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 (**userID**, firstName *NN*, lastName *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*, userID → user *NN*)
R04	UserManagement (**managementID**, state *NN*, status *NN*, userID → User *NN*)
R05	Moderator (**userID** → User)
R06	Administrator (**userID** → Moderator)
R07	Vote (**voteID**, like, dislike, userID → User *NN*, questionID → Question, answerID → Answer *CK questionID = NN XOR answerID = NN*)
R08	Question (**questionID**, userID→User *NN*, title *NN*, description *NN*, nrLikes *NN DF 0*, nrDislikes *NN DF 0*, questionDate *NN DF Today*)
R09	Answer (**answerID**, userID→User *NN*, questionID→Question *NN*, answerDate *NN DF Today*, content *NN*, nrLikes *NN DF 0*, nrDislikes *NN DF 0*)
R10	Comment (**commentID**, userID→User *NN*, questionID→Question, answerID→Answer *CK questionID = NN XOR answerID = NN*, content *NN*, commentDate *NN DF Today*)
R11	Report (**reportID**, userID→User, questionID→Question, answerID→Answer, commentID→Comment *CK userID = NN XOR questionID = NN XOR answerID = NN XOR commentID = NN*)
R12	ReportStatus (**statusID**, reportID→Report, state *NN DF unresolved CK state IN States*, comment, responsibleUser→Moderator *NN*)
R13	MarkedAnswer (__questionID__→Question, __answerID__→Answer)
R14	Following (**userID** → User, **labelID** → Label))
R15	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

Table R01 (User)	
Keys: {userID}, {username}, {email}	
Functional Dependencies	
FD0101	{userID} → {firstName, lastName, email, description, username, password, score}
FD0102	{username} → {userID, firstName, lastName, email, description, password, score}
FD0103	{email} → {userID, firstName, lastName, description, username, password, score}
NORMAL FORM	BCNF

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

Table R03 (Notification)	
Keys: {notificationID}	
Functional Dependencies	
FD0301	{notificationID} → {content, date, viewed, userID}
NORMAL FORM	BCNF

Table R04 (UserManagement)	
Keys: {managementID}	
Functional Dependencies	
FD0401	{managementID} → {state, status, userID}
NORMAL FORM	BCNF

Table R05 (Moderator)	
Keys: {userID}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R06 (Administrator)	
Keys: {userID}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R07 (Vote)	
Keys: {voteID}	
Functional Dependencies	
FD0701	{voteID} → {like, dislike, userID, questionID, answerID}
NORMAL FORM	BCNF

Table R08 (Question)	
Keys: {questionID}	
Functional Dependencies	
FD0801	{questionID} → {userID, title, description, nrLikes, nrDislikes, questionDate}
NORMAL FORM	BCNF

Table R09 (Answer)
Keys: {answerID}
Functional Dependencies

Table R09 (Answer)	
FD0901	{answerID} → {userID, questionID, answerDate, content, nrLikes, nrDislikes}
NORMAL FORM	BCNF

Table R10 (Comment)	
Keys: {commentID}	
Functional Dependencies	
FD1001	{commentID} → {userID, questionID, answerID, content, commentDate}
NORMAL FORM	BCNF

Table R11 (Report)	
Keys: {reportID}	
Functional Dependencies	
FD1101	{reportID} → {userID, questionID, answerID, commentID}
NORMAL FORM	BCNF

Table R12 (ReportStatus)	
Keys: {statusID}	
Functional Dependencies	
FD1201	{statusID} → {reportID, state, comment, responsibleUser}
NORMAL FORM	BCNF

Table R13 (MarkedAnswer)	
Keys: {questionID, answerID}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R14 (Following)	
Keys: {userID, labelID}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R15 (About)
Keys: {questionID, labelID}
Functional Dependencies
(none)
NORMAL FORM BCNF

