# 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).

## 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 | User (id\_user, username UK NN, password NN, email UNIQUE 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.*

## Domains

Below is the specification of useful additional domains:

|  |  |
| --- | --- |
| Today | Date DEFAULT current\_date |
| 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}, {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 | {username,email} → {id\_user, password, bioDescription, birthdate, profilePhoto, banned, deleted, points, id\_rank, id\_role} |
| FD0104 | {points} → {id\_rank} |
| FD0105 | {banned} → {deleted} |
| Normal Form: | Not in BCNF |
| Decomposition: |  |
| D0101 | User1(id\_user, username, password, email, bioDescription, birthdate, profilePhoto, banned, points, id\_role) |
| D0102 | User2(points,id\_rank) |
| D0103 | User3(banned,deleted) |

|  |  |
| --- | --- |
| 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}, {id\_user, title, description, photo, deleted} | | |
| Functional Dependencies: | |  |
| FD0601 | {id\_question} → {title, description, date, votes, photo, deleted, id\_user, name} | |
| FD0602 | {id\_user, title, description, photo, deleted} → {id\_question, date, votes, name} | |
| Normal Form: BCNF | |  |

|  |  |
| --- | --- |
| Table R07 (VoteQuestion) |  |
| Keys: {username}, {id\_answer} | |
| Functional Dependencies: none |  |
| Normal Form: BCNF |  |

|  |  |
| --- | --- |
| Table R08 (Answer) |  |
| Keys: {id\_answer}, {text, id\_question, username, photo, deleted} | |
| Functional Dependencies: |  |
| FD0801 | {id\_answer} → {text, id\_question, date, votes, photo, deleted, username} |
| FD0802 | {text, id\_question, username, photo, deleted} → {id\_answer, date, votes} |
| 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} |
| FD1102 | {deleted} → {active} |
| Normal Form: Not in BCNF |  |
| Decomposition: |  |
| D1101 | BestAnswer1(id\_bestAnswer, attributionDate, text, date, votes, photo, deleted) |
| D1102 | BestAnswer2(deleted, active) |

|  |  |
| --- | --- |
| 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 |  |

1. 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 user1 CASCADE;

DROP TABLE IF EXISTS user2 CASCADE;

DROP TABLE IF EXISTS user3 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 bestAnswer2 CASCADE;

DROP TABLE IF EXISTS bestAnswer1 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;

DROP TYPE IF EXISTS notificationType;

DROP TYPE IF EXISTS rankType;

DROP TYPE IF EXISTS roleType;

DROP FUNCTION IF EXISTS defaultphoto();

DROP FUNCTION IF EXISTS categoriequestionDate(id\_category integer);

DROP FUNCTION IF EXISTS answerDate(id\_answer integer);

DROP FUNCTION IF EXISTS reportQuestionDate(id\_question integer,data date);

DROP FUNCTION IF EXISTS reportAnswerDate(id\_answer integer,data date);

--Types--

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

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

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

--Functions --

CREATE FUNCTION defaultphoto() RETURNS text AS $$

BEGIN

RETURN 'defaultPhoto.png';

END;

$$ LANGUAGE plpgsql;

CREATE FUNCTION categoriequestionDate(id\_category integer) RETURNS date AS $$

BEGIN

RETURN (select "date" From question where $1 = question.id\_question);

END;

$$ LANGUAGE plpgsql;

CREATE FUNCTION answerDate(id\_answer integer) RETURNS date AS $$

BEGIN

RETURN (select "date" From answer where $1 = answer.id\_answer);

END;

$$ LANGUAGE plpgsql;

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 $1 = question.id\_question)<data;

END IF;

END;

$$ LANGUAGE plpgsql;

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 $1 = answer.id\_answer)<data;

END IF;

END;

$$ LANGUAGE plpgsql;

--Tables--

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 user2 (

id\_user SERIAL PRIMARY KEY,

points integer NOT NULL CONSTRAINT points\_ck CHECK (points >= 0),

id\_rank integer NOT NULL REFERENCES rank (id\_rank)

);

CREATE TABLE user3 (

id\_user SERIAL PRIMARY KEY ,

banned boolean NOT NULL,

deleted boolean NOT NULL

);

CREATE TABLE user1 (

id\_user SERIAL PRIMARY KEY,

username text NOT NULL CONSTRAINT username\_uk UNIQUE,

password text NOT NULL,

email text NOT NULL,

bioDescription text,

birthdate date NOT NULL,

profilePhoto text DEFAULT defaultPhoto(),

id\_user2 integer NOT NULL REFERENCES user2 (id\_user) ON UPDATE CASCADE ON DELETE CASCADE,

id\_user3 integer NOT NULL REFERENCES user3 (id\_user) ON UPDATE CASCADE ON DELETE CASCADE,

id\_role integer NOT NULL

);

CREATE TABLE follow (

follower integer NOT NULL REFERENCES user1 (id\_user) ON UPDATE CASCADE,

following integer NOT NULL REFERENCES user1 (id\_user) ON UPDATE CASCADE,

PRIMARY KEY(follower,following)

);

CREATE TABLE role (

id\_role SERIAL PRIMARY KEY,

type roleType NOT NULL DEFAULT 'member',

beginningDate date,

endDate date CONSTRAINT endDateBigger\_ck CHECK (endDate > beginningDate)

);

CREATE TABLE notification (

id\_notification SERIAL PRIMARY KEY,

description text NOT NULL,

type notificationType NOT NULL,

view boolean NOT NULL,

"date" date NOT NULL,

id\_user integer NOT NULL REFERENCES user1 (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" date NOT NULL DEFAULT now(),

votes integer NOT NULL DEFAULT 0,

photo text,

deleted boolean NOT NULL,

id\_user integer NOT NULL REFERENCES user1 (id\_user) ON UPDATE CASCADE

);

CREATE TABLE voteQuestion(

username integer NOT NULL REFERENCES user1 (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" date 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 user1 (id\_user)

);

CREATE TABLE voteAnswer(

username integer NOT NULL REFERENCES user1 (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 bestAnswer2 (

id\_bestAnswer SERIAL PRIMARY KEY ,

deleted boolean NOT NULL,

active boolean NOT NULL

);

CREATE TABLE bestAnswer1 (

id\_bestAnswer integer PRIMARY KEY REFERENCES answer (id\_answer) ON UPDATE CASCADE ON DELETE CASCADE,

attributionDate date NOT NULL CONSTRAINT attributionDate\_ck CHECK (answerDate(id\_bestAnswer) < attributionDate),

"text" text NOT NULL,

"date" date NOT NULL DEFAULT now() CONSTRAINT date\_ck CHECK (categoriequestionDate(id\_bestAnswer) < "date"),

id\_bestAnswer2 integer NOT NULL REFERENCES bestAnswer2 (id\_bestAnswer) ON UPDATE CASCADE ON DELETE CASCADE ,

votes integer NOT NULL DEFAULT 0,

photo text

);

CREATE TABLE faq(

id\_faq SERIAL PRIMARY KEY,

question text NOT NULL,

answer text NOT NULL

);

CREATE TABLE report(

id\_report SERIAL PRIMARY KEY,

"date" date NOT NULL CONSTRAINT reportDate\_ck CHECK (reportQuestionDate(id\_question,"date") = true OR reportAnswerDate(id\_answer,"date") = false),

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 user1 (id\_user) ON UPDATE CASCADE,

id\_report integer NOT NULL REFERENCES report (id\_report) ON UPDATE CASCADE,

PRIMARY KEY (username,id\_report)

);

## 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