Projeto de Bases de Dados, Parte 3

Nome	Número	Horas	Contribuição
André Guerra	86382	16	33,3%
André Ribeiro	86384	16	33,3%
Pedro Bernardo	86500	16	33,3%

Grupo 67

Turno: 6ª feira, 12h30
Docente: Taras Lykhenko
2018/2019

1 – Criação da Base de Dados

```
create table camara(
   n_camara    integer not null unique,
   constraint pk_Camara primary
key(n_camara)
);

create table video(
   n_camara          integer not null,
   data_hora_inicio      timestamp not null,
   data_hora_fim                  timestamp not
null,
   check (data_hora_fim > data_hora_inicio),
   constraint pk_video primary key(n_camara,
data_hora_inicio),
   constraint fk_video_n_camara foreign
key(n_camara) references camara
);

create table segmento_video(
   n_segmento        integer   not null,
   n_camara          integer          not null,
   data_hora_inicio      timestamp      not null,
   duracao           interval hour to second not
null,
   constraint pk_segmento_video primary
key(n_segmento, n_camara, data_hora_inicio),
   constraint fk_segmento_video_video foreign
key(n_camara, data_hora_inicio) references
video

);

create table localizacao(
   morada_local       varchar(255) not null
unique,
   constraint pk_localizacao primary
key(morada_local)
);

create table vigia(
   morada_local       varchar(255) not null ,
   n_camara          integer          not null,
   constraint pk_vigia primary
key(morada_local, n_camara),
   constraint fk_vigia_n_camara foreign
key(n_camara) references camara,
   constraint fk_vigia_morada_local foreign
key(morada_local) references localizacao
);

create table processo_socorro(
   n_proc_socorro        integer not null unique,
   constraint pk_process_socorro primary
key(n_proc_socorro)
);

create table evento_emergencia(
   n_telefone           varchar(15) not null,
   inst_chamada          timestamp not null,
   nome_pessoa           varchar(80) not null,
   morada_local          varchar(80) not null ,
   n_proc_socorro        integer not null,
   constraint pk_evento primary
key(n_telefone, inst_chamada),
   constraint fk_evento_morada_local foreign
key(morada_local) references localizacao,
   constraint fk_evento_n_proc_socorro
foreign key(n_proc_socorro) references
processo_socorro
);


create table entidade_meio(
   nome_entidade   varchar(200) not null
unique,
   constraint pk_entidade_meio primary key
(nome_entidade)
);

create table meio(
   n_meio     integer not null,
   nome_meio   varchar(80) not null,
   nome_entidade   varchar(200) not null,
   constraint pk_meio primary key(n_meio,
nome_entidade),
   constraint fk_meio_entidade foreign key
(nome_entidade) references entidade_meio
);

create table meio_combate(
   n_meio     integer not null,
   nome_entidade   varchar(200) not null,
   constraint pk_meio_combate primary
key(n_meio, nome_entidade),
   constraint fk_meio_combate_entidade
foreign key (n_meio, nome_entidade)
references meio
);

create table meio_apoio(
   n_meio     integer not null,
   nome_entidade   varchar(200) not null,
   constraint pk_meio_apoio primary
key(n_meio, nome_entidade),
```

```
   constraint fk_meio_apoio_entidade foreign
key (n_meio, nome_entidade) references meio
);

create table meio_socorro(
   n_meio      integer not null,
   nome_entidade   varchar(200) not null,
   constraint pk_meio_socorro primary
key(n_meio, nome_entidade),
   constraint fk_meio_socorro_entidade
foreign key (n_meio, nome_entidade)
references meio
);

create table transporta(
   n_meio      integer not null,
   nome_entidade   varchar(200) not null,
   n_proc_socorro  integer not null,
   n_vitimas       integer not null,
   constraint pk_transporta primary
key(n_meio, nome_entidade, n_proc_socorro),
   constraint fk_transporta_proc foreign
key(n_proc_socorro) references
processo_socorro,
   constraint fk_transporta_entidade foreign
key (n_meio, nome_entidade) references
meio_socorro
);

create table alocado(
   n_meio      integer not null,
   nome_entidade   varchar(200) not null,
   n_proc_socorro  integer not null,
   n_horas         integer not null,
   constraint pk_alocado primary key(n_meio,
nome_entidade, n_proc_socorro),
   constraint fk_alocado_proc foreign
key(n_proc_socorro) references
processo_socorro,
   constraint fk_alocado_entidade foreign key
(n_meio, nome_entidade) references
meio_apoio
);


create table acciona(
   n_meio      integer not null,
   nome_entidade   varchar(200) not null,
   n_proc_socorro  integer not null,
   constraint pk_acciona primary key(n_meio,
nome_entidade, n_proc_socorro),
   constraint fk_acciona_proc foreign
key(n_proc_socorro) references
processo_socorro,
   constraint fk_acciona_entidade foreign key
(n_meio, nome_entidade) references meio
);


create table coordenador(
   id      integer not null unique,
   constraint pk_coordenador primary key (id)
);

create table audita (
   id_coord      integer not null,
   n_meio        integer not null,
   nome_entidade varchar(200) not null,
   n_proc_socorro  integer not null,
   data_hora_inicio timestamp not null,
   data_hora_fim   timestamp not null,
   data_auditoria  date not null,
   texto           text not null,
   check(data_hora_inicio < data_hora_fim),
   check(data_auditoria <= current_date),
   constraint pk_audita primary key (id_coord,
n_meio, nome_entidade, n_proc_socorro),
   constraint fk_audita_id foreign key
(id_coord) references coordenador(id),
   constraint fk_audita_acciona foreign key
(n_meio, nome_entidade, n_proc_socorro)
references acciona
);

create table solicita(
   id_coord      integer not null,
   data_hora_inicio_video timestamp not null,
   n_camara        integer not null,
   data_hora_inicio timestamp not null,
   data_hora_fim   timestamp not null,
   check(data_hora_inicio < data_hora_fim),
   constraint pk_solicita primary key (id_coord,
data_hora_inicio_video, n_camara),
   constraint fk_solicita_id foreign key
(id_coord) references coordenador(id),
   constraint fk_solicita_video foreign key
(data_hora_inicio_video, n_camara) references
video(data_hora_inicio, n_camara)
);

CREATE OR REPLACE FUNCTION delete_proc()
RETURNS TRIGGER AS $BODY$
BEGIN
   DELETE FROM evento_emergencia WHERE
evento_emergencia.n_proc_socorro =
OLD.n_proc_socorro;
   RETURN OLD;
END;
$BODY$ language plpgsql;
```

```
CREATE TRIGGER delete_proc_trigger
   BEFORE DELETE
   ON processo_socorro
   FOR EACH ROW
   EXECUTE PROCEDURE delete_proc();


CREATE OR REPLACE FUNCTION delete_m()
RETURNS TRIGGER AS $BODY$
BEGIN
   DELETE FROM meio_apoio WHERE
meio_apoio.n_meio = OLD.n_meio AND
meio_apoio.nome_entidade=OLD.nome_entid
ade;
   DELETE FROM meio_combate WHERE
meio_combate.n_meio = OLD.n_meio AND
meio_combate.nome_entidade=OLD.nome_en
tidade;
   DELETE FROM meio_socorro WHERE
meio_socorro.n_meio = OLD.n_meio AND
meio_socorro.nome_entidade=OLD.nome_enti
dade;
   RETURN OLD;
END;
$BODY$ language plpgsql;

CREATE TRIGGER delete_meio
   BEFORE DELETE
   ON meio
   FOR EACH ROW
   EXECUTE PROCEDURE delete_m();


CREATE OR REPLACE FUNCTION delete_e()
RETURNS TRIGGER AS $BODY$
BEGIN
   DELETE FROM meio WHERE
meio.nome_entidade = OLD.nome_entidade;
   RETURN OLD;
END;
$BODY$ language plpgsql;

CREATE TRIGGER delete_entidade
   BEFORE DELETE
   ON entidade_meio
   FOR EACH ROW
   EXECUTE PROCEDURE delete_e();


CREATE OR REPLACE FUNCTION
insert_evento()
RETURNS TRIGGER AS $BODY$
BEGIN
   IF NOT EXISTS (SELECT * FROM
processo_socorro WHERE
processo_socorro.n_proc_socorro=NEW.n_pro
c_socorro)
   THEN
      INSERT INTO processo_socorro
VALUES(NEW.n_proc_socorro);
   END IF;

   IF NOT EXISTS (SELECT * FROM localizacao
WHERE
localizacao.morada_local=NEW.morada_local)
   THEN
      INSERT INTO localizacao
VALUES(NEW.morada_local);
   END IF;
   RETURN NEW;
END;
$BODY$ language plpgsql;


CREATE TRIGGER insert_evento_emergencia
BEFORE INSERT
ON evento_emergencia
FOR EACH ROW
EXECUTE PROCEDURE insert_evento();
```

