A5: RELATIONAL SCHEMA, VALIDATION AND SCHEMA REFINEMENT - LEARNIT (Q&A)

In this artifact, the Relational Schema obtained by mapping from the Conceptual Data Model is presented. It includes the relation schema, attributes, domains, primary keys, foreign keys and other integrity rules, such as UNIQUE KEY (UK), DEFAULT (DF), NOT NULL (NN) and CHECK (CK).

1. RELATIONAL SCHEMA

Below is the relation schema, in compact notation, for the LearnIt system. Apart from the notation already described above, primary key attributes are underlined and foreign keys attributes in the form attr → origin_table.

R01	User (id_user, username **UK NN**, password **NN**, email **UK NN,** bioDescription, birthdate **NN**, profilePhoto **NN DF** defaultPhoto, banned **NN**, deleted **NN**, points **NN CK** points >=0, id_rank → Rank **NN**, id_role → Role **NN**)
R02	Follow (follower → User, following → User)
R03	Role (id_role, type **NN DF** member **CK** type **IN** RoleType, beginningDate, endDate **CK** endDate > beginningDate)
R04	Rank (id_rank,name **UK NN DF** rookie **CK** name **IN** RankType , minValue **CK** minValue >= 0, maxValue **CK** maxValue > 0 **AND** maxValue > minValue)
R05	Notification (id_notification, description **NN**, type **NN CK** type **IN** NotificationType, view **NN**, date **NN**, user > User **NN**)
R06	Question (id_question → Category, name **NN,** title **NN**, description, date **NN DF** Today, votes **NN DF** 0, photo, deleted **NN**, id_user->User **NN**)
R07	VoteQuestion (username → User, id_question → Question)
R08	Answer (id_answer, text **NN**, date **NN DF** Today **CK** Questions.date < date, votes **NN DF** 0, photo, deleted **NN**, id_question → Question **NN**, user_post → User **NN**)
R09	VoteAnswer (username → User,id_answer → Answer)
R10	Comment (firstAnswer → Answer, secondAnswer → Answer)
R11	BestAnswer (id_bestAnswer → Answer, active **NN**, attributionDate **NN CK** Answer.date < attributionDate, text **NN**, date **NN DF** Today **CK** Questions.date < date, votes **NN DF** 0, photo, deleted **NN**)

R12	Faq (id_faq, question **NN**, answer **NN**)
R13	Report (id_report, date **NN CK (**Questions.date < date **OR** Answer.date < date), reason **NN**, id_question → Question, id_answer → Answer)
R14	UserReport (username → User, id_report → Report)
R15	Category (id_category, name **UK NN**)

Table 1 - Relational schema.

2. DOMAINS

Below is the specification of useful additional domains:

Today	Date DEFAULT current_date
DateTime	date **CK** (VALUE > '1900-01-01' AND VALUE <= now())
DefaultPhoto	String DEFAULT defaultphoto.png
NotificationType	Enum ('question', 'answer', 'comment', 'follow', 'vote')
RankType	Enum ('rookie','beginner','intermediate','enthusiast','advanced','veteran')
RoleType	Enum ('member', 'moderator', 'administrator')

Table 2 - Additional domains.

3. FUNCTIONAL DEPENDENCIES AND SCHEMA VALIDATION

In order to validate the Relational Schema obtained from the Conceptual Model, all non-trivial functional dependencies are now identified and the normalization of all relation schemas (altering previously defined relations if necessary) is accomplished in order to achieve a schema in BCNF. A relation is in BCNF if, for all non-trivial functional dependencies, the left side is a super key of the relation.

