|  |  |
| --- | --- |
| LogoEfeitrans | **Banco de Dados II Prática 2**  Profa. Vanessa Souza |

**Assunto:** Indexação

* Essa prática deve ser executada no SGBD PostgreSQL.

# Questão 1:

1. Restaurar o banco logradouro
2. Qual ID do banco?32965
3. Qual ID da tabela log\_logradouro?33014
4. Quais índices a tabela log\_logradouro possui? Uma: "log\_logradouro\_pkey"

SELECT \* FROM pg\_indexes WHERE tablename = 'log\_logradouro';

**Questão 2 – Índice Primário:** Para todas os comandos solicitados, utilize o comando EXPLAIN ANALYZE e avalie a quantidade de registros recuperada, o tempo para executar o comando e método de acesso.

1. Recuperar todos os registros da tabela log\_logradouro;

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro;

"Seq Scan on log\_logradouro (cost=0.00..24012.85 rows=909585 width=146) (actual time=0.106..109.311 rows=909585 loops=1)"

"Planning Time: 0.527 ms"

"Execution Time: 126.415 ms"

1. Quais são os índices existentes na tabela log\_logradouro?

1

1. Recuperar todos os registros da tabela log\_logradouro onde log\_nu\_sequencial

< 10

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial < 10

"**Index Scan using log\_logradouro\_pkey on** log\_logradouro (cost=0.42..8.57 rows=8 width=146) (actual time=0.083..0.084 rows=3 loops=1)"

" **Index Cond:** (log\_nu\_sequencial < 10)"

"Planning Time: 0.431 ms"

"Execution Time: 0.101 ms"

* COMO SÃO POUCO REGISTROS A SEREM BUSCADOS É FEITA UMA BUSCA POR INDICE.

1. Recuperar todos os registros da tabela log\_logradouro onde log\_nu\_sequencial

>10

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial > 10

"**Seq Scan** on log\_logradouro (cost=0.00..26286.81 rows=909576 width=146) (actual time=0.010..136.039 rows=909582 loops=1)"

" **Filter:** (log\_nu\_sequencial > 10)"

" Rows Removed by Filter: 3"

"Planning Time: 0.085 ms"

"Execution Time: 154.005 ms"

* COMO SÃO MUITOS REGISTRO O PLANO DE BUSCA É FEITO SEM INDICE DE MANEIRA SEQUÊNCIAL.

1. Recuperar o registro de log\_nu\_sequencial = 15 da tabela log\_logradouro

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial = 10

"**Index Scan using log\_logradouro\_pkey** on log\_logradouro (cost=0.42..8.44 rows=1 width=146) (actual time=0.071..0.072 rows=1 loops=1)"

**" Index Cond:** (log\_nu\_sequencial = 15)"

"Planning Time: 0.224 ms"

"Execution Time: 0.106 ms"

* UTILIZA O INDICE PARA LOCALIZAR E FAZER A BUSCA DO REGISTRO DADO A CONDIÇÃO

1. Deletar o registro de log\_nu\_sequencial = 15 da tabela log\_logradouro

EXPLAIN ANALYZE **DELETE** FROM CEP.log\_logradouro WHERE log\_nu\_sequencial = 15

"**Delete** on log\_logradouro (cost=0.42..8.44 rows=1 width=6) (actual time=0.029..0.029 rows=1 loops=1)"

" -> **Index Scan** using log\_logradouro\_pkey on log\_logradouro (cost=0.42..8.44 rows=1 width=6) (actual time=0.028..0.028 rows=0 loops=1)"

" **Index Cond:** (log\_nu\_sequencial = 15)"

"Planning Time: 0.062 ms"

"Execution Time: 0.052 ms"

* UTILIZA O INDICE PARA LOCALIZAR E FAZER A EXCLUSÃO DO REGISTRO DADO A CONDIÇÃO

1. Recuperar o registro de log\_nu\_sequencial = 15 da tabela log\_logradouro

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial = 10

"**Index Scan using log\_logradouro\_pkey** on log\_logradouro (cost=0.42..8.44 rows=1 width=146) (actual time=0.012..0.013 **rows=0** loops=1)"

" **Index Cond:** (log\_nu\_sequencial = 15)"

"Planning Time: 0.065 ms"

"Execution Time: 0.026 ms"

UTILIZA MESMA ESTRUTURA DE BUSCA. FAZ A BUSCA NORMALMENTE, POREM NÃO ENCONTA O REGISTRO NO BANCO.

