A5: Relational Schema, validation and schema refinement

1. Relational Schema

Relation schemas are specified in the compact notation:

Identifier	Relation
R01	user(id, username **NN** **UK**, password **NN**, name **NN**, bio, dateOfBirth **CK** dateOfBirth < Today, birthLocation, deactivationDate **DF** NULL, warns **DF** 0, location → City **NN**, rating **IN** RatingLevel)
R02	admin(id, userId → user)
R03	band(id, name **NN** **UK**, creationDate, ceaseDate, location → City **NN**)
R04	content(id, text **NN**, date **DF** Today)
R05	post(id, private **NN**, contentId → content **NN**, posterId → user **NN**, bandId → band)
R06	message(id, contentId → content **NN**, senderId → user **NN**, receiverId → user, bandId → band)
R07	comment(id, contentId → content **NN**, commenterId → user **NN**, postId → post **NN**)
R08	country(id, name **NN** **UK**)
R09	city(id, name **NN**, countryId → Country)
R10	genre(id, name **NN**, creatingAdminId → admin)
R11	skill(id, name **NN**, creatingAdminId → admin)
R12	report(id, text **NN**, date **DF** Today, reportedContentId → content, reportedUserId → user, reportedBandId → band, reporterId → user)
R13	ban(id, reason **NN**, banDate **DF** Today, ceaseDate, adminId → admin)

R14	notification_trigger(id, date **DF** Today, type **IN** NotificationTypes, originUserFollower → user_follower, originBandFollower → band_follower, originMessage → message, originComment → comment, originBandApplication → Application, originBandInvitation → band_invitation, originUserWarning → user_warning, originBandWarning → band_warning)
R15	user_skill(userId → user, skillId → skill , level **IN** SkillLevels **NN**)
R16	user_follower(id, (followingUserId → user, followedUserId → user) **UK**)
R17	user_rating(ratingUserId → user, ratedUserId → user, rate **IN** RatingLevel)
R18	user_genre(userId → user, genreId → genre)
R19	user_warning(id, (adminId → admin, userId → user) **NN**)
R20	band_genre(bandId → band, genreId → genre)
R21	band_membership(bandId → band, userId → user, isOwner **NN**, initialDate, ceaseDate)
R22	band_rating(ratedBandId → band, ratingUserId → userId, rate **IN** RatingLevel)
R23	band_warning(id, (adminId → admin, bandId → band) **NN**)
R24	band_follower(id, (userId → user, bandId → band) **UK NN**)
R25	band_application(id, (userId → user, bandId → band) **NN**, date **DF** Today, lastStatusDate, status **IN** ApplicationStatus)
R26	band_invitation(id, (userId → user, bandId → band) **NN**, date **DF** Today, lastStatusDate, status **IN** InvitationStatus)
R27	user_notification(notificationTriggerId → notification_trigger, userId → user , seen)

where UK means UNIQUE KEY, NN means NOT NULL, DF means DEFAULT and CK means CHECK.

2. Domains

Domain Name	Domain Specification
Today	DATE DEFAULT CURRENT_DATE
RatingLevel	{1,2,3,4,5}
SkillLevels	{1,2,3,4,5}
ApplicationStatus	ENUM('Accepted', 'Pending', 'Rejected', 'Cancelled')
NotificationTypes	ENUM('Warning, 'Post', 'Comment', 'Follow', 'Band', 'Invitation', 'Application', 'Message')
InvitationStatus	ENUM('Accepted', 'Pending', 'Rejected', 'Cancelled')

3. Functional Dependencies and schema validation

To validate the Relational Schema obtained from the Conceptual Model, all functional dependencies are identified and the normalization of all relation schemas is accomplished.

