A5: Relational Schema, validation and schema refinement

SegFault is a collaborative platform for programmers to learn, discuss different approaches, present ideas and share knowledge in a Q&A style. To this end, the following sections provide detailed insight into the project's relational schema, domains, functional dependencies and schema validation.

1. Relational Schema

> The Relational Schema includes the relation schemas, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK. Relation schemas are specified in the compact notation:

Relation reference	Relation Compact Notation
R01	Category(**ID**, name NN, description, numPosts NN CK numPosts >= 0)
R02	QuestionCategory(**questionID** → Question, **categoryID** → Category)
R03	Question(**ID** → Commentable, title NN, correctAnswer → Answer UK)
R04	Answer(**ID** → Commentable, questionID → Question NN)
R05	Commentable(**ID** → Message)
R06	Comment(**ID** → Message, commentableID → Commentable NN)
R07	Message(**ID**, creationDate NN DF Today, score NN DF 0, author -> User NN, numReports NN DF 0, isBanned NN DF False)
R08	MessageVersion(**ID**, content NN, timeStamp NN, messageID → Message NN, moderatorID -> Moderator)
R09	Vote(**messageID** → Message, **userID** → User, positive NN)
R10	User(**ID**, userName UK NN, email UK NN, passwordHash NN, bio, reputation NN)
R11	Moderator(**ID** → User)
R12	Notification(**ID**, description NN, date NN, read NN, userID → User)
R13	CommentableNotification(**ID** → Notification, commentableID → Commentable NN)
R14	BadgeNotification(**ID** → Notification, badgeID → Badge NN)
R15	BadgeAttainment(**userID** →User, **badgeID** → Badge, attainmentDate NN)
R16	Badge(**ID**, description NN)
R17	ModeratorBadge(**ID** → Badge)
R18	TrustedBadge(**ID** → Badge)

2. Domains

The specification of additional domains can also be made in a compact form, using the notation:

Domain Name	Domain Specification
Today	DATE DEFAULT CURRENT_DATE

3. 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. Should it be necessary, in case the scheme is not in the Boyce–Codd Normal Form (BCNF), the relational schema is refined using normalization.

Table R01	(Category)
Keys: {ID}	
Functional Dependencies	
FD0101	{ID} → {attribute, name, description, numPosts}
Normal Form	BCNF

Table R02	(QuestionCategory)
Keys: {(questionID, categoryID)}	
Functional Dependencies	
(none)	
Normal Form	BCNF

Table R03	(Question)
Keys: {ID, correctAnswer}	
Functional Dependencies	
FD0301	{ID} → {title, correctAnswer, numPosts}
FD0302	{correctAnswer} → {ID, title, numPosts}
Normal Form	BCNF

Table R04	(Answer)
Keys: {ID}	
Functional Dependencies	
FD0401	{ID} → {questionID}
Normal Form	BCNF

Table R05	(Commentable)
Keys: {ID}	
Functional Dependencies	
(none)	
Normal Form	BCNF

Table R06	(Comment)
Keys: {ID}	
Functional Dependencies	
FD0601	{ID} → {commentableID}
Normal Form	BCNF

Table R07	(Message)
Keys: {ID}	
Functional Dependencies	
FD0701	{ID} → {creationDate, score, author, numReports, isBanned}
Normal Form	BCNF

Table R08	(MessageVersion)
Keys: {ID, messageID}	
Functional Dependencies	
FD0801	{ID} → {content, timeStamp, messageID, moderatorID}
FD0802	{messageID} → {content, timeStamp, ID, moderatorID}
Normal Form	BCNF

Table R09	(Vote)
Keys: {(messageID, userID)}	
Functional Dependencies	
FD0901	{messageID, userID} → {positive}
Normal Form	BCNF

Table R10	(User)
Keys: {ID, username, email}	
Functional Dependencies	
FD1001	{ID} → {username, email, passwordHash, bio, reputation}
FD1002	{username} → {ID, email, passwordHash, bio, reputation}
FD1003	{email} → {username, ID, passwordHash, bio, reputation}
Normal Form	BCNF

Table R11	(Moderator)
Keys: {ID}	
Functional Dependencies	
(none)	
Normal Form	BCNF

Table R12	(Notification)
Keys: {ID}	
Functional Dependencies	

FD1201	{ID} → {description, date, read, userID}
Normal Form	BCNF

Table R13	(CommentableNotification)
Keys: {ID}	
Functional Dependencies	
FD1301	{ID} → {commentableID}
Normal Form	BCNF

Table R14	(BadgeNotification)
Keys: {ID}	
Functional Dependencies	
FD1401	{ID} → {badgeID}
Normal Form	BCNF

Table R15	(BadgeAttainment)
Keys: {(userID, badgeID)}	
Functional Dependencies	
FD1501	{userID, badgeID} → {attainmentDate}
Normal Form	BCNF

Table R16	(Badge)
Keys: {ID}	
Functional Dependencies	
FD1601	{ID} → {description}
Normal Form	BCNF

Table R17	(ModeratorBadge)
Keys: {ID}	
Functional Dependencies	
(none)	
Normal Form	BCNF

Table R18	(TrustedBadge)
Keys: {ID}	
Functional Dependencies	
(none)	

Normal Form	BCNF

> If necessary, description of the changes necessary to convert the schema to BCNF. Justification of the BCNF.

