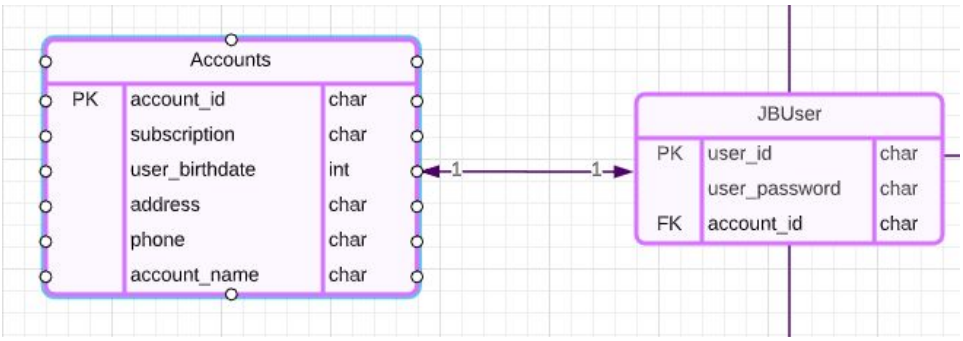
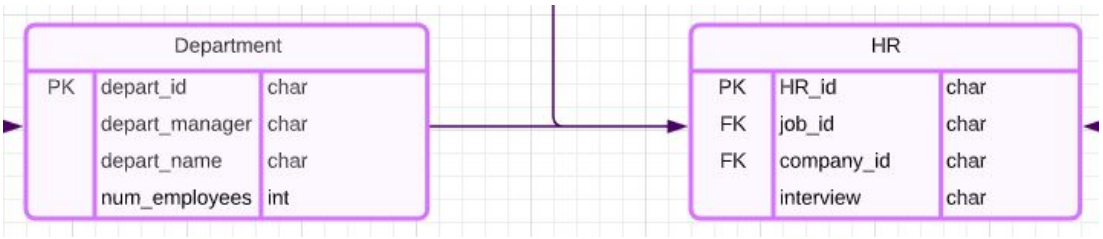
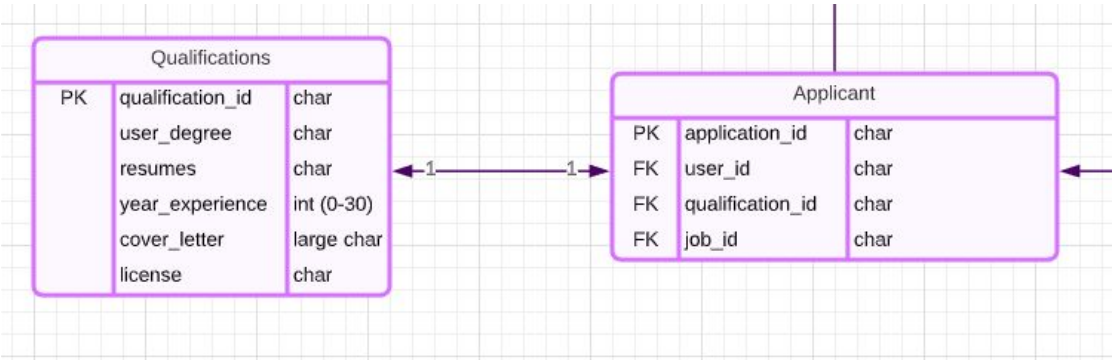
After the completion of our design for our DBMS system, we arrived at the step of normalizing our design. In order to normalize our design, we first outlined the functional dependencies in the system. For example, our one-to-one relationship between JBUser and Accounts.



