Relational Schema

User = (Username, First_Name, Last_Name, Address, Birthdate)

Employee = (Username [fk1], TaxID, Hired, Salary, Experience)
fk1: Username -> User.Username
TaxID is unique

Pilot = (Username [fk2], License_Type, Experience)
fk2: Username -> Employee.Username
License_Type is non-null

Worker = (Username [fk3])
fk3: Username -> Employee.Username

Owner = (Username [fk4])
fk4: Username -> User.Username

Location = (Label, X_Coord, Y_Coord, Space)

Service = (ID, Name, Label [fk5], Managed[fk6])
fk5: Label -> Location.Label
fk6: Managed -> Worker.Username

Restaurant = (Name, Spent, Rating, Label [fk7])
fk7: Label -> Location.Label

Fund = (Username [fk8], Name [fk9], Invested, Dt_Made)
fk8: Username -> Owner.Username
fk9: Name -> Restaurant.Name

Drone = (ID [fk10], Tag, Fuel, Capacity, Sales, Follow_Tag, Follow_ID [fk11], Pilot_Username [fk12], Hover [fk13], Homebase [fk 14])
fk10: ID -> Service.ID ID is non-null
fk11: Follow_Tag, Follow_ID -> Drone.Tag, Drone.ID,
fk12: Pilot_Username] -> Pilot.Username,
fk13: Hover -> Location.Label, Hover is non-null
fk14: Homebase -> Service.Label, Homebase is non-null

Ingredient = (Barcode, Name, Weight)

Contain = (Barcode [fk15], ID, Tag [fk16], Price, Quantity)
fk15: Barcode -> Ingredient.Barcode
fk16: ID, Tag -> Drone.ID, Drone.Tag

Works_for = (ID [fk17], Username [fk18])
fk17: ID -> Service.ID
fk18: Username -> Worker.Username

Experience vs. Experience
Experience in Employee is the number of months worked.
Experience in Pilot is the number of successful trips.

List of Assumptions

- License type is non-null "Each pilot must have a valid license type to signify that they have received the proper training to operate the drone safely"
- Drone.ID is non-null since every Drone has to be sponsored by a Service
- Drone.Fuel is greater than or equal to 0. Negative fuel doesn't make sense, but we do allow for a Drone to have no fuel at a given moment.
- Drone.Capacity is greater than 0 (since we assume a drone has to be be able to carry something)
- Drone.Sales can't be null. We set Sales to 0 by default. Assume Sales are positive.
- Drone.Hover is non-null, since every Drone has to be at a location
- Drove.Homebase is non-null, since every Drone has to have a homebase.
- Ingredient.Weight is greater than 0 (We assume every ingredient has a weight?)
- Contain.Price and Contain.Quantity are both greater than 0, for obvious reasons.

Unhandled Constraints

- Ensure all users are either owners or employees
- Ensure that an owner funds at least one restaurant
- Ensure that total number of packages doesn't exceed drone capacity
- Ensure that a service has at least one worker that is not temporarily unemployed
- Ensure that a service owns at least one drone
- Ensure that a drone is only owned by one service
- Ensure that managers of a service also work for that service
- Ensure that an employee at most manages one service
- Ensure that drones do not go to a new location unless they have enough fuel to return to home base
- Ensure that number of drones in a swarm do not exceed a location's space limit
- Ensure that a sale can only be made when a drone is located at that restaurant

- Ensure that the manager of a service has a higher salary than the other workers of that service
- Ensure that pilots with more successful trips have a higher salary than pilots with fewer
- Ensure that a pilot is only contractually hired at one service
- Ensure that no employee is both a worker and a pilot
- Ensure that a drone is controlled by only one pilot at a time
- Ensure that a swarm has only one leader drone
- Ensure that drones in a swarm either move to the same location or stay where they are
- Ensure that a drone is carrying at least one package on a delivery