A5: Relational Schema, validation and schema refinement

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 (**UK**), DEFAULT (**DF**), NOT NULL (**NN**) and CHECK (**CK**).

1. Relational schema

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

R01	admin(id, email **NN UK**, password **NN**)
R02	country(id, name **NN UK**)
R03	member(id, username **NN UK**, email **NN UK**, password **NN**, name **NN**, bio, profile_pic, score **NN**, is_banned **NN DF** false, is_moderator **NN**, country_id → country **NN**)
R04	flag(member_id → member, moderator_id → member, date **NN**, reason **NN**)
R05	follow_member(follower_id → member, following_id → member **CK** following_id <> follower_id)
R06	notification(id, type **NN CK** type **IN** NotificationOrigin, date **NN**, content **NN,** member_id → member **NN**, read **NN DF** false)
R07	topic(id, name **NN**, description, picture)
R08	follow_topic(member_id → member, topic_id → topic)
R09	question(id, title **NN**, content **NN**, date **NN**, views **NN CK** views >= 0, solved **NN DF** false, author_id → member **NN**)
R10	answer(id, content **NN**, date **NN**, views **NN CK** views >= 0, question_id → question **NN**, author_id → member **NN**)
R11	comment(id, content **NN**, date **NN**, question_id → question, answer_id → answer **CK** question_id isnull AND answer_id isnotnull OR question_id isnotnull AND answer_id isnull, author_id → member **NN**)
R12	question_topic(question_id → question, topic_id → topic)
R13	question_rating(question_id → question, member_id → member, rate **NN CK** rate = 1 OR rate = -1)
R14	answer_rating(answer_id → answer, member_id → member, rate **NN CK** rate = 1 OR rate = -1)

R15	comment_rating(<u>comment_id</u> → comment, <u>member_id</u> → member, rate **NN CK** rate = 1 OR rate = -1)
R16	question_report(<u>question_id</u> → question, <u>member_id</u> → member, date **NN**, reason **NN**)
R17	answer_report(<u>answer_id</u> → answer, <u>member_id</u> → member, date **NN**, reason **NN**)
R18	comment_report(<u>comment_id</u> → comment, <u>member_id</u> → member, date **NN**, reason **NN**)

Table 1 - Relational schema.

2. Domains

Below is the specification of useful additional domains:

NotificationOrigin	ENUM('Question', 'Answer', 'Comment', 'Rating', 'Follow', 'Mention')

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 (admin)	
Keys: {id}, {email}	
Functional Dependencies	
FD0101	{id} → {email, password}
FD0102	{email} → {id, password}
NORMAL FORM	BCNF

Table R02 (country)	
Keys: {id}, {name}	
Functional Dependencies	
FD0201	{id} → {name}
FD0202	{name} → {id}
NORMAL FORM	BCNF

Table R03 (member)	
Keys: {id}, {username}, {email}	
Functional Dependencies	
FD0301	{id} → {username, email, password, name, bio, profile_pic, score, is_banned, is_moderator, country_id}
FD0302	{email} → {id, username, password, name, bio, profile_pic, score, is_banned, is_moderator, country_id}
FD0303	{username} → {id, email, password, name, bio, profile_pic, score, is_banned, is_moderator, country_id}
NORMAL FORM	BCNF

Table R04 (flag)	
Keys: {member_id, moderator_id}	
Functional Dependencies	
FD0401	{member_id, moderator_id} → {date, reason}
NORMAL FORM	BCNF

Table R05 (follow_member)

Keys: {follower_id, following_id}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R06 (notification)	
Keys: {id}	
Functional Dependencies	
FD0601	{id} → {type, date, content, member_id, read}
NORMAL FORM	BCNF

Table R07 (topic)	
Keys: {id}	
Functional Dependencies	
FD0701	{id} → {name, description, picture}
NORMAL FORM	BCNF

Table R08 (follow_topic)	
Keys: {member_id, topic_id}	
Functional Dependencies	
(none)	
NORMAL FORM	BCNF

Table R09 (question)	
Keys: {id}	
Functional Dependencies	
FD0901	{id} → {title, content, date, views, solved, author_id}
NORMAL FORM	BCNF

Table R10 (answer)	
Keys: {id}	
Functional Dependencies	
FD1001	{id} → {content, date, views, question_id, author_id}
NORMAL FORM	BCNF

Table R11 (comment)	
Keys: {id}	
Functional Dependencies	
FD1101	{id} → {content, date, question_id, answer_id, author_id}
NORMAL FORM	BCNF

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

Table R13 (question_rating)	
Keys: {question_id, member_id}	
Functional Dependencies	
FD1301	{member_id, question_id} → rate
NORMAL FORM	BCNF