Table R01 (user)
Keys: { id, username }
Functional Dependencies
FD0101 : {id} → {username, password, name, bio, dateOfBirth, birthLocation, deactivationDate, warns, location, rating}
FD0102 : {username} → {id, password, name, bio, dateOfBirth, birthLocation, deactivationDate, warns, location, rating}
Normal Form : BCNF

Table R02 (admin)
Keys: { id }
Functional Dependencies
FD0201 : {id} → {userId}
Normal Form : BCNF

Table R03 (band)
Keys: { id, name }
Functional Dependencies
FD0301 : {id} → {name, creationDate, ceaseDate, location}
FD0302 : {name} → {id, creationDate, ceaseDate, location}
Normal Form : BCNF

Table R04 (content)
Keys: { id }
Functional Dependencies
FD0401 : {id} → {text, date}
Normal Form : BCNF

Table R05 (post)
Keys: { id }
Functional Dependencies
FD0501 : {id} → {private, contentId, posterId, bandId}
Normal Form : BCNF

Table R06 (message)
Keys: { id }
Functional Dependencies
FD0601 : {id} → {contentId, senderId, receiverId, bandId}
Normal Form : BCNF

Table R07 (comment)
Keys: { id }
Functional Dependencies
FD0701 : {id} → {contentId, commenterId, postId}
Normal Form : BCNF

Table R08 (country)
Keys: { id, name }
Functional Dependencies
FD0801 : {id} → {name}
FD0802 : {name} → {id}
Normal Form : BCNF

Table R09 (city)
Keys: { id }
Functional Dependencies
FD0901 : {id} → {name, countryId}
Normal Form : BCNF

Table R10 (genre)
Keys: { id }
Functional Dependencies
FD1001 : {id} → {name, creatingAdminId}
Normal Form : BCNF

Table R11 (skill)

Keys: { id }
Functional Dependencies
FD1101 : {id} → {name, creatingAdminId}
Normal Form : BCNF

Table R12 (report)
Keys: { id }
Functional Dependencies
FD1201 : {id} → {text, date, reportedContentId, reportedUserId, reportedBandId, reporterId}
Normal Form : BCNF

Table R13 (ban)
Keys: { id }
Functional Dependencies
FD1301 : {id} → {reason, banDate, ceaseDate, adminId}
Normal Form : BCNF

Table R14 (notification_trigger)
Keys: { id }
Functional Dependencies
FD1401 : {id} → {date, type, originUserFollower, originBandFollower, originMessage, originComment, originBandApplication, originBandInvitation, originUserWarning, originBandWarning}
Normal Form : BCNF

Table R15 (user_skill)
Keys: { {userId,skillId } }
Functional Dependencies
FD1501 : {userId, skillId} → {level}
Normal Form : BCNF

Table R16 (user_follower)
Keys: { id, (followingUserId, followedUserId) }
Functional Dependencies
FD1601 : {id} → {followingUserId, followedUserId}
FD1601 : {followingUserId, followedUserId} → {id}
Normal Form : BCNF

Table R17 (user_rating)
Keys: { {ratingUserId, ratedUserId} }
Functional Dependencies
FD1701 : {ratingUserId, ratedUserId} → {rating}
Normal Form : BCNF

Table R18 (user_genre)
Keys: { userId, genreId }
Functional Dependencies
(none)
Normal Form : BCNF

Table R19 (user_warning)

Keys: { id, (adminId, userId) }
Functional Dependencies
FD1901 : {id} → {adminId, userId}
FD1901 : {adminId, userId} → {id}
Normal Form : BCNF

Table R20 (band_genre)
Keys: { {bandId, genreId} }
Functional Dependencies
(none)
Normal Form : BCNF

Table R21 (band_membership)
Keys: { {bandId, userId} }
Functional Dependencies
FD2101 : {bandId, userId} → {isOwner, initialDate, ceaseDate}
Normal Form : BCNF

Table R22 (band_rating)
Keys: { {ratedBandId, ratingUserId} }
Functional Dependencies
FD2201 : {ratedBandId, ratingUserId} → {rating}
Normal Form : BCNF

Table R23 (band_warning)

Keys: { id, (adminId, bandId) }
Functional Dependencies
FD2301 : {id} → {adminId, bandId}
FD2301 : {adminId, bandId} → {id}
Normal Form : BCNF

Table R24 (band_follower)
Keys: { id, (followingUserId, followedBandId) }
Functional Dependencies
FD2401 : {id} → {followingUserId, followedBandId}
FD2401 : {followingUserId, followedBandId} → {id}
Normal Form : BCNF

