COMP20070 MySQL DB assignment

Database Name : madkour19724421

Tables:

1. **Candidate -** Has the basic information of the candidates which includes the following attributes:
    a. idcandidate - int (PRIMARY KEY)
    b. Firstname - varchar
    c. Surname - varchar
    d. Address - varchar
    e. Telephone number - decimal

2. **Department -** Has the basic information of the departments which includes the following attributes:
    a. Iddepartment - int (PRIMARY KEY)
    b. Department name - varchar
    c. Address - varchar
    d. Telephone number

3. **Position -** Represents various positions that the company can have. This table has the following attributes:
    a. Idposition - int (PRIMARY KEY)
    b. Position type - varchar

4. **Skills -** Includes various skills and has many-to-many relationship with both **Position and Candidate tables.**
   i.e, each candidate can have various skills, and the same skill can be possessed by various candidates.
   Ex: **Skills**: Problem Solving, Management skills.
   Position: A Manager or project lead of the Development department needs to have both Problem Solving and Management Skills.
   Candidate: A candidate applying for the above position might have both Problem Solving, Management skills in addition to many more skills.

   This table has the following attributes:
   a. Idskills - int (PRIMARY KEY)
   b. Skill_name - varchar

5. **CandidateSkills -** This table was the result of the many-to-many relationship between the Skills and Candidate tables. Therefore the primary keys of these two tables together form the primary key of the Candidate Skills table as shown below:
   a. candidateId - int (PRIMARY KEY) - FOREIGN KEY referencing idcandidate in the Candidate table.
   b. Skillid - int (PRIMARY KEY) - FOREIGN KEY referencing idskills in the Skills table.

6. **PositionSkills -** his table was the result of the many-to-many relationship between the Skills and Position tables. Therefore the

primary keys of these two tables together form the primary key of the Position Skills table as shown below:

c. Positionid - int (PRIMARY KEY) - FOREIGN KEY referencing idposition in the Position table.
d. Skillid - int (PRIMARY KEY) - FOREIGN KEY referencing idskills in the Skills table.

7. **PositionOfferedByDepartment -** This table was the result of the many-to-many relationship between **position** and **department** tables.

   I.e, a position can be offered by various departments(ex: supervisor, manager, executive) and each department can offer various positions.

   Ex: **Position**: Manager is a position offered by various departments like - Development department, Quality Testing department, Sales Department etc.

   Department: A development department can offer various positions in it: Manager, Supervisor, Software Engineer, Technical Lead, Project Manager, Intern etc.

   Therefore the primary keys of these two tables together form the primary key of the PositionOfferedByDepartment table as shown below:

   a. Positionid - int (PRIMARY KEY) - FOREIGN KEY referencing idposition in the Position table.
   b. Departmentid - int (PRIMARY KEY) - FOREIGN KEY referencing iddepartment in the Department table.

8. **Interview -** This is a weak entity whose corresponding strong entity is **Position.** Because if a position doesn't exist, there will be no interview. And here we have interviewDate and PositionId as the primary key as shown below:
    a. interviewDate - date (PRIMARY KEY)
    b. PositionId - int (PRIMARY KEY) -FOREIGN KEY referencinng idposition in the Position table.

9. **Invite -** A candidate is invited to an interview and he/she may or may not be hired. This information is stored in the invite table.
   This table is a relationship between the Interview and the Candidate table and has its own attribute - isHired. Hence forming this new table taking the primary keys of both Interview and Candidate tables as shown below:
    a. candidateid- int (PRIMARY KEY) - FOREIGN KEY referencinng idcandidate in the Candidate table.
    b. interviewForPosition - int (PRIMARY KEY) - FOREIGN KEY referencinng positionid in the Interview table.
    c. dateOfInterview - date (PRIMARY KEY) - FOREIGN KEY referencinng interviewDate in the Interview table.
    d. isHired - varchar - values: Yes or No

ASSUMPTIONS MADE:

All the assumptions and additions are already described in the table descriptions above. But have stressed the below points for clear understanding:

1. **A position can be offered by various departments(ex: supervisor, manager, executive) and each department can offer various positions.**
   Ex: **Position**: Manager is a position offered by various departments like - Development department, Quality Testing department, Sales Department etc.
   Department: A development department can offer various positions in it: Manager, Supervisor, Software Engineer, Technical Lead, Project Manager, Intern etc.

2. **If a position doesn't exist, there will be no interview.**

3. **Interpreted 'One department can hire many candidates in relation to a position' as 'One department can interview many candidates in relation to a position'. And hence used isHired as an attribute in the Interview table instead of creating a separate table for it.**

4. **As a result of the above mentioned assumptions(also as described under each table), added various new tables to**

represent the many to many relationships, weak entity - strong entity relationships.

Reaction Policies Used:

- On Delete No Action and On Update No Action which are the same as default - restrict.
- Reason: I don’t want any primary keys being updated or deleted if they are referenced by other tables.

Operating System Used: Windows 10

ER Diagram generated via Workbench: