Projeto de Bases de Dados

Parte 4

1ºSemestre - 2016-2017

Grupo 5
Turno 4ª Feira às 8h - BD225179L06

Pedro Orvalho nº 81151
Ana Leitão nº 81365
Manuel Galamba nº 81647

Horas de trabalho: 16 horas de trabalho por elemento

Índices

a)

1. **Índice composto em <A.morada, A.código> e <F.morada, F.código>** - índice hash-based pois este é o mais indicado para uma selecção por igualdade. Este é primário para a tabela arrenda pois *morada, código* é a chave primária e secundário para a tabela fiscaliza. Seria um índice denso pois há pelo menos uma entrada de dados no índice por cada valor da chave de pesquisa. E como não é necessário estar agrupado, pois é indexado por hash, é desagrupado.

   Índice agrupado em A.nif - considerando que o número de pesquisas é muito mais elevado que o número de alterações à tabela, um índice agrupado aumenta a rapidez das pesquisas quando se quer aceder a todos os dados na tabela, não sendo necessário aceder primeiro aos índices, acede-se diretamente à tabela.

2. **Índice em E.estado** - índice bitmap, pois este é o mais indicado quando o número de valores possíveis é muito reduzido, como é o caso dos estados no domínio do problema, e quando não são feitas alterações à tabela em número suficiente para tornar os bitmaps inviáveis.

b)

1. Verificámos com o comando “explain” que a query 1 durante a sua execução iria usar as chaves primárias da tabela arrenda, e o índice “morada” da foreign key da tabela fiscaliza.

```
mysql> explain 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;
+----+-------------+-------+--------+---------------+---------+---------+-------------------------------------+------+---------------------------------------------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref                                 | rows | Extra                                       |
+----+-------------+-------+--------+---------------+---------+---------+-------------------------------------+------+---------------------------------------------+
|  1 | SIMPLE      | F     | index  | morada        | morada  | 514     | NULL                                |    9 | Using index; Using temporary; Using filesort |
|  1 | SIMPLE      | A     | eq_ref | PRIMARY       | PRIMARY | 514     | ist181151.F.morada,ist181151.F.codigo |    1 |                                             |
+----+-------------+-------+--------+---------------+---------+---------+-------------------------------------+------+---------------------------------------------+
2 rows in set (0.01 sec)

mysql> show index from arrenda;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| arrenda |          0 | PRIMARY  |            1 | morada      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| arrenda |          0 | PRIMARY  |            2 | codigo      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| arrenda |          1 | nif      |            1 | nif         | A         |           0 |     NULL | NULL   |      | BTREE      |         |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
3 rows in set (0.00 sec)

mysql> show index from fiscaliza;
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| Table     | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
| fiscaliza |          0 | PRIMARY  |            1 | id          | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| fiscaliza |          0 | PRIMARY  |            2 | morada      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| fiscaliza |          0 | PRIMARY  |            3 | codigo      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| fiscaliza |          1 | morada   |            1 | morada      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
| fiscaliza |          1 | morada   |            2 | codigo      | A         |           0 |     NULL | NULL   |      | BTREE      |         |
+-----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+
5 rows in set (0.00 sec)
```

Uma vez que não é possível usar outros tipos de índice em mysql, não conseguimos optimizar mais o desempenho da query.

2. Repetindo o procedimento, usámos o comando “explain” para a query 2

```
mysql> explain 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
ere E.estado = "aceite");
+----+--------------------+-------+-------+----------------+---------+---------+-------------------------------------+-------+--------------------------+
| id | select_type        | table | type  | possible_keys  | key     | key_len | ref                                 | rows  | Extra                    |
+----+--------------------+-------+-------+----------------+---------+---------+-------------------------------------+-------+--------------------------+
|  1 | PRIMARY            | P     | index | NULL           | morada  | 514     | NULL                                | 24851 | Using where; Using index |
|  2 | DEPENDENT SUBQUERY | P     | ref   | PRIMARY,morada | morada  | 514     | func,func                           |     1 | Using where; Using index |
|  2 | DEPENDENT SUBQUERY | A     | ref   | PRIMARY,numero | PRIMARY | 514     | ist181151.P.morada,ist181151.P.codigo |     1 | Using index              |
|  2 | DEPENDENT SUBQUERY | E     | ref   | PRIMARY        | PRIMARY | 257     | ist181151.A.numero                  |     1 | Using where              |
+----+--------------------+-------+-------+----------------+---------+---------+-------------------------------------+-------+--------------------------+
4 rows in set (0.01 sec)
```

