```
FOR EACH ROW(EXECUTE PROCEDURE calc_totpr(pre_upd.quantity,
post_upd.quantity, pre_upd.total_price) INTO total_price)
```

## 3.15 Exercício

Considere as seguintes relações:

```
clientes(ncli: integer, cnome: string, nif string, morada: string,
               telefone string, email string);
quartos(nquarto integer, piso: integer, edificio: integer,
              vista: string, tipo: integer);
tipos(tipo:integer, designacao: string, precobase: double);
reservas(ncli: integer, nquarto: integer, inicio: date, fim: date);
```

Na tabela clientes, registam-se o número do cliente (ncli), o nome (cnome), a morada (morada), o número de telefone de contato (telefone) e o email (email). Na tabela quartos, registam-se o número do quarto (nquarto), o número do piso (piso), o número do edifício (edifício), una descrição sobre as vistas (vista) e o código do tipo de quarto (tipo). Os tipos de quarto estão numa tabela (tipos) que contem esse código (tipo), um texto com a designação (designacao) e o preço base dos quartos desse tipo (precobase). As reservas são registadas por quarto (nquarto) e por cliente (ncli), indicando a data de inicio (inicio) e a data de fim da reserva (fim). A data $fim$, sendo a data de saída, não é considerada como sendo um dia em que o quarto está reservado para aquele cliente. A data $fim$ é sempre estritamente superior à data *inicio*. Calcula-se o número de dias de reserva com a expressão $fim - inicio$.

Estão registados 10000 clientes, existem 500 quartos disponíveis, 5 tipos de quarto e foram feitas 100000 reservas. Para hoje, a taxa de ocupação é de 20%.

Pelo outro lado, na Álgebra Relacional, é possível definir o operador $\varphi$ para calcular operações de agregação sobre tabelas. Por exemplo, a operação sobre $A$, $op(A)$, na tabela $r$ é dada por $\varphi_{op(A)\,as\,Nome}(r)$, sendo o resultado colocado na varável *nome*. É possível também implementar operações de agregação com grupos. Por exemplo, na tabela $r$, para cada $A$, a operação $op(A)$ é dada por ${}_A\varphi_{op(A)\,as\,Nome}(r)$. O resultado é uma tabela com a coluna $A$ e a coluna $Nome$.

Iremos em primeiro lugar criar as tabelas. Note-se apenas que na tabela *reservas* para ter uma chave primária iremos usar uma *sequence*.

```
create table clientes
(ncli number(10) primary key,
 cnome varchar2(100),
 nif varchar2(10),
 morada varchar2(100),
 telefone varchar2(20),
 email varchar2(100));
```

```
create table quartos
(nquarto number(10) primary key,
piso number(2),
edificio number(2),
vista varchar2(100),
tipo number(10));

create table tipos
(tipo number(10) primary key,
designacao varchar2(100),
precobase number(20,2));

create table reservas
(numero number(20) primary key,
ncli number(10),
nquarto number(10),
inicio date,
fim date,
constraint cfim check (fim > inicio)
);

alter table reservas
add (foreign key(ncli) references clientes,
foreign key(nquarto) references quartos);

alter table quartos add
(foreign key(tipo) references tipos);


create sequence nreserva minvalue 0 start with 0;
```

Para testar a nossa base de dados e para testar as interrogações iremos inserir alguns dados nas tabelas.

```
insert into clientes(ncli,cnome) values(1,'Jose');
insert into clientes(ncli,cnome) values(2,'Luis');
insert into clientes(ncli,cnome) values(3,'Paulo');

insert into tipos(tipo,designacao) values(1,'Twin');
insert into tipos(tipo,designacao) values(2,'Duplo');

insert into quartos(nquarto,piso,edificio,vista,tipo)
values(101,1,1,'Mar',1);
```

```
insert into quartos(nquarto,piso,edificio,vista,tipo)
values(102,1,1,'Monte',2);

insert into quartos(nquarto,piso,edificio,vista,tipo)
values(201,2,1,'Mar',1);

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,1,101,
to_date('28-12-2011','dd-MM-yyyy'),
to_date('15-01-2012','dd-MM-yyyy'));

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,1,102,
to_date('28-12-2012','dd-MM-yyyy'),
to_date('15-01-2013','dd-MM-yyyy'));

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,2,201,
to_date('28-12-2012','dd-MM-yyyy'),
to_date('30-01-2013','dd-MM-yyyy'));

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,2,101,
to_date('10-05-2012','dd-MM-yyyy'),
to_date('12-05-2012','dd-MM-yyyy'));

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,3,101,
to_date('10-09-2012','dd-MM-yyyy'),
to_date('11-09-2012','dd-MM-yyyy'));

commit;
```

