Datenbankpraktikum 2020

Abgabe Aufgabe 1

Lukas Hempel & Thomas Pause
Matrikelnummern: 3739268 & 3720245
Bachelor Informatik
Betreuer: Martin Franke

Termin 1.Testat: 28.05.2020

1 Relationenmodell

Legende:

- primary key
- foreign key
- foreign as primary key

Tabellen:

- *Tag* (id, name);
- *TagClass* (id, name);
- *Continent* (id, name);
- *Country* (id, name, continent_id);
- *City* (id, name, country_id);
- *Person* (id, firstName, lastName, gender, birthday, email, speaks, browserUsed, locationIP, city_id);
- *Company* (id, name, country_id);
- *University* (id, name, city_id);
- *Forum* (id, title, creationDate, moderator);
- *Post* (id, language, imageFile, creationDate, browserUsed, locationIP, content, length, forum_id, author_id, country_id);
- *Comment* (id, creationDate, browserUsed, locationIP, content, length, author_id, country_id, reply_to_post_id, reply_to_comment_id);
- *Forum_hasMember_Person* (person_id, forum_id, joinDate);
- *Forum_hasTag_Tag* (forum_id, tag_id);
- *Tag_hasType_TagClass* (tag_id, tagClass_id);
- *TagClass_isSubclassOf_TagClass* (tag_parent_id, tag_child_id);
- *Post_hasTag_Tag* (post_id, tag_id);
- *Comment_hasTag_Tag* (comment_id, tag_id);
- *Person_knows_Person* (person_1_id, person_2_id, creationDate);

- *Person_studyAt_University* (`person_id`, `university_id`, classYear);
- *Person_workAt_Company* (`person_id`, `company_id`, workFrom);
- *Person_likes_Post* (`person_id`, `post_id`, creationDate);
- *Person_likes_Comment* (`person_id`, `comment_id`, creationDate);
- *Person_hasInterest_Tag* (`person_id`, `tag_id`);