Table R25 (band_application)
Keys: { id, (userId, bandId) }
Functional Dependencies
FD2501 : {id} → {userId, bandId, date, lastStatusDate, status}
FD2501 : {userId, bandId} → {id, date, lastStatusDate, status}
Normal Form : BCNF

Table R26 (band_invitation)
Keys: { id, (userId, bandId) }
Functional Dependencies
FD2601 : {id} → {bandId, userId, date, lastStatusDate, status}
FD2602 : {bandId, userId} → {id, date, lastStatusDate, status}

Normal Form : BCNF

Table R27 (user_notification)
Keys: { (notificationTriggerId, userId) }
Functional Dependencies
FD2701 : {notificationTriggerId, userId} → {seen}
Normal Form : BCNF

AS all relations schemas are in the Boyce–Codd Normal Form (BCNF), the relational schema is also in the BCNF and therefore there is no need to be refined using normalisation.

4. SQL Code

```
\c postgres;

DROP DATABASE IF EXISTS lbaw1712;
CREATE DATABASE lbaw1712;

\c lbaw1712;

/*****************************************************/
/****************** Country *************************/
/*****************************************************/

CREATE TABLE country (
    id SERIAL NOT NULL,
    name TEXT NOT NULL
);

ALTER TABLE ONLY country
    ADD CONSTRAINT country_pkey PRIMARY KEY (id);


/*****************************************************/
/****************** City ****************************/
/*****************************************************/

CREATE TABLE city (
    id SERIAL NOT NULL,
    name TEXT NOT NULL,
    countryId INTEGER NOT NULL
);
```

```sql
ALTER TABLE ONLY city
    ADD CONSTRAINT city_pkey PRIMARY KEY (id);

ALTER TABLE ONLY city
    ADD CONSTRAINT city_country_id_fkey
    FOREIGN KEY (countryId)
    REFERENCES country(id) ON UPDATE CASCADE;

/*****************************************************/
/***************** User ******************************/
/*****************************************************/

CREATE TABLE mb_user (

    id SERIAL NOT NULL,
    username TEXT NOT NULL,
    password TEXT NOT NULL,
    name TEXT NOT NULL,
    bio TEXT,
    dateOfBirth DATE,
    deactivationDate DATE,
    warns INTEGER DEFAULT 0,
    location INTEGER,
    rating REAL
);

ALTER TABLE ONLY mb_user
    ADD CONSTRAINT mb_user_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY mb_user
    ADD CONSTRAINT mb_user_username_unique
    UNIQUE (username);

ALTER TABLE ONLY mb_user
    ADD CONSTRAINT mb_user_dateOfBirth_past
    CHECK (dateOfBirth < now());

ALTER TABLE ONLY mb_user
    ADD CONSTRAINT mb_user_location_fkey
    FOREIGN KEY (location)
    REFERENCES city(id) ON UPDATE CASCADE;

ALTER TABLE ONLY mb_user
    ADD CONSTRAINT mb_user_rating_domain
    CHECK ((rating <= 5.0) AND (rating >= 0.0));


/*****************************************************/
/***************** Admin *****************************/
/*****************************************************/

CREATE TABLE admin(

    id SERIAL NOT NULL,
    userId INTEGER
);

ALTER TABLE ONLY admin
    ADD CONSTRAINT admin_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY admin
    ADD CONSTRAINT user_id_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id) ON UPDATE CASCADE;


/*****************************************************/
/***************** Band ******************************/
/*****************************************************/


CREATE TABLE band (

    id SERIAL NOT NULL,
    name char(50) NOT NULL,
    creationDate DATE,
    ceaseDate DATE,
    location INTEGER
);

ALTER TABLE ONLY band
    ADD CONSTRAINT band_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY band
    ADD CONSTRAINT band_name_unique
    UNIQUE (name);

ALTER TABLE ONLY band
    ADD CONSTRAINT band_creation_past
    CHECK (creationDate < now());

ALTER TABLE ONLY band
    ADD CONSTRAINT band_location_fkey
    FOREIGN KEY (location)
    REFERENCES city(id) ON UPDATE CASCADE;

/*****************************************************/
/***************** Content ***************************/
/*****************************************************/

CREATE TABLE content (

    id SERIAL NOT NULL,
    text TEXT NOT NULL,
    date TIMESTAMP DEFAULT now()
);

ALTER TABLE ONLY content
```

```
    ADD CONSTRAINT content_pkey
    PRIMARY KEY (id);


