FavTap

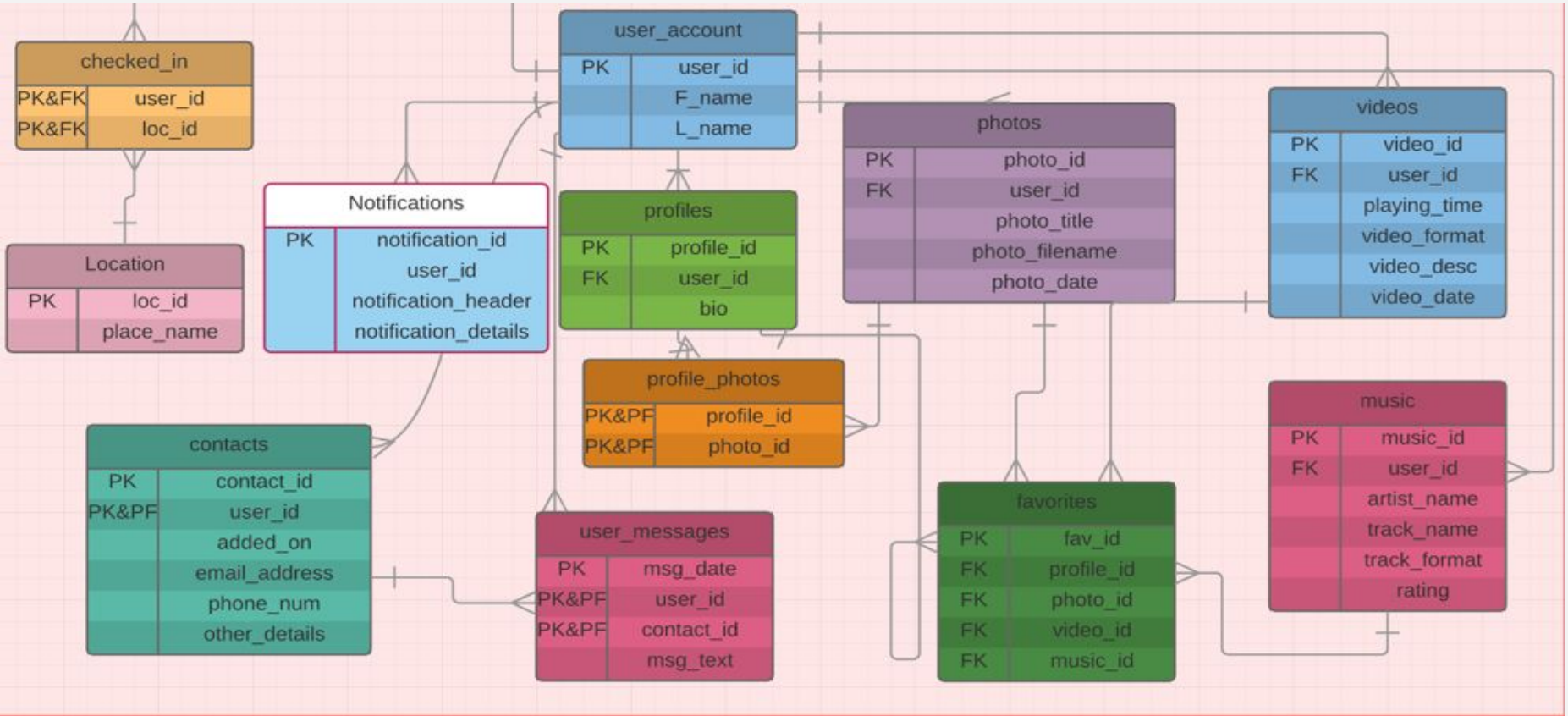
“A social media for all your favorite media”

Database Design Proposal
By: Kwame T . Darko

Executive Summary

TapFav will be a web based social media with the simple concept of simply listing your favorite media such as videos, music, and pictures. This document presents a general design and implementation of TapFav database and outlines the structure and tables involved in the design of the database system. The purpose of this database is to store all different user data and turn it into information which will then be used by the website to display certain things. The design has been tested and implemented on PostGres to provide a fully functional database that is in Boyce Codd third normal form.

