A6: Esquema relacional, validação e afinação

O objetivo principal deste artefacto é apresentar o esquema relacional obtido por mapeamento a partir do modelo conceptual de dados já elaborado. Este esquema relacional vai incluir todos os esquemas de relação, os atributos, as chaves (primárias e estrangeiras), as regras de integridade (*CHECK, UNIQUE, NOT NULL, DEFAULT*), bem como os domínios de cada atributo (ou seja, o tipo que estes vão assumir na base de dados). Serão ainda apresentados domínios adicionais do problema, o *Physical Data Model*, bem como refinamento e normalização dos vários esquemas.

```
Legenda: PK - Primary Key, FK - Foreign Key, UK - UNIQUE, NN - NOT NULL, DF
- DEFAULT, CK - CHECK.
Domínios dos atributos: TEXT, VARCHAR(n) (tamanho máximo), DATE, TIMESTAMP,
INTEGER, REAL.
Todos os campos começados por "id" têm como domínio SERIAL (incrementável) e
são sempre UNIQUE NOT NULL.
```

A6.1 Tabelas de entidades:

Representam o mapeamento das classes do modelo conceitual para o modelo relacional.

NOTA: *dataAtual* pode ser definido como um domínio adicional da seguinte forma - *DATE DEFAULT NOW()*.

Membro(idUtilizador PK, nomeUtilizador UK NN, email UK NN, password NN, dataNascimento NN CK, nomeCivil, tipoMembro NN)					
Chaves candidatas	idUtilizador, nomeUtilizador, e-mail				
Chave primária	idUtilizador				
Integrity rule: UNIQUE	nomeUtilizador, email				
Integrity rule: NOT NULL	nomeUtilizador, email, password, dataNascimento, tipoMembro				
Integrity rule: CHECK	dataAtual - dataNascimento >= 18, (tipoMembro == “Cliente” OR tipo membro == “Admin”)				
nomeUtilizador	**email**	**password**	**dataNascimento**	**nomeCivil**	**tipoMembro**
VARCHAR(20)	TEXT	TEXT	DATE	TEXT	VARCHAR(7)

```
Tabela 1.1 - Entidade Membro
```

MembroBanido(idMembroBanido→Membro PK FK, dataBanido DF NN, duracao, motivo)		
Chave primária	idMembroBanido	
Chave estrangeira	idMembroBanido (proveniente da entidade Membro)	
Integrity rule: NOT NULL	dataBanido	
Integrity rule: DEFAULT	dataBanido.equals(dataAtual)	
dataBanido	**duracao**	**motivo**
TIMESTAMP	INTEGER	TEXT

Tabela 1.2 - Entidade MembroBanido

Leilao(idLeilao PK, nome NN, descricao NN CK, licitacaoBase NN CK, licitacaoAtual DF, dataColocacao NN CK, duracao NN CK, idLeiloeiro→Membro NN FK, idMarca→Marca, idFeedbackLeiloeiro→Feedback FK DF, idFeedbackCliente→Feedback FK DF)					
Chave primária	idLeilao				
Chaves estrangeiras	idLeiloeiro (proveniente de Membro), idMarca (proveniente de Marca), idFeedbackLeiloeiro e idFeedbackCliente(proveniente de Feedback)				
Integrity rule: NOT NULL	nome, descricao, licitacaoBase, dataColocacao, duracao, idLeiloeiro				
Integrity rule: DEFAULT	licitacaoAtual DEFAULT NULL, idFeedbackLeiloeiro DEFAULT NULL, idFeedbackCliente DEFAULT NULL, dataColocacao.equals(dataAtual)				
Integrity rule: CHECK	CHECK(descricao.length > 10 && descricao.length < 2500), CHECK(licitacaoBase > 0.01), CHECK(dataColocacao < dataAtual), CHECK(duracao >= 1.0 && duracao =< 14.0), CHECK(idLeiloeiro is Cliente)				
nome	**descricao**	**licitacaoBase**	**licitacaoAtual**	**dataColocacao**	**duracao**
VARCHAR(50)	VARCHAR(2500)	REAL	REAL	TIMESTAMP	FLOAT

Tabela 1.3 - Entidade Leilão

Mensagem(idMensagem PK, idEmissor→Membro NN FK, idRecetor→Membro NN FK, texto NN CK, data DF NN)	
Chave primária	idMensagem
Chaves estrangeiras	idEmissor, idRecetor (provenientes de Membro)
Integrity rule: NOT NULL	texto, data, idEmissor, idRecetor
Integrity rule: DEFAULT	data.equals(dataAtual)
Integrity rule: CHECK	CHECK(texto.length > 0 && texto.length < 5000)
texto	**data**
VARCHAR(5000)	TIMESTAMP