/*******************************************************/
/***************** Post ******************************/
/*******************************************************/

CREATE TABLE post (

    id SERIAL NOT NULL,
    private BOOLEAN NOT NULL DEFAULT FALSE,
    contentId INTEGER NOT NULL,
    posterId INTEGER,
    bandId INTEGER
);

ALTER TABLE ONLY post
    ADD CONSTRAINT post_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY post
    ADD CONSTRAINT post_content_id_fkey
    FOREIGN KEY (contentId)
    REFERENCES content(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY post
    ADD CONSTRAINT poster_id_fkey
    FOREIGN KEY (posterId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY post
    ADD CONSTRAINT post_band_id_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;


/*******************************************************/
/***************** Message ***************************/
/*******************************************************/

CREATE TABLE message (

    id SERIAL NOT NULL,
    contentId INTEGER NOT NULL,
    senderId INTEGER,
    receiverId INTEGER,
    bandId INTEGER
);

ALTER TABLE ONLY message
    ADD CONSTRAINT message_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY message
    ADD CONSTRAINT message_content_id_fkey
    FOREIGN KEY (contentId)
    REFERENCES content(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY message
    ADD CONSTRAINT message_sender_id_fkey
    FOREIGN KEY (senderId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY message
    ADD CONSTRAINT message_receiver_id_fkey
    FOREIGN KEY (receiverId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY message
    ADD CONSTRAINT message_band_id_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;


/*******************************************************/
/***************** Comment ***************************/
/*******************************************************/

CREATE TABLE comment (

    id SERIAL NOT NULL,
    contentId INTEGER NOT NULL,
    commenterId INTEGER,
    postId INTEGER
);

ALTER TABLE ONLY comment
    ADD CONSTRAINT comment_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY comment
    ADD CONSTRAINT comment_content_id_fkey
    FOREIGN KEY (contentId)
    REFERENCES content(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY comment
    ADD CONSTRAINT commenter_id_fkey
    FOREIGN KEY (commenterId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;
```

```
ALTER TABLE ONLY comment
    ADD CONSTRAINT post_id_fkey
    FOREIGN KEY (postId)
    REFERENCES post(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;


/*****************************************************/
/***************** Genre ******************************/
/*****************************************************/

CREATE TABLE genre (

    id SERIAL NOT NULL,
    name TEXT NOT NULL,
    creatingAdminId INTEGER
);

ALTER TABLE ONLY genre
    ADD CONSTRAINT genre_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY genre
    ADD CONSTRAINT genre_name_unique
    UNIQUE (name);

ALTER TABLE ONLY genre
    ADD CONSTRAINT genre_creatingAdmin_id_fkey
    FOREIGN KEY (creatingAdminId)
    REFERENCES admin(id)
    ON UPDATE CASCADE
    ON DELETE SET NULL;

/*****************************************************/
/***************** Skill ******************************/
/*****************************************************/

CREATE TABLE skill (

    id SERIAL NOT NULL,
    name TEXT NOT NULL,
    creatingAdminId INTEGER
);

ALTER TABLE ONLY skill
    ADD CONSTRAINT skill_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY skill
    ADD CONSTRAINT skill_name_unique
    UNIQUE (name);

ALTER TABLE ONLY skill
    ADD CONSTRAINT skill_creatingAdmin_id_fkey
    FOREIGN KEY (creatingAdminId)
    REFERENCES admin(id)
    ON UPDATE CASCADE
    ON DELETE SET NULL;


/*******************************************************/
/****************** Report ****************************/
/*******************************************************/

CREATE TABLE report (

    id SERIAL NOT NULL,
    text TEXT NOT NULL,
    date TIMESTAMP DEFAULT now(),
    reportedContentId INTEGER,
    reportedUserId INTEGER,
    reportedBandId INTEGER,
    reporterUserId INTEGER
);

