Grupo: 05		Tema: 2	
a88333	João Teixeira	a88351	Guilherme Silva
a90251	Mohammed Rohaim		

O presente trabalho tem como ponto de partida a criação de um sistema de gestão para um clube de ténis e padel de luxo, localizado numa zona de grande prestígio e frequentado maioritariamente por residentes da Quinta do Lago e de Vale do Lobo. O clube dispõe de 22 courts — 10 de ténis em piso rápido, 4 de terra batida e 8 de padel — que são utilizados diariamente tanto para lazer como para torneios nacionais de elevado nível. A elevada procura e o perfil exigente dos utilizadores tornam indispensável um sistema robusto que assegure a organização e transparência na marcação e utilização dos courts.

Neste contexto, a base de dados assume um papel central. É ela que permitirá gerir de forma eficiente todas as reservas, distinguindo automaticamente os direitos e restrições de sócios e não sócios. Para além de permitir que sócios reservem com uma semana de antecedência e não sócios apenas com um dia~~, o sistema deverá contemplar uma hierarquia interna: sócios mais antigos ou com maior estatuto têm prioridade em caso de conflitos, reforçando o carácter exclusivo do clube.~~ Este controlo seria impraticável de forma manual, mas pode ser aplicado com rigor através da base de dados.

A gestão dos horários disponíveis é outro aspeto crítico. A base de dados não só organiza as reservas por court e intervalo de tempo, como também assegura que a utilização é otimizada, evitando sobreposições e períodos de inatividade. Além disso, calcula os preços automaticamente em função do horário, da duração da reserva ~~(1h a 2h30m)~~ (1h, 1h30min, 2h ou 2h30min), da necessidade de iluminação artificial no período noturno ~~(acréscimo de 5€ por reserva)~~ (cujo o acréscimo é configurável no sistema) e do tipo de piso. ~~Por exemplo, o preço base por hora para não sócios é de 15€ para terra batida, 10€ para piso rápido e 12,5€ para padel,~~ O tarifário base por hora, que varia consoante o piso, é gerido pela administração, com descontos de 30% para sócios. Em horário de pico (17h–21h), os valores sofrem um acréscimo ~~de 25%.~~ percentual, também ele definido no tarifário do sistema. ~~No caso de um grupo, basta haver um sócio para disfrutar das regalias.~~ Os descontos de sócio são aplicados automaticamente sempre que a reserva for efetuada por um utilizador com estatuto de sócio.

Se houver algum tipo de dano ~~quer seja no espaço quer seja no~~ no material do clube, o incidente será registado e o utilizador responsável terá de arrecadar com ~~de~~ 50% do prejuízo.

O sistema permitirá ainda ~~o registo detalhado de entradas e saídas de não sócios~~, a criação de perfis temporários para não sócios, limitados a três meses e o histórico de utilização de cada court. Estes dados são fundamentais para apoiar decisões de gestão, avaliar padrões de ocupação e planear futuros investimentos nas infraestruturas do clube.

Assim, a base de dados não é apenas uma ferramenta de apoio, mas sim a infraestrutura digital que garante eficiência, justiça e transparência na gestão de um espaço de prestígio. Ao centralizar toda a informação relativa a utilizadores, reservas e regras internas, assegura-se não só a qualidade do serviço oferecido, como também a sustentabilidade da operação do clube a longo prazo.

Estrutura do Modelo Entidade-Associação

Entidades e as suas descrições:

1. Utilizador - A entidade central que representa qualquer pessoa que interage com o clube (Entidade Base para Generalização).
2. Socio - Um Utilizador com privilégios de sócio (Especialização de Utilizador).
3. NaoSocio - Um Utilizador convidado ou temporário (Especialização de Utilizador).
4. Court - Representa um dos 22 campos físicos do clube.
5. Reserva - O evento de marcação de um court por um ou mais utilizadores.
6. Tarifario - Entidade que armazena as regras de negócio para o cálculo de preços, permitindo flexibilidade.
7. RegistoDano - Regista um incidente de dano ocorrido durante uma reserva.
8. ~~Material - Representa o material do clube que pode sofrer danos.~~
9. Torneio - Representa os torneios nacionais mencionados no texto da Parte 1.

Atributos:

1. Utilizador: ID_Utilizador (PK, Inteiro), Nome (Textual), Email (Textual, Chave Candidata), NIF (Inteiro, Chave Candidata), {Telefone} (Textual, Múltiplo Valor).
2. Socio: Herda: ID_Utilizador, Nome, Email, NIF, {Telefone}. Atributos próprios: Data_Admissao (Temporal), Estatuto_Socio (Textual).
3. NaoSocio: Herda: ID_Utilizador, Nome, Email, NIF, {Telefone}. Atributo próprio: Data_Criacao_Perfil (Temporal).
4. Court: ID_Court (PK, Inteiro), Nome_Court (Textual), Tipo_Piso (Textual).
5. Reserva: ID_Reserva (PK, Inteiro), DataHora_Inicio (Temporal), Duracao_Minutos (Inteiro), Preco_Final_Calculado() (Decimal), Requer_Iluminacao (Booleano).
6. Tarifario: ID_Tarifario (PK, Inteiro), Descricao (Textual), Tipo_Piso_Aplicavel (Textual), Preco_Base_Hora (Decimal), Acrescimo_Pico_Pct (Decimal), Acrescimo_Iluminacao (Decimal).
7. RegistoDano: ID_Dano (PK, Inteiro), Descricao (Textual), Valor_Prejuizo_Total (Decimal), Valor_Cobrado_Utilizador (Decimal).
8. ~~Material: ID_Material (PK, Inteiro), Descricao (Textual).~~
9. Torneio: ID_Torneio (PK, Inteiro), Nome (Textual), DataHora_Inicio (Temporal), DataHora_Fim (Temporal), DataHora_Fecho_Inscricoes (Temporal), Valor_Inscricao (Decimal), Num_Max_Inscritos (Inteiro). ~~Data_Inicio (Temporal), Data_Fim (Temporal).~~

*Nota: O Preco_Final_Calculado() leva parenteses porque é um atributo derivado

Associações e as suas descrições:

1. Generalização Utilizador (ISA) - Relação de especialização.

- Relação: Socio ISA Utilizador ; NaoSocio ISA Utilizador.
- Restrições: Total (todo Utilizador deve ser ou Socio ou NaoSocio) e Disjunta (não pode ser ambos).

2. Participa_Em - Associação ~~N:N~~ 1:N (Um-para-Muitos) que modela o grupo de utilizadores numa reserva.

- Relação: Utilizador (~~N~~ 1, Parcial) --- (Participa_Em) --- (N, Total) Reserva.
- Descrição: ~~Um Utilizador pode participar em muitas reservas (participação parcial). Uma Reserva tem de ter pelo menos um Utilizador (participação total).~~ Um Utilizador pode fazer muitas reservas. Uma Reserva é feita por exatamente um Utilizador.

