

Bases de Dados

Licenciatura em Engenharia Informática

Relatório do Trabalho Prático

Rede Social: Fãs de Sobremesas



UNIVERSIDADE DE ÉVORA

Miguel Pombeiro, 57829 | Miguel Rocha, 58501

Departamento de Informática
Universidade de Évora
Novembro 2024

Índice

1	Chaves	3
1.1	Membro(Nome, IdMembro, País, DataNasc)	3
1.2	Doce(Nome, Descrição, Género)	3
1.3	Amigo(Membro1, Membro2)	3
1.4	Criou(Membro, Doce)	3
1.5	Fez(Membro, Doce, Tempo, Aspeto, Sabor)	3
1.6	Ingrediente(Nome, Custo)	4
1.7	temIngrediente(Doce, Ingrediente, Quantidade)	4
2	Criação de Tabelas	5
2.1	Membro	5
2.2	Doce	5
2.3	Amigo	5
2.4	Criou	5
2.5	Fez	6
2.6	Ingrediente	6
2.7	temIngrediente	6
2.8	Chaves Estrangeiras	7
3	Inserção de Dados	8
3.a	Inserir na relação membro	8
3.b	Inserir nas relações ingrediente e temIngrediente	8
3.c	Inserir nas Relações criou, doce e fez	11
3.d	Inserir na Relação amigo	13
4	Perguntas: Expressões em Álgebra Relacional e SQL	15
4.a	Indique os nomes dos membros espanhóis que criaram doces Regionais que têm Chocolate e Canela como ingredientes.	15
4.b	Quais são os doces Tradicionais criados pelos amigos do "Joaquim José" que quando foram feitos tiveram pelo menos um 5 no aspeto?	16
4.c	Qual é o nome dos membros que só fizeram doces dos seus amigos?	17
4.d	Quais são os doces com Natas que não tiveram um 5 no sabor?	18
4.e	Quais são os membros que criaram doces com Canela e Leite?	19
4.f	Quais são os doces que têm Maçã ou Baunilha?	19
4.g	Qual é o custo do Arroz Doce?	20
4.h	Indique o custo de cada doce	20
4.i	Para cada membro indique o número de doces que criou	21

4.j	Qual é o nome do membro que criou mais doces?	22
4.k	Qual é o nome dos membros que fizeram o doce mais caro?	23
4.l	Quais são os membros que são amigos de todos os amigos do Joaquim José	24
4.m	Quais são os membros que fizeram todos os doces com Baunilha?	25
4.n	Quais são os doces que quando foram feitos tiveram sempre um 1 no tempo?	26
4.o	Quais são os doces mais baratos e mais rápidos de fazer?	27

1 Chaves

1.1 Membro(Nome, IdMembro, País, DataNasc)

Considerou-se que os nomes dos membros não são únicos.

Chaves Candidatas: (IdMembro)

Chave Primária: (IdMembro)

Chaves Estrangeiras: *Não há*

1.2 Doce(Nome, Descrição, Género)

Considerou-se que dois doces diferentes não devem ter uma mesma descrição.

Chaves Candidatas: (Nome) ; (Descrição)

Chave Primária: (Nome)

Chaves Estrangeiras: *Não há*

1.3 Amigo(Membro1, Membro2)

Chaves Candidatas: (Membro1, Membro2)

Chave Primária: (Membro1, Membro2)

Chaves Estrangeiras: (Membro1) - da relação "membro"
(Membro2) - da relação "membro"

1.4 Criou(Membro, Doce)

Chaves Candidatas: (Membro, Doce)

Chave Primária: (Membro, Doce)

Chaves Estrangeiras: (Membro) - da relação "membro"
(Doce) - da relação "doce"
(Membro, Doce) - da relação "fez"

1.5 Fez(Membro, Doce, Tempo, Aspeto, Sabor)

Considerou-se que um membro não pode fazer um mesmo doce mais do que uma vez.

Chaves Candidatas: (Membro, Doce)

Chave Primária: (Membro, Doce)

Chaves Estrangeiras: (Membro) - da relação "membro"
(Doce) - da relação "doce"

1.6 Ingrediente(Nome, Custo)

Chaves Candidatas: (Nome)

Chave Primária: (Nome)

Chaves Estrangeiras: *Não há*

1.7 temIngrediente(Doce, Ingrediente, Quantidade)

Chaves Candidatas: (Doce, Ingrediente)

Chave Primária: (Doce, Ingrediente)

Chaves Estrangeiras: (Doce) - Da relação "doce"

(Ingrediente) - Da relação "ingrediente"

2 Criação de Tabelas

2.1 Membro

```
1 CREATE TABLE membro (  
2     Nome TEXT NOT NULL,  
3     IdMembro VARCHAR(100) NOT NULL UNIQUE ,  
4     Pais VARCHAR(50) NOT NULL,  
5     DataNasc DATE NOT NULL,  
6     PRIMARY KEY (IdMembro)  
7 );
```

2.2 Doce

```
1 CREATE TABLE doce (  
2     Nome VARCHAR(50) NOT NULL UNIQUE,  
3     Descricao TEXT NOT NULL UNIQUE,  
4     Genero VARCHAR(50) NOT NULL,  
5     PRIMARY KEY (Nome)  
6 );
```

