...
256	DG   18807   Symphony No. 9   Beethoven ...
300	MER   75016   Coq d'or Suite   Rimsky ...
353	COL   31809   Symphony No. 9   Dvorak ...
396	DG   139201   Violin Concerto   Beethoven ...
442	FF   245   Good News   Sweet Honey In The ...

chave  
primária

*arquivo de dados*

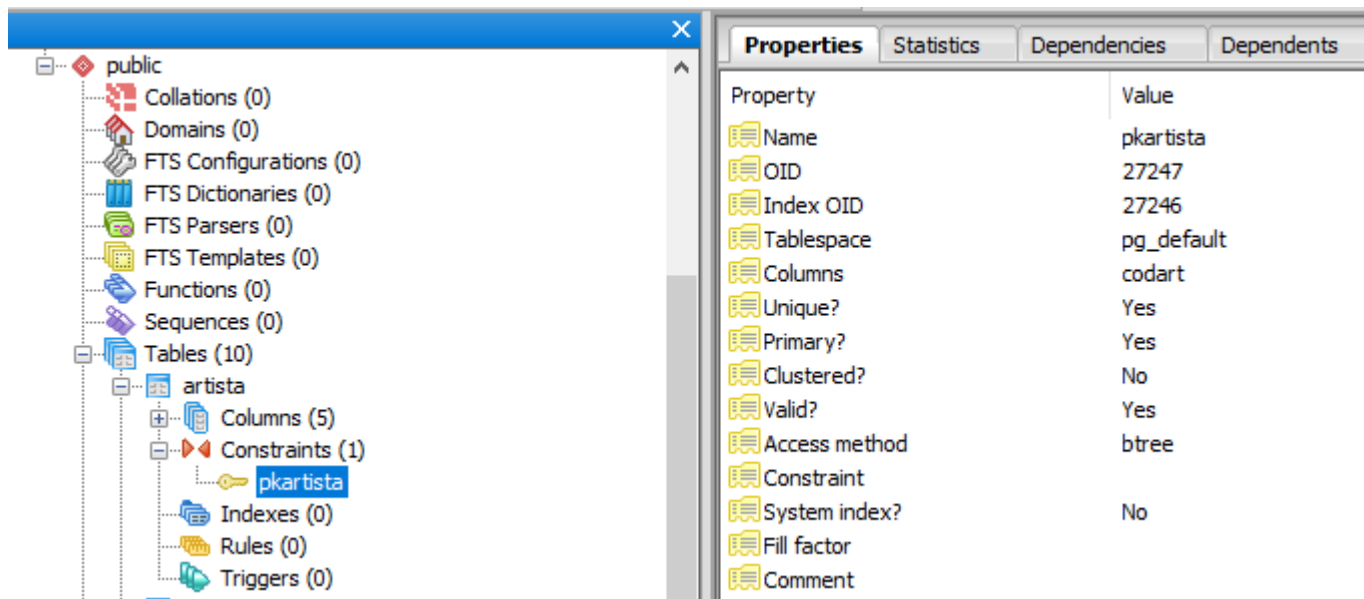
# Operações básicas no Índice

---

- i. Criar estrutura de índice e de dados;
- ii. Inserir registro
  - Inserção será feita **no arquivo de dados** e também no **índice**, que eventualmente será reorganizado.
- iii. Eliminar registro
  - Remove do **arquivo de dados** e remove também do **índice**.
    - A remoção do registro do índice pode exigir a sua reorganização.
- iv. Atualizar registro
  - Modifica o conteúdo do registro e atualiza índice.

# Índices e Chaves

- **Índices e chaves (*constraints*)** são conceitos diferentes, mas podem ser complementares
  - O SGBD normalmente cria, de forma automática, um índice para as **chaves primárias** e *constraints* **unique**

