1º Trabalho de Base de Dados

Relatório

Licenciatura em Engenharia Informática

Ano Letivo 2024/2025

Trabalho realizado por:
André Gonçalves – 58392
André Zhan - 58762

Índice

Objetivo e Resolução

Pretende-se desenvolver uma base de dados para gerir uma rede social de *Fãs* de sobremesas. A base de dados tem as seguintes relações:

membro(Nome, IdMemb, Pais, DataNasc);
doce(Nome, Descrição, Género);
amigo(Membro1, Membro2);
criou(Membro, Doce);
fez(Membro, Doce, Tempo, Aspeto, Sabor);
ingrediente(Nome, Custo);
temIngrediente(Doce, Ingrediente, Quantidade).

Para realizar o trabalho começámos por determinar as super chaves minimais de cada relação para depois pudermos determinar as chaves candidatas, primárias e estrangeiras.
Após obtermos todas as chaves necessárias passámos à criação da base de dados em SQL.

Depois de criar a base de dados, inserimos a informação pedida no exercício 3 do trabalho.

Por fim, para resolver as alíneas do exercício 4, começámos por determinar as expressões em SQL, para verificar no PostGres se os resultados estariam corretos e, só depois, passámos as expressões correspondentes para Álgebra Relacional.

Exercício 1 - Chaves Candidatas, Primárias e Estrangeiras

Relações/Chaves	Candidatas	Primária	Estrangeiras
membro	IdMemb	IdMemb	-
doce	Nome, Descrição	Nome	-
amigo	(Membro1, Membro2)	(Membro1, Membro2)	Membro1, Membro2
criou	(Membro, Doce)	(Membro, Doce)	Membro, Doce
fez	(Membro, Doce)	(Membro, Doce)	Membro, Doce
ingrediente	Nome	Nome	-
temIngrediente	(Doce, Ingrediente)	(Doce, Ingrediente)	Doce, Ingrediente

Exercício 2 – Criação das tabelas em SQL

Tabela membro:

```
CREATE TABLE membro (
    Nome VARCHAR(50) NOT NULL,
    IdMemb VARCHAR(50) PRIMARY KEY,
    Pais VARCHAR(50) NOT NULL,
    DataNasc DATE NOT NULL
);
```

NOTA: Na relação membro, primeiramente pensámos em usar valores inteiros para o atributo IdMemb porém acabámos por usar strings para facilitar o resto do trabalho. Para identificar os membros usamos o seguinte padrão na string: “id[nome_membro]”.

Tabela doce:

```
CREATE TABLE doce (
    Nome VARCHAR(100) PRIMARY KEY,
    Descricao TEXT NOT NULL,
    Genero VARCHAR(50) NOT NULL
);
```

Tabela amigo:

```
CREATE TABLE amigo (
    Membro1 VARCHAR(50) NOT NULL,
    Membro2 VARCHAR(50) NOT NULL,
    PRIMARY KEY (Membro1, Membro2),
    FOREIGN KEY (Membro1) REFERENCES membro(IdMemb),
    FOREIGN KEY (Membro2) REFERENCES membro(IdMemb),
    CHECK (Membro1 <> Membro2)
);
```

Tabela criou:

```
CREATE TABLE criou (
    Membro VARCHAR(50) NOT NULL,
    Doce VARCHAR(100) NOT NULL,
    PRIMARY KEY (Membro, Doce),
    FOREIGN KEY (Membro) REFERENCES membro(IdMemb),
    FOREIGN KEY (Doce) REFERENCES doce(Nome)
);
```

Tabela fez:

```
CREATE TABLE fez (
    Membro VARCHAR(50) NOT NULL,
    Doce VARCHAR(100) NOT NULL,
    Tempo INT CHECK (Tempo BETWEEN 1 AND 5) NOT NULL,
    Aspeto INT CHECK (Aspeto BETWEEN 1 AND 5) NOT NULL,
    Sabor INT CHECK (Sabor BETWEEN 1 AND 5) NOT NULL,
    PRIMARY KEY (Membro, Doce),
    FOREIGN KEY (Membro) REFERENCES membro(IdMemb),
    FOREIGN KEY (Doce) REFERENCES doce(Nome)
);
```