3. Ocorre_Em - Associação 1:N (Um-para-Muitos) que liga a reserva ao court físico.

- Relação: Court (1, Parcial) --- (Ocorre_Em) --- (N, Total) Reserva.
- Descrição: Uma Reserva tem de ocorrer em exatamente um Court (participação total). Um Court pode ter muitas reservas (participação parcial).

4. Aplica_Tarifa - Associação 1:N (Um-para-Muitos) que define o preço da reserva.
- Relação: Tarifario (1, Parcial) --- (Aplica_Tarifa) --- (N, Total) Reserva.
- Descrição: Uma Reserva tem de ter exatamente um Tarifario associado para calcular o preço (participação total). Um Tarifario pode ser aplicado a muitas reservas (participação parcial).

5. Reporta_Dano - Associação 1:N (Um-para-Muitos) que liga um dano à reserva onde ocorreu.
- Relação: Reserva (1, Parcial) --- (Reporta_Dano) --- (N, Total) RegistoDano.
- Descrição: Um RegistoDano tem de estar associado a uma Reserva (participação total). Uma Reserva pode (ou não) originar muitos registos de dano (participação parcial).

~~6. Implica_Material - Associação N:N (Muitos-para-Muitos) que detalha os materiais danificados.~~
- ~~Relação: RegistoDano (N, Total) --- (Implica_Material) --- (N, Parcial) Material.~~
- ~~Descrição: Um RegistoDano tem de implicar pelo menos um Material (participação total). Um Material pode estar em muitos registos de dano (participação parcial).~~

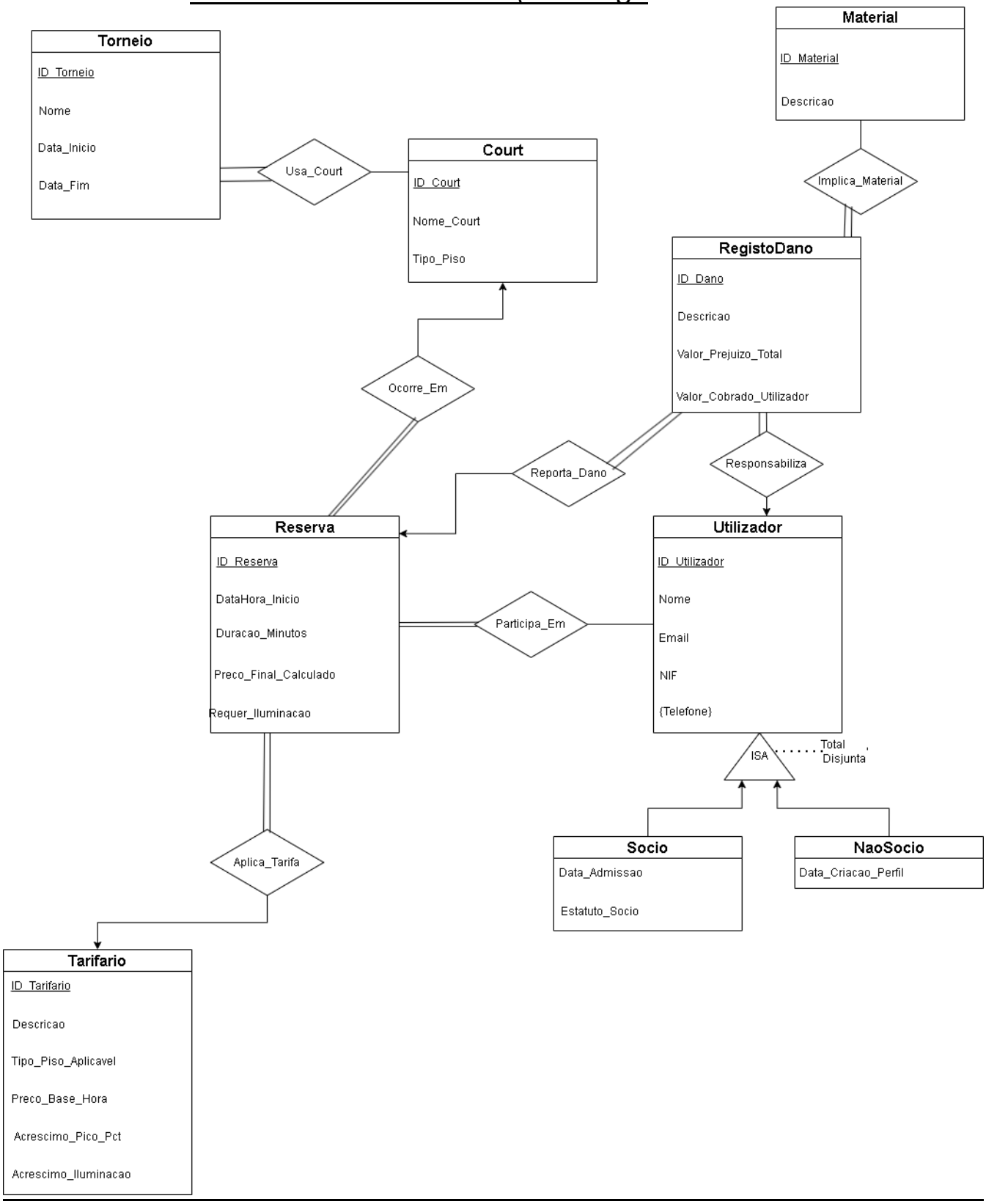
~~7. Responsabiliza - Associação 1:N (Um-para-Muitos) que identifica o culpado pelo dano.~~
- ~~Relação: Utilizador (1, Parcial) --- (Responsabiliza) --- (N, Total) RegistoDano.~~
- ~~Descrição: Um RegistoDano tem de ter exatamente um Utilizador responsável (participação total). Um Utilizador pode ser responsável por muitos danos (participação parcial).~~

8. Usa_Court - Associação N:N (Muitos-para-Muitos) que aloca courts a torneios.
- Relação: Torneio (N, Total) --- (Usa_Court) --- (N, Parcial) Court.
- Descrição: Um Torneio tem de usar pelo menos um Court (participação total). Um Court pode ser usado em muitos torneios (participação parcial).

