Tarefa

Módulo 8

Atividade 8.a – Índice com operador LIKE

- Conecte na base benchmark;
- Execute a seguinte query com explain:

EXPLAIN SELECT * FROM pgbench_accounts WHERE filler LIKE ‘ab%’;

```
benchmark=# EXPLAIN SELECT * FROM pgbench_accounts WHERE filler LIKE 'ab%';
                                  QUERY PLAN
------------------------------------------------------------------------------
 Gather  (cost=1000.00..218657.43 rows=1 width=97)
   Workers Planned: 2
   ->  Parallel Seq Scan on pgbench_accounts  (cost=0.00..217657.33 rows=1 width=97)
         Filter: (filler ~~ 'ab%'::text)
(4 rows)

benchmark=#
```

- Analise o plano de execução;
  Dica: antes do próximo passo, para uma execução mais rápida, execute o do create index o comando: SET maintenance_work_mem = ‘512MB’ por exemplo.

```
benchmark=# ALTER SYSTEM SET maintenance_work_mem = '512MB';
ALTER SYSTEM
benchmark=# select pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)
```

- Crie um índice na coluna filler;

```
benchmark=# CREATE INDEX idx_like ON pgbench_accounts(filler);
CREATE INDEX
```

- Execute a query novamente. O plano de execução foi alterado?

```
                                  QUERY PLAN
------------------------------------------------------------------------------
 Gather  (cost=1000.00..218657.43 rows=1 width=97)
   Workers Planned: 2
   ->  Parallel Seq Scan on pgbench_accounts  (cost=0.00..217657.33 rows=1 width=97)
         Filter: (filler ~~ 'ab%'::text)
(4 rows)
Não houve alteração do plano de execução
```

- Crie um índice na coluna filler usando um operador de classe de acordo com o tipo da coluna, como mostrado durante a sessão

```
benchmark=# CREATE INDEX idx_like2 ON pgbench_accounts(filler bpchar_pattern_ops);
CREATE INDEX
```

- Execute o explain novamente e analise o plano de execução. O que mudou?

```
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------
 Index Scan using idx_like2 on pgbench_accounts  (cost=0.56..8.58 rows=1 width=97)
   Index Cond: ((filler ~>=~ 'ab'::bpchar) AND (filler ~<~ 'ac'::bpchar))
   Filter: (filler ~~ 'ab%'::text)
(3 rows)
```

Atividade 8.b – FTS-Full-Text Search

- Execute os passos exemplos da explicação sobre FTS na base curso
- Crie a nova coluna para ser usada com FTS;

```
curso=# ALTER TABLE times ADD COLUMN historia_fts tsvector;
ALTER TABLE
```

- Carregue os dados desta coluna;

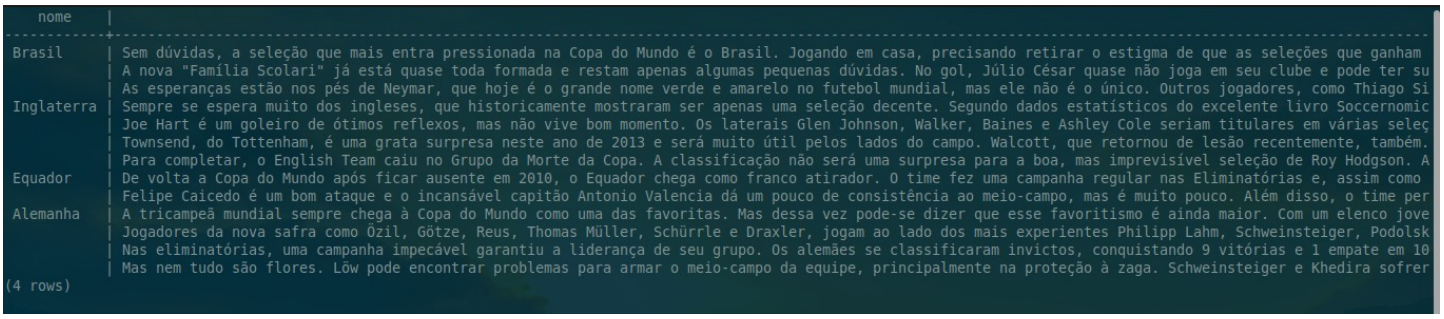
```
curso=# UPDATE times SET historia_fts = to_tsvector('portuguese', historia);
UPDATE 32
```

