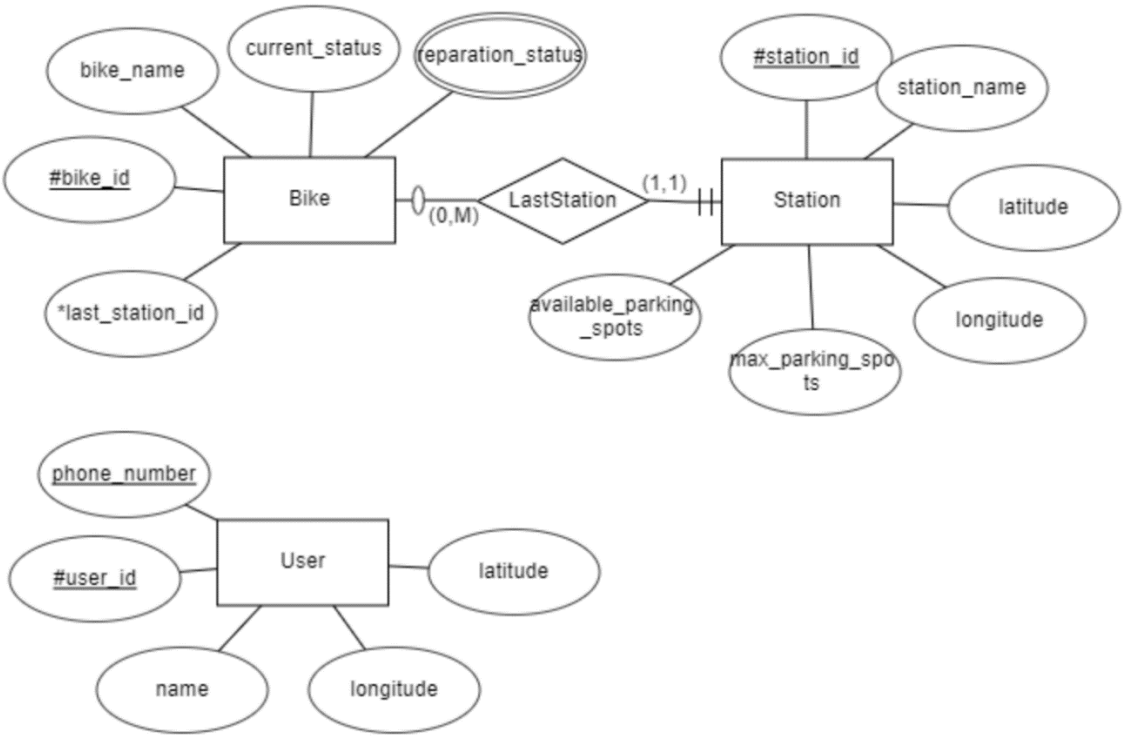
Assignment 1 by Thomas Barth

1.a)



Primary Keys: bike_id for bike, station_id for station and user_id. These are primary keys since each is unique and can be used to identify the entity.

Foreign Keys: last_station_id.

The last station id foreign key I was a bit unsure about since I did not find a good definition in the book, so I used the definition from (DePaul University, 2024) witch is linked below in references and it states: “A foreign key is an attribute that completes a relationship by identifying the parent entity.” Since it can be used to identify the last station I would argue this to be a foreign key and it completing the relationship between the bike and the station called LastStation.

I tried also to create another user with my phone number to find out if phone number would be unique or not. Bergen bysykkel did not allow this, so it must be unique to each user.

1.b)

Based on the information so far there is no direct relationship between a bike and a user, and a user and a station. We know that the user has a GPS position witch is used to find the closest station, but dose not imply there is a ownership between a user and a bike. The only explicit relationship is between the bike and the station to find the last station the bike was parked at, and determine the foreign key in bike referencing a station.

To deploy a real bike renting service however, we should add some additional relationships like renting, witch could be between the bike and the user.

2.a)

The proposed solution would cause a few unwanted dependencies:

1. Repetition of data:

A user having more than one subscription with all of the subscription attributes inside the user entity will create redundant data for this user, wasting storage.

2. Difficulti in querying:

Querying a user enity for the details about its different subscriptions will become very hard and complex and will require more logic like identifyers inside all of the subscription attributes to match them together.

