



# SISTEMAS DE INFORMAÇÃO E BASES DE DADOS

2<sup>A</sup> PARTE DO PROJETO

GRUPO 13

Diogo Proença, 75313                      Diogo Martins, 75462  
Bernardo Gomes, 75573

25 de Novembro de 2015

## Conteúdo

<b>1</b>	<b>Criação das tabelas da base de dados</b>	<b>1</b>
<b>2</b>	<b><i>Triggers</i> para prevenção de <i>overlapping periods</i></b>	<b>4</b>
2.1	<i>Triggers</i> de <i>insert</i> . . . . .	5
2.2	<i>Triggers</i> de <i>update</i> . . . . .	8
<b>3</b>	<b><i>SQL queries</i></b>	<b>9</b>

# 1 Criação das tabelas da base de dados

Para a criação das tabelas na base de dados, as instruções de SQL utilizadas foram as seguintes:

```
create table Patient(  
    number varchar(40),  
    name varchar(255),  
    address varchar(255),  
    primary key(number));  
  
create table PAN(  
    domain varchar(255),  
    phone varchar(20),  
    primary key(domain));  
  
create table Device(  
    serialnum integer,  
    manufacturer varchar(255),  
    description varchar(255),  
    primary key(serialnum , manufacturer));  
  
create table Sensor(  
    snum integer,  
    manuf varchar(255),  
    units varchar(255),  
    primary key(snum, manuf),  
    foreign key(snum, manuf)  
        references Device(serialnum , manufacturer));  
  
create table Actuator(  
    snum integer,  
    manuf varchar(255),  
    units varchar(255),  
    primary key(snum, manuf),  
    foreign key(snum, manuf)  
        references Device(serialnum , manufacturer));  
  
create table Municipality(  
    nut4code integer,  
    name varchar(255),  
    primary key(nut4code));  
  
create table Period(  
    start date,  
    end date,  
    check(start<=end),  
    primary key(start , end));
```

```

create table Reading(
    snum integer,
    manif varchar(255),
    datetime timestamp,
    value numeric(20,2),
    primary key(snum, manif, datetime),
    foreign key(snum, manif) references Sensor(snum, manif));

create table Setting(
    snum integer,
    manif varchar(255),
    datetime timestamp,
    value numeric(20,2),
    primary key(snum, manif, datetime),
    foreign key(snum, manif) references Actuator(snum, manif));

create table Wears(
    start date,
    end date,
    patient varchar(255),
    pan varchar(255),
    check(start<=end),
    primary key(start, end, patient),
    foreign key(start, end) references Period(start, end),
    foreign key(patient) references Patient(number),
    foreign key(pan) references PAN(domain));

create table Lives(
    start date,
    end date,
    patient varchar(255),
    muni integer,
    check(start<=end),
    primary key(start, end, patient),
    foreign key(start, end) references Period(start, end),
    foreign key(patient) references Patient(number),
    foreign key(muni) references Municipality(nut4code));

create table Connects(
    start date,
    end date,
    snum integer,
    manif varchar(255),
    pan varchar(255),
    check(start<=end),
    primary key(start, end, snum, manif),
    foreign key(start, end) references Period(start, end),
    foreign key(snum, manif) references Device(serialnum, manufacturer),
    foreign key(pan) references PAN(domain));

```

Relativamente às escolhas das variáveis o seguinte conjunto merece especial destaque:

- **variável *number* da tabela *Patient*** → ao termos levado a cabo pesquisa relativa a *Social Security Numbers* (SSN's) válidos, verificámos que este conjunto de números pode conter caracteres não numéricos (-). Além deste facto, pode também começar pelo número "0", o que invalida o uso de variáveis *integer*, *numeric* e *decimal* caso contrário o número armazenado iria perder dígitos. Utiliza-se assim, a variável *varchar*;
- **variável *phone* da tabela *PAN*** → tendo em conta a existência de números de telefone com prefixo diferente em cada país (*e.g.* Portugal - +351), e não sendo necessário que o paciente tenha um número do país onde o *Medical Center* está localizado, utiliza-se a variável *varchar*;
- **variável *serialnum* da tabela *Device*** → utiliza-se a variável *integer*. No entanto, poder-se-ia considerar também do tipo *varchar* pelo facto de em alguns casos os números de série conterem caracteres. No entanto, não tendo sendo especificado nada no enunciado, e não tendo obtido nenhuma informação sobre os números de série de aparelhos médicos, considera-se o caso em que estes podem ser representados por um número;
- Para as tabelas *Sensor* e *Actuator* é utilizado o mesmo critério que a tabela anterior;
- **variável *nut4code* da tabela *Municipality*** → para esta tabela, considera-se apenas os códigos postais semelhantes a Portugal, identificados por quatro números, tal como o nome da variável indica;
- **variáveis *start* e *end* da tabela *Period*** → escolhe-se o tipo *date* pelo facto de as relações com esta entidade não necessitar de precisões ao nível HH:mm:ss;
- para as tabelas *Period*, *Wears*, *Lives* e *Connects* foi colocada uma verificação das datas *check(start<=end)* de forma a garantir que o período está consistente;
- **variável *datetime* das tabelas *Reading* e *Setting*** → escolhe-se o tipo *timestamp* pelo facto de as medições médicas necessitarem de precisões ao nível HH:mm:ss;
- **variável *value* das tabelas *Reading* e *Setting*** → escolhe-se o tipo *numeric* pelo facto de permitir precisão ao nível de casas decimais.

Inicialmente, acrescentam-se as seguintes instruções de forma a apagar eventuais tabelas com o mesmo nome antes da criação das novas:

```
drop table if exists Reading;
drop table if exists Setting;
drop table if exists Sensor;
drop table if exists Actuator;
drop table if exists Connects;
drop table if exists Device;
drop table if exists Lives;
drop table if exists Wears;
drop table if exists Patient;
drop table if exists PAN;
drop table if exists Municipality;
drop table if exists Period;
```

## 2 *Triggers* para prevenção de *overlapping periods*

A função dos *triggers* é de prevenir a inserção/actualização de associações a PANs por parte de pacientes ou de aparelhos médicos em periodos de tempo sobrepostos.

A *error message* deve ser apresentada de forma a evitar que ocorram casos, para a tabela *Wears*, como os seguintes:

- Paciente 1 está ligado a PAN1 e PAN2 ao mesmo tempo;
- Paciente 1 e Paciente 2 estão ligados à PAN1 ao mesmo tempo.

A *error message* deve ser apresentada de forma a evitar que ocorram casos, para a tabela *Connects*, como o seguinte:

- Device 1 está ligado a PAN1 e PAN2 ao mesmo tempo.

A condição de sobreposição de períodos, independentemente da tabela em questão obtém-se através da análise do problema com a aplicação das leis de De Morgan. Ora, não existe sobreposição de períodos em dois casos:

1. o intervalo inserido/atualizado é completamente anterior aos intervalos já inseridos na tabela;
2. o intervalo inserido/atualizado é completamente posterior aos intervalos já inseridos na tabela.

Seja A um intervalo definido por startA e endA e o intervalo B definido por startB e endB. Os intervalos A e B não se encontram sobrepostos sempre que:  $(startA \geq endB) \cup (endA \leq startB)$ .

Negando a condição obtém-se todos os casos para os quais os intervalos A e B se encontram sobrepostos. Assim, através das leis de De Morgan obtém-se a seguinte condição para dois intervalos sobrepostos:  $(startA \leq endB) \cap (endA \geq startB)$ . Esta condição é aplicada em todos os *triggers* que se apresentam de seguida.

## 2.1 *Triggers de insert*

Para a tabela *Connects*, não se pode inserir o mesmo *device* em duas PANs distintas no mesmo período de tempo. Desta forma, para cada linha dentro da tabela onde se pretende inserir o novo aparelho, será avaliado *a priori* se este já se encontra contido nos registos da tabela. Posteriormente, avalia-se se este *device* possui um PAN diferente, associado em períodos de tempo sobrepostos aos que se encontram nos registos da tabela.

