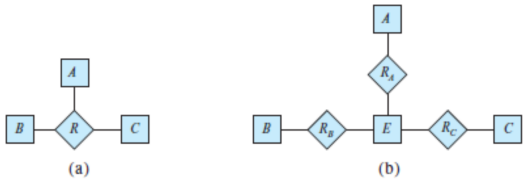
CISC 450 - HW 6

Marcus Haldane

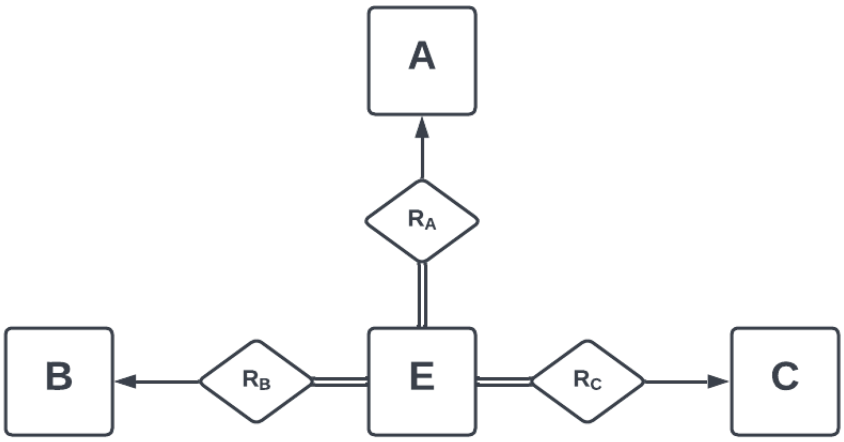
October 29, 2022

Task 1: Consider the representation of the ternary relationship figure (a) of using the binary relationships illustrated in figure (b) (attributes not shown).



(a.) Show a simple instance of E, A, B, C, RA, RB, and RC that cannot correspond to any instance of A, B, C, and R.
Take for example the following instance of E, A, B, C, R_A, R_B, and R_C:
$E = \{e_1, e_2\}$, $A = \{a_1\}$, $B = \{b_1\}$, $C = \{c_1, c_2\}$, $R_A = \{(e_1, a_1)\}$, $R_B = \{(e_1, b_1)\}$, and $R_C = \{(e_1, c_1), (e_2, c_2)\}$. Because of the tuple $\{(e_2, c_2)\}$, there exists some value in E which is has no relationship with some value in A and B. As a result, there cannot be a corresponding instance of A, B, C, and R which corresponds to E, R_A, R_B and R_C.

(b.) Modify the E-R diagram of Figure (b) to introduce constraints that will guarantee that any instance of E, A, B, C, RA, RB, and RC that satisfies the constraints will correspond to an instance of A, B, C, and R.
By forcing total participation constraints on E, R_A, R_B, and R_C, we are able to ensure that every tuple in E has a relationship with A, B and C.



(c.) Modify the preceding translation to handle total participation constraints on the ternary relationship.
A, B, and C totally participate in the relationship R, enforced by a total participation constraint between A and R_A, B and R_B, and C and R_C.

Task 2: (Refer to University Database) Consider a relation such as sec_course, generated from a many-to-one relationship set sec_course. Do the primary and foreign key constraints created on the relation enforce the many-to-one cardinality constraint? Explain why.
Given the following DDLs for the University Database Tables:

DDL for university.course

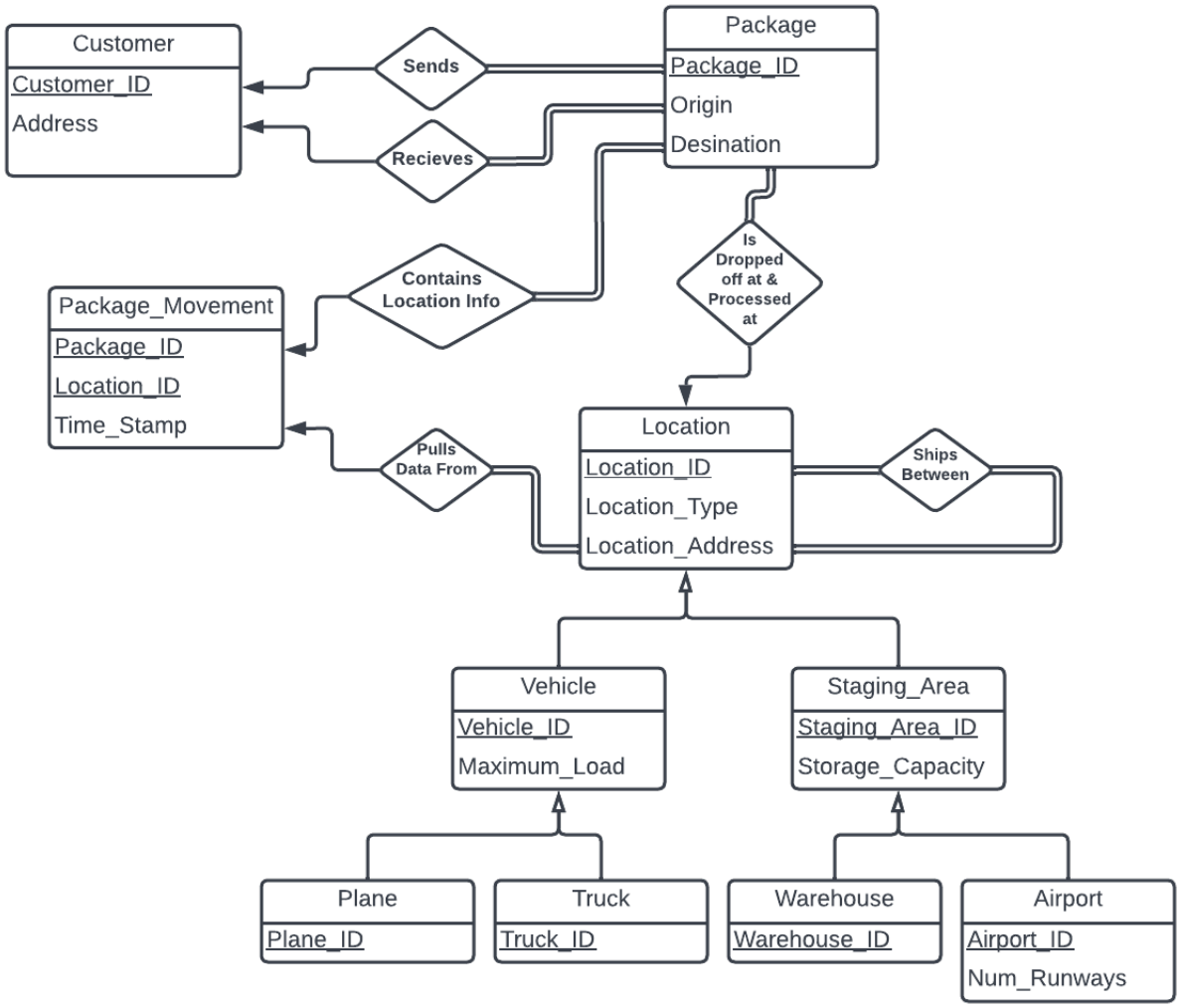
```
CREATE TABLE `course` (
  `course_id` varchar(8) NOT NULL,
  `title` varchar(50) DEFAULT NULL,
  `dept_name` varchar(20) DEFAULT NULL,
  `credits` decimal(2,0) DEFAULT NULL,
  PRIMARY KEY (`course_id`),
  KEY `dept_name` (`dept_name`),
  CONSTRAINT `course_ibfk_1` FOREIGN KEY (`dept_name`) REFERENCES `department` (`dept_name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

DDL for university.section

```
CREATE TABLE `section` (
  `course_id` varchar(8) NOT NULL,
  `sec_id` varchar(8) NOT NULL,
  `semester` varchar(6) NOT NULL,
  `year` decimal(4,0) NOT NULL,
  `building` varchar(15) DEFAULT NULL,
  `room_number` varchar(7) DEFAULT NULL,
  `time_slot_id` varchar(4) DEFAULT NULL,
  PRIMARY KEY (`course_id`,`sec_id`,`semester`,`year`),
  KEY `building` (`building`,`room_number`),
  CONSTRAINT `section_ibfk_1` FOREIGN KEY (`course_id`) REFERENCES `course` (`course_id`),
  CONSTRAINT `section_ibfk_2` FOREIGN KEY (`building`, `room_number`) REFERENCES `classroom` (`building`, `room_number`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

The primary and foreign key constraints of the sec_course relation are not entirely sufficient to enforce the necessary constraints for the sec_course relation. This is the case because there is no constraint which enforces total participation, meaning that an instance of a course or an instance of a section may or may not participate in the sec_course relation. That being said, as can be seen in the DDL for the university.section table, the primary key is defined as the attributes `course_id`, `sec_id`, `semester`, and `year`, with `course_id` referencing the university.course table as a foreign key. These characteristics are shared with the sec_course relation. Given this information, we can say that the many-to-one cardinality constraint is enforced for the sec_course relation, with the exception of cases where there is not participation on the part of all instances of the section and course tables.

