Code Home

A5: Relational schema, validation and schema refinement

Relational Schema

Relation schemas are specified in the compact notation:

R01	post (id, content **NN**, date **NN DF** Today, isVisible **NN**, points **NN DF** 0)
R02	answer (postID -> Post, isCorrect **NN**)
R03	question (postID -> Post¸ isClosed **NN**, nViews **NN DF** 0 **CK** > 0, tittle **NN**)
R04	postVote (postID -> Post, posterID -> user, value **NN CK** (== 1 OR == -1))
R05	postReport (postID -> Post, reporterID -> user, date **DF** Today)
R06	faqEntry (id, question **NN**, answer **NN**)
R07	user (id, username **NN**, pass_token, type **NN DF** REGULAR, auth_type **CK** auth_type=0 **OR** auth_type=1, email **UK NN**, state **NN DF** Active, description, img_path, points **NN DF** 0)
R08	contact (id, Message **NN**, userID->user, subjectID->subject **NN**)
R09	subject (id, Name **NN UK**)
R10	banInfo (id, duration, description **NN**, initDate **DF** Today, endDate **CK** endDate -> endDate != NULL **OR** isPermanent=True, isPermanent **NN**, userID->user **NN**, adminID -> user **CK** user. isAdmin == True **NN**)
R11	tagQuestion(tag_id->tag **NN**, question_id -> question **NN**)
R12	tag (id, name **NN UK**)
R13	team (id, name **NN UK**)
R14	teamMember (id, name **NN UK,** title **NN,** email **UK NN**, joinDate **NN**, img_path)
R15	teamToTeamMember (team_id->team **NN**, teamMember_id -> teamMember **NN**)

where UK means UNIQUE KEY, NN means NOT NULL, DF means DEFAULT and CK means CHECK.

GROUP1751, 19/03/2018

Domains

Specification of additional domains:

Today	DATE DEFAULT CURRENT_DATE
User Types	ENUM ('REGULAR', 'ADMINISTRATION')
user States	ENUM ('ACTIVE, 'BANNED')

Functional Dependencies and schema validation

To validate the Relational Schema obtained from the Conceptual Model, all functional dependencies are identified, and the normalization of all relation schemas is accomplished.

Table R01 (post)	
Keys: {id}	
Functional Dependencies	
FD0101	{id} → {content, date, isVisible, points}
NORMAL FORM	BCNF

Table R02 (answer)	
Keys: {postId}	
Functional Dependencies	
FD0201	{postId} → {isCorrect}
NORMAL FORM	BCNF

Table R03 (question)	
Keys: {postId}	
Functional Dependencies	
FD0301	{postId} → {isClosed, nViews, title}
NORMAL FORM	BCNF

Table R04 (postVote)	
Keys: {postId, posterId}	
Functional Dependencies	
FD0401	{postId, posterId} → {value}
NORMAL FORM	BCNF

Table R05 (postReport)	
Keys: {postId, reporterId}	
Functional Dependencies	
FD0501	{postId, reporterId} → {date}
NORMAL FORM	BCNF

Table R06 (faqEntry)	
Keys: {id}	
Functional Dependencies	
FD0601	{Id} → {question, answer}
NORMAL FORM	BCNF

Table R07 (user)	
Keys: {id}	
Functional Dependencies	
FD0701	{id} → {username, pass_token, type, auth_type, email, stateId->state, description, img_path, points}
FD0702	{username} → {id, pass_token, type, auth_type, email, stateId->state, description, img_path, points}
FD0703	{email} → {id, username, pass_token, type, auth_type, stateId->state, description, img_path, points}
NORMAL FORM	BCNF

Table R08 (contact)	
Keys: {id}	
Functional Dependencies	
FD0801	{id} → {message, userId->user, subjectId->subject}
NORMAL FORM	BCNF

Table R09 (subject)	
Keys: {id}	
Functional Dependencies	
FD0901	{id} → {name}
NORMAL FORM	BCNF

Table R10 (banInfo)	
Keys: {id}	
Functional Dependencies	
FD1001	{id} → {banDuration, description, initDate, endDate, isPermanent, userId->user NN, adminId->user}
NORMAL FORM	BCNF

Table R11 (tagQuestion)	
Keys: {tag_id, question_id}	
Functional Dependencies	
	(none)
NORMAL FORM	BCNF