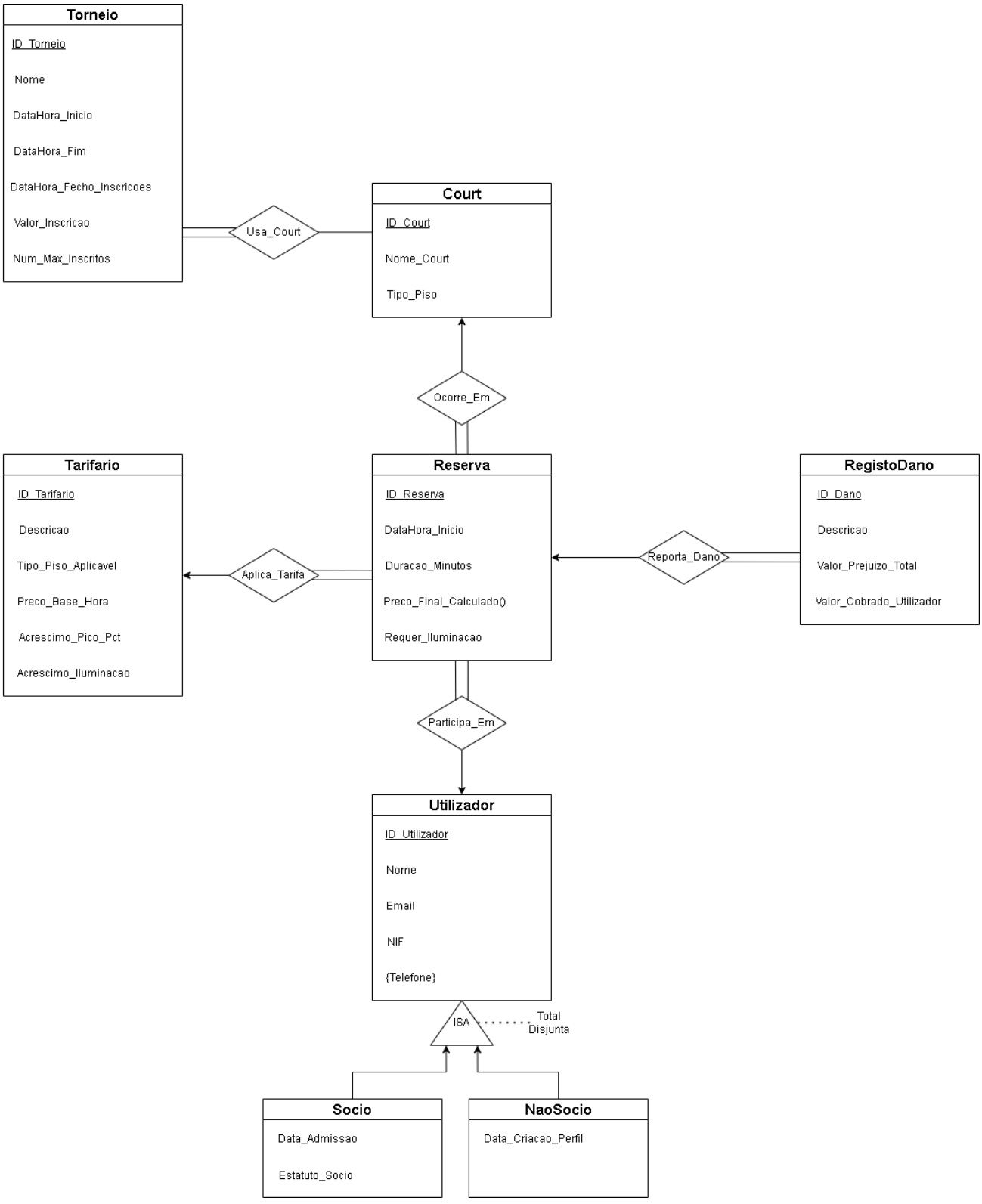
Restrições de integridade:

1. Restrição de Antecedência de Reserva - Um Utilizador do tipo Socio só pode reservar com 7 dias de antecedência, enquanto um NaoSocio só o pode fazer com 1 dia de antecedência.
2. Restrição de Duração da Reserva - O atributo Reserva.Duracao_Minutos deve ~~ser um valor entre 60 (1h) e 150 (2h30m).~~ assumir apenas um dos seguintes valores discretos: 60min (1h), 90min (1h30min), 120min (2h) ou 150min (2h30min).
3. Restrição de Perfil Temporário - Para um NaoSocio, o perfil não pode ser usado se tiverem passado mais de 3 meses desde a Data_Criacao_Perfil.
4. Restrição de Cálculo de Dano - O atributo RegistoDano.Valor_Cobrado_Utilizador deve ser calculado como 50% (metade) do RegistoDano.Valor_Prejuizo_Total.
5. Restrição de Desconto de Sócio - ~~No cálculo do preço final de uma Reserva, se pelo menos um dos utilizadores do grupo (associados pela relação Participa_Em) for do tipo Socio, deve ser aplicado um desconto de 30% ao valor total.~~ O desconto de 30% sobre o valor total da reserva é aplicado exclusivamente se o Utilizador responsável pela marcação (associado à Reserva pela relação Participa_Em) for do tipo Socio. Se a reserva for feita por um NaoSocio, o desconto não se aplica, independentemente de quem jogue com ele.

Modelo Entidade-Associação Antigo



Modelo Entidade-Associação Novo



Modelo Relacional

Utilizador(ID_Utilizador, Nome, Email, NIF)

NN(Nome)

NN(Email)

NN(NIF)

UNIQUE(Email)

UNIQUE(NIF)

Utilizador_Telefone(ID_Utilizador, Telefone**)**

FK(ID_Utilizador) -> Utilizador

NN(Telefone)

Socio(ID_Utilizador, Data_Admissao, Estatuto_Socio)

FK(ID_Utilizador) -> Utilizador

NN(Data_Admissao)

NaoSocio(ID_Utilizador, Data_Criacao_Perfil)

FK(ID_Utilizador) -> Utilizador

NN(Data_Criacao_Perfil)

Court(ID_Court, Nome_Court, Tipo_Piso)

NN(Nome_Court)

NN(Tipo_Piso)

Tarifario(ID_Tarifario, Descricao, Tipo_Piso_Aplicavel, Preco_Base_Hora, Acrescimo_Pico_Pct, Acrescimo_Iluminacao)

NN(Descricao)

NN(Preco_Base_Hora)

Reserva(ID_Reserva, DataHora_Inicio, Duracao_Minutos, ~~Preco_Final_Calculado~~, Requer_Iluminacao, ID_Court, ID_Tarifario, ID_Utilizador)

FK(ID_Court) -> Court

FK(ID_Tarifario) -> Tarifario

FK(ID_Utilizador) -> Utilizador

NN(DataHora_Inicio)

NN(Duracao_Minutos)

NN(Preco_Final_Calculado)

NN(Requer_Iluminacao)

NN(ID_Court)

NN(ID_Tarifario)

NN(ID_Utilizador)

RegistoDano(ID_Dano, Descricao, Valor_Prejuizo_Total, Valor_Cobrado_Utilizador, ID_Reserva, ~~ID_Utilizador~~)