2.3 Amigo

```
1 CREATE TABLE amigo (  
2     Membro1 VARCHAR(100) NOT NULL,  
3     Membro2 VARCHAR(100) NOT NULL,  
4     PRIMARY KEY (Membro1, Membro2)  
5 );
```

2.4 Criou

```
1 CREATE TABLE criou(  
2     Membro VARCHAR(100) NOT NULL,  
3     Doce VARCHAR(50) NOT NULL,  
4     PRIMARY KEY (Membro, Doce)  
5 );
```

2.5 Fez

```
1 CREATE TABLE fez (  
2     Membro VARCHAR(100) NOT NULL,  
3     Doce VARCHAR(50) NOT NULL,  
4     Tempo SMALLINT NOT NULL,  
5     Aspeto SMALLINT NOT NULL,  
6     Sabor SMALLINT NOT NULL,  
7     PRIMARY KEY (Membro, Doce)  
8 );
```

2.6 Ingrediente

```
1 CREATE TABLE ingrediente (  
2     Nome VARCHAR(50) NOT NULL UNIQUE,  
3     Custo NUMERIC(10, 7) NOT NULL,  
4     PRIMARY KEY (Nome)  
5 );
```

2.7 temIngrediente

```
1 CREATE TABLE temIngrediente (  
2     Doce VARCHAR(50) NOT NULL,  
3     Ingrediente VARCHAR(50) NOT NULL,  
4     Quantidade INT NOT NULL,  
5     PRIMARY KEY (Doce, Ingrediente)  
6 );
```

2.8 Chaves Estrangeiras

2.8.1 Amigo

```
1 ALTER TABLE amigo ADD FOREIGN KEY (Membro1) REFERENCES membro(IdMembro)
  on DELETE RESTRICT;
2 ALTER TABLE amigo ADD FOREIGN KEY (Membro2) REFERENCES membro(IdMembro)
  on DELETE RESTRICT;
```

2.8.2 Criou

```
1 ALTER TABLE criou ADD FOREIGN KEY (Membro) REFERENCES membro(IdMembro)
  on DELETE RESTRICT;
2 ALTER TABLE criou ADD FOREIGN KEY (Doce) REFERENCES doce(Nome) on DELETE
  RESTRICT;
3 ALTER TABLE criou ADD FOREIGN KEY (Membro, Doce) REFERENCES fez(Membro,
  Doce) on DELETE RESTRICT;
```

2.8.3 Fez

```
1 ALTER TABLE fez ADD FOREIGN KEY (Membro) REFERENCES membro(IdMembro) on
  DELETE RESTRICT;
2 ALTER TABLE fez ADD FOREIGN KEY (Doce) REFERENCES doce(Nome) on DELETE
  RESTRICT;
```

2.8.4 temIngrediente

```
1 ALTER TABLE temIngrediente ADD FOREIGN KEY (Doce) REFERENCES doce(Nome)
  on DELETE RESTRICT;
2 ALTER TABLE temIngrediente ADD FOREIGN KEY (Ingrediente) REFERENCES
  ingrediente(Nome) on DELETE RESTRICT;
```


3 Inserção de Dados

3.a Inserir na relação membro

```
1 INSERT INTO membro (nome, IdMembro, pais, datanasc) VALUES
2   ('Joaquim José', 'joaquimjose', 'Portugal', '1990-01-01'),
3   ('Maria de Lourdes
4     Modesto', 'marialourdesmodesto', 'Portugal', '1930-06-01'),
5   ('Alex Atala', 'alexatala', 'Brasil', '1968-06-03'),
6   ('Ferran Adrià', 'ferranadria', 'Espanha', '1962-05-14'),
7   ('Pierre Hérme', 'pierreherme', 'França', '1961-11-20'),
8   ('Elizabeth Falkner', 'elizabethfalkner', 'EUA', '1966-02-12'),
9   ('Lorraine Pascale', 'lorraineascale', 'Reino Unido', '1972-11-17'),
10  ('Gaston Lenôtre', 'gastonlenotre', 'França', '1920-05-28'),
11  ('Heston Blumenthal', 'hestonblumenthal', 'Reino Unido', '1966-05-27'),
12  ('Mineko Kato', 'minekokato', 'Japão', '1900-01-01');
13 COMMIT;
```