Table R12 (tag)	
Keys: {id}	
Functional Dependencies	
FD1201	{id} → {name}
NORMAL FORM	BCNF

Table R13 (team)	
Keys: {id}	
Functional Dependencies	
FD1201	{id} → {name}
NORMAL FORM	BCNF

Table R14 (teamMember)	
Keys: {id}	
Functional Dependencies	
FD1201	{id} → {name, email, img_path, joinDate, title}
NORMAL FORM	BCNF

Table R15 (teamToTeamMember)	
Keys: {team_id, teamMember_id}	
Functional Dependencies	
	(none)
NORMAL FORM	BCNF

As all relations schemas are in the Boyce–Codd Normal Form (BCNF), the relational schema is also in the BCNF and therefore there is no need to be refined using normalization.

SQL Code

```
CREATE TABLE "User" (
    id                  SERIAL CONSTRAINT userPK PRIMARY KEY,
    username            TEXT NOT NULL,
    type                TEXT NOT NULL DEFAULT 'REGULAR',
    pass_token          TEXT NOT NULL,
    auth_type           INTEGER NOT NULL,
    CHECK               (auth_type = 0 OR auth_type = 1),
    email               TEXT NOT NULL UNIQUE,
    state               TEXT NOT NULL DEFAULT 'ACTIVE',
    description         TEXT,
    img_path            TEXT NOT NULL DEFAULT '0.png',
    points              INTEGER NOT NULL DEFAULT 0
);

CREATE TABLE Subject(
    subjectID           SERIAL CONSTRAINT  subjectPK PRIMARY KEY,
    name                TEXT NOT NULL UNIQUE
);

CREATE TABLE Contact(
    id                  SERIAL CONSTRAINT contactPK PRIMARY KEY,
    message             TEXT NOT NULL,
    userID              INTEGER NOT NULL REFERENCES "User",
    subjectID           INTEGER NOT NULL REFERENCES Subject
);

CREATE TABLE BanInfo(
    id                  SERIAL CONSTRAINT banPK PRIMARY KEY,
    duration            BIGINT,
    description         TEXT NOT NULL,
    isPermanent         BOOLEAN NOT NULL ,
    initDate            TIMESTAMP WITH TIME zone DEFAULT now(),
    endDate             TIMESTAMP WITH TIME zone,
    CHECK               (((endDate IS NOT NULL AND endDate > now()) OR
isPermanent IS TRUE )),
    userID              INTEGER NOT NULL REFERENCES "User",
    adminID             INTEGER NOT NULL REFERENCES "User"
);

CREATE FUNCTION adminCheckProcedure() RETURNS TRIGGER AS $$
    BEGIN
        if (not((SELECT type from User where userID = NEW.adminID)='admin'))
THEN
            RAISE EXCEPTION 'User must be admin to ban';
        END IF
        RETURN NEW;
    END
$$ language plpgsql;

CREATE TRIGGER adminCheckTrigger
    BEFORE INSERT OR UPDATE on BanInfo
    EXECUTE PROCEDURE adminCheckProcedure();

CREATE TABLE Tag(
    id              SERIAL CONSTRAINT tagPK PRIMARY KEY,
    name            TEXT NOT NULL UNIQUE
```

```
);

CREATE TABLE Post (
       id          SERIAL CONSTRAINT postpk PRIMARY KEY,
       content     text NOT NULL,
       "date"      TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
       isVisible   boolean NOT NULL,
       points      INTEGER DEFAULT 0 NOT NULL
);

CREATE TABLE Answer (
       postID      SERIAL REFERENCES Post CONSTRAINT answerpk PRIMARY KEY,
       isCorrect   boolean NOT NULL
);

CREATE TABLE Question  (
       postID      SERIAL REFERENCES Post CONSTRAINT questionpk PRIMARY KEY,
       isClosed    boolean NOT NULL,
       nViews      BIGINT NOT NULL DEFAULT 0,
       CHECK (nViews > 0),
       tittle text NOT NULL
