Projeto de Bases de Dados Parte 4

Grupo 19 – Turno BD225179L09 Prof. Gabriel Pestana

81082 – Nuno Gonçalves (12 horas)

81205 – Alice Dourado (12 horas)

81500 – Rodrigo Rato (12 horas)

1. Índices

Query #1 - A)

Índices na tabela arrenda:

- Índice do tipo **HASH** para a coluna **morada**;
- Índice do tipo **HASH** para a coluna **codigo**;
- Índice do tipo **BTREE** para a coluna **nif**;

Índices na tabela fiscaliza:

- Índice do tipo **HASH** para a coluna **morada**;
- Índice do tipo **HASH** para a coluna **codigo**;

Os índices hash são colocados devido à comparação **a.morada = f.morada and a.codigo = f.código**, pois este tipo de índices são mais eficientes para operações de verificação de igualdade entre valores.

O índice BTREE sobre nif na tabela arrenda deve-se ao **group by a.nif**, que fica mais rápido ao percorrer a tabela por ordem, algo que é possível quando se tem este tipo de índice mas que é impossível de fazer num índice em hash onde determinar uma ordem de iteração da hash table é impossível.

Query #1 - B)

Para criar o índice BTREE do arrenda sobre a coluna nif usou-se a seguinte instrução:

CREATE INDEX nif_idx ON arrenda(nif) USING BTREE;

Os índices HASH seriam criados com as seguintes instruções:

CREATE INDEX morada_idx **ON** arrenda(morada) **USING HASH**;

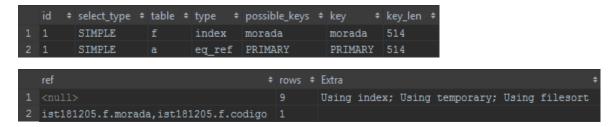
CREATE INDEX codigo_idx ON arrenda(codigo) USING HASH;

CREATE INDEX morada_idx ON fiscaliza(morada) USING HASH;

CREATE INDEX codigo_idx ON fiscaliza(codigo) USING HASH;

No entanto devido a limitações do *software* MySQL não é possível criar índices hash uma vez que o storage engine usado nestas tabelas não suporta este tipo de índices. Também não é necessário criar o índice nif_idx pois está associado a uma primary key da tabela logo é criado um índice automaticamente.

O plano de execução desta query foi o seguinte:



É possível observar que o índice btree que críamos não foi usado neste caso porque no resultado obtido há apenas um nif distinto com o conteúdo que está nas tabelas, mas possivelmente com um conteúdo diferente seria usado se fosse obtida uma maior quantidade de nifs a partir da query.

A primeira linha mostra que é usado um índice chamado morada que foi criado automaticamente na tabela fiscaliza, se fosse possível criar os índices hash que indicámos esses seriam usados em vez deste.

Query #2 – A)

Índice na tabela estado:

- Índice do tipo **HASH** para a coluna estado;

Índice na tabela **posto**:

- Índice do tipo **HASH** para as colunas morada, codigo_espaco;
- Índice do tipo **HASH** para as colunas morada, codigo;

Índice na tabela **aluga**:

- Índice do tipo **HASH** para a coluna numero;

O índice do estado é criado porque, na query, é usada no **where** a condição **e.estado = 'aceite'** pelo que um índice do tipo hash será a melhor escolha para este caso pois este tipo de índice é mais eficiente para operações de verificação de igualdade entre valores.

O índice da tabela posto é devido ao facto de, na cláusula **where** da query efetuada, é usada a condição (**p.morada, p.codigo_espaco**) **not in** (...), o que implica que o valor irá ser comparado com todos os

valores da tabela em segundo argumento e como isto é uma operação de verificação de igualdade o melhor índice para a situação é do tipo hash.

Os restantes índices são utilizados para melhorar o desempenho do **natural join** dentro da cláusula **not in,** que é efetuado através de comparações de valores de colunas com o mesmo nome, algo que é melhorado pelos hash indexes. Só é criado um índice numa das tabelas que possuem colunas correspondentes pois o **natural join** irá sempre percorrer uma das tabelas para verificar ser existe algum correspondente.

