Universidade Estadual de Campinas – Instituto de Computação
MC536 – Bancos de Dados – Profª. Claudia Bauzer Medeiros

Trabalho 3

agência matrimonial GetLove

Bruno Sepulveda (165207) · Giovanni Bertão (173325) · Guilherme Gama (173608)
Guilherme Perrotta (173846) · Gustavo Salibi (174135) · João Pedro Martins (176117)

ESQUEMA

Pessoa(id_pessoa, cpf, email, nome, data_nascimento, sexo, cidade, uf, numero_filhos, tabaco, alcool, outras_drogas, religiao, tipo_de_personalidade, latitude, longitude) - **500 tuplas**

Moderador(id_moderador, nome) - **10 tuplas**

Ocorrencia(id_ocorrencia, tipo, id_implicado, ref_arquivo_mensagem, data_ocorrencia) - ***250 tuplas***

Empresa(id_empresa, cnpj, nome, email, telefone, cidade, uf) - **50 tuplas**

Empresa_de_dados(id_empresa) - **3 tuplas**

Empresa_de_servicos(id_empresa) - **47 tuplas**

Combina_com(id_pessoa1, id_pessoa2, ref_arquivo_conversa, data_combinacao) - **1255 tuplas**

Aprova(id_pessoa, id_aprovado, data_aprovacao) - **4204 tuplas**

Tem_dados_vendidos_a(id_pessoa, id_empresa) - **900 tuplas**

Abre(id_ocorrencia, id_pessoa) - **250 tuplas**

Avalia(id_ocorrencia, id_moderador) - **200 tuplas**

Plano_premium(id_pessoa, data_expiracao) - **200 tuplas**

Servico(id_empresa, numero_servico, preco, categoria, descricao, latitude, longitude) - **1000 tuplas**

Foto(id_pessoa, numero_foto, ref_arquivo_imagem) - **1500 tuplas**

Imagem_servico(id_empresa, numero_servico, numero_imagem, ref_arquivo_imagem) - **3000 tuplas**

Alterações no esquema: O campo ref_arquivo_descricao da tabela Servicos foi alterado para descricao, assim podemos utilizar o recurso de pattern matching do SQL para buscar palavras-chave neste campo.

Consultas

Em vermelho estão as variáveis fornecidas pela aplicação.

Encontrar pessoas: em relação ao usuário, listar as primeiras 200 pessoas em ordem de afinidade e de distância utilizando os critérios

- Sexo (M, F ou ambos)
- Número de filhos
- Fuma tabaco (0 = não, 1 = sim, < 2 = tanto faz)
- Consome bebida alcoolica (0 = não, 1 = sim, < 2 = tanto faz)
- Utiliza outras drogas (0 = não, 1 = sim, < 2 = tanto faz)
- Religião ([0;10] religiões, < 11 = qualquer uma)
- Distância máxima em quilômetros (definida na função `dist_geo`, cujo código está no fim deste documento).

```
SELECT p2.id_pessoa, email, nome, data_nascimento, sexo, cidade, uf,
numero_filhos, tabaco, alcool, outras_drogas, religiao,
    abs(p1.tipo_de_personalidade - p2.tipo_de_personalidade) as afinidade,
dist_geo(p1.latitude, p1.longitude, p2.latitude, p2.longitude) as distancia
FROM
    (SELECT tipo_de_personalidade, latitude, longitude
     FROM Pessoa
     WHERE id_pessoa = 206) as p1,
    (SELECT *
     FROM Pessoa
     WHERE id_pessoa <> 206 and sexo = 'F' and numero_filhos < 5 and tabaco
< 2 and alcool < 2 and outras_drogas < 2 and religiao < 11 ) as p2
HAVING distancia < 1000
ORDER BY afinidade, distancia
LIMIT 200;
```

Gerar encontro: de acordo com as escolhas do usuário, mostra os primeiros 100 serviços compatíveis ordenados por distância e por preço. Critérios disponíveis

- Categoria (restaurantes, entretenimento, hospedagem)
- Busca de palavras na descrição
- Preço
- Distância máxima

```
SELECT id_empresa, numero_servico, preco, categoria, descricao,
dist_geo(lat1, long1, lat2, long2) as distancia
FROM
    (SELECT latitude as lat1, longitude as long1
     FROM Pessoa
     WHERE id_pessoa = 206) as pessoa,
    (SELECT id_empresa, numero_servico, preco, categoria, descricao,
latitude as lat2, longitude as long2
    FROM Servico
    WHERE categoria LIKE 'restaurantes' and preco < 1000 and descricao LIKE
'%japones%') as serv
HAVING distancia < 500
ORDER BY distancia, preco
LIMIT 100
```

