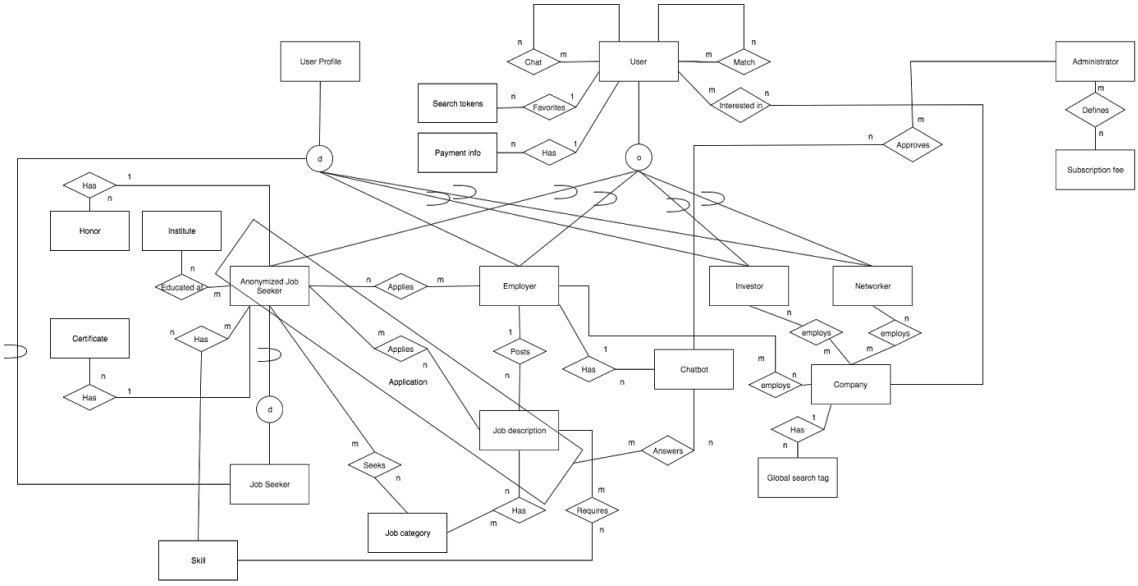
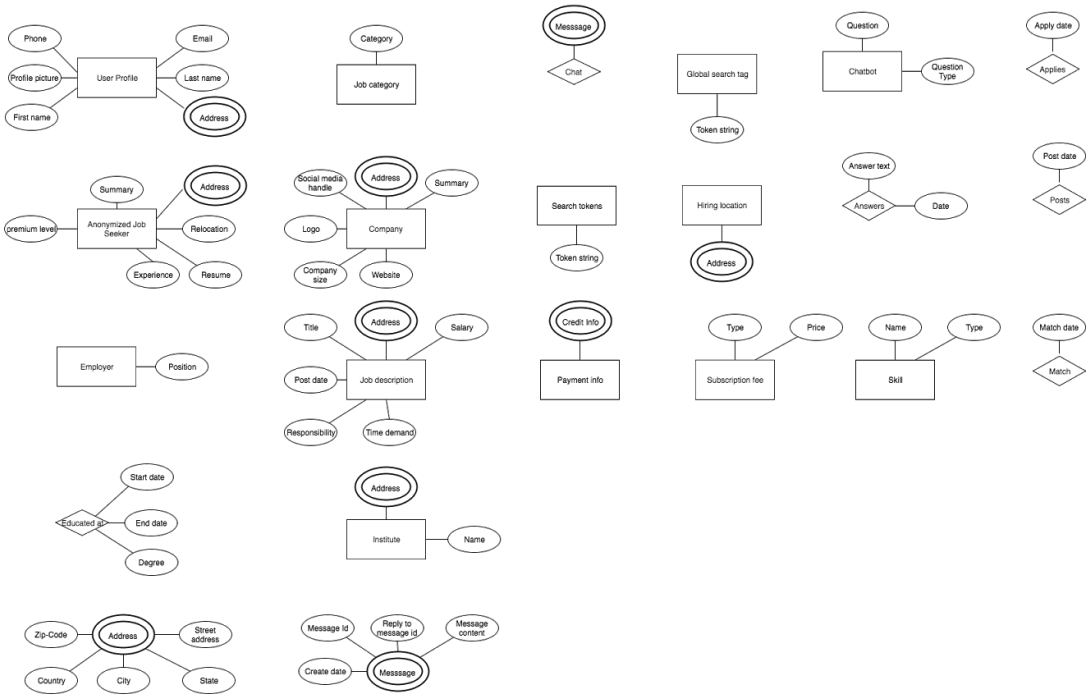
Assignment 2: Using a Relational Database Management System named PostgreSQL

Due date is 11:59 pm Friday February 16th.

Objective: This assignment requires you to implement a collection of tables using PostgreSQL, load the tables with data, write a few SQL queries, and optimize these queries by authoring index structures (or materialized views). The tables correspond to the ER diagram for the first assignment, see Figure 1.