FK(ID_Reserva) -> Reserva

~~FK(ID_Utilizador) -> Utilizador~~

NN(Descricao)

NN(Valor_Prejuizo_Total)

NN(ID_Reserva)

~~NN(ID_Utilizador)~~

~~**Material(**ID_Material, Descricao)~~

~~NN(Descricao)~~

Torneio(ID_Torneio, Nome, DataHora_Inicio, DataHora_Fim, DataHora_Fecho_Inscricoes, Valor_Inscricao, Num_Max_Inscritos ~~Data_Inicio, Data_Fim~~)

NN(Nome)

NN(DataHora_Inicio)

NN(DataHora_Fim)

NN(DataHora_Fecho_Inscricoes)

NN(Valor_Inscricao)

NN(Num_Max_Inscritos)

~~NN(Data_Inicio)~~

~~NN(Data_Fim)~~

~~**Participa(**ID_Utilizador, ID_Reserva**)**~~

~~FK(ID_Utilizador) -> Utilizador~~

~~FK(ID_Reserva) -> Reserva~~

~~**Implica(**ID_Dano, ID_Material**)**~~

~~FK(ID_Dano) -> RegistoDano~~

FK(ID_Material) -> Material

Aloca_Torneio(ID_Torneio, ID_Court**)**

FK(ID_Torneio) -> Torneio

FK(ID_Court) -> Court

Criação do Esquema (DDL):

Nota: O esquema da base de dados foi implementado garantindo a integridade dos dados através de Chaves Primárias, Estrangeiras, restrições CHECK e TRIGGERS."

```sql
/*
 * TRABALHO PRÁTICO - Grupo 05
 * Ficheiro: ddl.sql
 */

-- ============================================================================
-- 1. LIMPEZA DE ESTRUTURAS ANTIGAS
-- ============================================================================

DROP TRIGGER IF EXISTS trg_calculo_dano ON RegistoDano;
DROP TRIGGER IF EXISTS trg_verificar_regras_reserva ON Reserva;
DROP FUNCTION IF EXISTS fn_calculo_dano;
DROP FUNCTION IF EXISTS fn_verificar_regras_reserva;

DROP TABLE IF EXISTS Aloca_Torneio CASCADE;
DROP TABLE IF EXISTS Torneio CASCADE;
DROP TABLE IF EXISTS RegistoDano CASCADE;
DROP TABLE IF EXISTS Reserva CASCADE;
DROP TABLE IF EXISTS Tarifario CASCADE;
DROP TABLE IF EXISTS Court CASCADE;
DROP TABLE IF EXISTS NaoSocio CASCADE;
DROP TABLE IF EXISTS Socio CASCADE;
DROP TABLE IF EXISTS Utilizador_Telefone CASCADE;
DROP TABLE IF EXISTS Utilizador CASCADE;

-- ============================================================================
-- 2. CRIAÇÃO DE TABELAS
-- ============================================================================

CREATE TABLE Utilizador (
    ID_Utilizador INT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE,
    NIF INT NOT NULL UNIQUE
);

CREATE TABLE Utilizador_Telefone (
    ID_Utilizador INT,
    Telefone VARCHAR(20),
    PRIMARY KEY (ID_Utilizador, Telefone),
    FOREIGN KEY (ID_Utilizador) REFERENCES Utilizador(ID_Utilizador) ON DELETE CASCADE
);

CREATE TABLE Socio (
    ID_Utilizador INT PRIMARY KEY,
    Data_Admissao DATE NOT NULL,
    Estatuto_Socio VARCHAR(50),
    FOREIGN KEY (ID_Utilizador) REFERENCES Utilizador(ID_Utilizador) ON DELETE CASCADE
);

CREATE TABLE NaoSocio (
    ID_Utilizador INT PRIMARY KEY,
    Data_Criacao_Perfil DATE NOT NULL,
    FOREIGN KEY (ID_Utilizador) REFERENCES Utilizador(ID_Utilizador) ON DELETE CASCADE
);

