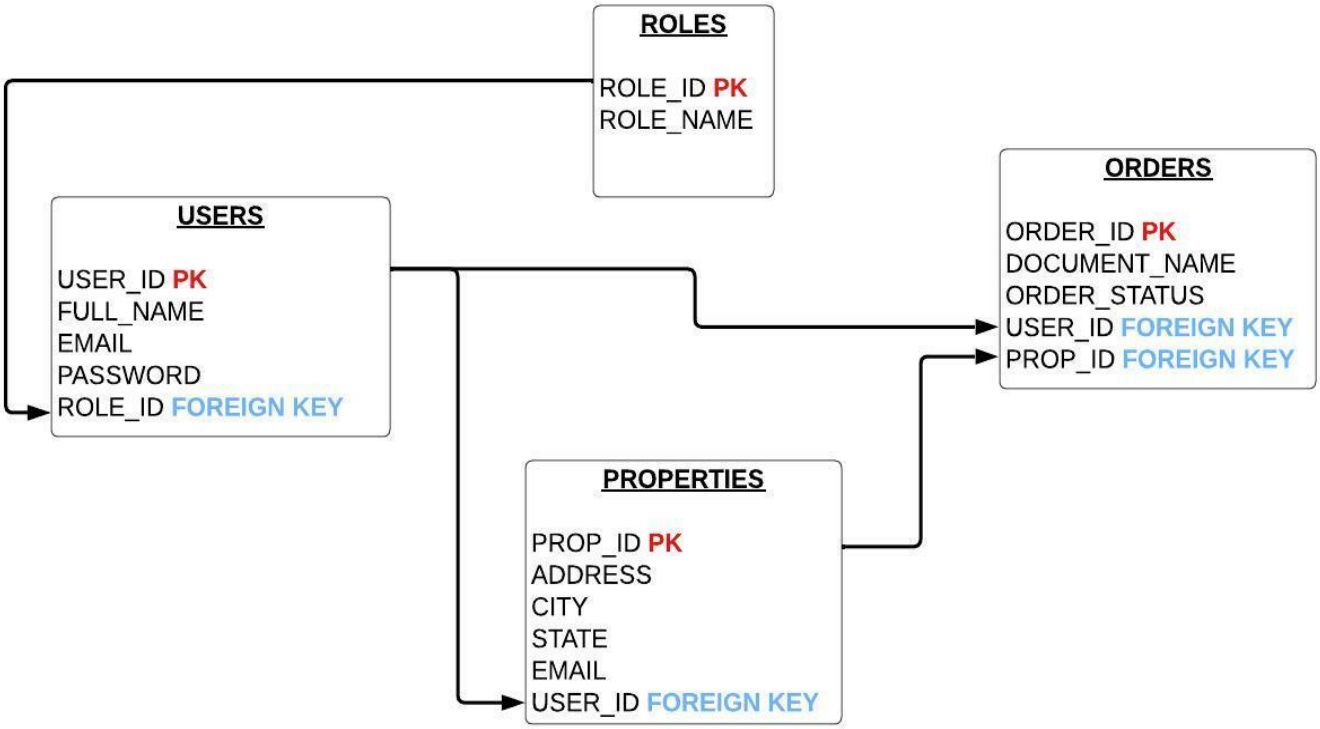
Database Schema



The system that needs to be built consists of 3 main elements which are the users, properties, and orders. Since there are two types of users, it is also important to specify each user's role in the system, so that is why we also need the roles table. Based on their roles the users will perform certain actions in the system which are submitting orders and adding properties information. That means that there has to be a relationship between the user's table and the properties table for those who are administrators in the system and another relationship between the user's table and the orders table for those that are clients in the system. In order to identify which property, the document/service is requested for, there should be a relationship between the orders table and the properties table.

The database schema I have provided would be the best solution because it stores all of the needed information in as few objects as possible. This makes the database clean, consistent, and easy to understand. It means that the database will reflect the real structure of the problem in its most simple way. This schema avoids redundancy by not storing duplicates and increases efficiency by storing only the data that is required. The relationships between objects make the database safe as long as the relationships between objects exist. Those relationships make it possible to access the expected data over time and the simple structure gives the administrator easy access to the database for retrieving any kind of needed data for this type of system. To conclude, this solution fits with the given system by storing all the needed data in a simple, efficient, and accessible way.

Bonus Questions

1. We want to add functionality for scheduling orders in the system. After the clients fill in the necessary information to create an order, instead of clicking “Submit”, they can click “Schedule Order”, select the date for when they want the order to be created, and when the selected date arrives, the order will be created. Please write a solution you think would best fit here in terms of database schema (adding tables, new fields, etc.).

In order to add this functionality, we need to create a new database object (table) for determining the order type. The new table will have two fields: order_type_id and order_type_name. The order_type_id (int/integer, PK) name will be the primary key and it will store the identifier of the order type. The order_type_name field(varchar/string) will store the descriptive name of the order type which can be a scheduled order or an actual order. This table would be related to the orders table. With that being said, there is a need for the type_id field in the orders table as a foreign key. We also must add two fields to store the order date. One will be created_at (timestamp) and the other one scheduled_date. The created field will always store data, unlike the scheduled_date field which will only store data when the type of order is a scheduled order.

2. What would you use to check if the scheduled orders should be converted to actual orders in a real running application? (Instead of manually going to the database every day and checking the scheduled order dates)

In order to avoid this repetitive manual task, I would use python automation. This means that instead of going to check the database manually, a python function can do it for us repeatedly. The function should consist of a query that updates the order type after comparing the order scheduled date with the actual date and making sure that both dates match. When the date of the actual day matches the scheduled orders, then when the administrators check the records of the database, the pre-scheduled orders for that day will show up as actual orders, which would let the administrators know that is time to proceed with the order. We need to run this function every day. In order to run the script every day the one way would be to use the *cron* scheduler. To conclude, for completing this task automatically all we need is the python script that checks and updates the database based on the query that is written inside the function and to schedule the script to run every day.