ALTER TABLE ONLY report
    ADD CONSTRAINT report_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY report
    ADD CONSTRAINT reported_content_id_fkey
    FOREIGN KEY (reportedContentId)
    REFERENCES content(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY report
    ADD CONSTRAINT reported_user_id_fkey
    FOREIGN KEY (reportedUserId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY report
    ADD CONSTRAINT reported_band_id_fkey
    FOREIGN KEY (reportedBandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY report
    ADD CONSTRAINT reporter_user_id_fkey
    FOREIGN KEY (reporterUserId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

/*******************************************************/
/******************** Ban ******************************/
/*******************************************************/

CREATE TABLE ban (
    id SERIAL NOT NULL,
    reason TEXT NOT NULL,
    banDate TIMESTAMP DEFAULT now(),
    ceaseDate TIMESTAMP,
    adminId INTEGER
);

ALTER TABLE ONLY ban
    ADD CONSTRAINT ban_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY ban
    ADD CONSTRAINT admin_id_fkey
    FOREIGN KEY (adminId)
    REFERENCES admin(id)
    ON UPDATE CASCADE
    ON DELETE SET NULL;


/*****************************************************/
/******************** User Skill ********************/
/*****************************************************/

CREATE TABLE user_skill (

    userId INTEGER NOT NULL,
    skillId INTEGER NOT NULL,
    level INTEGER NOT NULL

);

ALTER TABLE ONLY user_skill
    ADD CONSTRAINT user_skill_pkey
    PRIMARY KEY (userId,skillId);

ALTER TABLE ONLY user_skill
    ADD CONSTRAINT userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY user_skill
    ADD CONSTRAINT skillId_fkey
    FOREIGN KEY (skillId)
    REFERENCES skill(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY user_skill
    ADD CONSTRAINT user_skill_level_domain
    CHECK ((level <= 5) AND (level >= 1));


/*****************************************************/
/******************** User Follower *******************/
/*****************************************************/
```

```
CREATE TABLE user_follower (

    id SERIAL NOT NULL,
    followingUserId INTEGER NOT NULL,
    followedUserId INTEGER NOT NULL

);

ALTER TABLE ONLY user_follower
    ADD CONSTRAINT user_follower_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY user_follower
    ADD CONSTRAINT user_follower_unique_pair
    UNIQUE (followingUserId,followedUserId);

ALTER TABLE ONLY user_follower
    ADD CONSTRAINT followingUserId_fkey
    FOREIGN KEY (followingUserId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY user_follower
    ADD CONSTRAINT followedUserId_fkey
    FOREIGN KEY (followedUserId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

/*******************************************************/
/******************** User Rating *********************/
/*******************************************************/

CREATE TABLE user_rating (

    ratingUserid INTEGER NOT NULL,
    ratedUserId INTEGER NOT NULL,
    rate INTEGER NOT NULL

);

ALTER TABLE ONLY user_rating
    ADD CONSTRAINT user_rating_pkey
    PRIMARY KEY (ratingUserid,ratedUserId);

ALTER TABLE ONLY user_rating
    ADD CONSTRAINT user_rating_rating_userId_fkey
    FOREIGN KEY (ratingUserid)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY user_rating
    ADD CONSTRAINT user_rating_rated_userId_fkey
    FOREIGN KEY (ratedUserId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;
```

```sql
ALTER TABLE ONLY user_rating
    ADD CONSTRAINT user_rating_rate_domain
    CHECK ((rate <= 5) AND (rate >= 1));

/*****************************************************/
/******************* User Warning ********************/
/*****************************************************/

CREATE TABLE user_warning (

    id SERIAL NOT NULL,
    adminId INTEGER NOT NULL,
    userId INTEGER NOT NULL

);

ALTER TABLE ONLY user_warning
    ADD CONSTRAINT user_warning_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY user_warning
    ADD CONSTRAINT user_warning_adminId_fkey
    FOREIGN KEY (adminId)
    REFERENCES admin(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY user_warning