Table R01 (User)

Keys: {id_user}, {username}, {email}

Functional Dependencies:

FD0101	{id_user} → {username, password, email, bioDescription, birthdate, profilePhoto, banned, deleted, points, id_rank, id_role}
FD0102	{username} → {id_user, password, email, bioDescription, birthdate, profilePhoto, banned, deleted, points, id_rank, id_role}
FD0103	{email} → {id_user, password, email, bioDescription, birthdate, profilePhoto, banned, deleted, points, id_rank, id_role}

Normal Form: BCNF

Table R02 (Follow)

Keys: {follower}, {following}

Functional Dependencies: none

Normal Form: BCNF

Table R03 (Role)	
Keys: {id_role}	
Functional Dependencies:	
FD0301	{id_role} → {type, beginningDate, endDate}
Normal Form: BCNF	

Table R04 (Rank)	
Keys: {id_rank}, {name}	
Functional Dependencies:	
FD0401	{id_rank} → {name, minValue, maxValue}
FD0402	{name} → {id_rank, minValue, maxValue}
Normal Form: BCNF	

Table R05 (Notification)	
Keys: {id_notification}	
Functional Dependencies:	
FD0501	{id_notification} → {description, type, view, author}
Normal Form: BCNF	

Table R06 (Question)

Keys: {id_question}

Functional Dependencies:

FD0601	{id_question} → {title, description, date, votes, photo, deleted, id_user, name}

Normal Form: BCNF

Table R07 (VoteQuestion)

Keys: {username}, {id_answer}

Functional Dependencies: none

Normal Form: BCNF

Table R08 (Answer)

Keys: {id_answer}

Functional Dependencies:

FD0801	{id_answer} → {text, id_question, date, votes, photo, deleted, username}

Normal Form: BCNF

Table R09 (VoteAnswer)

Keys: {username}, {id_answer}

Functional Dependencies: none

Normal Form: BCNF

Table R10 (Comment)

Keys: {firstAnswer}, {secondAnswer}

Functional Dependencies: none

Normal Form: BCNF

Table R11 (BestAnswer)

Keys: {id_bestAnswer}

Functional Dependencies:

FD1101	{id_bestAnswer} → {active, attributionDate, text, date, votes, photo, deleted}

Normal Form: BCNF

Table R12 (Faq)

Keys: {id_faq}, {question}, {answer}

Functional Dependencies:

FD1201	{id_faq} → {question, answer}
FD1202	{question} → {id_faq, answer}
FD1203	{answer} → {id_faq, question}

Normal Form: BCNF

Table R13 (Report)

Keys: {id_report}

Functional Dependencies:

FD1301	{id_report} → {date,reason, id_question, id_answer}

Normal Form: BCNF

Table R14 (UserReport)

Keys: {username}, {id_report}

Functional Dependencies: none

Normal Form: BCNF

Table R15 (Category)	
Keys: {id_category}, {name}	
Functional Dependencies:	
FD1501	{id_category} → {name}
FD1502	{name} → {id_category}
Normal Form: BCNF	

