Databases project Part 1

Pontus Elmrin and Henry He

One paragraph description of project

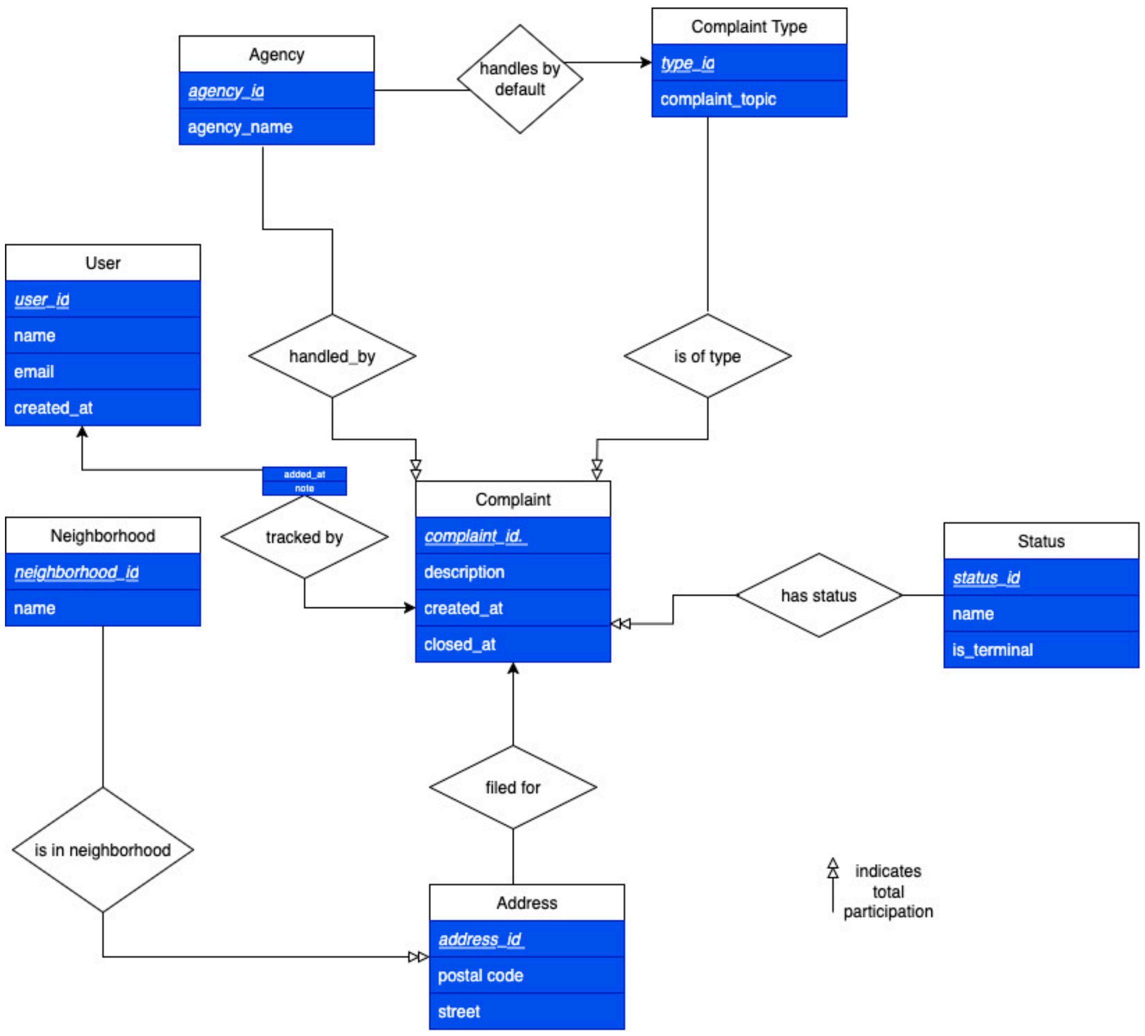
Highlight interesting and challenging parts
We want to make a database that tracks new york city agencies that allows users to see which agencies are slow or fast at fixing 311 complaints and if the speed differs across neighborhoods. The entity sets we identified are: Complaint type, complaint, agency, address, status, neighborhood, user.

