Esquema relacional, validação e schema refinement

Physical Data Model

O diagrama seguinte, *Physical Data Model*, representa o esquema interno da base de dados. Apresenta as entidades com os respetivos atributos e as relações entre estas entidades.

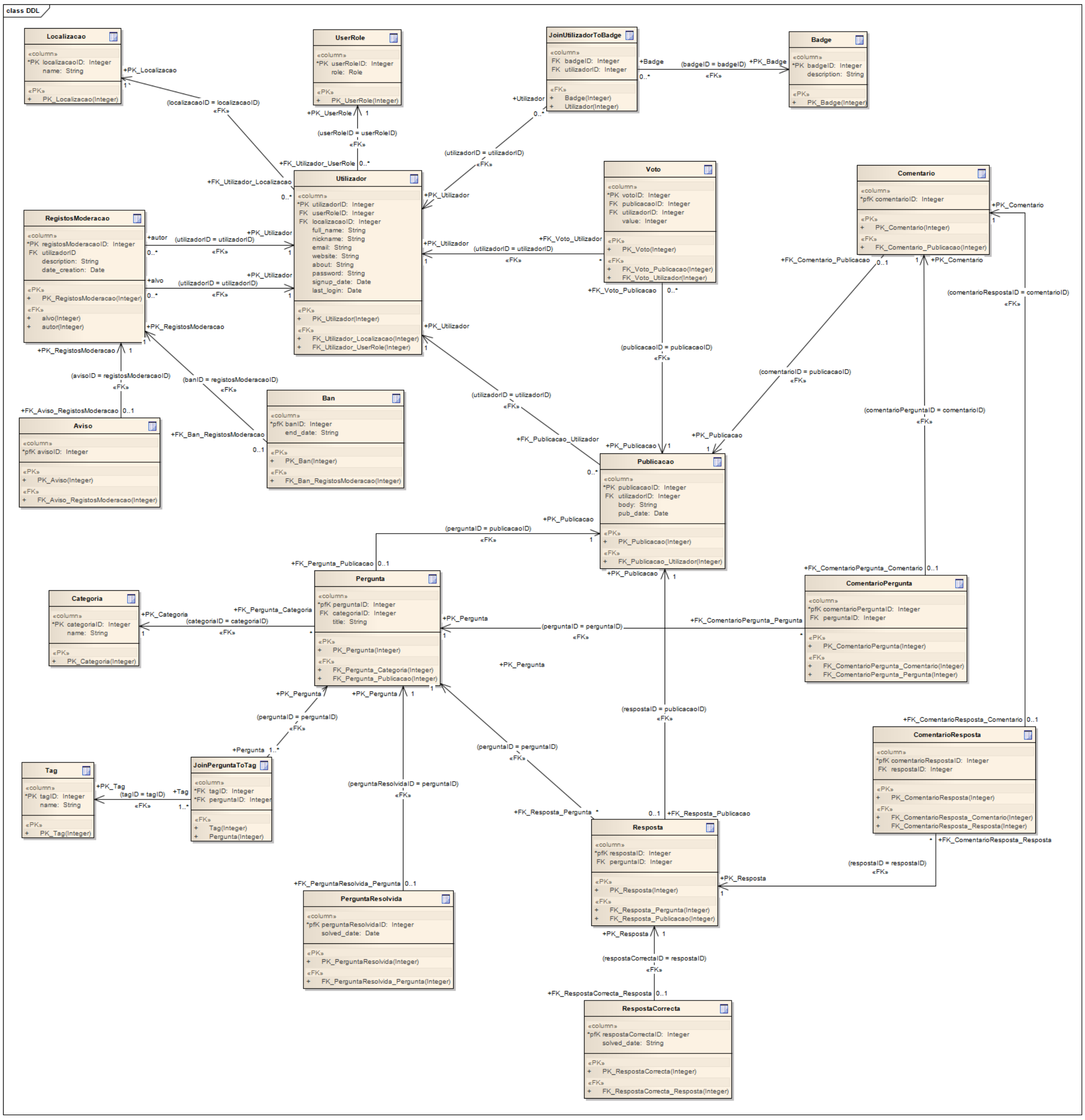

Figura 1: Physical Data Model

Esquema Relacional

#	Name	Relation
R01	Localizacao	Localizacao(<u>localizacaoID</u>, name NN UK)
R02	UserRole	UserRole(<u>userRoleID</u>, role NN)
R03	RegistosModeracao	RegistosModeracao(<u>registosModeracaoID</u>, utilizadorID → Utilizador NN, description NN, date_creation NN)
R04	Aviso	Aviso(<u>avisoID</u> → RegistosModeracao)
R05	Ban	Ban(<u>banID</u> → RegistosModeracao, end_date NN)
R06	Utilizador	Utilizador(<u>utilizadorID</u>, userRoleID → UserRole NN, localizacaoID → Localizacao, full_name NN, nickname UK NN , email UK NN, website, about, password NN, signup_date NN, last_login NN)
R07	UserBadge	UserBadge(<u>badgeID</u> → Badge, <u>utilizadorID</u> → Utilizador)
R08	Badge	Badge(<u>badgeID</u>, description NN)
R09	Publicacao	Publicacao(<u>publicacaoID</u>, utilizadorID → Utilizador NN, body NN, pub_date NN)
R10	Voto	Voto(<u>votoID</u>, publicacaoID → Publicacao NN, value NN)
R11	Pergunta	Pergunta(<u>perguntaID</u> → Publicacao, categoriaID → Categoria NN, title NN, solved_date)
R12	Resposta	Resposta(<u>respostaID</u> → Publicacao, perguntaID → Pergunta NN, solved_date)
R13	Comentario	Comentario(<u>comentarioID</u> → Publicacao)
R14	ComentarioPergunta	ComentarioPergunta(<u>comentarioPerguntaID</u> → Comentario, perguntaID → Pergunta NN)
R15	ComentarioResposta	ComentarioResposta(<u>comentarioRespostaID</u> → Comentario, respostaID → Resposta NN)
R16	Categoria	Categoria(<u>categoriaID</u>, name UK NN)
R17	QuestionTag	QuestionTag(<u>tagID</u> → Tag, <u>perguntaID</u> → Pergunta)
R18	Tag	Tag(<u>tagID</u>, name UK NN)

Tabela 1: Especificação do esquema relacional.

Dependências Funcionais

#	Name	Chaves Candidatas	Dependências Funcionais
R01	Localizacao	localizacaoID name	DF1: localizacaoID → name DF2: name → localizacaoID
R02	UserRole	userRoleID	DF3: userRoleID → role