4. SQL Code

```
PRAGMA foreign_keys = off;

-- Table: User
DROP TABLE IF EXISTS User;
CREATE TABLE User (
    userID          INTEGER         PRIMARY KEY,
    firstName       TEXT            NOT NULL,
    lastName        TEXT            NOT NULL,
    email           TEXT            NOT NULL,
    description     TEXT,
    username        TEXT            NOT NULL UNIQUE,
    password        TEXT            NOT NULL,
    score           INTEGER         DEFAULT 0
);

-- Table: Label
DROP TABLE IF EXISTS Label;
CREATE TABLE Label (
    labelID         INTEGER         PRIMARY KEY,
    name            TEXT            NOT NULL
);

-- Table: Notification
DROP TABLE IF EXISTS Notification;
CREATE TABLE Notification (
    notificationID  INTEGER         PRIMARY KEY,
    content         TEXT            NOT NULL,
    date            DATE            DEFAULT 'today' NOT NULL,
    viewed          BOOLEAN         DEFAULT FALSE,
    userID          INTEGER         REFERENCES "User" (userID)
);

-- Table: UserManagement
DROP TABLE IF EXISTS UserManagement;
CREATE TABLE UserManagement (
    managementID    INTEGER         PRIMARY KEY,
    state           TEXT            DEFAULT 'active' NOT NULL,
    status          TEXT            DEFAULT 'user' NOT NULL,
    userID          INTEGER         REFERENCES "User" (userID)
);
```

```sql
-- Table: Moderator
DROP TABLE IF EXISTS Moderator;
CREATE TABLE Moderator (
    moderatorID     INTEGER         REFERENCES "User" (userID)
);

-- Table: Administrator
DROP TABLE IF EXISTS Administrator;
CREATE TABLE Administrator (
    administratirID INTEGER         REFERENCES "Moderator" (moderatorID)
);

-- Table: Question
DROP TABLE IF EXISTS Question;
CREATE TABLE Question (
    questionID      INTEGER         PRIMARY KEY,
    userID          INTEGER         REFERENCES "User" (userID),
    title           TEXT            NOT NULL,
    description     TEXT            NOT NULL,
    nrLikes         INTEGER         DEFAULT 0 NOT NULL,
    nrDislikes      INTEGER         DEFAULT 0 NOT NULL,
    questionDate    DATE            DEFAULT 'today' NOT NULL
);

-- Table: Answer
DROP TABLE IF EXISTS Answer;
CREATE TABLE Answer (
    answerID        INTEGER         PRIMARY KEY,
    userID          INTEGER         REFERENCES "User" (userID),
    questionID      INTEGER         REFERENCES "Question" (questionID),
    answerDate      DATE            DEFAULT 'today' NOT NULL,
    content         TEXT            NOT NULL,
    nrLikes         INTEGER         DEFAULT 0 NOT NULL,
    nrDislikes      INTEGER         DEFAULT 0 NOT NULL
);

-- Table: Comment
DROP TABLE IF EXISTS Comment;
CREATE TABLE Comment (
    commentID       INTEGER         PRIMARY KEY,
    userID          INTEGER         REFERENCES "User" (userID),
    questionID      INTEGER         REFERENCES "Question" (questionID),
    answerID        INTEGER         REFERENCES "Answer" (answerID),
    commentDate     DATE            DEFAULT 'today' NOT NULL,
    content         TEXT            NOT NULL,
    CHECK (
        (questionID IS NOT NULL AND answerID IS NULL) OR
        (questionID IS NULL AND answerID IS NOT NULL)
    )
);

-- Table: Vote
DROP TABLE IF EXISTS Vote;
CREATE TABLE Vote (
```

```
    voteID          INTEGER         PRIMARY KEY,
    like            BOOLEAN,
    dislike         BOOLEAN,
    userID          INTEGER         REFERENCES "User" (userID),
    questionID      INTEGER         REFERENCES "Question" (questionID),
    answerID        INTEGER         REFERENCES "Answer" (answerID),
    CHECK (
        (questionID IS NOT NULL AND answerID IS NULL) OR
        (questionID IS NULL AND answerID IS NOT NULL)
    )
);

-- Table: Report
DROP TABLE IF EXISTS Report;
CREATE TABLE Report (
    reportID        INTEGER         PRIMARY KEY,
    userID          INTEGER         REFERENCES "User" (userID),
    questionID      INTEGER         REFERENCES "Question" (questionID),
    answerID        INTEGER         REFERENCES "Answer" (answerID),
    commentID       INTEGER         REFERENCES "Comment" (commentID),
    CHECK(
        (userID IS NOT NULL AND questionID IS NULL AND answerID IS NULL AND
commentID IS NULL) OR
        (userID IS NULL AND questionID IS NOT NULL AND answerID IS NULL AND
commentID IS NULL) OR
        (userID IS NULL AND questionID IS NULL AND answerID IS NOT NULL AND
commentID IS NULL) OR
        (userID IS NULL AND questionID IS NULL AND answerID IS NULL AND commentID
IS NOT NULL)
    )
);

-- Table: ReportStatus
DROP TABLE IF EXISTS ReportStatus;
CREATE TABLE ReportStatus (
    statusID        INTEGER         PRIMARY KEY,
    reportID        INTEGER         REFERENCES "Report" (reportID),
    state           TEXT            DEFAULT 'unresolved' NOT NULL,
    comment         TEXT,
    responsibleUser INTEGER         REFERENCES "Moderator" (moderatorID)
);

-- Table: MarkedAnswer
DROP TABLE IF EXISTS MarkedAnswer;
CREATE TABLE MarkedAnswer (
    questionID      INTEGER         REFERENCES "Question" (questionID),
    answerID        INTEGER         REFERENCES "Answer" (answerID)
);

-- Table: Following
DROP TABLE IF EXISTS Following;
CREATE TABLE Following (
    userID          INTEGER         REFERENCES "User" (userID),
    labelID         INTEGER         REFERENCES "Label" (labelID)
```

```
);

-- Table: About
DROP TABLE IF EXISTS About;
CREATE TABLE About (
    questionID      INTEGER         REFERENCES "Question" (questionID),
    labelID         INTEGER         REFERENCES "Label" (labelID)
);
```

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