

Projeto de Bases de Dados

Parte 3

1ºSemestre

Grupo 76

Luisa Santo nº 79758
Luana Marques nº 82364
Maria Lopes nº 86477

Horas de trabalho: 16 horas de trabalho por elemento

SQL

1) Contam-se os processos de socorro conforme o numero de vezes que cada um aparece na tabela acciona. Posteriormente procura-se o máximo, ou seja, o processo de socorro que envolveu maior número de meios distintos. Por fim faz-se a projeção dos processos de socorro cujo número de meios distintos é igual ao máximo;

```
SELECT numProcessoSocorro
FROM ( SELECT numProcessoSocorro, count(DISTINCT numMeio) AS nr_Meios
      FROM acciona GROUP BY numProcessoSocorro
      having(nr_Meios ) > max(nr_Meios ));
```

2) Para cada entidade fornecedora, contam-se todos os processos de socorro aos quais a entidade forneceu meios durante verão de 2018. Encontra-se o máximo de participações entre as entidades fornecedoras durante esse período e por fim projetam-se as que possuem tantas participações quanto o máximo encontrado.

```
SELECT nomeEntidade
FROM (SELECT nomeEntidade, count(DISTINCT numProcessoSocorro)
      AS nr_PS FROM acciona GROUP BY nomeEntidade
      having (nr_PS) > max(nr_PS)) as T
NATURAL JOIN eventoEmergencia
WHERE ( instanteChamada > '2018-06-21 00:00:00' AND instanteChamada < '2018-09-21 23:59:59' ) ;
```

3) Projetam-se todos os processos de socorro que foram acionados em agosto de 2018 em Monchique, mas que não foram sujeitos a auditoria, no fundo faz-se uma subtração da coluna numProcessoSocorro entre a tabela acciona e a tabela audita, consoante as restrições impostas.

```
SELECT numProcessoSocorro
FROM (SELECT numProcessoSocorro, nomeEntidade
      FROM acciona
      WHERE numProcessoSocorro NOT IN (SELECT numProcessoSocorro FROM audita)) AS T
NATURAL JOIN eventoEmergencia
WHERE ( nomeEntidade = 'Oliveira do Hospital'
      AND instanteChamada > '2018-01-01 00:00:00' AND instanteChamada < '2018-12-12 23:59:59' );
```

- 4) Projetam-se todos os segmentos de vídeo que cumprem a duração, período temporal, e local pedidos;

```
SELECT count(numSegmento) FROM segmentoVideo
NATURAL JOIN vigia
WHERE (moradaLocal='Monchique' and
       duracao > '0:01:00' and
       dataHoraInicio > ' 2018-08-01 00:00:00' and
       dataHoraInicio < ' 2018-08-31 23:59:59');
```

- 5) Primeiro faz-se uma seleção de todos os meios de apoio acionados por processos de socorro; Posteriormente projetam-se todos os meios de combate acionados por processos de socorro, mas que não estão presentes na tabela anterior; Por fim, quando obtermos a subtração das tabelas, podemos projetar os meios que cumprem essa restrição.

```
SELECT numMeio
FROM (SELECT numMeio
      FROM (meio NATURAL JOIN acciona)
      WHERE numMeio NOT IN (SELECT numMeio FROM
                           (SELECT numMeio FROM (meio NATURAL JOIN acciona)
                           WHERE ( nomeMeio = 'Apoio') ) ) AS T2
      natural join (meio NATURAL JOIN acciona) where nomeMeio='Combate');
```

- 6) Projetam-se as entidades que forneceram meios de combate a todos os processos de socorro que ativaram meios;

```
SELECT nomeEntidade
FROM processoSocorro NATURAL JOIN acciona
WHERE nomeEntidade IN (SELECT nomeEntidade FROM meioCombate);
```

Triggers

a) RI-1: "Nao podem existir processos de socorro sem estarem associados a um ou mais Eventos de Emergencia"

