Trabalho 8

Relatório

SCC-541 Laboratório de Bases de Dados

Leonardo Gonçalves Chahud - 5266649
Murilo Franchi - 9790760
Rafael Dantas - 12563686

Prof. Dr Caetano Traina Jr.
PAE: Igor Alberte R. Eleutério

Instruções

Considerando as Questões 1 e 2, devem ser definidos índices que ajudem a responder às consultas mais rapidamente. Depois, esses índices deverão ser avaliados quanto à sua eficiência. Devem ser utilizados um índice B-tree e um índice Hash, um para cada questão. A equipe deverá escolher um deles para cada questão e justificar sua escolha no relatório.

Considere os seguintes pontos:

- Avalie se é interessante utilizar as cláusulas **INCLUDE** e **WHERE** nos índices definidos. Lembre-se de justificar as escolhas no relatório.

- Para medir o tempo de execução de uma consulta, é interessante desenvolver uma função em PL/SQL para executar cada consulta 100 vezes e apresentar o tempo médio gasto. A função definida a seguir mostra um exemplo de como esse procedimento pode ser implementado (lembre-se que esse exemplo talvez precise ser adaptado às necessidades de cada questão).

- O procedimento de testes deve ser o seguinte:

    1. Definir uma consulta que contemple as condições de cada questão.
    2. Executar a função **Mede_tempo** antes de criar o índice, passando a consulta como parâmetro.
    3. Criar o índice.
    4. Executar novamente a função mede tempo para avaliar tempo da nova execução.

Dica: o comando **EXPLAIN ANALYZE <query>** permite avaliar como um comando <query> é executado pelo SGBD. Por exemplo, caso apareça **Seq_Scan** é feita uma leitura sequencial. Caso apareça **Index_Scan using <nome>** é utilizado o índice que tem o nome **<nome>**.

Exemplo:

```
create or replace function mede_tempo(q text)
      returns table (name text, nationality text) as
      $$
      declare
            ti time;
            tf time;
            i double precision;
            dif bigint;
      begin
            --registra o tempo inicial
            ti := clock_timestamp();
            for i in 0..100
            loop
                  execute q;
            end loop;

            --registra o tempo final
            tf := clock_timestamp();

            --calcula a diferenca em milisegundos
            dif := round(
                  (extract(epoch from tf) - extract(epoch from ti))/10
            );
            raise notice '% - % = %', tf, ti, dif;

            --retorna o resultado da consulta recebida
            return query execute q;
      end;
      $$language plpgsql;
```

Exercício 1

Dado o nome exato (**<nome>=Forename|Surname**), recuperar a nacionalidade do piloto.

Função para medir tempo:

```
create or replace function mede_tempo1(q text)
      returns table (nome text, nationality text) as
      $$
      declare
            ti time;
            tf time;
            i double precision;
            dif float;
      begin
            --registra o tempo inicial
            ti := clock_timestamp();
            for i in 0..100
            loop
                  execute q;
            end loop;

            --registra o tempo final
            tf := clock_timestamp();

            --calcula a diferenca
            dif := extract(epoch from tf) - extract(epoch from ti);
            raise notice '% - % = %', tf, ti, dif;

            --retorna o resultado da consulta recebida
            return query execute q;
      end;
      $$language plpgsql;
```

Função:

```
create or replace function nacionalidade_do_piloto(nome text, sobrenome
text)
      returns table(piloto text, nacionalidade text) as
      $$
      begin
            return query
                  select d.forename || ' ' || d.surname as name,
d.nationality
                        from driver d
                        where d.forename = nome and d.surname =
sobrenome;
      end;
      $$ language plpgsql;
```

```
select * from mede_tempo1('select * from
nacionalidade_do_piloto(''Sebastian'', ''Vettel'')');
```

Resultado antes do índice:

```
12:46:34.011909 - 12:46:33.991667 = 0.020242
```

Índice:

```
create index idx_driver_nationality
      on driver(forename, surname)
      include (nationality);
```

Resultado depois do índice:

```
12:46:59.283986 - 12:46:59.280174 = 0.003812
```

Exercício 2

Dado um um nome completo ou o padrão do início de um nome de uma ou mais cidades brasileiras, recuperar a Latitude, Longitude e População das cidades brasileiras que atendam ao nome ou padrão. Teste em uma consulta do tipo: **WHERE name LIKE 'nome%'**.

Função para medir tempo:

```
create or replace function mede_tempo2(q text)
      returns table (cidade text, lat numeric(13,5), lon numeric(13,5),
pop bigint) as
      $$
      declare
            ti time;
            tf time;
            i double precision;
            dif float;
      begin
            --registra o tempo inicial
            ti := clock_timestamp();
            for i in 0..100
            loop
                  execute q;
            end loop;

            --registra o tempo final
            tf := clock_timestamp();

            --calcula a diferenca
            dif := extract(epoch from tf) - extract(epoch from ti);
            raise notice '% - % = %', tf, ti, dif;

            --retorna o resultado da consulta recebida
            return query execute q;
      end;
      $$language plpgsql;
```

Função:

```
create or replace function info_cidade(nome text)
      returns table(cidade text, lat numeric(13,5), lon numeric(13,5),
pop bigint) as
      $$
      begin
            return query
                  select gc.name, gc.lat, gc.long, gc.population
                        from geocities15k gc
                        where gc.country = 'BR' and gc.name like (nome ||
'%');
      end
      $$language plpgsql;
```

```
select * from mede_tempo2('select * from info_cidade(''Rio'')');
```

Resultado antes do índice:

```
16:34:13.166589 - 16:34:12.801 = 0.365589
```

Índice:

```
create index idx_geocities15k_coord_pop
      on geocities15k(name)
      include (lat, long, population)
      where country = 'BR';
```

Resultado depois do índice:

```
16:34:26.041395 - 16:34:26.0209 = 0.020495
```

Exercício 3

Estruturas B-trees conseguem indexar consultas com predicados do tipo: `<atributo> LIKE ''%valor%''` ? (Considere que o atributo seja do tipo **TEXT**). Comente.

A estrutura de indexação B-tree só consegue indexar usando os operadores **like** ou **~** se o valor for constante ou se o valor for o começo de uma string.

Exemplo:

```
col like 'foo%' or col ~ '^foo'
```

Documentação:

- PostgreSQL: Documentation: 15: 11.2. Index Types