4. SQL CODE

Below is the necessary SQL code to (re)create the database. It can also be found on the project's page: \database\db.sql

```
DROP TABLE IF EXISTS "user" CASCADE;
DROP TABLE IF EXISTS follow CASCADE;
DROP TABLE IF EXISTS role CASCADE;
DROP TABLE IF EXISTS rank CASCADE;
DROP TABLE IF EXISTS category CASCADE;
DROP TABLE IF EXISTS question CASCADE;
DROP TABLE IF EXISTS voteQuestion CASCADE;
DROP TABLE IF EXISTS answer CASCADE;
DROP TABLE IF EXISTS voteAnswer CASCADE;
DROP TABLE IF EXISTS comment CASCADE;
DROP TABLE IF EXISTS bestAnswer CASCADE;
DROP TABLE IF EXISTS faq CASCADE;
DROP TABLE IF EXISTS report CASCADE;
DROP TABLE IF EXISTS userReport CASCADE;
DROP TABLE IF EXISTS notification CASCADE;

--Types--

DROP TYPE IF EXISTS notificationType;
CREATE TYPE notificationType AS ENUM ('question', 'answer', 'comment',
'follow', 'vote');

DROP TYPE IF EXISTS rankType;
CREATE TYPE rankType AS ENUM ('rookie', 'beginner', 'intermediate',
'enthusiastic', 'advanced', 'veteran');

DROP TYPE IF EXISTS roleType;
CREATE TYPE roleType AS ENUM ('member','moderator', 'administrator');

--Functions --

DROP FUNCTION IF EXISTS defaultphoto();
CREATE FUNCTION defaultphoto() RETURNS text AS $$
BEGIN
    RETURN 'defaultPhoto.png';
END;
$$ LANGUAGE plpgsql;
```

```
DROP FUNCTION IF EXISTS categoriequestionDate(id_category integer);
CREATE FUNCTION categoriequestionDate(id_category integer) RETURNS date AS
$$
BEGIN
    RETURN (select "date" From question where id_category =
question.id_question);
END;
$$ LANGUAGE plpgsql;


DROP FUNCTION IF EXISTS answerDate(id_answer integer);
CREATE FUNCTION answerDate(id_answer integer) RETURNS date AS $$
BEGIN
    RETURN (select "date" From answer where id_answer = answer.id_answer);
END;
$$ LANGUAGE plpgsql;


DROP FUNCTION IF EXISTS reportQuestionDate(id_question integer,data date);
CREATE FUNCTION reportQuestionDate(id_question integer,data date) RETURNS
boolean AS $$
BEGIN
    IF id_question IS NULL THEN RETURN false;
    ELSE
        RETURN (select "date" From question where id_question =
question.id_question)<data;
    END IF;
END;
$$ LANGUAGE plpgsql;


DROP FUNCTION IF EXISTS reportAnswerDate(id_answer integer,data date);
CREATE FUNCTION reportAnswerDate(id_answer integer,data date) RETURNS
boolean AS $$
BEGIN
    IF id_answer IS NULL THEN RETURN false;
    ELSE RETURN (select "date" From answer where id_answer =
answer.id_answer)<data;
    END IF;
END;
$$ LANGUAGE plpgsql;


DROP FUNCTION IF EXISTS rookieID();
CREATE FUNCTION rookieID() RETURNS integer AS $$
DECLARE
    id integer;
BEGIN
    SELECT rank.id_rank INTO id FROM rank WHERE rank.name = 'rookie';
    IF NOT EXISTS id THEN
        RAISE EXCEPTION 'A default rank rookie does not exist in the
database.';
    END IF;
    RETURN id;
END;
$$ LANGUAGE plpgsql;
```

```
-- DOMAINS--

DROP DOMAIN If EXISTS DateTime;
CREATE DOMAIN DateTime AS date
    CONSTRAINT date_ck CHECK (VALUE > '1900-01-01'::date AND VALUE <=now());


--Tables--

CREATE TABLE role(
    id_role SERIAL PRIMARY KEY,
    type roleType NOT NULL DEFAULT 'member',
    beginningDate DateTime,
    endDate DateTime CONSTRAINT endDateBigger_ck CHECK (endDate >
beginningDate)
);


CREATE TABLE rank (
    id_rank SERIAL PRIMARY KEY,
    name rankType NOT NULL DEFAULT 'rookie' CONSTRAINT name_uk UNIQUE,
    minValue integer CONSTRAINT minValue_ck CHECK (minValue>=0),
    maxValue integer CONSTRAINT maxValue_ck CHECK ((maxValue > 0) AND
(maxValue>minValue))
);


CREATE TABLE "user" (
    id_user SERIAL PRIMARY KEY,
    username text NOT NULL CONSTRAINT username_uk UNIQUE,
    password text NOT NULL,
    email text NOT NULL CONSTRAINT email_uk UNIQUE,
    bioDescription text,
    birthdate DateTime NOT NULL,
    profilePhoto text DEFAULT defaultPhoto(),
    points integer NOT NULL CONSTRAINT points_ck CHECK (points >= 0),
    id_rank integer NOT NULL DEFAULT rookieID() REFERENCES rank (id_rank) ON
UPDATE CASCADE,
    banned boolean NOT NULL,
