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

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

```
CREATE TABLE membro (
    Nome TEXT NOT NULL,
    IdMembro VARCHAR(100) NOT NULL UNIQUE ,
    Pais VARCHAR(50) NOT NULL,
    DataNasc DATE NOT NULL,
    PRIMARY KEY (IdMembro)
);
```

2.2 Doce

```
CREATE TABLE doce (
    Nome VARCHAR(50) NOT NULL UNIQUE,
    Descricao TEXT NOT NULL UNIQUE,
    Genero VARCHAR(50) NOT NULL,
    PRIMARY KEY (Nome)
);
```

2.3 Amigo

```
CREATE TABLE amigo (
    Membro1 VARCHAR(100) NOT NULL,
    Membro2 VARCHAR(100) NOT NULL,
    PRIMARY KEY (Membro1, Membro2)
);
```

2.4 Criou

```
CREATE TABLE criou(
    Membro VARCHAR(100) NOT NULL,
    Doce VARCHAR(50) NOT NULL,
    PRIMARY KEY (Membro, Doce)
);
```

2.5 Fez

```
CREATE TABLE fez (
    Membro VARCHAR(100) NOT NULL,
    Doce VARCHAR(50) NOT NULL,
    Tempo SMALLINT NOT NULL,
    Aspeto SMALLINT NOT NULL,
    Sabor SMALLINT NOT NULL,
    PRIMARY KEY (Membro, Doce)
);
```

2.6 Ingrediente

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

2.7 temIngrediente

```
CREATE TABLE temIngrediente (
    Doce VARCHAR(50) NOT NULL,
    Ingrediente VARCHAR(50) NOT NULL,
    Quantidade INT NOT NULL,
    PRIMARY KEY (Doce, Ingrediente)
);
```

