Projecto de Bases de Dados, Parte4

Professor Gabriel Pestana

Turno: Quinta-feira, 11:00-12:30

Grupo45

Inês Sequeira, nº81719

Pedro Gomes, nº81534

Rafael Belchior, nº80970

Esforço (em horas)

81719-------16horas

81534-------16horas

80970-------16horas

Indíces

Ex a)

Query 1:

Para melhorar o join podemos criar um índice composto sobre os atributos (morada, codigo) sobre a tabela arrenda, e um índice composto sobre os atributos (morada, codigo) sobre a tabela fiscaliza. Ambos os índices são do tipo btree. O SGBD escolhe percorrer uma das tabelas e usa o índice da outra para fazer o join, o que permite que o join seja feito mais eficientemente.

O índice mencionado para a tabela arrenda é um índice primário, pois contém a chave primária. O índice primário é agrupado, por default.

Query 2:

Para melhorar o natural join decidimos criar os seguintes índices:

- Índice primário, composto, sobre os atributos (morada, codigo), sobre a tabela posto e índice composto sobre os atributos (morada, codigo), sobre a tabela aluga;
- Índice sobre o atributo numero, sobre a tabela aluga, e índice sobre o atributo numero, sobre a tabela estado;

O índice (morada, codigo) sobre o posto é primário, pois contém a chave primária.

Os índices acima mencionados são todos do tipo btree. Nos casos mencionados acima, o SGBD escolhe percorrer uma das tabelas e usa o índice da outra para fazer o natural join, o que permite que este seja feito mais eficientemente.

Devido ao *where* com estado, podemos adicionar um índice sobre o atributo estado, sobre a tabela estado. Este índice permite filtrar os registos mais rapidamente. Se for escolhido o índice do tipo hash, pode ser mais eficiente do que o btree em alguns casos. Neste caso, o índice hash é útil se a hash table colocar diferentes valores do atributo ('Aceite', 'Pendente', etc) em diferentes entradas. Quando o estado 'Aceite' está isolado numa entrada da hash table, previsivelmente é mais rápido aceder aos aceites do que usando um índice btree.

Ex b)

Para testar o tempo das queries, populamos a nossa base de dados com cerca de 9 milhões de registos.

Usamos o explain do mysql para saber o plano de execução. Os printscreens mostrados abaixo são relativos ao explain.

Desativámos a cache com o intuito de esta não influenciar os tempos da execução das queries.

Query 1:

Não é necessário criar os índices sobre (morada, codigo) nas tabelas arrenda e fiscaliza, pois (morada, codigo) é primary key na tabela arrenda, e é foreign key na tabela fiscaliza. O SGBD cria estes índices automaticamente. Para testar se os índices permitem que a query seja mais rápida, removemos o nome da foreign key e desativámos a sua utilização.

Tempo sem os índices de FK: 9.23804400 s

```
1 row in set (9.24 sec)

+----+-------------+-------+-------+---------------+---------+---------+-----------------------------------+---------+----------------------------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref                               | rows    | Extra                                        |
+----+-------------+-------+-------+---------------+---------+---------+-----------------------------------+---------+----------------------------------------------+
|  1 | SIMPLE      | F     | index | NULL          | PRIMARY | 518     | NULL                              | 1000533 | Using index; Using temporary; Using filesort; Distinct |
|  1 | SIMPLE      | A     | eq_ref | PRIMARY      | PRIMARY | 514     | ist181534.F.morada,ist181534.F.codigo |       1 | Distinct                                     |
+----+-------------+-------+-------+---------------+---------+---------+-----------------------------------+---------+----------------------------------------------+
2 rows in set (0.00 sec)
```

Tempo com os índices de FK: 6.68024600 s

```
1 row in set (6.68 sec)

+----+-------------+-------+-------+---------------+-----------+---------+-----------------------------------+---------+----------------------------------------------+
| id | select_type | table | type  | possible_keys | key       | key_len | ref                               | rows    | Extra                                        |
+----+-------------+-------+-------+---------------+-----------+---------+-----------------------------------+---------+----------------------------------------------+
|  1 | SIMPLE      | F     | index | morcod_idx    | morcod_idx | 514     | NULL                              | 1000635 | Using index; Using temporary; Using filesort; Distinct |
|  1 | SIMPLE      | A     | eq_ref | PRIMARY      | PRIMARY   | 514     | ist181534.F.morada,ist181534.F.codigo |       1 | Distinct                                     |
+----+-------------+-------+-------+---------------+-----------+---------+-----------------------------------+---------+----------------------------------------------+
2 rows in set (0.00 sec)
```

Melhoria de performance: 9.23804400 / 6.68024600 = 1.38288979178
38% de melhoria

Query 2:

Não é necessário criar o índice sobre (morada, codigo) na tabela posto, pois (morada, codigo) é primary key na tabela posto. Não é necessário criar o índice sobre numero na tabela aluga, pois numero é foreign key na tabela aluga.