O *trigger* para a tabela *Connects* tem o seguinte código:

```
drop trigger if exists check_overlap_time_period_Device_PAN;

delimiter $$

create trigger check_overlap_time_period_Device_PAN
    before insert on Connects

for each row
begin
    if(
        exists(SELECT start , end FROM Connects
                WHERE(
                    (Connects.start <= new.end) and
                    (Connects.end >= new.start)
                    AND ((new.snum = Connects.snum) and
                       (new.manuf = Connects.manuf))
                    AND (new.pan != Connects.pan)
                )
            )
        )
        then
            call overlapping_time_periods_for_Device_PAN();
    end if;

end$$

delimiter ;
```

Relativamente à tabela *Wears*, o raciocínio aplicado é semelhante. Um paciente não pode ter duas PANs ao mesmo tempo e dois pacientes não podem partilhar a mesma PAN. Este caso, gera uma condição adicional, que não surgia no caso anterior. As condições são assim semelhantes às apresentadas anteriormente, sendo ainda verificado se o paciente se encontra na tabela e se a PAN que se pretende que o mesmo utilize está actualmente em utilização por um paciente distinto. No caso de o paciente se encontrar nos registos, verifica-se se a PAN que está a utilizar é a mesma e única no período de tempo.

O *trigger* para a tabela *Wears* tem o seguinte código:

```
drop trigger if exists check_overlap_time_period_Patient_PAN;

delimiter $$

create trigger check_overlap_time_period_Patient_PAN
                before insert on Wears

for each row
begin
if(
    exists(SELECT start , end FROM Wears
           WHERE(
               (Wears.start <= new.end) and
               (Wears.end >= new.start)
               AND (((new.patient != Wears.patient)
                   AND (new.pan = Wears.pan))
               OR ((new.patient = Wears.patient)
                   AND (new.pan != Wears.pan))))
           )
    )
    then
        call overlapping_time_periods_for_Patient_PAN();
    end if;

end$$

delimiter ;
```

Por forma a testar o funcionamento dos *triggers* atrás apresentados, utilizámos a base de dados fornecida no ficheiro *database\_triggers\_insert.sql*. Após correr as instruções, obtêm-se as seguintes tabelas:

```
mysql> select * from Connects;
+-----+-----+-----+-----+-----+
| start | end   | snum  | manuf | pan   |
+-----+-----+-----+-----+-----+
| 2011-10-09 | 2012-12-01 | 123456789 | Philips | www.pan1.pt |
| 2014-12-25 | 2015-01-01 | 123456790 | Philips | www.pan1.pt |
| 2015-04-01 | 2015-10-25 | 123456789 | Philips | www.pan1.pt |
| 2015-10-26 | 2015-11-26 | 123456789 | Philips | www.pan1.pt |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Figura 1: Tabela *Connects*



```
mysql> select * from Wears;
+-----+-----+-----+-----+
| start | end   | patient | pan   |
+-----+-----+-----+-----+
| 2011-10-09 | 2012-12-01 | 001-54245-1555555 | www.pan1.pt |
| 2014-12-25 | 2015-01-01 | 001-54245-1555555 | www.pan1.pt |
| 2015-04-01 | 2015-10-25 | 001-54245-1555555 | www.pan1.pt |
| 2015-10-26 | 2015-11-26 | 001-54245-1555555 | www.pan1.pt |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Figura 2: Tabela *Wears*

Os testes realizados foram sucessivas inserções de um dispositivo em PANs diferentes sobreposto a pelo menos uma entrada da tabela.

Os testes realizados estão contidos nos ficheiros *teste\_insert\_connects.sql* e *teste\_insert\_connects.sql*, apresentados em anexo e que contem uma série de instruções e respectivos comentários.