3.b Inserir nas relações ingrediente e temIngrediente

```
1 INSERT INTO ingrediente (Nome, Custo) VALUES
2   ('Açúcar', 0.00109),
3   ('Farinha', 0.00069),
4   ('Chocolate', 0.00995),
5   ('Ovos', 0.2),
6   ('Natas', 0.00375),
7   ('Leite', 0.00082),
8   ('Água', 0.00014),
9   ('Maçã', 0.00229),
10  ('Manteiga', 0.00672),
11  ('Alfarroba', 0.01316),
12  ('Baunilha', 3.495),
13  ('Canela', 0.4975),
14  ('Pimenta', 0.37556),
15  ('Café', 0.01845),
16  ('Arroz', 0.00119),
17  ('Limão', 0.00199),
18  ('Bolacha', 0.00211),
19  ('Amêndoa', 0.01495);
20 COMMIT;
```

```

1  INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
2
3  -- Arroz Doce
4      ('Arroz Doce', 'Arroz', 300),
5      ('Arroz Doce', 'Açúcar', 300),
6      ('Arroz Doce', 'Leite', 600),
7      ('Arroz Doce', 'Canela', 7),
8      ('Arroz Doce', 'Baunilha', 7),
9      ('Arroz Doce', 'Limão', 1),
10
11  -- Mousse de Chocolate
12      ('Mousse de Chocolate', 'Chocolate', 200),
13      ('Mousse de Chocolate', 'Água', 200),
14
15  --Bolo de Chocolate e Canela
16      ('Bolo de Chocolate e Canela', 'Canela', 30),
17      ('Bolo de Chocolate e Canela', 'Açúcar', 150),
18      ('Bolo de Chocolate e Canela', 'Farinha', 600),
19      ('Bolo de Chocolate e Canela', 'Ovos', 4),
20      ('Bolo de Chocolate e Canela', 'Leite', 900),
21      ('Bolo de Chocolate e Canela', 'Manteiga', 40),
22      ('Bolo de Chocolate e Canela', 'Chocolate', 150),
23
24  --Sericaia
25      ('Sericaia', 'Leite', 500),
26      ('Sericaia', 'Açúcar', 170),
27      ('Sericaia', 'Ovos', 6),
28      ('Sericaia', 'Canela', 30),
29      ('Sericaia', 'Limão', 1),
30
31  --Pudim de Abade de Priscos
32      ('Pudim Abade de Priscos', 'Ovos', 15),
33      ('Pudim Abade de Priscos', 'Açúcar', 500),
34      ('Pudim Abade de Priscos', 'Água', 650),
35      ('Pudim Abade de Priscos', 'Canela', 30),
36
37  --Bolo de Bolacha
38      ('Bolo de Bolacha', 'Manteiga', 250),
39      ('Bolo de Bolacha', 'Bolacha', 400),
40      ('Bolo de Bolacha', 'Açúcar', 200),
41      ('Bolo de Bolacha', 'Ovos', 3),
42      ('Bolo de Bolacha', 'Café', 90),

```

```

42      ('Bolo de Bolacha' ,
43
44      --Pão de Rala
45      ('Pão de Rala' ,
46      ('Pão de Rala' ,
47      ('Pão de Rala' ,
48
49      --Tarte de Alfarroba
50      ('Tarte de Alfarroba' ,
51      ('Tarte de Alfarroba' ,
52      ('Tarte de Alfarroba' ,
53      ('Tarte de Alfarroba' ,
54
55      --Ovos Moles de Aveiro
56      ('Ovos Moles de Aveiro' ,
57      ('Ovos Moles de Aveiro' ,
58      ('Ovos Moles de Aveiro' ,
59
60      --Tarte de Natas
61      ('Tarte de Natas' ,
62      ('Tarte de Natas' ,
63      ('Tarte de Natas' ,
64      ('Tarte de Natas' ,
65
66      --Tarte de Maça
67      ('Tarte de Maça' ,
68      ('Tarte de Maça' ,
69      ('Tarte de Maça' ,
70      ('Tarte de Maça' ,
71      ('Tarte de Maça' ,
72      ('Tarte de Maça' ,
73
74      COMMIT;
```

'Baunilha' , 7),

'Ovos' , 20),

'Açúcar' , 700),

'Amêndoa' , 500),

'Açúcar' , 150),

'Ovos' , 6),

'Amêndoa' , 100),

'Alfarroba' , 50),

'Ovos' , 8),

'Açúcar' , 150),

'Água' , 150),

'Natas' , 600),

'Ovos' , 6),

'Açúcar' , 200),

'Bolacha' , 300),

'Farinha' , 80),

'Açúcar' , 200),

'Leite' , 250),

'Água' , 250),

'Ovos' , 4),

'Maça' , 600);