Listar combinações: lista as combinações do usuário, começando pelas mais recentes.

```
SELECT *
FROM Combina_com
WHERE id_pessoa1 = 206 OR id_pessoa2 = 206
ORDER BY data_combinacao DESC;
```

Listar quem me aprovou: permite ver quem aprovou o usuário mas ainda não é uma combinação, começando pelas mais recentes.

```
SELECT *
FROM Aprova
WHERE id_aprovado = 149
ORDER BY data_aprovacao DESC;
```

Recuperar log de conversa:

```
SELECT ref_arquivo_conversa
FROM Combina_com
WHERE (id_pessoa1 = 206 AND id_pessoa2 = 27) OR (id_pessoa2 = 206 AND
id_pessoa1 = 27);
```

Listar ocorrências abertas por cliente x:

```
SELECT *
FROM Abre NATURAL JOIN Ocorrencia
WHERE Abre.id_pessoa = 361;
```

Listar ocorrências sobre cliente x:

```
SELECT *
FROM Ocorrencia
WHERE tipo LIKE "usuario" AND id_implicado = 345;
```

Listar ocorrências sobre a empresa x:

```
SELECT *
FROM Ocorrencia
WHERE tipo LIKE "empresa" AND id_implicado = 36;
```

Listar ocorrências sobre a getLove:

```
SELECT *
FROM Ocorrencia
WHERE id_implicado IS NULL;
```

Listar ocorrências abertas:

```
SELECT *
FROM Ocorrencia
WHERE id_ocorrencia NOT IN (SELECT id_ocorrencia FROM Avalia);
```

Atualizações

Em vermelho estão as variáveis fornecidas pela aplicação.

Remover combinação entre usuários com id X e Y

```
DELETE FROM Combina_com
WHERE (id_pessoa1 = X and id_pessoa2 = Y) or (id_pessoa1 = Y and id_pessoa2
= X);
```

Alterar perfil

(O usuário de id **1** alterou o perfil, suponha que ele passou a ser fumante).

```
UPDATE Pessoa
SET tabaco = 1
WHERE id_pessoa = 1;
```

Todos os campos do perfil podem ser alterados, inclusive data de nascimento. A query é algo genérico UPDATE Pessoa SET coluna1 = valor1, coluna2 = valor2, ..., colunaN = valorN WHERE id_pessoa=1;

Contudo, a API limita o que o usuário pode alterar em seu perfil. O usuário não pode atualizar seu próprio 'id_pessoa').

Refazer teste de personalidade

Suponha que o cliente de id **X** refaça o teste de personalidade e adquira um resultado igual a **Y**

```
UPDATE Pessoa
SET tipo_personalidade = Y
WHERE id_pessoa = X;
```

Recalcular posição geográfica

```
UPDATE Pessoa
SET latitude = NOVA_LAT, longitude = NOVA_LONG
WHERE id_pessoa = X;
```

Alterar fotos

a) Suponha que o usuário de id **X** queira adicionar *'n'* novas fotos:

```
INSERT INTO Foto VALUES(X, NUM_FOTOi, ARQi);
```

Para i = 1,2,3,...,n.
NUM_FOTO$_i$ representa a i-ésima foto adicionada quantitativamente e **ARQ**$_i$ representa o arquivo da i-ésima foto.

b) Suponha que o usuário de id **1** queira remover a foto *3*:

```
DELETE FROM Foto
WHERE id_pessoa = 1 AND numero_foto = 3;
```

c) Suponha que o usuário de id **1** queira remover todas as suas fotos:

```
DELETE FROM Foto
WHERE id_pessoa = 1;
```

Alterar plano

a) Suponha que o plano de alguem venceu:

```
DELETE FROM Plano_premium
WHERE data_expiracao < CURRENT_DATE();
```

b) Suponha que o id **1** adquiriu um plano que expira no dia **18-08-2019**:

```
INSERT INTO Plano_premium(id_pessoa, data_expiracao) VALUES(1,
'18-08-2019');
```