De seguida iremos responder às seguintes questões:

1. Quais são os nomes e números de telefone dos clientes que têm reserva para HOJE?
2. Quais são os quartos com vista para o mar que não têm reserva para HOJE?
3. Quais são os nomes e emails dos clientes que fizeram pelo menos duas reservas em 2012?
4. Quais são os pisos que têm quartos de todos os tipos?

5. Qual é o nome do cliente com maior número de dias de reserva em 2012? (no caso de existir vários clientes com o mesmo número deverão aparecer todos)

6. Qual é a taxa de ocupação do hotel por tipo de quarto em 2012? (define-se que a taxa de ocupação de um quarto é o rácio entre o número de dias de ocupação do quarto e o número de dias do ano; i.e. 366 dias em 2012).

**Questão 1**:
$\pi_{cnome,telefone}(clientes \propto \sigma_{inicio<=hoje<fim}(reservas))$

```
select cnome,telefone  from clientes,reservas
where clientes.ncli = reservas.ncli
and sysdate between inicio and fim - 1;
```

**Questão 2**:
$\sigma_{vista=Mar}(quartos) - (quartos \propto \sigma_{inicio<=hoje<fim}(reservas))$

```
select * from quartos where vista = 'Mar' minus
(select q.* from quartos q,reservas r
where q.nquarto = r.nquarto and sysdate between inicio and fim - 1);
```

**Questão 3**:
$r2012 = \pi_{numero,ncli,nquarto,maior(inicio,01-01-2012),menor(fim,31-12-2012)}$
$(\sigma_{inicio<01-01-2013\, e\, fim>01-01-2012}(reservas))$
$t = ncli\varphi_{count(*)\, as\, nr}(r2012)$
$\pi_{cnome,email}(clientes) \propto \sigma_{nr>1}(t)$

Iremos primeiro criar duas funções para que as reservas que comecem em anos diferentes de 2012 só considerem os dias de 2012. De seguida cria-se a VIEW com as reservas de 2012.

```
create or replace FUNCTION maior(d1 date,d2 date )
RETURN date is
BEGIN
if d1 > d2 then
  RETURN d1;
else
  return d2;
end if;
END;

CREATE OR REPLACE FUNCTION menor(d1 date,d2 date )
RETURN date is
BEGIN
if d1 > d2 then
  RETURN d2;
```

```
else
  return d2;
end if;
END;

create or replace view r2012 as select numero,ncli,nquarto,
maior(inicio,to_date('01-01-2012','dd-MM-yyyy')) as inicio,
menor(fim,to_date('31-12-2012','dd-MM-yyyy')) as fim
from reservas where to_char(inicio,'yyyymmdd') < '20130101'
and to_char(fim,'yyyymmdd') > '20120101';
```

A resposta á questão é a seguinte:

```
select c.cnome,c.email from clientes c where exists
(select r.ncli,count(*) as c1 from r2012 r where r.ncli = c.ncli
group by r.ncli having count(*) > 1);
```

**Questão 4**:
$\pi_{piso,tipo}(quartos) \div \pi_{tipo}(quartos)$

```
select distinct piso from quartos q1 where not exists
(select * from tipos t1 where not exists
(select * from quartos q2 where q2.tipo = t1.tipo and q2.piso = q1.piso));
```

**Questão 5**:
$dr = \pi_{nquarto,fim-inicio\,as\,dias}(r2012)$
$tr = ncli\varphi_{sum(dias)\,as\,tdias}(dr)$
$ntr = \varphi_{max(tdias)\,as\,m1}(tr)$
$\pi_{cnome}(\sigma_{tdias=ntr}(tr))$
Vamos criar uma view que calcule o número de dias de 2012 de cada reserva.

```
create view nr2012 as select ncli, sum(fim-inicio) as c1
from r2012 group by ncli;
```

