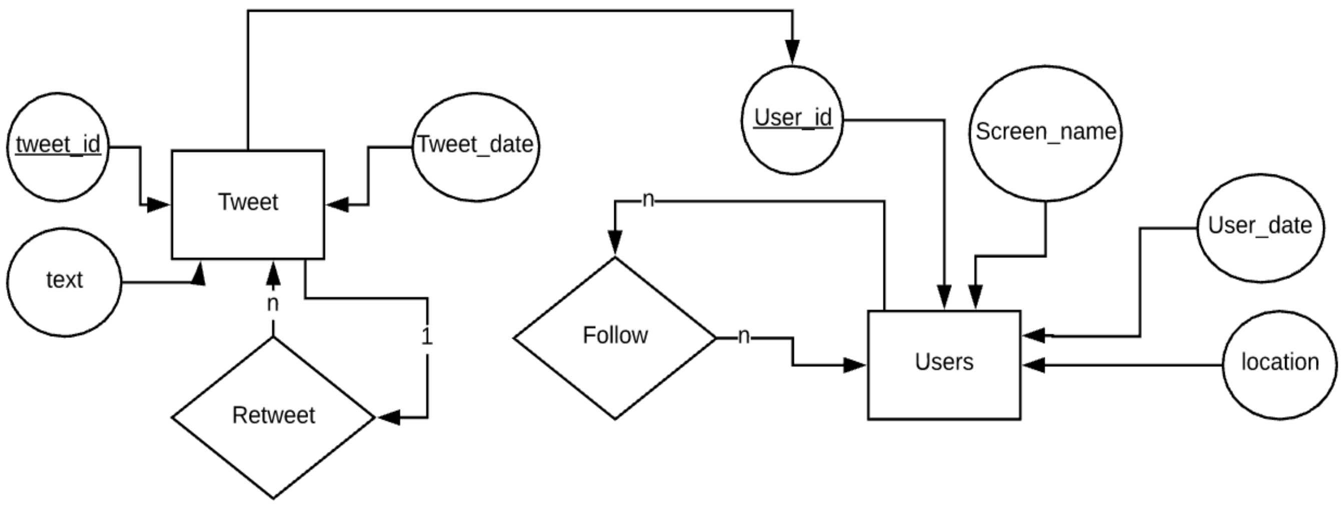
Andrés Mena Godino y Juan Moreno Díez. Group 1291

Lab report:

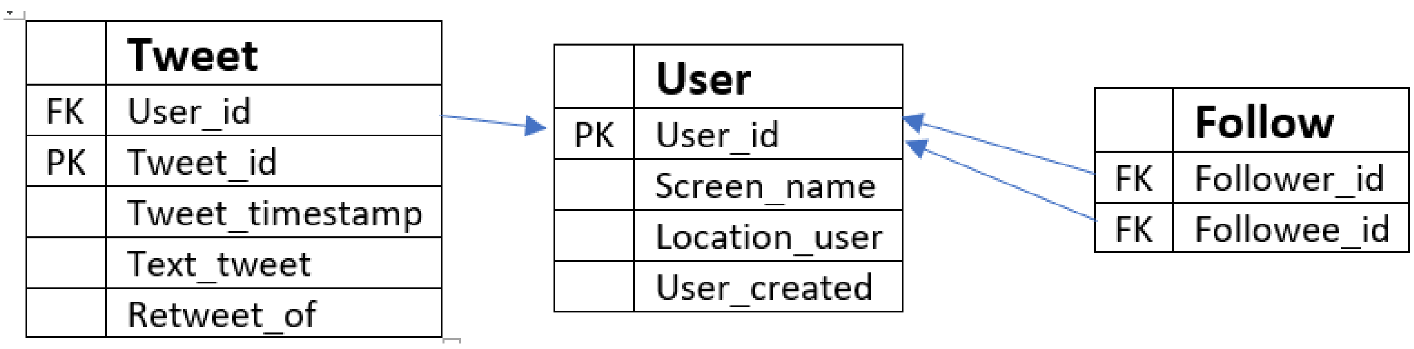
Relational design and SQL

Andrés Mena Godino y Juan Moreno Díez. Group 1291

Excercise 1: ER diagram



Excercise 2: Relational Design



	Tweet
FK	User_id
PK	Tweet_id
	Tweet_timestamp
	Text_tweet
	Retweet_of

	User
PK	User_id
	Screen_name
	Location_user
	User_created

	Follow
FK	Follower_id
FK	Followee_id

Andrés Mena Godino y Juan Moreno Díez. Group 1291

EXERCISE 4.

In p1_tweet, create a collection of tables that contain, with little or no redundancy, the information about tweets, users, and retweets. You will have to decide what tables to create, what columns will they have, and what types of SQL data are the best choice for each attribute.