Tabela 1.4 - Entidade Mensagem

Notificacao(idNotificacao PK, idUtilizador→Membro NN FK, texto NN, data DF NN)	
Chave primária	idNotificacao
Chaves estrangeira	idUtilizador (proveniente de Membro)
Integrity rule: NOT NULL	texto, data, idUtilizador
Integrity rule: DEFAULT	data.equals(dataAtual)
texto	**data**
TEXT	TIMESTAMP

Tabela 1.5 - Entidade Notificação

Imagem(idImagem PK, idLeilao→Leilao NN FK, linkImagem UK NN)	
Chaves candidatas	idImagem, linkImagem
Chave primária	idImagem
Chave estrangeira	idLeilao (proveniente de Leilao)
Integrity rule: UNIQUE	linkImagem
Integrity rule: NOT NULL	linkImagem, idLeilao

Imagem(idImagem PK, idLeilao→Leilao NN FK, linkImagem UK NN)
linkImagem
TEXT

Tabela 1.6 - Entidade Imagem

Licitacao(idLicitacao PK, idLeilao→Leilao NN FK, idMembro→Membro NN FK, valor NN, data DF NN)	
Chave primária	idLicitacao
Chaves estrangeiras	idLeilao (proveniente de Leilao) e idMembro (proveniente de Membro)
Integrity rule: NOT NULL	valor, data, idLeilao, idMembro
Integrity rule: DEFAULT	data.equals(dataAtual)
Integrity rule: CHECK	CHECK(idMembro is Cliente)
valor	**data**
REAL	TIMESTAMP

Tabela 1.7 - Entidade Licitação

Marca(idMarca PK, nome UK NN)	
Chaves candidatas	idMarca, nome
Chave primária	idMarca
Integrity rule: UNIQUE	nome
Integrity rule: NOT NULL	nome
nome	
VARCHAR(20)	

Tabela 1.8 - Entidade Marca

Feedback(idFeedback PK, texto NN CK, data DF NN, valor DF CK)		
Chave primária	idFeedback	
Integrity rule: NOT NULL	texto, data	
Integrity rule: DEFAULT	valor (= 5), data.equals(dataAtual)	
Integrity rule: CHECK	CHECK(texto.length >= 1 && texto.length ⇐ 30), CHECK(valor >= 0 && valor ⇐ 5)	
texto	**data**	**valor**
VARCHAR(30)	TIMESTAMP	INTEGER

Tabela 1.9 - Entidade Feedback

FeedbackComprador(idFeedback→Feedback FK, idComprador→Membro FK)	
Chave primária	(idFeedback, idComprador)
Chaves estrangeiras	idFeedback (proveniente de Feedback), idComprador (proveniente de Membro)

Tabela 1.10 - Entidade FeedbackComprador (feedback atribuído ao vencedor de um leilão pelo dono desse leilão)

FeedbackLeiloeiro(idFeedback→Feedback FK, idLeiloeiro→Membro FK)	
Chave primária	(idFeedback, idLeiloeiro)
Chaves estrangeiras	idFeedback (proveniente de Feedback), idLeiloeiro (proveniente de Membro)

Tabela 1.11 - Entidade FeedbackLeiloeiro (feedback atribuído ao dono de um leilão pelo vencedor desse leilão)

A6.2 Tabelas de relações:

Representam o mapeamento das relações muitos-para-muitos do modelo conceitual para o modelo relacional.

Preferencia(idCliente→Membro FK, idMarca→Marca FK)	
Chave primária	(idCliente, idMarca)
Chaves estrangeiras	idCliente (proveniente de Membro), idMarca (proveniente de Marca)

Tabela 2.1 - Preferência é uma relação de muitos-para-muitos entre Membro (no papel de cliente) e Marca

Registo(idCliente→Membro FK, idLeilao→Leilao FK)	
Chave primária	(idCliente, idLeilao)
Chaves estrangeiras	idCliente (proveniente de Membro), idLeilao (proveniente de Leilao)

Tabela 2.2 - Registo é uma relação de muitos-para-muitos entre Membro (no papel de cliente) e Leilão

A6.3 Dependências funcionais e normalização:

De seguida, são apresentadas todas as dependências funcionais não-triviais para as tabelas em cima referidas:

Membro	idUtilizador → nomeUtilizador, e-mail, password, dataNascimento, nomeCivil
	nomeUtilizador → idUtilizador, e-mail, password, dataNascimento, nomeCivil
	e-mail → idUtilizador, nomeUtilizador, e-mail, password, dataNascimento, nomeCivil
MembroBanido	idMembroBanido → dataBanido, duracao, motivo
Leilao	idLeilao → nome, descricao, licitacaoBase, licitacaoAtual, dataColocacao, duracao, idLeiloeiro, idMarca, idFeedbackLeiloeiro, idFeedbackCliente
Mensagem	idMensagem → idEmissor, idRecetor, texto, data
Notificacao	idNotificacao → idUtilizador, texto, data
Imagem	idImagem → linkImagem
	linkImagem → idImagem
Licitacao	idLicitacao → idLeilao, idMembro, valor, data
Marca	idMarca → nome
	nome → idMarca
Feedback	idFeedback → texto, data, valor

FeedbackComprador	-
FeedbackLeiloeiro	-
Preferencia	-
Registo	-

Tabela 3.1 - Dependências funcionais das tabelas mencionadas

Uma relação encontra-se na **Forma Normal de *Boyce-Codd*** se não existirem dependências entre os atributos não-chave (ou seja, se o lado esquerdo de cada dependência funcional for uma chave) e se todos os atributos não chave dependerem funcionalmente de todas as chaves candidatas (ou seja, se a partir de cada chave é possível percorrer todos os atributos da relação). Como dá para observar pela análise às dependências funcionais, essas condições são reunidas em todas as tabelas, pelo que o esquema já se encontra na Forma Normal de *Boyce-Codd* e não precisa de ser normalizado.