Para criar o índice sobre estado usamos índice tipo btree, pois a versão do mysql disponibilizada não permite o uso de índices tipo hash.

Tempo sem índice estado: 0.03946300 s

```
3 rows in set (0.04 sec)

+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
| id | select_type        | table | type  | possible_keys   | key     | key_len | ref                                     | rows  | Extra                                   |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
|  1 | PRIMARY            | P     | range | NULL            | morada  | 514     | NULL                                    |     3 | Using where; Using index for group-by |
|  2 | DEPENDENT SUBQUERY | P     | ref   | PRIMARY,morada  | morada  | 514     | func,func                               |     4 | Using where; Using index              |
|  2 | DEPENDENT SUBQUERY | A     | ref   | PRIMARY,numero  | PRIMARY | 514     | ist181534.P.morada,ist181534.P.codigo | 55596 | Using index                           |
|  2 | DEPENDENT SUBQUERY | E     | ref   | PRIMARY         | PRIMARY | 257     | ist181534.A.numero                      |     1 | Using where                           |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
```

Tempo com índice estado, mas não usa o índice: 0.03994700 s

```
3 rows in set (0.04 sec)

+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
| id | select_type        | table | type  | possible_keys   | key     | key_len | ref                                     | rows  | Extra                                   |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
|  1 | PRIMARY            | P     | range | NULL            | morada  | 514     | NULL                                    |     3 | Using where; Using index for group-by |
|  2 | DEPENDENT SUBQUERY | P     | ref   | PRIMARY,morada  | morada  | 514     | func,func                               |     4 | Using where; Using index              |
|  2 | DEPENDENT SUBQUERY | A     | ref   | PRIMARY,numero  | PRIMARY | 514     | ist181534.P.morada,ist181534.P.codigo | 55596 | Using index                           |
|  2 | DEPENDENT SUBQUERY | E     | ref   | PRIMARY,est_idx | PRIMARY | 257     | ist181534.A.numero                      |     1 | Using where                           |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
```

Neste caso, o SGBD não usa o índice estado pois, na tabela estado, existem cerca de 25% registos com estado='Aceite'. O SGBD apenas usa um índice se o número de registos que ele prevê verificar for cerca de 10%/15%, ou menos (a percentagem muda de acordo com o SGBD). Se o SGBD prever verificar muitos registos, é preferível percorrer toda a tabela, pois não compensa usar o índice, devido ao custo adicional de aceder à tabela de índices.

Mudámos os registos na base de dados de modo a existir menos de 10% de registos com estado='Aceite' na tabela estado. Desta forma, o índice por estado já foi usado. De seguida mostramos o tempo da query e o resultado do explain com poucos registos com estado='Aceite'.

Tempo sem índice estado e restantes não criados automaticamente: 0.01669800 s

```
3 rows in set (0.02 sec)

+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
| id | select_type        | table | type  | possible_keys   | key     | key_len | ref                                     | rows  | Extra                                   |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
|  1 | PRIMARY            | P     | range | NULL            | morada  | 514     | NULL                                    |     3 | Using where; Using index for group-by |
|  2 | DEPENDENT SUBQUERY | P     | ref   | PRIMARY,morada  | morada  | 514     | func,func                               |     4 | Using where; Using index              |
|  2 | DEPENDENT SUBQUERY | A     | ref   | PRIMARY,numero  | PRIMARY | 514     | ist181534.P.morada,ist181534.P.codigo | 55592 | Using index                           |
|  2 | DEPENDENT SUBQUERY | E     | ref   | PRIMARY         | PRIMARY | 257     | ist181534.A.numero                      |     1 | Using where                           |
+----+--------------------+-------+-------+-----------------+---------+---------+-----------------------------------------+-------+-----------------------------------------+
```