Para cumprir esta restrição de integridade optamos por implementar um trigger, accionado antes de ser inserido um novo processo socorro na tabela **processosocorro**. Caso este novo processo tenha um **número do processo socorro** que não exista na tabela de **eventos de emergencia**, é chamada uma função de erro que interrompe e impede a inserção do processo de socorro na tabela.

```
-- RI-1: Não podem existir processos de socorro sem estarem associados a um ou mais Eventos de Emergencia
CREATE OR REPLACE FUNCTION nenhum_evento_de_emergencia() RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.numProcessoSocorro NOT IN (
        SELECT numProcessoSocorro FROM eventoEmergencia)) THEN
        RAISE '0 numero de processo % não equivale a nenhum evento de emergencia', NEW.numProcessoSocorro
        USING HINT = 'Ver lista de eventos de emergencia';
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_processosocorro ON processoSocorro;
CREATE TRIGGER update_processosocorro
BEFORE INSERT OR UPDATE ON processoSocorro
FOR EACH ROW
EXECUTE PROCEDURE nenhum_evento_de_emergencia();
```

b) RI-2: "A data e a hora do inicio da auditoria tem que ser inferior à data e hora do fim dessa mesma auditoria e a data da auditoria tem que ser inferior à data actual"

Esta restrição de integridade foi implementada através de um trigger que é acionado antes da inserção de valores na tabela **audita**. Este trigger consiste na chamada de uma função de erro que interrompe e impede a inserção de uma nova entrada nesta tabela quando tentam inserir dados e o timestamp da **data e hora de inicio** não é menor que a o timestamp da **data e hora fim** e, em particular, quando o **dia da auditoria** é mais recente que o dia actual.

```
-- R2: A data e a hora do inicio da auditoria tem que ser inferior 'a data e hora do fim dessa mesma auditoria
CREATE OR REPLACE FUNCTION datas_mal_formatadas() RETURNS TRIGGER AS
$$
BEGIN
    IF (NEW.datahorainicio > NEW.datahorafim) THEN
        RAISE 'data hora inicio não pode ser superior a data hora fim'
        USING HINT = 'hora inicio < hora fim';
    END IF;
    IF (NEW.dataauditoria > NOW()) THEN
        RAISE '% é superior a data de hoje', NEW.dataauditoria
        USING HINT = 'data auditoria inferior a ';
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_auditoria ON audita;
CREATE TRIGGER update_auditoria
BEFORE INSERT OR UPDATE ON audita
```

Desenvolvimento da Aplicação

Para explicar a nossa aplicação iremos apresentar um modelo geral de inserção e remoção de entradas nas tabelas que se aplicam a vários dos pontos pedidos.

A aplicação é acedida através do endereço <http://web.ist.utl.pt/ist179758/bd-ist/web/> que direcciona à página inicial. A imagem à direita é um *screenshot* desta página que possui as funções da nossa aplicação com correspondência aos pontos pedidos no enunciado.

Todo o código relacionado diretamente com a base de dados é executado dentro de um *try* com um *catch* de

Base de Dados

Inicio

[Adicionar ou Remover Locais](#)
[Adicionar ou Remover Eventos de Emergencia](#)
[Adicionar ou Remover Processos de Socorro](#)
[Adicionar ou Remover Meios](#)
[Adicionar ou Remover Entidades](#)
[Adicionar, Editar ou Remover Meios de Combate](#)
[Adicionar, Editar ou Remover Meios de Apoio](#)
[Adicionar, Editar ou Remover Meios de Socorro](#)
[Mostrar Processos de Socorro](#)
[Mostrar Meios](#)
[Associar Processos de Socorro a Meios](#)
[Associar Processos de Socorro a Eventos de Emergencia](#)
[Mostrar Meios de um Processo de Socorro](#)
[Mostrar Meios de um Processo de Socorro de um incendio](#)

exceções no final.

Todos os ficheiros .php que acedem à base de dados iniciam a secção de php com o código na imagem à direita.

```
$host = "db.ist.utl.pt";
$user = "ist179758";
$password = "bd2018";
$dbname = $user;
$db = new PDO("pgsql:host=$host;dbname=$dbname", $user, $password);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Inserção de dados nas tabelas

```
try
{
    $host = "db.ist.utl.pt";
    $user = "ist179758";
    $password = "bd2018";
    $dbname = $user;
    $db = new PDO("pgsql:host=$host;dbname=$dbname", $user, $password);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $db->query("start transaction;");
    $sql = "INSERT INTO meioapoio (nummeio, nomeentidade) VALUES (?,?)";
    $stmt = $db->prepare($sql);
    $stmt->execute(array($nummeio, $nomeentidade));
    $db->query("commit;");