**Questão 3 – Índice Secundário:** Para todas os comandos solicitados, utilize o comando EXPLAIN ANALYZE E avalie a quantidade de registros recuperada, o tempo para executar o comando e método de acesso.

1. Recuperar todos os registros da tabela log\_logradouro onde ufe\_sg = 'AC' e log\_tipo\_logradouro = 'Avenida';

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE ufe\_sg = 'AC' AND log\_tipo\_logradouro = 'Avenida';

"**Gather** (cost=1000.00..21621.81 rows=199 width=146) (actual time=0.723..113.248 rows=139 loops=1)"

" Workers Planned: 2"

" Workers Launched: 2"

" **-> Parallel Seq Scan** on log\_logradouro (cost=0.00..20601.91 rows=83 width=146) (actual time=33.646..73.911 rows=46 loops=3)"

" **Filter**: (((ufe\_sg)::text = 'AC'::text) AND ((log\_tipo\_logradouro)::text = 'Avenida'::text))"

" Rows Removed by Filter: 303148"

"Planning Time: 0.087 ms"

"Execution Time: 113.273 ms"

1. Crie um índice (btree) na coluna *ufe\_sg* da tabela log\_logradouro

CREATE INDEX **func\_ufe\_sg** ON CEP.log\_logradouro USING btree (ufe\_sg );

1. Refaça a questão a.

"**Bitmap Heap Scan** on log\_logradouro (cost=36.12..7750.27 rows=199 width=146) (actual time=0.322..2.703 rows=139 loops=1)"

" Recheck Cond: ((ufe\_sg)::text = 'AC'::text)"

" Filter: ((log\_tipo\_logradouro)::text = 'Avenida'::text)"

" Rows Removed by Filter: 3126"

" **Heap Blocks**: exact=74"

" **-> Bitmap Index Scan** on **func\_ufe\_sg** (cost=0.00..36.07 rows=3153 width=0) (actual time=0.307..0.308 rows=3265 loops=1)"

" **Index Cond**: ((ufe\_sg)::text = 'AC'::text)"

"**Planning Time: 0.528 ms"**

**"Execution Time: 2.737 ms"**

**MELHOU MUITO A AGILIDADE DA BUSCA AO UTILIZAR O INDICE E O BITMAP.**

1. Crie um índice (btree) na coluna *log\_tipo\_logradouro* da tabela log\_logradouro

CREATE INDEX **func\_log\_tipo\_logradouro** ON CEP.log\_logradouro USING btree

(log\_tipo\_logradouro);

1. Refaça a questão a.

"**Bitmap Heap Scan on log\_logradouro** (cost=662.39..1388.94 rows=199 width=146) (actual time=3.648..3.699 rows=139 loops=1)"

" Recheck Cond: (((ufe\_sg)::text = 'AC'::text) AND ((log\_tipo\_logradouro)::text = 'Avenida'::text))"

**" Heap Blocks**: exact=29"

" -> **BitmapAnd**  (cost=662.39..662.39 rows=199 width=0) (actual time=3.639..3.640 **rows=0** loops=1)"

" -> **Bitmap Index Scan** on func\_ufe\_sg (cost=0.00..36.07 rows=3153 width=0) (actual time=0.085..0.085 rows=3265 loops=1)"

" Index Cond: ((ufe\_sg)::text = 'AC'::text)"

" -> **Bitmap Index Scan** on func\_log\_tipo\_logradouro (cost=0.00..625.97 rows=57273 width=0) (actual time=3.396..3.396 rows=56624 loops=1)"

**" Index Cond:** ((log\_tipo\_logradouro)::text = 'Avenida'::text)"

"Planning Time: 0.084 ms"

"Execution Time: 2.618 ms"

* MELHOROU UM POUCO O RESULTADO ( COMO OS DADOS FORAM INSERIDOS COM ALEATORIDADE NO BANCO POSSA INTERFERIR).
* FOI REALIZADO DOIS BITMAP SENDO UM CONTENDO OS INDICES DE UFE\_SG E O OUTRO CONTEDO OS INDICES LOG\_TIPO\_LOGADOURO), DEPOIS E REALIZADO A INTERSEÇÃO DE AMBOS, OU SEJA, SO QUANDO AMBAS AS CONDIÇÕES FOREM ESTABELECIDAS O BITMAP PERMANECERÁ COM O REGISTRO PRÉ-SELECIONADO, EM SEGUIDA SERÁ REALIZADA A VERREDURA FINAL CONFIRMANDO AMBAS AS CONDIÇÕES ESTABELECIDAS.

**Questão 4 – Índice Composto:** Para todas os comandos solicitados, utilize o comando EXPLAIN ANALYZE E avalie a quantidade de registros recuperada, o tempo para executar o comando e método de acesso.