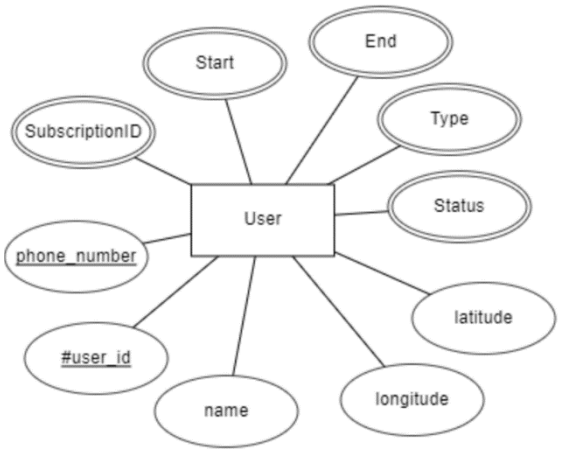


Figure 1

3. Adding new subscriptions:

It would be impossible to create new subscriptions without first having a user, making development trickier. You also must write all the subscription data again everytime you add a user.

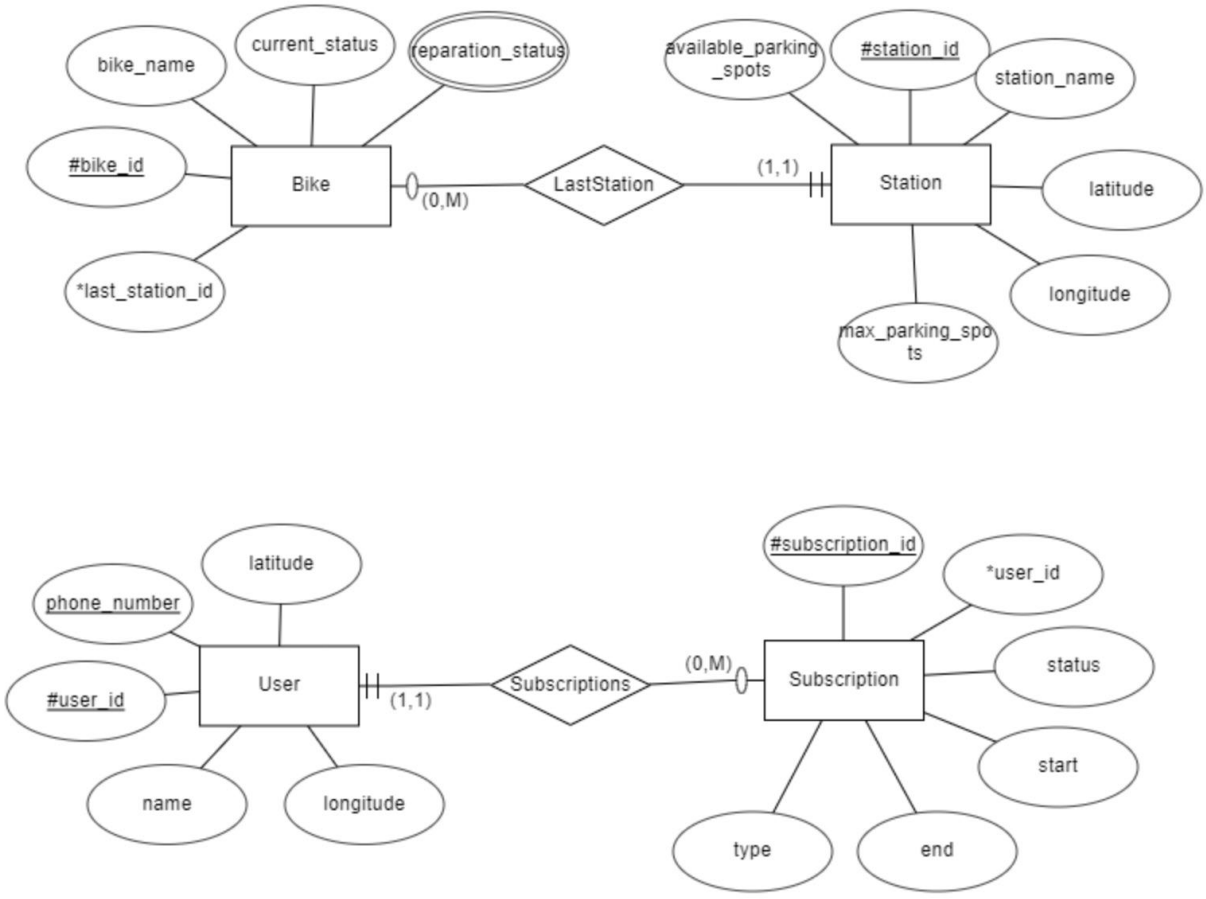
2.b)

To solve the issues listed above it is simples to separate the subscription attributes to a new entity we can call subscription. This entity/table will have its own attributes like: subscriptionID, userID, status, start, end and type.