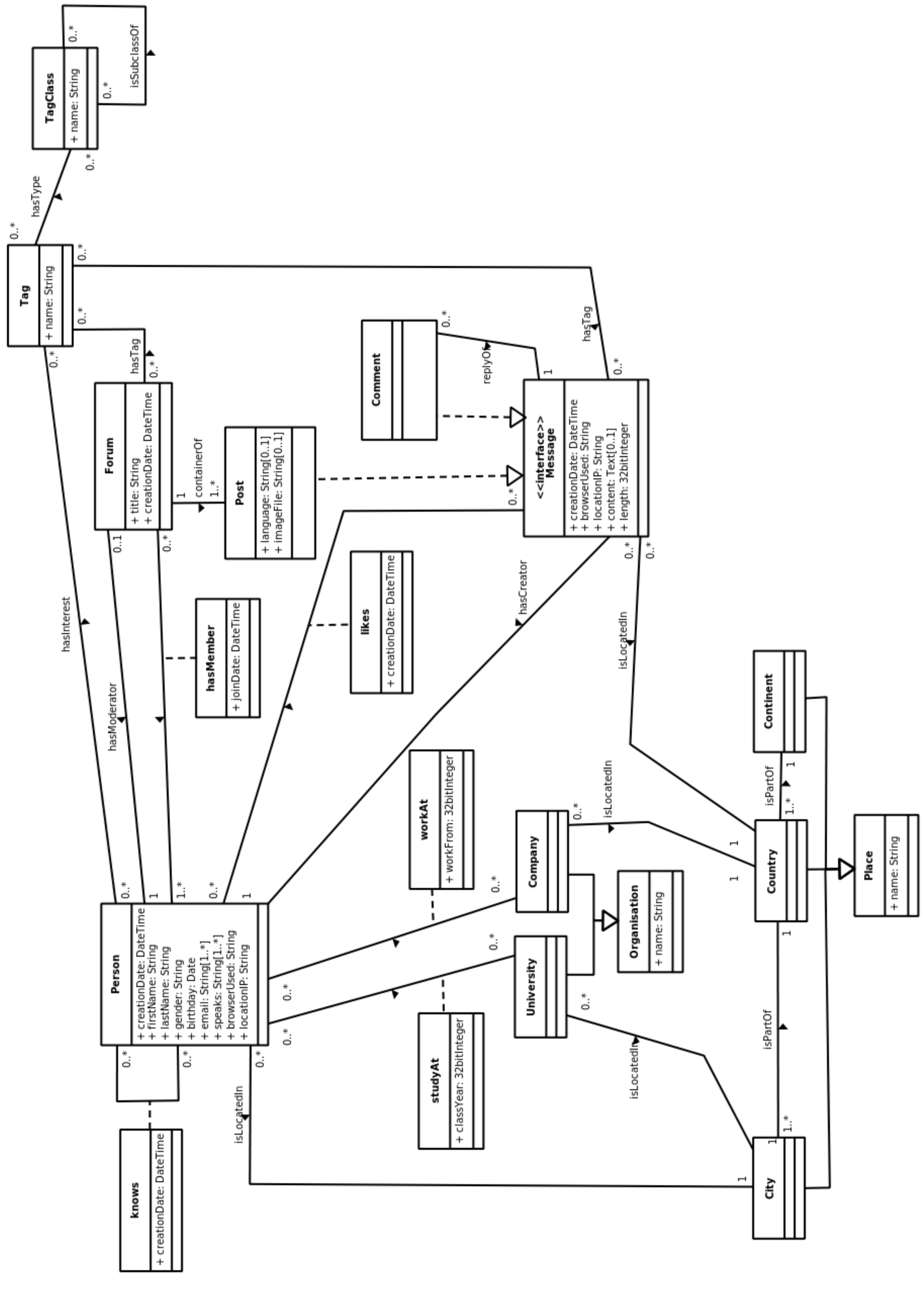
2 Tabellen (SQL) incl. Constraints

```
CREATE FUNCTION valid_email(b boolean, v VARCHAR)
    RETURNS boolean
    AS $$
    SELECT $2 ~ '^[\w\.-]+@[\w+\.-]+\.[\w]{2,4}$' AS result $$
    LANGUAGE sql;

CREATE OPERATOR =%= (
    PROCEDURE = valid_email,
    LEFTARG = boolean,
    RIGHTARG = varchar
);

CREATE TABLE tag(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    url TEXT
);

CREATE TABLE tagclass(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    url TEXT
);

CREATE TABLE continent(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    url TEXT
);

CREATE TABLE country(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    continent_id BIGINT NOT NULL REFERENCES continent(id) ON DELETE
       CASCADE ON UPDATE CASCADE,
    url TEXT
);

CREATE TABLE city(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    country_id BIGINT NOT NULL REFERENCES country(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    url TEXT
);

CREATE TABLE person(
    id BIGSERIAL PRIMARY KEY,
    creationDate TIMESTAMP NOT NULL,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(100) NOT NULL,
    gender VARCHAR(7) NOT NULL,
    birthday Date NOT NULL,
    email VARCHAR[] NOT NULL, -- ArrayType bc [1..*]
    speaks VARCHAR[] NOT NULL, -- ArrayType bc [1..*]
    browserUsed VARCHAR(50) NOT NULL,
    locationIP VARCHAR(40) NOT NULL,
    city_id BIGINT NOT NULL REFERENCES city(id) ON DELETE CASCADE ON
       UPDATE CASCADE,

    CONSTRAINT birthday_not_in_future CHECK (birthday <= NOW()::DATE),
    CONSTRAINT vaild_email CHECK (TRUE =%= ALL(email))
);

CREATE TABLE company(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    url TEXT,
    country_id BIGINT NOT NULL REFERENCES country(id) ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE university(
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(200) NOT NULL,
    url TEXT,
    city_id BIGINT NOT NULL REFERENCES city(id) ON DELETE CASCADE ON
       UPDATE CASCADE
);

CREATE TABLE forum(
    id BIGSERIAL PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    creationDate TIMESTAMP NOT NULL,
    moderator BIGINT REFERENCES person(id) ON DELETE SET NULL ON UPDATE
        CASCADE
);

CREATE TABLE post(
    id BIGSERIAL PRIMARY KEY,
    language VARCHAR(2), -- Achtung, hier soll Null erlaubt sein
    imageFile VARCHAR(150), -- Achtung, hier soll Null erlaubt sein
    creationDate TIMESTAMP NOT NULL,
    browserUsed VARCHAR(50) NOT NULL,
    locationIP VARCHAR(40) NOT NULL,
    content TEXT, -- Achtung, hier soll Null erlaubt sein
    length INT NOT NULL,
    forum_id BIGINT NOT NULL REFERENCES forum(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    author_id BIGINT REFERENCES person(id) ON DELETE SET NULL ON UPDATE
        CASCADE,
    country_id BIGINT NOT NULL REFERENCES country(id) ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE comment(
    id BIGSERIAL PRIMARY KEY,
    creationDate TIMESTAMP NOT NULL,
    browserUsed VARCHAR(50) NOT NULL,
    locationIP VARCHAR(40) NOT NULL,
    content TEXT, -- Achtung, hier soll Null erlaubt sein
    length INT NOT NULL,
    author_id BIGINT REFERENCES person(id) ON DELETE SET NULL ON UPDATE
        CASCADE,
    country_id BIGINT NOT NULL REFERENCES country(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    reply_to_post_id BIGINT REFERENCES post(id) ON DELETE SET NULL ON
       UPDATE CASCADE,
    reply_to_comment_id BIGINT REFERENCES comment(id) ON DELETE SET
       NULL ON UPDATE CASCADE,

    CONSTRAINT belongs_to_message_or_post CHECK (((reply_to_comment_id
       IS NOT NULL) AND (reply_to_post_id IS NULL)) OR ((
       reply_to_comment_id IS NULL) AND (reply_to_post_id IS NOT NULL))
       ) -- noch schauen ob das so geht, besser XOR!
);

CREATE TABLE forum_hasMember_person(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    forum_id BIGINT NOT NULL REFERENCES forum(id) ON DELETE CASCADE ON
