TÉCNICO LISBOA

Projecto de Bases de Dados

Parte 4

LETI 2016-2017 - 1º Semestre

Grupo nº 50 - Turno BD8179L07 (5ª feira às 8h30)

Pedro Silva - 77929 (18 horas – tempo estimado)

Duarte Silva - 79762 (18 horas – tempo estimado)

Ana Rita Rocha - 79779 (18 horas – tempo estimado)

Índices

```
select A.nif
from Arrenda A
      inner join Fiscaliza F
      on A.morada = F.morada
      and A.codigo = F.codigo
group by A.nif
having count(distinct F.id) = 1
```

Figura 1 - 1ª interrogação

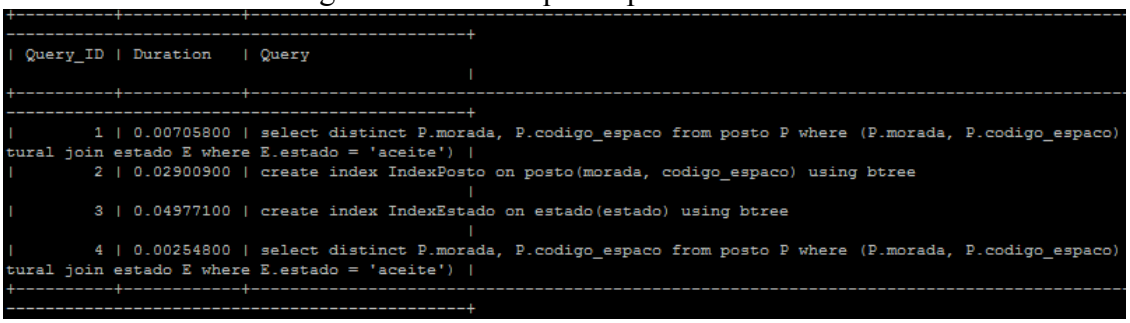
a) Uma vez que o engine das tabelas utilizadas é InnoDB, só podemos utilizar índices do tipo BTREE. No entanto, nesta 1ª interrogação, o que otimizaria realmente o tempo de execução seria a utilização de índices do tipo HASH devido às comparações como "A.morada = F.morada" ou "A.codigo = F.codigo", isto porque queremos comparar valores únicos e não uma gama de valores.
Como se estão a usar chaves primárias, e estas não se repetem, não existe maneira de criar otimização na pesquisa.
Quando é feito um INNER JOIN cria-se uma tabela temporária e estas não podem ser otimizadas.

```
select distinct P.morada, P.codigo_espaco
from Posto P
where (P.morada, P.codigo_espaco) not in (
    select P.morada, P.codigo_espaco
    from Posto P
      natural join Aluga A
      natural join Estado E
    where E.estado = 'aceite')
```

Figura 2 - 2ª interrogação

a) É notório, olhando imediatamente para a 2ª interrogação, que um índice do tipo BTREE aplicado à tabela *estado* e ao argumento *estado* ajudaria a otimizar o tempo de execução pois, com este índice, todos os dados da tabela *estado* ficam organizados alfabeticamente e assim que um argumento for diferente de 'aceite' este poderá interromper a sua execução pois sabe que não haverão mais estados iguais.

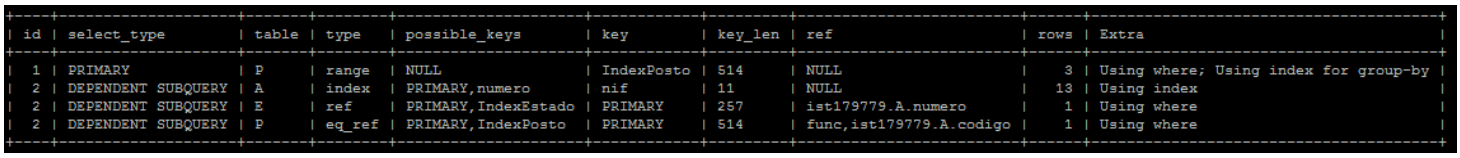
Criando um índice também na tabela *posto* com os argumentos *morada* e *código_espaco* otimiza a interrogação devido ao facto de a interrogação "recorrer" 3 vezes a estes argumentos sendo que da primeira vez usa um "select distinct".