Tabela ingrediente:

```
CREATE TABLE ingrediente (
    Nome VARCHAR(100) PRIMARY KEY,
    Custo DECIMAL(10, 2) NOT NULL
);
```

Tabela temIngrediente:

```
CREATE TABLE temIngrediente (
    Doce VARCHAR(100) NOT NULL,
    Ingrediente VARCHAR(100) NOT NULL,
    Quantidade INT NOT NULL,
    PRIMARY KEY (Doce, Ingrediente),
    FOREIGN KEY (Doce) REFERENCES doce(Nome),
    FOREIGN KEY (Ingrediente) REFERENCES ingrediente(Nome)
);
```

Exercício 3 – Expressões em SQL para a inserção de dados

a) Inserção dos 10 membros

```
INSERT INTO membro (Nome, IdMemb, Pais, DataNasc) VALUES
    ('Ana', 'idana', 'Portugal', '1992-04-10'),
    ('Bernardo', 'idbernardo', 'Portugal', '1983-07-23'),
    ('Carlota', 'idcarlota', 'Espanha', '1997-08-15'),
    ('Carlos', 'idcarlos', 'Portugal', '1989-11-27'),
    ('Ema', 'idema', 'Brasil', '1996-01-13'),
    ('Emanuel', 'idemanuel', 'Itália', '1988-03-19'),
    ('Gabriela', 'idgabriela', 'Portugal', '1997-04-15'),
    ('Francisco', 'idfrancisco', 'Alemanha', '2001-11-05'),
    ('Isabel', 'idisabel', 'Espanha', '2002-06-25'),
    ('Jonathan', 'idjonathan', 'Portugal', '1999-02-17');
```

b) Inserção dos 13 ingredientes na tabela ingrediente

```
INSERT INTO ingrediente (Nome, Custo) VALUES
    ('Açucar', 0.6),
    ('Farinha', 0.2),
    ('Chocolate', 1.1),
    ('Ovos', 0.2),
    ('Natas', 0.5),
    ('Leite', 0.8),
    ('Água', 0.1),
    ('Maçã', 0.7),
    ('Manteiga', 1.2),
```

('Alfarroba', 1.3),
('Baunilha', 1.6),
('Canela', 0.9),
('Pimenta', 0.7);

NOTA: Usámos todos menos a pimenta.

c) Inserção dos 10 doces

-- Inserção dos doces na tabela doce
INSERT INTO doce (Nome, Descricao, Genero) VALUES
('Pastel de Nata', 'Massa folhada com natas', 'Tradicional'),
('Tiramisu', 'Doce de café e rum', 'Regional'),
('Mousse de Chocolate', 'Mousse cremosa de chocolate', 'Regional'),
('Pudim de Caramelo', 'Pudim com caramelo', 'Tradicional'),
('Biscoitos', 'Biscoitos crocantes com pepitas de chocolate', 'Regional'),
('Tarte de Maçã', 'Tarte com maça caramelizada', 'Internacional'),
('Torta de Amêndoa', 'Torta de amêndoa', 'Doce de Camada'),
('Bolo de Chocolate', 'Bolo com chocolate', 'Regional'),
('Bolo de Alfarroba', 'Bolo com alfarroba e frutos secos', 'Regional'),
('Tarte de Limão', 'Tarte doce com limão fresco', 'Tradicional');

NOTA: De acordo com o enunciado, o atributo Descrição na tabela doce seria para introduzir uma receita do doce correspondente, mas para facilitar o trabalho decidimos pôr descrições simples de cada doce.