Eks: SUBSCRIPTION(#SubscriptionID, *UserID, Status, Start, End, Type)

2.c)

When adding the subscription entity and the relationship be between the subscription entity and the user entity we have the following diagram:



3.a)

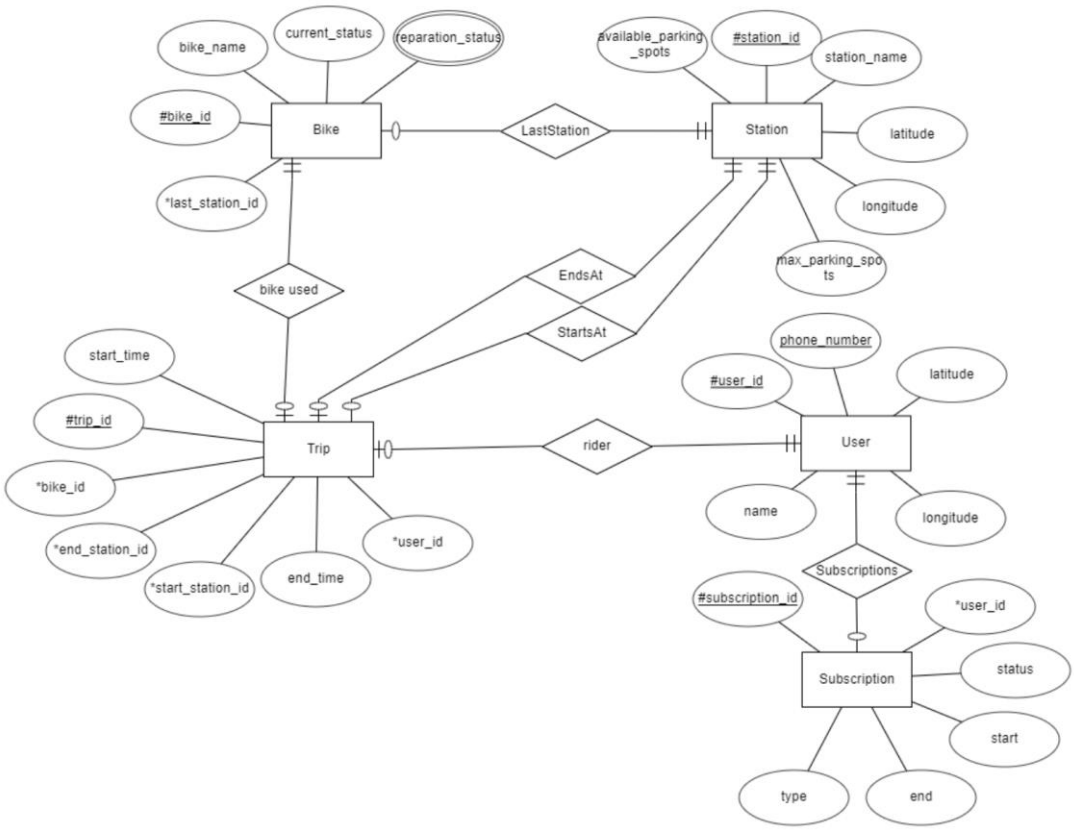
The attributes for the trip entity given by the database description:

1. Trip_id – primary key, an identifier for each trip.
2. User_id - foreign key, references the user taking the trip.
3. Bike_id – foreign key, references the bike the user is using.
4. Start_time – the time the trip started.
5. end_time – this time the trip ended.
6. Start_station_id – foreign key, references the id from the start station.
7. End_station_id - foreign key, references the id from the end station.

Table:

Trip(#trip_id, *user_id, *bike_id, start_time, end_time, *start_station_id, *end_station_id)

3.b)



3.c)

1NF is remove redundant/repeating data. Rule we can follow are: 1. All attributes should be atomic, so it should not contain multiple values. 2. An attribute should have a domain. 3. Each attribute should have a unique name. 4. Order of the data in the attributes should not matter.