Task 3: Design a database for a worldwide package delivery company (e.g., DHL or FedEx). The database must be able to keep track of customers who ship items and customers who receive items; some customers may do both. Each package must be identifiable and trackable, so the database must be able to store the location of the package and its history of locations. Locations include trucks, planes, airports, and warehouses. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints.



```sql
create table Customer
     (Customer_ID numeric(8,0) not null,
     Address varchar(120) not null,
     primary key (Customer_ID)
     );

create table Package
     (Package_ID numeric(8,0) not null,
     Origin varchar(120) not null,
     Destination varchar(120) not null,
     primary key (Package_ID),
     foreign key (Origin) references Customer (Address),
     foreign key (Destination) references Customer (Address)
     );

create table Package_Movement
     (Package_ID numeric(8,0) not null,
     Location_ID numeric(8,0) not null,
     Time_Stamp date,
     primary key (Package_ID, Location_ID),
     foreign key (Package_ID) references Package (Package_ID),
     foreign key (Location_ID) references Location (Location_ID)
     );

create table Location
     (Location_ID numeric(8,0) not null,
     Location_Type varchar(30),
     Location_Address varchar(120),
     primary key (Location_ID)
     );

create table Vehicle
     (Vehicle_ID numeric(8,0) not null,
     Maximum_Load numeric(6,0),
     primary key (Vehicle_ID)
     );

create table Staging_Area
     (Staging_Area_ID numeric(8,0) not null,
     Storage_Capacity numeric(10,0),
     primary key (Staging_Area_ID)
     );

create table Plane
     (Plane_ID numeric(8,0) not null,
     primary key (Plane_ID)
     );

create table Truck
     (Truck_ID numeric(8,0) not null,
     primary key (Truck_ID)
     );

create table Warehouse
     (Warehouse_ID numeric(8,0) not null,
     primary key (Warehouse_ID)
     );

create table Airport
     (Airport_ID numeric(8,0) not null,
     Num_Runways numeric(2,0),
     primary key (Airport_ID)
     );
```

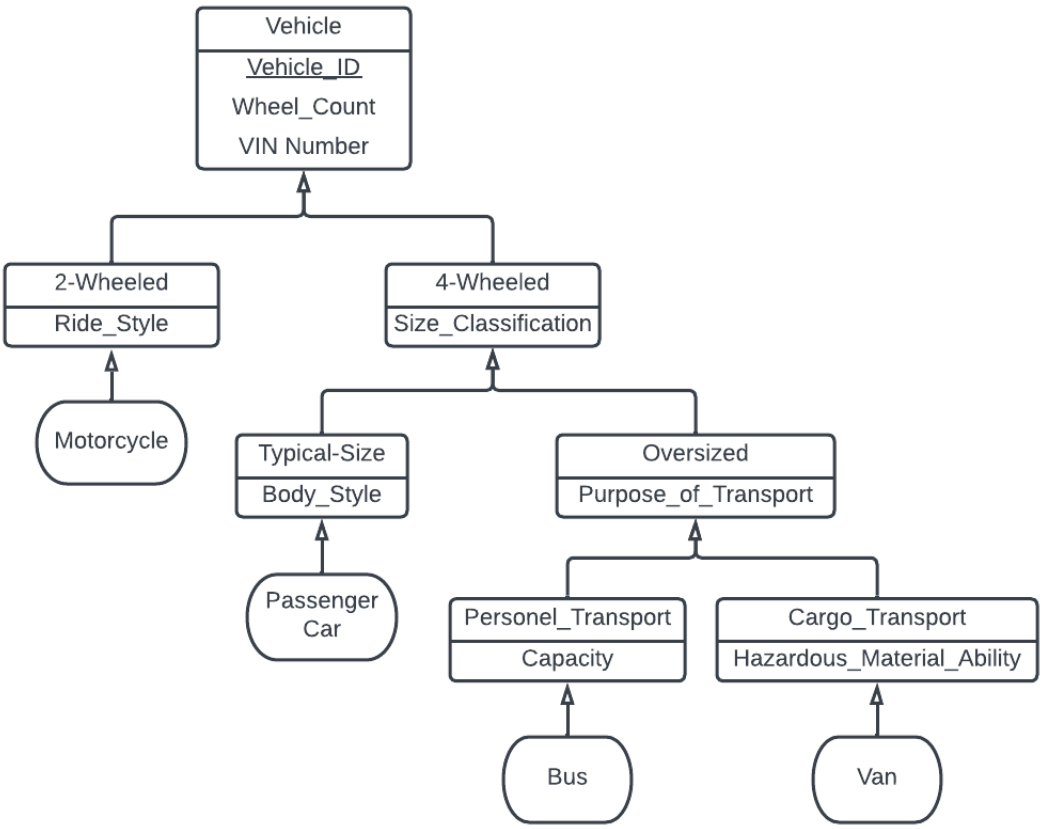
Task 4: Design a generalization–specialization hierarchy for a motor vehicle sales company. The company sells motorcycles, passenger cars, vans, and buses. Justify your placement of attributes at each level of the hierarchy. Explain why they should not be placed at a higher or lower level.