    ADD CONSTRAINT user_warning_userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;


/*****************************************************/
/******************* Band Genre **********************/
/*****************************************************/

CREATE TABLE band_genre (

    bandId INTEGER NOT NULL,
    genreId INTEGER NOT NULL
);

ALTER TABLE ONLY band_genre
    ADD CONSTRAINT band_genre_pkey
    PRIMARY KEY (bandId,genreId);

ALTER TABLE ONLY band_genre
    ADD CONSTRAINT band_genre_bandId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_genre
    ADD CONSTRAINT band_genre_genreId_fkey
    FOREIGN KEY (genreId)
    REFERENCES genre(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;


/*****************************************************/
/******************* Band Membership ****************/
/*****************************************************/

CREATE TABLE band_membership (

    bandId INTEGER NOT NULL,
    userId INTEGER NOT NULL,
    isOwner BOOLEAN NOT NULL,
    initialDate DATE,
    ceaseDate DATE
);

ALTER TABLE ONLY band_membership
    ADD CONSTRAINT band_membership_pkey
    PRIMARY KEY (bandId,userId);

ALTER TABLE ONLY band_membership
    ADD CONSTRAINT band_membership_bandId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_membership
    ADD CONSTRAINT band_membership_userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;


/*****************************************************/
/******************* User Rating ********************/
/*****************************************************/

CREATE TABLE band_rating (

    ratingUserid INTEGER NOT NULL,
    ratedBandId INTEGER NOT NULL,
    rate INTEGER NOT NULL

);

ALTER TABLE ONLY band_rating
    ADD CONSTRAINT band_rating_pkey
    PRIMARY KEY (ratingUserid,ratedBandId);

ALTER TABLE ONLY band_rating
    ADD CONSTRAINT band_rating_rating_userId_fkey
    FOREIGN KEY (ratingUserid) REFERENCES mb_user(id)
    ON UPDATE CASCADE;
```

```
ALTER TABLE ONLY band_rating
    ADD CONSTRAINT band_rating_rated_bandId_fkey
    FOREIGN KEY (ratedBandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_rating
    ADD CONSTRAINT band_rating_rate_domain
    CHECK ((rate <= 5) AND (rate >= 1));


/*****************************************************/
/******************* Band Warning ********************/
/*****************************************************/

CREATE TABLE band_warning (

    id SERIAL NOT NULL,
    adminId INTEGER NOT NULL,
    bandId INTEGER NOT NULL

);

ALTER TABLE ONLY band_warning
    ADD CONSTRAINT band_warning_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY band_warning
    ADD CONSTRAINT band_warning_adminId_fkey
    FOREIGN KEY (adminId)
    REFERENCES admin(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_warning
    ADD CONSTRAINT band_warning_userId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

/*****************************************************/
/******************* Band Follower *******************/
/*****************************************************/

CREATE TABLE band_follower (

    id SERIAL NOT NULL,
    userId INTEGER NOT NULL,
    bandId INTEGER NOT NULL

);

ALTER TABLE ONLY band_follower
    ADD CONSTRAINT band_follower_pkey
    PRIMARY KEY (id);
```

```
ALTER TABLE ONLY band_follower
    ADD CONSTRAINT band_follower_unique_pair
    UNIQUE (userId,bandId);

ALTER TABLE ONLY band_follower
    ADD CONSTRAINT followingUserId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_follower
    ADD CONSTRAINT followedBandId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

/******************************************************/
/******************* Band Application ****************/
/******************************************************/

CREATE TYPE BAND_APPLICATION_STATUS
AS ENUM ('canceled', 'pending', 'accepted', 'rejected');

CREATE TABLE band_application (

    id SERIAL NOT NULL,
    userId INTEGER NOT NULL,
    bandId INTEGER NOT NULL,
    date TIMESTAMP DEFAULT now(),
    lastStatusDate DATE,
    status BAND_APPLICATION_STATUS DEFAULT 'pending'

);

ALTER TABLE ONLY band_application
    ADD CONSTRAINT band_application_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY band_application