```

```sql
CREATE TABLE Court (
    ID_Court INT PRIMARY KEY,
    Nome_Court VARCHAR(50) NOT NULL,
    Tipo_Piso VARCHAR(50) NOT NULL
);

CREATE TABLE Tarifario (
    ID_Tarifario INT PRIMARY KEY,
    Descricao VARCHAR(100) NOT NULL,
    Tipo_Piso_Aplicavel VARCHAR(50),
    Preco_Base_Hora DECIMAL(5,2) NOT NULL,
    Acrescimo_Pico_Pct DECIMAL(5,2),
    Acrescimo_Iluminacao DECIMAL(5,2)
);

CREATE TABLE Reserva (
    ID_Reserva INT PRIMARY KEY,
    DataHora_Inicio TIMESTAMP NOT NULL,
    Duracao_Minutos INT NOT NULL,
    Requer_Iluminacao BOOLEAN NOT NULL DEFAULT FALSE,
    ID_Court INT NOT NULL,
    ID_Tarifario INT NOT NULL,
    ID_Utilizador INT NOT NULL,
    FOREIGN KEY (ID_Court) REFERENCES Court(ID_Court),
    FOREIGN KEY (ID_Tarifario) REFERENCES Tarifario(ID_Tarifario),
    FOREIGN KEY (ID_Utilizador) REFERENCES Utilizador(ID_Utilizador),
    CONSTRAINT CHK_Duracao_Valida CHECK (Duracao_Minutos IN (60, 90, 120, 150))
);

CREATE TABLE RegistoDano (
    ID_Dano INT PRIMARY KEY,
    Descricao VARCHAR(255) NOT NULL,
    Valor_Prejuizo_Total DECIMAL(10,2) NOT NULL,
    -- O Valor Cobrado será calculado automaticamente pelo Trigger (50%)
    Valor_Cobrado_Utilizador DECIMAL(10,2),
    ID_Reserva INT NOT NULL,
    FOREIGN KEY (ID_Reserva) REFERENCES Reserva(ID_Reserva)
);

CREATE TABLE Torneio (
    ID_Torneio INT PRIMARY KEY,
    Nome VARCHAR(100) NOT NULL,
    DataHora_Inicio TIMESTAMP NOT NULL,
    DataHora_Fim TIMESTAMP NOT NULL,
    DataHora_Fecho_Inscricoes TIMESTAMP NOT NULL,
    Valor_Inscricao DECIMAL(10,2) NOT NULL,
    Num_Max_Inscritos INT NOT NULL
);

CREATE TABLE Aloca_Torneio (
    ID_Torneio INT,
    ID_Court INT,
    PRIMARY KEY (ID_Torneio, ID_Court),
    FOREIGN KEY (ID_Torneio) REFERENCES Torneio(ID_Torneio),
    FOREIGN KEY (ID_Court) REFERENCES Court(ID_Court)
);

-- ================================================================================
-- 3. TRIGGERS E FUNÇÕES
-- ================================================================================

/*
 * TRIGGER 1: Validação de Reservas
 * Implementa:
 * - Regra de Antecedência: Sócios (7 dias) vs Não Sócios (1 dia).
 * - Regra de Perfil Temporário: Não Sócios com perfil > 3 meses não podem reservar.
 */
CREATE OR REPLACE FUNCTION fn_verificar_regras_reserva()
RETURNS TRIGGER AS $$
DECLARE
    v_data_criacao DATE;
    v_e_socio BOOLEAN;
    v_diferenca_dias INT;
    v_meses_perfil INT;
BEGIN
```

```
132.     -- Verificar se é sócio
133.     SELECT EXISTS(SELECT 1 FROM Socio WHERE ID_Utilizador = NEW.ID_Utilizador) INTO
v_e_socio;
134.
135.     -- Calcular antecedência da reserva em dias
136.     v_diferenca_dias := DATE_PART('day', NEW.DataHora_Inicio - CURRENT_TIMESTAMP);
137.
138.     IF v_e_socio THEN
139.         -- Regra Sócio: Antecedência máxima de 7 dias
140.         IF v_diferenca_dias > 7 THEN
141.             RAISE EXCEPTION 'Erro: Sócios só podem reservar com 7 dias de
antecedência.';
142.         END IF;
143.     ELSE
144.         -- Lógica para NÃO SÓCIOS
145.         SELECT Data_Criacao_Perfil INTO v_data_criacao FROM NaoSocio WHERE
ID_Utilizador = NEW.ID_Utilizador;
146.
147.         -- Calcular idade do perfil em meses
148.         v_meses_perfil := (DATE_PART('year', CURRENT_DATE) - DATE_PART('year',
v_data_criacao)) * 12 +
149.                           (DATE_PART('month', CURRENT_DATE) - DATE_PART('month',
v_data_criacao));
150.
151.         -- Regra: Perfil expira após 3 meses
152.         IF v_meses_perfil >= 3 THEN
153.              RAISE EXCEPTION 'Erro: Perfil de Não Sócio expirado (mais de 3 meses).';
154.         END IF;
155.
156.         -- Regra Não Sócio: Antecedência máxima de 1 dia
157.         IF v_diferenca_dias > 1 THEN
158.             RAISE EXCEPTION 'Erro: Não Sócios só podem reservar com 1 dia de
antecedência.';
159.         END IF;
160.     END IF;
161.
162.     RETURN NEW;
163. END;
164. $$ LANGUAGE plpgsql;
165.
166. CREATE TRIGGER trg_verificar_regras_reserva
167. BEFORE INSERT ON Reserva
168. FOR EACH ROW EXECUTE FUNCTION fn_verificar_regras_reserva();
169.
170. /*
171.  * TRIGGER 2: Cálculo Automático de Danos
172.  * Implementa:
173.  * - Regra de Cobrança: O valor cobrado ao utilizador é sempre 50% do prejuízo total.
174.  */
175.
176. CREATE OR REPLACE FUNCTION fn_calculo_dano()
177. RETURNS TRIGGER AS $$
178. BEGIN
179.     NEW.Valor_Cobrado_Utilizador := NEW.Valor_Prejuizo_Total * 0.50;
180.     RETURN NEW;
181. END;
182. $$ LANGUAGE plpgsql;
183.
184. CREATE TRIGGER trg_calculo_dano
185. BEFORE INSERT ON RegistoDano
186. FOR EACH ROW EXECUTE FUNCTION fn_calculo_dano();
187.
```

Povoamento de Dados (DML):

Nota: O povoamento foi realizado utilizando datas dinâmicas (CURRENT_DATE e NOW()) e intervalos temporais. Esta opção garante que as regras de negócio (como a validade do perfil de 3 meses ou a antecedência das reservas) funcionam corretamente independentemente da data em que o script for executado.

```sql
/*
 * TRABALHO PRÁTICO - Grupo 05
 * Ficheiro: dml_insert.sql
 */

-- ============================================================================
-- 1. UTILIZADORES
-- ============================================================================

INSERT INTO Utilizador (ID_Utilizador, Nome, Email, NIF) VALUES
    (1,  'Carlos Admin',      'admin@clubedeluxo.pt',       100000001),
    (2,  'Maria Receção',     'maria.rececao@clubedeluxo.pt', 100000002),
    -- Sócios Gold
    (10, 'João Teixeira',     'joao.teixeira@mail.com',     200300400),
    (11, 'Guilherme Silva',   'guilherme.silva@mail.com',   210310410),
    (12, 'Mohammed Rohaim',   'mohammed.rohaim@mail.com',   220320420),
    (13, 'Cristiano Ronaldo', 'cr7@luxo.pt',                999888777),
    (14, 'Madonna Ciccone',   'madonna@music.com',          888777666),
    (15, 'José Mourinho',     'special.one@coach.com',      777666555),
    -- Sócios Silver
    (20, 'António Costa',     'antonio.costa@mail.pt',      123123123),
    (21, 'Ana Gomes',         'ana.gomes@mail.pt',          321321321),
    (22, 'Ricardo Araújo',    'ricardo.araujo@mail.pt',     456456456),
    -- Sócios Normal
    (23, 'Beatriz Lebre',     'beatriz.lebre@mail.pt',      654654654),
    (24, 'Tiago Bettencourt', 'tiago.musica@mail.pt',       987987987),
    (25, 'Marisa Liz',        'marisa.liz@mail.pt',         147147147),
    (26, 'Herman José',       'herman@humor.pt',            258258258),
    (27, 'Catarina Furtado',  'catarina@tv.pt',             369369369),
    -- Não Sócios
    (30, 'John Smith',        'john.smith@uk.mail.com',     500000001),
    (31, 'Pierre Dupont',     'pierre.dupont@fr.mail.com',  500000002),
    (32, 'Hans Müller',       'hans.muller@de.mail.com',    500000003),
    (33, 'Ana Visitante',     'ana.visitante@mail.com',     230330430),
    (34, 'Pedro Turista',     'pedro.turista@mail.com',     240340440),
    (35, 'Sofia Passageira',  'sofia.pass@mail.com',        500000010),
    (36, 'Lucas Espanhol',    'lucas.es@mail.com',          500000011),
    (37, 'Emma British',      'emma.uk@mail.com',           500000012),
    (38, 'Sven Svensson',     'sven.sw@mail.com',           500000013),
    (39, 'Giulia Rossi',      'giulia.it@mail.com',         500000014);

-- ============================================================================
-- 2. ESPECIALIZAÇÃO (Sócios, Não Sócios e Contactos)
-- ============================================================================

-- 2.1 Sócios
INSERT INTO Socio (ID_Utilizador, Data_Admissao, Estatuto_Socio) VALUES
    (1,  '2010-01-01', 'Staff'),
    (2,  '2015-05-01', 'Staff'),
    (10, '2020-01-15', 'Gold'),
    (11, '2021-05-20', 'Gold'),
    (12, '2019-03-10', 'Gold'),
    (13, '2018-08-01', 'Gold'),
    (14, '2022-06-01', 'Gold'),
    (15, '2020-12-25', 'Gold'),
    (20, '2023-01-10', 'Silver'),
    (21, '2023-02-15', 'Silver'),
    (22, '2023-03-20', 'Silver'),
    (23, '2024-01-05', 'Normal'),
    (24, '2024-02-10', 'Normal'),
    (25, '2024-03-15', 'Normal'),
    (26, '2024-04-20', 'Normal'),
    (27, '2024-05-25', 'Normal');

-- 2.2 Não Sócios (Datas dinâmicas para evitar expiração automática)
INSERT INTO NaoSocio (ID_Utilizador, Data_Criacao_Perfil) VALUES
    (30, CURRENT_DATE - INTERVAL '1 month'),
    (31, CURRENT_DATE - INTERVAL '1 month'),
    (32, CURRENT_DATE - INTERVAL '2 days'),
    (33, CURRENT_DATE - INTERVAL '5 days'),
    (34, CURRENT_DATE - INTERVAL '10 days'),
    (35, CURRENT_DATE - INTERVAL '1 month'),
    (36, CURRENT_DATE - INTERVAL '2 months'), -- Ainda válido (<3 meses)
    (37, CURRENT_DATE - INTERVAL '1 day'),
    (38, CURRENT_DATE - INTERVAL '1 day'),
```

```
76.     (39, CURRENT_DATE - INTERVAL '1 day');
77.
78. -- 2.3 Telefones
79. INSERT INTO Utilizador_Telefone (ID_Utilizador, Telefone) VALUES
80.     (10, '910000001'), (10, '910000002'),
81.     (11, '960000001'),
82.     (12, '930000001'), (12, '210000001'),
83.     (13, '999999999'),
84.     (30, '+44 7700 9000'),
85.     (31, '+33 6 12 34 56'),
86.     (32, '+49 170 123456');
87.
88. -- ============================================================================
89. -- 3. COURTS E TARIFÁRIOS
90. -- ============================================================================
91.
92. INSERT INTO Court (ID_Court, Nome_Court, Tipo_Piso) VALUES
93.     (1,  'Central Rápido',     'Piso Rápido'),
94.     (2,  'Court Rápido 2',     'Piso Rápido'),
95.     (3,  'Court Rápido 3',     'Piso Rápido'),
96.     (4,  'Court Rápido 4',     'Piso Rápido'),
97.     (5,  'Court Rápido 5',     'Piso Rápido'),
98.     (6,  'Court Rápido 6',     'Piso Rápido'),
99.     (7,  'Court Rápido 7',     'Piso Rápido'),
100.    (8,  'Court Rápido 8',     'Piso Rápido'),
101.    (9,  'Court Rápido 9',     'Piso Rápido'),
102.    (10, 'Court Rápido 10',    'Piso Rápido'),
103.    (11, 'Terra Batida A',     'Terra Batida'),
104.    (12, 'Terra Batida B',     'Terra Batida'),
105.    (13, 'Terra Batida C',     'Terra Batida'),
106.    (14, 'Terra Batida D',     'Terra Batida'),
107.    (15, 'Padel Panorâmico 1', 'Padel'),
108.    (16, 'Padel Panorâmico 2', 'Padel'),
109.    (17, 'Padel Vidro 3',      'Padel'),
110.    (18, 'Padel Vidro 4',      'Padel'),
111.    (19, 'Padel Vidro 5',      'Padel'),
112.    (20, 'Padel Vidro 6',      'Padel'),
113.    (21, 'Padel Vidro 7',      'Padel'),
114.    (22, 'Padel Vidro 8',      'Padel');
115.
116. INSERT INTO Tarifario
117. (ID_Tarifario, Descricao, Tipo_Piso_Aplicavel, Preco_Base_Hora, Acrescimo_Pico_Pct,
Acrescimo_Iluminacao)
118. VALUES
119.    (1, 'Ténis Rápido Standard',      'Piso Rápido',  10.00, 0.25, 5.00),
120.    (2, 'Ténis Terra Batida Premium', 'Terra Batida', 15.00, 0.25, 5.00),
121.    (3, 'Padel Standard',             'Padel',        12.50, 0.25, 5.00);
122.
123. -- ============================================================================
124. -- 4. RESERVAS
125. -- Nota: Usamos NOW() para garantir que o script funciona em qualquer data
126. -- sem violar regras de antecedência ou histórico.
127. -- ============================================================================
128.
129. -- 4.1 Reservas Passadas (Histórico)
130. INSERT INTO Reserva (ID_Reserva, DataHora_Inicio, Duracao_Minutos, Requer_Iluminacao,
ID_Court, ID_Tarifario, ID_Utilizador) VALUES
131.    (100, NOW() - INTERVAL '30 days', 60,  FALSE, 1,  1, 10),
132.    (101, NOW() - INTERVAL '30 days', 90,  FALSE, 11, 2, 11),
133.    (102, NOW() - INTERVAL '29 days', 60,  TRUE,  15, 3, 30),
134.    (103, NOW() - INTERVAL '29 days', 90,  TRUE,  15, 3, 31),
135.    (104, NOW() - INTERVAL '28 days', 120, FALSE, 2,  1, 13),
136.    (105, NOW() - INTERVAL '28 days', 60,  FALSE, 3,  1, 14),
137.    (106, NOW() - INTERVAL '27 days', 150, FALSE, 11, 2, 20),
138.    (107, NOW() - INTERVAL '27 days', 60,  FALSE, 12, 2, 21),
139.    (108, NOW() - INTERVAL '26 days', 90,  TRUE,  1,  1, 15),
140.    (109, NOW() - INTERVAL '25 days', 60,  FALSE, 16, 3, 32);
141.
142. -- 4.2 Reservas Futuras (Atenção às regras de antecedência!)
143. INSERT INTO Reserva (ID_Reserva, DataHora_Inicio, Duracao_Minutos, Requer_Iluminacao,
ID_Court, ID_Tarifario, ID_Utilizador) VALUES
144.    -- Não Sócio: Marca para daqui a 2h (Válido: < 24h antecedência)
145.    (300, NOW() + INTERVAL '2 hours', 60,  FALSE, 1,  1, 35),
146.    (301, NOW() + INTERVAL '3 hours', 90,  FALSE, 2,  1, 36),
147.
```

```
148.     -- Sócio: Marca para amanhã (Válido: < 7 dias antecedência)
149.     (302, NOW() + INTERVAL '1 day',   60,  FALSE, 11, 2, 10),
150.     (303, NOW() + INTERVAL '1 day',   120, FALSE, 15, 3, 11),
151.
152.     -- Sócio VIP/Gold: Marca para daqui a 3-6 dias
153.     (400, NOW() + INTERVAL '3 days',  90,  FALSE, 1,  1, 13),
154.     (401, NOW() + INTERVAL '4 days',  60,  TRUE,  15, 3, 14),
155.     (402, NOW() + INTERVAL '5 days',  120, FALSE, 11, 2, 15),
156.     (403, NOW() + INTERVAL '6 days',  60,  FALSE, 5,  1, 10);
157.
158. -- ============================================================================
159. -- 5. DANOS
160. -- Nota: Valor_Cobrado é NULL para testar o Trigger que calcula 50% automaticamente.
161. -- ============================================================================
162.
163. INSERT INTO RegistoDano (ID_Dano, Descricao, Valor_Prejuizo_Total,
Valor_Cobrado_Utilizador, ID_Reserva) VALUES
164.     (1, 'Raquete partiu o vidro lateral do campo de Padel', 400.00, NULL, 102), --
Trigger deve definir 200€
165.     (2, 'Rede rasgada por mau uso',                          150.00, NULL, 100), --
Trigger deve definir 75€
166.     (3, 'Projetor de luz partido com bola',                  300.00, NULL, 108), --
Trigger deve definir 150€
167.     (4, 'Banco de descanso partido',                          80.00, NULL, 300); --
Trigger deve definir 40€
168.
169. -- ============================================================================
170. -- 6. TORNEIOS
171. -- Nota: Datas futuras calculadas relativamente a "hoje".
172. -- ============================================================================
173.
174. INSERT INTO Torneio
175. (ID_Torneio, Nome, DataHora_Inicio, DataHora_Fim, DataHora_Fecho_Inscricoes,
Valor_Inscricao, Num_Max_Inscritos)
176. VALUES
177.     (1, 'Open de Verão Quinta do Lago',
178.      NOW() + INTERVAL '2 months',
179.      NOW() + INTERVAL '2 months 2 days',
180.      NOW() + INTERVAL '1 month 25 days',
181.      50.00, 32),
182.
183.     (2, 'Torneio Solidário Padel VIP',
184.      NOW() + INTERVAL '3 months',
185.      NOW() + INTERVAL '3 months 1 day',
186.      NOW() + INTERVAL '2 months 25 days',
187.      75.00, 16);
188.
189. -- Alocação de Courts aos Torneios
190. INSERT INTO Aloca_Torneio (ID_Torneio, ID_Court) VALUES
191.     (1, 1), (1, 2), (1, 11), (1, 12),
192.     (2, 15), (2, 16);
193.
```