2.8 Chaves Estrangeiras

2.8.1 Amigo

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

2.8.2 Criou

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

2.8.3 Fez

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

2.8.4 temIngrediente

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

3 Inserção de Dados

3.a Inserir na relação membro

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

3.b Inserir nas relações ingrediente e temIngrediente

```
INSERT INTO ingrediente (Nome, Custo) VALUES
    ('Açúcar',      0.00109),
    ('Farinha',     0.00069),
    ('Chocolate',   0.00995),
    ('Ovos',        0.2),
    ('Natas',       0.00375),
    ('Leite',       0.00082),
    ('Água',        0.00014),
    ('Maçã',        0.00229),
    ('Manteiga',    0.00672),
    ('Alfarroba',   0.01316),
    ('Baunilha',    3.495),
    ('Canela',      0.4975),
    ('Pimenta',     0.37556),
    ('Café',        0.01845),
    ('Arroz',       0.00119),
    ('Limão',       0.00199),
    ('Bolacha',     0.00211),
    ('Amêndoa',     0.01495);
COMMIT;
```

```sql
INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
    -- Arroz Doce
    ('Arroz Doce',                  'Arroz',     300),
    ('Arroz Doce',                  'Açúcar',    300),
    ('Arroz Doce',                  'Leite',     600),
    ('Arroz Doce',                  'Canela',    7),
    ('Arroz Doce',                  'Baunilha',  7),
    ('Arroz Doce',                  'Limão',     1),

    -- Mousse de Chocolate
    ('Mousse de Chocolate',         'Chocolate', 200),
    ('Mousse de Chocolate',         'Água',      200),

    --Bolo de Chocolate e Canela
    ('Bolo de Chocolate e Canela',  'Canela',    30),
    ('Bolo de Chocolate e Canela',  'Açúcar',    150),
    ('Bolo de Chocolate e Canela',  'Farinha',   600),
    ('Bolo de Chocolate e Canela',  'Ovos',      4),
    ('Bolo de Chocolate e Canela',  'Leite',     900),
    ('Bolo de Chocolate e Canela',  'Manteiga',  40),
    ('Bolo de Chocolate e Canela',  'Chocolate', 150),

    --Sericaia
    ('Sericaia',                    'Leite',     500),
    ('Sericaia',                    'Açúcar',    170),
    ('Sericaia',                    'Ovos',      6),
    ('Sericaia',                    'Canela',    30),
    ('Sericaia',                    'Limão',     1),

    --Pudim de Abade de Priscos
    ('Pudim Abade de Priscos',      'Ovos',      15),
    ('Pudim Abade de Priscos',      'Açúcar',    500),
    ('Pudim Abade de Priscos',      'Água',      650),
    ('Pudim Abade de Priscos',      'Canela',    30),

    --Bolo de Bolacha
    ('Bolo de Bolacha',             'Manteiga',  250),
    ('Bolo de Bolacha',             'Bolacha',   400),
    ('Bolo de Bolacha',             'Açúcar',    200),
    ('Bolo de Bolacha',             'Ovos',      3),
    ('Bolo de Bolacha',             'Café',      90),
```

```
42      ('Bolo de Bolacha',             'Baunilha',  7),
43
44      --Pão de Rala
45      ('Pão de Rala',                 'Ovos',      20),
46      ('Pão de Rala',                 'Açúcar',    700),
47      ('Pão de Rala',                 'Amêndoa',   500),
48
49      --Tarte de Alfarroba
50      ('Tarte de Alfarroba',          'Açúcar',    150),
51      ('Tarte de Alfarroba',          'Ovos',      6),
52      ('Tarte de Alfarroba',          'Amêndoa',   100),
53      ('Tarte de Alfarroba',          'Alfarroba', 50),
54
55      --Ovos Moles de Aveiro
56      ('Ovos Moles de Aveiro',        'Ovos',      8),
57      ('Ovos Moles de Aveiro',        'Açúcar',    150),
58      ('Ovos Moles de Aveiro',        'Água',      150),
59
60      --Tarte de Natas
61      ('Tarte de Natas',              'Natas',     600),
62      ('Tarte de Natas',              'Ovos',      6),
63      ('Tarte de Natas',              'Açúcar',    200),
64      ('Tarte de Natas',              'Bolacha',   300),
65
66      --Tarte de Maça
67      ('Tarte de Maçã',               'Farinha',   80),
68      ('Tarte de Maçã',               'Açúcar',    200),
69      ('Tarte de Maçã',               'Leite',     250),
70      ('Tarte de Maçã',               'Água',      250),
71      ('Tarte de Maçã',               'Ovos',      4),
72      ('Tarte de Maçã',               'Maçã',      600);
73
74  COMMIT;
```