A6.4 Código SQL:

```
-- Função necessária para o constraint confirmaCliente funcionar:

CREATE OR REPLACE FUNCTION confirmaCliente(
        idCliente INTEGER
    )
  RETURNS VARCHAR(5) AS
$confirmaCliente$
BEGIN
 IF EXISTS (SELECT * FROM Membro WHERE idCliente = Membro.idUtilizador AND
Membro.tipoMembro LIKE 'Cliente')
 THEN
    RETURN 'True';
 END IF;
 RETURN 'False';
END;
$confirmaCliente$ LANGUAGE plpgsql;

-- Tabelas

DROP TABLE IF EXISTS Membro CASCADE
;

CREATE TABLE Membro
(
    idUtilizador SERIAL,
    nomeUtilizador VARCHAR(20) NOT NULL,
    email TEXT NOT NULL,
    password TEXT NOT NULL,
    dataNascimento DATE NOT NULL,
    nomeCivil TEXT,
        tipoMembro VARCHAR(7),

    PRIMARY KEY (idUtilizador),
```

```
    UNIQUE (nomeUtilizador),
    UNIQUE (email),
    CONSTRAINT dataCorreta CHECK (EXTRACT(YEAR FROM CURRENT_DATE)-
EXTRACT(YEAR FROM DataNascimento) >= 18),
        CONSTRAINT tipoMembroCorreto CHECK (tipoMembro LIKE 'Admin' OR
tipoMembro LIKE 'Cliente')
)
;

DROP TABLE IF EXISTS MembroBanido CASCADE
;

CREATE TABLE MembroBanido
(

    idMembroBanido INTEGER,
    dataBanido TIMESTAMP NOT NULL DEFAULT NOW(),
    duracao INTEGER,
    motivo TEXT,

    PRIMARY KEY (idMembroBanido),
    FOREIGN KEY (idMembroBanido)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Mensagem CASCADE
;

CREATE TABLE Mensagem
(
    idMensagem SERIAL,
    idEmissor INTEGER NOT NULL,
    idRecetor INTEGER NOT NULL,
    texto VARCHAR(5000) NOT NULL,
    dataMensagem TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idMensagem),
    FOREIGN KEY (idRecetor)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idEmissor)
        REFERENCES Membro(idUtilizador),
    CONSTRAINT comprimentoMensagem CHECK (CHAR_LENGTH(texto) >  AND
CHAR_LENGTH(texto) < 5000)
)
;

DROP TABLE IF EXISTS Notificacao CASCADE
;

CREATE TABLE Notificacao
```

```
(
    idNotificacao SERIAL,
    idUtilizador INTEGER NOT NULL,
    texto TEXT NOT NULL,
    dataNotificacao TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idNotificacao),
    FOREIGN KEY (idUtilizador)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Marca CASCADE
;

CREATE TABLE Marca
(
    idMarca SERIAL,
    nome VARCHAR(20) NOT NULL,

    PRIMARY KEY (idMarca),
    UNIQUE(nome)
)
;

DROP TABLE IF EXISTS Feedback CASCADE
;

CREATE TABLE Feedback
(
    idFeedback SERIAL,
    texto VARCHAR(30) NOT NULL,
    dataFeedback TIMESTAMP NOT NULL DEFAULT NOW(),
    valor INTEGER DEFAULT 5,

    PRIMARY KEY (idFeedback),
    CONSTRAINT tamanhoFeedback CHECK (CHAR_LENGTH(texto) >= 1 AND
CHAR_LENGTH(texto) <= 30),
    CONSTRAINT valorFeedback CHECK (valor >=  AND valor <= 5)
)
;

DROP TABLE IF EXISTS FeedbackComprador CASCADE
;

CREATE TABLE FeedbackComprador
(
    idFeedback INTEGER,
    idComprador INTEGER,

    PRIMARY KEY (idFeedback, idComprador),
```

```
    FOREIGN KEY (idComprador)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idFeedback)
        REFERENCES Feedback(idFeedback)
)
;

DROP TABLE IF EXISTS FeedbackLeiloeiro CASCADE
;

CREATE TABLE FeedbackLeiloeiro
(
    idFeedback INTEGER,
    idLeiloeiro INTEGER,

    PRIMARY KEY (idFeedback, idLeiloeiro),
    FOREIGN KEY (idLeiloeiro)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idFeedback)
        REFERENCES Feedback(idFeedback)
)
;

DROP TABLE IF EXISTS Leilao CASCADE
;

CREATE TABLE Leilao
(
    idLeilao  SERIAL,
    nome VARCHAR(50) NOT NULL,
    descricao VARCHAR(5000) NOT NULL,
    licitacaoBase FLOAT NOT NULL,
    licitacaoAtual FLOAT DEFAULT NULL,
    dataColocacao TIMESTAMP NOT NULL DEFAULT NOW(),
    duracao FLOAT NOT NULL,
    idLeiloeiro INTEGER NOT NULL,
    idMarca INTEGER,
    idFeedbackLeiloeiro INTEGER DEFAULT NULL,
    idFeedbackCliente INTEGER DEFAULT NULL,

    PRIMARY KEY (idLeilao),
    FOREIGN KEY (idLeiloeiro)
        REFERENCES Membro(idUtilizador),
    FOREIGN KEY (idMarca)
        REFERENCES Marca(idMarca),
    FOREIGN KEY (idFeedbackLeiloeiro)
        REFERENCES Feedback(idFeedback),
    FOREIGN KEY (idFeedbackCliente)
        REFERENCES Feedback(idFeedback),
    CONSTRAINT tamanhoDescricao CHECK (CHAR_LENGTH(descricao) > 10 AND
CHAR_LENGTH(descricao) < 2500),
```

```
    CONSTRAINT valorLicitacaoBase CHECK (licitacaoBase > 0.01),
    CONSTRAINT dataCorreta CHECK (dataColocacao <= NOW()),
    CONSTRAINT duracaoLeilao CHECK (duracao >= 1.0 AND duracao <= 14.0),
        CONSTRAINT confirmaCliente CHECK (confirmaCliente(idLeiloeiro) =
'True')
)
;

DROP TABLE IF EXISTS Imagem CASCADE
;

CREATE TABLE Imagem
(
    idImagem SERIAL,
    idLeilao INTEGER NOT NULL,
    link TEXT NOT NULL,

    PRIMARY KEY (idImagem),
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao),
    UNIQUE(link)
)
;

DROP TABLE IF EXISTS Licitacao CASCADE
;

CREATE TABLE Licitacao
(
    idLicitacao SERIAL,
    idLeilao INTEGER NOT NULL,
    idCliente INTEGER NOT NULL,
    valor FLOAT NOT NULL,
    dataLicitacao TIMESTAMP NOT NULL DEFAULT NOW(),

    PRIMARY KEY (idLicitacao),
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador) ,
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao)
        ON DELETE CASCADE,

        CONSTRAINT confirmaCliente CHECK ( confirmaCliente(idCliente) =
'True' )
)
;

DROP TABLE IF EXISTS Preferencias CASCADE
;

CREATE TABLE Preferencias
```

```
(
    idCliente INTEGER,
    idMarca INTEGER,

    PRIMARY KEY (idCliente , idMarca),
    FOREIGN KEY (idMarca)
        REFERENCES Marca(idMarca) ,
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador)
)
;

DROP TABLE IF EXISTS Registo CASCADE
;

CREATE TABLE Registo
(
    idCliente INTEGER,
    idLeilao INTEGER,

    PRIMARY KEY (idCliente , idLeilao),
    FOREIGN KEY (idLeilao)
        REFERENCES Leilao(idLeilao) ,
    FOREIGN KEY (idCliente)
        REFERENCES Membro(idUtilizador)
)
;
```

— *AutoLeilões, Lda.*

[Voltar à página principal]