COMP20270, Assignment

Brian Manning,

Operating System: Windows 10

Short Description of my Database:

I have created a database called manning17324576 that could be used for a recruitment system for multiple hospitals to record information relating to hospitals, candidates, positions, interviews, and job offers. The database consists of several relationships which I have detailed better under the Assumptions below. My database consists of 8 tables which you can see below in the ER model.

Assumptions and Additions Made:

The way that I would imagine this database would be used would be as follows:

1. All hospitals using the system would be added to the hospitals table along with their addresses to the addresses table. Hospitals would then be added as needed in the future.
2. A hospital creates a new position in the positions table and add skills for that position into the skills table. The link would then be made between the position and skills using the position_skills table.
3. Candidates who apply for the position are added to the candidates table. Their addresses are added to the addresses table. Their skills are added to the skills table if they do not already exist in it and the skills are linked using the candidate_skills table. The ID of the position they are currently applying for is also added to the database. If a candidate chooses to apply for another position in the future this position field can then be updated.
4. Certain applications are chosen for interviews. These interviews are added to the interviews table.
5. Certain interviewees are then chosen to be hired. The accepted_status is changed from 0 to 1 to represent the candidate having been offered the position. When the candidate accepts the position, this value would then be changed to 2 to represent this.

Other Assumptions & Additions:

- I assumed that a candidate may apply for more than one position in the future. For this reason, I have the position ID column in both the candidates table and the interviews table. The position ID column in the candidates table represents the position the candidate is currently applying for while the column in the interviews table represents the position that they interviewed for. This allows for more flexibility in the future and would help stop duplication of candidate data
- I assumed that addresses could be stored in the same format for both hospitals and candidates and for this reason I created one table called addresses where hospital and candidates can store their addresses. Each candidate/hospital are assigned an address id, which links them to their address. I made adding addresses to this table easy by implementing adding addresses into the stored procedures for adding candidates and hospitals. I felt it was necessary to store granular information on addresses to allow for filtering by county, country etc. I would prefer the user to use the stored procedure where they are also adding the address at the same time to prevent errors.
- A hospital can request many interviews for a position using the interviews table. They simply must record the relevant ID of the position and the candidate along with the further

information. One candidate can be invited to many interviews in the same fashion – just adding another entry to the interviews table.

- I decided to store whether a candidate had been offered a position and whether they accepted it in the interviews table. This would allow one candidate to interview for many positions in the future and accept each one.
- I added interview date, time, location, and notes to the interviews table. This allows the interviewer to store information regarding a specific interview. They can add notes before and after the interview and can change the accepted_status as needed.

Reaction Policies:

Throughout the tables in my database, I choose to use the following reaction policies across all of my Foreign Keys:

1. **On Update:** Cascade – I choose to use cascade here to ensure that once data is changed in one table of the database, it would change throughout the database. This would ensure the quality of the data returned by the application using the database and allow for changes to be made to IDs etc. where needed. I used this policy throughout my database to allow for changes to be made to IDs where needed.
2. **On Delete:** Restrict – I choose to use restrict here to ensure that data is not deleted from a database unless explicitly needed. This will prevent possible data loss from accidental deletion. If information relating to a specific hospital, candidate etc needs to be deleted this will ultimately make it more manual but will ensure that data is not deleted throughout the database accidentally.

