



SISTEMAS DE INFORMAÇÃO E BASES DE DADOS

2^A PARTE DO PROJETO

GRUPO 13

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

25 de Novembro de 2015

Conteúdo

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

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 de associações a PAN's 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.

2.1 *Triggers de insert*

Os triggers desenvolvidos foram os seguintes:

```
delimiter $$
create trigger check_overlap_time_period_Patient_PAN

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 ;

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 ;
```

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)

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 1: Tabelas para teste de *insert* dos *triggers*

Os testes realizados foram sucessivas inserções de um dispositivo em PAN's 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 verificam-se os seguintes resultados:

```
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Device_PAN d
oes not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Device_PAN d
oes not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Device_PAN d
oes not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Device_PAN d
oes not exist
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails ('ist175573`.`Connects`, CONSTRAINT `Connects_ibfk_1` FOREIGN KEY (`start`,
`end`) REFERENCES `Period` (`start`, `end`))
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails ('ist175573`.`Connects`, CONSTRAINT `Connects_ibfk_1` FOREIGN KEY (`start`,
`end`) REFERENCES `Period` (`start`, `end`))
```

Figura 2: Resultados dos testes do *trigger* de inserção para a tabela *Connects*

Verifica-se deste modo, que, tal como era esperado, as quatro primeiras tentativas de inserção de valores na tabela falharam devido à existência de *overlap* e as restantes seriam inseridas caso os *time periods* estivessem na tabela *Period*.


```

ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails (`ist175573`.`Wears`, CONSTRAINT `Wears_ibfk_1` FOREIGN KEY (`start`, `end`
) REFERENCES `Period` (`start`, `end`))
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails (`ist175573`.`Wears`, CONSTRAINT `Wears_ibfk_1` FOREIGN KEY (`start`, `end`
) REFERENCES `Period` (`start`, `end`))
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1305 (42000): PROCEDURE ist175573.overlapping_time_periods_for_Patient_PAN
does not exist
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails (`ist175573`.`Wears`, CONSTRAINT `Wears_ibfk_1` FOREIGN KEY (`start`, `end`
) REFERENCES `Period` (`start`, `end`))
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint f
ails (`ist175573`.`Wears`, CONSTRAINT `Wears_ibfk_1` FOREIGN KEY (`start`, `end`
) REFERENCES `Period` (`start`, `end`))

```

Figura 3: Resultados dos testes do *trigger* de inserção para a tabela *Wears*

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*.

3 SQL queries