1. Apague os índices criados na letra b e na letra d.

DROP INDEX func\_ufe\_sg ;

DROP INDEX func\_log\_tipo\_logradouro;

1. Crie um índice composto utilizando os atributos *ufe\_sg* e *log\_tipo\_logradouro*

*CREATE INDEX composto ON CEP.log\_logradouro USING btree (ufe\_sg,log\_tipo\_logradouro);*

1. Refaça a questão 2ª
2. EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro;

"Seq Scan on log\_logradouro (cost=0.00..24012.84 rows=909584 width=146) (actual time=0.011..109.438 rows=909584 loops=1)"

"Planning Time: 0.052 ms"

"Execution Time: 127.989 ms"

1. 2 - log\_logradouro\_pkey e *composto*
2. EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial < 10

"Index Scan using log\_logradouro\_pkey on log\_logradouro (cost=0.42..8.57 rows=8 width=146) (actual time=0.004..0.004 rows=3 loops=1)"

" Index Cond: (log\_nu\_sequencial < 10)"

"Planning Time: 0.140 ms"

"Execution Time: 0.019 ms"

1. EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial > 10

"Seq Scan on log\_logradouro (cost=0.00..26286.80 rows=909575 width=146) (actual time=0.018..167.530 rows=909581 loops=1)"

" Filter: (log\_nu\_sequencial > 10)"

" Rows Removed by Filter: 3"

"Planning Time: 0.177 ms"

"Execution Time: 187.413 ms"

1. EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_nu\_sequencial = 15

"Index Scan using log\_logradouro\_pkey on log\_logradouro (cost=0.42..8.44 rows=1 width=146) (actual time=0.024..0.024 rows=0 loops=1)"

" Index Cond: (log\_nu\_sequencial = 15)"

"Planning Time: 0.100 ms"

"Execution Time: 0.042 ms"

1. EXPLAIN ANALYZE DELETE FROM CEP.log\_logradouro WHERE log\_nu\_sequencial = 15

"Delete on log\_logradouro (cost=0.42..8.44 rows=1 width=6) (actual time=0.012..0.013 rows=0 loops=1)"

" -> Index Scan using log\_logradouro\_pkey on log\_logradouro (cost=0.42..8.44 rows=1 width=6) (actual time=0.010..0.010 rows=0 loops=1)"

" Index Cond: (log\_nu\_sequencial = 15)"

"Planning Time: 0.069 ms"

"Execution Time: 0.035 ms"

1. Compare os resultados das questões 2a, 2c, 2e e 3c

2a= iguais, mesma estrutura de busca

2c= iguais, mesma estrutura de busca

2e= iguais, mesma estrutura de busca

3c= diferente, havia sido feito uma interseção de 2 bitmap e agora foi realizado apenas uma busca pelo indice composto, o que fez melhorar o tempo de execução.

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE ufe\_sg = 'AC' AND log\_tipo\_logradouro = 'Avenida';

"**Index Scan using composto** on log\_logradouro (cost=0.42..736.95 rows=199 width=146) (actual time=0.215..0.287 rows=139 loops=1)"

**" Index Cond:** (((ufe\_sg)::text = 'AC'::text) AND ((log\_tipo\_logradouro)::text = 'Avenida'::text))"

"Planning Time: 0.102 ms"

"Execution Time: 0.305 ms"

1. Recuperar todos os registros da tabela log\_logradouro onde ufe\_sg = 'MG'

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE ufe\_sg = 'MG';

"Bitmap Heap Scan on log\_logradouro (cost=866.44..16747.98 rows=77163 width=146) (actual time=3.607..30.379 rows=77288 loops=1)"

" Recheck Cond: ((ufe\_sg)::text = 'MG'::text)"

" Heap Blocks: exact=1349"

" -> **Bitmap Index** Scan on composto (cost=0.00..847.15 rows=77163 width=0) (actual time=3.500..3.501 rows=77288 loops=1)"

" **Index Cond**: ((**ufe\_sg**)::text = 'MG'::text)"

"Planning Time: 0.105 ms"

"Execution Time: 31.888 ms"

* **Reliza um busca utilizando o bitmap com indice.**

1. Recuperar todos os registros da tabela log\_logradouro onde log\_tipo\_logradouro = 'Avenida';

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE log\_tipo\_logradouro = 'Avenida';

"**Bitmap Heap Scan on log\_logradouro** (cost=9972.62..25605.54 rows=57273 width=146) (actual time=22.552..192.503 rows=56624 loops=1)"