Alterar lista de empresas parceiras

a) Remover Empresa de id **1**

```
DELETE FROM Imagem_servico
WHERE id_empresa = 1;
DELETE FROM Servico
WHERE id_empresa = 1;
DELETE FROM Tem_dados_vendidos_a
WHERE id_empresa = 1;
DELETE FROM Empresa_de_dados
WHERE id_empresa = 1;
DELETE FROM Empresa_de_servicos
WHERE id_empresa = 1;
DELETE FROM Empresa
WHERE id_empresa = 1;
```

b) Alterar dados da Empresa de id **1**
Semelhante ao alterar perfil dos usuários. Como exemplo, suponha que a empresa mudou para a cidade **W** e seu telefone para **Y**.

```
UPDATE Empresa
SET cidade='W', telefone='Y'
WHERE id_empresa = 1.
```

c) Empresa exclusivamente de dados passa a ser empresa exclusivamente de serviços

```
DELETE FROM Tem_dados_vendidos_a
WHERE id_empresa = 1;
DELETE FROM Empresa_de_dados WHERE id_empresa = 1;
INSERT INTO Empresa_de_servicos VALUES(1);
```

d) Adicionar uma nova Empresa
Como os campos não podem ser '*NULL*'.

```
INSERT INTO Empresa VALUES(999,'CNPJ','NOME'',EMAIL', 'TELEFONE', 'CIDADE',
'UF');
```

Consecutivamente a empresa sera listada como 'Empresa_de_servicos' **ou** 'Empresa_de_dados':

```
INSERT INTO Empresa_de_servicos VALUES(999);
OU
INSERT INTO Empresa_de_dados VALUES(999);
```

Alterar lista de produtos

a) Remover Produto de id **1** da empresa id **1**

```
DELETE FROM Imagem_servico
WHERE id_empresa = 1 AND numero_servico = 1;
DELETE FROM Servico
WHERE id_empresa = 1 AND numero_servico = 1;
```

b) Inserir novo produto
Suponha que um produto de preço **2**, de categoria **entretenimento**, com a descriçao **Z** e que está localizado nas coordenadas (**50,50**) seja oferecido pela empresa de id **1**.

```
 INSERT INTO Servico(id_empresa, numero_servico, preco, categoria,
descricao, latitude, longitude) VALUES(1, 1, 2, 'entretenimento', 'Z', 50,
50);
```

Consecutivamente, o serviço pode ou não receber uma imagem(s) descritiva **K**.

```
INSERT INTO Imagem_servico VALUES(1, 1, 1, 'K');
```

c) Atualizar preço ou localização do produto de número **1** oferecido por **1**

```
UPDATE Produto
SET preco=500, latitude= 55, longitude=55
WHERE id_empresa = 1 AND numero_servico = 1;
```

Alterar imagem de produto

a) A empresa de id **1** quer adicionar *'n'* novas imagens ao produto de número **1**
Suponha que $\mathbf{K}_i$ seja o *i-ésimo* arquivo de imagem que se quer adicionar, para i=1,2,3,...,n.

```
INSERT INTO Imagem_servico VALUES(1, 1, 999,'K_i');
```

b) A empresa de id **1** quer remover *'n'* imagens de um produto de número **1**
Suponha que **7** seja o número do *i-ésimo* arquivo de imagem que se quer remover, para i=1,2,3,...,n.

```
DELETE FROM Imagem_servico
WHERE id_empresa = 1 and numero_servico = 1 and numero_imagem = 7;
```

c) A empresa de id **1** quer remover todas as imagens de um produto de número **1**

```
DELETE FROM Imagem_servico
WHERE id_empresa = 1 AND numero_servico = 1;
```

Apêndice: funções definidas pelo usuário

dist_geo: Calcula a distância em quilômetros entre duas coordenadas geográficas.

```
DELIMITER //
      CREATE FUNCTION  `dist_geo`
      (lat1 FLOAT, lon1 FLOAT, lat2 FLOAT, lon2 FLOAT)
RETURNS FLOAT
BEGIN
      DECLARE pi, q1, q2, q3 FLOAT;
      DECLARE rads FLOAT DEFAULT 0;
      SET pi = PI();
      SET lat1 = lat1 * pi / 180;
      SET lon1 = lon1 * pi / 180;
      SET lat2 = lat2 * pi / 180;
      SET lon2 = lon2 * pi / 180;
      SET q1 = POW(SIN((lat1 - lat2) / 2),2);
      SET q2 = POW(SIN((lon1 - lon2) / 2),2);
      SET q3 = COS(lat1) * COS(lat2);
      SET rads = ASIN(SQRT(q1 + q3 * q2));
      RETURN 6378.388 * rads;
END //
```

A função é uma implementação da fórmula de Haversine[1]:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

Onde *r* é o raio da Terra (presumido aqui como 6378.388 km), ϕ_1, ϕ_2 são as latitudes dos dois pontos, e λ_1, λ_2 são as longitudes dos dois pontos.

---

[1] https://en.wikipedia.org/wiki/Haversine_formula