2NF is making sure all the attributes in the entity are only reliant on the primary key and that the database is already in 1NF. This means that you can get any meaningful data from an entity from just the primary key. For example we could ask: BIKENAME from BIKE where BIKE_ID = 001.

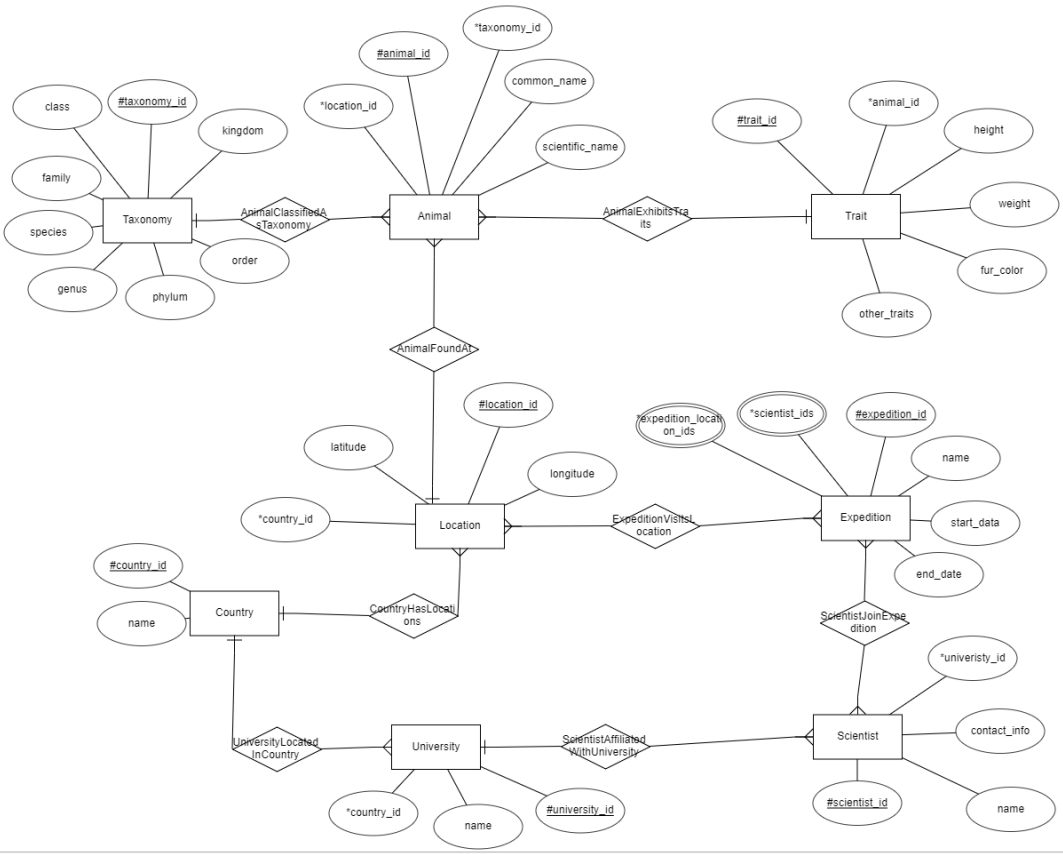
3NF is that the database should have no transitive dependencies, so there should be no indirect relationship between the entities attributes and that the table is already in 2NF.

BCNF is also known as the Boyce-Codd Normal Form and tells us that for any dependency where A derives B, then A should be a super key and the table is already in 3NF. This means that A must be a key attribute and B cannot be a key attribute.