- What attributes are good primary/foreign keys?
  For the table user, the optimal primary key is the user_id, as it is unique for each user. In the table tweets, the primary key is tweet_id, for the same reason as before, it is an unique identifier for the tweet. As a foreign key in the tweet table, we used user_id, so that we are able to relate the user with the information they tweeted. In the follow table, we used as foreign keys both attributes (follower_id and followee_id), relating them with the user_id (user table). These are good so that we do not need to include all the information about the user in the follow table, and it is enough with a "link" to the user table.

- What tables must you crate to store the given information without redundancy?
  In order to avoid redundancy, we first need to create two auxiliary tables, in which we will dump the data from the files. Then, we copy the information we need and consider necessary without being redundant, into the tables (users, tweets, follow) we had previously created.

In order to create this database correctly, we had to take various programming decisions. Firstly, we decided to make two big tables (temporary) in which we would store all the information from the files. In this temporary tables, we would simply use very big varchars (as we would get rid of those tables). Once we had all the information in the database, we created some tables to correctly store the information. We had to do this taking into account possible redundancies in the files and used the function distinct in order to do so. While introducing the information from the auxiliary table to the final tables, we had to do a casting for some of the data, as we had store everything as varchars; ids needed to be transformed to integers, information about dates needed to be transformed to timestamp (or use the function date_to for the toy database as it was a custom date type). For the big database, we needed to change the integers for bigints , so that all the data (which was now bigger) could fit in the table. The queries we came up with, focused on making sure that the connection between users and their tweets where correct (by using natural join), as well as assuring that the empty spaces (retweet of) had been correctly created.

Query 1: select screen_name, tweet_timestamp, text_tweet from tweet natural join user1 where tweet_timestamp >= '2018-01-01' and tweet_timestamp <= '2018-12-31'

- This first query, took 13 msec in the small db vs 5:23 minutes in the big one (which retrieved 1666770 rows)

Query 2: select screen_name, text_tweet from tweet natural join user1 where text_tweet like '%not%'

- This second query, took 11 msec in the small db vs 4.8 secs in the big one (which retrieved 28141 rows)

Query 3: select text_tweet, retweet_of from tweet where retweet_of <>'NULL'

- This third query, took 12 msec in the small db vs 5:42 minutes in the big one (which retrieved 2369595 rows)

SQL CODE FOR TOY DATABASE:

```
create table aux(
   user_id varchar (32),
   screen_name varchar(64),
   location_user varchar(128),
   user_created varchar(64),
   tweet_id varchar (256),
   tweet_timestamp varchar (64),
   text_tweet varchar (280),
   retweet_of varchar(16)
   );

create table aux2(
   follower_id varchar (32),
   follower_screenName varchar(64),
   followee_id varchar (32),
   followee_screenName varchar (64)
   );

create table user1(
   user_id int primary key,
   screen_name varchar(64),
   location_user varchar(128),
   user_created varchar(64)
   );
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

```
create table tweet(
    user_id int,
    tweet_id int primary key,
    tweet_timestamp date,
    text_tweet varchar (280),
    retweet_of varchar(16),
    foreign key (user_id) references user1(user_id)
    );
create table follow(
    follower_id int,
    followee_id int,
    foreign key (follower_id) references user1(user_id),
    foreign key (followee_id) references user1(user_id)
    );

copy aux from '/home/juan/Descargas/small_tweets.txt' with delimiter '          ' NULL 'NULL';
copy aux2 from '/home/juan/Descargas/small_follow.txt' with delimiter '          ' NULL 'NULL';

insert into user1
select distinct cast (user_id as int), screen_name,location_user, user_created
from aux;

insert into tweet
select distinct cast (user_id as int), cast (tweet_id as int), to_date(tweet_timestamp, 'MM DD
YYYY'),text_tweet, retweet_of
from aux
where text_tweet <>'NULL';

insert into follow
select cast (follower_id as int), cast(followee_id as int)
from aux2 natural join user1
where cast (follower_id as int)=user1.user_id and cast (follower_id as int)=user1.user_id;
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

PART II: QUERIES

A)

```
select tweet_timestamp, text_tweet

from tweet

order by tweet_timestamp

asc  limit 1
```

```
"2009-01-02 22:07:58";"Barcelona. Guardiola resta importancia al retraso de Messi:
El entrenador del Barcelona, Pep Guardiola, justific.. http://tinyurl.com/9ham7f"
```

```
select tweet_timestamp, text_tweet

from tweet

order by tweet_timestamp

desc  limit 1
```

```
"2018-09-20 16:05:32";"We're ruled by "institutions that find it impossible to deal w
democracy" read @chakrabortty weep for Greece& Europe.
```