```

```
       UPDATE CASCADE,
    joinDate TIMESTAMP NOT NULL,
    PRIMARY KEY (person_id, forum_id)
);

CREATE TABLE forum_hasTag_tag(
    forum_id BIGINT NOT NULL REFERENCES forum(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    tag_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    PRIMARY KEY (forum_id, tag_id)
);

CREATE TABLE tag_hasType_tagclass(
    tag_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    tagclass_id BIGINT NOT NULL REFERENCES tagclass(id) ON DELETE
       CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (tag_id, tagclass_id)
);

CREATE TABLE tagclass_isSubclassOf_tagclass(
    tag_parent_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    tag_child_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    PRIMARY KEY (tag_parent_id, tag_child_id)
);

CREATE TABLE post_hasTag_tag(
    post_id BIGINT NOT NULL REFERENCES post(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    tag_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    PRIMARY KEY (post_id, tag_id)
);

CREATE TABLE comment_hasTag_tag(
    comment_id BIGINT NOT NULL REFERENCES comment(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    tag_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    PRIMARY KEY (comment_id, tag_id)
);

CREATE TABLE person_knows_person(
    person_1_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    person_2_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
```

```
        ON UPDATE CASCADE,
    creationDate TIMESTAMP NOT NULL,
    PRIMARY KEY (person_1_id, person_2_id)
);

CREATE TABLE person_studyAt_university(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    university_id BIGINT NOT NULL REFERENCES university(id) ON DELETE
       CASCADE ON UPDATE CASCADE,
    classYear INT NOT NULL,
    PRIMARY KEY (person_id, university_id)
);

CREATE TABLE person_workAt_company(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    company_id BIGINT NOT NULL REFERENCES company(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    workFrom INT NOT NULL,
    PRIMARY KEY (person_id, company_id)
);

CREATE TABLE person_likes_post(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    post_id BIGINT NOT NULL REFERENCES post(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    creationDate TIMESTAMP NOT NULL,
    PRIMARY KEY (person_id, post_id)
);

CREATE TABLE person_likes_comment(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    comment_id BIGINT NOT NULL REFERENCES comment(id) ON DELETE CASCADE
        ON UPDATE CASCADE,
    creationDate TIMESTAMP NOT NULL,
    PRIMARY KEY (person_id, comment_id)
);

CREATE TABLE person_hasInterest_Tag(
    person_id BIGINT NOT NULL REFERENCES person(id) ON DELETE CASCADE
       ON UPDATE CASCADE,
    tag_id BIGINT NOT NULL REFERENCES tag(id) ON DELETE CASCADE ON
       UPDATE CASCADE,
    PRIMARY KEY (person_id, tag_id)
);
```

3 UML-Diagramm


TagClass
+ name: String
isSubclassOf
hasType
Tag
+ name: String
hasTag
hasTag
Forum
+ title: String
+ creationDate: DateTime
containerOf
Post
+ language: String[0..1]
+ imageFile: String[0..1]
Comment
replyOf
<<interface>>
Message
+ creationDate: DateTime
+ browserUsed: String
+ locationIP: String
+ content: Text[0..1]
+ length: 32bitInteger
hasInterest
hasModerator
hasMember
+ joinDate: DateTime
likes
+ creationDate: DateTime
hasCreator
isLocatedIn
Continent
isPartOf
workAt
+ workFrom: 32bitInteger
Company
isLocatedIn
Country
Place
+ name: String
Organisation
+ name: String
Person
+ creationDate: DateTime
+ firstName: String
+ lastName: String
+ gender: String
+ birthday: Date
+ email: String[1..*]
+ speaks: String[1..*]
+ browserUsed: String
+ locationIP: String
University
studyAt
+ classYear: 32bitInteger
isLocatedIn
isPartOf
isLocatedIn
knows
+ creationDate: DateTime
City