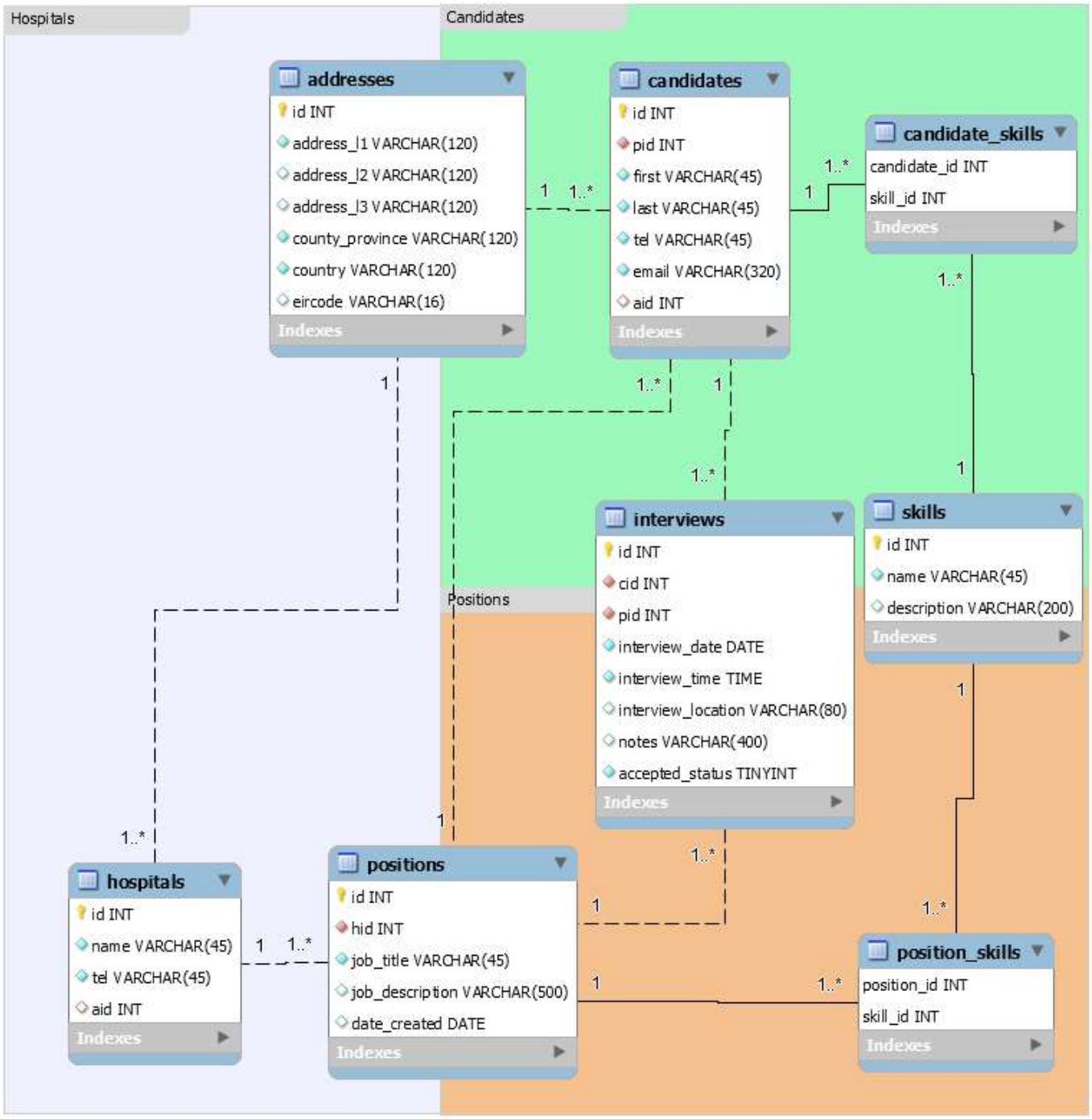
COMP20270, Assignment

Brian Manning,

Operating System: Windows 10

Entity-Relationship Diagram:

Below you can see the ER Diagram of my database. I used colours to help illustrate the relationships.



COMP20270, Assignment

Brian Manning,

Operating System: Windows 10

Guide to the Stored Procedures

a. Adding data to the tables (in alphabetical order)

Each query below that adds an autoincrementing ID returns that ID to the user so they can use it in their next query. Eg adding a candidate and then adding their skills.

1. add_address_return_aid: This adds an address to the address table and then returns the id of that address to then be used when adding a candidate or a hospital
2. add_candidate_with_aid: This adds a new candidate to the candidates table along with their address id
3. add_candidate_with_address: This adds a candidate's address to the address table first, returns that address id and then adds the candidate details along with the address id to the candidates table
4. add_candidate_skill: This adds a candidate id along with the id of a skill to the candidate_skills table.
5. add_hospital_with_aid: This adds a new hospital to the hospitals table along with their address id
6. add_hospital_with_address: This adds a hospital's address to the address table first, returns that address id and then adds the hospital details along with the address id to the hospitals table
7. add_interview: This adds the details of an interview to the interviews table
8. add_position: This adds positon details to the positions table
9. add_position_skill: This adds one of the required skill ids of a position along with the position id to the position_skills table
10. add_skill_return_skill_id: This adds a new skill to the skills table

Required Stored Procedures:

These are numbered as they were asked in Step 4.

1. find_hospital_by_id
2. find_hospital_by_name: Note full name must be used for this.
3. find_candidates_by_surname: Note full surname must be used for this.
4. find_candidates_with_pos_skill
5. find_count_of_offered_cand: This counts both candidates that have been offered positions and not accepted it (1) and those that have been offered and accepted it (2)
6. find_pos_by_skill: Assume this meant that we would be searching by the name of the skill here rather than the ID.
7. find_pos_skill_nursing
8. find_pos_sort_hosp – For this, as my database allow multiple hospitals to request multiple positions, they are sorted first by the hospital advertising the position and then by the job title of that position
9. find_interview_by_date: DATE format YYYY-MM-DD
10. find_cand_by_inter_date: DATE format YYYY-MM-DD
11. find_cand_inter_twice