Tempo com índice estado e restantes não criados automaticamente: 0.01464200 s

```
+----+--------------------+-------+-------+-----------------------------+---------+---------+-------------------+------+-----------------------------+
| id | select_type        | table | type  | possible_keys               | key     | key_len | ref               | rows | Extra                       |
+----+--------------------+-------+-------+-----------------------------+---------+---------+-------------------+------+-----------------------------+
|  1 | PRIMARY            | P     | index | NULL                        | morada  | 514     | NULL              |    9 | Using where; Using index |
|  2 | DEPENDENT SUBQUERY | P     | ref   | PRIMARY,morada              | PRIMARY | 257     | func              |    1 | Using where                 |
|  2 | DEPENDENT SUBQUERY | E     | ref   | PRIMARY,est_idx             | est_idx | 257     | const             |   13 | Using where; Using index |
|  2 | DEPENDENT SUBQUERY | A     | ref   | PRIMARY,numero,morada_idx | numero  | 257     | ist181534.E.numero |    1 | Using where; Using index |
+----+--------------------+-------+-------+-----------------------------+---------+---------+-------------------+------+-----------------------------+
```

Melhoria de performance: 0.01669800 / 0.01464200 = 1.14041797569
14% de melhoria

Data Warehouse

1) Esquema de uma estrela com informações sobre reservas e com dimensões Utilizador, Localização, Data, Tempo

estrela_schema.sql

```
DROP TABLE IF EXISTS reserva_facts;
DROP TABLE IF EXISTS date_dimension;
DROP TABLE IF EXISTS time_dimension;
DROP TABLE IF EXISTS location_dimension;
DROP TABLE IF EXISTS user_dimension;

CREATE TABLE user_dimension(
        uid int AUTO_INCREMENT,
        nif varchar(9) NOT NULL UNIQUE,
        nome varchar(80) NOT NULL,
    telefone varchar(26) NOT NULL,
        PRIMARY KEY(uid));

CREATE TABLE location_dimension(
        lid int AUTO_INCREMENT,
        posto varchar(255),
        espaco varchar(255),
        edificio varchar(255) NOT NULL,
        PRIMARY KEY(lid));

CREATE TABLE time_dimension(
        tid int AUTO_INCREMENT,
        hora int NOT NULL,
        minuto int NOT NULL,
        PRIMARY KEY(tid));

CREATE TABLE date_dimension(
        did int AUTO_INCREMENT,
        dia int NOT NULL,
        semana int NOT NULL,
        mes int NOT NULL,
        semestre int NOT NULL,
        ano int NOT NULL,
        PRIMARY KEY(did));

CREATE TABLE reserva_facts(
        pago numeric(19,4) NOT NULL,
        duracao int NOT NULL,
        uid int NOT NULL,
        lid int NOT NULL,
        tid int NOT NULL,
        did int NOT NULL,
        PRIMARY KEY(uid, tid, did), -- assumindo que um utilizador não paga duas ou mais reservas na mesma hora e minuto do mesmo dia
        FOREIGN KEY(uid) REFERENCES user_dimension(uid),
        FOREIGN KEY(lid) REFERENCES location_dimension(lid),
        FOREIGN KEY(tid) REFERENCES time_dimension(tid),
        FOREIGN KEY(did) REFERENCES date_dimension(did));
```

estrela_populate.sql

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

INSERT INTO location_dimension(posto, espaco, edificio) SELECT codigo, codigo_espaco, morada FROM posto;

INSERT INTO location_dimension(espaco, edificio) SELECT codigo, morada FROM espaco;

INSERT INTO location_dimension(edificio) SELECT morada FROM edificio;

INSERT INTO reserva_facts(pago, duracao, uid, lid, tid, did)
        SELECT o.tarifa*DATEDIFF(o.data_fim, o.data_inicio), DATEDIFF(o.data_fim, o.data_inicio), ud.uid, ld.lid, td.tid, dd.did
        FROM paga p NATURAL JOIN aluga a NATURAL JOIN oferta o
                JOIN user_dimension ud ON ud.nif = a.nif
                JOIN location_dimension ld ON ld.espaco = a.codigo AND ld.edificio = a.morada
                JOIN time_dimension td ON td.hora = hour(p.data) AND td.minuto = minute(p.data)
                JOIN date_dimension dd ON dd.dia = day(p.data)
                                        AND dd.mes = month(p.data)
                                        AND dd.ano = year(p.data)
                WHERE ld.posto IS NULL;