-- Inserção dos doces na tabela criou (10 doces criados por 5 membros diferentes)
INSERT INTO criou (Membro, Doce) VALUES
('idana', 'Mousse de Chocolate'),
('idbernardo', 'Tarte de Maçã'),
('idcarlota', 'Bolo de Chocolate'),
('idcarlos', 'Pudim de Caramelo'),
('idema', 'Biscoitos'),
('idana', 'Tiramisu'),
('idbernardo', 'Torta de Amêndoa'),
('idcarlota', 'Pastel de Nata'),
('idcarlos', 'Bolo de Alfarroba'),
('idema', 'Tarte de Limão');

-- Inserção dos doces na tabela fez (cada doce foi feito por 2 membros, e um dos doces foi feito por 3 membros, neste caso, foi o bolo de alfarroba)
INSERT INTO fez (Membro, Doce, Tempo, Aspeto, Sabor) VALUES
('idana', 'Bolo de Chocolate', 3, 4, 5),

('idbernardo', 'Bolo de Chocolate', 4, 3, 5),
('idcarlota', 'Tarte de Maçã', 2, 5, 4),
('idcarlos', 'Tarte de Maçã', 3, 4, 3),
('idema', 'Mousse de Chocolate', 1, 5, 5),
('idana', 'Mousse de Chocolate', 1, 4, 4),
('idcarlota', 'Pudim de Caramelo', 5, 4, 3),
('idfrancisco', 'Pudim de Caramelo', 3, 5, 3),
('idcarlos', 'Biscoitos', 2, 5, 4),
('idgabriela', 'Biscoitos', 2, 4, 4),
('idbernardo', 'Tiramisu', 4, 5, 5),
('idana', 'Tiramisu', 5, 5, 3),
('idema', 'Pastel de Nata', 3, 5, 4),
('idana', 'Pastel de Nata', 4, 4, 5),
('idbernardo', 'Bolo de Alfarroba', 3, 4, 4),
('idcarlota', 'Bolo de Alfarroba', 2, 5, 3),
('idema', 'Bolo de Alfarroba', 5, 4, 5),
('idbernardo', 'Torta de Amêndoa', 4, 4, 5),
('idisabel', 'Torta de Amêndoa', 3, 4, 4),
('idjonathan', 'Tarte de Limão', 3, 4, 5),
('idemanuel', 'Tarte de Limão', 5, 5, 5);

d) Inserção das amizades

-- O membro com IdMemb = ‘idana’ é amigo de todos os restantes membros

```
INSERT INTO amigo (Membro1, Membro2) VALUES
    ('idana', 'idbernardo'), ('idbernardo', 'idana'),
    ('idana', 'idcarlota'), ('idcarlota', 'idana'),
    ('idana', 'idcarlos'), ('idcarlos', 'idana'),
    ('idana', 'idema'), ('idema', 'idana'),
    ('idana', 'idemanuel'), ('idemanuel', 'idana'),
    ('idana', 'idgabriela'), ('idgabriela', 'idana'),
    ('idana', 'idfrancisco'), ('idfrancisco', 'idana'),
    ('idana', 'idisabel'), ('idisabel', 'idana'),
    ('idana', 'idjonathan'), ('idjonathan', 'idana');
```

-- 5 membros com pelo menos 3 amigos cada

```
INSERT INTO amigo (Membro1, Membro2) VALUES
    ('idbernardo', 'idcarlota'), ('idcarlota', 'idbernardo'),
    ('idbernardo', 'idcarlos'), ('idcarlos', 'idbernardo'),
    ('idbernardo', 'idema'), ('idema', 'idbernardo'),
    ('idcarlota', 'idcarlos'), ('idcarlos', 'idcarlota'),
```

('idcarlota', 'idema'), ('idema', 'idcarlota'),
('idcarlos', 'idema'), ('idema', 'idcarlos'),
('idcarlos', 'idemanuel'), ('idemanuel', 'idcarlos'),
('idema', 'idemanuel'), ('idemanuel', 'idema');