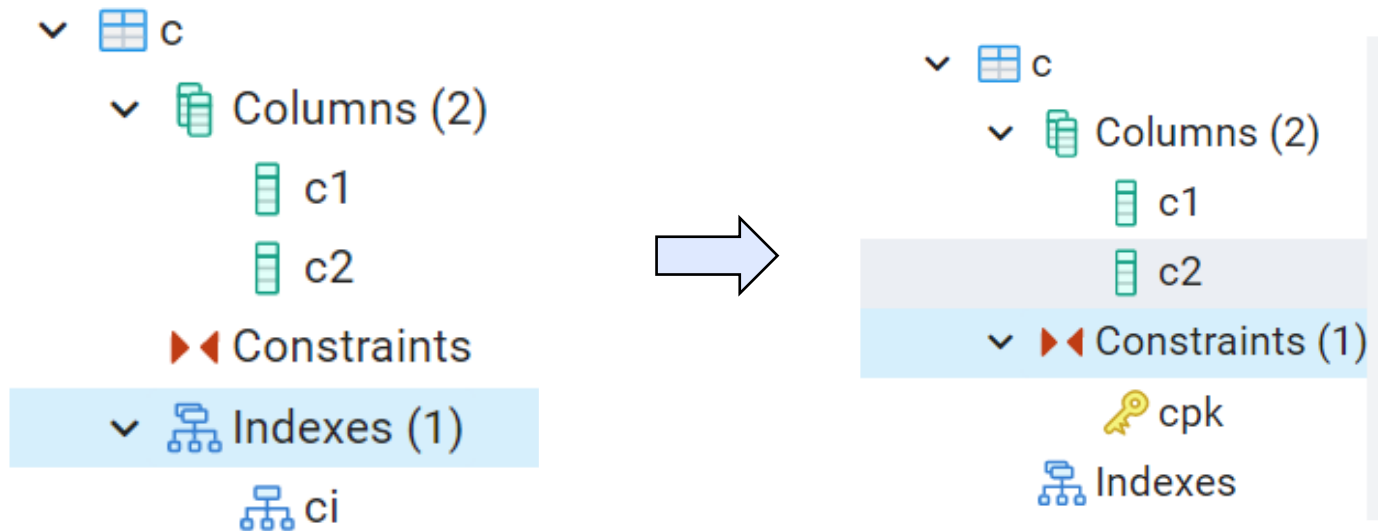


# Índices e Chaves - Exemplo

Create table **c** (c1 INT, c2 INT);

Create unique INDEX **ci** ON **c** (c1, c2);

Alter table **c** ADD CONSTRAINT cpk PRIMARY KEY  
USING INDEX **ci**;

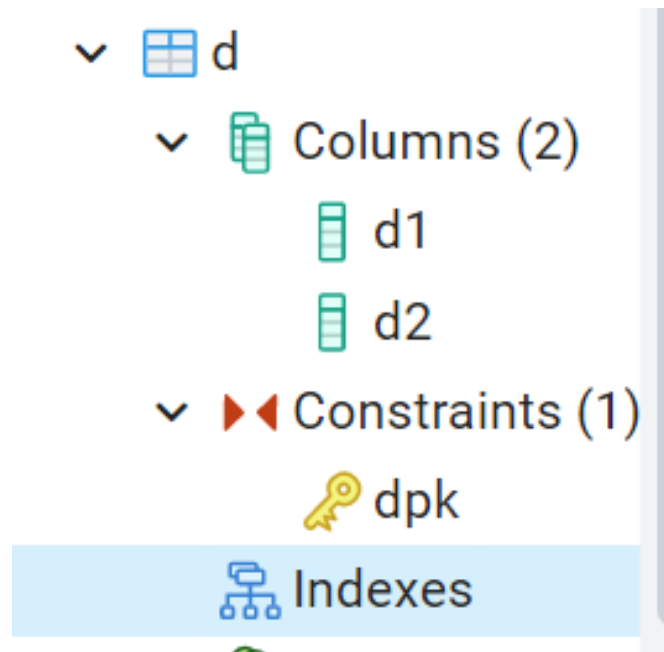


# Índices e Chaves - Exemplo



```
CREATE TABLE d (d1 INT, d2 INT);
```

```
ALTER TABLE d ADD CONSTRAINT dpk PRIMARY  
KEY (d1,d2);
```





# Visão de índices

---

```
SELECT *  
FROM pg_indexes  
WHERE tablename = 'c';
```

```
SELECT *  
FROM pg_indexes  
WHERE tablename = 'd';
```

```
SELECT *  
FROM pg_indexes  
WHERE tablename = 'categoria';
```

# Índice Composto

- Um **índice composto** ou **índice concatenado** é criado com várias colunas de uma tabela
- Índices compostos podem acelerar a recuperação de dados para as instruções SELECT em que a cláusula **where** referencia **as colunas** do índice.