3.c Inserir nas Relações criou, doce e fez

```
1 INSERT INTO criou (Membro, Doce) VALUES
2     ('marialourdesmodesto', 'Arroz Doce'),
3     ('hestonblumenthal', 'Mousse de Chocolate'),
4     ('ferranadria', 'Bolo de Chocolate e Canela'),
5     ('pierreherme', 'Sericaia'),
6     ('marialourdesmodesto', 'Pudim Abade de Priscos'),
7     ('elizabethfalkner', 'Bolo de Bolacha'),
8     ('hestonblumenthal', 'Pão de Rala'),
9     ('marialourdesmodesto', 'Tarte de Alfarroba'),
10    ('alexatala', 'Ovos Moles de Aveiro'),
11    ('ferranadria', 'Tarte de Natas'),
12    ('pierreherme', 'Tarte de Maçã');
13
14 COMMIT;
```



```
1 INSERT INTO doce (Nome, Descricao, Genero) VALUES
2     ('Arroz Doce', 'Arroz cozido em leite com açúcar.', 'Tradicional'),
3     ('Mousse de Chocolate', 'Mousse de chocolate com água.',
4         'Tradicional'),
5     ('Bolo de Chocolate e Canela', 'Simples Bolo com chocolate e
6         canela.', 'Regional'),
7     ('Sericaia', 'Doce conventual alentejano cozido num prato de
8         barro.', 'Regional'),
9     ('Pudim Abade de Priscos', 'Pudim de ovos com toucinho.', 'Regional'),
10    ('Bolo de Bolacha', 'Bolachas embebidas em café e
11        natas.', 'Tradicional'),
12    ('Pão de Rala', 'Doce conventual alentejano com amêndoa e
13        ovos.', 'Regional'),
14    ('Tarte de Alfarroba', 'Tarte com farinha de alfarroba.', 'Regional'),
15    ('Ovos Moles de Aveiro', 'Doce conventual aveirense com ovos e
16        açúcar.', 'Regional'),
17    ('Tarte de Natas', 'Tarte com natas e açúcar.', 'Tradicional'),
18    ('Tarte de Maçã', 'Tarte com maçã e canela.', 'Tradicional');
19
20 COMMIT;
```

```

1  INSERT INTO fez (Membro, Doce, Tempo, Aspeto, Sabor) VALUES
2      ('marialourdesmodesto', 'Arroz Doce', 4, 2, 4),
3      ('hestonblumenthal', 'Arroz Doce', 3, 2, 5),
4
5      ('hestonblumenthal', 'Mousse de Chocolate', 1, 3, 4),
6      ('pierreherme', 'Mousse de Chocolate', 1, 5, 4),
7      ('minekokato', 'Mousse de Chocolate', 3, 1, 2),
8
9      ('ferranadria', 'Bolo de Chocolate e Canela', 5, 3, 4),
10     ('alexatala', 'Bolo de Chocolate e Canela', 2, 2, 3),
11
12     ('pierreherme', 'Sericaia', 1, 2, 4),
13     ('alexatala', 'Sericaia', 2, 1, 1),
14
15     ('marialourdesmodesto', 'Pudim Abade de Priscos', 1, 5, 5),
16     ('joaquimjose', 'Pudim Abade de Priscos', 2, 1, 4),
17
18     ('elizabethfalkner', 'Bolo de Bolacha', 1, 4, 5),
19     ('lorrainepascale', 'Bolo de Bolacha', 2, 5, 5),
20     ('hestonblumenthal', 'Bolo de Bolacha', 2, 5, 5),
21
22     ('minekokato', 'Pão de Rala', 4, 1, 1),
23     ('lorrainepascale', 'Pão de Rala', 4, 2, 2),
24     ('hestonblumenthal', 'Pão de Rala', 3, 3, 3),
25
26     ('marialourdesmodesto', 'Tarte de Alfarroba', 2, 5, 5),
27     ('ferranadria', 'Tarte de Alfarroba', 1, 1, 1),
28
29     ('alexatala', 'Ovos Moles de Aveiro', 1, 5, 5),
30     ('joaquimjose', 'Ovos Moles de Aveiro', 1, 4, 2),
31
32     ('ferranadria', 'Tarte de Natas', 2, 1, 1),
33     ('joaquimjose', 'Tarte de Natas', 1, 5, 4),
34
35     ('pierreherme', 'Tarte de Maçã', 5, 4, 4),
36     ('marialourdesmodesto', 'Tarte de Maçã', 5, 5, 5);
37
38  COMMIT;

```

3.d Inserir na Relação amigo

```
1  INSERT INTO amigo (Membro1, Membro2) VALUES
2      ('joaquimjose',      'marialourdesmodesto'),
3      ('joaquimjose',      'alexatala'),
4      ('joaquimjose',      'ferranadria'),
5      ('joaquimjose',      'pierreherme'),
6      ('joaquimjose',      'elizabethfalkner'),
7      ('joaquimjose',      'lorraineascale'),
8      ('joaquimjose',      'gastonlenotre'),
9      ('joaquimjose',      'hestonblumenthal'),
10     ('joaquimjose',      'minekokato'),
11     ('marialourdesmodesto', 'joaquimjose'),
12     ('alexatala',         'joaquimjose'),
13     ('ferranadria',       'joaquimjose'),
14     ('pierreherme',       'joaquimjose'),
15     ('elizabethfalkner',  'joaquimjose'),
16     ('lorraineascale',    'joaquimjose'),
17     ('gastonlenotre',     'joaquimjose'),
18     ('hestonblumenthal',  'joaquimjose'),
19     ('minekokato',        'joaquimjose'),
20
21     ('hestonblumenthal',  'gastonlenotre'),
22     ('hestonblumenthal',  'minekokato'),
23     ('gastonlenotre',     'hestonblumenthal'),
24     ('minekokato',        'hestonblumenthal'),
25
26     ('alexatala',         'lorraineascale'),
27     ('lorraineascale',    'alexatala'),
28
29     ('ferranadria',       'elizabethfalkner'),
30     ('ferranadria',       'alexatala'),
31     ('elizabethfalkner',  'ferranadria'),
32     ('alexatala',         'ferranadria'),
33
34     ('marialourdesmodesto', 'lorraineascale'),
35     ('marialourdesmodesto', 'elizabethfalkner'),
36     ('lorraineascale',    'marialourdesmodesto'),
37     ('elizabethfalkner',  'marialourdesmodesto'),
38
39
```

```
40      ('gastonlenotre' ,      'minekokato' ),
41      ('gastonlenotre' ,      'marialourdesmodesto' ),
42      ('gastonlenotre' ,      'ferranadria' ),
43      ('gastonlenotre' ,      'pierreherme' ),
44      ('gastonlenotre' ,      'elizabethfalkner' ),
45      ('gastonlenotre' ,      'lorraineascale' ),
46      ('gastonlenotre' ,      'alexatala' ),
47      ('minekokato' ,          'gastonlenotre' ),
48      ('marialourdesmodesto' , 'gastonlenotre' ),
49      ('ferranadria' ,          'gastonlenotre' ),
50      ('pierreherme' ,          'gastonlenotre' ),
51      ('elizabethfalkner' ,     'gastonlenotre' ),
52      ('lorraineascale' ,       'gastonlenotre' ),
53      ('alexatala' ,            'gastonlenotre' );
54
55 COMMIT;
```