R03	RegistosModeracao	registosModeracaoID	DF4: registosModeracaoID → utilizadorID, description, date_creation
R05	Ban	banID	DF5: banID → end_date
R06	Utilizador	utilizadorID nickname email	DF6: utilizadorID → userRoleID, localizacaoID, localizacao, full_name, nickname, email, website, about, password, signup_date, last_login DF7: nickname → utilizadorID, userRoleID, localizacaoID, localizacao, full_name, email, website, about, password, signup_date, last_login DF8: email → utilizadorID, userRoleID, localizacaoID, localizacao, full_name, nickname, website, about, password, signup_date, last_login

#	Name	Chaves Candidatas	Dependências Funcionais
R08	Badge	badgeID	DF9: badgeID → description
R09	Publicacao	publicacaoID	DF10: publicacaoID → utilizadorID, body, pub_date
R10	Voto	votoID	DF11: votoID → publicacaoID, value
R11	Pergunta	perguntaID	DF12: perguntaID → categoriaID, title, solved_date
R12	Resposta	respostaID	DF13: respostaID → perguntaID, solved_date
R14	ComentarioPergunta	comentarioPerguntaID	DF14: comentarioPerguntaID → perguntaID
R15	ComentarioResposta	comentarioRespostaID	DF15: comentarioRespostaID → respostaID
R16	Categoria	categoriaID name	DF16: categoriaID → name DF17: name → categoriaID
R18	Tag	tagID name	DF18: tagID → name DF19: name → tagID

Tabela 2: Especificação das dependências funcionais.

As tabelas UserBadge(R7) e QuestionTag(R17) não têm nenhuma dependência funcional relevante, uma vez que têm dois atributos, ambos foreign keys, que constituem a respectiva primary key. Já as tabelas Aviso(R4) e Comentario(R13) contêm apenas um atributo, a sua primary key, pelo que também não estão representadas na Tabela 2.

Domínios

role	ENUM('Admin','Editor','Autenticado')
value	ENUM(-1,0,1)

Tabela 3: Especificação dos domínios.

Validação Esquema Relacional

O esquema relacional apresentado encontra-se normalizado e na Forma Normal de Boyce-Codd. Para cada dependência funcional, do esquema, do tipo X → Y verifica-se sempre pelo menos uma das seguintes duas condições: é dependência funcional trivial (Y ⊆ X) ou X é superchave do esquema.

