# **Projeto de Bases de Dados**, Parte 3


**Grupo nº 38**

**Turno de Segunda feira 12:30 - Lab14**

**Docente: Taras Lykhenko**

| **Aluno** | **Esforço (em horas)** | **Percentagem relativa de contribuição** |
|---|---|---|
| Francisco Nicolau - 86419 | 15 | 33.(3)% |
| Manuel Correia - 86470 | 15 | 33.(3)% |
| Miguel Valério - 86483 | 15 | 33.(3)% |

# Criação da Base de Dados (Schemas)

No inicio do ficheiro schema.sql, é feito uma instrução *drop* por cada uma das tabelas que se vai criar em seguida. Isto serve para que caso haja uma tabela anterior com o mesmo nome esta seja apagada e criada a tabela que se pretende.

```
create table Camara (
    numCamara integer,
    primary key(numCamara) );

create table Local (
    moradaLocal varchar(255),
    primary key (moradaLocal) );

create table EntidadeMeio (
    nomeEntidade varchar(200),
    primary key (nomeEntidade) );

create table Coordenador (
    idCoordenador integer,
    primary key (idCoordenador) );

create table Video (
    dataHoraInicio timestamp,
    dataHoraFim timestamp not null,
    numCamara integer,
    primary key(dataHoraInicio, numCamara),
    foreign key(numCamara) references Camara(numCamara) on delete cascade on update cascade );

create table SegmentoVideo (
    numSegmento integer,
    duracao interval not null,
    dataHoraInicio timestamp,
    numCamara integer,
    primary key (numSegmento, dataHoraInicio, numCamara),
    foreign key (dataHoraInicio, numCamara) references Video(dataHoraInicio, numCamara) on delete
        cascade on update cascade );

create table ProcessoSocorro (
    numProcessoSocorro integer,
    primary key (numProcessoSocorro) );
```

```
create table Meio (
    numMeio integer,
    nomeMeio varchar(30) not null,
    nomeEntidade varchar(200),
    primary key (numMeio, nomeEntidade),
    foreign key (nomeEntidade) references EntidadeMeio(nomeEntidade) on delete cascade on update
        cascade );

create table MeioCombate (
    numMeio integer,
    nomeEntidade varchar(200),
    primary key (numMeio, nomeEntidade),
    foreign key (numMeio, nomeEntidade) references Meio(numMeio, nomeEntidade) on delete cascade on
        update cascade );

create table MeioApoio (
    numMeio integer,
    nomeEntidade varchar(200),
    primary key (numMeio, nomeEntidade),
    foreign key (numMeio, nomeEntidade) references Meio(numMeio, nomeEntidade) on delete cascade on
        update cascade );

create table MeioSocorro (
    numMeio integer,
    nomeEntidade varchar(200),
    primary key (numMeio, nomeEntidade),
    foreign key (numMeio, nomeEntidade) references Meio(numMeio, nomeEntidade) on delete cascade on
        update cascade );

create table Vigia (
    moradaLocal varchar(255),
    numCamara integer,
    primary key (moradaLocal, numCamara),
    foreign key (moradaLocal) references Local(moradaLocal) on delete cascade on update cascade,
    foreign key (numCamara) references Camara(numCamara) on delete cascade on update cascade );

create table EventoEmergencia (
    numTelefone varchar(9),
    instanteChamada timestamp,
    nomePessoa varchar(80) not null ,
    moradaLocal varchar(255) not null ,
    numProcessoSocorro integer,
    primary key (numTelefone, instanteChamada),
    foreign key (moradaLocal) references Local(moradaLocal) on delete cascade on update cascade,
    foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro) on delete cascade
        on update cascade,
    unique (numTelefone, nomePessoa) );
```

```
create table Transporta (
    numMeio integer,
    nomeEntidade varchar(200),
    numVitimas integer not null,
    numProcessoSocorro integer,
    primary key (numMeio, nomeEntidade, numProcessoSocorro),
    foreign key (numMeio, nomeEntidade) references MeioSocorro(numMeio, nomeEntidade) on delete
        cascade on update cascade,
    foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro) on delete cascade
        on update cascade );

create table Alocado (
    numMeio integer,
    nomeEntidade varchar(200),
    numHoras interval not null,
    numProcessoSocorro integer,
    primary key (numMeio, nomeEntidade, numProcessoSocorro),
    foreign key (numMeio, nomeEntidade) references MeioApoio(numMeio, nomeEntidade) on delete
        cascade on update cascade,
    foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro) on delete cascade
        on update cascade );

create table Acciona (
    numMeio integer,
    nomeEntidade varchar(200),
    numProcessoSocorro integer,
    primary key (numMeio, nomeEntidade, numProcessoSocorro),
    foreign key (numMeio, nomeEntidade) references Meio(numMeio, nomeEntidade) on delete cascade on
        update cascade,
    foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro) on delete cascade
        on update cascade );

create table Audita (
    idCoordenador integer,
    numMeio integer,
    nomeEntidade varchar(200),
    numProcessoSocorro integer,
    datahoraInicio timestamp not null,
    datahoraFim timestamp not null,
    dataAuditoria date not null,
    texto text,
    primary key (idCoordenador, numMeio, nomeEntidade, numProcessoSocorro),
    foreign key (numMeio, nomeEntidade, numProcessoSocorro) references
        Acciona(numMeio, nomeEntidade, numProcessoSocorro) on delete cascade on update cascade,
    foreign key (idCoordenador) references Coordenador(idCoordenador) on delete cascade on update
        cascade,
    check (datahoraInicio < datahoraFim),
    check (dataAuditoria <= (current_date)) );
```

```
create table Solicita (
    idCoordenador integer,
    dataHoraInicioVideo timestamp,
    numCamara integer,
    dataHoraInicio timestamp not null,
    dataHoraFim timestamp not null,
    primary key (idCoordenador, dataHoraInicioVideo, numCamara),
    foreign key (idCoordenador) references Coordenador(idCoordenador) on delete cascade on update
        cascade,
    foreign key (dataHoraInicioVideo, numCamara) references Video(dataHoraInicio, numCamara) on delete
        cascade on update cascade );
```