Queries:

Q.1: Deve seguir o padrão “select from where”, com duas tabelas no from e pelo menos um filtro no where.

Devolve o Nome, Email e NIF de todos os Sócios que têm estatuto 'Gold'.

```
SELECT
  u.Nome, u.Email, u.NIF, s.Estatuto_Socio
FROM
  Utilizador u, Socio s
WHERE
  u.ID_Utilizador = s.ID_Utilizador
```

```
  AND s.Estatuto_Socio = 'Gold';
```

Q.2: Deve seguir o padrão “select from where group by”

Devolve o tipo de piso e o número total de reservas efetuadas para cada tipo.

```
SELECT
  c.Tipo_Piso, COUNT(r.ID_Reserva) AS Total_Reservas
FROM
  Reserva r
JOIN
  Court c ON r.ID_Court = c.ID_Court
GROUP BY
  c.Tipo_Piso;
```

Q.3: Deve seguir o padrão “select from where group by having”

Devolve o nome dos Courts que tiveram mais do que 1 reserva registada no histórico.

```
SELECT
  c.Nome_Court, COUNT(r.ID_Reserva) AS Qtd_Reservas
FROM
  Court c
JOIN
  Reserva r ON c.ID_Court = r.ID_Court
GROUP BY
  c.Nome_Court
HAVING
  COUNT(r.ID_Reserva) > 1;
```

Q.4: Deve incluir um outer join (de qualquer tipo)

Lista todos os Utilizadores registados e, se forem Sócios, mostra a data de admissão (mesmo que não sejam sócios, devem aparecer na lista).

```
SELECT
  u.Nome, u.Email, s.Data_Admissao, s.Estatuto_Socio
FROM
  Utilizador u
LEFT OUTER JOIN
  Socio s ON u.ID_Utilizador = s.ID_Utilizador;
```

Q.5: Deve usar uma nested query

Devolve o nome dos utilizadores que fizeram reservas em courts de 'Padel'.

```
SELECT
  Nome
FROM
```

```
  Utilizador
WHERE
  ID_Utilizador IN (
    SELECT r.ID_Utilizador
    FROM Reserva r
    JOIN Court c ON r.ID_Court = c.ID_Court
    WHERE c.Tipo_Piso = 'Padel'
  );
```

Q.6: Deve usar uma tabela temporária, através da utilização de “with”

Utilizando uma tabela temporária, calcula o total de minutos jogados por cada utilizador e mostra apenas aqueles que jogaram.

```
WITH MinutosJogados AS (
  SELECT
    ID_Utilizador, SUM(Duracao_Minutos) AS Total_Tempo
  FROM
    Reserva
  GROUP BY
    ID_Utilizador
)
SELECT
  u.Nome, m.Total_Tempo
FROM
  Utilizador u
JOIN
  MinutosJogados m ON u.ID_Utilizador = m.ID_Utilizador;
```

Q.7: Deve usar exists ou not exists

Devolve o nome dos Utilizadores que NUNCA fizeram nenhuma reserva.

```
SELECT
  u.Nome
FROM
  Utilizador u
WHERE NOT EXISTS (
  SELECT 1
  FROM Reserva r
  WHERE r.ID_Utilizador = u.ID_Utilizador
);
```

Q.8: Deve usar um self-join, i.e., ter duas vezes a mesma tabela na cláusula from

Lista pares de Courts diferentes que tenham exatamente o mesmo Tipo de Piso.

```
SELECT
  c1.Nome_Court AS Court_A,
```

```
    c2.Nome_Court AS Court_B,
    c1.Tipo_Piso
FROM
    Court c1, Court c2
WHERE
    c1.Tipo_Piso = c2.Tipo_Piso
    AND c1.ID_Court < c2.ID_Court
ORDER BY
    c1.Tipo_Piso;
```

Q.9: Deve implementar a expressão sql equivalente a uma divisão em álgebra relacional

