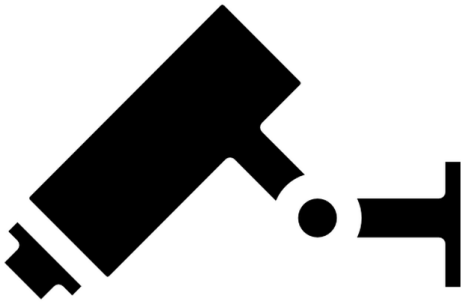
PROJETO BASE DE DADOS 2018/19



NOME	NÚMERO	ESFORÇO
Francisco Santos	86421	10 horas
João Torres	87540	10 horas
Yuri Dagot	82318	10 horas

Grupo 81 - Turno – Quarta Feira 09:30

LEIC-A

INSTITUTO SUPERIOR TÉCNICO

Relatório Parte 4

RESTRIÇÕES DE INTEGRIDADE

a)

```
DROP TRIGGER IF EXISTS chk_alocado_trigger AS alocado;

CREATE OR REPLACE FUNCTION chk_alocado() returns Trigger AS $$
BEGIN
  DECLARE n int;
  SET n = (
    SELECT COUNT(numMeio)
    FROM meioApoio NATURAL JOIN Accionado AS table
    WHERE new.numMeio = table.numMeio AND new.numProcessoSocorro= table.numProcessoSocorro
    );
  IF n < 1 THEN
    RAISE EXCEPTION 'Nonexistent Half',
    USING HINT = 'Check if Half alocaded to the HelpProcess'
  END if;
end;
$$ Language plpgsql;

CREATE TRIGGER chk_alocado_trigger after update on alocado
FOR EACH ROW EXECUTE PROCEDURE chk_alocado();
```

b)

```
DROP TRIGGER IF EXISTS chk_solicita_trigger AS solicita;

CREATE OR REPLACE FUNCTION chk_solicita() returns Trigger AS $$
BEGIN
  DECLARE n int;
  SET n = (
    SELECT COUNT(idCoordenador)
    FROM vigia NATURAL JOIN eventoEmergencia NATURAL JOIN audita AS table
    WHERE new.idCoordenador = table.idCoordenador AND new.numCamara = table.numCamara
    );
  IF n < 1 THEN
    RAISE EXCEPTION 'Nonexistent Coordinator ID',
    USING HINT = 'Check if Coordinator auditates the local adress'
  END if;
end;
$$ Language plpgsql;

CREATE TRIGGER chk_solicita_trigger after update on solicita
FOR EACH ROW EXECUTE PROCEDURE chk_solicita();
```

INDICES

Os índices são uma maneira comum de melhorar o desempenho de uma base de dados. Um índice permite que o servidor de uma base de dados localize linhas específicas muito mais rápido do que seria possível sem um índice. Mas os índices também adicionam sobrecarga ao sistema de banco de dados como um todo, portanto devem ser usados de maneira sensata e eficiente.

1)

```
select dataHoraInício, dataHoraFim
from video V, vigia I
where V.numCamara = I.numCamara
      and V.numCamara = 10
      and I.moradaLocal = “Loures”
```

a) Para acelerar esta interrogações deve podem ser utilizados 2 tipos de índice: um HASH ou um BITMAP pois existe uma comparação de atributos e também uma condição AND que pode ser utilizada por ambos. Para esta alínea vamos utilizar o HASH. Este índice optimiza queries que tenham testes de igualdade. Os atributos numCamera e moradaLocal fazem parte da chave primária de Camera e Vigia, assim pode-se acelerar o processo de pesquisa, considerando um número grande de entradas com a mesma moradaLocal de um dado evento!

b)

```
                Table "public.video"
       Column         |  Type   | Modifiers
--------------------+---------+-----------
 datahorainiciovideo | date    | not null
 datahorafim         | date    | not null
 numcamera           | integer | not null
Indexes:
    "video_pkey" PRIMARY KEY, btree (datahorainiciovideo)
```

```
                Table "public.vigia"
   Column    |          Type          | Modifiers
-------------+------------------------+-----------
 moradalocal | character varying(255) | not null
 numcamera   | integer                | not null
Indexes:
    "vigia_pkey" PRIMARY KEY, btree (moradalocal, numcamera)
```

Quando é criada a base de dados no postgreSQL, automaticamente é fornecido um índice BTree a tabela. Neste caso iremos criar outro índice HASH (Já que o BITMAP não é suportado pelo postgreSQL) como foi o escolhido na alínea anterior para poder comparar com o antigo índice e ver se houve melhoria.
Para a criação do índice HASH em video:

```
CREATE INDEX num_camera_indx ON video USING HASH(numCamera);
```