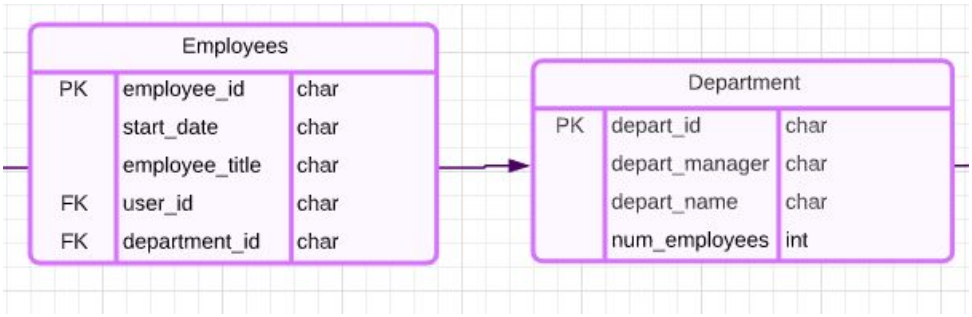
A user can have one account, and each account may only have one user. Therefore, the functional dependency that would express this relationship would be, account_id → user_id and user_id →account_id. Another example is our one-to-one relationship between Department and HR.



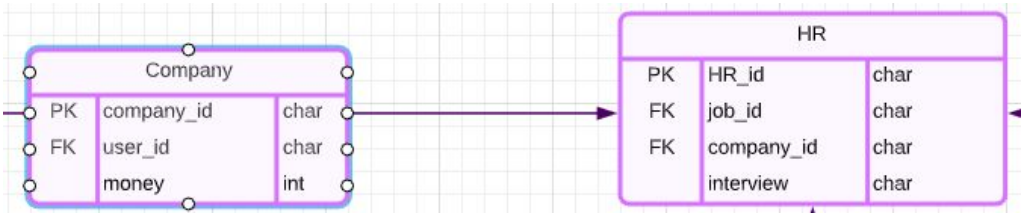
A Department can have one HR, and each HR may only have one Department. Therefore, the functional dependency that would express this relationship would be, depart_id → HR_id and HR_id → depart_id. One more Example would be our one-to-one relationship between Qualifications and Applicant.



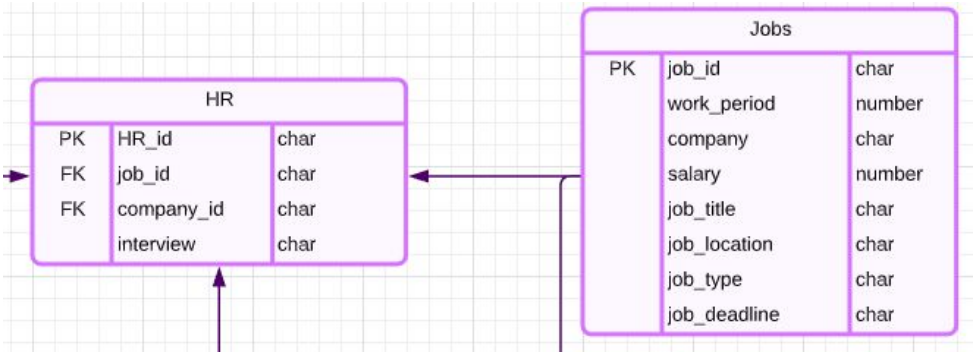
An Applicant can have one set of qualifications, and each set of qualifications may only have one Applicant. Therefore, the functional dependency that expresses this relationship would be, application_id → qualification_id and qualification_id →application_id. We can also look at an example between the one-to-many relationship between our Department and Employee entity.



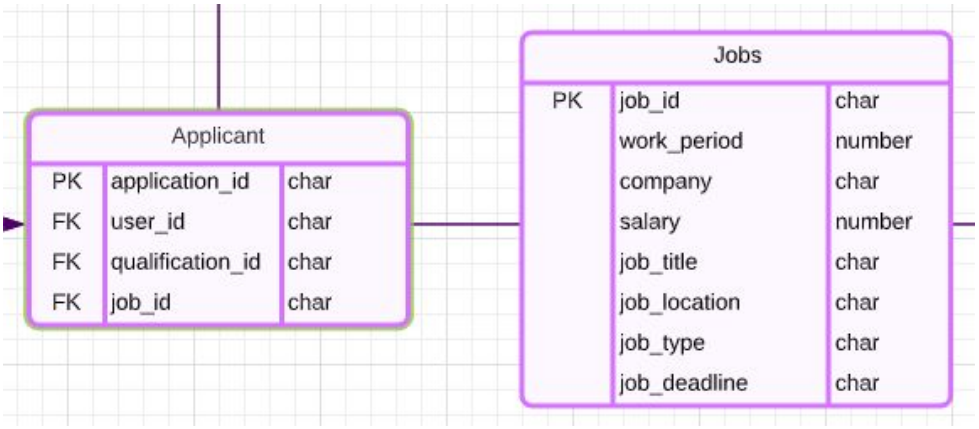
This functional dependency shows how the department can have several employees, but an employee can only have one department. Due to this fact, the functional dependency that expresses this relationship would be, employee_id → depart_id. More examples of the functional dependency of one-to-many relationship are shown below.