Query_ID	Duration	Query
1	0.00705800	select distinct P.morada, P.codigo_espaco from posto P where (P.morada, P.codigo_espaco) tural join estado E where E.estado = 'aceite')
2	0.02900900	create index IndexPosto on posto(morada, codigo_espaco) using btree
3	0.04977100	create index IndexEstado on estado(estado) using btree
4	0.00254800	select distinct P.morada, P.codigo_espaco from posto P where (P.morada, P.codigo_espaco) tural join estado E where E.estado = 'aceite')

Figura 3 - tempo de execução antes e depois da criação dos índices

Como ilustra a figura 3, a 2ª interrogação demorou 0.007058s a executar-se e após a criação dos 2 índices, demorou apenas 0.002548s.



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	P	range	NULL	IndexPosto	514	NULL	3	Using where; Using index for group-by
2	DEPENDENT SUBQUERY	A	index	PRIMARY,numero	nif	11	NULL	13	Using index
2	DEPENDENT SUBQUERY	E	ref	PRIMARY,IndexEstado	PRIMARY	257	ist179779.A.numero	1	Using where
2	DEPENDENT SUBQUERY	P	eq_ref	PRIMARY,IndexPosto	PRIMARY	514	func,ist179779.A.codigo	1	Using where

Figura 4 - explain da query

b) CREATE INDEX IndexPosto on posto(morada, código_espaco) USING BTREE;
CREATE INDEX IndexEstado on estado(estado) USING BTREE;

Data Warehouse

1.

- Criação do esquema em estrela (do ficheiro reserva_star_shcema.sql):

data_dimension (Data):

```
DROP TABLE IF EXISTS `data_dimension`;
CREATE TABLE `data_dimension`(
      `data_key` int(11) NOT NULL,
      `dia` int(11) NOT NULL,
      `semana` int(11) NOT NULL,
      `mes` int(11) NOT NULL,
      `semestre` int(11) NOT NULL,
      `ano` int(11) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

user_dimension (Utilizador que reservou):

```
DROP TABLE IF EXISTS `user_dimension`;
CREATE TABLE `user_dimension` (
      `nif_key` varchar(9) NOT NULL,
     `nome` varchar(80) NOT NULL,
     `telefone` varchar(26) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

localização_dimension (Localização):

```
DROP TABLE IF EXISTS `localizacao_dimension`;
CREATE TABLE `localizacao_dimension` (
      `localizacao_key` varchar(255) NOT NULL,
      `morada` varchar(255) NOT NULL,
      `codigo` varchar(255) NOT NULL,
      `codigo_espaco` varchar(255)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

tempo_dimension (Tempo):

```
DROP TABLE IF EXISTS `tempo_dimension`;
CREATE TABLE `tempo_dimension` (
      `tempo_key` int(11) NOT NULL,
      `hora_do_dia` int(11) NOT NULL,
      `minuto_do_dia` int(11) NOT NULL
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

reservas_informação (Reservas):

```
DROP TABLE IF EXISTS `reservas_informacao`;
CREATE TABLE `reservas_informacao` (
      `montante` numeric(19,4) NOT NULL,
      `duracao` int(11) NOT NULL,
      `data_key` int(11) NOT NULL,
      `nif_key` varchar(9) NOT NULL,
      `localizacao_key` varchar(255) NOT NULL,
      `tempo_key` int(11) NOT NULL,
     KEY `idx_reservas` (`data_key`,`nif_key`,`localizacao_key`,
`tempo_key`) USING BTREE
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- Instruções SQL necessárias para carregar o esquema em estrela a partir das tabelas existentes (do ficheiro pop.sql):

Populate para a tabela data_dimension:

```
DROP PROCEDURE IF EXISTS load_date_dim;
delimiter //
CREATE PROCEDURE load_date_dim()
```

```
BEGIN
   DECLARE v_full_date DATETIME;
   SET v_full_date = '2016-01-01 00:00:00';
   WHILE v_full_date < '2018-01-01 00:00:00' DO
   INSERT INTO data_dimension(data_key,dia,semana,mes,semestre,
ano) VALUES (
           YEAR(v_full_date) * 10000 + MONTH(v_full_date)*100 +
DAY(v_full_date),
           DAY(v_full_date),
           WEEK(v_full_date),
           MONTH(v_full_date),
           QUARTER(v_full_date),
           YEAR(v_full_date)
       );
       SET v_full_date = DATE_ADD(v_full_date, INTERVAL 1 DAY);
   END WHILE;
END;
//
```

Populate para a tabela user_dimension:

```
INSERT INTO user_dimension
SELECT nif, nome, telefone
FROM user;
```

Populate para a tabela localizacao_dimension:

```
INSERT INTO localizacao_dimension
SELECT CONCAT(O.morada,O.codigo, COALESCE(P.codigo_espaco, ''))
AS localizacao_key, O.morada, O.codigo, P.codigo_espaco
FROM oferta O LEFT JOIN posto P ON O.codigo = P.codigo;
```

Populate para a tabela tempo_dimension:

```
DROP PROCEDURE IF EXISTS load_tempo_dim;
delimiter //
CREATE PROCEDURE load_tempo_dim()
BEGIN
   DECLARE v_full_date DATETIME;
   SET v_full_date = '2016-01-01 00:00:00';
   WHILE v_full_date < '2016-01-02 00:00:00' DO
       INSERT INTO tempo_dimension(
          tempo_key,
           hora_do_dia,
           minuto_do_dia
       ) VALUES (
           HOUR(v_full_date)*100 + MINUTE(v_full_date),
           HOUR(v_full_date),
           MINUTE(v_full_date)
       );
```

```
       SET v_full_date = DATE_ADD(v_full_date, INTERVAL 1
MINUTE);
   END WHILE;
END;
//
```

Populate para a tabela reserva_informacao:

```
DROP PROCEDURE IF EXISTS load_reservas_info;
delimiter //
CREATE PROCEDURE load_reservas_info()
BEGIN
     DECLARE done INT DEFAULT FALSE;
     DECLARE m_nif_key VARCHAR(9);
     DECLARE m_dataa TIMESTAMP;
     DECLARE data_key INT;
     DECLARE tempo_key INT;
     DECLARE localizacao_key VARCHAR(255);
     DECLARE m_morada VARCHAR(255);
     DECLARE m_codigo VARCHAR(255);
     DECLARE m_codigo_espaco VARCHAR(255);
     DECLARE m_numero VARCHAR(255);
     DECLARE m_montante NUMERIC(19,4);
     DECLARE m_duracao INT;

DECLARE cur1 CURSOR FOR
SELECT nif, data, O.morada, O.codigo, P.codigo_espaco,
tarifa*DATEDIFF(data_fim, data_inicio) as montante,
DATEDIFF(data_fim, data_inicio) as duration, numero
FROM paga NATURAL JOIN estado NATURAL JOIN aluga NATURAL JOIN
oferta O LEFT JOIN posto P ON O.codigo = P.codigo
WHERE estado = 'Paga'
GROUP BY numero;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur1;
    read_loop: LOOP
      FETCH cur1 INTO m_nif_key, m_dataa, m_morada, m_codigo,
m_codigo_espaco, m_montante, m_duracao, m_numero;
        IF done THEN
          LEAVE read_loop;
        END IF;

SET data_key = 10000*YEAR(m_dataa) + 100*MONTH(m_dataa) +
DAY(m_dataa);

SET tempo_key = 100*HOUR(m_dataa) + MINUTE(m_dataa);

SET localizacao_key =
CONCAT(m_morada,m_codigo,COALESCE(m_codigo_espaco, ''));

INSERT INTO reservas_informacao VALUES(m_montante, m_duracao,
data_key, m_nif_key, localizacao_key, tempo_key);
```

```
     END LOOP;

    CLOSE cur1;
END;
//
```

2.

```
select codigo_espaco, codigo, dia, mes,avg(montante)
from localizacao_dimension natural join data_dimension natural
join reservas_informacao
group by,codigo_espaco, codigo, dia, mes with ROLLUP
UNION
select codigo_espaco, codigo, dia, mes,avg(montante)
from localizacao_dimension natural join data_dimension natural
join reservas_informacao
group by codigo_espaco,codigo,mes with ROLLUP
UNION
select codigo_espaco, codigo, dia, mes,avg(montante)
from localizacao_dimension natural join data_dimension natural
join reservas_informacao
group by codigo,dia,mes with ROLLUP
UNION
select codigo_espaco, codigo, dia, mes,avg(montante)
from localizacao_dimension natural join data_dimension natural
join reservas_informacao
group by codigo_espaco,mes with ROLLUP;
```