SQL Scripts

delete.sql

```sql
DROP TABLE IF EXISTS "Warning" CASCADE;
DROP TABLE IF EXISTS "Badge" CASCADE;
DROP TABLE IF EXISTS "Ban" CASCADE;
DROP TABLE IF EXISTS "Category" CASCADE;
DROP TABLE IF EXISTS "Comment" CASCADE;
DROP TABLE IF EXISTS "Question" CASCADE;
DROP TABLE IF EXISTS "UserBadge" CASCADE;
DROP TABLE IF EXISTS "Location" CASCADE;
DROP TABLE IF EXISTS "Publication" CASCADE;
DROP TABLE IF EXISTS "ModRegister" CASCADE;
DROP TABLE IF EXISTS "Answer" CASCADE;
DROP TABLE IF EXISTS "Tag" CASCADE;
DROP TABLE IF EXISTS "UserRole" CASCADE;
DROP TABLE IF EXISTS "User" CASCADE;
DROP TABLE IF EXISTS "Vote" CASCADE;
DROP TABLE IF EXISTS "QuestionTag" CASCADE;
DROP TABLE IF EXISTS  "CommentAnswer" CASCADE;
DROP TABLE IF EXISTS  "CommentQuestion" CASCADE;
```

create.sql

```
CREATE TABLE "Warning"
(
    warningid SERIAL PRIMARY KEY,
    CONSTRAINT "FK_Warning_ModRegister"
        FOREIGN KEY ("warningid") REFERENCES "ModRegister"
("modregisterid") ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "Badge"
(
    badgeid SERIAL PRIMARY KEY,
    description VARCHAR(100) NOT NULL,
    CONSTRAINT badge_description CHECK(CHAR_LENGTH(description) >= 10
AND CHAR_LENGTH(description) <= 100)
);

CREATE TABLE "Ban"
(
    banid SERIAL PRIMARY KEY,
    end_date TIMESTAMP,
    CONSTRAINT "FK_Ban_ModRegister"
       FOREIGN KEY ("banid") REFERENCES "ModRegister" ("modregisterid")
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "Category"
(
    categoryid SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    CONSTRAINT valid_category CHECK(CHAR_LENGTH(name) >= 3 AND
CHAR_LENGTH(name) <= 50)
);

CREATE TABLE "Comment"
(
    commentid SERIAL PRIMARY KEY,
    CONSTRAINT "FK_Comment_Publication"
        FOREIGN KEY ("commentid") REFERENCES "Publication"
("publicationid") ON DELETE CASCADE ON UPDATE CASCADE
```

```
);

CREATE TABLE "CommentQuestion"
(
    commentid SERIAL PRIMARY KEY,
    questionid INTEGER NOT NULL,
    CONSTRAINT "FK_CommentQuestion_Comment"
        FOREIGN KEY ("commentid") REFERENCES "Comment" ("commentid") ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT "FK_CommentQuestion_Question"
        FOREIGN KEY ("questionid") REFERENCES "Question" ("questionid")
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "CommentAnswer"
(
    commentid SERIAL PRIMARY KEY,
    answerid INTEGER NOT NULL,
    CONSTRAINT "FK_CommentAnswer_Comment"
        FOREIGN KEY ("commentid") REFERENCES "Comment" ("commentid") ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT "FK_CommentAnswer_Answer"
        FOREIGN KEY ("answerid") REFERENCES "Answer" ("answerid") ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "Location"
(
    locationid SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE "Question"
(
    questionid SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    categoryid INTEGER NOT NULL,
    solved_date TIMESTAMP,
    CONSTRAINT title_length CHECK (CHAR_LENGTH(title) >= 3 AND
CHAR_LENGTH(title) <= 50),
    CONSTRAINT "FK_Question_Category"
        FOREIGN KEY ("categoryid") REFERENCES "Category" ("categoryid")
ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT "FK_Question_Publication"
        FOREIGN KEY ("questionid") REFERENCES "Publication"
("publicationid") ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE "Publication"
(
    publicationid SERIAL PRIMARY KEY,
    body VARCHAR(1000) NOT NULL ,
    creation_date TIMESTAMP DEFAULT now() NOT NULL,
    userid INTEGER NOT NULL,
    CONSTRAINT body_length CHECK (CHAR_LENGTH(body) >= 10 AND
CHAR_LENGTH(body) <= 1000),
    CONSTRAINT "FK_Publication_User"
        FOREIGN KEY ("userid") REFERENCES "User" ("userid") ON DELETE
SET NULL ON UPDATE CASCADE
);

CREATE TABLE "ModRegister"
(
    modregisterid SERIAL PRIMARY KEY,
    date_creation TIMESTAMP DEFAULT now() NOT NULL,
    reason VARCHAR(200) NOT NULL,