Ao estudarmos a query verificámos que se existisse um índice para o estado na tabela estado, isto iria causar um melhoramento do desempenho da query, uma vez que em vez do “where” iria ser usado o índice.
Criámos um índice para o estado na tabela estado, com o comando “create index estado_idx on estado(estado);” e comparámos os tempos.

```
mysql> show profiles;
+----------+-------------+---------------
-------------------------------------+
| Query_ID | Duration    | Query
                                     |
+----------+-------------+---------------
-------------------------------------+
|        1 | 47.41953000 | select distinc
 estado E where E.estado = "aceite") |
+----------+-------------+---------------
-------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> show profiles;
+----------+-------------+---------------
-------------------------------------+
| Query_ID | Duration    | Query
                                     |
+----------+-------------+---------------
-------------------------------------+
|        1 | 25.75507700 | select distinc
 estado E where E.estado = "aceite") |
+----------+-------------+---------------
-------------------------------------+
1 row in set (0.00 sec)
```

Os tempos com índice foram melhores (screenshot da direita) e correndo o comando “explain” verificámos passou a ser usado o indice estado_idx em vez do “where” para esta query.

```
mysql> explain 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 jo
in estado E where E.estado = 'aceite');
+----+--------------------+-------+------------+-------+--------------------+------------+---------+-----------------------------------+------+----------+------------------------------------
---+
| id | select_type        | table | partitions | type  | possible_keys      | key        | key_len | ref                               | rows | filtered | Extra
  |
+----+--------------------+-------+------------+-------+--------------------+------------+---------+-----------------------------------+------+----------+------------------------------------
---+
|  1 | PRIMARY            | P     | NULL       | range | morada             | morada     | 514     | NULL                              |    4 |   100.00 | Using where; Using index for group-
by |
|  2 | DEPENDENT SUBQUERY | P     | NULL       | ref   | PRIMARY,morada     | PRIMARY    | 257     | func                              |    3 |    11.11 | Using where
  |
|  2 | DEPENDENT SUBQUERY | A     | NULL       | ref   | PRIMARY,numero     | PRIMARY    | 514     | project.P.morada,project.P.codigo |    1 |   100.00 | Using index
  |
|  2 | DEPENDENT SUBQUERY | E     | NULL       | ref   | PRIMARY,estado_idx | estado_idx | 514     | const,project.A.numero            |    1 |   100.00 | Using index
  |
+----+--------------------+-------+------------+-------+--------------------+------------+---------+-----------------------------------+------+----------+------------------------------------
---+
4 rows in set, 1 warning (0.00 sec)
```

create index estado_idx on estado(estado);

Data Warehouse

Esquema em estrela

A tabela **reserva_dimension** contém a informação de cada reserva (data e hora de pagamento, localização, duração e montante pago). A tabela **user_dimension** contém os detalhes de cada utilizador (nif, nome e telefone). A tabela **localizacao_dimension** contém a morada e códigos de cada posto e espaço. A tabela **tempo_dimension** caracteriza cada minuto de um dia e a tabela **data_dimension** caracteriza cada dia dos anos 2016 e 2017.

```
create table user_dimension (
  nif varchar(9) not null unique,
  nome varchar(80) not null,
  telefone varchar(26) not null,
  primary key(nif)
);
```

```
create table localizacao_dimension (
  localizacao_id varchar(765) not null unique,
  morada varchar(255) not null,
  codigo_espaco varchar(255) not null,
  codigo_posto varchar(255),
  primary key(localizacao_id)
);
```

```
create table reserva_dimension (
  reserva_id varchar(255) not null unique,
  tempo_id int not null,
  data_id int not null,
  nif varchar(9) not null,
  localizacao_id varchar(765) not null,
  duracao int not null,
  montante_pago numeric(20,4) not null,
  primary key(tempo_id, data_id, nif, localizacao_id),
  foreign key(tempo_id) references tempo_dimension(tempo_id),
  foreign key(data_id) references data_dimension(data_id),
  foreign key(nif) references user_dimension(nif),
  foreign key(localizacao_id) references localizacao_dimension(localizacao_id)
);
```

```
create table data_dimension (
  data_id int not null unique,
  ano int not null,
  semestre int not null,
  mes_do_ano int not null,
  semana_do_ano int not null,
  dia_do_ano int not null,
  dia_do_mes int not null,
  dia_da_semana varchar(10) not null,
  primary key(data_id)
);
```

```
create table tempo_dimension (
  tempo_id int not null unique,
  hora_do_dia int not null,
  minuto_do_dia int not null,
  minuto_da_hora int not null,
  primary key(tempo_id)
);
```

Tabela user_dimension:
Esta tabela é populada com informação proveniente da tabela *user* através do seguinte código sql

```
INSERT INTO user_dimension SELECT * FROM user;
```

Tabela localizacao_dimension:
Esta tabela é populada com a morada e código do espaço de todos as entradas das tabelas *posto* e *espaco*. São também inseridos os códigos dos postos e no caso dos espaços é NULL nessa coluna.

```
INSERT INTO localizacao_dimension
  SELECT concat(morada, '|', codigo) AS localizacao_id , morada,
  codigo AS codigo_espaco, null AS codigo_posto FROM espaco
UNION
  SELECT concat(morada, '|', codigo_espaco, '|', codigo) AS localizacao_id, morada,
  codigo_espaco, codigo AS codigo_posto FROM posto;
```