e) Inserção de valores necessários adicionais

Após testarmos as nossas expressões em SQL obtidas para o exercício 4, chegámos à conclusão que seria necessário inserir os seguintes valores para haver no mínimo um tuplo em todas as seguintes alíneas:

```
-- Valores necessários para alínea 4.a)
INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Bolo de Chocolate', 'Chocolate', 250),
    ('Bolo de Chocolate', 'Canela', 2);

-- Valores necessários para alínea 4.b)
-- Inserção do membro Joaquim José
INSERT INTO membro (Nome, IdMemb, Pais, DataNasc) VALUES
    ('Joaquim José', 'idjoaquimjose', 'Portugal', '1987-05-11');
-- Inserção dos amigos de Joaquim José que criaram doces tradicionais
INSERT INTO amigo (Membro1, Membro2) VALUES
    ('idjoaquimjose', 'idema'), ('idema', 'idjoaquimjose'),
    ('idjoaquimjose', 'idcarlota'), ('idcarlota', 'idjoaquimjose'),
    ('idjoaquimjose', 'idcarlos'), ('idcarlos', 'idjoaquimjose');

-- Inserção do ingrediente "Natas" no doce Pastel de Nata para a alínea 4.d)
INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Pastel de Nata', 'Natas', 150);


-- Valores necessários para alínea 4.e) (inserção do ingrediente "Leite")
INSERT INTO temingrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Bolo de Chocolate', 'Leite', 250);


-- Valores necessários para alínea 4.f)
INSERT INTO temingrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Tarte de Maçã', 'Maçã', 150),
    ('Pastel de Nata', 'Baunilha', 2);

-- Inserção do arroz doce, valores necessários para alinea 4.g)
INSERT INTO doce (Nome, Descricao, Genero) VALUES
    ('Arroz Doce', 'Arroz com leite e canela e açucar', 'Tradicional');
INSERT INTO criou (Membro, Doce) VALUES
    ('idjonathan', 'Arroz Doce');
```

```
INSERT INTO temingrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Arroz Doce', 'Açucar', 250),
    ('Arroz Doce', 'Leite', 200),
    ('Arroz Doce', 'Canela', 3);
```

-- Inserção de valores necessários para alinea 4.h)

```
INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Tiramisu', 'Ovos', 6),
    ('Tiramisu', 'Chocolate', 3),
    ('Tiramisu', 'Açucar', 150),
    ('Mousse de Chocolate', 'Manteiga', 20),
    ('Mousse de Chocolate', 'Chocolate', 1),
    ('Mousse de Chocolate', 'Açucar', 50),
    ('Pudim de Caramelo', 'Ovos', 4),
    ('Pudim de Caramelo', 'Leite', 200),
    ('Pudim de Caramelo', 'Açucar', 150),
    ('Biscoitos', 'Farinha', 350),
    ('Biscoitos', 'Açucar', 150),
    ('Biscoitos', 'Ovos', 2),
    ('Torta de Amêndoa', 'Farinha', 280),
    ('Torta de Amêndoa', 'Açucar', 120),
    ('Torta de Amêndoa', 'Ovos', 3),
    ('Bolo de Alfarroba', 'Alfarroba', 150),
    ('Bolo de Alfarroba', 'Açucar', 170),
    ('Bolo de Alfarroba', 'Ovos', 5),
    ('Tarte de Limão', 'Manteiga', 80),
    ('Tarte de Limão', 'Leite', 170),
    ('Tarte de Limão', 'Água', 100),
    ('Tarte de Maçã', 'Leite', 160),
    ('Tarte de Maçã', 'Açucar', 80);
```

-- Inserção de valores necessários para 4.j)

```
INSERT INTO doce (Nome, Descricao, Genero) VALUES
    ('Doce de Natas', 'Doce caseiro de natas', 'Tradicional');
INSERT INTO criou (Membro, Doce) VALUES
    ('idana', 'Doce de Natas');
INSERT INTO temIngrediente (Doce, Ingrediente, Quantidade) VALUES
    ('Doce de Natas', 'Ovos', 5),
    ('Doce de Natas', 'Leite', 200),
    ('Doce de Natas', 'Açucar', 110);
```

Exercício 4 – Expressões em SQL e Álgebra Relacional

a) Indique os nomes dos membros espanhóis que criaram doces regionais que têm chocolate e canela como ingredientes.

