Projeto Bases de Dados

2016/2017

**4ª feira 8h30 - Grupo 9**

**81900** – Nuno Anselmo

**81936** – Liliana Oliveira

**82047** – André Mendes

Parte 4

Esforço dedicado:

**81900 – 8h**

**81936 – 8h**

**82047 – 8h**

Índices

1. **Primeira query :**

O índice mais indicado seria um índice B-Tree sobre os campos morada e código nas tabelas fiscaliza e arrenda, para acelerar o *INNER JOIN*.

**Segunda query:**

Para a tabela estado, um índice nos campos *estado* e *numero*, de tipo B-Tree, pois um índice do tipo Hash não permite matches parciais da chave. Este índice permitiria realizar uma match parcial do número, e com isso encontrar o estado, sendo possível realizar o *NATURAL JOIN* (e depois o *WHERE* com o estado) apenas com o índice.

Para a tabela aluga, um índice sobre os campos *morada*, *codigo* e *numero*, novamente B-Tree e pela mesma razão: permitir que a query “qual o número a que está associado um tuplo (morada, código)” seja realizada apenas sobre o índice.

Para a tabela posto, um índice nos campos *morada*, *codigo*, e *codigo\_espaco*, novamente B-Tree, para acelerar o join e permitir que toda a query seja feita apenas sobre os índices.

Com estes três índices, toda a query pode ser realizada sobre os índices, sem aceder a blocos em disco.

1. Não será necessário criar index para a primeira query porque o MySQL cria índices por omissão sobre a chave primária de cada tabela, que foi a proposta de índice indicada acima. Executando a query com *EXPLAIN*, os índices estão a ser usados.

Para a segunda query, os seguintes índices devem ser criados:

CREATE INDEX estado\_numero ON estado(estado, numero);

CREATE INDEX morada\_codigo\_numero ON aluga(morada, codigo, numero);

CREATE UNIQUE INDEX morada\_codigo\_cespaco ON posto(morada, codigo, codigo\_espaco);

Executando a query com *EXPLAIN* os índices não estão a ser usados mas aparecem como “*possible\_keys*”, indicando que foi feita uma escolha por parte do SGBD de não os usar apesar de estes estarem presentes e serem relevantes para a query. Suspeitamos que isto seja devido a uma baixa cardinalidade no nosso ambiente de testes, em que operar sobre a tabela é tão ou mais rápido do que operar sobre os índices.

Data Warehouse

**1.**

DELIMITER //

# Generates the date dimension, with all the days of the years 2016 and 2017.

DROP PROCEDURE IF EXISTS load\_date\_dimension //

CREATE PROCEDURE load\_date\_dimension()

BEGIN

DECLARE full\_date DATETIME;

SET full\_date = '2016-01-01 00:00:00';

WHILE full\_date < '2018-01-01 00:00:00' DO

      INSERT INTO date\_dimension (date\_id, dia, semana, mes, semestre, ano) VALUES (

        YEAR(full\_date) \* 10000 + MONTH(full\_date) \* 100 + DAY(full\_date),

        DAY(full\_date),

        # Week starting with a Sunday and range from 1-52

        WEEK(full\_date, 2),

        MONTH(full\_date),

        # If the month is lesser then the 7th month, its first semester, otherwise second.

        IF(MONTH(full\_date) < 7, 1, 2),

        YEAR(full\_date)

      );

      SET full\_date = DATE\_ADD(full\_date, INTERVAL 1 DAY);

    END WHILE;

END;

//

# Generates the time dimension, with all minutes of the day, since 00:00 until 23:59

DROP PROCEDURE IF EXISTS load\_time\_dimension //

CREATE PROCEDURE load\_time\_dimension()

  BEGIN

    DECLARE full\_day DATETIME;

    SET full\_day = '2016-01-01 00:00:00';

