



TÉCNICO
LISBOA

Bases de Dados

Licenciatura em Engenharia Telecomunicações e Informática

Projeto de Bases de Dados Parte 4

Índices
Data Warehouse

4^{af}-8:30H

Grupo 63

Bruno Santos – 82053 (14 horas)
Daniel Reigada – 82064 (14 horas)
Afonso Caetano – 82539 (14 horas)

Índices

1. a)

- Índice esparsos hash para (*morada, codigo*) em *Arrenda* e *Fiscaliza* de forma a otimizar o inner join das duas tabelas, pois este utiliza estas duas colunas;
- Índice esparsos hash para *id* em *Fiscaliza* de forma a otimizar a contagem de id's distintos.

2. a)

- Índice esparsos hash para (*morada, codigo_espaco*) em *Posto* de forma a otimizar a Função "NOT IN"
- Índice esparsos hash para *aceite* em *Estado* de forma a otimizar a procura de estados com valor 'aceite'

O índices organizados em *Hash tables* são os mais eficientes para comparações de igualdade (<valor1> = <valor2>) por essa razão e, porque todas as comparações feitas nas queries referidas são deste tipo, foram estes os tipos de índices escolhidos.

1. b)

- Não é necessário criar índice para (*morada, codigo*) em *Arrenda* pois estas colunas são as chaves primárias desta tabela e têm por defeito um índice;
- Também não existe a necessidade de criar índice para (*morada, codigo*) em *Fiscaliza* pois, tal como nas chaves primárias, são criados índices por defeito para Chaves estrangeiras;
- Após a criação deste índice é possível verificar (através do plano de execução da query) que este passa a ser utilizado, bem como pelo menor tempo de execução.

CREATE INDEX id_index **ON** Fiscaliza (id) **USING** HASH;

2. b)

- Tal como no primeiro caso não é necessário criar índices para (*morada, codigo_espaco*) pois este par de colunas já tem por defeito um índice associado por ser uma chave estrangeira;
- Após a criação deste índice é possível verificar (através do plano de execução da query) que este passa a ser utilizado, bem como pelo menor tempo de execução.

CREATE INDEX estado_index **ON** Estado (estado) **USING** HASH;

Nota: o MySQL com o *engine* InnoDB (engine por defeito) não suporta índices do tipo HASH

Data Warehouse

- dwSchema.sql

```
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS Localizacao
CASCADE;
DROP TABLE IF EXISTS Tempo CASCADE;
DROP TABLE IF EXISTS Data CASCADE;
DROP TABLE IF EXISTS Reserva_Factos
CASCADE;
SET FOREIGN_KEY_CHECKS = 1;
```

```
CREATE TABLE Localizacao
(
    location_id INTEGER NOT NULL
    AUTO_INCREMENT,
    posto INTEGER,
    espaco INTEGER,
    edificio VARCHAR(255),
    PRIMARY KEY (location_id)
);
```

```
CREATE TABLE Tempo
(
    time_id INTEGER NOT NULL
    AUTO_INCREMENT,
    time_of_day INTEGER NOT NULL,
    hour_of_day INTEGER NOT NULL,
    minute_of_day INTEGER NOT NULL,
    PRIMARY KEY (time_id)
);
```

```
CREATE TABLE Data
(
    date_id INTEGER NOT NULL,
    date_day INTEGER NOT NULL,
    date_week INTEGER NOT NULL,
    date_month_number INTEGER NOT NULL,
    date_semester INTEGER NOT NULL,
    date_year INTEGER NOT NULL,
    data DATE NOT NULL,
    PRIMARY KEY (date_id)
);
```

```
CREATE TABLE Reserva_Factos
(
    reserva_id INTEGER NOT NULL,
    date_id INTEGER NOT NULL,
    time_id INTEGER NOT NULL,
```

```
location_id INTEGER NOT NULL,
nif INTEGER NOT NULL,
value INTEGER NOT NULL,
duration INTEGER NOT NULL,
PRIMARY KEY (reserva_id, date_id, time_id),
FOREIGN KEY (date_id) REFERENCES
Data (date_id)
ON DELETE CASCADE,
FOREIGN KEY (time_id) REFERENCES
Tempo (time_id)
ON DELETE CASCADE,
FOREIGN KEY (location_id) REFERENCES
Localizacao (location_id)
ON DELETE CASCADE,
FOREIGN KEY (nif) REFERENCES User (nif)
ON DELETE CASCADE
);
```