    $db = null;
    echo("<p>Meio adicionada com sucesso!</p>");
}
}
```

Este excerto de código corresponde à inserção de um novo meio de apoio na tabela **meiodeapoio**.

Começamos por escrever um “sql statement template” para inserir um meio de apoio, de seguida este é enviado para a base de dados mas os argumentos não são especificados (“?”). Depois a base de dados compila, faz uma optimização ao “statement” e guarda o resultado sem o executar. Ao fazermos “\$stmt->execute” a aplicação faz uma conexão entre os valores e os parâmetros não especificados anteriormente.

Remoção de entradas das tabelas

Este excerto de código corresponde à remoção de um meio de apoio da tabela **meiodeapoio**.

Aqui também são feitos “prepare statements” como na inserção nas tabelas, a única diferença é que a query em vez de inserir vai remover da tabela em questão.

```
$nummeio = $_REQUEST['nummeio'];
$nomeentidade = $_REQUEST['nomeentidade'];
try
{
    $host = "db.ist.utl.pt";
    $user = "ist179758";
    $password = "bd2018";
    $dbname = $user;
    $db = new PDO("pgsql:host=$host;dbname=$dbname", $user, $password);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $db->query("start transaction;");
    $sql = "DELETE FROM meioapoio WHERE nummeio=? AND nomeentidade=?";
    $stmt = $db->prepare($sql);
    $stmt->execute(array($nummeio, $nomeentidade));
    $db->query("commit;");

    $db = null;
    echo("<p>Meio removido com sucesso!</p>");
}
```

Formato Geral da Aplicação

A imagem em baixo mostra os menus dos meios. A opção Adicionar Meio direcciona a um menu onde o utilizador escolhe nome da entidade onde o quer inserir (imagem em cima à direita).

Meios
[Adicionar Meios](#)

Numero do Meio	Nome do Meio	Nome da Entidade	
1	Socorro	exercito_de_cirek	Remover Meios
2	Socorro	policia_de_jaboh	Remover Meios
3	Socorro	policia_de_sufip	Remover Meios
4	Socorro	exercito_de_tigig	Remover Meios
5	Combate	policia_de_nezon	Remover Meios
6	Socorro	forca aerea_de_zecil	Remover Meios
7	Apoio	policia_de_hojor	Remover Meios
8	Socorro	policia_de_cegiv	Remover Meios
9	Apoio	bombeiros_de_wosiz	Remover Meios
10	Socorro	municipio_de_nopuk	Remover Meios

Nome da Entidade

exercito_de_cirek	Escolher Nome da Entidade
policia_de_jaboh	Escolher Nome da Entidade
policia_de_sufip	Escolher Nome da Entidade
exercito_de_tigig	Escolher Nome da Entidade
policia_de_nezon	Escolher Nome da Entidade
forca aerea_de_zecil	Escolher Nome da Entidade
policia_de_hojor	Escolher Nome da Entidade
policia_de_cegiv	Escolher Nome da Entidade
bombeiros_de_wosiz	Escolher Nome da Entidade

Criar Meios
Numero do Meio:
Nome do Meio:

Após ser escolhido o nome da entidade, é pedido ao utilizador para preencher os restantes atributos de Meios (imagem à direita).

Schema.sql e Populate.sql

Em relação ao ficheiro schema.sql, acrescentamos em todas as tabelas nas foreign keys “ON UPDATE CASCADE”, o que significa que caso estas chaves sejam alteradas a alteração propaga-se para as tabelas que utilizam estas chaves.

Relativamente ao ficheiro populate.sql utilizamos um script criado por nós com o nome de **scriptpopulate.py**.

Script usado para preencher o ficheiro sql populate

O ficheiro principal pode ser visto em baixo. Usamos uma lista que tinha todas as tabelas a ser preenchidas, e ao verificar a respectiva tabela, iteravamos a mesma no ciclo de 100, para conseguirmos gerar 100 elementos.

