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

Relation Reference	Relation Compact Notation
R01	user(**ID**, first_name *NN*, last_name *NN*, email *UK NN*, bio, username *UK NN*, password *NN*, *DF 0*)
R02	label(**ID**, name *NN*)
R03	notification(**ID**, content *NN*, date *DF Now*, viewed *DF False*, user_id → user *NN*)
R04	user_management(**ID**, status *NN*, user_id → user *UK NN*)
R05	vote(**ID**, vote, user_id → user *NN*, question_id → question, answer_id → answer *CK question_id = NN XOR answer_id = NN*)
R06	question(**ID**, user_id → user *NN*, title *NN*, description *NN*, nr_likes *NN DF 0*, nr_dislikes *NN DF 0*, question_date *NN DF Now*)
R07	answer(**ID**, user_id → user *NN*, question_id → question *NN*, answer_date *NN DF Now*, content *NN*, nr_likes *NN DF 0*, nr_dislikes *NN DF 0*, marked_answer *NN DF FALSE*)
R08	comment(**ID**, user_id → user *NN*, questionID→Question, answerID→Answer *CK question_id = NN XOR answer_id = NN*, content *NN*, comment_date *NN DF Now*)
R09	report(**ID**, 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(**ID**, report_id → report, state *NN DF unresolved CK state IN States*, comment, responsible_user → user_management *NN*)
R11	following(**userID** → user, **labelID** → label))
R12	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
Now	DATE DEFAULT CURRENT_TIMESTAMP_
Report States	ENUM ('unresolved', 'reviewing', 'resolved')
User Status	ENUM ('normal', 'moderator', 'administrator', 'banned')

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: {id}, {username}, {email}	
Functional Dependencies	
FD0101	{id} → {first_name, last_name, email, bio, username, password, score}
FD0102	{username} → {user_id, first_name, last_name, email, bio, password, score, markedAnswer}
FD0103	{email} → {user_id, first_name, last_name, bio, username, password, score}
NORMAL FORM	BCNF

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

Table R03 (notification)	
Keys: {id}	
Functional Dependencies	
FD0301	{id} → {content, date, viewed, user_id}
NORMAL FORM	BCNF

Table R04 (user_management)
Keys: {id}

Table R04 (user_management)	
Functional Dependencies	
FD0401	{id} → {status, user_id}
NORMAL FORM	BCNF

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

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

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

Table R08 (comment)	
Keys: {id}	
Functional Dependencies	
FD0801	{id} → {user_id, question_id, answer_id, content, comment_date}
NORMAL FORM	BCNF

Table R09 (report)
Keys: {id}
Functional Dependencies

Table R09 (report)	
FD0901	{id} → {user_id, question_id, answer_id, comment_id}
NORMAL FORM	BCNF

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

Table R11 (following)	
Keys: {user_id, label_id}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

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

4. SQL Code

```sql
-- Table: user
DROP TABLE IF EXISTS "user" CASCADE;
CREATE TABLE "user" (
    id              SERIAL          PRIMARY KEY,
    first_name      TEXT            NOT NULL,
    last_name       TEXT            NOT NULL,
    email           TEXT            NOT NULL UNIQUE,
    bio             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 (
    id                SERIAL            PRIMARY KEY,
    name              TEXT              NOT NULL
);

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

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

-- Table: question
DROP TABLE IF EXISTS question CASCADE;
CREATE TABLE question (
    id                SERIAL            PRIMARY KEY,
    user_id           INTEGER           REFERENCES "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 'Now' NOT NULL
);

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

-- Table: comment
DROP TABLE IF EXISTS comment CASCADE;
CREATE TABLE comment (
    id                 SERIAL            PRIMARY KEY,
    user_id            INTEGER           REFERENCES "user" (id) NOT NULL,
    question_id        INTEGER           REFERENCES "question" (id),
    answer_id        INTEGER          REFERENCES "answer" (id),
    comment_date     DATE             DEFAULT 'Now' 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 (
    id               SERIAL           PRIMARY KEY,
    "vote"           BOOLEAN          NOT NULL,
    user_id          INTEGER          REFERENCES "user" (id) NOT NULL,
    question_id      INTEGER          REFERENCES "question" (id),
    answer_id        INTEGER          REFERENCES "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 (
    id               SERIAL           PRIMARY KEY,
    user_id          INTEGER          REFERENCES "user" (id),
    question_id      INTEGER          REFERENCES "question" (id),
    answer_id        INTEGER          REFERENCES "answer" (id),
    comment_id       INTEGER          REFERENCES "comment" (id),
    CHECK(
        (question_id IS NOT NULL AND answer_id IS NULL AND comment_id IS NULL) OR
        (question_id IS NULL AND answer_id IS NOT NULL AND comment_id IS NULL) OR
        (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 (
    id               SERIAL           PRIMARY KEY,
    report_id        INTEGER          REFERENCES "report" (id) NOT NULL,
    state            TEXT             DEFAULT 'unresolved' NOT NULL,
    comment          TEXT,
    responsible_user INTEGER          REFERENCES "user_management" (user_id) NOT
NULL
);

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

```
);

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

Revision history

1. First submission (23/03/2020).
2. Deleted Administrator and Moderator tables. Changed all id's to "id" and other minor changes (28/03/2020).
3. Tested sql code, fixed some errors (29/03/2020).

GROUP2064, 29/03/2020

- [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