INSERT INTO reserva_facts(pago, duracao, uid, lid, tid, did)
        SELECT o.tarifa*DATEDIFF(o.data_fim, o.data_inicio), DATEDIFF(o.data_fim, o.data_inicio), ud.uid, ld.lid, td.tid, dd.did
        FROM paga p NATURAL JOIN aluga a NATURAL JOIN oferta o NATURAL JOIN posto po
                JOIN user_dimension ud ON ud.nif = a.nif
                JOIN location_dimension ld ON ld.posto = a.codigo AND ld.espaco = po.codigo_espaco AND ld.edificio = a.morada
                JOIN time_dimension td ON td.hora = hour(p.data) AND td.minuto = minute(p.data)
                JOIN date_dimension dd ON dd.dia = day(p.data)
                                        AND dd.mes = month(p.data)
                                        AND dd.ano = year(p.data);
```

generate_date.sql

```
DELIMITER //

DROP PROCEDURE IF EXISTS generate_date //
CREATE PROCEDURE generate_date ()
BEGIN
  DECLARE dia int DEFAULT 1;
  DECLARE semana int DEFAULT 0;
  DECLARE mes int DEFAULT 1;
  DECLARE semestre int DEFAULT 1;
  DECLARE ano int DEFAULT 2016;
  DECLARE diaMax int DEFAULT 30;
  DECLARE diaAno int DEFAULT 1;
  WHILE ano <= 2017 DO
    WHILE mes <= 12 DO
      IF (mes IN (1, 3, 5, 7, 8, 10, 12)) THEN
        SET diaMax = 31;
```

```
        ELSEIF (mes IN (4, 6, 9, 11)) THEN
          SET diaMax = 30;
        ELSEIF (mes = 2 && ano = 2016) THEN
          SET diaMax = 29;
        ELSEIF (mes = 2 && ano = 2017) THEN
          SET diaMax = 28;
        END IF;
        WHILE dia <= diaMax DO
          IF (diaAno % 7 = 1) THEN
            SET semana = semana + 1;
          END IF;
          IF (mes <= 6) THEN
            SET semestre = 1;
          ELSE
            SET semestre = 2;
          END IF;
          INSERT INTO date_dimension(dia, semana, mes, semestre, ano) VALUES(dia, semana, mes, semestre,
ano);
          SET dia = dia + 1;
          SET diaAno = diaAno + 1;
        END WHILE;
        SET dia = 1;
        SET mes = mes + 1;
      END WHILE;
      SET semana = 0;
      SET mes = 1;
      SET ano = ano + 1;
      SET diaAno = 1;
   END WHILE;
END //

DELIMITER ;

TRUNCATE date_dimension; -- to remove all registers from table date_dimension
CALL generate_date;
```

generate_time.sql

```
DELIMITER //

DROP PROCEDURE IF EXISTS generate_time //
CREATE PROCEDURE generate_time ()
BEGIN
   DECLARE hour int DEFAULT 0;
   DECLARE minute int DEFAULT 0;
   WHILE hour <= 23 DO
```

```
    WHILE minute <= 59 DO
      INSERT INTO time_dimension(hora, minuto) VALUES(hour, minute);
      SET minute = minute + 1;
    END WHILE;
    SET minute = 0;
    SET hour = hour + 1;
  END WHILE;
END //

DELIMITER ;

TRUNCATE time_dimension; -- to remove all registers from table time_dimension
CALL generate_time;
```

2) Consulta OLAP

```
SELECT espaco, posto, dia, mes, AVG(pago)
FROM reserva_facts
      NATURAL JOIN location_dimension
      NATURAL JOIN date_dimension
GROUP BY espaco, posto, dia, mes WITH ROLLUP

UNION

SELECT espaco, posto, dia, mes, AVG(pago)
FROM reserva_facts
      NATURAL JOIN location_dimension
      NATURAL JOIN date_dimension
GROUP BY posto, dia, mes, espaco WITH ROLLUP

UNION

SELECT espaco, posto, dia, mes, AVG(pago)
FROM reserva_facts
      NATURAL JOIN location_dimension
      NATURAL JOIN date_dimension
GROUP BY dia, mes, espaco, posto WITH ROLLUP

UNION

SELECT espaco, posto, dia, mes, AVG(pago)
FROM reserva_facts
      NATURAL JOIN location_dimension
      NATURAL JOIN date_dimension
GROUP BY mes, espaco, posto, dia WITH ROLLUP;
```