When designing the generalization-specialization hierarchy below, I started with the fact that everything sold by this motor vehicle company is a vehicle. That is the classification that each of their products falls into, and as such this category takes its place at the top of the hierarchy. Following this I considered a number of possibilities for how vehicles could be further subdivided in meaningful ways. The most meaningful difference between this broad category of vehicles is the general physical structure and method of function of these vehicles. This is how I came to separating the vehicles category into 2-wheeled vehicles and 4-wheeled vehicles. A passenger car, bus and van are all driven in a relatively similar manner, as all of the above consist of 4 wheels, a drive train, differential, exterior body, steering wheel, etc... Motorcycles, on the other hand, drive much more like a bicycle than any of the 4-wheeled vehicles. A motorcycle has handlebars rather than a steering wheel. It has a chain or belt rather than a drive train. When driving a motorcycle, it is important for the operator to lean into and out of turns to keep the motorcycle upright. For these reasons, I separated the 2-wheeled vehicles and 4-wheeled vehicles.

The next separation I felt the need to make was between the purpose for the 4-wheeled vehicle was intended for; and consequently the size of the vehicle. As a general rule, passenger cars will almost always be smaller than a bus or van, because there is a totally different purpose in mind for each type of vehicle. Passenger cars are meant for singular personal transport, and their size reflects this. Alternatively, buses and vans are meant to move large quantities of people or materials. Consequently, I made the first division of the 4-wheeled vehicles on the basis of their status as typical-size or oversized.

That final distinction was that of being designed for moving large quantities of people vs moving materials. Buses are obviously meant to be public transport, moving many people. Vans are meant to have a large volume which can be used to fit a large capacity of materials generally.

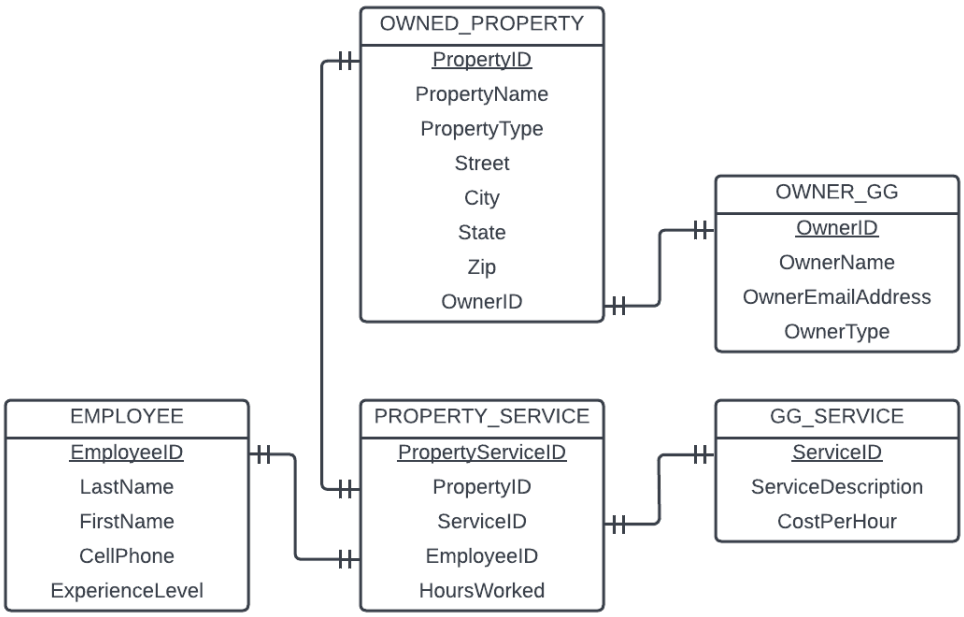
There are many ways that this hierarchy could be made, but I believe that these distinctions are meaningful and helpful for the purposes of organization.



