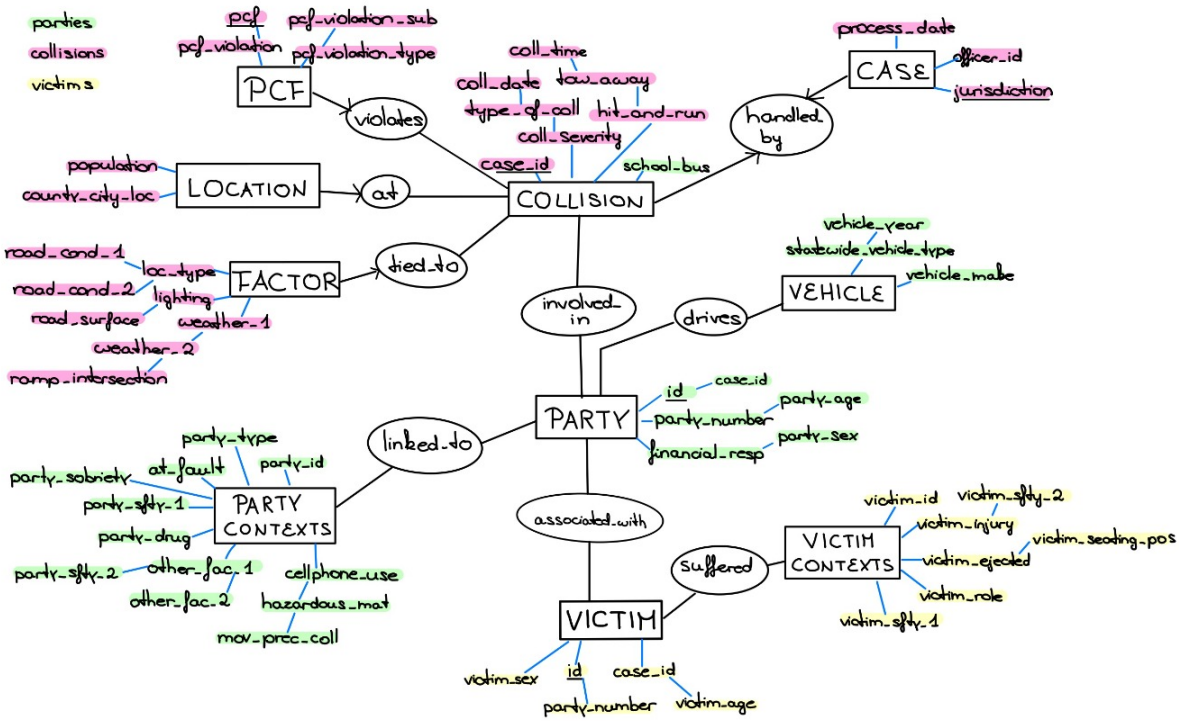
ER Model



Data constraints

Participation constraints

- Partial participation from `Collision` in the `involved in` relationship: `case_id 10` has no involved party.
- 'Exactly one' participation from `Party` in the `involved in` relationship: Every party has an associated `case_id` that is unique.
- Partial participation from `Party` in the 'associated with' relationship: Some `case_id` have no victim e.g. `case_id 1,2`.
- 'Exactly one' participation from `Victim` in the `associated with` relationship: Given a `case_id`, if there is a victim, there is a party (unique).

Additional constraints

- In the project description, each attribute that is nullable is clearly stated:

  ```
  > Blank or - - Not Stated
  ```

  Hence we can mark as `NOT NULL` every other attribute, except for `PRIMARY KEY`s, which are implicitely so.

- Since we are using a star schema, we cannot express the following constraints in the SQL code: every collision has a location, a factor, a pcf and a case. Likewise, every party has a vehicle and a party context. Finally, every victim has a victim context.

- As said in the moodle forum, a victim is not a party.

Design choices

Star schema

We decided to cluster attributes into separate entities following a star schema. Some groups are obvious, others are debatable.

The obvious groups are:

- For the `Collision` entity, some attributes form the logical groups `Pcf`, `Location`, `Factor`, `Case`.
- Similarly, for the `Victim` entity, the only logical group is `Vehicle`.

We also decided to add two less obvious groups:

- For the `Party` entity, attributes which are orthogonal to the collision are not stored in a separate entity (age, sex, ...)

  Attributes which are about the context of the collision are stored in a `PartyContext` entity.

- Similarly for the `Victim` entity, we have a `VictimContext` entity.

Attribute types in the SQL code

- In the project description, some attributes are enums: they can only take on specific pre-defined values. Therefore, we can let them be `INTEGER` and have lookup tables when we dump the `.csv`s into a `SQL` database.