The relationship sets we identified are:

- complaint is handled by agency (many to one),
- complaint is filed for address (many to one),
- complaint is of type complaint type (many to one),
- agency handles by default complaint type (one to many),
- complaint has status status (many to one),
- complaint is tracked by user (many to many),
- address is in neighborhood (many to one)

Below is an E-R diagram. We will do the front-end version.

ER model



Real world constraints

- **complaint**: closed_at is greater than/equal to created_at
- **natural identifiers**: agency.name, complaint_type.complaint_topic, neighborhood.name, status.name are unique
- **postal code:** must be valid US ZIP code format
- **deletion:** once a complaint references an agency, complainttype, status, neighborhood, they cannot be deleted
- **policy constraint:** for each complaint type, there is at most one agency designated as the default handler

Data plan

We will use the data from the dataset "311 Service Requests from 2010 to Present" provided by new york city open data. Link here: https://data.cityofnewyork.us/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9/about_data

Description of user interaction plan

We have three views for the users:

Neighborhood view. The user can input a neighborhood and the application will return some statistics about that neighborhood, such as average speed taken for requests, counts by complaint type, etc.

Agency view. The user can input an agency and get their average completion speed. Can compare to city-wide benchmark.

User profile view. The application will also feature a user profile that contains user info like name and tracked complaints. So someone can find the complaints they have made/track complaints they are interested in and add them to their user profile.

Description of contingency plan

If one student drops, the project doesn't encompass the user interaction plan.

DDL implementation of ER model

```
create table agency
(agency_id varchar(10),
 agency_name varchar(25) not null UNIQUE,
 primary key(agency_id))
```

```
create table complaint
( complaint_id varchar(8),
  description varchar(500),
  created_at timestamp not null,
  closed_at timestamp,

  agency_id varchar(10) not null,
  complaint_type_id varchar(10) not null,
  address_id varchar(12) not null,
  status_id varchar(8) not null,
  primary key(complaint_id),
  foreign key (agency_id)  references agency,
  ON UPDATE CASCADE ON DELETE RESTRICT
  foreign key (complaint_type_id)    references complaint_type,
  ON UPDATE CASCADE ON DELETE RESTRICT
  foreign key (address_id) references address,
  ON UPDATE CASCADE ON DELETE RESTRICT
  foreign key (status_id)  references status,
  ON UPDATE CASCADE ON DELETE RESTRICT
  check (closed_at is null or closed_at >= created_at))
```

```
create table status
(status_id varchar(8),
 name varchar(25) not null UNIQUE,
 is_terminal boolean not null,
 primary key(status_id))
```

```
create table complaint_type
(complaint_type_id varchar(10),
 complaint_topic varchar(100) not null UNIQUE,
 primary key (complaint_type_id))
```

```
create table neighborhood
(neighborhood_id varchar(10),
 name varchar(60) not null UNIQUE,
 primary key(neighborhood_id))
```

```
create table address
(address_id varchar(12),
 neighborhood_id varchar(10) not null,
 street1 varchar(50) not null,
 street2 varchar(50),
 postal_code varchar(10) not null,
 CHECK (postal_code ~ '^[0-9]{5}(-[0-9]{4})?$')
 foreign key (neighborhood_id) references neighborhood,
 primary key(address_id) )
```

```
create table user
( user_id varchar(12),
  name varchar(20) not null,
  email_address varchar(50) UNIQUE,
  created_at timestamp,
  primary key(user_id))
```

```
create table tracked_by
( user_id varchar(12),
  complaint_id varchar(8),
  added_at timestamp,
  note varchar(240),
  primary key(user_id ,  complaint_id),
  foreign key(complaint_id) references complaint,
  foreign key(user_id) references user)
```

```
create table handle_by_default
( complaint_type_id  varchar(10),
  agency_id varchar(8) not null,
  primary key(complaint_type_id),
  foreign key(complaint_type_id) references complaint_type,
  foreign key(agency_id) references agency)
```