No final foi ainda definido um *trigger* (e o *drop* de eventuais *triggers* com o mesmo nome) que verifica a restrição de integridade seguinte: Todos os processos de socorro têm de estar associados a um ou mais eventos de emergência.

```
create or replace function check_ProcSOS() returns trigger as $body$
begin
    if not exists (
        select * from EventoEmergencia EE where EE.numProcessoSocorro = new.numProcessoSocorro )
    then
        raise exception 'Processo de Socorro % nao associado a nenhum Evento de Emergencia.',
            new.numProcessoSocorro
        using hint = 'Verifique o numero do Processo de Socorro';
    end if;
    return null;
end;
$body$ language plpgsql;

create constraint trigger check_ProcSOS_trigger after insert or update on ProcessoSocorro deferrable
initially deferred for each row execute procedure check_ProcSOS();
```

## Queries

```
1) with TmeiosAccionados as (
        select numProcessoSocorro, count(*) as numMeiosAccionados from Acciona
            group by numProcessoSocorro )
    select numProcessoSocorro, numMeiosAccionados
    from TmeiosAccionados
        natural join (
            select max(numMeiosAccionados) as numMeiosAccionados from TmeiosAccionados
        ) TmaxMeios;

2) with TnumProcessosEntidade as (
        select nomeEntidade, count(*) as numProcessos from Acciona natural join EventoEmergencia
        where instanteChamada >= timestamp '2018-06-21 00:00:00' and
            instanteChamada <= timestamp '2018-09-23 23:59:59' group by nomeEntidade )
    select nomeEntidade, numProcessos from TnumProcessosEntidade
        natural join ( select max(numProcessos) as numProcessos from TnumProcessosEntidade
        ) TentidadeMax;
```

**3)**
```
select numProcessoSocorro from EventoEmergencia natural join Acciona
where extract(year from instanteChamada) = 2018 and moradaLocal = 'Oliveira do Hospital'
except
select numProcessoSocorro from Audita;
```

**4)**
```
select count(*) as numSegmentos from SegmentoVideo natural join Vigia
where duracao > interval '60 secs' and moradaLocal = 'Monchique' and
    extract(year from dataHoraInicio) = 2018 and extract(month from dataHoraInicio) = 8;
```

**5)**
```
select * from MeioCombate
except
select numMeio, nomeEntidade from MeioApoio natural join Acciona;
```

**6)**
```
select distinct nomeEntidade from MeioCombate entComb
where not exists (
select numProcessoSocorro from Acciona
except
select numProcessoSocorro from (Acciona natural join MeioCombate) accComb
where entComb.nomeEntidade = accComb.nomeEntidade);
```

## Aplicação Web

A aplicação Web que desenvolvemos tem no seu centro um ficheiro HTML (*index.html*) que estrutura a página e, com base na opção escolhida, referência o ficheiro PHP correspondente à ação a realizar. Estes estão todos dependentes do ficheiro *login.php* que, como o nome indica, realiza o login que permite o acesso à base de dados através da criação do PDO (*PHP Data Object*) *$db.* Isto é realizado ao incluir este nos restantes PHPs por meio da instrução *include*.

Posto isto, as opções incluídas são adicionar, associar, apagar, editar e listar, cada uma com o seu ficheiro PHP. Nestes, por meio de *switch statements* realiza-se tanto um comando echo de modo a criar os elementos HTML para os formulários apropriados à realização das operações e, em caso de a ação ser submetida, atua em concordância com esta sobre o PDO $db por meio de transações. A exceção a este padrão é *list.php* que ao meramente listar os conteúdos de uma tabela não aguarda a confirmação da ação por parte do utilizador na página do ficheiro PHP. Todas estas operações são realizadas dentro de blocos *try* no sentido de 'apanhar' excepções que possam ocorrer que são depois passadas ao utilizador por meio de um comando *echo*.

Por fim, é de notar que as diversas operações (ficheiros PHP) são independentes umas das outras, estando no entanto todas dependentes do *login.php*. Afinal, se não estiver apropriadamente formulado ou ocorrer algum erro, os restantes não conseguem operar.