E/R Diagram


checked_in
PK&FK user_id
PK&FK loc_id
Location
PK loc_id
place_name
Notifications
PK notification_id
user_id
notification_header
notification_details
user_account
PK user_id
F_name
L_name
profiles
PK profile_id
FK user_id
bio
profile_photos
PK&PF profile_id
PK&PF photo_id
photos
PK photo_id
FK user_id
photo_title
photo_filename
photo_date
videos
PK video_id
FK user_id
playing_time
video_format
video_desc
video_date
music
PK music_id
FK user_id
artist_name
track_name
track_format
rating
contacts
PK contact_id
PK&PF user_id
added_on
email_address
phone_num
other_details
user_messages
PK msg_date
PK&PF user_id
PK&PF contact_id
msg_text
favorites
PK fav_id
FK profile_id
FK photo_id
FK video_id
FK music_id

User_account Table

SQL

```
CREATE TABLE IF NOT EXISTS
user_account(
      user_id SERIAL NOT NULL UNIQUE,
      F_name VARCHAR(50) NOT NULL,
      L_name VARCHAR(50) NOT NULL,
      PRIMARY KEY (user_id)
);
```

Functional Dependencies

User_id -> F_name, L_name

Sample Data

	user_id integer	f_name character varying(50)	l_name character varying(50)
1	1	Kwame	Darko
2	2	Alan	Labouseur
3	3	Dan	Njoku
4	4	Ed	Achziger
5	5	John	Brennan
6	6	Devin	Cividni
7	7	Juston	Christian
8	8	Nicholas	Foster
9	9	Melvin	Forkpa
10	10	John	Khanda

Profiles Table

SQL

```
CREATE TABLE IF NOT EXISTS Profiles(
      Profile_id SERIAL NOT NULL UNIQUE,
      User_id INTEGER NOT NULL,
      Bio VARCHAR(200),
      PRIMARY KEY (profile_id),
      FOREIGN KEY (user_id) REFERENCES
user_account(user_id)
);
```

Functional Dependencies

Profile_id -> User_id, Bio

Sample Data Preview

profile_id integer	user_id integer	bio character varying(200)
1	1	test test test tesr
2	2	best site ever
3	3	new to the site
4	4	hello world
5	5	well hello there
6	6	welcome to my profile
7	7	hellllooooooooo
8	8	ok ok ok
9	9	hmmmmmm
10	10	kskskskskkskskskks

Contacts Table

SQL

```
CREATE TABLE IF NOT EXISTS contacts(
      Contact_id SERIAL NOT NULL UNIQUE,
      User_id INTEGER NOT NULL,
      Added_on DATE NOT NULL,
      Email_address VARCHAR(30) NOT NULL,
      Phone_num VARCHAR(10),
      Other_details VARCHAR(100),
      PRIMARY KEY(contact_id),
      FOREIGN KEY (user_id) REFERENCES
user_account(user_id)
);
```

Functional dependencies:

Contact_id > user_id, date_contact_from, email_address, web_site, contact_name, work_phone, cell_mobile_phone, other details

Sample Data Preview

contact_id integer	user_id integer	added_on date	email_address character varying(30)	phone_num character varying(10)	other_details character varying(100)
1	1	2016-04-11	tenktenk@gmail.com	3478225555	he is very nice
2	2	2016-04-11	alanlab@gmail.com	2025550141	cannot give up, ever
3	3	2016-04-11	dannjoku@gmail.com	2025550143	likes skirts that twirl well
4	4	2016-04-11	edachzi@gmail.com	2025550145	feels like no shoes they ever wear are
5	5	2016-04-11	johnbrennan@gmail.com	2023550141	is convinced astrology is the scientifi
6	6	2016-04-11	devinviid@gmail.com	2025550949	paints their fingernails a different co
7	7	2016-04-11	justonchris@gmail.com	2025552222	cannot end a conversation without insul
8	8	2016-04-11	nicholasfos@gmail.com	2025550666	treats all of their friends like they a
9	9	2016-04-11	melvinfork@gmail.com	9999999999	wears either mismatched socks or shoela
10	10	2016-04-11	johnkhanda@gmail.com	9849299949	eats colors.

