A5: Relational schema, validation and schema refinement

Our project, Answerly, is a web application for collaborative Questions and Answers.

This artifact contains the Relational Schema obtained by mapping from the Conceptual Data Model. The Relational Schema includes the relation schema, attributes, domains, primary keys, foreign keys and other integrity rules: UNIQUE, DEFAULT, NOT NULL, CHECK...

1. Relational Schema

In the generalizations that include administrator as a subclass of moderator, and this last one as a subclass of user, we chose the **Use Nulls** technique for representing the relations, because they are heavily overlapping and with only one subclass per generalization.

Relation Reference	Relation Compact Notation
R01	user(**userID**, first_name *NN*, last_name *NN*, email *UK NN*, description, username *UK NN*, password *NN*, score *DF 0*)
R02	label(**labelID**, name *NN*)
R03	notification(**notificationID**, content *NN*, date *DF Today*, viewed *DF False*, user_id → user *NN*)
R04	user_management(**managementID**, state *NN*, status *NN*, user_id → user *NN*)
R05	vote(**voteID**, like, dislike, user_id → user *NN*, question_id → question, answer_id → answer *CK question_id = NN XOR answer_id = NN*)
R06	question(**questionID**, user_id → user *NN*, title *NN*, description *NN*, nr_likes *NN DF 0*, nr_dislikes *NN DF 0*, question_date *NN DF Today*)
R07	answer(**answerID**, user_id → user *NN*, question_id → question *NN*, answer_date *NN DF Today*, content *NN*, nr_likes *NN DF 0*, nr_dislikes *NN DF 0*)
R08	comment(**commentID**, user_id → user *NN*, questionID→Question, answerID→Answer *CK question_id = NN XOR answer_id = NN*, content *NN*, comment_date *NN DF Today*)
R09	report(**reportID**, userID→User, questionID→Question, answerID→Answer, commentID→Comment *CK user_id = NN XOR question_id = NN XOR answer_id = NN XOR comment_id = NN*)
R10	report_status(**statusID**, report_id → report, state *NN DF unresolved CK state IN States*, comment, responsibleUser→Moderator *NN*)
R11	marked_answer(**questionID** → question, **answerID** → answer)
R12	following(**userID** → user, **labelID** → label))
R13	about(**questionID** → question, **labelID** → label)

- UK means UNIQUE KEY
- NN means NOT NULL
- DF means DEFAULT
- CK means CHECK

2. Domains

Specification of additional domains:

Domain Name	Domain Specification
Today	DATE DEFAULT CURRENT DATE
States	ENUM ('unresolved', 'reviewing', 'resolved')

3. Functional Dependencies and schema validation

In the following tables, all relations are in the Boyce-Codd Normal Form, since for each non trivial functional dependency A → B, A is a (super)key of the relation.

Table R01 (user)	
Keys: {user_id}, {username}, {email}	
Functional Dependencies	
FD0101	{user_id} → {first_name, last_name, email, description, username, password, score}
FD0102	{username} → {user_id, first_name, last_name, email, description, password, score}
FD0103	{email} → {user_id, first_name, last_name, description, username, password, score}
NORMAL FORM	BCNF

Table R02 (label)	
Keys: {label_id}	
Functional Dependencies	
FD0201	{label_id} → {name}
NORMAL FORM	BCNF

Table R03 (notification)	
Keys: {notification_id}	
Functional Dependencies	

Table R03 (notification)	
FD0301	{notification_id} → {content, date, viewed, user_id}
NORMAL FORM	BCNF

Table R04 (user_management)	
Keys: {management_id}	
Functional Dependencies	
FD0401	{management_id} → {state, status, user_id}
NORMAL FORM	BCNF

Table R05 (vote)	
Keys: {vote_id}	
Functional Dependencies	
FD0501	{vote_id} → {like, dislike, user_id, question_id, answer_id}
NORMAL FORM	BCNF

Table R06 (question)	
Keys: {question_id}	
Functional Dependencies	
FD0601	{question_id} → {user_id, title, description, nr_likes, nr_dislikes, question_date}
NORMAL FORM	BCNF