SQL:

```
WITH doces AS (
   SELECT Doce
   FROM temingrediente, doce
   WHERE Ingrediente = 'Canela' AND Genero = 'Regional' AND doce.Nome =
temIngrediente.Doce

INTERSECT

   SELECT Doce
   FROM temIngrediente, doce
   WHERE Ingrediente = 'Chocolate' AND Genero = 'Regional' AND doce.Nome =
temIngrediente.Doce

)

SELECT DISTINCT nome
FROM membro, (criou NATURAL INNER JOIN doces) AS s
WHERE Pais = 'Espanha' AND membro.IdMemb = s.Membro
```

Álgebra Relacional:

doces ← πDoce(σIngrediente = ‘Canela’ ∧ Genero = ‘Regional’ ∧ doce.Nome = temIngrediente.Doce(doce x temIngrediente) ∩ πDoce(σIngrediente = ‘Chocolate’ ∧ Genero = ‘Regional’ ∧ doce.Nome = temIngrediente.Doce(doce x temIngrediente)

πNome(σPais = ‘Espanha’ ∧ membro.IdMemb = criou.Membro ∧ criou.Doce = doces.Doce(membro x criou x doces))

b) Quais são os doces tradicionais criados pelos amigos do “Joaquim José” que quando foram feitos tiveram pelo menos um 5 no aspeto?

SQL:

```
SELECT DISTINCT d.Nome
FROM doce d
JOIN criou c ON d.Nome = c.Doce
JOIN amigo a ON c.Membro = a.Membro2
JOIN fez f ON d.Nome = f.Doce
WHERE a.Membro1 = 'idjoaquimjose'
  AND d.Genero = 'Tradicional'
  AND f.Aspeto = 5;
```

Álgebra Relacional:

πdoce.Nome(σamigo.Membro1 = ‘idjoaquimjose’ ∧ doce.Genero = ‘Tradicional’ ∧ fez.Aspeto = 5 ∧ doce.Nome = criou.Doce ∧ criou.Membro = amigo.Membro2 ∧ doce.Nome = fez.Doce(doce x criou x fez x amigo))

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

SQL:

```
SELECT DISTINCT m.Nome
FROM membro m
WHERE NOT EXISTS (
    SELECT 1
    FROM fez f
    WHERE f.Membro = m.IdMemb
      AND NOT EXISTS (
          SELECT 1
          FROM amigo a
          JOIN criou c ON c.Membro = a.Membro2
          WHERE a.Membro1 = m.IdMemb AND c.Doce = f.Doce
      )
);
```

Álgebra Relacional:

doces_feitos ← πMembro, Doce(fez)

docesCriadosPorAmigos ← πMembro1, Doce(σMembro2 = criou.Membro(amigo x criou))

docesNaoCriadosPorAmigos ← doces_feitos – docesCriadosPorAmigos

resultado ← πIdMemb(membro) – πMembro(docesNaoCriadosPorAmigos)

πNome(membro ⋈ resultado)

NOTA: Nesta alínea nós interpretamos que o enunciado pedia o nome dos membros que só fizeram doces criados pelos seus amigos.

d) Quais são os doces com Natas que não tiveram um 5 no Sabor?

SQL:

```
SELECT DISTINCT d.Nome
FROM doce d
JOIN temingrediente ti ON d.Nome = ti.Doce
JOIN fez f ON d.Nome = f.Doce
WHERE ti.Ingrediente = 'Natas'
  AND f.Sabor < 5;
```

Álgebra Relacional:

πdoce.Nome(σtemIngrediente.Ingrediente = ‘Natas’ ∧ fez.Sabor < 5 ∧ doce.Nome = temIngrediente.Doce ∧ doce.Nome = fez.Doce(doce x temIngrediente x fez))

NOTA: Nesta alínea nós pensámos em duas interpretações diferentes para o enunciado. Uma em que o exercício pretendia que apresentássemos os doces com natas em que alguma vez que foram feitos não tiveram um 5 no Sabor e outra em que o mesmo pretendia que procurássemos os doces com natas que nunca tiveram um 5 no Sabor. Decidimos apresentar a resolução para a primeira interpretação, já que o enunciado não refere a palavra “nunca”.

e) Quais são os membros que criaram doces com canela e leite?