Table R14 (answer_rating)	
Keys: {answer_id, member_id}	
Functional Dependencies	
FD1401	{member_id, answer_id} → {rate}
NORMAL FORM	BCNF

Table R15 (comment_rating)	
Keys: {comment_id, member_id}	
Functional Dependencies	
FD1501	{member_id, comment_id} → {rate}
NORMAL FORM	BCNF

Table R16 (question_report)	
Keys: {question_id, member_id}	
Functional Dependencies	
FD1601	{member_id, question_id} → {date, reason}
NORMAL FORM	BCNF

Table R17 (answer_report)	
Keys: {answer_id, member_id}	
Functional Dependencies	
FD1701	{member_id, answer_id} → {date, reason}
NORMAL FORM	BCNF

Table R18 (comment_report)	
Keys: {comment_id, member_id}	
Functional Dependencies	
FD1801	{member_id, comment_id} → {date, reason}
NORMAL FORM	BCNF

All relations schemas are in BCNF (the left side is always a super key of the relation, as specified in the beginning of the section), therefore the relational schema is also in BCNF and there is no need to be normalised.

4. SQL Code

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

```
-- NotificationOrigin enum
CREATE TYPE notification_origin AS ENUM (
        'Question',
        'Answer',
        'Comment',
        'Rating',
        'Follow',
        'Mention'
);

CREATE TABLE admin (
        id SERIAL NOT NULL,
        email VARCHAR(40) NOT NULL,
        password VARCHAR(35) NOT NULL,
        CONSTRAINT admin_pk PRIMARY KEY (id),
        CONSTRAINT admin_uk UNIQUE (email)
);
```

```
CREATE TABLE country (
	id SERIAL NOT NULL,
	name VARCHAR(30) NOT NULL,
	CONSTRAINT country_pk PRIMARY KEY (id),
	CONSTRAINT country_uk UNIQUE (name)
);

CREATE TABLE member (
	id SERIAL NOT NULL,
	username VARCHAR(20) NOT NULL,
	email VARCHAR(40) NOT NULL,
	password VARCHAR(35) NOT NULL,
	name VARCHAR(35) NOT NULL,
	bio text,
	profilePic text,
	score INTEGER NOT NULL,
	is_banned BOOLEAN NOT NULL DEFAULT false,
	is_moderator BOOLEAN NOT NULL,
	country_id INTEGER NOT NULL,
	CONSTRAINT member_pk PRIMARY KEY (id),
	CONSTRAINT member_username_uk UNIQUE (username),
	CONSTRAINT member_email_uk UNIQUE (email),
	CONSTRAINT member_fk FOREIGN KEY (country_id) REFERENCES country (id) ON UPDATE
CASCADE ON DELETE RESTRICT
);

CREATE TABLE flag (
	member_id INTEGER NOT NULL,
	moderator_id INTEGER NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	reason text NOT NULL,
	CONSTRAINT flag_pk PRIMARY KEY (member_id, moderator_id),
	CONSTRAINT flag_member_fk FOREIGN KEY (member_id) REFERENCES member (id) ON UPDATE
CASCADE ON DELETE CASCADE,
	CONSTRAINT flag_moderator_fk FOREIGN KEY (member_id) REFERENCES member (id) ON
UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE follow_member (
	follower_id INTEGER NOT NULL,
	following_id INTEGER NOT NULL,
	CONSTRAINT follow_member_ck CHECK (follower_id <> following_id),
	CONSTRAINT follow_member_pk PRIMARY KEY (follower_id, following_id),
	CONSTRAINT follow_member_follower_fk FOREIGN KEY (follower_id) REFERENCES member
(id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT follow_member_following_fk FOREIGN KEY (following_id) REFERENCES member
(id) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE notification (
	id SERIAL NOT NULL,
	type notification_origin NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
```

```
	content text NOT NULL,
	member_id INTEGER NOT NULL,
	read BOOLEAN NOT NULL DEFAULT false,
	CONSTRAINT notification_pk PRIMARY KEY (id),
	CONSTRAINT notification_fk FOREIGN KEY (member_id) REFERENCES member (id)
);

CREATE TABLE topic (
	id SERIAL NOT NULL,
	name VARCHAR(25) NOT NULL,
	description text,
	picture text,
	CONSTRAINT topic_pk PRIMARY KEY (id)
);

CREATE TABLE follow_topic (
	member_id INTEGER NOT NULL,
	topic_id INTEGER NOT NULL,
	CONSTRAINT follow_topic_pk PRIMARY KEY (member_id, topic_id),
	CONSTRAINT follow_topic_member_fk FOREIGN KEY (member_id) REFERENCES member (id) ON
UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT follow_topic_topic_fk FOREIGN KEY (topic_id) REFERENCES topic (id) ON
UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE question (
	id SERIAL NOT NULL,
	title text NOT NULL,
	content text NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	views INTEGER NOT NULL CHECK (views >= 0),
	solved BOOLEAN NOT NULL DEFAULT false,
	author_id INTEGER NOT NULL,
	CONSTRAINT question_pk PRIMARY KEY (id),
	CONSTRAINT question_fk FOREIGN KEY (author_id) REFERENCES member (id) ON UPDATE
CASCADE ON DELETE CASCADE
);

CREATE TABLE answer (
	id SERIAL NOT NULL,