2 – SQL

```
1)
SELECT n_proc_socorro
FROM acciona group by n_proc_socorro
having Count(*) >= all (
   select Count(*)
   from acciona
   group by n_proc_socorro
);
```

```
2)
SELECT v.nome_entidade
FROM (
      SELECT nome_entidade
      FROM acciona natural join
evento_emergencia
      WHERE inst_chamada BETWEEN '2018-06-21
11:07:00' AND '2018-09-23 01:54:00'
   ) as v group by v.nome_entidade
HAVING Count(*) >= all (
   SELECT Count(*)
   FROM (
      SELECT nome_entidade
      FROM acciona natural join
evento_emergencia
      WHERE inst_chamada BETWEEN '2018-06-21
11:07:00' AND '2018-09-23 01:54:00'
   ) as verao group by verao.nome_entidade
);

3)
SELECT v.n_proc_socorro
FROM ( SELECT distinct n_proc_socorro
      FROM acciona natural join
evento_emergencia
      WHERE inst_chamada BETWEEN '2018-01-01
00:00:00' AND '2018-12-31 23:59:59' AND
morada_local = 'Oliveira do Hospital'
   ) v
WHERE v.n_proc_socorro NOT IN (
   SELECT distinct n_proc_socorro
   FROM audita natural join ( SELECT distinct
n_proc_socorro
      FROM acciona natural join
evento_emergencia
      WHERE inst_chamada BETWEEN '2018-01-01
00:00:00' AND '2018-12-31 23:59:59' AND
morada_local = 'Oliveira do Hospital'
   ) as b
);

4)
SELECT count(*)
FROM segmento_video natural join video natural
join camara natural join vigia
WHERE morada_local = 'Monchique' AND
data_hora_inicio BETWEEN '2018-08-01 00:00:00'
AND '2018-08-31 23:59:59'
   AND duracao > '0 00:01:00';

5)
SELECT n_meio, nome_entidade
FROM meio_combate natural join acciona
WHERE (n_meio, nome_entidade) NOT IN (SELECT
n_meio, nome_entidade FROM meio_apoio
natural join acciona );

6)
SELECT distinct nome_entidade
FROM acciona a
WHERE NOT EXISTS (
   SELECT distinct n_proc_socorro
   FROM acciona
   EXCEPT
   SELECT n_proc_socorro
   FROM acciona b
   WHERE b.nome_entidade = a.nome_entidade
);
```