4 Perguntas: Expressões em Álgebra Relacional e SQL

4.a Indique os nomes dos membros espanhóis que criaram doces Regionais que têm Chocolate e Canela como ingredientes.

Álgebra Relacional

$$\begin{aligned}
 sp_cria_regional &\leftarrow \pi_{\text{membro.Nome}, \text{criou.Doce}} \left(\sigma_{\begin{array}{l} \text{País} = \text{"Espanha"} \\ \wedge \\ \text{membro.IdMembro} = \text{criou.membro} \\ \wedge \\ \text{doce.género} = \text{"Regional"} \\ \wedge \\ \text{doce.nome} = \text{criou.doce} \end{array}} (membro \times criou \times doce) \right) \\
 choc_canela &\leftarrow \pi_{tI1.Doce} \left(\sigma_{\begin{array}{l} tI1.Doce = tI2.Doce \\ \wedge \\ tI1.Ingrediente = \text{"Canela"} \\ \wedge \\ tI2.Ingrediente = \text{"Chocolate"} \end{array}} (temIngrediente \text{ as } tI1 \times temIngrediente \text{ as } tI2) \right) \\
 \pi_{nome} (sp_cria_regional \bowtie choc_canela)
 \end{aligned}$$

PostgreSQL

```

1 WITH sp_cria_regional AS (
2   SELECT membro.nome, criou.doce
3   FROM membro, criou, doce
4   WHERE membro.pais = 'Espanha' AND
5         membro.idmembro = criou.membro AND
6         doce.nome = criou.doce AND
7         doce.genero = 'Regional'),
8
9   choc_canela AS (
10    SELECT tI1.doce
11    FROM temingrediente AS tI1, temingrediente AS tI2
12    WHERE tI1.doce = tI2.doce AND
13          tI1.ingrediente = 'Canela' AND
14          tI2.ingrediente = 'Chocolate')
15
16 SELECT DISTINCT sp_cria_regional.nome
17 FROM sp_cria_regional NATURAL INNER JOIN choc_canela;

```


4.b Quais são os doces Tradicionais criados pelos amigos do "Joaquim José" que quando foram feitos tiveram pelo menos um 5 no aspeto?

Nesta questão são pedidas informações relacionadas com os amigos de "Joaquim José". Sendo esta expressão um nome e não um identificador único (superchave), pode existir, na base de dados, mais do que uma pessoa com este mesmo atributo (nome). Se tal ocorrer, serão obtidas respostas inesperadas, nomeadamente, não relacionadas com o Joaquim José em questão.

Álgebra Relacional

$$\begin{aligned}
 amigos_joaquim &\leftarrow \pi_{\text{membro2 as membro}} \left(\sigma_{\substack{\text{membro1} = \text{Idmembro} \\ \text{Nome} = \text{"Joaquim José"}}} (amigo \times membro) \right) \\
 doce_amigos &\leftarrow \pi_{\text{doce.nome as Doce}} \left(\sigma_{\substack{\text{doce.género} = \text{"Tradicional"} \\ \text{doce.nome} = \text{criou.doce}}} ((criou \bowtie amigos_joaquim) \times doce) \right) \\
 \pi_{\text{Doce}} (\sigma_{\text{aspeto} \geq 5} (doce_amigos \bowtie fez))
 \end{aligned}$$

PostgreSQL

```

1 CREATE VIEW amigos_joaquim AS (
2     SELECT membro2 AS membro
3     FROM amigo, membro
4     WHERE membro1 = idmembro AND
5           nome = 'Joaquim José');
6
7 WITH doce_amigos AS (
8     SELECT doce.nome AS doce
9     FROM (criou NATURAL INNER JOIN amigos_joaquim), doce
10    WHERE doce.genero = 'Tradicional' AND
11          doce.nome = criou.doce)
12
13 SELECT DISTINCT doce
14 FROM doce_amigos NATURAL INNER JOIN fez
15 WHERE aspeto >= 5;

```

4.c Qual é o nome dos membros que só fizeram doces dos seus amigos?

