DESCRIPTION:

My database contains 7 tables:

1) *hospitaldetails*: Contains hospital information with 'HospitalID' as primary key.

2) *candidatedetails*: Contains candidate information with 'CandidateID' as primary key.

3) *positiondetails*: Contains information about each position on offer from each hospital. 'PositionID' is a primary key, and 'HospitalAdvertising' is a foreign key to 'HospitalID' in *hospitaldetails*.

4) *interviewdetails*: Contains interview information with 'InterviewPosition' as a foreign key to 'PositionID' in *positiondetails* and 'InterviewCandidate' as a foreign key to 'CandidateID' in *candidatedetails*.

5) *candidateskills*: Contains a many-to-many relationship, listing what skills each candidate possesses. 'SkillID' is a foreign key to 'SkillID' in *skilldetails*, and 'CandidateID' is a foreign key to 'CandidateID' in *candidatedetails*.

6) *positionskills*: Contains a many-to-many relationship, listing the skills required for each position offered. Similar functionality to *candidateskills*, but this time 'CandidateID' is substituted for 'PositionID', which is a foreign key to 'PositionID' in *positiondetails*.

7) *skilldetails*: Contains information liking the primary key 'SkillID' to its actual name in 'SkillName'.

ASSUMPTIONS/ADDITIONS:

I created the *candidateskills* and *positionskills* tables in order to list the skills each candidate possessed and the skills each position required. This avoided duplication and clutter in the *candidatedetails* and *positiondetails* tables, as it allowed me to have multiple data points per individual position/candidate. The 'Skills' attributes were removed from both tables as a result of this decision.

I created the *skilldetails* table in order to reduce ambiguity in naming conventions for various skills. For example, if a candidate lists their skill as 'Administrator', and the job position is looking for the term 'Administrative', there wouldn't be a match if there was a foreign key in place. By grouping these terms into IDs with the *skilldetails* table, this issue is largely avoided. It also saves time since the names would only have to be inputted initially, and after that their associated number would be used.

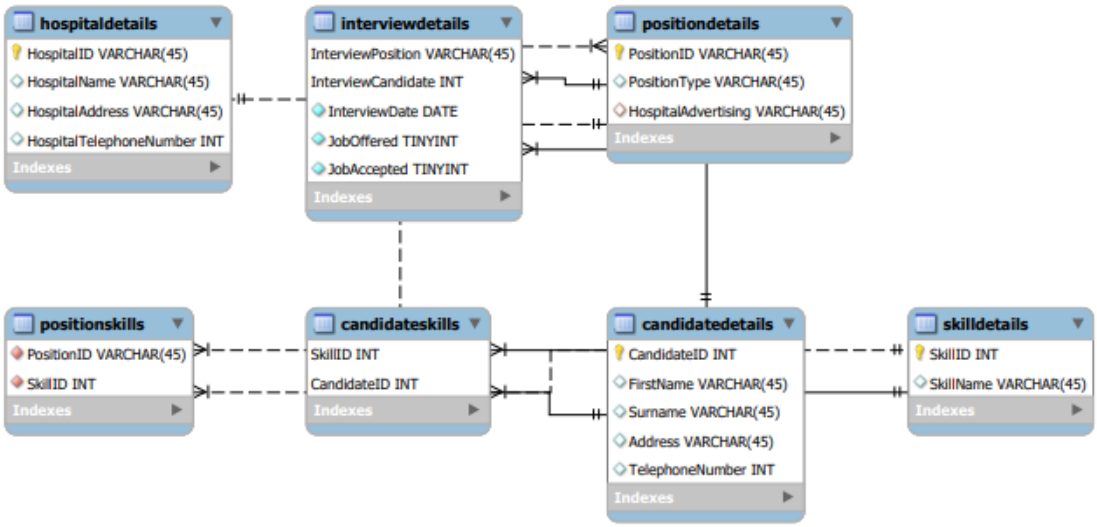
Since the table *interviewdetails* wasn't explicitly described in the assignment handout, I used the attributes 'InterviewPosition' and 'InterviewCandidate' in order to link candidates interviewing for positions via foreign keys. Additionally, I used the data type 'DATE' for 'InterviewDate', as it provided a non-ambiguous date format. I also used 'TINYINT' which is a Boolean depicting whether the candidate was offered or accepted the position. In this case 1=True, and 0=False. I decided to include the 'JobAccepted' attribute since it would be valuable to see whether the candidate accepted the offer, not just the fact that they were offered the position in the first place.

In regards to Step 4, I made a number of assumptions when completing the questions due to some ambiguity in the questions' wording. For the most part, when I saw the word 'given' I assumed a parametric query was warranted. For (10), I interpreted the words 'specific date' to mean a date that is baked into the query (non-parametric), therefore I chose the arbitrary date '2020-10-05' to illustrate the question. Additionally, I interpreted the word 'only' to mean that the query would be looking for candidates who only had one interview on the database.

In order for ease of use of the parametric queries I created, I have prepared a few example queries to try out:

1) BOS
2) Boston Hospital
3) Murphy
4) BOS001
5) Not parametric
6) Nursing
7) Not parametric
8) Not parametric
9) 2020-10-22
10) Not parametric
11) Not parametric

ENTITY-RELATIONSHIP DIAGRAM:



REACTION POLICIES:

I used ‘CASCADE’ for both ‘Update’ and ‘Delete’ for all of my foreign keys. I believe this makes for an agile database that allows for old/irrelevant data to be rid of, and it also makes updating primary keys much less of a headache since all associated data points will change accordingly. While there are dangers to this approach since the cascade option can create a snowball effect of errors, I believe that it is worth it in exchange of an easily modifiable database, which a hospital’s interview system will definitely need with the amount of candidates and positions on offer at full scale.

OPERATING SYSTEM:

Windows 10