);

CREATE TABLE TagQuestion(
    question_id  SERIAL NOT NULL REFERENCES Question,
    tag_id       SERIAL NOT NULL REFERENCES Tag,
    PRIMARY KEY(question_id, tag_id)
);

CREATE TABLE PostVote (
       postID      SERIAL REFERENCES Post NOT NULL,
       posterID    BIGINT REFERENCES "User" NOT NULL,
       value       INTEGER NOT NULL,
       CHECK (value = 1 OR value = -1),
       PRIMARY KEY(postID, posterID)
);

CREATE TABLE PostReport (
       postID      SERIAL NOT NULL,
       reporterID  BIGINT NOT NULL,
       date        TIMESTAMP WITH TIME zone DEFAULT now() NOT NULL,
       PRIMARY KEY(postID, reporterID)
);

CREATE TABLE FaqEntry (
       id          SERIAL CONSTRAINT postreportpk PRIMARY KEY,
       question    text NOT NULL,
       answer      text NOT NULL
);

CREATE TABLE Team (
  id          SERIAL CONSTRAINT teamPk PRIMARY KEY,
  name        TEXT NOT NULL
);

CREATE TABLE TeamMember (
  id          SERIAL CONSTRAINT teamMemberPK PRIMARY KEY,
  name        TEXT NOT NULL,
  email        TEXT NOT NULL,
  title       TEXT NOT NULL,
  joinDate    TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
  img_path    TEXT NOT NULL DEFAULT '0.png'
);

CREATE TABLE TeamToTeamMember (
  teamId        SERIAL,
  teamMemberID  SERIAL,
  PRIMARY KEY (teamId,teamMemberID)
);


ALTER TABLE Contact
    ADD CONSTRAINT userID_fk FOREIGN KEY (userID) REFERENCES "User"(id) ON
UPDATE CASCADE;

ALTER TABLE Contact
    ADD CONSTRAINT subjectIDfk FOREIGN KEY (subjectID) REFERENCES
Subject(subjectID) ON UPDATE CASCADE;

ALTER TABLE BanInfo
    ADD CONSTRAINT userIDfk FOREIGN KEY (userID) REFERENCES "User"(id) ON
UPDATE CASCADE;

ALTER TABLE BanInfo
    ADD CONSTRAINT adminIDfk FOREIGN KEY (adminID) REFERENCES "User"(id) ON
UPDATE CASCADE;

ALTER TABLE Answer
    ADD CONSTRAINT postIDfk FOREIGN KEY (postID) REFERENCES Post(id) ON
UPDATE CASCADE;

ALTER TABLE Question
    ADD CONSTRAINT postIDfk FOREIGN KEY (postID) REFERENCES Post(id) ON
UPDATE CASCADE;

ALTER TABLE TagQuestion
    ADD CONSTRAINT question_idFK FOREIGN KEY (question_id) REFERENCES
Question(postID) ON UPDATE CASCADE;

ALTER TABLE TagQuestion
    ADD CONSTRAINT tag_idFK FOREIGN KEY (tag_id) REFERENCES Tag(id) ON UPDATE
CASCADE;

ALTER TABLE PostVote
    ADD CONSTRAINT postIdFk FOREIGN KEY (postID) REFERENCES Post(id) ON
UPDATE CASCADE;

ALTER TABLE ONLY PostVote
      ADD CONSTRAINT postreport_user_fk FOREIGN KEY (posterID) REFERENCES
"User"(id) ON UPDATE CASCADE;

ALTER TABLE ONLY PostReport
      ADD CONSTRAINT postreport_post_fk FOREIGN KEY (postID) REFERENCES
Post(id) ON UPDATE CASCADE;

ALTER TABLE ONLY PostReport
      ADD CONSTRAINT postreport_user_fk FOREIGN KEY (reporterId) REFERENCES
"User"(id) ON UPDATE CASCADE;
```

```
ALTER TABLE ONLY TeamToTeamMember
      ADD CONSTRAINT teamtoteamember_teamid_fk FOREIGN KEY (teamID)
REFERENCES Team(id) ON UPDATE CASCADE;

ALTER TABLE ONLY TeamToTeamMember
      ADD    CONSTRAINT    teamtoteamember_teammemberid_fk    FOREIGN    KEY
(teamMemberID) REFERENCES TeamMember(id) ON UPDATE CASCADE;
```

Group

- Davide Henrique Fernandes da Costa, up201503995@fe.up.pt
- Dinis Filipe da Silva Trigo, up201504196@fe.up.pt
- Diogo Afonso Duarte Reis, up201505472@fe.up.pt
- Tiago José Sousa Magalhães, up201607931@fe.up.pt