Hospital Database

COMP20070 MYSQL DB ASSIGNMENT (2023-24)

Sneha Chhipa
STUDENT NUMBER: 22320616

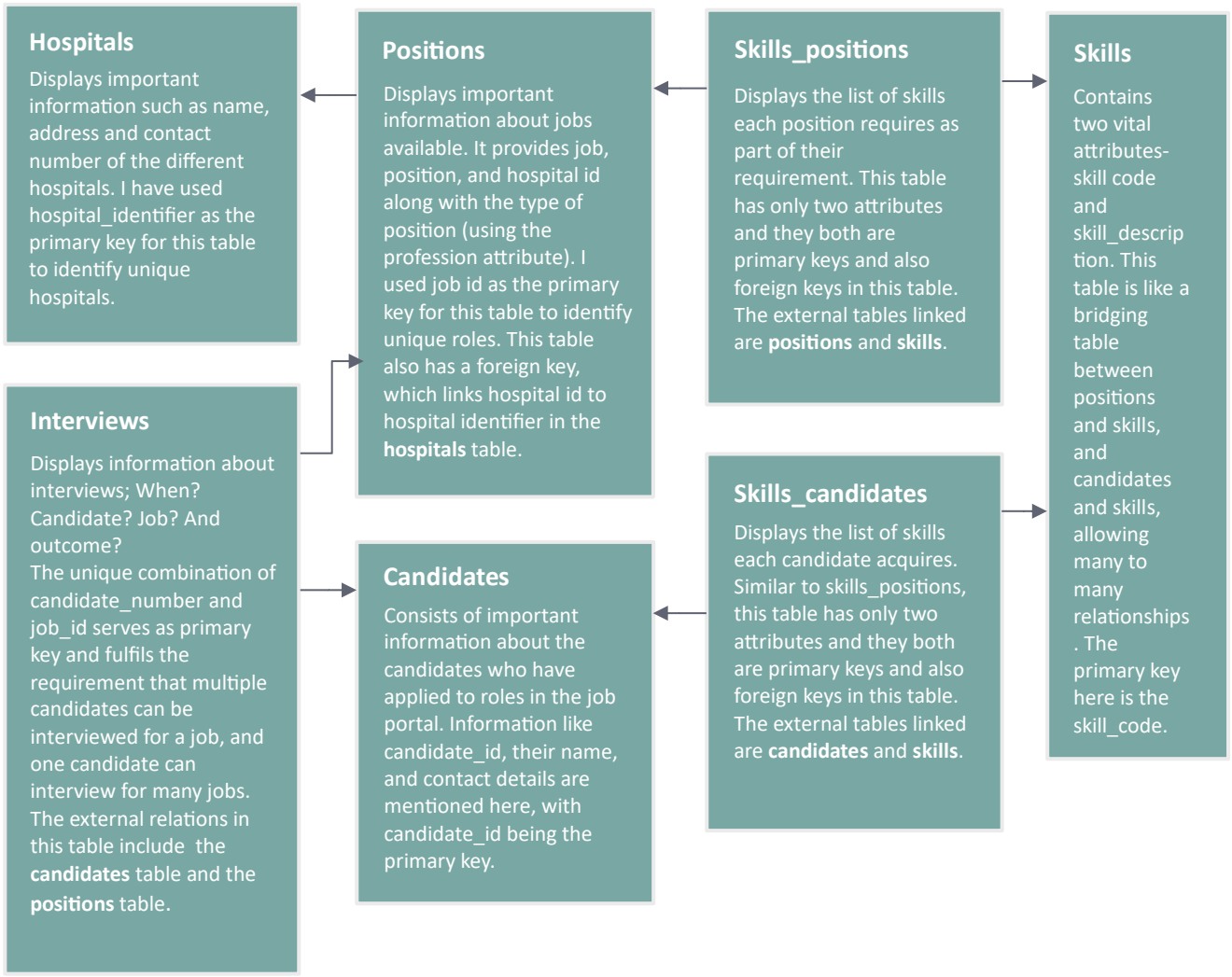
used Mac to complete this assignment.

A description of my database….

The name of my database is "chhipa22320616" and it represents the information as required by the scenario mentioned in the assignment specification. The scenario is the following: *"Hospitals advertise positions which require specific skills (e.g., nursing, administrative, etc.). Candidates may be invited to interviews for the positions."* Thus, my database manages the information about hospital job postings and the candidates who are applying for the different postings. To represent this information effectively, I have created seven tables in the database where each table contains information related to a specific element of the scenario. I have broken down the scenario in the following elements: hospitals, positions, skills, skills_positions, candidates, skills_candidates, and interview. Each element has its own table, each table has primary keys, and some tables are also linked together using foreign keys.

The following is a detailed description of the different tables I have in my database:

hospitals (hospital_identifier, hospital_name, address, telephone_number)
positions (job_id, position_identifier, hospital_identifier, profession)
skills (skill_code, skill_description)
skills_positions (job_id, skill_id)
candidates (candidate_identifier, firstname, surname, address, telephone_number)
skills_candidates (candidate_id, skill_code)
interviews (candidate_number, job_code, interview_date, outcome)



Assumptions/Additions made….