    WHILE full\_day < '2016-01-01 23:59:59' DO

      INSERT INTO time\_dimension (time\_id, hora, minuto) VALUES (

        HOUR(full\_day) \* 100 + MINUTE(full\_day),

        HOUR(full\_day),

        MINUTE(full\_day)

      );

      SET full\_day = DATE\_ADD(full\_day, INTERVAL 1 MINUTE);

    END WHILE;

END //

# Loads all users into the user dimension.

DROP PROCEDURE IF EXISTS load\_user\_dimension //

CREATE PROCEDURE load\_user\_dimension()

  BEGIN

    INSERT INTO user

      SELECT

        nif,

        nome,

        telefone

      FROM proj.user;

  END //

# Loads all the locations into the local dimension.

# All workspaces will have the cod\_posto as null.

DROP PROCEDUREIF EXISTS load\_local\_dimension //

CREATE PROCEDURE load\_local\_dimension()

  BEGIN

    # Unions all workspaces and workstations.

    INSERT INTO local\_dimension

      SELECT

        CONCAT(morada, codigo\_espaco, IFNULL(codigo\_posto, '')) AS local\_id,

        codigo\_espaco                                           AS cod\_espaco,

        codigo\_posto                                            AS cod\_posto,

        morada                                                  AS cod\_edificio

      FROM ((SELECT

               morada,

               codigo AS codigo\_espaco,

               NULL      codigo\_posto

             FROM proj.espaco)

            UNION ALL (SELECT

                         morada,

                         codigo\_espaco,

                         codigo AS codigo\_posto

                       FROM proj.posto)) AS local;

  END //

DROP PROCEDURE IF EXISTS load\_reserva //

CREATE PROCEDURE load\_reserva()

  BEGIN

    INSERT INTO reserva

      SELECT

       nif                                                                              AS nif,

        YEAR(data\_pagamento) \* 10000 + MONTH(data\_pagamento) \* 100 + DAY(data\_pagamento) AS date\_id,

        HOUR(data\_pagamento) \* 100 + MINUTE(data\_pagamento)                              AS time\_id,

        CONCAT(morada, codigo\_espaco, IFNULL(codigo\_posto, ''))                          AS local\_id,

        (DATEDIFF(data\_fim, data\_inicio) + 1) \* tarifa                                   AS total\_pago,

       DATEDIFF(data\_fim, data\_inicio)                                                  AS duracao\_em\_dias

      FROM (SELECT

              nif,

              morada,

              codigo AS codigo\_espaco,

              NULL   AS codigo\_posto,

              data\_inicio,

              data\_fim,

              data   AS data\_pagamento,

              tarifa

            FROM proj.aluga

              NATURAL JOIN proj.oferta

              NATURAL JOIN proj.espaco

              JOIN proj.paga ON paga.numero = aluga.numero

            UNION ALL

            SELECT

              nif,

              morada,

              codigo\_espaco,

              codigo AS codigo\_posto,

              data\_inicio,

              data\_fim,

              data   AS data\_pagamento,

              tarifa

            FROM proj.aluga

              NATURAL JOIN proj.oferta

              NATURAL JOIN proj.posto

              JOIN proj.paga ON paga.numero = aluga.numero) AS ReservasAlugadasEPagas;

  END //

# Loads the data warehouse.

DROP PROCEDURE IF EXISTS load\_data\_warehouse //

CREATE PROCEDURE load\_data\_warehouse()

  BEGIN

    CALL load\_time\_dimension();

    CALL load\_date\_dimension();

    CALL load\_user\_dimension();

    CALL load\_local\_dimension();

    CALL load\_reserva();

  END //

DELIMITER ;

**2.**

SELECT

  local\_id,

  date\_id,

*avg*(total\_pago)

FROM reserva

GROUP BY local\_id, date\_id WITH ROLLUP

UNION ALL

SELECT

  NULL AS local\_id,

  date\_id,

*avg*(total\_pago)

FROM reserva

GROUP BY date\_id;