Instituto Superior Técnico

Projeto de Bases de Dados

Parte 3

Grupo 40:

- Artur Fortunato, nº 86388 (33% de contribuição, 10 horas de esforço)
- Jorge Pacheco, nº 86457 (33% de contribuição, 10 horas de esforço)
- Miguel Rocha, nº 86482 (33% de contribuição, 10 horas de esforço)

Turno: BD22517957L03 (Lab 10)

Docente de Laboratório: André Vasconcelos

Criação e População da Base de Dados

Tendo em conta o número elevado de registos (comando "insert" em SQL) requeridos, criámos um script em Python3 que criou e preencheu o ficheiro **"populate.sql"**.

Cada função do script foi otimizada de acordo com o ficheiro **"schema.sql"**, onde está especificado a criação das múltiplas tabelas referentes à base de dados que representa o esquema racional do Anexo A em linguagem PostgreSQL. Os "drop table" iniciais servem para limpar os dados da base de dados antes de ser populada outra vez usando o **"populate.sql"**.

```
drop table if exists Solicita cascade;
drop table if exists SegmentoVideo cascade;
drop table if exists Video cascade;
drop table if exists Vigia cascade;
drop table if exists Camara cascade;
drop table if exists EventoEmergencia cascade;
drop table if exists Locais cascade;
drop table if exists Audita cascade;
drop table if exists Transporta cascade;
drop table if exists MeioCombate cascade;
drop table if exists Alocado cascade;
drop table if exists MeioSocorro cascade;
drop table if exists Aciona cascade;
drop table if exists MeioApoio cascade;
drop table if exists ProcessoSocorro cascade;
drop table if exists Coordenador cascade;
drop table if exists Meio cascade;
drop table if exists EntidadeMeio cascade;
```

- A entidade **"Camara"** tem uma chave primária que corresponde a um inteiro com o nome de numCamara.

```
create table Camara
   (numCamara int not null unique,
   constraint pk_camara primary key(numCamara));
```

- A entidade **"Video"** é uma entidade fraca, cuja chave primária corresponde a um *timestamp* (com o formato YYYY-MM-DD HH:MM:SS) com o nome dataHoraInicio, e tem uma *foreign key* que corresponde ao numCamara que o vídeo está associado. A dataHoraFim tem de ser superior a dataHoraInicio.

```
create table Video
   (dataHoraInicio timestamp not null,
   dataHoraFim timestamp not null,
   numCamara int not null ,
   constraint pk_video primary key (dataHoraInicio),
   constraint fk_camara foreign key (numCamara) references Video(numCamara),
   constraint chk_video_time check (dataHoraFim > dataHoraInicio));
```

- A entidade **"SegmentoVideo"** é uma entidade fraca, cuja chave primária corresponde a uma chave composta entre o numCamara, o numSegmento e a dataHoraInicio. O numCamara é uma *foreign key* que provém da Camara a que o Video está associado. A dataHoraInicio é uma *foreign key* que provém do Video.

```
create table SegmentoVideo
   (numSegmento int not null ,
   duracao int not null,
   dataHoraInicio timestamp not null,
   numCamara int not null ,
   constraint pk_segment primary key (numCamara, numSegmento, dataHoraInicio),
   constraint fk_camara foreign key (numCamara) references Video(numCamara),
   constraint fk_dataHoraInicio foreign key (dataHoraInicio) references Video(dataHoraInicio));
```

- A entidade **"Locais"** tem uma chave primária que corresponde a um *string* de tamanho variável que tem o nome de moradaLocal.

```
create table Locais
        (moradaLocal varchar(255) not null unique,
        constraint pk_local primary key (moradaLocal));
```

- A entidade **"Vigia"** tem duas *foreign key*: moradaLocal que provém de Locais e numCamara que provém de Camara.

```
create table Vigia
        (moradaLocal varchar(255) not null,
        numCamara int not null unique,
        constraint fk_local foreign key (moradaLocal) references Locais(moradaLocal),
        constraint fk_camara foreign key (numCamara) references Camara(numCamara));
```

- A tabela **"ProcessoSocorro"** tem uma chave primária que corresponde a um inteiro e tem o nome de numProcessoSocorro.

```
create table ProcessoSocorro
        (numProcessoSocorro int not null unique,
        constraint pk_numProcessoSocorro primary key (numProcessoSocorro));
```

- A entidade **"EventoEmergencia"** tem uma chave primária que corresponde a uma chave composta entre o numTelefone, que é uma *string* de tamanho fixo 9, e o nomePessoa, que corresponde a uma *string* de tamanho variável. A moradaLocal é uma *foreign key* que provém de Locais e o numProcessoSocorro é uma foreign key que provém de ProcessoSocorro.

```
create table EventoEmergencia
        (numTelefone char(9) not null,
        instanteChamada timestamp not null,
        nomePessoa varchar(30) not null,
        moradaLocal varchar(255) not null,
        numProcessoSocorro int,
        constraint pk_witness primary key(numTelefone, nomePessoa),
        constraint fk_local foreign key (moradaLocal) references Locais(moradaLocal),
        constraint fk_numProcessoSocorro foreign key (numProcessoSocorro) references
ProcessoSocorro(numProcessoSocorro));
```