	content text NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	views INTEGER NOT NULL CHECK (views >= 0),
	question_id INTEGER NOT NULL,
	author_id INTEGER NOT NULL,
	CONSTRAINT answer_pk PRIMARY KEY (id),
	CONSTRAINT answer_member_fk FOREIGN KEY (author_id) REFERENCES member (id) ON
UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT answer_question_fk FOREIGN KEY (question_id) REFERENCES question (id)
);

CREATE TABLE comment (
	id SERIAL NOT NULL,
```

```
	content text NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	question_id INTEGER,
	answer_id INTEGER,
	author_id INTEGER NOT NULL,
	CONSTRAINT comment_ck CHECK ((question_id IS NULL AND answer_id IS NOT NULL) OR
(question_id IS NOT NULL AND answer_id IS NULL)), --XOR
	CONSTRAINT comment_pk PRIMARY KEY (id),
	CONSTRAINT comment_question_fk FOREIGN KEY (question_id) REFERENCES question (id)
ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT comment_member_fk FOREIGN KEY (author_id) REFERENCES member (id) ON
UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT comment_answer_fk FOREIGN KEY (answer_id) REFERENCES answer (id) ON
UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE question_topic (
	question_id INTEGER NOT NULL,
	topic_id INTEGER NOT NULL,
	CONSTRAINT question_topic_pk PRIMARY KEY (question_id, topic_id),
	CONSTRAINT question_topic_question_fk FOREIGN KEY (question_id) REFERENCES question
(id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT question_topic_topic_fk FOREIGN KEY (topic_id) REFERENCES topic (id) ON
UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE question_rating (
	question_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	rate INTEGER NOT NULL CHECK (rate = 1 OR rate = -1),
	CONSTRAINT question_rating_pk PRIMARY KEY (question_id, member_id),
	CONSTRAINT question_rating_question_fk FOREIGN KEY (question_id) REFERENCES
question (id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT question_rating_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE answer_rating (
	answer_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	rate INTEGER NOT NULL CHECK (rate = 1 OR rate = -1),
	CONSTRAINT answer_rating_pk PRIMARY KEY (answer_id, member_id),
	CONSTRAINT answer_rating_answer_fk FOREIGN KEY (answer_id) REFERENCES answer (id)
ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT answer_rating_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE comment_rating (
	comment_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	rate INTEGER NOT NULL CHECK (rate = 1 OR rate = -1),
	CONSTRAINT comment_rating_pk PRIMARY KEY (comment_id, member_id),
```

```
	CONSTRAINT comment_rating_comment_fk FOREIGN KEY (comment_id) REFERENCES comment
(id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT comment_rating_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE question_report (
	question_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	reason text NOT NULL,
	CONSTRAINT question_report_pk PRIMARY KEY (question_id, member_id),
	CONSTRAINT question_report_question_fk FOREIGN KEY (question_id) REFERENCES
question (id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT question_report_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE answer_report (
	answer_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	reason text NOT NULL,
	CONSTRAINT answer_report_pk PRIMARY KEY (answer_id, member_id),
	CONSTRAINT answer_report_answer_fk FOREIGN KEY (answer_id) REFERENCES answer (id)
ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT answer_report_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE comment_report (
	comment_id INTEGER NOT NULL,
	member_id INTEGER NOT NULL,
	"date" TIMESTAMP WITH TIME zone NOT NULL,
	reason text NOT NULL,
	CONSTRAINT comment_report_pk PRIMARY KEY (comment_id, member_id),
	CONSTRAINT comment_report_comment_fk FOREIGN KEY (comment_id) REFERENCES comment
(id) ON UPDATE CASCADE ON DELETE CASCADE,
	CONSTRAINT comment_report_member_fk FOREIGN KEY (member_id) REFERENCES member (id)
ON UPDATE CASCADE ON DELETE CASCADE
);
```

Revision history

Changes made to the first submission:

1. Added default values to some attributes in the relational schema
2. Updated SQL Code (remove separate constraint declaration)

GROUP1725, 25/03/2018
António Cunha Seco Fernandes de Almeida, up201505836@fe.up.pt

Bruno Manuel Nascimento Costa Galvinas Piedade, up201505668@fe.up.pt
Diogo Luis Rey Torres, up201506428@fe.up.pt
João Paulo Madureira Damas, up201504088@fe.up.pt