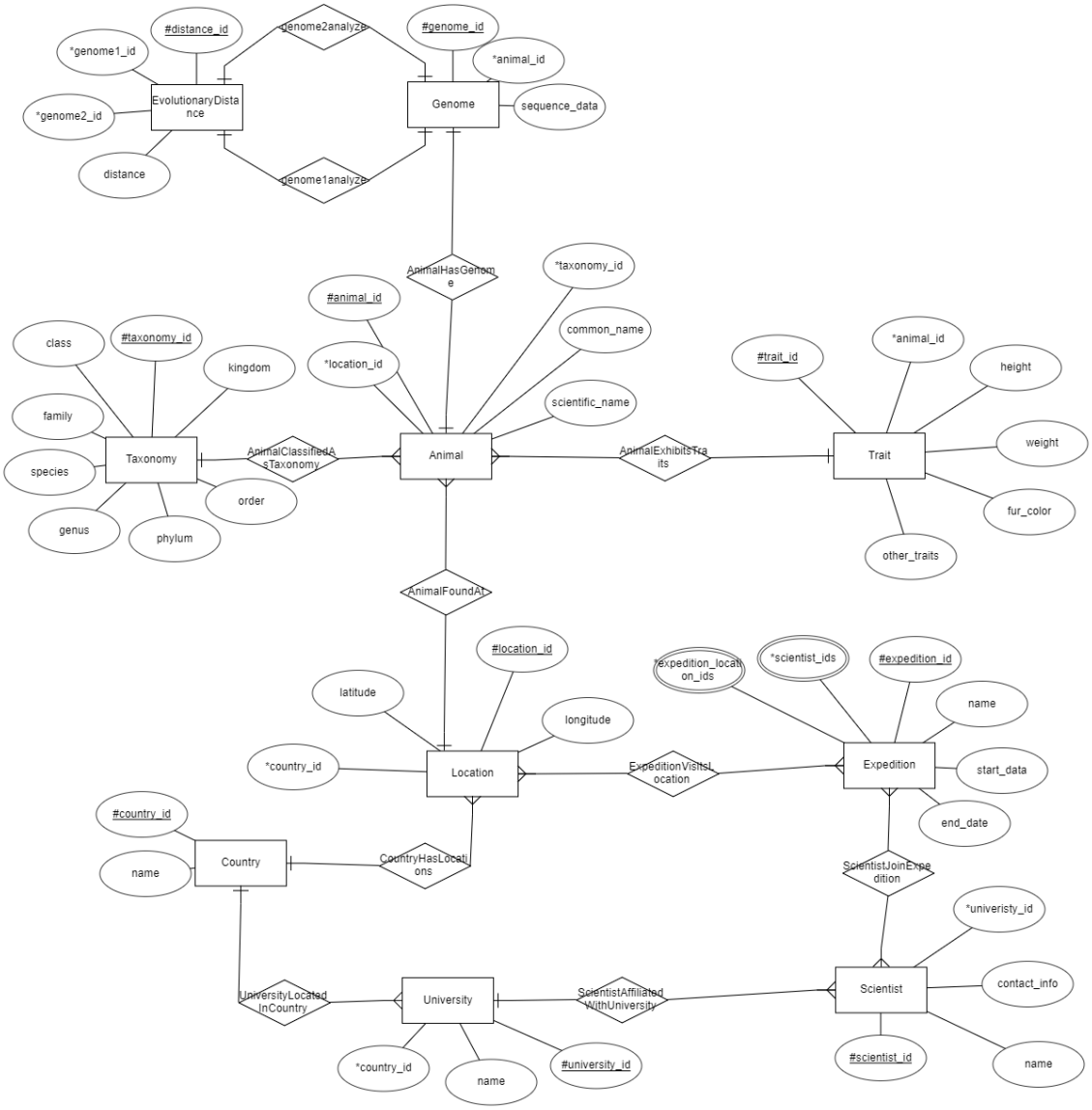
3.d)

First, I would like to do is something about the repair status, since this is a multivalued attribute we can't really call it atomic. So I would like to create an entity called repair rapport and remove the repair status from the bike enitity: RepairRapport(#raport_id, *bike_id, rapport, data). Now the ER diagram is up to the first normalization form. We fulfil the second normalization from already seeing all of our entities have ids and all attributes are relevant for the entity they are connected too. We also already fulfil the third from where the whole database has no indirect relationships between the attributes. We also fulfil the BCNF form where all of our primary keys works as a super key, so we have no non-key-attributes which derives parts of a key attribute.

4, subproblem 1)



4, subproblem 2)



Referanser

DePaul University. (2024). *https://condor.depaul.edu/gandrus/240IT/accesspages/primary-foreign-keys.htm#Entities_with_Key_Attributes*.

Sources:

1. https://condor.depaul.edu/gandrus/240IT/accesspages/primary-foreign-keys.htm#Entities_with_Key_Attributes
2.