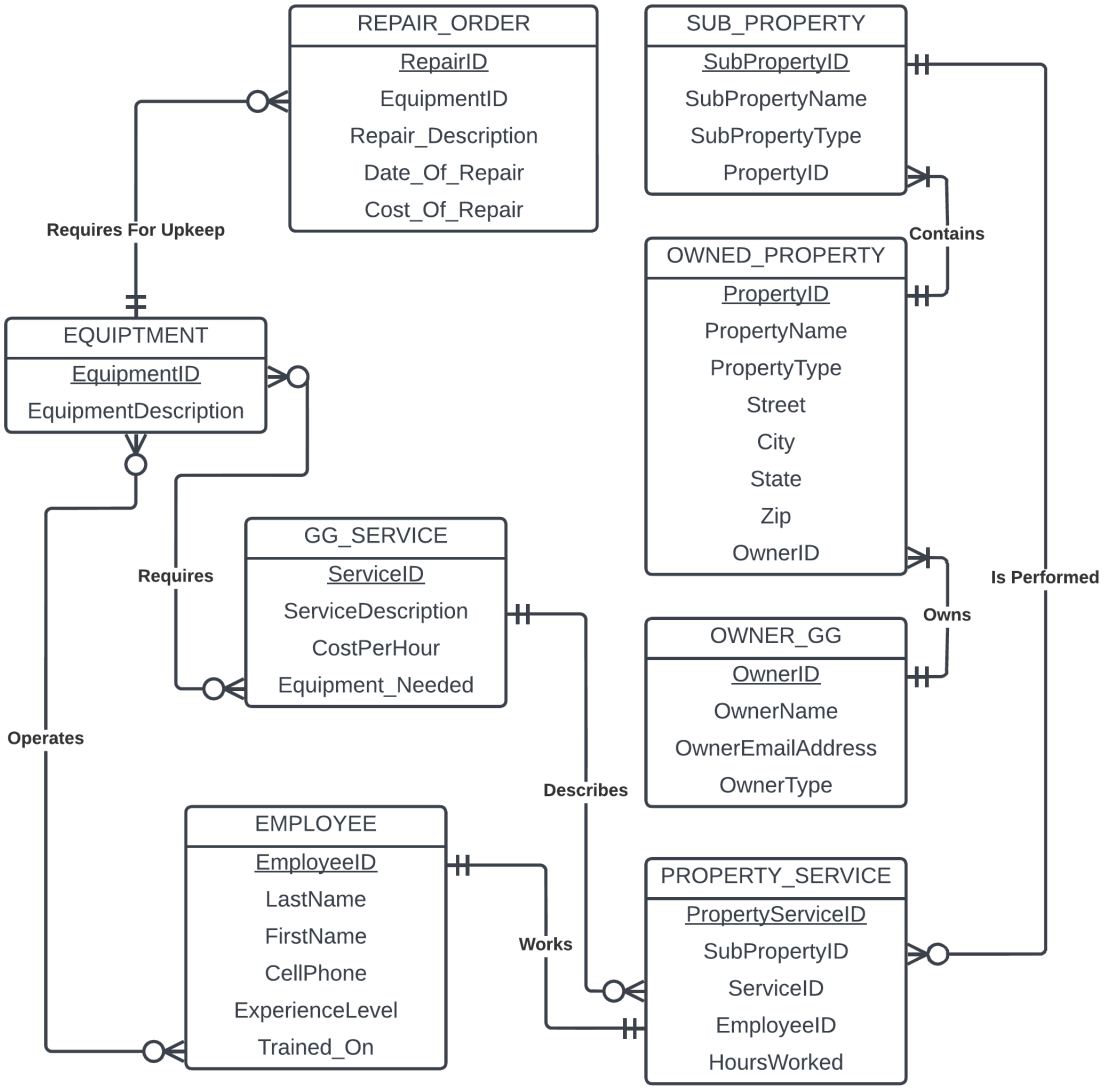
Task 5: Garden Glory wants to expand its database applications beyond the recording of property services. The company still wants to maintain data on owners, properties, employees, services, and the service work done at the properties, but it wants to include other data as well. Specifically, Garden Glory wants to track equipment, how it is used during services, and equipment repairs. In addition, employees need to be trained before they use certain equipment, and management wants to be able to determine who has obtained training on which equipment.