    deleted boolean NOT NULL,
    id_role integer NOT NULL REFERENCES role (id_role) ON UPDATE CASCADE
);


CREATE TABLE follow (
    follower integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE,
    following integer NOT NULL REFERENCES "user" (id_user) ON UPDATE
CASCADE,
    PRIMARY KEY(follower,following)
);


CREATE TABLE notification (
    id_notification SERIAL PRIMARY KEY,
    description text NOT NULL,
    type notificationType NOT NULL,
    view boolean NOT NULL,
    "date" DateTime NOT NULL,
    id_user integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE
);
```

```
CREATE TABLE category(
    id_category SERIAL PRIMARY KEY,
    name text NOT NULL CONSTRAINT categoryname_uk UNIQUE
);


CREATE TABLE question(
    id_question integer PRIMARY KEY NOT NULL REFERENCES category
(id_category) ON UPDATE CASCADE ON DELETE CASCADE,
    name text NOT NULL,
    title text NOT NULL,
    description text,
    "date" DateTime NOT NULL DEFAULT now(),
    votes integer NOT NULL DEFAULT 0,
    photo text,
    deleted boolean NOT NULL,
    id_user integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE
);


CREATE TABLE voteQuestion(
    username integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE,
    id_question integer NOT NULL REFERENCES question (id_question) ON UPDATE
CASCADE,
    PRIMARY KEY (username,id_question)
);


CREATE TABLE answer(
    id_answer SERIAL PRIMARY KEY,
    "text" text NOT NULL,
    "date" DateTime NOT NULL DEFAULT now() CONSTRAINT date_ck CHECK
(categoriequestionDate(id_answer) < "date"),
    votes integer NOT NULL DEFAULT 0,
    photo text,
    deleted boolean NOT NULL,
    id_question integer NOT NULL REFERENCES question (id_question),
    user_post integer NOT NULL REFERENCES "user" (id_user)
);


CREATE TABLE voteAnswer(
    username integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE,
    id_answer integer NOT NULL REFERENCES answer (id_answer) ON UPDATE
CASCADE,
    PRIMARY KEY (username,id_answer)
);


CREATE TABLE comment(
    firstAnswer integer NOT NULL REFERENCES answer (id_answer) ON UPDATE
CASCADE,
    secondAnswer integer NOT NULL REFERENCES answer (id_answer) ON UPDATE
CASCADE,
    PRIMARY KEY (firstAnswer,secondAnswer)
);


CREATE TABLE bestAnswer (
    id_bestAnswer integer PRIMARY KEY REFERENCES answer (id_answer) ON
UPDATE CASCADE ON DELETE CASCADE,
```

```
    attributionDate DateTime NOT NULL CONSTRAINT attributionDate_ck CHECK
(answerDate(id_bestAnswer) < attributionDate),
    "text" text NOT NULL,
    "date" DateTime NOT NULL DEFAULT now() CONSTRAINT date_ck CHECK
(categoriequestionDate(id_bestAnswer) < "date"),
    deleted boolean NOT NULL,
    active boolean NOT NULL,
    votes integer NOT NULL DEFAULT 0,
    photo text
);


CREATE TABLE faq(
    id_faq SERIAL PRIMARY KEY,
    question text NOT NULL CONSTRAINT question_uk UNIQUE,
    answer text NOT NULL CONSTRAINT answer_uk UNIQUE
);


CREATE TABLE report(
    id_report SERIAL PRIMARY KEY,
    "date" DateTime NOT NULL CONSTRAINT reportDate_ck CHECK
(reportQuestionDate(id_question,"date") = true OR
reportAnswerDate(id_answer,"date") = true),
    reason text NOT NULL,
    id_question integer REFERENCES question (id_question),
    id_answer integer REFERENCES answer (id_answer)
);


CREATE TABLE userReport(
    username integer NOT NULL REFERENCES "user" (id_user) ON UPDATE CASCADE,
    id_report integer NOT NULL REFERENCES report (id_report) ON UPDATE
CASCADE,
    PRIMARY KEY (username,id_report)
);
```

REVISION HISTORY

Changes made to the first submission:

1. We added a new domain to restrict dates: DateTime.
2. We have eliminated some decompositions by correcting the primary keys in User table and Question table.

GROUP1834:

- João Augusto dos Santos Lima, up201605314@fe.up.pt
- José Diogo da Cunha Moreira Trindade Martins, up201504761@fe.up.pt
- Juliana Maria Cruz Marques, up201605568@fe.up.pt (Editor)
- Leonor Ribeiro e Sousa Mendes de Freitas, up201207603@fe.up.pt