

# Projeto Bases de Dados

## 2016/2017

### Parte 4

#### **4ª feira 8h30 - Grupo 9**

**81900** – Nuno Anselmo

**81936** – Liliana Oliveira

**82047** – André Mendes

Esforço dedicado:

**81900 – 8h**

**81936 – 8h**

**82047 – 8h**

# Índices

## a) 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.

- b) 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_espaco 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 AS 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;
```