    ADD CONSTRAINT band_application_userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_application
    ADD CONSTRAINT band_application_bandId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

/******************************************************/
/******************* Band Invitation *****************/
/******************************************************/

CREATE TYPE BAND_INVITATION_STATUS
AS ENUM ('canceled', 'pending', 'accepted', 'rejected');
```

```
CREATE TABLE band_invitation(

    id SERIAL NOT NULL,
    userId INTEGER NOT NULL,
    bandId INTEGER NOT NULL,
    date TIMESTAMP DEFAULT now(),
    lastStatusDate DATE,
    status BAND_INVITATION_STATUS DEFAULT 'pending'

);

ALTER TABLE ONLY band_invitation
    ADD CONSTRAINT band_invitation_pkey
    PRIMARY KEY (id);

ALTER TABLE ONLY band_invitation
    ADD CONSTRAINT band_invitation_userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY band_invitation
    ADD CONSTRAINT band_invitation_bandId_fkey
    FOREIGN KEY (bandId)
    REFERENCES band(id)
    ON UPDATE CASCADE;

/*******************************************************/
/************ Notification Trigger *******************/
/*******************************************************/

CREATE TYPE NOTIFICATION_TYPE AS ENUM (
    'user_follower', 'band_follower', 'message', 'comment', 'band_application',
    'band_invitation', 'user_warning', 'band_warning',
    'band_invitation_accepted','band_invitation_rejected',
    'band_application_accepted', 'band_application_rejected');

CREATE TABLE notification_trigger (

    id SERIAL NOT NULL,
    date TIMESTAMP NOT NULL DEFAULT now(),
    type NOTIFICATION_TYPE,
    originUserFollower INTEGER,
    originBandFollower INTEGER,
    originMessage INTEGER,
    originComment INTEGER,
    originBandApplication INTEGER,
    originBandInvitation INTEGER,
    originUserWarning INTEGER,
    originBandWarning INTEGER
);

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_pkey
    PRIMARY KEY (id);
```

```sql
ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_user_follower_fkey
    FOREIGN KEY (originUserFollower)
    REFERENCES user_follower(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_band_follower_fkey
    FOREIGN KEY (originBandFollower)
    REFERENCES band_follower(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_message_fkey
    FOREIGN KEY (originMessage)
    REFERENCES message(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_comment_fkey
    FOREIGN KEY (originComment)
    REFERENCES comment(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_band_application_fkey
    FOREIGN KEY (originBandApplication)
    REFERENCES band_application(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_band_invitation_fkey
    FOREIGN KEY (originBandInvitation)
    REFERENCES band_invitation(id)
    ON UPDATE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_user_warning_fkey
    FOREIGN KEY (originUserWarning)
    REFERENCES user_warning(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY notification_trigger
    ADD CONSTRAINT notification_trigger_origin_band_warning_fkey
    FOREIGN KEY (originBandWarning)
    REFERENCES band_warning(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

/*****************************************************/
```

```
/**************** User Notification ********************/
/*******************************************************/

CREATE TABLE user_notification (

    notification_trigger_id INTEGER NOT NULL,
    userId INTEGER NOT NULL
);

ALTER TABLE ONLY user_notification
    ADD CONSTRAINT user_notification_pkey
    PRIMARY KEY (notification_trigger_id, userId);

ALTER TABLE ONLY user_notification
    ADD CONSTRAINT user_notification_notification_trigger_fkey
    FOREIGN KEY (notification_trigger_id)
    REFERENCES notification_trigger(id)
    ON UPDATE CASCADE
    ON DELETE CASCADE;

ALTER TABLE ONLY user_notification
    ADD CONSTRAINT user_notification_userId_fkey
    FOREIGN KEY (userId)
    REFERENCES mb_user(id)
    ON UPDATE CASCADE;
```

Revision history

GROUP1712, 12/03/2018

> João Pinheiro, up201104913@fe.up.pt

> Leonardo Teixeira, up201502848@fe.up.pt

> Danny Soares, up201505509@fe.up.pt

> João Azevedo, up201503256@fe.up.pt