3.c Inserir nas Relações criou, doce e fez

```
INSERT INTO criou (Membro, Doce) VALUES
    ('marialourdesmodesto',    'Arroz Doce'),
    ('hestonblumenthal',       'Mousse de Chocolate'),
    ('ferranadria',            'Bolo de Chocolate e Canela'),
    ('pierreherme',            'Sericaia'),
    ('marialourdesmodesto',    'Pudim Abade de Priscos'),
    ('elizabethfalkner',       'Bolo de Bolacha'),
    ('hestonblumenthal',       'Pão de Rala'),
    ('marialourdesmodesto',    'Tarte de Alfarroba'),
    ('alexatala',              'Ovos Moles de Aveiro'),
    ('ferranadria',            'Tarte de Natas'),
    ('pierreherme',            'Tarte de Maçã');

COMMIT;
```

```
INSERT INTO doce (Nome, Descricao, Genero) VALUES
    ('Arroz Doce','Arroz cozido em leite com açúcar.','Tradicional'),
    ('Mousse de Chocolate','Mousse de chocolate com água.',
        'Tradicional'),
    ('Bolo de Chocolate e Canela','Simples Bolo com chocolate e
        canela.','Regional'),
    ('Sericaia','Doce conventual alentejano cozido num prato de
        barro.','Regional'),
    ('Pudim Abade de Priscos','Pudim de ovos com toucinho.','Regional'),
    ('Bolo de Bolacha','Bolachas embebidas em café e
        natas.','Tradicional'),
    ('Pão de Rala','Doce conventual alentejano com amêndoa e
        ovos.','Regional'),
    ('Tarte de Alfarroba','Tarte com farinha de alfarroba.','Regional'),
    ('Ovos Moles de Aveiro','Doce conventual aveirense com ovos e
        açúcar.','Regional'),
    ('Tarte de Natas','Tarte com natas e açúcar.','Tradicional'),
    ('Tarte de Maçã','Tarte com maça e canela.','Tradicional');

COMMIT;
```

```
INSERT INTO fez (Membro, Doce, Tempo, Aspeto, Sabor) VALUES
    ('marialourdesmodesto',    'Arroz Doce',                4, 2, 4),
    ('hestonblumenthal',       'Arroz Doce',                3, 2, 5),

    ('hestonblumenthal',       'Mousse de Chocolate',       1, 3, 4),
    ('pierreherme',            'Mousse de Chocolate',       1, 5, 4),
    ('minekokato',             'Mousse de Chocolate',       3, 1, 2),

    ('ferranadria',            'Bolo de Chocolate e Canela', 5, 3, 4),
    ('alexatala',              'Bolo de Chocolate e Canela', 2, 2, 3),

    ('pierreherme',            'Sericaia',                  1, 2, 4),
    ('alexatala',              'Sericaia',                  2, 1, 1),

    ('marialourdesmodesto',    'Pudim Abade de Priscos',    1, 5, 5),
    ('joaquimjose',            'Pudim Abade de Priscos',    2, 1, 4),

    ('elizabethfalkner',       'Bolo de Bolacha',           1, 4, 5),
    ('lorrainepascale',        'Bolo de Bolacha',           2, 5, 5),
    ('hestonblumenthal',       'Bolo de Bolacha',           2, 5, 5),

    ('minekokato',             'Pão de Rala',               4, 1, 1),
    ('lorrainepascale',        'Pão de Rala',               4, 2, 2),
    ('hestonblumenthal',       'Pão de Rala',               3, 3, 3),

    ('marialourdesmodesto',    'Tarte de Alfarroba',        2, 5, 5),
    ('ferranadria',            'Tarte de Alfarroba',        1, 1, 1),

    ('alexatala',              'Ovos Moles de Aveiro',      1, 5, 5),
    ('joaquimjose',            'Ovos Moles de Aveiro',      1, 4, 2),

    ('ferranadria',            'Tarte de Natas',            2, 1, 1),
    ('joaquimjose',            'Tarte de Natas',            1, 5, 4),

    ('pierreherme',            'Tarte de Maçã',             5, 4, 4),
    ('marialourdesmodesto',    'Tarte de Maçã',             5, 5, 5);

COMMIT;
```

3.d Inserir na Relação amigo

```
INSERT INTO amigo (Membro1, Membro2) VALUES
    ('joaquimjose',         'marialourdesmodesto'),
    ('joaquimjose',         'alexatala'),
    ('joaquimjose',         'ferranadria'),
    ('joaquimjose',         'pierreherme'),
    ('joaquimjose',         'elizabethfalkner'),
    ('joaquimjose',         'lorrainepascale'),
    ('joaquimjose',         'gastonlenotre'),
    ('joaquimjose',         'hestonblumenthal'),
    ('joaquimjose',         'minekokato'),
    ('marialourdesmodesto', 'joaquimjose'),
    ('alexatala',           'joaquimjose'),
    ('ferranadria',         'joaquimjose'),
    ('pierreherme',         'joaquimjose'),
    ('elizabethfalkner',    'joaquimjose'),
    ('lorrainepascale',     'joaquimjose'),
    ('gastonlenotre',       'joaquimjose'),
    ('hestonblumenthal',    'joaquimjose'),
    ('minekokato',          'joaquimjose'),

    ('hestonblumenthal',    'gastonlenotre'),
    ('hestonblumenthal',    'minekokato'),
    ('gastonlenotre',       'hestonblumenthal'),
    ('minekokato',          'hestonblumenthal'),

    ('alexatala',           'lorrainepascale'),
    ('lorrainepascale',     'alexatala'),

    ('ferranadria',         'elizabethfalkner'),
    ('ferranadria',         'alexatala'),
    ('elizabethfalkner',    'ferranadria'),
    ('alexatala',           'ferranadria'),

    ('marialourdesmodesto', 'lorrainepascale'),
    ('marialourdesmodesto', 'elizabethfalkner'),
    ('lorrainepascale',     'marialourdesmodesto'),
    ('elizabethfalkner',    'marialourdesmodesto'),