With regard to properties, Garden Glory has determined that most of the properties it services are too large and complex to be described in one record. The company wants the database to allow for many subproperty descriptions of a property. Thus, a particular property might have subproperty descriptions such as Front Garden, Back Garden, Second-Level Courtyard, and so on. For better accounting to the customers, services are to be related to the subproperties rather than to the overall property.

(a.) Draw an E-R data model for the Garden Glory database schema. Use the IE Crow's Foot E-R model for your E-R diagrams. Justify the decisions you make regarding minimum and maximum cardinalities.



(b.) Extend and modify the E-R data model to meet Garden Glory's new requirements. Use the IE Crow's Foot E-R model for your E-R diagrams. Create appropriate identifiers and attributes for each entity. Justify the decisions you make regarding minimum and maximum cardinalities.



- I chose to require a 1-to-Many relationship between `EQUIPMENT` and `REPAIR_ORDER`s, because for each order that is generated when a piece of equipment needs repairing, only 1 piece of equipment is being fixed, but over the course of a piece of equipment's lifetime, it may require many `REPAIR_ORDER`s.
- I chose to require a Many-to-Many relationship between the `EQUIPMENT` and `EMPLOYEE`s, because for each a piece of equipment, there may be zero, one, or many `EMPLOYEE`s who are trained to use that piece of `EQUIPMENT`. Additionally, for each `EMPLOYEE`, they may be trained on zero, one, or many pieces of `EQUIPMENT`.
- I chose to require a Many-to-Many relationship between `EQUIPMENT` and `GG_SERVICES`s, because for each `GG_SERVICE`, there may be zero, one, or many pieces of `EQUIPMENT` which are necessary to perform the job. Additionally, for each piece of `EQUIPMENT`, there may be zero, one, or many `GG_SERVICES` which require that piece of `EQUIPMENT`.
- I chose to require a 1-to-1 relationship between the `EMPLOYEE` and the associated `PROPERTY_SERVICE`s, because for each `PROPERTY_SERVICE`, there is one `EMPLOYEE` that goes to a job and performs the `PROPERTY_SERVICE`.

- I chose to require a 1-to-Many relationship between the GG_SERVICEs and PROPERTY_SERVICEs, because for each PROPERTY_SERVICE, there is only one GG_SERVICE that describes the details of that PROPERTY_SERVICE. However, for every GG_SERVICE, there may be many PROPERTY_SERVICEs which have been performed and billed as that particular GG_SERVICE.
- I chose to require a 1-to-Many relationship between the SUB_PROPERTYs and the associated PROPERTY_SERVICEs, because for each PROPERTY_SERVICE, there is only one SUB_PROPERTY on which the PROPERTY_SERVICE is performed on. However, for each SUB_PROPERTY, there may be many PROPERTY_SERVICEs performed on that SUB_PROPERTY.
- I chose to require a 1-to-Many relationship between the SUB_PROPERTYs and the associated OWNED_PROPERTY, because each SUB_PROPERTY is associated with 1 and only 1 OWNED_PROPERTY, but each OWNED_PROPERTY may be broken down into 1 or many SUB_PROPERTYs.
- I chose to require a 1-to-Many relationship between the OWNED_PROPERTYs and the associated OWNER_GG, because each OWNED_PROPERTY is associated with 1 and only 1 OWNER_GG, but each OWNER_GG may own 1 or many OWNED_PROPERTYs.

(c.) Describe how you would go about validating the model in part B.
The best way to validate the model from part B would be to use 3NF algorithms to test for 3NF. If we consider functional dependencies whose right-hand side consists of a single attribute, that should be sufficient.