Query #2 - B)

Para criar se criarem os índices usaram-se a seguintes instruções:

CREATE INDEX estado_idx **ON** estado(estado) **USING HASH**;

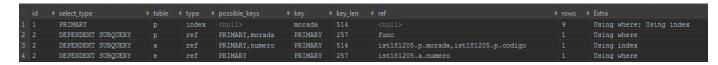
CREATE INDEX posto_idx1 ON posto(morada, codigo_espaco) USING HASH;

CREATE INDEX posto_idx2 **ON** posto(morada, codigo) **USING HASH**;

CREATE INDEX aluga_idx **ON** aluga(numero) **USING HASH**;

Não é possível criar estes índices em MySQL devido às limitações mencionadas anteriormente.

O plano de execução desta query foi o seguinte:



Para além dos índices das primary keys o outro índice usado é o índice 'morada' que foi criado automaticamente na tabela posto, mas se fosse possível criar/usar índices hash o nosso índice iria ser escolhido em detrimento do índice predefinido em btree.

2. Data Warehouse

Schema para a base de dados OLAP:

```
drop table if exists reserva_estrela;
                                          create table local_dim(
drop table if exists tempo_dim;
                                           morada_codigo varchar(510) not null,
drop table if exists local_dim;
                                           edificio varchar(255) not null,
drop table if exists user_dim;
                                           espaco varchar(255) not null,
drop table if exists data_dim;
                                           posto varchar (255),
                                           primary key(morada_codigo)
create table tempo dim(
tempo int not null unique,
hora int not null,
                                          create table user dim(
 minuto int not null.
                                           nif varchar(9) not null unique,
                                           nome varchar(80) not null,
primary key(tempo)
                                            telefone varchar(26) not null,
):
                                           primary key(nif)
                                          );
create table data_dim(
 data int not null unique,
                                          create table reserva estrela(
 dia int not null,
                                             numero varchar(255) not null unique,
                                             montante_pago numeric(19,4) not null,
 semana int not null.
 mes int not null.
                                             duração dias int not null,
 semestre int not null,
 ano int not null,
                                             nif varchar(9) not null,
primary key(data)
                                             morada_codigo varchar(510) not null,
                                             tempo int not null,
);
                                             data int not null,
create table local dim(
                                             foreign key(nif) references user_dim(nif),
morada_codigo varchar(510) not null,
edificio varchar(255) not null,
                                             foreign key(morada_codigo) references
espaco varchar(255) not null,
                                          local dim(morada codigo),
posto varchar (255),
                                             foreign key(tempo) references tempo_dim(tempo),
primary key(morada_codigo)
                                            foreign key(data) references data_dim(data),
                                             primary key(numero, nif, morada codigo, tempo, data)
);
                                          );
```

Query OLAP:

SELECT espaco, posto, dia, mes, AVG(montante_pago) **AS**

AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim GROUP BY espaco, posto, data

UNION

SELECT espaco, posto, dia, **NULL**, *AVG*(montante_pago)

AS AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim GROUP BY espaco,posto, dia

UNION

SELECT espaco, posto, **NULL**, mes, *AVG*(montante pago)

AS AVG MONTANTE PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim GROUP BY espaco, posto, mes

UNION

SELECT espaco, NULL, dia, mes, AVG(montante_pago)

AS AVG MONTANTE PAGO

FROM reserva estrela NATURAL JOIN data dim

NATURAL JOIN local_dim GROUP BY espaco, data

UNION

SELECT NULL, posto, dia, mes, AVG(montante_pago) **AS**

AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim WHERE posto IS NOT NULL GROUP BY posto, data

UNION

SELECT espaco, posto, NULL, NULL,

AVG(montante_pago) AS AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim GROUP BY espaco, posto

UNION

SELECT NULL, NULL, dia, mes, AVG(montante_pago) AS GROUP BY mes

AVG_MONTANTE_PAGO

FROM reserva estrela NATURAL JOIN data dim

NATURAL JOIN local_dim

GROUP BY (data)

UNION

SELECT espaco, **NULL**, dia, **NULL**, *AVG*(montante_pago)