- Crie o índice com o tipo apropriado;

```
curso=# CREATE INDEX idx_historia_fts ON times USING GIN(historia_fts);
CREATE INDEX
```

- Execute a consulta.

```
SELECT nome, historia FROM times WHERE historia_fts @@ to_tsquery ('portuguese','campo & mundo');
```



```
    nome    |
------------+-------------------------------------------------------------------------------------------------------------------------------------------------------
 Brasil     | Sem dúvidas, a seleção que mais entra pressionada na Copa do Mundo é o Brasil. Jogando em casa, precisando retirar o estigma de que as seleções que ganham
            | A nova "Família Scolari" já está quase toda formada e restam apenas algumas pequenas dúvidas. No gol, Júlio César quase não joga em seu clube e pode ter su
            | As esperanças estão nos pés de Neymar, que hoje é o grande nome verde e amarelo no futebol mundial, mas ele não é o único. Outros jogadores, como Thiago Si
 Inglaterra | Sempre se espera muito dos ingleses, que historicamente mostraram ser apenas uma seleção decente. Segundo dados estatísticos do excelente livro Soccernomic
            | Joe Hart é um goleiro de ótimos reflexos, mas não vive bom momento. Os laterais Glen Johnson, Walker, Baines e Ashley Cole seriam titulares em várias seleç
            | Townsend, do Tottenham, é uma grata surpresa neste ano de 2013 e será muito útil pelos lados do campo. Walcott, que retornou de lesão recentemente, também.
            | Para completar, o English Team caiu no Grupo da Morte da Copa. A classificação não será uma surpresa para a boa, mas imprevisível seleção de Roy Hodgson. A
 Equador    | De volta a Copa do Mundo após ficar ausente em 2010, o Equador chega como franco atirador. O time fez uma campanha regular nas Eliminatórias e, assim como
            | Felipe Caicedo é um bom ataque e o incansável capitão Antonio Valencia dá um pouco de consistência ao meio-campo, mas é muito pouco. Além disso, o time per
 Alemanha   | A tricampeã mundial sempre chega à Copa do Mundo como uma das favoritas. Mas dessa vez pode-se dizer que esse favoritismo é ainda maior. Com um elenco jove
            | Jogadores da nova safra como Özil, Götze, Reus, Thomas Müller, Schürrle e Draxler, jogam ao lado dos mais experientes Philipp Lahm, Schweinsteiger, Podolsk
            | Nas eliminatórias, uma campanha impecável garantiu a liderança de seu grupo. Os alemães se classificaram invictos, conquistando 9 vitórias e 1 empate em 10
            | Mas nem tudo são flores. Löw pode encontrar problemas para armar o meio-campo da equipe, principalmente na proteção à zaga. Schweinsteiger e Khedira sofrer
(4 rows)
```

- Examine se e como as novas generated columns poderiam ajudar no FTS

```
As Generated Columns podem diminuir o overhead de leitura, acessando a informação já computada direto da tabela
```

Atividade 8.c – Particionamento Nativo

Abra uma conexão com a base curso

```
postgres@debian10:~$ psql -d curso
psql (13.1)
Type "help" for help.

curso=#
```

Execute os passos do exemplo de Particionamento Declarativo.