Devolve o nome dos utilizadores que já efetuaram reservas em TODOS os tipos de piso disponíveis no clube (Piso Rápido, Terra Batida, Padel).

```
SELECT u.Nome
FROM Utilizador u
WHERE NOT EXISTS (
    (SELECT DISTINCT Tipo_Piso FROM Court)
    EXCEPT
    (SELECT DISTINCT c.Tipo_Piso
     FROM Reserva r
     JOIN Court c ON r.ID_Court = c.ID_Court
     WHERE r.ID_Utilizador = u.ID_Utilizador)
);
```

Q.10: Deve filtrar campos de texto através da comparação com “like”

Devolve o nome e email dos utilizadores cujo email pertence ao domínio 'mail.com'.

```
SELECT
    Nome, Email
FROM
    Utilizador
WHERE
    Email LIKE '%@mail.com';
```

Q.11: Deve usar uma nested query com comparação usando a cláusula ALL

Devolve as reservas que tiveram a duração máxima registada no sistema (ou seja, duração maior ou igual a todas as outras).

```
SELECT
    * FROM
    Reserva
WHERE
    Duracao_Minutos >= ALL (SELECT Duracao_Minutos FROM Reserva);
```

Q.12: Query livre, isto é, podem elaborar qualquer tipo de query, mas a complexidade da mesma é relevante para avaliação

Lista o Top dos Sócios que mais gastaram em reservas, mostrando o nome, o estatuto e o valor total gasto (Preço base * horas), ordenado do maior para o menor.

```
SELECT
   u.Nome,
   s.Estatuto_Socio,
   SUM((t.Preco_Base_Hora * (r.Duracao_Minutos / 60.0))) AS Total_Gasto_Estimado
FROM
   Reserva r
JOIN
   Utilizador u ON r.ID_Utilizador = u.ID_Utilizador
JOIN
   Socio s ON u.ID_Utilizador = s.ID_Utilizador
JOIN
   Tarifario t ON r.ID_Tarifario = t.ID_Tarifario
GROUP BY
   u.Nome, s.Estatuto_Socio
ORDER BY
   Total_Gasto_Estimado DESC;
```

Stored procedure ou função:

Nota: Foi criada a função fn_calcular_preco_reserva para responder à necessidade de calcular o valor final a pagar. Como o preço depende de variáveis dinâmicas (uso de iluminação, duração do jogo) e da regra de negócio que atribui 30% de desconto aos Sócios, este cálculo não deve ser armazenado estaticamente, mas sim calculado.

```
 1. /*
 2.  * TRABALHO PRÁTICO - Grupo 05
 3.  * Ficheiro: proc_func.sql
 4.  */
 5.
 6. CREATE OR REPLACE FUNCTION fn_calcular_preco_reserva(p_id_reserva INT)
 7. RETURNS DECIMAL(10,2) AS $$
 8. DECLARE
 9.     v_preco_base DECIMAL(10,2);
10.     v_custo_luz DECIMAL(10,2);
11.     v_duracao_min INT;
12.     v_requer_luz BOOLEAN;
13.     v_id_utilizador INT;
14.     v_e_socio BOOLEAN;
15.     v_preco_final DECIMAL(10,2);
16.     v_horas DECIMAL(10,2);
17. BEGIN
18.     -- 1. Obter dados da reserva e tarifário associado
19.     SELECT
20.         r.Duracao_Minutos, r.Requer_Iluminacao, r.ID_Utilizador,
21.         t.Preco_Base_Hora, t.Acrescimo_Iluminacao
22.     INTO
23.         v_duracao_min, v_requer_luz, v_id_utilizador,
24.         v_preco_base, v_custo_luz
25.     FROM Reserva r
26.     JOIN Tarifario t ON r.ID_Tarifario = t.ID_Tarifario
27.     WHERE r.ID_Reserva = p_id_reserva;
28.
29.     -- 2. Converter duração (minutos) para horas decimais
30.     v_horas := v_duracao_min::DECIMAL / 60.0;
31.
```

```
32.     -- 3. Cálculo do valor base: (Preço por Hora * Duração)
33.     v_preco_final := v_preco_base * v_horas;
34.
35.     -- 4. Adicionar custo fixo de iluminação se requisitado
36.     IF v_requer_luz THEN
37.         v_preco_final := v_preco_final + v_custo_luz;
38.     END IF;
39.
40.     -- 5. Aplicação da Regra de Desconto de Sócio (Regra 5)
41.     -- Se o utilizador for Sócio, aplica-se um desconto de 30% sobre o total
42.     SELECT EXISTS(SELECT 1 FROM Socio WHERE ID_Utilizador = v_id_utilizador) INTO
v_e_socio;
43.
44.     IF v_e_socio THEN
45.         v_preco_final := v_preco_final * 0.70;
46.     END IF;
47.
48.     RETURN ROUND(v_preco_final, 2);
49. END;
50. $$ LANGUAGE plpgsql;
51.
```

Vistas:

Nota: Para dar resposta ao requisito funcional de limitação e controlo de acesso aos dados, foram implementadas duas vistas (views) distintas no sistema. A primeira vista, denominada V_Agenda_Courts, tem um caráter operacional e público, tendo sido desenhada para alimentar os ecrãs de informação na receção do clube. O seu principal objetivo é garantir a privacidade dos clientes, ocultando dados sensíveis como o Nome, NIF e Email, e expondo apenas a informação necessária para a gestão de horários (data, duração e identificação do court). Já a segunda vista, V_Contactos_VIP, possui um fim estratégico e foi concebida para auxiliar o departamento de Marketing. Esta estrutura abstrai a complexidade das junções entre tabelas e filtra automaticamente apenas os sócios com estatuto 'Gold', facilitando a extração rápida de listas de contacto para campanhas exclusivas sem que o utilizador tenha de aceder diretamente às tabelas de base.

```
 1. /*
 2.  * TRABALHO PRÁTICO - Grupo 05
 3.  * Ficheiro: views.sql
 4.  */
 5.
 6. CREATE OR REPLACE VIEW V_Agenda_Courts AS
 7. SELECT
 8.     r.DataHora_Inicio,
 9.     r.Duracao_Minutos,
10.     c.Nome_Court,
11.     c.Tipo_Piso
12. FROM Reserva r
13. JOIN Court c ON r.ID_Court = c.ID_Court
14. ORDER BY r.DataHora_Inicio DESC;
15.
16. CREATE OR REPLACE VIEW V_Contactos_VIP AS
17. SELECT
18.     u.Nome,
19.     u.Email,
20.     s.Data_Admissao
21. FROM Utilizador u
22. JOIN Socio s ON u.ID_Utilizador = s.ID_Utilizador
23. WHERE s.Estatuto_Socio = 'Gold';
24.
```