Álgebra Relacional

$$fc \leftarrow \pi_{\text{fazedor}, \text{criador}} (\sigma_{\text{fazedor.doce} = \text{criador.doce}} (\text{faz} \times \text{criou}))$$

$$\text{faz_amigos} \leftarrow \pi_{\text{fazedor}, \text{criador}} (\sigma_{\text{fazedor} = \text{membro1} \wedge \text{criador} = \text{membro2}} (\text{faz} \times \text{amigo}))$$

$$\text{faz_n_amigos} \leftarrow \pi_{\text{fazedor}, \text{criador}} (fc) - \pi_{\text{fazedor}, \text{criador}} (\text{faz_amigos})$$

$$\text{n_doces} \leftarrow \text{membro} \mathcal{G}_{\text{Count}(\text{doce})} \text{ as ndoces} (\text{faz})$$

$$\text{answer} \leftarrow \pi_{\text{idMembro}} \left(\sigma_{\text{idMembro} = \text{membro} \wedge \text{ndoces} > 0} (\text{membro} \times \text{n_doces}) \right) - \pi_{\text{fazedor}} (\text{faz_n_amigos})$$

$$\pi_{\text{nome}} (\text{answer} \bowtie \text{membro})$$

PostgreSQL

```

1 WITH doce_pessoas AS (
2     SELECT membro, COUNT(doce) AS ndoces
3     FROM fez
4     GROUP BY membro)
5
6 SELECT DISTINCT Nome
7 FROM membro AS m, doce_pessoas
8 WHERE idmembro = doce_pessoas.membro AND
9     ndoces > 0 AND
10    NOT EXISTS (SELECT DISTINCT doce
11                FROM fez
12                WHERE fez.membro = m.idmembro
13
14                EXCEPT
15
16                SELECT DISTINCT doce
17                FROM criou, amigo
18                WHERE amigo.membro1 = m.idmembro AND
19                criou.membro = amigo.membro2);

```

4.d Quais são os doces com Natas que não tiveram um 5 no sabor?

Esta questão pode ter várias interpretações diferentes que irão dar origem a diferentes expressões de álgebra relacional e a diferentes *queries* em SQL. Desta forma, foram exploradas duas abordagens que serão explanadas adiante.

Numa primeira abordagem, considerou-se que se estava à procura de qualquer doce que não tivesse sido avaliado com 5 no sabor por todos os utilizadores que o fizeram. Desta forma, bastava que apenas um dos utilizadores que fez um determinado doce não o tivesse avaliado com 5 para que este doce fosse resultado da *query*.

Álgebra Relacional

$$\pi_{Doce}(\sigma_{sabor < 5}(faz \bowtie temIngrediente))$$

PostgreSQL

```
1 | SELECT DISTINCT doce
2 | FROM fez NATURAL INNER JOIN temingrediente
3 | WHERE sabor < 5 AND
4 |     ingrediente = 'Natas'
```

Para a segunda abordagem, considerou-se que se estava à procura de qualquer doce que não tivesse sido avaliado com 5 no sabor por nenhum dos utilizadores que o fizeram. Desta forma, bastava que um dos utilizadores que fez um determinado doce o tivesse avaliado com 5 para que este doce não fosse resultado da *query*.

Álgebra Relacional

$$\pi_{Doce}(\sigma_{ingrediente = \text{"Natas"}}(temIngrediente)) - \pi_{Doce}(\sigma_{Sabor = 5}(faz))$$

PostgreSQL

```
1 | SELECT DISTINCT doce
2 | FROM temingrediente
3 | WHERE ingrediente = 'Natas'
4 |
5 | EXCEPT
6 |
7 | SELECT DISTINCT doce
8 | FROM fez
9 | WHERE sabor = 5
```

4.e Quais são os membros que criaram doces com Canela e Leite?

Álgebra Relacional

$$\pi_{\text{membro}} \left(\sigma_{\begin{array}{c} ti1.doce = ti2.doce \\ \wedge \\ ti1.doce = criou.doce \\ \wedge \\ ti1.ingrediente = "Canela" \\ \wedge \\ ti2.ingrediente = "Leite" \end{array}} (criou \times temIngrediente \text{ as } ti1 \times temIngrediente \text{ as } ti2) \right)$$

PostgreSQL

```

1 | SELECT DISTINCT membro
2 | FROM criou, temingrediente AS ti1, temingrediente AS ti2
3 | WHERE ti1.doce = ti2.doce AND
4 |       ti1.doce = criou.doce AND
5 |       ti1.ingrediente = 'Canela' AND
6 |       ti2.ingrediente = 'Leite';

```

4.f Quais são os doces que têm Maçã ou Baunilha?

Álgebra Relacional

$$\pi_{\text{Doce}} (\sigma_{\text{Ingrediente} = "Maçã" \vee \text{Ingrediente} = "Baunilha"} (temIngrediente))$$

PostgreSQL

```

1 | SELECT DISTINCT doce
2 | FROM temingrediente
3 | WHERE ingrediente = 'Maçã' OR
4 |       ingrediente = 'Baunilha';

```

4.g Qual é o custo do Arroz Doce?