```

```
40      ('gastonlenotre',       'minekokato'),
41      ('gastonlenotre',       'marialourdesmodesto'),
42      ('gastonlenotre',       'ferranadria'),
43      ('gastonlenotre',       'pierreherme'),
44      ('gastonlenotre',       'elizabethfalkner'),
45      ('gastonlenotre',       'lorrainepascale'),
46      ('gastonlenotre',       'alexatala'),
47      ('minekokato',          'gastonlenotre'),
48      ('marialourdesmodesto', 'gastonlenotre'),
49      ('ferranadria',         'gastonlenotre'),
50      ('pierreherme',         'gastonlenotre'),
51      ('elizabethfalkner',    'gastonlenotre'),
52      ('lorrainepascale',     '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

$$sp\_cria\_regional \leftarrow \pi_{\substack{\text{membro.Nome,}\\ \text{criou.Doce}}} \left(\sigma_{\substack{\text{País} = \text{“Espanha“} \\ \wedge \\ \text{membro.IdMembro} = \text{criou.membro} \\ \wedge \\ \text{doce.género} = \text{“Regional“} \\ \wedge \\ \text{doce.nome} = \text{criou.doce}}} (membro \times criou \times doce) \right)$$

$$choc\_canela \leftarrow \pi_{tI1.Doce} \left(\sigma_{\substack{\text{tI1.Doce} = \text{tI2.Doce} \\ \wedge \\ \text{tI1.Ingrediente} = \text{“Canela“} \\ \wedge \\ \text{tI2.Ingrediente} = \text{“Chocolate“}}} (\text{temIngrediente as tI1} \times \text{temIngrediente as tI2}) \right)$$

$$\pi_{nome} \left(sp\_cria\_regional \bowtie choc\_canela \right)$$

PostgreSQL

```sql
WITH sp_cria_regional AS (
  SELECT membro.nome, criou.doce
  FROM membro, criou, doce
  WHERE membro.pais = 'Espanha' AND
        membro.idmembro = criou.membro AND
        doce.nome = criou.doce AND
        doce.genero = 'Regional'),

  choc_canela AS (
    SELECT tI1.doce
    FROM temingrediente AS tI1, temingrediente AS tI2
    WHERE tI1.doce = tI2.doce AND
          tI1.ingrediente = 'Canela' AND
          tI2.ingrediente = 'Chocolate')

SELECT DISTINCT sp_cria_regional.nome
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

$$amigos\_joaquim \leftarrow \pi_{\text{membro2 as membro}} \left(\sigma_{\substack{membro1 = Idmembro \\ \wedge \\ Nome = \text{“}Joaquim\ José\text{“}}} (amigo \times membro) \right)$$

$$doce\_amigos \leftarrow \pi_{\text{doce.nome as Doce}} \left(\sigma_{\substack{doce.género = \text{“}Tradicional\text{“} \\ \wedge \\ doce.nome = criou.doce}} ((criou \bowtie amigos\_joaquim) \times doce) \right)$$

$$\pi_{\text{Doce}} \left(\sigma_{aspeto \geq 5} \left(doce\_amigos \bowtie fez \right) \right)$$