- Crie a tabela principal e três tabelas filhas, para os anos 2012, 2013 e 2014;

```
curso=# CREATE TABLE item_financeiro (iditem int, data timestamp, descricao varchar(50),
valor numeric(10,2)) PARTITION BY RANGE (data);
CREATE TABLE
curso=# CREATE TABLE item_financeiro_2012 PARTITION OF item_financeiro FOR VALUES FROM
('2012-01-01') TO ('2013-01-01');
CREATE TABLE
curso=# CREATE TABLE item_financeiro_2013 PARTITION OF item_financeiro FOR VALUES FROM
('2013-01-01') TO ('2014-01-01');
CREATE TABLE
curso=# CREATE TABLE item_financeiro_2014 PARTITION OF item_financeiro FOR VALUES FROM
('2014-01-01') TO ('2015-01-01');
CREATE TABLE
curso=# \dt
                    List of relations
 Schema |         Name          |       Type        |  Owner
--------+-----------------------+-------------------+----------
 public | cidades               | table             | postgres
 public | grupos                | table             | postgres
 public | grupos_times          | table             | postgres
 public | item_financeiro       | partitioned table | postgres
 public | item_financeiro_2012  | table             | postgres
 public | item_financeiro_2013  | table             | postgres
 public | item_financeiro_2014  | table             | postgres
 public | jogos                 | table             | postgres
 public | taba                  | table             | postgres
 public | tabb                  | table             | postgres
 public | times                 | table             | postgres
(11 rows)
```

- Crie os índices;

```
curso=# CREATE INDEX ON item_financeiro(data);
CREATE INDEX
```

- Execute o seguinte script para gerar 100 mil registros aleatórios:

```
INSERT INTO item_financeiro SELECT generate_series(1,100000), timestamp '2012-01-
01' + random() * (timestamp '2015-01-01' - timestamp '2012-01-
01'),'descrição...', (random()*1000)::numeric(10,2);
```

```
curso=# INSERT INTO item_financeiro SELECT generate_series(1,100000), timestamp '2012-01-
01' + random() * (timestamp '2015-01-01' - timestamp '2012-01-01'), 'descrição...',
(random()*1000)::numeric(10,2);
INSERT 0 100000
```

- Atualize as estatísticas da tabela principal (ANALYZE VERBOSE)

```
curso=# analyze verbose item_financeiro;
INFO:  analyzing "public.item_financeiro" inheritance tree
INFO:  "item_financeiro_2012": scanned 278 of 278 pages, containing 33312 live rows and 0
dead rows; 10000 rows in sample, 33312 estimated total rows
INFO:  "item_financeiro_2013": scanned 278 of 278 pages, containing 33329 live rows and 0
dead rows; 10000 rows in sample, 33329 estimated total rows
INFO:  "item_financeiro_2014": scanned 278 of 278 pages, containing 33359 live rows and 0
dead rows; 10000 rows in sample, 33359 estimated total rows
INFO:  analyzing "public.item_financeiro_2012"
INFO:  "item_financeiro_2012": scanned 278 of 278 pages, containing 33312 live rows and 0
dead rows; 30000 rows in sample, 33312 estimated total rows
INFO:  analyzing "public.item_financeiro_2013"
INFO:  "item_financeiro_2013": scanned 278 of 278 pages, containing 33329 live rows and 0
dead rows; 30000 rows in sample, 33329 estimated total rows
INFO:  analyzing "public.item_financeiro_2014"
INFO:  "item_financeiro_2014": scanned 278 of 278 pages, containing 33359 live rows and 0
dead rows; 30000 rows in sample, 33359 estimated total rows
ANALYZE
curso=#
```

- Analise o conteúdo das tabelas pai e filha;
- Execute o EXPLAIN ANALYZE em consultas na tabela pai para ver o plano de execução:
  - Procure por todos os registros e veja o plano

```
curso=# explain (analyze)
curso-# select * from item_financeiro;
                                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Append  (cost=0.00..2334.00 rows=100000 width=33) (actual time=0.015..15.724 rows=100000 loops=1)
   ->  Seq Scan on item_financeiro_2012 item_financeiro_1  (cost=0.00..611.12 rows=33312 width=33) (actual time=0.014..2.822
rows=33312 loops=1)
   ->  Seq Scan on item_financeiro_2013 item_financeiro_2  (cost=0.00..611.29 rows=33329 width=33) (actual time=0.011..2.402
rows=33329 loops=1)
   ->  Seq Scan on item_financeiro_2014 item_financeiro_3  (cost=0.00..611.59 rows=33359 width=33) (actual time=0.012..2.996
rows=33359 loops=1)
 Planning Time: 0.135 ms
 Execution Time: 19.861 ms
(6 rows)

curso=#
```