User_messages Table

SQL

```
CREATE TABLE IF NOT EXISTS user_messages(
      User_id INTEGER NOT NULL,
      Contact_id INTEGER NOT NULL,
      msg_date DATE NOT NULL,
      Msg_text VARCHAR(500) NOT NULL,
PRIMARY KEY(user_id, contact_id, msg_date)
FOREIGN KEY user_id REFERENCES user_account
(user_id)
);
```

Functional Dependencies

(user_id, contact_id, date_time_of_msg) > msg_text

Sample Data Preview

user_id integer	contact_id integer	msg_date date	msg_text character varying(500)
1	1	2016-04-11	hello kwame world
2	4	2016-04-11	soo.....
3	5	2016-04-11	im going to go to class now
7	5	2016-04-11	ok...
2	3	2016-04-11	What my grade looking like
5	1	2016-04-11	HMU when you are free
4	1	2016-04-11	you never wrote me back bro
8	3	2016-04-11	Whats up BRO!!
9	3	2016-04-11	How is everything
10	2	2016-04-11	Are you ok?

Location Table

SQL

```
CREATE TABLE IF NOT EXISTS Location(
        loc_id SERIAL NOT NULL UNIQUE,
        place_name VARCHAR(50) NOT NULL,
        PRIMARY KEY (loc_id)
);
```

Functional dependencies:

(loc_id) > place_name

Sample Data Preview

loc_id integer	place_name character varying(50)
1	Marist
2	marist
3	vassar
4	Yale
5	Rutger

Checked_in

SQL

```
CREATE TABLE checked_in(
     user_id INTEGER NOT NULL,
     loc_id INTEGER NOT NULL,
     PRIMARY KEY(user_id, loc_id),
     FOREIGN KEY (user_id)
REFERENCES user_account(user_id),
     FOREIGN KEY (loc_id) REFERENCES
location(loc_id)
);
```

Functional dependencies:

(user_id, loc_id) > user_id, loc_id

Sample Data Preview

user_id integer	loc_id integer
1	1
5	1
4	2
6	2
3	1
6	4
8	3
9	4
2	5
3	4

Videos Table

SQL

```
CREATE TABLE IF NOT EXISTS  Videos(
        Video_id SERIAL NOT NULL UNIQUE,
        User_id INTEGER NOT NULL,
        Playing_time INTEGER NOT NULL,
        Video_format VARCHAR(4) NOT NULL,
        Video_desc VARCHAR(100),
        Video_date DATE,
        PRIMARY KEY(video_id),
        FOREIGN KEY (user_id) REFERENCES user_account(user_id)
);
```

Functional dependencies

Video_id > user_id, playing_time, video_format, Video_desc, Video_date

Sample Data Preview

video_id integer	user_id integer	playing_time integer	video_format character varying(4)	video_desc character varying(100)	video_date date
1	1	120	mp4	my beautiful video	2016-04-11
2	2	10	mp4	hitting a ball	2016-04-11
3	4	100	mp4	hitting a ball	2016-04-05
4	6	104	mp4	riding my bike	2016-04-06
5	2	510	mp4	me coding	2016-04-11
6	6	210	mp4	counting money	2016-02-11
7	7	40	mp4	walking to the gym	2016-03-11
8	8	60	mp4	going to class	2016-04-04
9	3	180	mp4	my first go pro	2016-04-05
10	9	104	mp4	talking to my friend	2016-04-03

Music Table

SQL

```
CREATE TABLE IF NOT EXISTS Music(
	Music_id SERIAL NOT NULL UNIQUE,
	User_id INTEGER NOT NULL,
	Artist_name VARCHAR(50) NOT NULL,
	Track_name VARCHAR(100) NOT NULL,
	Track_format VARCHAR(4) NOT NULL,
	rating INTEGER,
	PRIMARY KEY (music_id),
	FOREIGN KEY (user_id)
	REFERENCES user_account(user_id)
);
```

Sample Data Preview

music_id integer	user_id integer	artist_name character varying(50)	track_name character varying(100)	track_format character varying(4)	rating integer
1	1	kwamedarko	kwamemixtape	mp3	4
2	2	alanlab	alanmixtape	mp3	3
3	3	50cent	getrich	mp3	5
4	4	tyga	tygaworld	mp3	4
5	5	fattrel	gleesh	mp3	5
6	6	brysontiller	trapsoul	mp3	5
7	7	whoknows	getit	mp3	1
8	8	jondoe	luvsic	mp3	2
9	9	jiggaa	hmmmm	mp3	4
10	10	michaeljackson	savetheworld	mp3	5