VENDOR_PARTS		
VEND ID	PART NO	UNIT COST
1012	10-440	.25
1012	10-441	.39
1012	457	4.95
1010	10-440	.27
1010	457	5.10
1220	08-300	1.33
1012	08-300	1.19
1292	457	5.28

Concatenated Index  
(index with multiple columns)

# Índice Composto - Exemplo

---

```
CREATE TABLE testeIn (id integer, maior integer, menor integer, nome varchar(10));
```

```
Insert into testeIn values (1,200,30,'X');
```

```
Insert into testeIn values(2, 300,23,'Y');
```

```
Insert into testeIn values(3, 200,30,'Z');
```

```
SELECT nome
```

```
FROM testeIn
```

```
WHERE maior = 200 AND menor = 30;
```

```
CREATE INDEX idx_testeIn_maior_menor  
ON testeIn (maior, menor);
```

# Índices – Como escolher?

---

Select \*

From filme;

```
select * from filme;  
explain select * from filme;
```

QUERY PLAN		
	text	
1	Seq Scan on filme (cost=0.00..1.07 rows=7 width=88)	

# EXPLAIN e principais operações

---

Operação	Descrição
Seq Scan	Básica, representa a leitura sequencial na tabela.
Index Scan	Ocorre quando há a leitura/uso de índice para evitar a leitura inteira da tabela ou para evitar um sort.
Sort	Ocorre principalmente para satisfazer a cláusula ORDER BY, mas pode ser usada também quando se necessita da ordenação de um conjunto de dados
Unique	Principalmente usada quando há a cláusula DISTINCT
Nested Loop	Usada para processar JOINS entre tabelas. Não processa a tabela interna completamente.
Hash e hash JOIN	Usadas para JOINS ou UNIONs. As colunas que fazem a junção não precisam estar ordenadas.

# Exemplo

```
select * from filme;  
explain select * from filme;
```

Data Output	Explain	Messages	Notifications
	QUERY PLAN		
	text		
1	Seq Scan on filme (cost=0.00..1.07 rows=7 width=88)		

- **Custo inicial estimado:** tempo gasto antes que a fase de saída possa começar (e.g., para fazer a classificação em um nó de classificação).
- **Custo total estimado:** Supondo que o nó do plano é executado até a conclusão.
- **Número estimado de saídas de linhas**
- **Largura média estimada das linhas geradas por este nó do plano (em bytes).**

# Índices – Como escolher?

---

Select \*

From filme

Where **ano** = 2021;

**CREATE INDEX** anoIn **ON** filme(**ano**);

\*\* É tarefa do desenvolvedor de banco de dados prever **quais índices** trarão **benefícios às consultas**

---

# Testando índice

```
-- Teste de índice
-- drop table testafilme;

--Criando nova tabela
create table testaFilme as select * from filme;
select * from testafilme;

-- Populando nova tabela
```



# Quando Criar Índices?

---

- **A tabela é volumosa**, muito consultada e pouco atualizada;
- A coluna (ou colunas) é (são) usada(s) frequentemente em cláusulas **WHERE**, **order by** (e order by com Limit), **group by** ou em *joins*
- A coluna contém inúmeros valores distintos
- A tabela tem muitas linhas, e as consultas normalmente retornam poucas linhas
- Existem **campos que são consultados em forma de intervalo**: Datas, siglas, códigos de chaves estrangeiras