Tabela Tempo_dimension:
Esta tabela é populada com todas as horas e minutos de um dia através do seguinte código:

```
CREATE PROCEDURE load_time_dimension()
  BEGIN
    SET @tempo_inicio = '00:00';
    SET @tempo_fim = '23:59:59';
    SET @tempo = @tempo_inicio;
    SET @minuto_dia = 0;
    WHILE @tempo <= @tempo_fim DO
      INSERT INTO tempo_dimension VALUES(
        time_format(@tempo, "%H%i"),
        hour(@tempo),
        @minuto_dia,
        minute(@tempo)
      );
      SET @tempo = addtime(@tempo, '00:01');
      SET @minuto_dia = hour(@tempo)*60 + minute(@tempo);
    END WHILE;
  END //

CALL load_time_dimension()//
```

Tabela data_dimension:

Esta tabela é populada com todos os dias dos anos 2016 e 2017 através do seguinte código:

```
CREATE PROCEDURE load_date_dimension()
  BEGIN
    SET @data_inicio = '2016-01-01';
    SET @data_fim = '2017-12-31';
    SET @data = @data_inicio;

    WHILE @data <= @data_fim DO
      IF quarter(@data) <= 2
        THEN SET @semestre = 1;
        ELSE SET @semestre = 2;
      END IF;
      INSERT INTO data_dimension VALUES(
        date_format(@data, "%Y%m%d"),
        year(@data),
        @semestre,
        month(@data),
        week(@data),
        dayofyear(@data),
        day(@data),
        dayname(@data)
      );
      SET @data = date_add(@data, INTERVAL 1 DAY);
    END WHILE;
  END //

CALL load_date_dimension()//
```

Tabela reserva_dimension:

Esta tabela é populada através da seguinte query:

```
INSERT INTO reserva_dimension
  SELECT numero AS reserva_id, time_format(data, "%H%i") AS tempo_id, date_format(data,"%Y%m%d")
  AS data_id, nif, concat(morada,codigo) AS localizacao_id, DATEDIFF(data_fim,data_inicio)
  AS duracao, DATEDIFF(data_fim,data_inicio)*tarifa AS montante_pago
  FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN espaco
  WHERE 2015 < YEAR(data) and YEAR(data) < 2018
UNION
  SELECT numero AS reserva_id, time_format(data, "%H%i") AS tempo_id, date_format(data,"%Y%m%d")
  AS data_id, nif, concat(morada,codigo_espaco, codigo) AS localizacao_id, DATEDIFF(data_fim,data_inicio)
  AS duracao, DATEDIFF(data_fim,data_inicio)*tarifa AS montante_pago
  FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN posto
  WHERE 2015 < YEAR(data) and YEAR(data) < 2018;
```

Consulta OLAP

Para ser criado um cubo em mysql temos que recorrer ao uso de vários agrupamentos rollup, neste caso seis, para os conjuntos:

- (codigo_espaco, codigo_posto, dia_do_mes, mes_do_ano)
- (codigo_posto, dia_do_mes, mes_do_ano, codigo_espaco)
- (dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto)
- (mes_do_ano, codigo_espaco, codigo_posto, dia_do_mes)
- (codigo_espaco, dia_do_mes, codigo_posto, mes_do_ano)
- (codigo_posto, mes_do_ano, codigo_espaco, dia_do_mes)

sendo assim feito o group by por todos os subconjuntos possíveis para estas quatro dimensões, como ocorreria num cubo.

Este código é uma repetição para cada um dos seis conjuntos de uma query que cria uma tabela com a média do montante feito por um alugavel, o dia do mês, o mês do ano e o código do espaço e do posto. Estes valores são provenientes da junção das tabelas reserva_dimension (montante), data_dimension (dia e mês) e localizacao_dimension (código do espaço e do posto). É feita uma união de todas estas queries.

```
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension NATURAL JOIN localizacao_dimension
  GROUP BY codigo_espaco, codigo_posto, dia_do_mes, mes_do_ano WITH rollup
UNION
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension natural join localizacao_dimension
  GROUP BY codigo_posto, dia_do_mes, mes_do_ano, codigo_espaco WITH rollup
UNION
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension NATURAL JOIN localizacao_dimension
  GROUP BY dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto WITH rollup
UNION
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension NATURAL JOIN localizacao_dimension
  GROUP BY mes_do_ano, codigo_espaco, codigo_posto, dia_do_mes WITH rollup
UNION
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension NATURAL JOIN localizacao_dimension
  GROUP BY codigo_espaco, dia_do_mes, codigo_posto, mes_do_ano WITH rollup
UNION
  SELECT AVG(montante_pago) AS valor_medio_pago, dia_do_mes, mes_do_ano, codigo_espaco, codigo_posto
  FROM  reserva_dimension NATURAL JOIN data_dimension NATURAL JOIN localizacao_dimension
  GROUP BY codigo_posto, mes_do_ano, codigo_espaco, dia_do_mes WITH rollup;
```