# A5: Esquema relacional, Validação e Afinação

Este artefacto contém o Esquema Relacional obtido através do mapeamento do Modelo Concetual.

## Esquema Relacional

| |
|---|
| User(id, bio, email UK NN, photo, name NN, password NN, lastUpdatedNotification NN) |
| BannedUser(userId → User, reason NN, expires NN) |
| Moderator(userId → User, promotedById → User NN) |
| PrivilegeLevel(id, canCreateQuestion NN, canEditOwnContent NN, canEditAnyContent NN, canCloseOwnQuestion NN, canCloseAnyQuestion NN, canBanUser NN, canPromoteToAdmin NN, canPromoteToModerator NN, canReply NN, canVote NN, canEditProfile NN, canFollowContent NN, canDeleteOwnContent NN, canDeleteAnyContent NN, canChangeOwnQuestionTag NN, canChangeAnyQuestionTag NN, canSelectBestAnswerOnOwnQuestion NN) |
| Content(id, creatorId → User NN, creationDate NN, text NN, deleted NN) |
| Question(contentId → Content, title NN, closed NN) |
| Reply(contentId → Content, parentId → Content NN) |
| Tag(id, name NN) |
| PendingTag(id, name NN) |
| QuestionTags(contentId → Content, tagId → Tag) |
| Vote(id, userId → User NN, contentId → Content NN, positive NN) |
| Notification(id, userId → User NN, date NN, read NN, voteId → Vote, replyId → Reply) |
| FollowContent(contentId → Content, followerId → User) |

Tabela 1: Esquema relacional. Legenda: NN - NOT NULL; UK - UNIQUE.

## Domínios

A especificação de domínios adicionais também pode ser feita de uma forma compacta, usando a notação:

| | |
|---|---|
| ban_reason | ENUM('0','1','2','3') (0 → “bad_language”, 1 → “inappropriate_content”, 2→ “spam”, 3→ “disrespect_against_others”) |

## Restrições

Alguns atributos das tabelas têm condições que precisam de ser verificadas, segundo as restrições abaixo:

| Tabela | Restrição |
|---|---|
| Notification | Cada entrada da tabela Notification tem um e só um dos atributos voteId e replyId atribuídos |