SQL:

```
SELECT DISTINCT c.Membro
FROM criou c
JOIN temingrediente ti1 ON c.Doce = ti1.Doce
JOIN temingrediente ti2 ON c.Doce = ti2.Doce
WHERE ti1.Ingrediente = 'Canela'
  AND ti2.Ingrediente = 'Leite';
```

Álgebra Relacional:

πcriou.Membro(σti1.Ingrediente = ‘Canela’ ∧ ti2.Ingrediente = ‘Leite’ ∧ criou.Doce = ti1.Doce ∧ criou.Doce = ti2.Doce(temIngrediente as ti1 x temIngrediente as ti2 x criou))

f) Quais são os doces que têm maçã ou baunilha?

SQL:

```
SELECT DISTINCT d.Nome
FROM doce d
JOIN temingrediente ti ON d.Nome = ti.Doce
WHERE ti.Ingrediente = 'Maçã' OR ti.Ingrediente = 'Baunilha'
```

Álgebra Relacional:

πDoce(σIngrediente = ‘Maçã’(temIngrediente)) ∪ πDoce(σIngrediente = ‘Baunilha’ (temIngrediente))

g) Qual é o custo do arroz doce?

SQL:

```
SELECT SUM(ti.Quantidade * i.Custo) AS Custo_Arroz_Doce
FROM temingrediente ti
JOIN ingrediente i ON ti.Ingrediente = i.Nome
WHERE ti.Doce = 'Arroz Doce';
```

Álgebra Relacional:

arroz_doce ← πdoce.Nome, Quantidade, Custo(σdoce.Nome = 'Arroz doce' ∧ temIngrediente.Ingrediente = ingrediente.Nome ∧ doce.Nome = temIngrediente.Doce(doce x ingrediente x temIngrediente))

Nome **G** sum(ΠQuantidade*Custo)(arroz_doce)

h) Indique o custo de cada doce.

SQL:

```
SELECT d.Nome, SUM(ti.Quantidade * i.Custo) AS Custo
FROM doce d
JOIN temingrediente ti ON d.Nome = ti.Doce
JOIN ingrediente i ON ti.Ingrediente = i.Nome
GROUP BY d.Nome;
```

Álgebra Relacional:

doces ← πdoce.Nome, Quantidade, Custo(σtemIngrediente.Ingrediente = ingrediente.Nome ∧ doce.Nome = temIngrediente.Doce(doce x ingrediente x temIngrediente))

Nome **G** sum(ΠQuantidade*Custo)(doces)

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

SQL:

```
SELECT c.Membro, COUNT(c.Doce) AS Numero_Doces
FROM criou c
GROUP BY c.Membro;
```

Álgebra Relacional:

membro **G** count(doce)(criou)

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

SQL:

```
WITH r AS (
  SELECT membro.nome, COUNT(criou.doce) AS num_doces
  FROM membro, criou
```

```
  WHERE membro.IdMemb = criou.membro
  GROUP BY membro.nome
)

SELECT nome
FROM r
WHERE num_doces = (SELECT MAX(num_doces) FROM r);
```

Álgebra Relacional:

membros_doces ← πmembro.Nome, criou.Doce(σmembro.IdMemb = criou.Membro(membro x criou))

numero_doces ← Nome **G** count(Doce) as n (membros_doces)

maiorNumeroDoces ← **G** max(n) as nmax (numero_doces)

πNome(σn = nmax(numero_doces x maiorNumeroDoces))

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

SQL:

```
WITH Custo_Doce AS (
   SELECT d.Nome, SUM(ti.Quantidade * i.Custo) AS Custo
   FROM doce d
   JOIN temingrediente ti ON d.Nome = ti.Doce
   JOIN ingrediente i ON ti.Ingrediente = i.Nome
   GROUP BY d.Nome
)

SELECT DISTINCT m.Nome
FROM fez f
JOIN membro m ON f.Membro = m.IdMemb
JOIN Custo_Doce cd ON f.Doce = cd.Nome
WHERE cd.Custo = (SELECT MAX(Custo) FROM Custo_Doce);
```

Álgebra Relacional:

doces ← πdoce.Nome, ti.Quantidade, i.Custo(σdoce.Nome = ti.Doce ∧ ti.Ingrediente = i.Nome(doce x temIngrediente as ti x Ingrediente as i))

custo_doce ← Nome **G** sum(ΠQuantidade*Custo) as c (doces)

doce_caro ← Nome **G** max(c)(custo_doce)

πmembro.Nome(σfez.Membro = membro.IdMemb ∧ fez.Doce = dc.Nome(membro x fez x doce_caro as dc))

l) Quais são os membros que são amigos de todos os amigos do "Joaquim José'?