B)

```
create view num_followers as

select followee_id, count(follower_id) as number_followers

from follow

group by followee_id


select avg (number_followers) as density

from num_followers
```

```
Density

76.6499940169917434
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

D)

```
create view followed1 as

select follower_id as follower

from follow

where followee_id='760839'


create view followed2 as

select follower_id as follower

from follow

where followee_id='811737'


select f1 as Common_followers

from followed1 as f1,followed2 as f2

where f1.follower=f2.follower


Common_followers_ids

1)"(13139)"

2)"(855441)"

3)"(1614411)"

4)"(3144281)"

5)"(4029671)"

6)"(5412422)"

7)"(6471142)"

8)"(14349895)"

9)"(14395077)"

10)"(14436317)"

11)"(18944456)"

12)"(31090827)"

13)"(169962449)"
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

E)

```
create view followers_user1 as
select followee_id as followee
from follow
where follower_id='811737'

create view followers_user2 as
select followee_id as followee
from follow
where follower_id='769919'

select f1 as both_follow
from followers_user1 as f1,followers_user2 as f2
where f1.followee=f2.followee

both_follow_ids
1)"(5412422)"
2)"(10274252)"
3)"(11419202)"
4)"(13170872)"
5)"(14436317)"
6)"(15115726)"
7)"(16076032)"
8)"(17208081)"
9)"(18757892)"
10)"(18944456)"
11)"(20085289)"
12)"(20929745)"
13)"(39483072)"
```

```
14)"(41786228)"
15)"(49006538)"
16)"(94208950)"
17)"(103841173)"
18)"(121183700)"
19)"(124690469)"
20)"(140203389)"
........ (36 rows)
```

G)

```
create view number_of_followers as
select followee_id, count (*) as num_followers
from follow
group by followee_id

create view most_number_followers as
select max(num_followers) as most_followed
from number_of_followers

create view id_most_followed as
select followee_id
from most_number_followers f1, number_of_followers f2
where f1.most_followed=f2.num_followers

select screen_name
from user1 as u1, id_most_followed as u2
where u1.user_id=u2.followee_id
```

```
"Atleti"
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

I)

```
create view id_of_most_rt as
select user_id, count(*)
from tweet
group by user_id
order by count desc

select screen_name
from id_of_most_rt natural join user1
limit 1
"Atleti"
```

Andrés Mena Godino y Juan Moreno Díez. Group 1291

Made up query 1: Show the number of followers of the user who tweetted the most in 2018

```
create view tweets_2018 as

select *

from tweet

where tweet_timestamp >= '2018-01-01 00:00:00' and tweet_timestamp <= '2018-12-31
00:00:00'


create view most_twitters as

select user_id, count(*) as number_tweets

from tweets_2018

group by user_id

order number_tweets


create view max_number_tweets_2018 as

select max(number_tweets) as maximum_tweets

from most_twitters


create view id_most_tweeter as

select user_id

from most_twitters as m1, max_number_tweets_2018 as m2

where m1.number_tweets=m2.maximum_tweets


select screen_name

from user1 as u1, id_most_tweeter as u2

where u1.user_id=u2.user_id


"Atleti"
```

Made up query 2: From the users who registered using the location Spain (or Espaa), select the one who has the most followers, and show the screen name of those followers.

```
create view spanish_users as
select *
from user1
where location_user like 'Espaa' or location_user like 'Spain';

create view followers_of_spanish as
select user_id, count(follower_id) as num_followers
from spanish_users, follow
where spanish_users.user_id=followee_id
group by spanish_users.user_id;

create view max_num_followers as
select max(num_followers) as max_num_followers
from followers_of_spanish;

create view most_popular_spanish_id as
select user_id
from max_num_followers as a1, followers_of_spanish as a2
where a1.max_num_followers=a2.num_followers;

create view ids_followees_spanish as
select follower_id
from most_popular_spanish_id, follow
where followee_id=most_popular_spanish_id.user_id;

select screen_name
from user1, ids_followees_spanish
```

```
where user1.user_id=ids_followees_spanish.follower_id
```

```
Screen_name
1)"fluzo"
2)"edans"
3)"raulsensato"
4)"carballo"
5)"luisrull"
6)"juanmadiaz"
7)"ferpectamente"
8)"marininmonroe"
9)"CarolinaD"
10)"teseo"
11)"davidperez"
12)"angeljimenez"
13)"nazaret"
14)"Lentejitas"
15)"el_pais"
16)"iGSMr1982"
17)"levante_emv"
18)"gvisoc"
19)"solobasket"
20)"juanlusanchez"
…….. (2921 rows)
```