- storedprocstriggers.sql
(continuação)

```
#populates Date table
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(
            date_id,
            date_day,
            date_week,
            date_month_number,
            date_semester,
            date_year,
            data
        ) 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),
            MONTH(v_full_date) / 6,
            YEAR(v_full_date),
            DATE(v_full_date)
        );
        SET v_full_date = DATE_ADD(v_full_date,
            INTERVAL 1 DAY);
```

```

END WHILE;
END;

```

```

DELIMITER ;

```

```

#populates Time table

```

```

DROP PROCEDURE IF EXISTS

```

```

load_time_dim;

```

```

DELIMITER

```

```

CREATE PROCEDURE load_time_dim()

```

```

BEGIN

```

```

    DECLARE v_full_time DATETIME;

```

```

    SET v_full_time = '1996-04-25 00:00:00';

```

```

    WHILE v_full_time < '1996-04-25 23:59:59'

```

```

DO

```

```

    INSERT INTO Tempo (

```

```

        time_of_day,

```

```

        hour_of_day,

```

```

        minute_of_day

```

```

    ) VALUES (

```

```

        HOUR(v_full_time)*60 +

```

```

        MINUTE(v_full_time),

```

```

        HOUR(v_full_time),

```

```

        MINUTE(v_full_time)

```

```

    );

```

```

    SET v_full_time = DATE_ADD(v_full_time,

```

```

    INTERVAL 1 MINUTE);

```

```

END WHILE;

```

```

END;

```

```

DELIMITER ;

```

- populateDW.sql

```

SET FOREIGN_KEY_CHECKS = 0;

```

```

TRUNCATE Tempo;

```

```

TRUNCATE Data;

```

```

TRUNCATE Localizacao;

```

```

TRUNCATE Reserva_Factos;

```

```

SET FOREIGN_KEY_CHECKS = 1;

```

```

CALL load_date_dim();

```

```

CALL load_time_dim();

```

```

INSERT INTO Localizacao (espaco, edificio) (

```

```

    SELECT

```

```

        codigo,

```

```

        morada

```

```

    FROM Espaco);

```

```

INSERT INTO Localizacao (posto, espaco,
edificio) (

```

```

    SELECT

```

```

        codigo,

```

```

        codigo_espaco,

```

```

        morada

```

```

    FROM Posto);

```

```

INSERT INTO Reserva_Factos (reserva_id,
date_id, time_id, location_id, nif, value,

```

```

duration) (

```

```

    SELECT

```

```

        numero,

```

```

        date_id,

```

```

        time_id,

```

```

        location_id,

```

```

        nif,

```

```

        tarifa * (DATEDIFF(data_fim, data_inicio) +
1),

```

```

        DATEDIFF(data_fim, data_inicio) + 1

```

```

    FROM Paga P

```

```

    NATURAL JOIN Aluga

```

```

    NATURAL JOIN Oferta O

```

```

    NATURAL LEFT JOIN Posto

```

```

    JOIN Localizacao L

```

```

    ON L.edificio = O.morada AND

```

```

        (L.espaco = O.codigo AND

```

```

        L.posto IS NULL) OR

```

```

        (L.posto = O.codigo AND

```

```

        L.espaco = Posto.codigo_espaco)

```

```

    JOIN Data D

```

```

    ON D.date_day = DAY(P.data) AND

```

```

        D.date_month_number =

```

```

    MONTH(P.data) AND

```

```

        D.date_year = YEAR(P.data)

```

```

    JOIN Tempo T

```

```

    ON T.hour_of_day = HOUR(P.data) AND

```

```

        T.minute_of_day = MINUTE(P.data)

```

```

);