In order to create a database as per the specification requirement, there were a few assumptions I had to make. I also added some minor concepts, to successfully achieve the database design I planned.

- In relation to the positions, I assumed that each position requires four to five different skills.
- The variable types I used are INT (for id numbers), VARCHAR (for strings) and DATE (for interview dates).

- According to the assignment specification, each position must have a position identifier and each hospital must have a hospital identifier. I assume that a unique position identifier may be associated with two different positions in two different hospitals. Therefore, to avoid this ambiguity and to also make the identification of a position easier, I introduced a new attribute called job_id. This attribute serves as a primary key in the positions table and represents the different unique positions.

- In the positions table, I renamed the attribute "type of position" as "profession". This attribute describes the role title of the position.

- To establish a many to many relationship between *positions* ⇔*skills_positions*, and *candidates* ⇔ *skills_candidates*, I created a new table called **skills**. This table represents each skill with a unique code, and this code is then used the in the positions table and candidates table as an id to map to the skill in the skills table. This way also reduces the redundancy of typing the different skills multiple times in the database.

- In the specification it is stated that the database should include whether a candidate has been successful in receiving an offer for a position. To incorporate this feature in my database, I added an attribute called *outcome*. This represents the status of the application process of a candidate. If their interview has already been completed, then the value of *outcome* is either *Pass* or *Fail*. If their interview is yet to happen, then the value of *outcome* is kept as *InProgress*.

Reaction Policies used….

In my database, the tables I created are linked together using foreign keys and external relations. When implementing interrelation constraints, it is important to wisely choose the reaction policies. These policies guides the database management system the action it should take in the inner table when an update/delete occurs in the external referenced table.

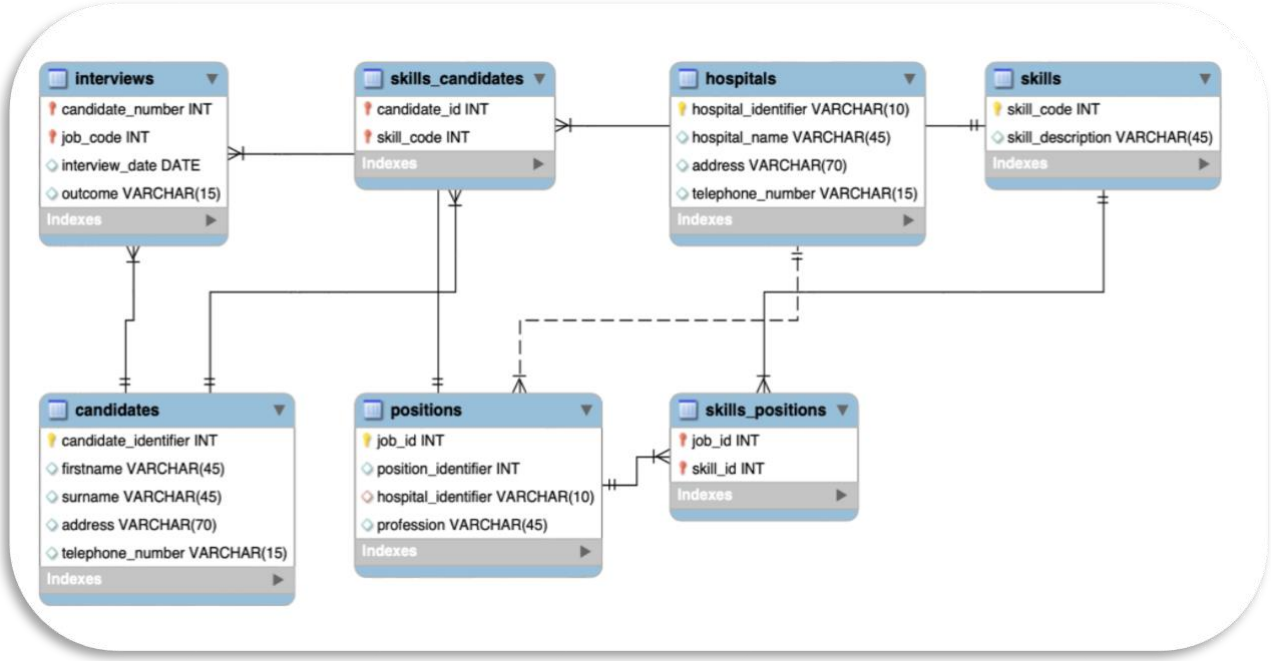
I used the following reaction policy for all of the foreign key constraints in my database:

On UPDATE -> CASCADE
I used this policy as it allows the update/change to reflect in the inner/root table.

On DELETE -> RESTRICT
I used this policy because it restricts the deletion of the tuple in the referenced table. I did not use SET NULL as a reaction policy because the attributes which were foreign keys, also served as a

primary key of the table. Therefore, setting a primary key as null is not accepted by the database management system.

The Entity-Relationship (ER) diagram…



The Operating System used…

used Mac to complete this assignment.