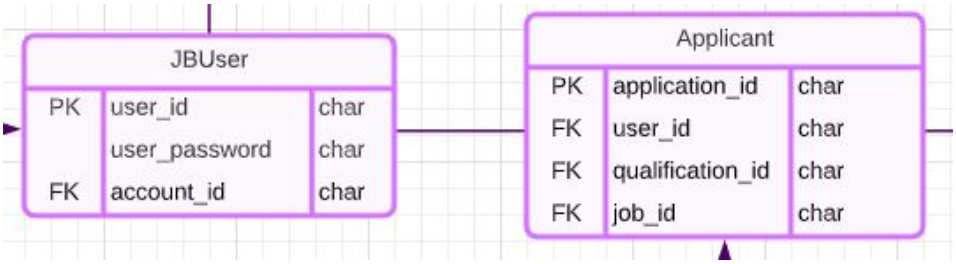
This functional dependency shows how HR can have service to several Companies, but a Company can only communicate with one HR department. Due to this fact, the functional dependency that expresses this relationship would be, company_id → HR_id.



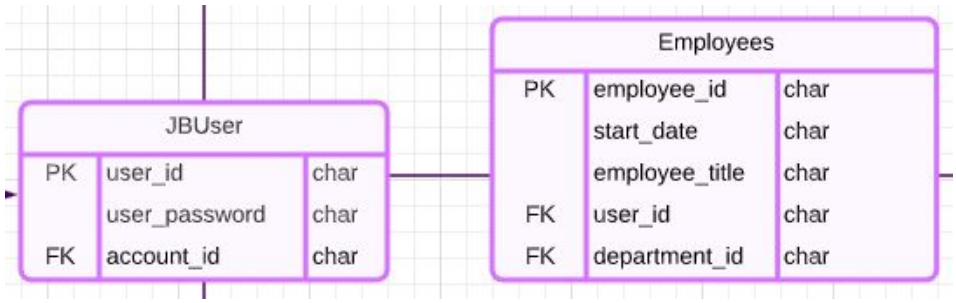
This functional dependency shows how the HR can manage several jobs, but a job can only be managed by one HR. Due to this fact, the functional dependency that expresses this relationship would be, job_id → HR_id. In the case that there is a many-to-many relationship, there is no functional dependency. For example, the relationship between Applicant and Jobs.



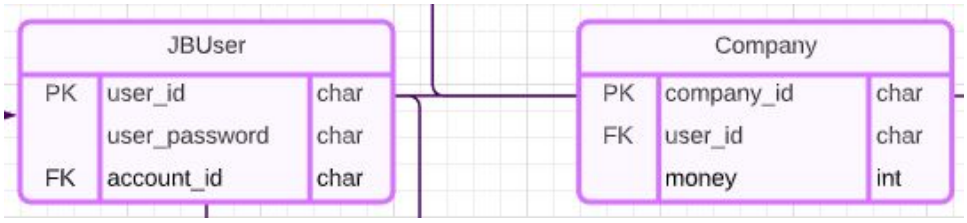
As we can see, many applicants can apply to one job, however one applicant can apply to many jobs. Therefore we can see that there are no functional dependencies to capture here.



As we can see, many users can be an applicant, however all applicants are a user. Therefore we can see that there are no functional dependencies to capture here.



As we can see, many users can be an employee, however all employees are a user. Therefore we can see that there are no functional dependencies to capture here.



As we can see, many users can be a company, however all companies are users. Therefore we can see that there are no functional dependencies to capture here. By identifying all the functional dependencies, we were able to optimize our DBMS system by reducing the redundancies in our system.