Álgebra Relacional

$$custo_quantidade \leftarrow \pi_{custo, doce, \frac{(\sigma_{nome=ingrediente} (ingrediente \times temIngrediente))}{quantidade}}$$
$$doce_custo \leftarrow Doce \mathcal{G}_{sum(custo * quantidade)} as custo (custo_quantidade)$$
$$\pi_{custo} (\sigma_{Nome = "Arroz Doce"} (doce_custo))$$

PostgreSQL

```
1 CREATE VIEW doce_custo AS(  
2     SELECT temIngrediente.doce,  
3           SUM(ingrediente.custo * temIngrediente.quantidade) AS custo  
4     FROM ingrediente JOIN temIngrediente ON  
5           ingrediente.nome = temIngrediente.ingrediente  
6     GROUP BY temingrediente.doce);  
7  
8 SELECT custo  
9 FROM doce_custo  
10 WHERE doce = 'Arroz Doce';
```

4.h Indique o custo de cada doce

Álgebra Relacional

$$custo_quantidade \leftarrow \pi_{custo, doce, \frac{(\sigma_{nome=ingrediente} (ingrediente \times temIngrediente))}{quantidade}}$$
$$doce_cust \leftarrow Doce \mathcal{G}_{sum(custo * quantidade)} as custo (custo_quantidade)$$
$$\pi_{Doce, Custo} (doce_custo)$$

PostgreSQL

```
1 SELECT *  
2 FROM doce_custo
```

*Foi utilizada a *view* "doce_custo" criada na alínea 4.(g)

4.i Para cada membro indique o número de doces que criou

Álgebra Relacional

$$no_doce_memb \leftarrow \text{membro} \mathcal{G}_{count(Doce)} \text{ as } no_doces (criou)$$
$$\pi_{membro, no_doces} (no_doce_memb)$$

PostgreSQL

```
1 CREATE VIEW no_doce_memb AS (  
2     SELECT membro, COUNT(doce) AS no_doces  
3     FROM criou  
4     GROUP BY membro);  
5  
6 SELECT *  
7 FROM no_doce_memb;
```

4.j Qual é o nome do membro que criou mais doces?

Álgebra Relacional

$$no_doce_memb \leftarrow membro \mathcal{G}_{count}(Doce) \text{ as } no_doces(criou)$$
$$memb_menos_doce \leftarrow \pi_{doces1.membro} \left(\sigma_{doces1.no_doces < doces2.no_doces} \left(\begin{matrix} no_doce_memb \text{ as } doces1 \\ \times \\ no_doce_memb \text{ as } doces2 \end{matrix} \right) \right)$$
$$answer_id \leftarrow \pi_{membro} (no_doce_memb) - \pi_{membro} (memb_menos_doce)$$
$$\pi_{nome} (\sigma_{idmembro = answer_id.membro} (answer_id \times membro))$$

PostgreSQL

```
1 WITH memb_menos_doce AS (  
2     SELECT doces1.membro  
3     FROM no_doce_memb AS doces1, no_doce_memb AS doces2  
4     WHERE doces1.no_doces < doces2.no_doces),  
5  
6     answer_id AS (  
7         SELECT membro  
8         FROM no_doce_memb  
9  
10        EXCEPT  
11  
12        SELECT membro  
13        FROM memb_menos_doce)  
14  
15  
16 SELECT nome  
17 FROM answer_id, membro  
18 WHERE idmembro = answer_id.membro;
```

*Foi utilizada a *view* "no_memb_doce" criada na alínea 4.(i)

4.k Qual é o nome dos membros que fizeram o doce mais caro?

Álgebra Relacional

$$custo_quantidade \leftarrow \pi_{custo, doce, \frac{(\sigma_{nome=ingrediente} (ingrediente \times temIngrediente))}{quantidade}}$$
$$doce_custo \leftarrow Doce \mathcal{G}_{sum(custo * quantidade)} as\ custo\ (custo_quantidade)$$
$$menos_caros \leftarrow \pi_{dc1.doce} (\sigma_{dc1.custo < dc2.custo} (doce_custo\ as\ dc1 \times doce_custo\ as\ dc2))$$
$$mais_caro \leftarrow \pi_{doce}(doce_custo) - \pi_{doce}(menos_caros)$$
$$\pi_{nome} (\sigma_{faz.doce = mais_caro.doce \wedge idmembro = fez.membro} (membro \times fez \times mais_caro))$$

PostgreSQL

```
1 WITH menos_caros AS (  
2     SELECT dc1.doce  
3     FROM doce_custo AS dc1, doce_custo AS dc2  
4     WHERE dc1.custo < dc2.custo),  
5  
6     mais_caro AS(  
7         SELECT doce  
8         FROM doce_custo  
9  
10        EXCEPT  
11  
12        SELECT doce  
13        FROM menos_caros)  
14  
15 SELECT nome  
16 FROM membro, fez, mais_caro  
17 WHERE fez.doce = mais_caro.doce AND  
18        idmembro = fez.membro;
```

*Foi utilizada a *view* "doce_custo" criada na alínea 4.(g)

4.1 Quais são os membros que são amigos de todos os amigos do Joaquim José

De forma a responder a esta questão, é necessário analisar de uma forma mais cuidada a relação "amigo". Esta relação representa a amizade entre dois membros da rede social e, assim sendo, é possível inferir que a amizade tem reciprocidade, ou seja, se Joaquim é amigo de alguém, também esse membro terá de ser amigo de Joaquim. Este facto vai de encontro à forma, sugerida no enunciado, de representar os tuplos simétricos das várias amizades entre membros.

Assim, se um membro for amigo de Joaquim, mas também amigo de todos os seus outros amigos, e de forma a garantir que ele seja um resultado a esta questão, seria necessário considerar que os vários membros da rede social seriam amigos de si próprios, o que não faz sentido. Outra solução para esta situação, seria garantir que o membro cujas amizades se estão a verificar não é considerado quando é feita a comparação com a tabela dos amigos de Joaquim e, desta forma, não seria necessário que este membro fosse amigo de si próprio.