  The alternative would be to let them be `VARCHAR`. The problem with this approach is that determining the max length means looking up the max number of characters for each attribute.

  Granted: creating lookup tables would require the same amount of work; however it leads to substantial data compression.

  Note that these attributes are the same that are `nullable`.

- Similarly, `tow_away` from `Collisions` can be translated from a `float` (`0.0` or `1.0`) to a `BIT`.
- The rest of the attributes are clearly `INTEGER` from the project description as well as upon inspection of the values in the `.csv` files.

SQL code

Collision

```
CREATE TABLE Collisions(case_id INTEGER,
                        collision_date DATE NOT NULL,
                        collision_time TIME NOT NULL,
                        type_of_collision CHAR(1),
                        collision_severity CHAR(1) NOT NULL,
                        hit_and_run CHAR(1) NOT NULL,
                        tow_away BIT NOT NULL,
                        PRIMARY KEY(case_id))


CREATE TABLE Pcfs(case_id INTEGER NOT NULL,
                    pcf_violation INTEGER NOT NULL,
                    pcf_violation_category CHAR(1),
                    pcf_violation_subsection CHAR(1),
                    FOREIGN KEY(case_id) REFERENCES Collisions(case_id))


CREATE TABLE Locations(case_id INTEGER NOT NULL,
                        population INTEGER,
```

```
                        county_city_location INTEGER NOT NULL,
                        FOREIGN KEY(case_id) REFERENCES Collisions(case_id))


CREATE TABLE Factors(case_id INTEGER NOT NULL,
                        location_type CHAR(1),
                        lighting CHAR(1),
                        road_condition_1 CHAR(1),
                        road_condition_2 CHAR(1),
                        road_surface CHAR(1),
                        weather_1 CHAR(1),
                        weather_2 CHAR(1),
                        FOREIGN KEY(case_id) REFERENCES Collisions(case_id))


CREATE TABLE Cases(case_id INTEGER NOT NULL,
                    process_date DATE NOT NULL,
                    officer_id INTEGER NOT NULL,
                    jurisdiction INTEGER NOT NULL,
                    FOREIGN KEY(case_id) REFERENCES Collisions(case_id))
```

Party

Only permanent attributes of a `Party` are stored in the `Parties` table. Attributes from the context of the collision are stored in the `PartyContexts` table.

```
CREATE TABLE Parties(id INTEGER,
                        case_id INTEGER NOT NULL,
                        party_number INTEGER NOT NULL,
                        finanicial_responsibility CHAR(1),
                        party_age INTEGER NOT NULL,
                        party_sex CHAR(1),
                        PRIMARY KEY(id),
                        FOREIGN KEY(case_id) REFERENCES Collisions(case_id))


CREATE TABLE PartyContexts(party_id INTEGER NOT NULL,
                        at_fault CHAR(1) NOT NULL,
                        cellphone_use CHAR(1),
                        hazardous_materials CHAR(1),
                        movement_preceding_collision CHAR(1),
                        other_associate_factor_1 CHAR(1),
                        other_associate_factor_2 CHAR(1),
                        party_drug_physical CHAR(1),
                        party_safety_equipment_1 CHAR(1),
                        party_safety_equipment_2 CHAR(1),
                        party_sobriety CHAR(1),
                        FOREIGN KEY(party_id) REFERENCES Parties(id))


CREATE TABLE Vehicles(party_id INTEGER NOT NULL,
                        school_bus_related CHAR(1),
                        statewide_vehicle_type CHAR(1),
                        vehicle_make VARCHAR(30) NOT NULL,
                        vehicle_year INTEGER NOT NULL,
                        FOREIGN KEY(party_id) REFERENCES Parties(id))
```

Victim

All collision context specific attributes of a `Victim` are stored in a separate table.

```
CREATE TABLE Victims(id INTEGER,
                        case_id INTEGER NOT NULL,
                        party_number INTEGER NOT NULL,
                        victim_age INTEGER NOT NULL,
                        victim_sex BIT,
```

```
                        PRIMARY KEY(id),
                        FOREIGN KEY(case_id) REFERENCES Collisions(case_id))


CREATE TABLE VictimContexts(victim_id INTEGER NOT NULL,
                            victim_degree_of_injury CHAR(1) NOT NULL,
                            victim_ejected CHAR(1),
                            victim_role CHAR(1) NOT NULL,
                            victim_safety_equipment_1 CHAR(1),
                            victim_safety_equipment_2 CHAR(1),
                            victim_seating_position CHAR(1),
                            FOREIGN KEY(victim_id) REFERENCES Victims(id))
```