Correndo os ficheiros de teste verifica-se um de quatro dos seguintes resultados, consoante as tabelas a alterar e consoante seja uma *update*/inserção válido ou inválido:

```
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Device_PAN does not exist
```

Figura 3: Mensagem de erro de inserção para a tabela *Connects*

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`ist175573`.`Connects`, CONSTRAINT `Connects_ibfk_1` FOREIGN KEY (`start`, `end`) REFERENCES `Period` (`start`, `end`))
```

Figura 4: Mensagem sucesso de inserção para a tabela *Connects*

```
ERROR 1305 (42000): PROCEDURE ist175462.overlapping_time_periods_for_Patient_PAN does not exist
```

Figura 5: Resultados dos testes do *trigger* de inserção para a tabela *Connects*

```
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`ist175462`.`Wears`, CONSTRAINT `Wears_ibfk_1` FOREIGN KEY (`start`, `end`) REFERENCES `Period` (`start`, `end`))
```

Figura 6: Resultados dos testes do *trigger* de inserção para a tabela *Connects*

Em caso de sucesso, ou seja, sem sobreposição, surge uma mensagem de erro devido ao facto de o período inserido não existir na tabela *Period*. Caso estivesse, não ocorreria o erro e a tabela seria atualizada.

## 2.2 Triggers de update

À semelhança dos testes que realizámos para os *triggers* de *insert* para as duas tabelas, os ficheiros de teste para os *triggers* de *update* encontram-se no ficheiro .zip, com os nomes *teste\_update\_connects.sql* e *teste\_update\_wears.sql*.

Os *triggers* relativos a esta secção só diferem dos anteriores na medida em que são efectuados antes de um *update* em vez de antes de um *insert*. Assim, nos troços de código anteriores, onde se lê

```
create trigger check_overlap_time_period_Patient_PAN before insert on Wears
```

 deve-se ler

```
create trigger check_overlap_time_period_Patient_PAN before update on Wears
```

 e onde se lê

```
create trigger check_overlap_time_period_Device_PAN before insert on Connects
```

 deve-se ler

```
create trigger check_overlap_time_period_Device_PAN before insert on Connects.
```

No entanto, para esta secção foi utilizado um maior número de testes pois existe um maior número de casos de atualizações possíveis do que inserções.

Assim, a base de dados utilizada para os testes foi a presente no ficheiro *database\_triggers\_update.sql*, sendo obtidas as seguintes tabelas:

```
mysql> select * from Connects;
```

start	end	snum	manuf	pan
2011-10-09	2012-12-01	123456789	Philips	www.pan1.pt
2014-12-25	2015-01-01	123456790	Philips	www.pan1.pt
2015-04-01	2015-10-25	123456789	Philips	www.pan1.pt
2015-10-26	2015-11-26	123456789	Philips	www.pan1.pt
2013-11-25	2013-12-01	123456789	Philips	www.pan3.pt

5 rows in set (0.00 sec)

Figura 7: Tabela *Connects*

```
mysql> select * from Wears;
```

start	end	patient	pan
2011-10-09	2012-12-01	001-54245-1555555	www.pan1.pt
2013-11-25	2013-12-01	001-54245-1555575	www.pan1.pt
2015-04-01	2015-10-25	001-54245-1555555	www.pan1.pt
2015-10-26	2015-11-26	001-54245-1555555	www.pan1.pt
2014-12-25	2015-01-01	001-54245-1555555	www.pan3.pt

5 rows in set (0.00 sec)

Figura 8: Tabela *Wears*

Na tabela da figura 7, a linha que é atualizada é sempre a última, correspondente a um *device* igual a pelo menos um que se encontra na tabela. Porém, possui um PAN diferente destes.

Na tabela da figura 8, as linhas que são atualizadas são a segunda e última linhas que testam, respectivamente, dois pacientes possuírem a mesma PAN simultaneamente e o mesmo paciente ter duas PANs simultaneamente.

### 3 SQL queries

(a)

A *query* realizada para a alínea em questão foi a seguinte (presente no ficheiro *3a.sql*):