AS AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim

NATURAL JOIN local_dim GROUP BY espaco, dia

UNION

SELECT espaco, NULL, NULL, mes, AVG(montante_pago) AS

AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local_dim
GROUP BY espaco, mes

UNION

SELECT NULL, posto, NULL, mes, AVG(montante_pago) AS

AVG MONTANTE PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local dim

WHERE posto IS NOT NULL

GROUP BY posto, mes

UNION

SELECT NULL, posto, dia, NULL, AVG(montante_pago) AS

AVG MONTANTE PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local dim

WHERE posto IS NOT NULL

GROUP BY posto, dia

UNION

SELECT espaco, **NULL**, **NULL**, **NULL**, *AVG*(montante pago)

AS AVG MONTANTE PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local_dim GROUP BY espaco

UNION

SELECT NULL, posto, NULL, NULL, AVG(montante_pago)

AS AVG MONTANTE PAGO

FROM reserva estrela NATURAL JOIN data dim NATURAL

JOIN local_dim

WHERE posto IS NOT NULL

GROUP BY posto

UNION

SELECT NULL, NULL, mes, AVG(montante_pago) AS

AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local_dim
GROUP BY mes

UNION

SELECT NULL, NULL, dia, NULL, AVG(montante_pago) AS

AVG_MONTANTE_PAGO

FROM reserva_estrela NATURAL JOIN data_dim NATURAL

JOIN local_dim GROUP BY dia

UNION

SELECT NULL, NULL, NULL, NULL, AVG(montante_pago)

AS AVG_MONTANTE_PAGO

FROM reserva estrela NATURAL JOIN data dim NATURAL

JOIN local_dim;

Populate das tabelas do esquema em estrela:

delimiter //	call buildtimedimension();
drop procedure if exists buildtimedimension// create procedure buildtimedimension ()	call builddatedimension();
begin	
declare begin_time TIME;	INSERT INTO user_dim SELECT * from
delete from tempo_dim;	user;
set begin_time = '00:00:00'; while begin_time <= '23:59:00' do	INSERT INTO
insert into	local_dim(`morada_codigo`,`edificio`,`espac
tempo_dim(`tempo`,`hora`,`minuto`)	o`,`posto`)
values (HOUR(begin_time)*10000 +	SELECT
MINUTE(begin_time)*100,HOUR(begin_time),	CONCAT(morada,codigo),morada,codigo_es
MINUTE(begin_time));	paco,codigo
set begin_time = ADDTIME(begin_time,	FROM posto;
'0:1:0.0');	•
end while;	
end//	INSERT INTO
delimiter;	local_dim(`morada_codigo`,`edificio`,`espac
	0`)
delimiter //	SELECT
drop procedure if exists builddatedimension//	CONCAT(morada,codigo),morada,codigo
create procedure builddatedimension ()	FROM espaco;
begin	
declare begin_date DATE;	INSERT INTO
delete from data_dim;	reserva_estrela(`numero`,`nif`,`duracao_dias`
set begin_date = '2016-1-1';	,`montante_pago`,`morada_codigo`,`data`,`te
while begin_date <= '2017-12-31' do	mpo`)
insert into	SELECT
<pre>data_dim(`data`, `dia`, `semana`, `mes`, `semestre`, `ano`)</pre>	numero,nif, datediff (data_fim, data_inicio), datediff (data_fim, data_inicio)*tarifa,
values (YEAR(begin_date) * 10000 +	concat(morada,codigo),YEAR(data) * 10000
MONTH(begin_date)*100 +	+ <i>MONTH</i> (data)*100 + <i>DAY</i> (data),
DAY(begin_date),DAY(begin_date),WEEK(begin_	$HOUR(\mathbf{data})^*10000 + MINUTE(\mathbf{data})^*100$
_date),MONTH(begin_date),IF(MONTH(begin_	FROM aluga NATURAL JOIN oferta
date) $< 7, 1, 2$), YEAR(begin_date));	NATURAL JOIN paga;
set begin_date = $DATE_ADD$ (begin_date,	1 3 · 7
INTERVAL 1 DAY);	
end while;	
end//	
delimiter ;	