- A entidade **"EntidadeMeio"** tem uma chave primária que é uma *string* de tamanho varíavel, que tem nome de nomeEntidade.

```
create table EntidadeMeio
        (nomeEntidade varchar(20) not null unique,
        constraint pk_nomeEntidade primary key (nomeEntidade));
```

- A entidade **"Meio"** tem uma chave primária que corresponde a uma chave composta entre o numMeio, que é um inteiro, e o nomeEntidade, que é uma *foreign key* que provém de EntidadeMeio.

```
create table Meio
        (numMeio int not null,
        nomeMeio varchar(30) not null,
        nomeEntidade varchar(20) not null,
        constraint pk_meio primary key (numMeio, nomeEntidade),
        constraint fk_nomeEntidade foreign key (nomeEntidade) references EntidadeMeio(nomeEntidade));
```

- A entidade **"MeioCombate"** tem uma chave primária, que corresponde a uma *foreign key* composta entre o numMeio e o nomeEntidade que provém de Meio.

```
create table MeioCombate
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        constraint pk_meioCombate primary key (numMeio, nomeEntidade),
        constraint fk_meio foreign key (numMeio, nomeEntidade) references Meio(numMeio,
nomeEntidade));
```

- A entidade **"MeioApoio"** tem uma chave primária, que corresponde a uma *foreign key* composta entre o numMeio e o nomeEntidade que provém de Meio.

```
create table MeioApoio
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        constraint pk_meioApoio primary key (numMeio, nomeEntidade),
        constraint fk_meio foreign key (numMeio, nomeEntidade) references Meio(numMeio,
nomeEntidade));
```

- A entidade **"MeioSocorro"** tem uma chave primária, que corresponde a uma *foreign key* composta entre o numMeio e o nomeEntidade que provém de Meio.

```
create table MeioSocorro
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        constraint fk_meioSocorro primary key (numMeio, nomeEntidade),
        constraint fk_meio foreign key (numMeio, nomeEntidade) references Meio(numMeio,
nomeEntidade));
```

- A entidade **"Transporta"** tem uma *foreign key* composta entre o numMeio e o nomeEntidade que provém de MeioSocorro e a *foreign key* numProcessoSocorro que provém de ProcessoSocorro.

```
create table Transporta
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        numVitimas int not null ,
        numProcessoSocorro int not null,
        constraint fk_meio foreign key (numMeio, nomeEntidade) references MeioSocorro(numMeio,
nomeEntidade),
        constraint numProcessoSocorro foreign key (numProcessoSocorro) references
ProcessoSocorro(numProcessoSocorro));
```

- A entidade **"Alocado"** tem uma *foreign key* composta entre o numMeio e o nomeEntidade que provém de MeioApoio e a *foreign key* numProcessoSocorro que provém de ProcessoSocorro.

```
create table Alocado
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        numHoras int not null ,
        numProcessoSocorro int,
        constraint fk_meio foreign key (numMeio, nomeEntidade) references MeioApoio(numMeio, nomeEntidade),
        constraint numProcessoSocorro foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro));
```

- A entidade **"Aciona"** tem uma chave primária composta entre o numMeio, nomeEntidade – que são uma *foreign key* que provém de Meio – e numProcessoSocorro, que é uma *foreign key* de ProcessSocorro.

```
create table Aciona
        (numMeio int not null,
        nomeEntidade varchar(20) not null,
        numProcessoSocorro int not null,
        constraint pk_aciona primary key (numMeio, nomeEntidade, numProcessoSocorro),
        constraint fk_meio foreign key (numMeio, nomeEntidade) references Meio(numMeio, nomeEntidade),
        constraint numProcessoSocorro foreign key (numProcessoSocorro) references ProcessoSocorro(numProcessoSocorro));
```

- A entidade **"Coordenador"** tem uma chave primária que corresponde a um inteiro, que tem nome de idCoordenador.

```
create table Coordenador
        (idCoordenador int not null unique ,
        constraint idCoordenador primary key (idCoordenador));
```

- A entidade **"Audita"** tem uma *foreign key* composta entre numMeio, nomeEntidade e numProcessoSocorro que provêm de Aciona e uma *foreign key* idCoordenador que provém de Coordenador. A datahoraInicio tem de ser inferior à datahoraFim (do tipo YYYY-MM-DD HH:MM:SS), e dataAuditoria tem de ser inferior ou igual à atual data (do tipo YYYY-MM-DD).

```
create table Audita
        (idCoordenador int not null,
        numMeio int not null,
        nomeEntidade varchar(20) not null,
        numProcessoSocorro int,
        datahoraInicio timestamp not null,
        datahoraFim timestamp not null,
        dataAuditoria date not null,
        texto text not null,
        constraint numMeio foreign key (numMeio, nomeEntidade, numProcessoSocorro) references Aciona(numMeio, nomeEntidade, numProcessoSocorro),
        constraint idCoordenador foreign key (idCoordenador) references Coordenador(idCoordenador),
        constraint chk_audit_time check (datahoraInicio < datahoraFim),
        constraint chk_audit_date check (dataAuditoria <= current_date));
```