Functional dependencies

Music_id > user_id, artist_name, track_name, track_format, rating

Photos Table

SQL

```
CREATE TABLE IF NOT EXISTS Photos(
      Photo_id SERIAL NOT NULL UNIQUE,
      User_id INTEGER NOT NULL,
      Photo_title VARCHAR(50) NOT NULL,
      Photo_filename VARCHAR(100) NOT NULL,
      Photo_date DATE NOT NULL,
      PRIMARY KEY(Photo_id),
      FOREIGN KEY (user_id) REFERENCES
user_account(user_id)
);
```

Functional dependencies

Photo_id > user_id, photo_title, photo_filename, photo_date

Sample Data Preview

photo_id integer	user_id integer	photo_title character varying(50)	photo_filename character varying(100)	photo_date date
1	1	My beautiful photo	jasjajajajaja.jpg	2016-04-11
2	1	My beautiful photo	jasjajajajaja.jpg	2016-04-11
3	4	myface	myface.jpg	2016-04-11
4	4	pictureofmyfood	pictureofmygood.png	2016-04-11
5	7	getmoneymoney	getmoneymoney.jpg	2016-04-11
6	5	My photo	3432423423423.gif	2016-04-11
7	2	Sin city	11100494888.png	2016-04-11
8	6	Out having fun	4424234234.jpg	2016-04-11
9	9	My cousins	2342432432423.png	2016-04-11
10	10	The hill	4442244235.jpg	2016-04-11

Profile_photos Table

SQL

```
CREATE TABLE Profile_photos(
      Profile_id INTEGER NOT NULL,
      Photo_id INTEGER NOT NULL,
      PRIMARY KEY(profile_id, photo_id),
FOREIGN KEY (profile_id) REFERENCES profiles
(profile_id),
FOREIGN KEY (photo_id) REFERENCES photos
(photo_id)
);
```

Functional dependencies

(Profile_id, photo_id) > profile_id, photo_id

Sample Data Preview

profile_id integer	photo_id integer
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

Favorites Table

SQL

```
CREATE TABLE IF NOT EXISTS favorites(
      Fav_id SERIAL NOT NULL UNIQUE,
      profile_id INTEGER NOT NULL,
      Photo_id INTEGER,
      Video_id INTEGER,
      Music_id INTEGER,
      PRIMARY KEY (fav_id),
      FOREIGN KEY (profile_id) REFERENCES profiles
(profile_id),
      FOREIGN KEY (photo_id) REFERENCES photos
(photo_id),
      FOREIGN KEY(video_id) REFERENCES videos
(video_id),
      FOREIGN KEY (music_id) REFERENCES music
(music_id)
);
```

Functional dependencies

(fav_id) > profile_id, user_id, photo_id, video_id, music_id

Sample Data Preview

fav_id integer	profile_id integer	photo_id integer	video_id integer	music_id integer
11	2	1	1	1
12	3	1	1	1
13	4	1	1	1
14	5	1	1	1
15	2	1	1	1
16	2	1	1	1
17	2	1	1	1
18	2	1	1	1
19	10	1	1	1

Notifications Table

SQL

```
CREATE TABLE IF NOT EXISTS Notifications(
      Notification_id SERIAL NOT NULL UNIQUE,
      User_id INTEGER NOT NULL,
      Notification_header VARCHAR(20) NOT NULL,
      Notification_details VARCHAR(200) NOT NULL,
      PRIMARY KEY (Notification_id),
      FOREIGN KEY (User_id) REFERENCES
user_account(user_id)
);
```

Functional Dependencies

```
Notification_id -> user_id , notification_header,
notofication_details
```

Sample Data Preview

	notification_id integer	user_id integer	notification_header character varying(20)	notification_details character varying(200)
1	11	1	Music favorited	user two has favorited Palm Trees
2	12	1	Music favorited	user one has favorited Palm Trees

VIEW: profileMusicFavorites

```
--Show all user music favorites

CREATE VIEW profileMusicFavorites AS

SELECT DISTINCT u.user_id, u.f_name, u.l_name, m.track_name

FROM user_account u, music m, profiles pr, favorites f

WHERE pr.user_id = u.user_id

AND pr.profile_id = f.profile_id

AND f.music_id = m.music_id

ORDER BY u.user_id, u.f_name, u.l_name;
```