```

- olapDW.sql

```

SELECT

```

```

    avg(value),

```

```

    date_month_number,

```

```

    date_day,

```

```

    posto,

```

```

    espaco

```

```

FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY date_month_number,
date_day,
posto,
espaco WITH ROLLUP
UNION
SELECT
    avg(value),
    date_month_number,
    date_day,
    posto,
    espaco
FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY
    date_day,
    posto,
    espaco,
    date_month_number WITH ROLLUP
UNION
SELECT
    avg(value),
    date_month_number,
    date_day,
    posto,
    espaco
FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY
    posto,
    espaco,
    date_month_number,

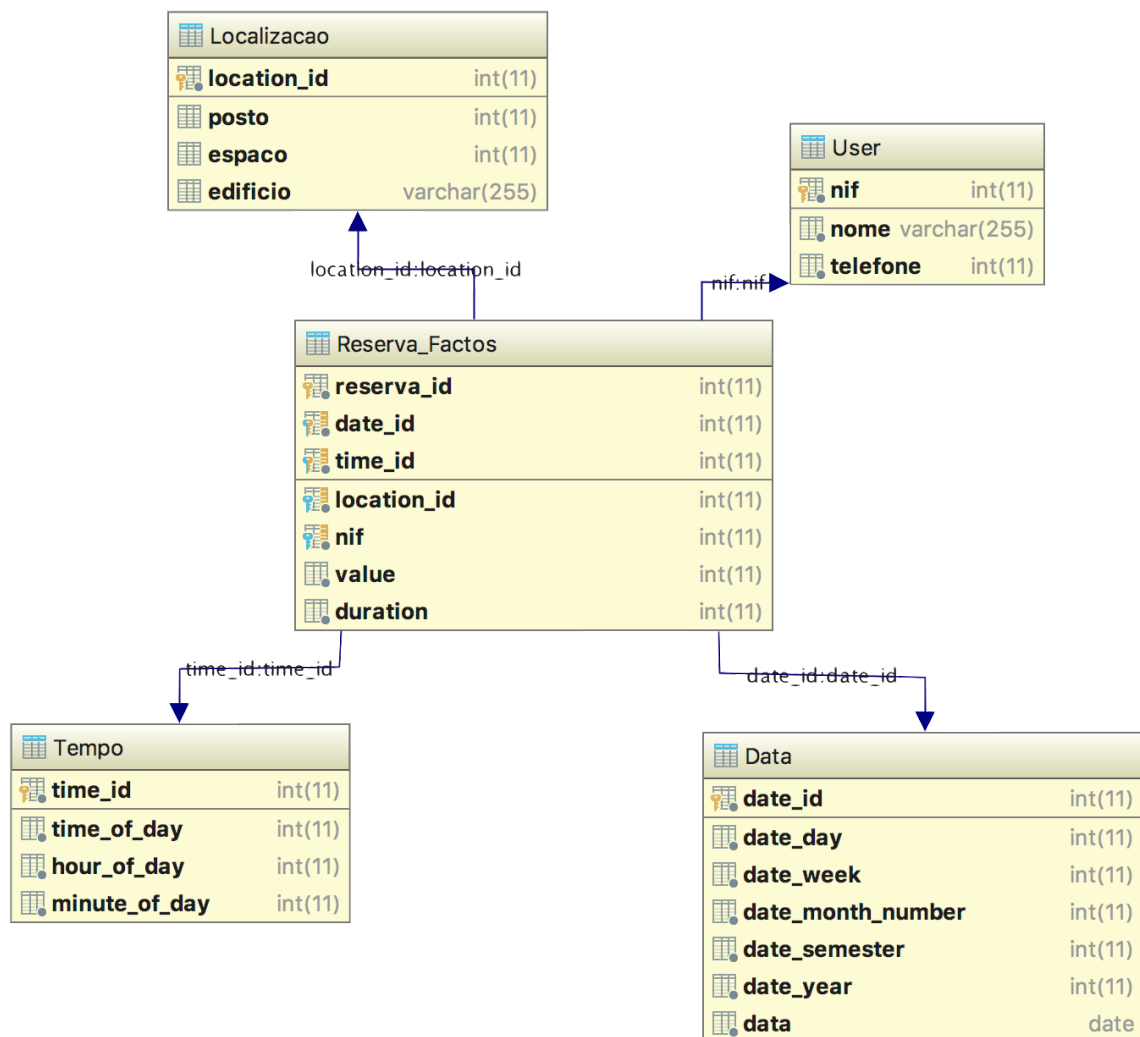
```

```

    date_day WITH ROLLUP
UNION
SELECT
    avg(value),
    date_month_number,
    date_day,
    posto,
    espaco
FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY
    espaco,
    date_month_number,
    date_day,
    posto WITH ROLLUP
UNION
SELECT
    avg(value),
    NULL AS date_month_number,
    date_day,
    NULL AS posto,
    espaco
FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY espaco, date_day
UNION SELECT
    avg(value),
    date_month_number,
    NULL AS date_day,
    posto,
    NULL AS espaco
FROM Reserva_Factos
NATURAL JOIN Data
NATURAL JOIN (SELECT
    IFNULL(posto, -9999) AS posto,
    espaco,
    location_id
FROM Localizacao) A
GROUP BY date_month_number, posto;

```

Esquema em estrela



Na construção da tabela localização é possível ter NULL no código do posto pois para identificar os espaços não é necessário o código deste. Quando se executa o OLAP verifica-se se o campo posto é NULL, se for é substituído por um número por defeito, neste caso é -9999 de forma a não aparecer mais que uma linha com tudo a NULL originada pelo ROLLUP.