INDICE BTree no Video -> Time: 1.581ms

INDICE HASH no Video -> Time: 0.756ms

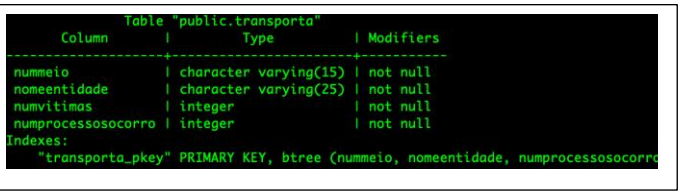
Apesar de a nossa base de dados ser pequena, podemos verificar uma certa melhoria com os testes antes e depois de criado o índice HASH.
Para a tabela Vigia, o método de criação é igual ao do Video e também pode se verificar uma descida/melhoria de tempo de execução.

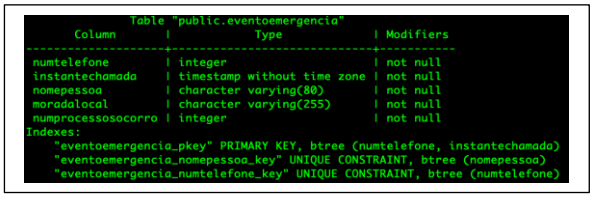
2)

```
select sum(numVitimas)
from transporta T, EventoEmergencia E
where T.numProcessoSocorro = E.numProcessoSocorro
group by numTelefone, instanteChamada
```

a) Para acelerar esta interrogação deve ser utilizado o índice HASH de chave de pesquisa <numTelefone, instanteChamada> nas tabelas Transporta e EventoEmergencia. Como o WHERE é feito com uma relação de igualdade, onde T.numProcessoSocorro = E.numProcessoSocorro torna a procura mais eficiente do que o índice BTree já criado pela tabela inicialmente.
Como os atributos que são utilizados são chaves primárias da sua respetiva tabela, então não é necessário percorrer toda a tabela para obter os resultados necessários.

b)





Tal como em 1.b), quando são criadas as tabelas, por definição os postgreSQL, fornece automaticamente um índice BTree. Para uma melhor eficácia iremos tratar com um índice HASH.
Para a criação do índice HASH em transporta:

```
CREATE INDEX numProcSoc_indx ON transporta USING HASH(numProcessoSocorro);
```

INDICE BTree no transporta -> Time: 3.609 ms
INDICE HASH no transporta -> Time: 1.7544ms

Apesar de a nossa base de dados ser pequena, podemos verificar uma certa melhoria com os testes antes e depois de criado o índice HASH.

DATA WAREHOUSE

Criação das Tabelas

```sql
DROP TABLE IF EXISTS fact_table;
DROP TABLE IF EXISTS d_tempo;
DROP TABLE IF EXISTS d_meio;
DROP TABLE IF EXISTS d_evento;

-- CRIACAO DAS TABELAS

CREATE TABLE d_evento(
  idEvento SERIAL,
  numTelefone INT NOT NULL,
  instanteChamada date,
  PRIMARY KEY(idEvento));

CREATE TABLE d_meio(
  idMeio SERIAL,
  numMeio VARCHAR(15) NOT NULL,
  nomeMeio VARCHAR(15) NOT NULL,
  nomeEntidade VARCHAR(25) NOT NULL,
  tipo VARCHAR(15)  ,
  PRIMARY KEY(idMeio));

CREATE TABLE d_tempo(
  idData SERIAL ,
  dia int NOT NULL,
  mes int NOT NULL,
  ano int NOT NULL,
  PRIMARY KEY(idData));

CREATE TABLE fact_table(
  idFact SERIAL,
  idEvento INT ,
  idMeio INT ,
  idData INT ,
  PRIMARY KEY(idFact),
  FOREIGN KEY (idEvento) REFERENCES d_evento(idEvento),
  FOREIGN KEY (idMeio) REFERENCES d_meio(idMeio),
  FOREIGN KEY (idData) REFERENCES d_tempo(idData) ON DELETE CASCADE ON UPDATE CASCADE);
```

Populate

```sql
INSERT INTO d_evento (numTelefone, instanteChamada)
  SELECT numTelefone, instanteChamada
    FROM eventoEmergencia;


INSERT INTO d_meio (numMeio, nomeMeio, nomeEntidade)
  SELECT numMeio, nomeMeio, nomeEntidade
    FROM meio;

INSERT INTO d_tempo (dia, mes, ano)
  SELECT EXTRACT (DAY FROM instanteChamada) as dia,
       EXTRACT (MONTH FROM instanteChamada) as  mes,
       EXTRACT (YEAR FROM instanteChamada) as ano
       FROM eventoEmergencia;

INSERT INTO fact_table(idEvento, idMeio, idData)
    SELECT idEvento ,idMeio, idData FROM d_evento , d_meio ,d_tempo ;
```