3 – Arquitetura da aplicação PHP

A aplicação PHP começa com um simples ficheiro html contendo um menu básico com hiperligações para os ficheiros PHP que tratam das páginas base de interação com as entidades de destaque das funcionalidades pretendidas: Processos Socorro, Meios, Locais de incêndio e Eventos de Emergência.
Os ficheiros PHP estão organizados em diretorias, agrupados pelas quatro entidades referidas anteriormente. Para cada uma, é apresentada uma página principal com botões que representam as ações possíveis como “Inserir”, “Remover” e “Editar”.
Estas páginas utilizam HTML forms para receber dados do utilizador e os enviar para os ficheiros PHP que tratam de realizar a opção selecionada pelo utilizador.
Por exemplo, ao adicionar um Meio é pedido ao utilizador que escolha uma das entidades registadas na base de dados e que escreva um nome para o Meio a adicionar. A entidade selecionada e o nome dado são depois enviados para um ficheiro chamado

“insertMeios.php” ao clicar no botão “Submit”. Esta página vai tratar de inserir o Meio, com os dados inseridos pelo utilizador, na base de dados.
Em toda a aplicação são sempre apresentadas ao utilizador escolhes que ele de facto pode fazer. Por exemplo, quando o utilizador pretende remover um meio de socorro, só pode selecionar para remover os meios de socorro que nunca transportaram vítimas, de forma a manter a integridade da base de dados.
Devido à relação de dependência entre Eventos de Emergência e Processos de Socorro, optámos por juntar as funcionalidades de “Inserir” destas duas entidades numa só. Ao adicionar um Evento de Emergência é pedido ao utilizador que introduza também um número identificador de um Processo de Socorro. Caso este Processo de Socorro não exista então também é criado. Esta é então a unica maneira de criar Processos de Socorro, uma vez que um processo de socorro não pode existir sem estar associado a um Evento de Emergência.
Devido também a esta dependência, sempre que o utilizador pretende associar um Processo Socorro a um Evento de Emergência, a aplicação PHP verifica se o processo que consequentemente irá perder a associação ao Evento de Emergência selecionado, passa a ter 0 associações a outros Eventos. Neste caso, não é permitido ao utilizador realizar a associação pretendida.
Sempre que o utilizador pretende remover um Processo de Socorro do sistema, só o pode fazer se o Processo de Socorro nunca tiver accionado meios. Isto poderia acontecer no caso de um falso alarme por exemplo, em que seria criado um Evento de Emergência e consequentemente um Processo de Socorro, mas seria irrelevante mantê-lo na base de dados uma vez que não passou de um falso alarme e não foram mobilizados meios.
Pelo raciocínio análogo, também não é possível remover um Meio se este já tiver sido accionado, nem um Meio de Socorro que tenha transportado vítimas ou um Meio de Apoio que tenha sido alocado.