Description: Recall the description of Assignment 1. Here is one possible ER diagram:





Step 1: Implement the following tables using PostgreSQL and populate them with sample data set provided at
https://drive.google.com/file/d/1L2-C8cMR-Sl_7sCQjiWoMPRJP2V6XlTw/view?usp=sharing
This data is generated randomly based on public data sets. It consists of 100,000 users (90,000 job seekers and 10,000 employers), more than 200,000 messages, 10,000 job listings and 10,000 job applications. One CSV file per table. CSV file names are the same as table names.

Tables are as below. Primary key(s) of each table is underlined while its foreign keys are in bold italic font.

1. User(userId : bigint, phone : varchar, email : varchar, profile picture url: varchar, first name : varchar, last name : varchar, street address : varchar, city : varchar, state : varchar, country : varchar, zipcode : varchar)
2. Message(messageId : bigint, ***fromUserId : bigint***, ***toUserId : bigint***, replyMessageId : bigint, message : varchar, creationDate : date)
3. JobSeeker(***userId : bigint,*** summary : varchar, is_relocation_ok : bool, experience : varchar, premium level : int)
4. Institute(instituteId : bigint, name : varchar)
5. JobSeekerEducation(***userId : bigint, instituteId : bigint,*** startDate : date, endDate : date, degree : varchar)
6. Skill(skillId : bigint, skillName : varchar)
7. JobSeekerSkill(***userId : bigint, skillId : bigint***)
8. JobCategory(categoryId : bigint, categoryName : varchar)
9. JobSeekerCategory(***userId : bigint, categoryId : bigint***)
10. JobSeekerHonor(***userId : bigint,*** honorName : varchar)
11. JobSeekerCertificate(***userId : bigint,*** certificateName : varchar)
12. Employer(***userId : bigint,*** position : varchar)
13. JobList(jobId : bigint, ***userId : bigint,*** title : varchar, salary : varchar, postDate : date, responsibility : varchar, time demand : varchar)
14. JobListCategory(***jobId : bigint, categoryId : bigint***)
15. Chatbot(questionId : bigint, ***userId : bigint,*** question : varchar)
16. Company(companyId : bigint, companyName : varchar, social media handle : varchar, summary : varchar, logo : varchar, company size : varchar, website : varchar)
17. JobApplication(applicationId : bigint, ***jobSeekerId : bigint, jobId : bigint***, applyDate : date)
18. ChatbotAnswer(***applicationId : bigint, questionId : bigint***, answer : varchar)
19. Employment(***companyId : bigint, userId : bigint***)

Step 2: Once you have created the tables and populated them with data, write SQL queries for each of the following:

Query-1: Show Employer 90109 (id=90109)'s five recent messages received from other employers. For each message, retrieve the sender's first name, last name, profile image URL, message content, and message date. Your query should produce data in these columns. Your query should be able to run against another employer by simply replacing employerId 90109.

First Name	Last Name	Profile Image URL	Message content	Message date
...	...	...	...	...

Query-2: Show Employer 90196's top 5 job listings that receive the highest number of job applications. For each job listing, show all columns in the JobList table except jobId and userId. Your query should produce data in these columns. Your query should be able to run against another employer by simply replacing employerId 90196.

Title	Salary	Post Date	Responsibility	Time Demand
...	...	...	...	...

Query-3: Show Employer 90196's job applicants for job listing with jobId = 4401. For each job applicant show their chatbot answers associated with their corresponding questions, summary, experience, educations, skills, honors and certificates. With multiple skills for an applicant, concatenate these skills and show as one value for the skill column. The same applies for other fields. Your query should produce data in these columns. Your query should be able to run against another job listing by simply replacing jobId 4401.

Summary	Experience	Education Institute names	Skill names	Honors	Certificates	Questions	Answers
...	...	...	...	...	...	...	...

Query-4: Show user 1's messages.

Show all messages of a user group by message threads. For each message, show both user's profile picture URL, first name, last name, message content and message date. A message thread starts with a record that has replyMessageId equals -1. Your query should produce data in these columns. Your query should be able to run against another user by simply replacing user 1. (HINT: use recursive sql)

Message Id	Sender First Name	Sender Last Name	Sender Profile Image URL	ReplyId	Receiver First Name	Receiver Last Name	Receiver Profile Image URL	Message content	Message date
...	...	...	...	...	...	...	...	...	...

Query-5: Retrieve top 5 companies with more than 100 job applications from 2015 to 2018 sorted by the number of received job applications in descending order. Your query should produce data in these columns.

Company Name	Company URL	Number of applications received
...	...	...

Query-6: Retrieve the top 5 market vacancy of job categories in 2017 sorted by market vacancy in descending order. The market vacancy of a job category is defined by (# of job listing in a category / # of job applications in a category). Your query should produce data in these columns.

Job category	Market vacancy
...	...

Step 3: Now, for each query, author indexes or materialized views to expedite the processing of that query.

Deliverable: A pdf file containing the dump of your PostgreSQL schema, your physical design (i.e., choice of index structures and materialized views), the SQL query for each of the six queries along with their number of rows retrieved and execution time, and a screen shot of the results retrieved by each query.

Possible points is 100.

Due date is 11:59 pm Friday February 16th.

Notes:
Download PostgreSQL from https://www.postgresql.org/download/

We have a cluster of computers available for this assignment. Send an e-mail to the TA and he will assign you a windows box.