- A entidade **"Solicita"** tem uma *foreign key* idCoordenador que provém do Coordenador, uma *foreign key* dataHoraInicioVideo que provém de Video e uma *foreign key* numCamara que vem da Camara.

```
create table Solicita
    (idCoordenador int not null,
    dataHoraInicioVideo timestamp not null,
    numCamara int not null,
    dataHoraInicio timestamp not null,
    dataHoraFim timestamp not null,
    constraint idCoordenador foreign key (idCoordenador) references Coordenador(idCoordenador),
    constraint dataHoraInicioVideo foreign key (dataHoraInicioVideo) references Video(dataHoraInicio),
    constraint numCamara foreign key (numCamara) references Camara(numCamara));
```

Consultas SQL

As seguintes consultas SQL pretendem responder às consultas do enunciado:

1. SELECT numprocessosocorro FROM aciona
   GROUP BY numprocessosocorro
   HAVING COUNT(nummeio) >= ALL(SELECT COUNT(nummeio) FROM aciona
   GROUP BY numprocessosocorro);

2. SELECT nomeentidade FROM eventoemergencia NATURAL JOIN aciona
   WHERE instantechamada <= '2018-09-23 23:59:59'
   AND instantechamada >= '2018-06-21 00:00:00'
   GROUP BY nomeentidade HAVING COUNT(numprocessosocorro) >= ALL
   (SELECT COUNT(numprocessosocorro) FROM eventoemergencia
   NATURAL JOIN aciona WHERE instantechamada <= '2018-09-23 23:59:59 AND instantechamada >= '2018-06-21 00:00:00' GROUP BY nomeentidade);

3. SELECT numprocessosocorro FROM eventoemergencia NATURAL JOIN aciona
   WHERE moradalocal = 'Oliveira do Hospital' AND instantechamada >= '2018-01-01 00:00:00' AND instantechamada <= '2018-12-31 23:59:00'
   AND numprocessosocorro NOT IN (SELECT numprocessosocorro FROM audita);

4. SELECT COUNT(numsegmento)
   FROM segmentovideo NATURAL JOIN video NATURAL JOIN vigia
   WHERE duração > 60 AND moradalocal = 'Monchique'
   AND datahorafim <= '2018-08-31 23:59' AND datahorainicio >= '2018-08-01 00:00';

5. SELECT nummeio, nomeentidade FROM meiocombate
   EXCEPT (SELECT nummeio, nomeentidade FROM alocado
   GROUP BY nummeio, nomeentidade);

6. SELECT nomeentidade FROM meiocombate NATURAL JOIN aciona
   WHERE NOT EXISTS (SELECT numprocessosocorro FROM processosocorro
   EXCEPT SELECT numprocessosocorro FROM aciona);

Aplicação desenvolvida em PHP

A aplicação PHP está estruturada da seguinte forma:

- Um ficheiro index.php que serve como interface para o programa, onde é possível executar todas as ações pedidas no enunciado. Permite também ver todas as tabelas editáveis (Locais, Eventos de emergência, Processos de socorro, Meios, Entidades, Meio de combate, Meio de socorro e Meio de apoio), de forma a facilitar a visualização das alterações à base de dados.
- Vários ficheiros PHP com scripts para adicionar elementos às tabelas.
- 2 ficheiros PHP para associar processos de socorro a meios e processos de socorro a eventos de emergência.
- Um ficheiro config.php para configurar a ligação à base de dados.
- Um ficheiro deleteFromTable.php para apagar elementos das tabelas.
- Dois ficheiros para listar meios acionados num certo local e processos de socorro associados a um local.
- Dois ficheiros para atualizar o conteúdo das tabelas meio de combate, meio de socorro e meio de apoio.

Através de forms e links inseridos no ficheiros index.php, este ficheiro permite realizar ações como adicionar, editar ou remover entradas das tabelas, executando outros *scripts*. Os botões de apagar e editar estão ao lado de cada entrada na tabela e o campos de input para adicionar elementos a cada tabela estão no topo das mesmas.

No topo de todas as tabelas estão quatro secções:

- A primeira permite listar os meios de socorro acionados em processos de socorro originados num determinado local (inserido no campo de input respetivo).
- A segunda permite listar os meios acionados num determinado processo (inserido no campo de input respetivo).
- A terceira permite associar um processo de socorro a meios.
- A última permite associar processos de socorro a eventos de emergência.

Com esta aplicação dão-se resposta a todas operações pedidas no enunciado, adicionando a funcionalidade de visualizar as tabelas que são editáveis diretamente.