PostgreSQL

```sql
CREATE VIEW amigos_joaquim AS (
    SELECT membro2 AS membro
    FROM amigo, membro
    WHERE membro1 = idmembro AND
          nome = 'Joaquim José');

WITH doce_amigos AS (
    SELECT doce.nome AS doce
    FROM (criou NATURAL INNER JOIN amigos_joaquim), doce
    WHERE doce.genero = 'Tradicional' AND
          doce.nome = criou.doce)

SELECT DISTINCT doce
FROM doce_amigos NATURAL INNER JOIN fez
WHERE aspeto >= 5;
```

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

Álgebra Relacional

$$fc \leftarrow \pi_{fez.membro\ as\ fazedor,\ criou.membro\ as\ criador}\ (\sigma_{fez.doce\ =\ criou.doce}\ (fez\ \times\ criou))$$

$$fez\_amigos \leftarrow \pi_{\text{fazedor, criador}}\ (\sigma_{\text{fc.fazedor = membro1}\ \wedge\ \text{fc.criador = membro2}}(fc\ \times\ amigo))$$

$$fez\_n\_amigos \leftarrow \pi_{fazedor,\ criador}\ (fc)\ -\ \pi_{fazedor,\ criador}\ (fez\_amigos)$$

$$n\_doces \leftarrow {}_{membro}\mathcal{G}_{Count(doce)\ as\ ndoces}\ (fez)$$

$$answer \leftarrow \pi_{idMembro}\left(\sigma_{\substack{idMembro\ =\ membro \\ \wedge \\ ndoces\ >\ 0}}\ (membro \times n\_doces)\right) - \pi_{fazedor}\ (fez\_n\_amigos)$$

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

PostgreSQL

```sql
WITH doce_pessoas AS (
   SELECT membro, COUNT(doce) AS ndoces
   FROM fez
   GROUP BY membro)

SELECT DISTINCT Nome
FROM membro AS m, doce_pessoas
WHERE idmembro = doce_pessoas.membro AND
      ndoces > 0 AND
      NOT EXISTS (SELECT DISTINCT doce
                  FROM fez
                  WHERE fez.membro = m.idmembro

                  EXCEPT

                  SELECT DISTINCT doce
                  FROM criou, amigo
                  WHERE amigo.membro1 = m.idmembro AND
                        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

$$\pi_{Doce}\left(\sigma_{sabor\ <\ 5}\ (fez\ \bowtie\ temIngrediente)\right)$$

PostgreSQL

```
SELECT DISTINCT doce
FROM fez NATURAL INNER JOIN temingrediente
WHERE sabor < 5 AND
      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

$$\pi_{Doce}\left(\sigma_{ingrediente\ =\ \text{“}Natas\text{“}}\ (temIngrediente)\right)\ -\ \pi_{Doce}\left(\sigma_{Sabor\ =\ 5}\ (fez)\right)$$

PostgreSQL

```
SELECT DISTINCT doce
FROM temingrediente
WHERE ingrediente = 'Natas'

EXCEPT

SELECT DISTINCT doce
FROM fez
WHERE sabor = 5
```

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

Álgebra Relacional

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

PostgreSQL

```sql
SELECT DISTINCT membro
FROM criou, temingrediente AS ti1, temingrediente AS ti2
WHERE ti1.doce = ti2.doce AND
      ti1.doce = criou.doce AND
      ti1.ingrediente = 'Canela' AND
      ti2.ingrediente = 'Leite';
```

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

Álgebra Relacional

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

PostgreSQL

```sql
SELECT DISTINCT doce
FROM temingrediente
WHERE ingrediente = 'Maçã' OR
      ingrediente = 'Baunilha';
```

4.g Qual é o custo do Arroz Doce?