  - Procure por um registro dentro de um dia específico
  - Procure por registros em uma faixa de datas entre 2012 e 2013, por exemplo

```
curso=# explain analyze
curso-# select * from item_financeiro where data = '2013-03-13 11:34:31.34768';
                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Index Scan using item_financeiro_2013_data_idx on item_financeiro_2013 item_financeiro
(cost=0.29..8.31 rows=1 width=33) (actual time=0.009..0.010 rows=1 loops=1)
   Index Cond: (data = '2013-03-13 11:34:31.34768'::timestamp without time zone)
 Planning Time: 0.086 ms
 Execution Time: 0.024 ms
(4 rows)
```

Atividade 8.d - Remover dados de tabelas particionadas

Ainda no contexto do exercício anterior:
- Apague todos os registros de 2012.
- Qual a forma mais eficiente?

```
curso=# drop table item_financeiro_2012;
DROP TABLE
```

Mais eficiente será o drop na tabela referente a partição

Atividade 8.e – Estimar effective_cache_size

Baseado no que foi explicado, calcule qual o effective_cache_size para seu servidor.

some o valor do parâmetro shared_buffers ao valor observado da memória sendo usada para cache em seu servidor

```
postgres@debian10:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:          987Mi        67Mi       760Mi        33Mi       159Mi       753Mi
Swap:         1.9Gi       9.0Mi       1.9Gi
```

Valor: 33M + 159M = 192M (essa fiquei na dúvida)

Atividade 8.f - Paralelismo de Queries

- Conecte-se na base benchmark
- Altere a configuração necessária para permitir até 4 processos paralelos por query

```
benchmark=# alter system set max_parallel_workers_per_gather = 4;
ALTER SYSTEM
```

- Exiba o plano de execução de uma query calculando a média da coluna abalance da tabela pgbench_accounts

```
benchmark=# explain analyze
benchmark-# select avg(abalance) from pgbench_accounts;
                                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=218657.55..218657.56 rows=1 width=32) (actual time=19047.747..19047.852 rows=1 loops=1)
   ->  Gather  (cost=218657.33..218657.54 rows=2 width=32) (actual time=19044.979..19047.831 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=217657.33..217657.34 rows=1 width=32) (actual time=19035.052..19035.053 rows=1 loops=3)
               ->  Parallel Seq Scan on pgbench_accounts  (cost=0.00..207240.67 rows=4166667 width=4) (actual time=27.545..18124.842 rows=3333333 loops=3)
 Planning Time: 0.155 ms
 Execution Time: 19047.931 ms
(8 rows)
```

- Houve paralelismo na execução da query ?

```
...
Workers Planned: 2
Workers Launched: 2
...
```

Sim

Atividade 8.g – Estruturação do Banco

Suponha um ambiente com uma base de dados de um sistema de vendas online.

- Ele possui uma tabela muito grande de pedidos com dados de um longo tempo;
- Possui uma tabela de produtos absurdamente acessada, por dezenas de índices extremamente usados;
- Por regras de auditoria absolutamente tudo deve ser registrado no log de erros e informações, incluindo todas as queries.
- Sugira como estruturar esse banco. O que pode ser feito no nível de tabelas, partições, filesystem e organização, além de tipos de discos?

Quando ao sistema de arquivos pode-se utilizar o ext4, que garante uma boa performance, além da configuração da diretiva **noatime** para não registrar a hora de acesso dos arquivos do banco de dados. Para a tabela de pedidos, podemos criar partições agrupando por ano, melhorando a acesso aos registros. Os discos podem ser organizados em **RAID 1+0** garantindo desempenho e segurança, apesar do grande consumo de espaço para se fazer o espelhamento.