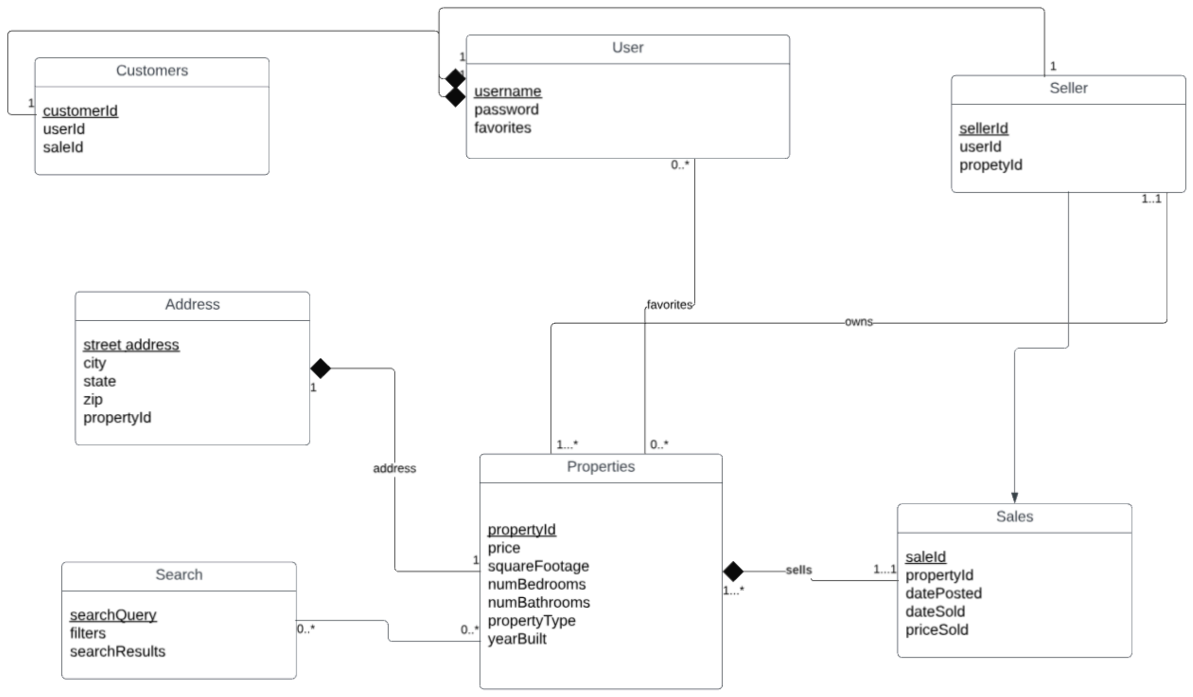

Assumptions

- We assume that in order for there to be Customers there must also be listings up for sale
- We assume that there are properties already on the website in order for search to work
- We assume that there could be multiple houses that are being sold by one account
- We assume that users can be either customers OR sellers

User is going to be one of our main entities as it has the main attributes that are required to make an account on our website. We chose to have both customers and sellers both as entities instead of attributes because customers and sellers have a different list of things they can do on our website. We thought that this would make things easier when developing. Properties is our other main entity as it has the main attributes for when a user makes a listing on our website. We decided to have addresses as an entity instead of an attribute because it is something that all properties have and also since there are multiple components to a singular address. Sales is also an entity because there is a decent amount of information that we would like to list about a sale so we need to make it an entity to store all the attributes we would like to display. This is similar to our search feature where there are multiple things that could be searched for. We also made it an entity so it could interact with the properties entity as that is what we want to search for. The user entity is linked to the customers, sellers, and properties entities. This is because a user can either be a customer, seller, or both and a user is most likely going to look at properties and all the other entities that connect with it. The seller entity is connected to sales as we infer that if a user has a “seller” account, they would like to view statistics related to sales of either their own house or houses like theirs so they know what the market is like and use the data to possibly set a price for their own house. The properties attribute is linked to the user, search, and sales attributes. All of the information that the users want come from the properties attribute hence the link. The search attribute is linked to properties because we intend for our search feature to work for all the properties that we have listed on our website. The sales attribute is linked to properties because all of the sales information is supposed to come directly from the properties.

Normalizing Database

We chose to go with 3NF because adhering to stricter constraints for BCNF would not make much of a difference for our database. Currently, almost all of our tables have attributes based on their primary keys. This means that we don't have anything that would cause redundancies, thus meaning there isn't much of a reason to use BCNF and make the queries overcomplicated.

- User: username → password, favorites
- Customers: customerid → userId, saleId
- Sales: saleId → propertyId, datePosted, dateSold, priceSold
- Properties: propertyId → price, squareFootage, numBedrooms, numBathrooms, propertyType, yearBuilt
- Search: searchQuery → filters, searchResults
- Address: street_address → city, state, zip, propertyId
- Seller: sellerId → userId, propertyId

Relational Schema

- User(userId INT [PK], userName VARCHAR(50), password VARCHAR(50), favorites VARCHAR(255))
- Seller(userId INT [FK to User.userId], sellerId INT [PK], propertyId INT [FK to Properties.propertyId])
- Customer(userId INT [FK to User.userId], customerId INT [PK], propertyId INT [FK to Properties.propertyId])
- Address(propertyId INT [PK], streetAddress VARCHAR(100), city VARCHAR(50), state CHAR(2), zip INT)
- Properties(propertyId INT [PK], price DECIMAL(10,2), squareFootage INT, numBedrooms INT, numBathrooms INT, propertyType VARCHAR(50), yearBuilt INT)
- Search(searchQuery VARCHAR(255), filters VARCHAR(255), searchResults VARCHAR(255))
- Sales(saleId INT [PK], propertyId INT [FK to Properties.propertyId], datePosted DATE, dateSold DATE, priceSold DECIMAL(10,2))