Table R07 (answer)	
Keys: {answer_id}	
Functional Dependencies	
FD0701	{answer_id} → {user_id, question_id, answer_date, content, nr_likes, nr_dislikes}
NORMAL FORM	BCNF

Table R08 (comment)	
Keys: {comment_id}	
Functional Dependencies	
FD0801	{comment_id} → {user_id, question_id, answer_id, content, comment_date}

Table R08 (comment)	
NORMAL FORM	BCNF

Table R09 (report)	
Keys: {report_id}	
Functional Dependencies	
FD0901	{report_id} → {user_id, question_id, answer_id, comment_id}
NORMAL FORM	BCNF

Table R10 (report_status)	
Keys: {status_id}	
Functional Dependencies	
FD1001	{status_id} → {report_id, state, comment, responsible_user}
NORMAL FORM	BCNF

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

Table R12 (following)	
Keys: {user_id, label_id}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R13 (about)	
Keys: {question_id, label_id}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

4. SQL Code

```
-- Table: user
DROP TABLE IF EXISTS "user" CASCADE;
CREATE TABLE "user" (
    user_id         SERIAL          PRIMARY KEY,
    first_name      TEXT            NOT NULL,
    last_name       TEXT            NOT NULL,
    email           TEXT            NOT NULL UNIQUE,
    description     TEXT,
    username        TEXT            NOT NULL UNIQUE,
    password        TEXT            NOT NULL,
    score           INTEGER         NOT NULL DEFAULT 0
);

-- Table: label
DROP TABLE IF EXISTS label CASCADE;
CREATE TABLE label (
    label_id        SERIAL          PRIMARY KEY,
    name            TEXT            NOT NULL
);

-- Table: notification
DROP TABLE IF EXISTS notification CASCADE;
CREATE TABLE notification (
    notification_id SERIAL          PRIMARY KEY,
    content         TEXT            NOT NULL,
    date            DATE            DEFAULT 'today' NOT NULL,
    viewed          BOOLEAN         DEFAULT FALSE NOT NULL,
    user_id         INTEGER         REFERENCES "user" (user_id) NOT NULL
);

-- Table: user_management
DROP TABLE IF EXISTS user_management CASCADE;
CREATE TABLE user_management (
    management_id   SERIAL          PRIMARY KEY,
    state           TEXT            DEFAULT 'active' NOT NULL,
    status          TEXT            DEFAULT 'user' NOT NULL,
    user_id         INTEGER         REFERENCES "user" (user_id) NOT NULL
);

-- Table: moderator
DROP TABLE IF EXISTS moderator CASCADE;
CREATE TABLE moderator (
    moderator_id     INTEGER         REFERENCES "user" (user_id) NOT NULL UNIQUE
);

-- Table: administrator
DROP TABLE IF EXISTS administrator CASCADE;
CREATE TABLE administrator (
    administrator_id INTEGER         REFERENCES "moderator" (moderator_id) NOT
NULL
```

```sql
);

-- Table: question
DROP TABLE IF EXISTS question CASCADE;
CREATE TABLE question (
    question_id     SERIAL          PRIMARY KEY,
    user_id         INTEGER         REFERENCES "user" (user_id) NOT NULL,
    title           TEXT            NOT NULL,
    description     TEXT            NOT NULL,
    nr_likes        INTEGER         DEFAULT 0 NOT NULL,
    nr_dislikes     INTEGER         DEFAULT 0 NOT NULL,
    question_date   DATE            DEFAULT 'today' NOT NULL
);

-- Table: answer
DROP TABLE IF EXISTS answer CASCADE;
CREATE TABLE answer (
    answer_id        SERIAL          PRIMARY KEY,
    user_id          INTEGER         REFERENCES "user" (user_id) NOT NULL,
    question_id      INTEGER         REFERENCES "question" (question_id) NOT NULL,
    answer_date      DATE            DEFAULT 'today' NOT NULL,
    content          TEXT            NOT NULL,
