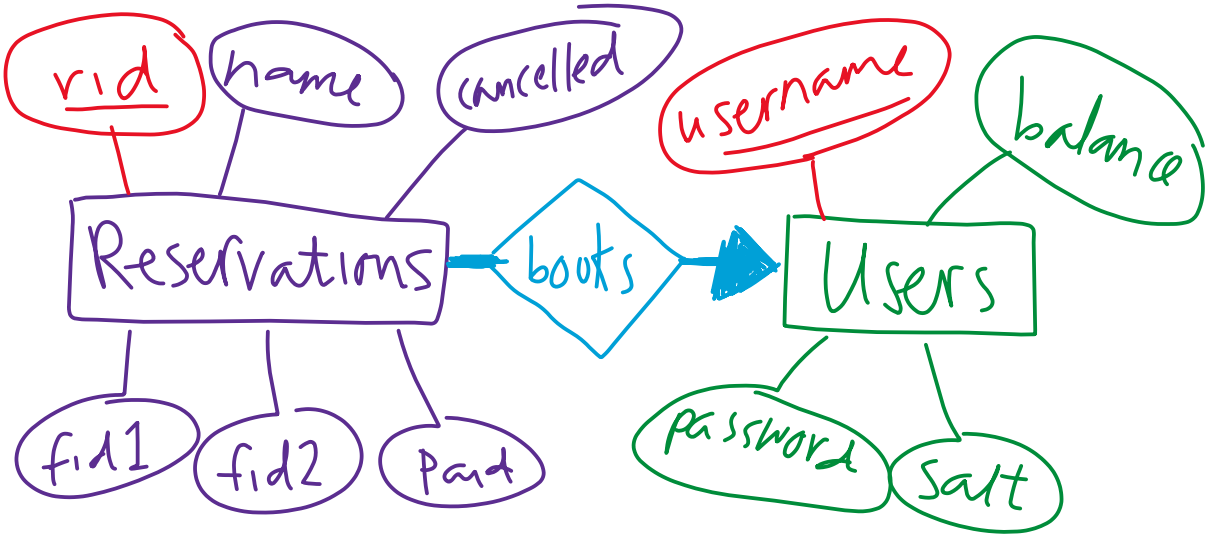
Visual Reference:



I decided to have Users and Reservations as new tables as each access of the program has a chance to affect other Users and Reservations. Storing it locally would make it so accesses are independent and would not influence each other, thus the two tables must persist on Azure. The fields were created with access in-mind: For Users, we would not want a user's direct credentials, so we have a hashed password and the salt stored to re-access in a secure way. Since each user has a DISTINCT username, that tells me that a user's username should be the primary key. For reservations to functionally work with our methods, it should have an id to save the DISTINCT id for a reservation. No reservation has the same ID, so I also knew this should be a primary key. Knowing whether reservations are cancelled and cancelling them would impact capacity checks, so that is a field I must store for functional use. Reservations also fall under a user, thus we keep track of that as well so we are able to gather reservations for a user. By the definition of a "reservation", flight(s) must be included and a good way to track them is by their unique flight ids as our fields. For both Users and Reservations as it is seen, I have put fields that would specifically affect other clients running Query.java and need to be up-to-date in access.

On the contrary, data that is only in-memory were unique to ONE client running Query.java. A primary example of this is of the "search" history that persists only on one user's end directly in their session and ONLY in their session. The second instance of local storage is tracking login since we allow only the same user to be logged in. These accesses create local copies that only needs to be temporary and for that one client. Additional examples if needed of this are any of the local, private variables used in methods that usually save private/individual data (personalized to them from a query) just to display.
Hope this made sense!! Thank you for listening. :)