```
def main():
    frmt = '%H:%M:%S %Y-%m-%d'
    times = randomtimes("13:30:00 2018-01-20", "04:50:34 2018-01-30", frmt, 101)
    numbers = randomphones(9, 1000)
    names = generatelistname(1000)
    entities = generatelistentity(1000)
    entity_names = [meios[randint(0, 2)] for _ in range(100)]
    f = open("populate.sql", "w+")
    for i in tables:
        for j in range(100):
            if i == "camara": populatecamara(f, i, j)
            if i == "video": populatevideo(f, i, j, times)
            if i == "segmentoVideo": populatesegmentovideo(f, i, j, times)
            if i == "local": populatelocal(f, i, j)
            if i == "vigia": populatevigia(f, i, j)
            if i == "eventoEmergencia": populateeventoemergencia(f, i, j, numbers, names, times)
            if i == "processoSocorro": populateprocessosocorro(f, i, j)
            if i == "entidadeMeio": populateentidademeio(f, i, j, entities)
            if i == "meio": populatemeio(f, i, j, entities, entity_names)
            if i == "meioCombate": populatemeiocombate(f, i, j, entities, entity_names)
            if i == "meioApoio": populatemeioapoio(f, i, j, entities, entity_names)
            if i == "meioSocorro": populatemeiosocorro(f, i, j, entities, entity_names)
            if i == "transporta": populatetransporta(f, i, j, entities, entity_names)
            if i == "alocado": populatealocado(f, i, j, entities, entity_names)
            if i == "acciona": populateacciona(f, i, j, entities)
            if i == "coordenador": populatecoordenador(f, i, j)
            if i == "audita": populateaudita(f, i, j, entities, entity_names)
            if i == "solicita": populatesolicita(f, i, j, times)
        f.write("\n")
    f.close()

if __name__ == "__main__":
    main()
```

Usamos funções auxiliares para gerar **intervalos de tempos** sequenciais, para gerar **números de telefone** aleatórios, para gerar **nomes** aleatórios e nomes de entidades aleatórias seguindo um padrão de "**nome_base**" + **random word**, onde nome_base vinha de uma lista de meios oficiais dados no enunciado.

Os requisitos para correr este script é a utilização do **python > 3.X**, **numpy** (apesar de conseguirmos correr o mesmo sem isso bastando tirar as instancias de np.array) e **time**.

```
def randomtimes(start, end, frmt, n):
    stime = datetime.datetime.strptime(start, frmt)
    etime = datetime.datetime.strptime(end, frmt)
    td = etime - stime
    dates = [random.random() * td + stime for _ in range(n)]
    dates = sorted(dates)
    return dates

def randomphones(d, n):
    rangestart = 10**(d-1)
    rangeend = (10**d)-1
    numbers = [randint(rangestart, rangeend) for _ in range(n)]
    return random.sample(numbers, 100)

def randomnames(m):
    vowels = "aeiou"
    consonants = "".join(set(string.ascii_lowercase) - set(vowels))
    word = ""
    for i in range(m):
        if i % 2 == 0:
            word += random.choice(consonants)
        else:
            word += random.choice(vowels)
    return word

def generatelistname(n):
    names = []
    for i in range(n):
        names += [randomnames(5)]
    return random.sample(names, 100)

def generatelistentity(n):
    baseline = ["bombeiros", "exercito", "forca aerea", "policia", "municipio"]
    names = []
    for i in range(n):
        ind = randint(0, 4)
        names += [baseline[ind] + "_de_" + randomnames(5)]
    return random.sample(names, 100)
```

Para preencher, demos uso a manipulação de ficheiros e à utilização das funções auxiliares, combinando-as para inserir conteúdo do tipo **insert into table_name values (table_args)**. A baixo pode-se ver duas utilizações para preencher a tabela **coordenador** e **audita**.

```
def populatecoordenador(f, i, j):
    f.write("insert into {} values ({});\n".format(i, j + 1))

def populateaudita(f, i, j, entities, entity_names):
    times_h = randomtimes("13:28:00 2018-01-01", "08:50:34 2018-03-02", '%H:%M:%S %Y-%m-%d', 101)
    f.write("insert into {} values ({} , {}, '{}', {}, '{}', '{}', '{}','{}');\n".format(i, j+1, j+1, \
        entities[j], j+1, \
        times_h[j].replace(microsecond=0), \
        times_h[j+1].replace(microsecond=0), \
        str(times_h[j].replace(microsecond=0))[:10], \
        "Auditoria do processo " + str(j+1) + " que comecou no dia " + str(times_h[j].replace(microsecond=0)) \
        + " e acabou no dia " + str(times_h[j+1].replace(microsecond=0)) + " da entidade " + \
        str(entities[j]) + " do Meio de " + str(entity_names[j]) ))
```