4. SQL Code

```
DROP TABLE IF EXISTS "user" CASCADE;
DROP TABLE IF EXISTS moderator CASCADE;
DROP TABLE IF EXISTS message CASCADE;
DROP TABLE IF EXISTS commentable CASCADE;
DROP TABLE IF EXISTS question CASCADE;
DROP TABLE IF EXISTS answer CASCADE;
DROP TABLE IF EXISTS category CASCADE;
DROP TABLE IF EXISTS question_category CASCADE;
DROP TABLE IF EXISTS comment CASCADE;
DROP TABLE IF EXISTS message_version CASCADE;
DROP TABLE IF EXISTS vote CASCADE;
DROP TABLE IF EXISTS badge CASCADE;
DROP TABLE IF EXISTS moderator_badge CASCADE;
DROP TABLE IF EXISTS trusted_badge CASCADE;
DROP TABLE IF EXISTS notification CASCADE;
DROP TABLE IF EXISTS commentable_notification CASCADE;
DROP TABLE IF EXISTS badge_notification CASCADE;
DROP TABLE IF EXISTS badge_attainment CASCADE;

CREATE TABLE "user" (
    id BIGSERIAL PRIMARY KEY,
    username TEXT NOT NULL UNIQUE,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    biography TEXT,
    reputation SMALLINT NOT NULL
);

CREATE TABLE moderator (
    id BIGINT PRIMARY KEY REFERENCES "user"(id)
);

CREATE TABLE message (
    id BIGSERIAL PRIMARY KEY,
    creation_date TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    score INTEGER DEFAULT 0 NOT NULL,
    num_reports SMALLINT DEFAULT 0 NOT NULL,
    is_banned BOOLEAN DEFAULT FALSE,
    author BIGINT NOT NULL REFERENCES "user"(id)
);

CREATE TABLE commentable (
    id BIGINT PRIMARY KEY REFERENCES message(id)
);

CREATE TABLE question (
    id BIGINT PRIMARY KEY REFERENCES commentable(id),
    title TEXT NOT NULL,
    correct_answer BIGINT UNIQUE
);

CREATE TABLE answer (
    id BIGINT PRIMARY KEY REFERENCES commentable(id),
    question_id BIGINT NOT NULL REFERENCES question(id)
);

CREATE TABLE category (
    id SERIAL PRIMARY KEY,
    name TEXT NOT NULL,
    description TEXT,
    num_posts INTEGER DEFAULT 0 NOT NULL
);

CREATE TABLE question_category (
    question_id BIGINT REFERENCES question(id),
    category_id INTEGER REFERENCES category(id),
    PRIMARY KEY (question_id, category_id)
);

CREATE TABLE comment (
    id BIGINT PRIMARY KEY REFERENCES message(id),
    commentable_id BIGINT NOT NULL REFERENCES commentable(id)
);

CREATE TABLE message_version (
```

```
    id BIGSERIAL PRIMARY KEY,
    content TEXT NOT NULL,
    message_id BIGINT REFERENCES message(id),
    creation_time TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    moderator_id BIGINT NOT NULL REFERENCES moderator(id)
);

CREATE TABLE vote (
    message_id BIGINT NOT NULL REFERENCES message(id),
    user_id BIGINT NOT NULL REFERENCES "user"(id),
    positive BOOLEAN NOT NULL,
    PRIMARY KEY (message_id, user_id)
);

CREATE TABLE badge (
    id SERIAL PRIMARY KEY,
    description TEXT NOT NULL
);

CREATE TABLE moderator_badge (
    id INTEGER PRIMARY KEY REFERENCES badge(id)
);

CREATE TABLE trusted_badge (
    id INTEGER PRIMARY KEY REFERENCES badge(id)
);

CREATE TABLE notification (
    id BIGSERIAL PRIMARY KEY,
    description TEXT NOT NULL,
    "date" TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    read BOOLEAN NOT NULL,
    user_id BIGINT NOT NULL REFERENCES "user"(id)
);

CREATE TABLE commentable_notification (
    id BIGINT PRIMARY KEY REFERENCES notification(id),
    commentable_id BIGINT NOT NULL REFERENCES commentable(id)
);

CREATE TABLE badge_notification (
    id BIGINT PRIMARY KEY REFERENCES notification(id),
    badge_id BIGINT NOT NULL REFERENCES badge(id)
);

CREATE TABLE badge_attainment (
    user_id BIGINT NOT NULL REFERENCES "user"(id),
    badge_id SMALLINT NOT NULL REFERENCES badge(id),
    attainment_date TIMESTAMP WITH TIME ZONE DEFAULT now() NOT NULL,
    PRIMARY KEY (user_id, badge_id)
);

ALTER TABLE question
  ADD FOREIGN KEY (correct_answer) REFERENCES answer(id) ON UPDATE CASCADE;
```

Revision history

GROUP1763, 15/03/2018

André Cruz, up201503776@fe.up.pt
Daniel Marques, up201503822@fe.up.pt
Edgar Carneiro, up201503784@fe.up.pt
João Carvalho, up201504875@fe.up.pt