```
drop procedure if exists
    blood_pressure_prev6m_readings_patient;

delimiter $$

create procedure blood_pressure_prev6m_readings_patient
    (in p_number varchar(40))

begin

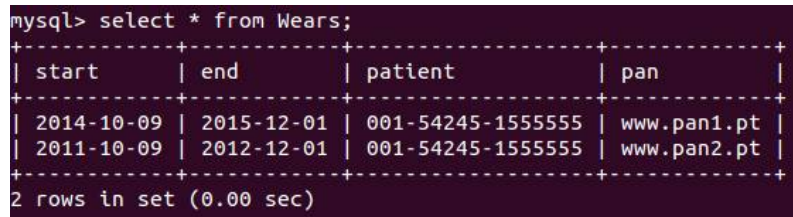
SELECT DISTINCT W.patient , R.datetime ,
                R.value FROM Wears as W,
                Connects as C, Reading as R, Device as D
        WHERE(
            (R.datetime >= DATE_SUB(CURDATE() , INTERVAL
6 MONTH))
            AND ((R.snum = D.serialnum)
            AND (R.manuf = D.manufacturer))
            AND ((R.snum = C.snum)
            AND (R.manuf = C.manuf))
            AND ((date(R.datetime) >= C.start)
            AND (date(R.datetime) <= C.end))
            AND ((date(R.datetime) >= W.start)
            AND (date(R.datetime) <= W.end))
            AND (C.pan = W.pan)
            AND ((D.description = 'blood_pressure')
            AND (W.patient = p_number));

end$$

delimiter ;
```

Para testar o bom funcionamento do *procedure*, utilizámos a base de dados presente em *database3a.sql*, com o ficheiro de teste *teste3a.sql*.

As tabelas obtidas deverão ser as seguintes:



```
mysql> select * from Wears;
```

start	end	patient	pan
2014-10-09	2015-12-01	001-54245-1555555	www.pan1.pt
2011-10-09	2012-12-01	001-54245-1555555	www.pan2.pt

2 rows in set (0.00 sec)

Figura 9: Tabela *Wears*

```
mysql> select * from Connects;
```

start	end	snum	manuf	pan
2015-04-01	2015-04-25	123456789	Philips	www.pan1.pt
2015-04-01	2015-10-25	123456789	RPG	www.pan1.pt
2015-04-01	2015-10-25	123456789	Sony	www.pan1.pt
2015-04-01	2015-04-25	123456789	Samsung	www.pan2.pt
2015-04-01	2015-10-25	123456789	LG	www.pan2.pt
2015-04-01	2015-10-25	123456789	Panasonic	www.pan2.pt

```
6 rows in set (0.00 sec)
```

Figura 10: Tabela *Connects*

```
mysql> select * from Reading;
```

snum	manuf	datetime	value
123456789	LG	2015-10-24 09:45:00	13.00
123456789	Panasonic	2015-01-24 09:45:00	20.00
123456789	Philips	2015-10-24 09:45:00	11.00
123456789	RPG	2015-10-24 09:45:00	10.00
123456789	Samsung	2015-10-24 09:45:00	14.00
123456789	Sony	2015-01-24 09:45:00	12.00

```
6 rows in set (0.00 sec)
```

Figura 11: Tabela *Reading*

```
mysql> select * from Device;
```

serialnum	manufacturer	description
123456789	LG	blood pressure
123456789	Panasonic	blood pressure
123456789	Philips	blood pressure
123456789	RPG	blood pressure
123456789	Samsung	blood pressure
123456789	Sony	blood pressure
123456790	Philips	insuline meter
123456791	Philips	insuline meter

```
8 rows in set (0.01 sec)
```

Figura 12: Tabela *Device*

Foram criados seis *devices* de diferentes marcas, sendo que metade (RPG, Philips, Sony) foram associados à PAN cujo domínio é *www.pan1.pt* e a restante (LG, Samsung, Panasonic) é colocada no PAN cujo domínio é *www.pan2.pt*. Apenas um paciente, identificado pelo número 001-54245-1555555 irá conectar-se a estas PANs em dois períodos de tempo diferentes.

Para o teste referido o resultado obtido será o seguinte:

```
mysql> call blood_pressure_prev6m_readings_patient('001-54245-155555');
+-----+-----+-----+
| patient | datetime | value |
+-----+-----+-----+
| 001-54245-155555 | 2015-10-24 09:45:00 | 10.00 |
+-----+-----+-----+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.01 sec)
```

Figura 13: Resultado do teste da *query* 3 (a)

A explicação deste resultado, pode-se resumir em três pontos:

1. O Reading de RPG encontra-se dentro do período da data actual até menos 6 meses, Philips também mas Sony não. Por outro lado no PAN2 o Reading de LG encontra-se dentro do período da data actual até menos 6 meses, Samsung também, mas Panasonic não;
2. O Reading de RPG encontra-se no período em que o Device está conectado ao PAN1, mas o Reading de Philips não. No PAN2 o Reading de LG encontra-se dentro do período em que o Device está conectado ao PAN2, mas o Reading de Samsung não;
3. O Reading de RPG encontra-se no período em que o Patient veste o PAN1, mas o Reading de LG não se encontra no período em que o Patient veste o PAN2.

Desta forma, o primeiro ponto exclui Sony e Panasonic, o segundo ponto exclui Philips e Samsung e o terceiro Ponto exclui LG, restando RPG. A *reading* associada a este aparelho encontra-se apresentada na figura 13.

(b)

A *query* realizada para a alínea em questão foi a seguinte (presente no ficheiro *3b.sql*):

```
SELECT max(m.name) , m.nut4code
FROM Municipality as m, Device as d, Lives as l,
     Wears as w, Connects as c, Patient as pat
WHERE(
    (l.muni = m.nut4code)
    AND ((l.patient = w.patient) AND (l.patient = pat.number))
    AND ((w.patient = pat.number) AND (w.pan = c.pan))
    AND ((c.manuf = d.manufacturer) AND (c.snum = d.serialnum))
    AND ((w.start <= current_date) AND (l.start <= current_date)
    AND (c.start <= current_date)) AND ((w.end >= current_date)
    AND (l.end >= current_date) AND (c.end >= current_date))
    AND (d.manufacturer = 'Philips'));
```

Para testar o bom funcionamento do *procedure*, utilizámos a base de dados presente em *database3b.sql*, com o ficheiro de teste *teste3b.sql*.



O resultado obtido deverá ser o seguinte:

```
mysql> source 3b.sql
+-----+-----+
| max(m.name) | nut4code |
+-----+-----+
| Quarteira   | 8125     |
+-----+-----+
1 row in set (0.01 sec)
```

Figura 14: Resultado do teste da *query* 3 (b)

Para o teste em questão, foram criados nove *Devices* com manufacturer Philips com serial number diferentes mas consecutivos, três pacientes correspondentes aos elementos do grupo, três Municípios onde cada elemento viverá e quatro PANs.

Para testar, fez-se com que o paciente Bernardo utilizasse duas PANS: PAN1 e PAN2. O paciente Diogo utilizasse a PAN3 e o paciente Proença utilizasse a PAN4.

A PAN1 possui três *Devices*, PAN2 apenas um, PAN3 dois, e por fim PAN4 três *Devices*, tendo todos os dispositivos descrições diferentes.

Os *Patients* Bernardo e Diogo vivem em Quarteira e Alcochete respectivamente num período que abrange a data actual, enquanto que o *Patient*. Proença vive no Montijo num período que não abrange a data actual.

Dos quatro *Devices* que se encontram nos dois PANs que Bernardo utiliza, apenas três cumprem as especificações todas. Bernardo utiliza o PAN1 numa data onde a data actual se encontra contida porém tal não acontece com PAN2, utilizando este PAN num período que não inclui a data actual.

Todos os *Devices* que se encontram nas Pans de Diogo e de Proença cumprem todas as especificações de tempo tanto de *Wears* como *Connects* em relação à data actual.

Como Bernardo possui três *Devices* que cumprem todas as especificações vence perante Diogo uma vez que este possui apenas dois.

Como Bernardo vive num município numa data que abrange a actual vence perante Proença uma vez que este não cumpre esta condição.

Assim o município que possui agora o maior número de *Devices Philips* será Quarteira.

(c)

A *query* realizada para a alínea em questão foi a seguinte (presente no ficheiro *3c.sql*):