" **Recheck Cond**: ((log\_tipo\_logradouro)::text = 'Avenida'::text)"

" **Heap Blocks:** exact=11468"

" -> **Bitmap Index Scan** **on composto** (cost=0.00..9958.31 rows=57273 width=0) (actual time=21.415..21.415 rows=56624 loops=1)"

" **Index Cond: ((log\_tipo\_logradouro)::**text = 'Avenida'::text)"

"Planning Time: 0.136 ms"

"Execution Time: 193.783 ms"

* **Também Reliza um busca utilizando o bitmap com indice.**

1. Recuperar todos os registros da tabela log\_logradouro onde ufe\_sg = 'AC' OU log\_tipo\_logradouro = 'Avenida';

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE ufe\_sg = 'AC' or log\_tipo\_logradouro = 'Avenida';

"**Bitmap Heap Scan on log\_logradouro** (cost=10024.49..25847.90 rows=60228 width=146) (actual time=6.496..26.883 rows=59750 loops=1)"

" Recheck Cond: (((ufe\_sg)::text = 'AC'::text) **OR** ((log\_tipo\_logradouro)::text = 'Avenida'::text))"

" **Heap Blocks**: exact=11493"

" **-> BitmapOr** (cost=10024.49..10024.49 rows=60427 width=0) (actual time=5.059..5.060 **rows=0** loops=1)"

" -> **Bitmap Index Scan on composto** (cost=0.00..36.07 rows=3153 width=0) (actual time=0.133..0.134 rows=3265 loops=1)"

" **Index Cond**: ((ufe\_sg)::text = 'AC'::text)"

" **-> Bitmap Index Scan on composto** (cost=0.00..9958.31 rows=57273 width=0) (actual time=4.924..4.925 rows=56624 loops=1)"

" **Index Cond:** ((log\_tipo\_logradouro)::text = 'Avenida'::text)"

"Planning Time: 0.149 ms"

"Execution Time: 28.508 ms"

**SÃO REALIZADOS DOIS BITMAP UM COM CADA UM DOS ATRIBUTOS DO INDICE COMPOSTO, DEPOIS É FEITA UNIAO DE AMBAS AS PRE SELEÇÕES DE CADA BITMAP, OU SEJA, SERÁ REALIZADA A VARREDURA FAZENDO UMA REVERIFICAÇÃO DAS CONDIÇOES NOS PRE-SELICIONADOS DE DE AMBOS OS ATRIBUTOS.**

**Questão 5 – Comandos Diversos:** Para todas os comandos solicitados, utilize o comando EXPLAIN ANALYZE E avalie a quantidade de registros recuperada, o tempo para executar o comando e método de acesso.

1. Recuperar os 20 primeiros registros da tabela log\_logradouro onde ufe\_sg = 'AC' e log\_tipo\_logradouro = 'Avenida';

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro WHERE ufe\_sg = 'AC' AND log\_tipo\_logradouro = 'Avenida' LIMIT 20;

"**Limit** (cost=0.42..74.45 rows=20 width=146) (actual time=0.038..0.049 rows=20 loops=1)"

" -> **Index Scan** **using composto** on log\_logradouro (cost=0.42..736.95 rows=199 width=146) (actual time=0.037..0.047 rows=20 loops=1)"

" **Index Cond: (((ufe\_sg)::text = 'AC'::text) AND ((log\_tipo\_logradouro)::**

text = 'Avenida'::text))"

"**Planning Time: 0.114 ms"**

**"Execution Time: 0.071 ms"**

* **UTILIZA APENAS O INDICE PRA FAZER A VARREDURA E DADO A QUANTIDADE BAIXA DE REGISTROS RECUPERADOS RETORNADO É BEM RAPIDA SUA EXECUÇÃO**

1. Refaça a questão 2a ordenando os resultados pelo atributo cep.

EXPLAIN ANALYZE SELECT \* FROM CEP.log\_logradouro ORDER BY (CEP);

"**Gather Merge** (cost=111825.09..200262.97 rows=757986 width=146) (actual time=1370.064..2197.647 rows=909584 loops=1)"

" Workers Planned: 2"

" Workers Launched: 2"

" -> **Sort** (cost=110825.07..111772.55 rows=378993 width=146) (actual time=1311.433..1533.029 rows=303195 loops=3)"

" Sort Key: cep"

" Sort Method: external merge Disk: 34232kB"

" Worker 0: Sort Method: external merge Disk: 33464kB"

" Worker 1: Sort Method: external merge Disk: 31888kB"