## Código SQL

lbaw1612_create.sql

```
CREATE TYPE ban_reason AS ENUM (
    'bad_language',
    'inappropriate_content',
    'spam',
    'disrespect_against_others'
);

CREATE TABLE USER (
    id INTEGER NOT NULL,
    bio text,
    email text NOT NULL,
    photo text,
    name text NOT NULL,
    password text NOT NULL,
    privilegeLevelId INTEGER REFERENCES PrivilegeLevel(id) NOT NULL
    PRIMARY KEY (id)
);

CREATE TABLE BannedUser (
    userId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    explanation text,
    expires DATE NOT NULL,
    reason ban_reason NOT NULL,
    PRIMARY KEY (userId)
);

CREATE TABLE Moderator (
    userId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    promotedById INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    PRIMARY KEY (userId)
);

CREATE TABLE PrivilegeLevel (
    id INTEGER NOT NULL,
    canCreateQuestion BOOLEAN NOT NULL,
    canEditOwnContent BOOLEAN NOT NULL,
    canEditAnyContent BOOLEAN NOT NULL,
    canCloseOwnQuestion BOOLEAN NOT NULL,
    canCloseAnyQuestion BOOLEAN NOT NULL,
    canBanUser BOOLEAN NOT NULL,
    canPromoteToAdmin BOOLEAN NOT NULL,
    canPromoteToModerator BOOLEAN NOT NULL,
    canReply BOOLEAN NOT NULL,
    canVote BOOLEAN NOT NULL,
    canEditProfile BOOLEAN NOT NULL,
    canFollowContent BOOLEAN NOT NULL,
    canDeleteOwnContent BOOLEAN NOT NULL,
    canDeleteAnyContent BOOLEAN NOT NULL,
    canChangeOwnQuestionTag BOOLEAN NOT NULL,
    canChangeAnyQuestionTag BOOLEAN NOT NULL,
    canSelectBestAnswerOnOwnQuestion BOOLEAN NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE Content (
    id INTEGER NOT NULL,
    creatorId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    creationDate DATE NOT NULL,
    text text NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE Question (
    contentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    title text NOT NULL,
    closed INTEGER NOT NULL,
    PRIMARY KEY (contentId);
);

CREATE TABLE Reply (
    contentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    parentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    PRIMARY KEY (contentId)
);

CREATE TABLE Tag (
    id INTEGER NOT NULL,
    name text NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE PendingTag (
    id INTEGER NOT NULL,
    name text NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE QuestionTags (
    contentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    tagId INTEGER REFERENCES Tag(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    PRIMARY KEY (contentId,tagId);
```

```
);

CREATE TABLE Vote (
    id INTEGER NOT NULL,
    userId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    contentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    positive BOOLEAN NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE Notification (
    id INTEGER NOT NULL,
    userId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    DATE DATE NOT NULL,
    voteId INTEGER REFERENCES Vote(id) ON UPDATE CASCADE ON DELETE
CASCADE,
    replyId INTEGER REFERENCES Reply(contentId) ON UPDATE CASCADE ON
DELETE CASCADE,
    READ BOOLEAN,
    PRIMARY KEY (id)
);

CREATE TABLE FollowContent (
    contentId INTEGER REFERENCES Content(id) ON UPDATE CASCADE ON
DELETE CASCADE NOT NULL,
    followerId INTEGER REFERENCES USER(id) ON UPDATE CASCADE ON DELETE
CASCADE NOT NULL,
    PRIMARY KEY (contentId, followerId)
);
```

## Validação

Iremos em seguida, apresentar todas as dependências funcionais da base de dados e vamos também verificar, se estas dependências se encontram na forma normal de Boyce-Codd.

Na tabela **User** :

| |
|---|
| <u>id</u> → id, bio, email, photo, name, password, lastUpdatedNotification |
| <u>email</u> → id, bio, email, photo, name, password, lastUpdatedNotification |

Na tabela **BannedUser** :

| |
|---|
| <u>userId</u> → userId, reason, expires |

Na tabela **Moderator** :

<u>userId</u> → userId, promotedById

Na tabela **PrivilegeLevel** :

<u>id</u> → id, canCreateQuestion, canEditOwnContent, canEditAnyContent, canCloseOwnQuestion, canCloseAnyQuestion, canBanUser, canPromoteToAdmi, canPromoteToModerator, canReply, canVote, canEditProfile, canFollowContent, canDeleteOwnContent, canDeleteAnyContent, canChangeOwnQuestionTag, canChangeAnyQuestionTag, canChangeAnyQuestionTag

Na tabela **Content** :

<u>id</u> → id, creatorId, creationDate, text, deleted

Na tabela **Question** :

<u>contentId</u> → contentId, title, closed

Na tabela **Reply** :

<u>contentId</u> → contentId, parentId

Na tabela **Tag** :

<u>id</u> → id, name

Na tabela **PendingTag** :

<u>id</u> → id, name

Na tabela **QuestionTags** :

<u>contentId</u>, <u>tagId</u> → contentId, tagId

Na tabela **Vote** :

<u>id</u> → id, userId, contentId, positive

Na tabela **Notification** :

<u>id</u> → id, userId, date, read, voteId, replyId

Na tabela **FollowContent** :

<u>contentId</u>, <u>followerId</u> → contentId, followerId

Depois de analisar todas as chaves de todas as tabelas, verificámos que ou são super-chaves, ou em cada chave (X → Y), Y é um subconjunto de X. Concluindo assim, que as tabelas se encontram na forma normal de Boyce-Codd.