Álgebra Relacional

Em álgebra relacional, quando se realiza a operação de divisão, não é possível excluir o membro cujas amizades se estão a averiguar do conjunto de amizades do Joaquim, pelo que se considerou que a relação "amigo" contém a representação amigo de si próprio.

$$amigos_joaquim \leftarrow \pi_{\text{membro2 as membro}} \left(\sigma_{\substack{\text{membro1} = Id_{\text{membro}} \\ \text{Nome} = \text{"Joaquim José"}}} (amigo \times membro) \right)$$

$$r \leftarrow \pi_{\text{membro1, membro2}} (amigo)$$

$$s \leftarrow \pi_{amigos_joaquim.membro \text{ as } membro2} (amigos_joaquim)$$

$$r \div s$$

PostgreSQL

Para fazer a *query* em PostgreSQL, não se considerou o membro cujas amizades se estão a verificar na comparação com os amigos de Joaquim.

```
1 SELECT DISTINCT membro1
2 FROM amigo AS am
3 WHERE NOT EXISTS (SELECT DISTINCT membro
4                   FROM amigos_joaquim
5                   WHERE am.membro1 != membro
6
7                   EXCEPT
8
9                   SELECT DISTINCT membro2
10                  FROM amigo
11                  WHERE am.membro1 = amigo.membro1);
```

*Foi utilizada a *view* "amigos_joaquim" criada na alínea 4.(b)

4.m Quais são os membros que fizeram todos os doces com Baunilha?

Álgebra Relacional

$$r \leftarrow \pi_{\text{membro, doce}}(f\acute{e}z)$$
$$s \leftarrow \pi_{\text{doce}}(\sigma_{\text{Ingrediente}=\text{"Baunilha"}}(temIngrediente))$$
$$r \div s$$

PostgreSQL

```
1 SELECT DISTINCT membro
2 FROM fez AS f
3 WHERE NOT EXISTS (SELECT DISTINCT doce
4                   FROM temingrediente
5                   WHERE ingrediente = 'Baunilha'
6
7                   EXCEPT
8
9                   SELECT DISTINCT doce
10                  FROM fez
11                  WHERE f.membro = fez.membro);
```

4.n Quais são os doces que quando foram feitos tiveram sempre um 1 no tempo?

Álgebra Relacional

$$avg_time \leftarrow Doce \mathcal{G}_{avg(Tempo)} as\ avg_t (faz)$$
$$\pi_{Doce} (\sigma_{avg_t = 1} (avg_time))$$

PostgreSQL

```
1 CREATE VIEW avg_time AS(  
2     SELECT doce, AVG(tempo) AS avg_t  
3     FROM fez  
4     GROUP BY doce);  
5  
6 SELECT doce  
7 FROM avg_time  
8 WHERE avg_t = 1;
```

4.o Quais são os doces mais baratos e mais rápidos de fazer?

Na tabela "fez", assumiu-se que quanto mais baixo o valor no atributo "Tempo" mais rápida seria a confeção do doce. Assim sendo, um valor 1 em "Tempo" corresponde ao mais rápido e um valor 5 ao mais lento.

Álgebra Relacional

$$avg_time \leftarrow Doce \mathcal{G}_{avg(Tempo)} as\ avg_t\ (fez)$$

$$menos_baratos \leftarrow \pi_{dc1.doce} (\sigma_{dc1.custo > dc2.custo} (doce_custo\ as\ dc1 \times doce_custo\ as\ dc2))$$

$$mais_baratos \leftarrow \pi_{nome} (doce) - \pi_{doce} (menos_baratos)$$

$$menos_rapidos \leftarrow \pi_{t1.doce} (\sigma_{t1.avg_t > t2.avg_t} (avg_time\ as\ t1 \times avg_time\ as\ t2))$$

$$mais_rapidos \leftarrow \pi_{doce} (fez) - \pi_{doce} (menos_rapidos)$$

$$\pi_{doce}(mais_baratos) \cap \pi_{doce}(mais_rapidos)$$

PostgreSQL

```
1 WITH menos_rapidos AS(
2     SELECT DISTINCT t1.doce
3     FROM avg_time AS t1, avg_time AS t2
4     WHERE t1.avg_t > t2.avg_t),
5
6 mais_rapidos AS(
7
8     SELECT doce
9     FROM fez
10
11     EXCEPT
12
13     SELECT doce
14     FROM menos_rapidos),
15
16 menos_baratos AS(
17     SELECT DISTINCT dc1.doce
18     FROM doce_custo AS dc1, doce_custo AS dc2
19     WHERE dc1.custo > dc2.custo),
20
21 mais_baratos AS(
22     SELECT doce
23     FROM fez
24
25     EXCEPT
26
27     SELECT doce
28     FROM menos_baratos)
29
30 SELECT doce
31 FROM mais_baratos
32
33 INTERSECT
34
35 SELECT doce
36 FROM mais_rapidos;
```

*Foram utilizadas as *views* "doce_custo" criada na alínea 4.(g) e "avg_time" criada na alínea 4.(n)