

Instituto Superior Técnico

Projeto de Base de Dados,

Parte 4

Ruben Correia nº85909

Raquel Pereira nº76171

Joaquim Esteves nº77020

Grupo nº 99

Turno: Quinta-feira 12:30h – 14:00h

Esforço: 72h

Prof. Gabriel Pestana

**Índices**

**a)**

1.

No caso desta interrogação, não faz sentido a criação de índices tendo em conta que as pesquisas são feitas com base nas primary keys e nas foreign keys, nomeadamente, da tabela Arrenda (morada, código, nif) e da tabela Fiscaliza(id). A nossa base de dados cria automaticamente índices para primary keys e para foreign keys, logo, não há maneira de acelerar a execução desta interrogação.

**2.**

No caso desta interrogação, faria sentido a criação de um índice desagrupado denso por estado da tabela estado. Este índice seria mais eficiente se fosse de dispersão (Hash-based) pois a pesquisa é feita por igualdade (estado = “aceite”) composto estado seria melhor este ser um índice do tipo HASH pois é mais eficiente para comparações de igualdade como o = mas como o nosso MySQL só suport índices do tipo BTREE este não pode ser implementado.

**Data Warehouse**

**1.**

**local\_dimension (DT)**

local\_id (PK)

local\_building

local\_space

local\_post

payed\_amount

**user\_dimension (DT)**

user\_nif (PK)

user\_telephone

user\_name

**reservation\_fact (FT)**

reservation\_id (PK)

local\_id (PK, FK)

date\_id (PK, FK)

user\_nif (PK, FK)

time\_id (PK, FK)

status\_id (PK, FK)

duration\_in\_days

**time\_dimension (DT)**

time\_id (PK)

time\_hour

time\_minute

**date\_dimension (DT)**

date\_id (PK)

date\_day

date\_week

date\_month\_number

date\_semester

date\_year

**status\_dimension (DT)**

status\_id (PK)

status

**Esquema da estrela:**

CREATE TABLE user\_dimension (

user\_nif varchar (9) NOT NULL UNIQUE,

user\_name varchar (80) NOT NULL,

user\_telephone varchar (26) NOT NULL,

primary key(user\_nif));

CREATE TABLE local\_dimension (

local\_id integer NOT NULL UNIQUE,

local\_building varchar (255) NOT NULL,

local\_post varchar (255) DEFAULT 'NO POSTO AVAILABLE',

local\_space varchar (255) DEFAULT 'NO ESPACO AVAILABLE',

primary key (local\_id));

CREATE TABLE time\_dimension (

time\_id integer NOT NULL UNIQUE,

time\_hour integer,

time\_minute integer,

primary key (time\_id));

CREATE TABLE date\_dimension (

date\_id integer NOT NULL UNIQUE,

date\_day integer,

date\_week integer,

date\_month varchar (255),

date\_year integer,

date\_semester integer,

primary key(date\_id));

CREATE TABLE status\_dimension (

status\_id integer NOT NULL UNIQUE,

status varchar (255));

CREATE TABLE reservation\_fact (

reservation\_id varchar (255) NOT NULL UNIQUE,

user\_nif varchar (9) NOT NULL,

local\_id integer NOT NULL,

time\_id integer NOT NULL,

date\_id integer NOT NULL,

status varchar (255) NOT NULL,

payed\_amount numeric (19,4) NOT NULL,

duration\_in\_days integer NOT NULL,

primary key (reservation\_id, user\_nif, local\_id, time\_id, date\_id),

foreign key (user\_nif) references user\_dimension(user\_nif) ON DELETE CASCADE,

foreign key (local\_id) references local\_dimension(local\_id) ON DELETE CASCADE,

foreign key (time\_id) references time\_dimension(time\_id) ON DELETE CASCADE,

foreign key (date\_id) references date\_dimension(date\_id) ON DELETE CASCADE);

(O anexo 1 contém o populate das tabelas)

**2.**

A consulta OLAP, para obter o cubo com o valor médio pago sobre a dimensões localização(local\_id) e data(date\_id), é a seguinte:

SELECT all\_reserved\_local.local\_id, all\_reserved\_local.date\_id, AVG(all\_reserved\_local.payed\_amount) AS avg\_payed

FROM

(SELECT \*

FROM (SELECT l.local\_id as local\_id, l.payed\_amount, r.date\_id

FROM local\_dimension l

INNER JOIN reservation\_fact r

ON l.local\_id = r.local\_id

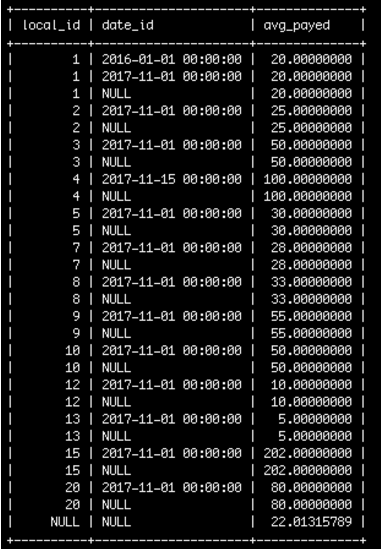
)

AS reserved\_local NATURAL JOIN reservation\_fact

)

AS all\_reserved\_local

GROUP BY all\_reserved\_local.local\_id, all\_reserved\_local.date\_id WITH ROLLUP;



**Fig.1 –** Cubo obtido, como exemplo, a partir da nossa consulta OLAP

**Anexo 1**

DROP PROCEDURE IF EXISTS insert\_time;

DROP PROCEDURE IF EXISTS insert\_date;

DELIMITER $$

CREATE PROCEDURE insert\_time()

BEGIN

SET @time\_id = 0;

SET @time\_hour = 0;

SET @time\_minute = 0;

WHILE @time\_hour < 24

DO

SET @time\_minute = 0;

WHILE @time\_minute < 60

DO

INSERT INTO time\_dimension(time\_id, time\_hour, time\_minute) VALUES(@time\_id, @time\_hour, @time\_minute);

SET @time\_minute = @time\_minute +1;

SET @time\_id = @time\_id+1;

END WHILE;

SET @time\_hour = @time\_hour+1;

END WHILE;

END;

$$

CREATE PROCEDURE insert\_date()

BEGIN

DECLARE v\_full\_date DATETIME;

DECLARE semester integer;

SET v\_full\_date = '2016-01-01 00:00:00';

SET semester = 1;

WHILE v\_full\_date < '2018-01-01 00:00:00' DO

INSERT INTO date\_dimension (

date\_id,

date\_year,

date\_month\_number,

date\_week,

date\_day,

date\_semester

) VALUES (

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

YEAR(v\_full\_date),

MONTH(v\_full\_date),

WEEK(v\_full\_date),

DAY(v\_full\_date),

semester

);

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

IF MONTH(v\_full\_date) >= 2 AND MONTH(v\_full\_date) <= 8 THEN

SET semester = 2;

ELSE

SET semester = 1;

END IF;

END WHILE;

END;

$$

DELIMITER ;