Álgebra Relacional

$$custo\_quantidade \leftarrow \pi_{\substack{custo,\ doce,\\ quantidade}}(\sigma_{nome=ingrediente}\ (ingrediente\ \times\ temIngrediente))$$

$$doce\_custo \leftarrow {}_{Doce}\mathcal{G}_{sum(custo\ *\ quantidade)\ as\ custo}\ (custo\_quantidade)$$

$$\pi_{\text{custo}}\ (\sigma_{Nome\ =\ \text{“}Arroz\ Doce\text{“}}\ (doce\_custo))$$

PostgreSQL

```sql
CREATE VIEW doce_custo AS(
    SELECT temIngrediente.doce,
           SUM(ingrediente.custo * temIngrediente.quantidade) AS custo
    FROM ingrediente JOIN temIngrediente ON
         ingrediente.nome = temIngrediente.ingrediente
    GROUP BY temingrediente.doce);

SELECT custo
FROM doce_custo
WHERE doce = 'Arroz Doce';
```

4.h Indique o custo de cada doce

Álgebra Relacional

$$custo\_quantidade \leftarrow \pi_{\substack{custo,\ doce,\\ quantidade}}(\sigma_{nome=ingrediente}\ (ingrediente\ \times\ temIngrediente))$$

$$doce\_cust \leftarrow {}_{Doce}\mathcal{G}_{sum(custo\ *\ quantidade)\ as\ custo}\ (custo\_quantidade)$$

$$\pi_{Doce,\ Custo}\ (doce\_custo)$$

PostgreSQL

```sql
SELECT *
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

$$no\_doce\_memb \leftarrow {}_{membro}\mathcal{G}_{count(Doce)\ as\ no\_doces}\ (criou)$$

$$\pi_{membro,\ no\_doces}\ (no\_doce\_memb)$$

PostgreSQL

```sql
CREATE VIEW no_doce_memb AS (
    SELECT membro, COUNT(doce) AS no_doces
    FROM criou
    GROUP BY membro);

SELECT *
FROM no_doce_memb;
```

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

Álgebra Relacional

$$no\_doce\_memb \leftarrow {}_{membro}\mathcal{G}_{count(Doce)\ as\ no\_doces}(criou)$$

$$memb\_menos\_doce \leftarrow \pi_{\text{doces1.membro}}\left(\sigma_{\text{doces1.no\_doces} < \text{doces2.no\_doces}}\begin{pmatrix} no\_doce\_memb\ as\ doces1 \\ \times \\ no\_doce\_memb\ as\ doces2 \end{pmatrix}\right)$$

$$answer\_id \leftarrow \pi_{membro}\ (no\_doce\_memb)\ -\ \pi_{membro}\ (memb\_menos\_doce)$$

$$\pi_{nome}\ (\sigma_{idmembro\ =\ answer\_id.membro}\ (answer\_id \times membro))$$

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

$$custo\_quantidade \leftarrow \pi_{\substack{custo,\ doce,\\ quantidade}}(\sigma_{nome=ingrediente}\ (ingrediente\ \times\ temIngrediente))$$

$$doce\_custo \leftarrow\ _{Doce}\mathcal{G}_{sum(custo\ *\ quantidade)\ as\ custo}\ (custo\_quantidade)$$

$$menos\_caros \leftarrow \pi_{\text{dc1.doce}}\ (\sigma_{\text{dc1.custo} < \text{dc2.custo}}\ (doce\_custo\ as\ dc1 \times doce\_custo\ as\ dc2))$$

$$mais\_caro \leftarrow \pi_{doce}(doce\_custo)\ -\ \pi_{doce}(menos\_caros)$$

$$\pi_{\text{nome}}\left(\sigma_{fez.doce\ =\ mais\_caro.doce\ \wedge\ idmembro\ =\ fez.membro}\ (membro \times fez \times mais\_caro)\right)$$

PostgreSQL

```sql
WITH menos_caros AS (
    SELECT dc1.doce
    FROM doce_custo AS dc1, doce_custo AS dc2
    WHERE dc1.custo < dc2.custo),

  mais_caro AS(
    SELECT doce
    FROM doce_custo

    EXCEPT

    SELECT doce
    FROM menos_caros)

