This document is about the process of going from the ER diagram to the database tables.

ENTITIES:

Global Variables

```
create table global_variables
(
    number_posts           integer,
    number_posts_per_day   integer,
    number_posts_per_week  integer,
    number_posts_per_month integer,

    constraint fk_users foreign key (user_id) references users
);
```

In this entity of global variables we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The user_id, as a primary key for the table, needs to be an integer value that does not contain a null value or is empty. The number_posts, number_posts_per_day, number_posts_per_week and number_post_per_month will be integer value that save the statistics of post posted per time periods.

. . .

Users

```
create table users
(
    user_id           serial            not null,
    user_name         varchar(15)       not null,
    user_lastname     varchar(25)       not null,
    user_username     varchar(30)       not null,
    user_email        varchar(30)       not null,
    user_password     varchar(40)       not null,
    user_follows      integer default 0 not null,
    user_followers    integer default 0 not null,
    user_notification integer default 0 not null,
    user_block        integer default 0 not null,
    user_posts        integer default 0 not null,

    constraint users_pkey primary key (user_id),
    constraint username unique (user_username),
    constraint email unique (user_email)
);
```

In this entity of users we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

For the primary key, user_id, the value is set as serial so the next values are sequential and null so its value cannot be empty, meaning that each user will take up a unique key when an account is created. The user_name is set to be variable string of length of at least 15, the user_lastname has been set to contain no more than a string length of 25, the user_username has a string length set to at least 30, user_email has a string length set to at least 30, the user_password has a string length set to at least 40, all these defined to be not null. For the next attributes (user_follows, user_followers, user_notification, user_block, user_posts) their data types have been specified to be integer values starting at cero where none of them can contain null or empty values. The user table constraints follow that user_pk (user primary key) is made from the user_id but the user_id is not the primary key of this table, also the username and email are required to be unique values for each user account.

...

Notifications

```
create table notification
(
    notification_id serial  not null,
    user_id         integer not null,

    constraint notification_pkey primary key (notification_id),
    constraint fk_users foreign key (user_id) references users
);
```

In this entity of notifications we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The notification_id is set to be serial valued and the user_id an integer, both cannot be empty or null values. The constraints for the table are that the notification_pkey (notification primary key) is taken from the notification_id attribute. The user_id, referenced from the users table/entity, will be used as a foreign key named fk_users for the notifications table. Each notification will be set to a user account.

...

Message

```
create table message
(
    message_id      serial            not null,
    user_id         integer           not null,
    message_likes   integer default 0 not null,
    message_reads   integer default 0 not null,
    message_shares  integer default 0 not null,
    message_replies integer default 0 not null,
    message_content varchar(60)       not null,
    is_reply        integer default 0 not null,

    constraint message_pkey primary key (message_id),
    constraint users foreign key (user_id) references users on delete
cascade,
    constraint message foreign key (is_reply) references message on delete
cascade
);
```

In this entity message we defined the attributes and their data types as it is showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

In this table the message_id value will be used as the message_pkey (message primary key) of the table, the user_id value referenced from the users table will be used as the foreign key which will deleted any replies connected if the user_id attribute is deleted.
Here the attribute is_reply reference to the PK of the user that made the reply.

. . .

RELATIONSHIPS

Follow

```
create table follow
(
    fid      integer default nextval('follow_fid_seq'::regclass) not null,
    follower integer                                             not null,
    followed integer                                             not null,

    constraint follow_pkey primary key (fid),
    constraint follower foreign key (follower) references users on delete
cascade,
    constraint followed foreign key (followed) references users on delete
```

```
cascade
);
```

In this table follow we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The fid will change sequentially from the specified value each time a new follow event has occurred, the follower and followed will be integer values containing the total number of each per user account. This table will use the value of fid for follow_pkey as the primary key and the values of follower and followed will be referenced from the relationship with the users entity, each time the parent (users) is deleted the same will follow with the childs (follower and followed). For example, if a user account is deleted then the users this account used to follow will automatically remove the deleted account from the follower or followed list.

...

Block

```
create table block
(
    bid     integer default nextval('block_bid_seq'::regclass) not null,
    blocker integer,
    blocked integer,

    constraint block_pkey primary key (bid),
    constraint blocker foreign key (blocker) references users on delete
cascade,
    constraint blocked foreign key (blocked) references users on delete
cascade
);
```

In this table we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The bid will change sequentially from the specified value each time a new block event has occurred, the blocker and blocked will be integer values containing the total number of each per user account. This table will use the value of bid for block_pkey as the primary key and the values of blocker and blocked will be referenced from the relationship with the users entity, each time the parent (users) is deleted the same will follow with the childs (blocker and blocked).

...

Likes

```
create table likes
(
    like_id    serial  not null,
    message_id integer not null,
    liker_id   integer not null,


    constraint likes_pkey primary key (like_id),
    constraint fk_message foreign key (message_id) references message,
    constraint likes_liker_id_fkey foreign key (liker_id) references users
);
```

In this table likes we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The like_id serial value will be used for the likes_pkey primary key for the table. The likes_id value will also be used as a foreign key (likes_liker_id_fkey) to reference the user table, where each message will be linked to the user account that made the like event on the message. The message_id referenced from the message table will be used as a foreign key to link each like with the message.

...

Share

```
create table share
(
    sid        integer default nextval('share_sid_seq'::regclass) not null,
    user_id    integer                                            not null,
    message_id integer                                            not null,


    constraint share_pkey primary key (sid),
    constraint "user" foreign key (user_id) references users on delete
cascade,
    constraint message foreign key (message_id) references message on delete
cascade
);
```

In this table share we defined the attributes and their data types as it's showed in the schema above, specifying the

constraints on the certain attributes at the end of the declaration statement.

The sid serial value will be used for the share_pkey primary key for the table. The message_id value will also be used as a foreign key to reference the message table, where each message will be linked to the user account that made the share event on the message. The user_id will be used also as a foreign key user that references the users entity in which if for example, a share event is deleted if the user account linked by the user_id is deleted.

...

Unlike

```
create table unlikes
(
    unlike_id  serial  not null,
    message_id integer not null,
    unliker_id integer not null,


    constraint unlikes_pkey primary key (unlike_id),
    constraint fk_message foreign key (message_id) references message,
    constraint unlikes_unliker_id_fkey foreign key (unliker_id) references
users
);
```

In this entity we defined the attributes and their data types as it's showed in the schema above, specifying the constraints on the certain attributes at the end of the declaration statement.

The unlike_id serial value will be used for the primary key unlikes_pkey of the table, the unliker_id value which references the users entity table will be used as a foreign key named unlikes_unliker_id_fkey which will link each user account to the unlike event. Also as a foreign key the message_id value will be used as the fk_message for the table which links the message entity table with each unlike event.

...