    nr_likes         INTEGER         DEFAULT 0 NOT NULL,
    nr_dislikes      INTEGER         DEFAULT 0 NOT NULL
);

-- Table: comment
DROP TABLE IF EXISTS comment CASCADE;
CREATE TABLE comment (
    comment_id       SERIAL          PRIMARY KEY,
    user_id          INTEGER         REFERENCES "user" (user_id) NOT NULL,
    question_id      INTEGER         REFERENCES "question" (question_id),
    answer_id        INTEGER         REFERENCES "answer" (answer_id),
    comment_date     DATE            DEFAULT 'today' NOT NULL,
    content          TEXT            NOT NULL,
    CHECK (
        (question_id IS NOT NULL AND answer_id IS NULL) OR
        (question_id IS NULL AND answer_id IS NOT NULL)
    )
);

-- Table: vote
DROP TABLE IF EXISTS vote CASCADE;
CREATE TABLE vote (
    vote_id          SERIAL          PRIMARY KEY,
    "like"           BOOLEAN         NOT NULL,
    dislike          BOOLEAN         NOT NULL,
    user_id          INTEGER         REFERENCES "user" (user_id) NOT NULL,
    question_id      INTEGER         REFERENCES "question" (question_id),
    answer_id        INTEGER         REFERENCES "answer" (answer_id),
    CHECK (
        (question_id IS NOT NULL AND answer_id IS NULL) OR
        (question_id IS NULL AND answer_id IS NOT NULL)
    )
```

```
);

-- Table: report
DROP TABLE IF EXISTS report CASCADE;
CREATE TABLE report (
    report_id       SERIAL          PRIMARY KEY,
    user_id         INTEGER         REFERENCES "user" (user_id),
    question_id     INTEGER         REFERENCES "question" (question_id),
    answer_id       INTEGER         REFERENCES "answer" (answer_id),
    comment_id      INTEGER         REFERENCES "comment" (comment_id),
    CHECK(
        (user_id IS NOT NULL AND question_id IS NULL AND answer_id IS NULL AND
comment_id IS NULL) OR
        (user_id IS NULL AND question_id IS NOT NULL AND answer_id IS NULL AND
comment_id IS NULL) OR
        (user_id IS NULL AND question_id IS NULL AND answer_id IS NOT NULL AND
comment_id IS NULL) OR
        (user_id IS NULL AND question_id IS NULL AND answer_id IS NULL AND
comment_id IS NOT NULL)
    )
);

-- Table: report_status
DROP TABLE IF EXISTS report_status CASCADE;
CREATE TABLE report_status (
    status_id       SERIAL          PRIMARY KEY,
    report_id       INTEGER         REFERENCES "report" (report_id) NOT NULL,
    state           TEXT            DEFAULT 'unresolved' NOT NULL,
    comment         TEXT,
    responsible_user INTEGER        REFERENCES "moderator" (moderator_id) NOT
NULL
);

-- Table: marked_answer
DROP TABLE IF EXISTS marked_answer CASCADE;
CREATE TABLE marked_answer (
    question_id     INTEGER         REFERENCES "question" (question_id) NOT NULL,
    answer_id       INTEGER         REFERENCES "answer" (answer_id) NOT NULL
);

-- Table: following
DROP TABLE IF EXISTS following CASCADE;
CREATE TABLE following (
    user_id         INTEGER         REFERENCES "user" (user_id) NOT NULL,
    label_id        INTEGER         REFERENCES "label" (label_id) NOT NULL
);

-- Table: about
DROP TABLE IF EXISTS about CASCADE;
CREATE TABLE about (
    question_id     INTEGER         REFERENCES "question" (question_id) NOT NULL,
    label_id        INTEGER         REFERENCES "label" (label_id) NOT NULL
);
```

Revision history

1. First submission (23/03/2020).

GROUP2064, 23/03/2020

- [Editor] Antonio Pedro Reis Ribeiro Sousa Dantas, up201703878@fe.up.pt
- Eduardo João Santana Macedo, up201703658@fe.up.pt
- Nuno Miguel Teixeira Cardoso, up201706162@fe.up.pt
- Paulo Roberto Dias Mourato, up201705616@fe.up.pt