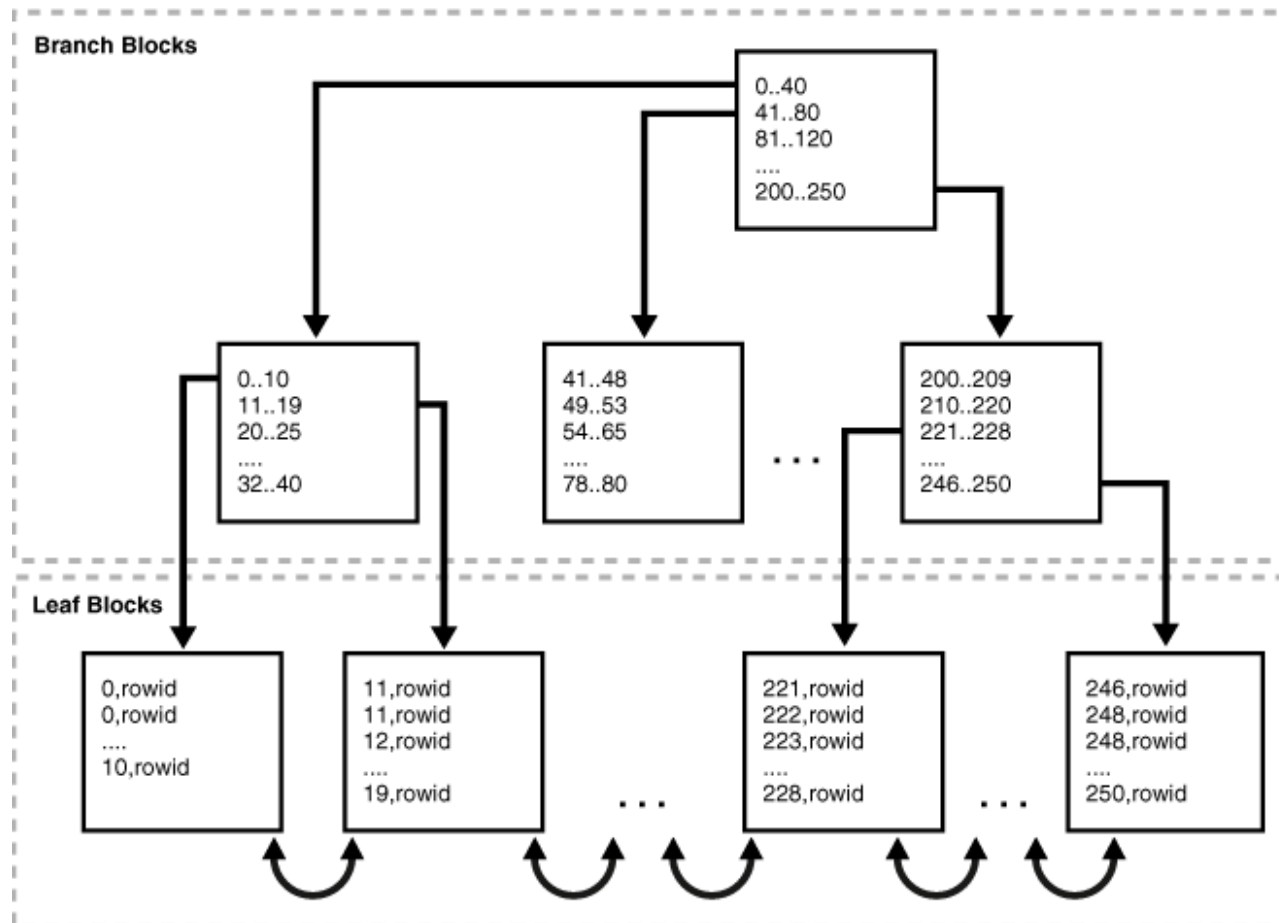
# Quando não Criar Índices?

---

- **A tabela é pequena**
  - Tabelas pequenas se ajustam facilmente ao cache do SGBD
- As colunas não aparecem em condições ou *joins*
- A tabela é atualizada muito freqüentemente
- A coluna ou colunas já possuem uma *constraint* **PK ou unique**;
- A coluna é do tipo text.

# Índice - implementação

- BTree
- Princípio básico: **dividir os dados do índice em blocos**



# Outro exemplo

---

- Pode-se ajustar um índice pela ordenação da B-tree
  - Incluindo opções como ASC, DESC, NULLS FIRST e NULLS LAST na criação do índice
    - CREATE INDEX test2\_info\_nulls\_low ON test2 (info NULLS FIRST);
    - CREATE INDEX test3\_desc\_index ON test3 (desc NULLS LAST);
- Podem satisfazer ORDER BY x ASC NULLS FIRST ou ORDER BY x DESC NULLS LAST
- Podem ser usados em múltiplas colunas também

# Carga de dados

---

- As cargas de dados volumosos devem ser realizadas sem índices
  - Por quê?
    - Quando cada linha é adicionada, a estrutura do índice é rebalanceada, e vão ser muitas linhas em sequência, o que provocará perda de performance
- O que fazer?
  - Deve-se eliminar o índice, carregar os dados e, em seguida, recriar o índice.
  - SGBDs, em geral, possuem carregadores para grandes volumes de dados que fazem isso automaticamente

# Banco em produção

---

- Na construção dos índices, a tabela é bloqueada automaticamente para instruções de inserção
- A criação de índices para as tabelas é uma operação cara
- Se precisar:
  - `CREATE INDEX CONCURRENTLY  
index_name ON table_name using btree  
(column);`