Instituto Superior Técnico

Projeto de Bases de Dados

Parte 3

Eduardo Janicas: 78974 (4H)

Diana Antunes: 82448 (6H)

Nuno Fernande: 80774 (4H)

Número de grupo: 44

Turno: BD225179L06

Criação da Base de Dados

SQL

a)

SELECT DISTINCT morada, codigo\_espaco

FROM posto p

WHERE NOT EXISTS (

SELECT morada

FROM aluga a

WHERE p.codigo = a.codigo

);

Selecionámos a morada e o código de espaço de todos os postos exceto dos que já foram ou estão alugados, ou seja, na tabela aluga. Para tal usámos uma sub-query que seleciona a morada dos postos alugados. Ficando apenas com os espaços que têm postos que nunca foram alugados.

b)

SELECT morada

FROM aluga NATURAL JOIN reserva

GROUP BY morada

HAVING COUNT(morada) >= (

SELECT AVG(temp.count)

FROM (

SELECT COUNT(morada) as count

FROM aluga NATURAL JOIN reserva

GROUP BY morada) as temp

);

Selecionámos a morada dos edifícios onde, por morada, verificámos que o número de reservas é superior à média de número de reservas de todos os edifícios. Para tal usámos uma sub-query que seleciona o número de vezes que a morada se encontra reservada e comparamos esse número com a média de modo a ficar com os edifícios com maior número de reservas que a média.

c)

SELECT nif, nome

FROM user

NATURAL JOIN(

SELECT nif

FROM aluga NATURAL JOIN fiscaliza

GROUP BY nif

HAVING COUNT(DISTINCT id) = 1

) AS temp;

Selecionámos o nif e o nome do utilizador cujos alugáveis foram fiscalizados sempre pelo mesmo fiscal. Para tal usámos uma sub-query que para cada nif de utilizador seleciona-o se tiver apenas um id de fiscal associado.

d)

SELECT morada, codigo\_espaco, pago

FROM

(SELECT morada, codigo\_espaco, sum((datediff(data\_fim, data\_inicio))\*tarifa)

as pago

FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN posto

WHERE (data between '2016-01-01 00:00:01' and '2016-12-31 23:59:59')

GROUP BY codigo\_espaco, morada) as temp

UNION

(SELECT morada, codigo as codigo\_espaco, sum((datediff(data\_fim, data\_inicio))\*tarifa) as pago

FROM oferta NATURAL JOIN aluga NATURAL JOIN paga NATURAL JOIN espaco

WHERE (data between '2016-01-01 00:00:01' and '2016-12-31 23:59:59')

GROUP BY codigo\_espaco, morada);

e)

SELECT morada, codigo\_espaco

FROM (SELECT morada, codigo\_espaco, COUNT(codigo) as count

FROM posto

GROUP BY morada, codigo\_espaco) as temp

NATURAL JOIN

(SELECT aceite.morada as morada, aceite.codigo\_espaco as codigo\_espaco, count(aceite.codigo) as count

FROM (SELECT DISTINCT morada, codigo\_espaco, codigo

FROM posto NATURAL JOIN aluga NATURAL JOIN estado

WHERE estado = 'aceite') AS aceite

GROUP BY morada, codigo\_espaco) as total\_aceite;

Restrições de Integridade

a)

DELIMITER //

DROP TRIGGER IF exists overlapping\_offers;

CREATE TRIGGER overlapping\_offers

BEFORE INSERT ON oferta

FOR EACH ROW

BEGIN

IF exists(SELECT \* FROM oferta WHERE codigo = NEW.codigo AND morada = NEW.morada

AND ((new.data\_inicio >= data\_inicio AND new.data\_inicio <= data\_fim)

OR (new.data\_fim >= data\_inicio AND new.data\_fim <= data\_fim)

OR (new.data\_inicio <= data\_inicio AND new.data\_fim >= data\_fim)))

THEN call erro\_overlapping\_offers();

END IF;

END //

DELIMITER ;

b)

DELIMITER //

DROP TRIGGER IF exists pay\_date\_check;

CREATE TRIGGER pay\_date\_check

BEFORE INSERT ON paga

FOR EACH ROW

BEGIN

IF exists(SELECT \* FROM estado WHERE numero = NEW.numero AND NEW.data <= time\_stamp)

THEN call erro\_pay\_date\_check();

END IF;

END //

DELIMITER ;

Aplicação

No modelo