SQL:

```
SELECT m1.IdMemb
FROM membro m1
WHERE NOT EXISTS (
   SELECT *
   FROM amigo a1
   WHERE a1.Membro1 = 'idjoaquimjose'
    AND NOT EXISTS (
       SELECT *
       FROM amigo a2
       WHERE a2.Membro1 = m1.IdMemb
        AND a2.Membro2 = a1.Membro2));
```

Álgebra Relacional:

amigos_joaquimjose ← πMembro2(σMembro1 = 'idjoaquimjose'(amigo))

πMembro1, Membro2(amigo) ÷ πMembro2(amigos_joaquimjose)

m) Quais são os membros que fizeram todos os doces com baunilha?

SQL:

```
SELECT m.Nome
FROM membro m
WHERE NOT EXISTS (
   SELECT *
   FROM temingrediente ti
   WHERE ti.Ingrediente = 'Baunilha'
    AND NOT EXISTS (
       SELECT *
       FROM fez f
       WHERE f.Membro = m.IdMemb
        AND f.Doce = ti.Doce
```

```
    )
);
```

Álgebra Relacional:

doces_baunilha ← πDoce(σIngrediente = ‘Baunilha’(temIngrediente))

πMembro, Doce(fez) ÷ πDoce(doces_baunilha)

n) Quais são os doces que quando foram feitos tiveram sempre um 1 no Tempo?

SQL:

```
SELECT f.Doce
FROM fez f
GROUP BY f.Doce
HAVING MIN(f.Tempo) = 1 AND MAX(f.Tempo) = 1;
```

Álgebra Relacional:

r ← πDoce(σTempo > 1(fez))

πDoce(fez) – πDoce(r)

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

SQL:

```
SELECT Doce
FROM (
SELECT t.Doce, SUM(i.Custo * t.Quantidade) AS CustoTotal
FROM temIngrediente t
JOIN ingrediente i ON t.Ingrediente = i.Nome
GROUP BY t.Doce
HAVING SUM(i.Custo * t.Quantidade) = (
   SELECT MIN(CustoTotal)
   FROM (
      SELECT SUM(i.Custo * t.Quantidade) AS CustoTotal
      FROM temIngrediente t
      JOIN ingrediente i ON t.Ingrediente = i.Nome
```

```
        GROUP BY t.Doce
    ) AS Custos
)
)

INTERSECT

SELECT Doce
FROM fez
GROUP BY Doce
HAVING AVG(Tempo) = (
    SELECT MIN(TempoMedio)
    FROM (
        SELECT AVG(Tempo) AS TempoMedio
        FROM fez
        GROUP BY Doce
    ) AS Tempos
);
```

Álgebra Relacional:

r ← πDoce, Custo, Quantidade(σIngrediente = Nome(ingrediente x temIngrediente))

custo_total ← Doce **G** sum(ΠCusto * Quantidade) as ct (r)

min_custo ← **G** min(ct) as mct (custo_total)

mais_baratos ← πDoce(σct = mct(custo_total ⋈ min_custo))

tempo_medio ← Doce **G** avg(Tempo) as tm (fez)

min_tempo ← **G** min(tm) as mtm (tempo_medio)

mais_rapidos ← πDoce(σtm = mtm(tempo_medio ⋈ min_tempo))

πDoce(mais_baratos) ∩ πDoce(mais_rapidos)

NOTA: Nesta alínea, nós pensámos em duas interpretações diferentes para o enunciado. Uma em que o exercício pretendia que juntássemos os doces mais baratos com os mais rápidos e outra em que o mesmo pretendia que procurássemos os doces mais baratos e ao mesmo tempo mais rápidos. Decidimos apresentar a resolução para a segunda interpretação, em que usamos um intersect entre os doces mais baratos e os mais rápidos.