Sample Data Preview

user_id integer	f_name character varying(50)	l_name character varying(50)	track_name character varying(100)
2	Alan	Labouseur	kwamemixtape
3	Dan	Njoku	kwamemixtape
4	Ed	Achziger	kwamemixtape
5	John	Brennan	kwamemixtape
10	John	Khanda	kwamemixtape

VIEW: userCheckins

```
--Show all user checkins

CREATE VIEW userCheckins AS

SELECT u.user_id, u.f_name, u.l_name, l.
place_name

FROM user_account u, checked_in c, location

WHERE u.user_id = c.user_id

AND l.loc_id = c.loc_id

ORDER BY u.user_id;
```

Sample Data Preview

user_id integer	f_name character varying(50)	l_name character varying(50)	place_name character varying(50)
1	Kwame	Darko	Marist
2	Alan	Labouseur	Rutger
3	Dan	Njoku	Yale
3	Dan	Njoku	Marist
4	Ed	Achziger	marist
5	John	Brennan	Marist
6	Devin	Cividni	Yale
6	Devin	Cividni	marist
8	Nicholas	Foster	vassar
9	Melvin	Forkpa	Yale

VIEW: profilePhotoFavorites

```
--Show all user photo favorites

CREATE VIEW profilePhotoFavorites AS

SELECT DISTINCT u.user_id, u.f_name, u.l_name, p.photo_title

FROM user_account u, photos p, profiles pr, favorites f

WHERE pr.user_id = u.user_id

AND pr.profile_id = f.profile_id

AND f.photo_id = p.photo_id

ORDER BY u.user_id, u.f_name, u.l_name;
```

Sample Data Previews

user_id integer	f_name character varying(50)	l_name character varying(50)	photo_title character varying(50)
2	Alan	Labouseur	My beautiful photo
3	Dan	Njoku	My beautiful photo
3	Dan	Njoku	myface
4	Ed	Achziger	My beautiful photo
5	John	Brennan	My beautiful photo
10	John	Khanda	My beautiful photo

VIEW: profileVideoFavorites

```
--Show all user video favorites

CREATE VIEW profileVideoFavorites AS

SELECT u.user_id, u.f_name, u.l_name, v.video_desc

FROM user_account u, videos v, profiles pr, favorites f

WHERE pr.user_id = u.user_id

AND pr.profile_id = f.profile_id

AND f.video_id = v.video_id

ORDER BY u.user_id, u.f_name, u.l_name;
```

Sample Data Preview

user_id integer	f_name character varying(50)	l_name character varying(50)	video_desc character varying(100)
2	Alan	Labouseur	my beautiful video
3	Dan	Njoku	hitting a ball
3	Dan	Njoku	my beautiful video
4	Ed	Achziger	my beautiful video
5	John	Brennan	my beautiful video
10	John	Khanda	my beautiful video

Stored Procedures

The three stored procedures don’t return any output, but just insert a record into the notification table. The videoNotification function inserts a video notification. The musicNotification function inserts a music notification. Lastly the photoNotification inserts a photo notification. The last trigger function checks for which type of favorite it was, then calls the appropriate notification function from the other 3.

Stored Procedure: videoNotification

```
--Takes video id and profile id as parameters, and adds the corresponding information to the notifications table

CREATE OR REPLACE FUNCTION videoNotification(vid int, pid int)

RETURNS void AS $$

DECLARE

Usr_id INTEGER;

FavUser_name TEXT;

Fav_name VARCHAR(100);

Fav_type VARCHAR(20);

BEGIN

Usr_id := (

SELECT user_id

FROM videos

WHERE video_id = vid);

Fav_name := (

SELECT video_desc

FROM videos

WHERE video_id = vid);

Fav_type := 'Video';
```

Videonotification continued

```
FavUser_name := (

SELECT (F_name || ' ' || L_name) AS
full_name

FROM user_account

WHERE user_id = (SELECT user_id

             FROM profiles

             WHERE profile_id = pid)

);
```

```
INSERT INTO notifications(user_id,
notification_header,
notification_details)

VALUES (usr_id, fav_type || '
favorited', favUser_name || ' has
favorited ' || fav_name);

END;

$$ LANGUAGE plpgsql;
```