A resposta á questão é a seguinte:

```
select cnome from clientes where ncli in
(select ncli from nr2012 where c1 = (select max(c1) from nr2012));
```

**Questão 6**:
$\pi_{nquarto,fim-inicio\,as\,dias}(r2012) = dr2012$
$taxtipo = tipo, nquarto\varphi_{sum(dias)/366\,as\,taxa}(dr2012)$
$tipo\varphi_{avg(taxa)}(taxtipo)$

```
select tipo, avg(taxa) from
(select tipo,nquarto,sum(dias) / 365 as taxa from
(select q1.nquarto,fim-inicio as dias,tipo from r2012,quartos q1
where q1.nquarto = r2012.nquarto)
group by tipo,nquarto)
group by tipo;
```

Admita que é necessário validar que não é possível reservar um quarto para mais de um cliente no mesmo dia. Para o efeito desenvolveu-se o seguinte trigger:

```
create or replace trigger t4
before insert on reservas
for each row
declare d date; c1 number;
begin
loop
d := :new.inicio;
select count(*) into c1 from reservas
where nquarto = :new.nquarto
and d between inicio and fim - 1;
if (c1 > 0) then
RAISE_APPLICATION_ERROR(-20000, 'Reserva nao disponivel');
end if;
d := d + 1;
if d >= :new.fim then
  exit;
end if;
end loop;
end trigger;
```

Para testar o trigger, iremos tentar executar os seguintes comandos:

```
insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,3,101,
to_date('12-09-2012','dd-MM-yyyy'),
to_date('13-09-2012','dd-MM-yyyy'));

insert into reservas(numero,ncli,nquarto,inicio,fim)
values(nreserva.nextval,1,101,
to_date('12-09-2012','dd-MM-yyyy'),
to_date('13-09-2012','dd-MM-yyyy'));
```

O segundo $INSERT$ dá erro enquanto o primeiro é executado com sucesso.

Iremos agora calcular o custo da expressão:
$\pi_{cnome,telefone}(clientes \bowtie \sigma_{inicio<=hoje<fim}(reservas))$
usando a tabela:

| Expressão | Custo |
|---|---|
| $t_1$ | $card(t_1)$ se $t_1$ é um operando simples |
| $t_1$ | $custo(t_1)$ se $t_1$ é uma operação |
| $t_1 \otimes t_2$ | $card(t_1) * card(t_2) + custo(t_1) + custo(t_2)$ |
| $t_1 \bowtie_{A_i} t_2$ | $card(t_1) + card(t_2) + custo(t_1) + custo(t_2)$ |
| $t_1 \cup t_2$ | $card(t_1) + card(t_2) + custo(t_1) + custo(t_2)$ |
| $t_1 \setminus t_2$ | $card(t_1) + card(t_2) + custo(t_1) + custo(t_2)$ |
| $\sigma_{Cond}(t_1)$ | $card(t_1) + custo(t_1)$ |
| $\Pi_{A_i,...,A_j} t_1$ | $custo(t_1)$ |

O custo da expressão é:
$custo1 = card(clientes)+card(\sigma_{inicio<=hoje<fim}(reservas))+custo(clientes)+custo(\sigma_{inicio<=hoje<fim}(reservas))$

Considerando que:
$custo(\sigma_{inicio<=hoje<fim}(reservas)) = card(reservas) + custo(reservas) = 2 * card(reservas)$
$custo1 = 10000 + 100 + 10000 + 2 * 100000 = 220100$

Uma expressão equivalente à anterior é:
$\pi_{cnome,telefone}\sigma_{inicio<=hoje<fim}(clientes \bowtie reservas)$

Nesse caso o custo é:
$custo2 = card(clientes \bowtie reserves) + card(clientes) * 2 + card(reserves) * 2$
$custo2 = 100000 + 20000 + 200000 = 320000$

A primeira expressão é mais eficiente de que a segunda porque tem um custo inferior.