    userid_author INTEGER NOT NULL,
    userid_target INTEGER NOT NULL,
    CONSTRAINT author
        FOREIGN KEY ("userid_author") REFERENCES "User" ("userid") ON
DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT target
        FOREIGN KEY ("userid_target") REFERENCES "User" ("userid") ON
DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "Answer"
(
    answerid SERIAL PRIMARY KEY,
    questionid INTEGER NOT NULL,
    solved_date TIMESTAMP,
    CONSTRAINT "FK_Answer_Question"
        FOREIGN KEY ("questionid") REFERENCES "Question" ("questionid")
ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT "FK_Answer_Publication"
        FOREIGN KEY ("answerid") REFERENCES "Publication"
("publicationid") ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "Tag"
(
    tagid SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    CONSTRAINT valid_tag CHECK(CHAR_LENGTH(name) >= 3 AND
CHAR_LENGTH(name) <= 20)
);

CREATE TABLE "UserRole"
(
    roleid SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    CONSTRAINT user_role CHECK(name IN ('Admin', 'Editor',
'Authenticated'))
);

CREATE TABLE "User"
(
    userid SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    email VARCHAR(70) NOT NULL,
    password VARCHAR(50) NOT NULL,
    fullname VARCHAR(200),
    about VARCHAR(500),
    website VARCHAR(150),
    signup_date DATE DEFAULT CURRENT_DATE NOT NULL,
    last_login TIMESTAMP,
    locationid INTEGER NOT NULL,
    roleid INTEGER NOT NULL,
    CONSTRAIN valid_date CHECK(last_login > signup_date),
    CONSTRAINT valid_password CHECK(CHAR_LENGTH(password) >= 8 AND
CHAR_LENGTH(password) < 50),
    CONSTRAINT valid_username CHECK(CHAR_LENGTH(username) >= 1 AND
CHAR_LENGTH(username) < 20),
    CONSTRAINT valid_fullname CHECK(CHAR_LENGTH(fullname) >= 6 AND
CHAR_LENGTH(fullname) <= 50),
    CONSTRAINT valid_email CHECK(CHAR_LENGTH(email) >= 6 AND
CHAR_LENGTH(email) <= 50),
    CONSTRAINT "FK_User_Location"
        FOREIGN KEY ("locationid") REFERENCES "Location" ("locationid")
ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT "FK_User_UserRole"
        FOREIGN KEY ("roleid") REFERENCES "UserRole" ("roleid") ON
DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE "Vote"
(
    voteid SERIAL PRIMARY KEY,
    VALUE INTEGER DEFAULT  NOT NULL,
    publicationid INTEGER NOT NULL,
    userid INTEGER NOT NULL,
    CONSTRAINT vote_values CHECK(VALUE =  OR VALUE = 1 OR VALUE = -1),
    CONSTRAINT "FK_Vote_Publication"
        FOREIGN KEY ("publicationid") REFERENCES "Publication"
("publicationid") ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT "FK_Vote_User"
        FOREIGN KEY ("userid") REFERENCES "User" ("userid") ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE "QuestionTag" (
    questionid INTEGER NOT NULL,
    tagid INTEGER NOT NULL,
    PRIMARY KEY(questionid,tagid),
    CONSTRAINT "Tag"
        FOREIGN KEY ("tagid") REFERENCES "Tag" ("tagid") ON DELETE
CASCADE ON UPDATE CASCADE,
    CONSTRAINT "Question"
        FOREIGN KEY ("questionid") REFERENCES "Question" ("questionid")
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE "UserBadge" (
    userid INTEGER NOT NULL,
    badgeid INTEGER NOT NULL,
    PRIMARY KEY(userid,badgeid),
    CONSTRAINT "Badge"
        FOREIGN KEY ("badgeid") REFERENCES "Badge" ("badgeid") ON
DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT "User"
        FOREIGN KEY ("userid") REFERENCES "User" ("userid") ON DELETE
CASCADE ON UPDATE CASCADE
);
```

Revisão

- remoção check constraints modelo relacional
- melhor especificação de domínios
- coluna identificativa relações esquema relacional
- constraints e primary keys nas tabelas
- primary keys nas tabelas QuestionTag e UserBadge
- alterado ON DELETE CASCADE para ON DELETE SET NULL nos casos apropriados
- especificadas chaves candidatas e dependências funcionais