```
SELECT distinct manufacturer
FROM Device as d, Connects as c, Wears as w, Lives as l
WHERE(
  ((w.start >= '2014-01-01') OR (w.end <= '2014-12-31'))
  AND (c.start <= w.end) AND (c.end >= w.start)
  AND (l.start <= w.end) AND (l.end >= w.start)
  AND (w.pan = c.pan)      AND (w.patient = l.patient)
  AND ((d.manufacturer = c.manuf) AND (d.serialnum = c.snum))
  AND (d.description = 'scale'));
```

Para testar o bom funcionamento do *procedure*, utilizámos a base de dados presente em *database3c.sql*, com o ficheiro de teste *teste3c.sql*. O resultado obtido deverá ser o seguinte:

```
mysql> source 3c.sql
+-----+
| manufacturer |
+-----+
| Panasonic    |
| Philips      |
+-----+
2 rows in set (0.00 sec)
```

Figura 15: Resultado do teste da *query* 3 (c)

Foram criados oito *Devices* todos descritos por *scale*, mas com *serial number* e *manufacturers* diferentes. Estes oito *Devices* foram divididos por quatro *Patients* correspondentes aos três elementos do grupo e o sem-abrigo Zé-Manel.

O primeiro paciente Bernardo Gomes vive em Quarteira no ano de 2014, e utiliza o PAN1 durante os anos de 2013 e 2014, sendo que essa PAN tem dois *Devices* (*Philips* e *RPG*) conectados, um entre os anos 2013 e 2014 e o outro no ano de 2015.

O segundo paciente Diogo Martins vive em Alcochete no ano de 2014, e utiliza o PAN2 e PAN3 em duas alturas distintas, o primeiro entre 2014 e 2015 e o segundo entre 2012 e 2013, sendo que o PAN2 possui um *Device* (*Panasonic*) ligado nos anos 2013 a 2015. O PAN3 tem também um dispositivo (*LG*) ligado nos anos 2014 e 2014.

O terceiro paciente Diogo Proença vive no Montijo entre 2011 e 2012, utilizando o PAN4 entre 2013 e 2014, sendo que o PAN4 possui dois *Devices* (*Sony* e *Samsung*) associados, ambos ligados entre 2013 e 2014.

O último paciente Zé Manel encontra-se de momento sem-abrigo, ou não vive em nenhum município presente na base de dados do *Medical Center*. Porém utiliza o PAN5 entre 2013 e 2014 com dois *Devices* (*Siemens* e *HP*) instalados nesse PAN entre 2013 e 2014.

O resultado da Query será o apresentado na figura 15, com os seguintes dispositivos:

- *Philips* → pois encontra-se conectado entre 2013 e 2014 num intervalo contido no período em que Bernardo utiliza o PAN durante o ano de 2014, acrescentado ao facto que Bernardo vive num município num intervalo também contido no período em que utiliza o PAN;
- *Panasonic* → pois encontra-se conectado a PAN2 entre 2014 e 2015, num intervalo contido no período em que Diogo utiliza o PAN durante o ano de 2014, acrescentado ao facto que Diogo vive num município num intervalo também contido no período em que utiliza o PAN.

Os dispositivos restantes apresentados de seguida, não cumprem as especificações pelas seguintes razões:

- *RPG* → não cumpre a especificação de estar ligado ao PAN na mesma altura que Bernardo vive num município e utiliza essa PAN durante o ano de 2014.

- *LG*  $\rightarrow$  não cumpre a especificação de Diogo utilizar a PAN durante o ano de 2014 simultaneamente quando vive num município em 2014 e ter *Devices* ligados à PAN.
- *Sony e Samsung*  $\rightarrow$  não cumprem a especificação de Proença estar a viver num município durante um ano de 2014 simultaneamente a estar a utilizar uma PAN e essa PAN ter tais *Devices* ligados.
- *Siemens e HP*  $\rightarrow$  não cumprem a especificação pois Zé Manel não se encontra a viver num município coberto pelo Medical Center.