SELECT nome
FROM membro, fez, mais_caro
WHERE fez.doce = mais_caro.doce AND
      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.

$$amigos\_joaquim \leftarrow \pi_{\text{membro2 as membro}} \left(\sigma_{\substack{membro1 = Idmembro \\ \wedge \\ Nome = \text{“}Joaquim\ José\text{“}}} (amigo \times membro) \right)$$

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

$$s \leftarrow \pi_{amigos\_joaquim.membro\ as\ membro2}(amigos\_joaquim)$$

$$r \div s$$

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.

```
SELECT DISTINCT membro1
FROM amigo AS am
WHERE NOT EXISTS (SELECT DISTINCT membro
                  FROM amigos_joaquim
                  WHERE am.membro1 != membro

                  EXCEPT

                  SELECT DISTINCT membro2
                  FROM amigo
                  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

$r \leftarrow \pi_{\text{membro, doce}}(fez)$

$s \leftarrow \pi_{doce}(\sigma_{Ingrediente=\text{“}Baunilha\text{“}}(temIngrediente))$

$r \div s$

PostgreSQL

```
SELECT DISTINCT membro
FROM fez AS f
WHERE NOT EXISTS (SELECT DISTINCT doce
                  FROM temingrediente
                  WHERE ingrediente = 'Baunilha'

                  EXCEPT

                  SELECT DISTINCT doce
                  FROM fez
                  WHERE f.membro = fez.membro);
```

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

Álgebra Relacional

$$avg\_time \leftarrow {}_{Doce}\mathcal{G}_{avg(Tempo)\ as\ avg\_t}\ (fez)$$

$$\pi_{Doce}\ (\sigma_{avg\_t\ =\ 1}\ (avg\_time))$$

PostgreSQL

```sql
CREATE VIEW avg_time AS(
    SELECT doce, AVG(tempo) AS avg_t
    FROM fez
    GROUP BY doce);

SELECT doce
FROM avg_time
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

$$avg\_time \leftarrow {}_{Doce}\mathcal{G}_{avg(Tempo)\ as\ avg\_t}\ (fez)$$

$$menos\_baratos \leftarrow \pi_{\text{dc1.doce}}\ (\sigma_{\text{dc1.custo} > \text{dc2.custo}}\ (doce\_custo\ as\ dc1 \times doce\_custo\ as\ dc2))$$

$$mais\_baratos \leftarrow \pi_{nome}\ (doce)\ -\ \pi_{doce}\ (menos\_baratos)$$

$$menos\_rapidos \leftarrow \pi_{\text{t1.doce}}\ (\sigma_{\text{t1.avg\_t} > \text{t2.avg\_t}}\ (avg\_time\ as\ t1 \times avg\_time\ as\ t2))$$

$$mais\_rapidos \leftarrow \pi_{doce}\ (fez)\ -\ \pi_{doce}\ (menos\_rapidos)$$

$$\pi_{\text{doce}}(mais\_baratos)\ \cap\ \pi_{\text{doce}}(mais\_rapidos)$$

PostgreSQL

```sql
WITH menos_rapidos AS(
    SELECT DISTINCT t1.doce
    FROM avg_time AS t1, avg_time AS t2
    WHERE t1.avg_t > t2.avg_t),

mais_rapidos AS(

    SELECT doce
    FROM fez

    EXCEPT

    SELECT doce
    FROM menos_rapidos),

menos_baratos AS(
    SELECT DISTINCT dc1.doce
    FROM doce_custo AS dc1, doce_custo AS dc2
    WHERE dc1.custo > dc2.custo),

mais_baratos AS(
    SELECT doce
    FROM fez

    EXCEPT

    SELECT doce
    FROM menos_baratos)

SELECT doce
FROM mais_baratos

INTERSECT

SELECT doce
FROM mais_rapidos;
```

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