" -> **Parallel Seq** **Scan** on log\_logradouro (cost=0.00..18706.93 rows=378993 width=146) (actual time=0.016..27.723 rows=303195 loops=3)"

**"Planning Time: 0.087 ms"**

**"Execution Time: 2222.561 ms"**

**É realizado um conjunto de busca sequencial dos registros de modo que cada**

**conjunto possa trabalhar em paralelo e depois ordenado pelo sort e por**

**fim juntado (merge) de modo a preservar a ordem.**

1. Avalie a consulta :

EXPLAIN ANALYZE

SELECT bai\_nu\_sequencial\_ini, COUNT(bai\_nu\_sequencial\_ini) FROM cep.log\_logradouro

WHERE ufe\_sg = 'AC' and log\_tipo\_logradouro = 'Avenida' GROUP BY bai\_nu\_sequencial\_ini;

"**HashAggregate**  (cost=737.94..739.92 rows=198 width=12) (actual time=0.102..0.108 rows=59 loops=1)"

" **Group Key:** bai\_nu\_sequencial\_ini"

**" Batches: 1 Memory Usage: 40kB**"

" -> **Index Scan using composto** on log\_logradouro (cost=0.42..736.95 rows=199 width=4) (actual time=0.023..0.059 rows=139 loops=1)"

" **Index Cond: (((ufe\_sg)::text = 'AC'::text) AND ((log\_tipo\_logradouro**)::text = 'Avenida'::text))"

"Planning Time: 0.086 ms"

"Execution Time: 0.142 ms"

**Neste caso é utilizado o indice para a varredura e uma tabela hash de 40kb interna para armazenar os dados.**

1. Avalie a consulta :

EXPLAIN ANALYZE

SELECT \* FROM cep.log\_bairro WHERE bai\_nu\_sequencial IN (SELECT bai\_nu\_sequencial

FROM cep.log\_faixa\_bairro WHERE fcb\_rad\_ini = '69918' )

"**Nested Loop** (cost=1832.10..1906.67 rows=9 width=38) (actual time=281.373..281.451 rows=21 loops=1)"

" **-> HashAggregate** (cost=1831.81..1831.90 rows=9 width=4) (actual time=281.345..281.349 rows=21 loops=1)"

" Group Key: log\_faixa\_bairro.bai\_nu\_sequencial"

" Batches: 1 Memory Usage: 24kB"

" -> **Seq Scan** on log\_faixa\_bairro (cost=0.00..1831.79 rows=9 width=4) (actual time=0.022..281.328 rows=21 loops=1)"

**" Filter: ((fcb\_rad\_ini**)::text = '69918'::text)"

" Rows Removed by Filter: 92202"

" -> **Index Scan using log\_bairro\_pkey** on log\_bairro (cost=0.29..8.31 rows=1 width=38) (actual time=0.004..0.004 **rows=1** loops=21)"

" **Index Cond:** (bai\_nu\_sequencial = log\_faixa\_bairro.bai\_nu\_sequencial)"

"**Planning Time: 458.641 ms"**

**"Execution Time: 281.487 ms"**

Utiliza o indice para buscar os registros que apresenta **bai\_nu\_sequencial** da tabela log\_bairro igual da tabela log\_faixa\_bairro, em seguida será varrido **de forma sequencial** na tabela log\_faixa\_bairro os registros que apresentam bai\_nu\_sequencial **conforme o filtro** fcb\_rad\_ini = '69918' e por fim são guardados os registros numa hash interna.o nested loop indica que para cada linha da tabela externa, deve-se percorrer todas as linhas da tabela interna e verificar se elas correspondem à condição de junção. **Como so existia um unico bai\_nu\_sequencial no laço mais externo, então este únicoregistro foi utilizado para comparar com todos os registros do nó interno, a fim de estabelecer a condiçaõ do filtro interno. Ao final os registros que satisfezeram ambos os laços são armazenados na tabela hash.**

# Questão 6 – Tamanho dos arquivos físicos:

a) Teste os comandos:

* SELECT pg\_size\_pretty (pg\_relation\_size('nome do índice criado na questão 4b’));
* SELECT pg\_size\_pretty(pg\_relation\_size('cep.composto'))= **6272 kb**
* SELECT pg\_size\_prettY(pg\_relation\_size('cep.log\_logradouro')); = **117 MB**
* **SELECT pg\_size\_pretty(pg\_total\_relation\_size('**cep.log\_logradouro**'));** = **142 MB**

# LEITURAS SUGERIDAS:

* <https://www.postgresql.org/docs/9.2/using-explain.html>
* <https://use-the-index-luke.com/sql/explain-plan/postgresql/operations>