Stored Procedure: musicNotification

```
--Takes music id and profile id as parameters, and adds the corresponding information to the notifications table

CREATE OR REPLACE FUNCTION musicNotification(mid int, pid int)

RETURNS void AS $$

DECLARE

Usr_id INTEGER;

FavUser_name TEXT;

Fav_name VARCHAR(100);

Fav_type VARCHAR(20);

BEGIN

Usr_id := (

SELECT user_id

FROM music

WHERE music_id = mid);

Fav_name := (

SELECT track_name

FROM music

WHERE music_id = mid);

Fav_type := 'Music';
```

MusicNotification continued

```
FavUser_name := (

SELECT (F_name || ' ' || L_name) AS
full_name

FROM user_account

WHERE user_id = (SELECT user_id

             FROM profiles

             WHERE profile_id = pid)

);
```

```
INSERT INTO notifications(user_id,
notification_header,
notification_details)

VALUES (usr_id, fav_type || '
favorited', favUser_name || ' has
favorited ' || fav_name);

END;

$$ LANGUAGE plpgsql;
```

STORED PROCEDURE : PHOTONOTIFICATION

```
--Takes photo id and profile id as parameters, and adds the corresponding information to the notifications table

CREATE OR REPLACE FUNCTION photoNotification(phid int, pid int)

RETURNS void AS $$

DECLARE

Usr_id INTEGER;

FavUser_name TEXT;

Fav_name VARCHAR(100);

Fav_type VARCHAR(20);

BEGIN

Usr_id := (

SELECT user_id

FROM photos

WHERE photo_id = phid);

Fav_name := (

SELECT photo_title

FROM photos

WHERE photo_id = phid);

Fav_type := 'Photo';
```

photoNotification continued

```
FavUser_name := (

SELECT (F_name || ' ' || L_name) AS
full_name

FROM user_account

WHERE user_id = (SELECT user_id

            FROM profiles

            WHERE profile_id = pid)

);
```

```
INSERT INTO notifications(user_id,
notification_header,
notification_details)

VALUES (usr_id, fav_type || '
favorited', favUser_name || ' has
favorited ' || fav_name);

END;

$$ LANGUAGE plpgsql;
```

Trigger Function

The function checks for the type of media that has been favorited when a new record is inserted or updated in the favorites table. It than calls the appropriate media function to send a notification to the user. (Record inserted into notification table)

Trigger function

```
CREATE OR REPLACE FUNCTION notify()

RETURNS trigger AS $$

BEGIN

IF NEW.photo_id IS NULL

AND NEW.video_id IS NULL THEN

EXECUTE musicNotification(NEW.
music_id, NEW.profile_id);

ELSIF NEW.music_id IS NULL

AND NEW.video_id IS NULL THEN

EXECUTE photoNotification(NEW.photo_id, NEW.
profile_id);

ELSIF NEW.photo_id IS NULL

AND NEW.music_id IS NULL THEN

EXECUTE videoNotification(NEW.video_id, NEW.
profile_id);

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER notifcation AFTER
INSERT OR UPDATE ON favorites FOR
EACH ROW EXECUTE PROCEDURE
notify();
```

Security

The purpose of this is to define the user roles and grant privileges to certain users. There is an admin user and a regular user. The admin user has full access to the database while the regular user is restricted. For example the admin user has full access to insert, update,or delete all tables while the regular can only access tables that have data related to the user.

Security

```
--admin user

CREATE ROLE admin;

GRANT ALL ON ALL TABLES

IN SCHEMA PUBLIC

TO admin;
```

```
--social network user, cannot insert new users
or profiles, but can access other tables

CREATE ROLE user;

GRANT SELECT, INSERT, UPDATE ON
location, user_messages,
profile_photos, favorites, photos,
videos, music, contacts, checked_in

TO user;

GRANT SELECT, UPDATE ON users,
profiles

TO user;
```

Implementation Notes

Favorites

- Made to list all the favorite media a user has favorited. A user can have many

CheckedÖin

- When a user access a location that is recognized by the database a list of known place names will get shown to the user

Profiles

- A user account can have multiple profiles but can only log into a single one at a time. For example a user can have a profile where they share specifically just photos and videos and another profile for music. Alsostores the bio.

Known Problems

- A user media is accessed publicly. There is no restriction to who can see their content
- There is no limit to the number of notifications sent to the user

Future Enhancements

- **Bumped Into:** Send a notification to the user if they are currently checked into the same location as one of their contacts

- Add a user rating to a user profile. A user can rate a user profile based on their favorite content.