COMP20240

MySQL DB assignment (2019-2020)

Student Name:	**John O'Meara**
Student ID Number:	**19200612**
Operating System:	**Windows 10 Pro 64Bit**

Contents

Section 1 – Description of Tables

Table 01 – “applicants”

This table records the details of all applicants referenced by a unique applicant id number.

Attributes:

“app-id”	Primary Key,	Integer(11),	Unique,	Not Null
“firstname”	Variable Characters(45),		Not Null	
“lastname”	Variable Characters(45),		Not Null	
“address01”	Variable Characters(45),		Not Null	
“address02”	Variable Characters(45),		Default Value Null	
“address03”	Variable Characters(45),		Default Value Null	
“postcode”	Variable Characters(45),		Not Null	
“telephone”	Variable Characters(45) [this allows spaces],		Default Value Null	
“email”	Variable Characters(45),		Default Value Null	

Table 02 – “skills”

This table records the skills that are relevant for any jobs referenced by a unique skillcode.

Attributes:

“skillcode”	Primary Key,	Variable Characters(10),	Unique,	Not Null
“skillname”	Variable Characters(45),		Not Null	

Table 03 – “applicants_with_skills”

This table exists to allow a many-to-many relationship between “applicants” and “skills”

Any one applicant can have more than one skill.

Any one skill can be possessed by more than one Applicant.

The primary key of this table is a composite key of both attributes.

Attributes:

“app_id” Primary Key, Integer(11), Not Null

[note this can’t be unique as it may occur beside many skills]

[The foreign key “App” links this to “app_id” in the table “applicants”]

[The reaction policies chosen were “Cascade” on delete and on update as this is merely a link table – any updates or deletions to the primary data should cascade here]

“skillcode” Primary Key, Variable Characters(10), Not Null

[note this can’t be unique as it may occur beside many applicants]

[The foreign key “skill” links this to “skillcode” in the table “skills”]

[The reaction policies chosen were “Cascade” on delete and on update as this is merely a link table – any updates or deletions to the primary data should cascade here]

Table 04 – “universities”

This table records the details of all universities referenced by a unique university id number.

Attributes:

“un_id”	Primary Key,	Integer(11),	Unique,	Not Null
“universityname”	Variable Characters(45),	Not Null		
“address01”	Variable Characters(45),	Not Null		
“address02”	Variable Characters(45),	Default Value Null		
“telephone”	Variable Characters(45)) [this allows spaces],	Not Null		

Table 05 – “job_description”

This table contains a number of unique job descriptions referenced by a unique description id number.

It references the university to which it applies by foreign key

Each job description may have a number of job positions available.

Attributes:

“desc_id”	Primary Key,	Integer(11),	Unique,	Not Null
“jobtype”	Variable Characters(45),		Not Null	
“universityid”	Integer(11),	Not Null		

[The foreign key “uni” links this to “un_id” in the universities table]

[the reaction policies chosen were “cascade” in both cases as any change to the university identifier should be shown here, and if a university is deleted, the position no longer applies]

“num_positions”	Integer(11),	Not Null	Default Value 1
“interview_levels”	Integer(3),	Not Null,	Default Value 1

[This describes how many rounds of interviews are required to be successful. E.g., get the job if successful in first interview, or be referrered to a second interview]

Table 06 – “skills_for_job_description”

This table exists to allow a many-to-many relationship between “job_description” and “skills”

Any one job_description can require more than one skill.

Any one skill can be required by more than one job Description.

The primary key of this table is a composite key of both attributes.

Attributes:

“descid” Primary Key, Integer(11), Not Null

[note this can’t be unique as it may occur beside many skills]

[The foreign key “desc” links this to “desc_id” in the table “job_description”]

[The reaction policies chosen were “Cascade” on delete and on update as this is merely a link table – any updates or deletions to the primary data should cascade here]

“skillcode” Primary Key, Variable Characters(10), Not Null

[note this can’t be unique as it may occur beside many job descriptions]

[The foreign key “skills” links this to “skillcode” in the table “skills”]

[The reaction policies chosen were “Cascade” on delete and on update as this is merely a link table – any updates or deletions to the primary data should cascade here]

Table 07 – “interviews”

This table contains a number of interviews referenced by a unique interview id number.

It references the applicant for the position by foreign key

It also references the job description by foreign key

Attributes:

“int_id” Primary Key, Integer(11), Unique, Not Null

“date” DATE, Not Null

“app_id” Integer(11), Not Null

[The foreign key “app_id” links this to “app_id” in the applicants table]

[the reaction policies chosen were “cascade” in both cases as any change to the applicant identifier should be shown here, and if an applicant is deleted, the interview no longer exists]

[As an applicant can be called for multiple interviews, this cannot be ‘unique’]

“job_desc” Integer(11), Not Null

[The foreign key “job_desc” links this to “desc_id” in the job_description table]

[the reaction policies chosen were “cascade” in both cases as any change to the job description identifier should be shown here, and if a job description is deleted, the interview no longer exists]

[This is not constrained as unique since many interviews can be arranged for each job description]

“interview_level” Integer(3), Not Null Default Value 1

[This records which level (of the number of levels specified in the job description) the interview is.]

“interview_outcome” Integer(3), Not Null, Default Value 1

[After completion of the interview, the outcome of the interview – call for next round, offer job, candidate unsuccessful - can be recorded here.]

Table 08 – “positions_available”

This table contains separate entries for each of the positions available for any given job description. Each position is referenced by a unique position id number.

Attributes:

“positionid” Primary Key, Integer(11), Unique, Not Null

“job_description” Integer(11), Not Null

[The foreign key “job” links this to “desc_id” in the job_description table]

[the reaction policies chosen were “cascade” in both cases as any change to the job description identifier should be shown here, and if a job description is deleted, the position no longer exists]

“applicant” Integer(11), Unique, Not Null

[The foreign key “applicant” links this to “app_id” in the applicants table]

[the reaction policies chosen were “cascade” in both cases as any change to the applicant identifier should be shown here, and if an applicant is deleted, the position is no longer assigned]

[As an applicant can only fill one position, this has been constrained as ‘unique’]

Section 2 – Stored Procedures

Procedure 01 - “01_find_uni_by_name”

This is a simple parametric query that takes “universityname” as a parameter and returns the full tuple(s) associated with any instance(s) of that name from the table “universities”

Procedure 02 - “02_find_uni_by_id”

This is a simple parametric query that takes “un_id” as a parameter and returns the full tuple associated with the instance of that id from the table “universities”.

Procedure 03 - “03_find_applicant_by_surname”

This is a simple parametric query that takes “lastname” as a parameter and returns the full tuple(s) associated with any instance(s) of that surname from the table “applicants”.

Procedure 04 - “04_find_applicant_by_id”

This is a simple parametric query that takes “app_id” as a parameter and returns the full tuple associated with the instance of that id from the table “applicants.

Procedure 05 - “05_find_applicant_by_job_id”

This is a complex parametric query that takes a “desc_id” as a parameter and returns the full tuple associated with the applicant(s) that have that have an instance of that skill associated. It selects a triple join between the tables “applicants”, “applicants_with_skills” and “skills_for_job_description” correlating “app_id” with “app_id” in the first two tables and “skillcode” with “skillcode” in the second two tables. Only the attributes from the orignal “applicants” table have been selected in the returned tuple.

Procedure 06 - “06_find_job_by_skill”

This is a complex parametric query that takes “skillname” as a parameter and returns the full tuple associated with the job description(s) that have an instance of that skill associated. It selects a triple join between the tables “job_description”, “skills_for_job_description” and “skills” correlating “desc_id” with “descid” in the first two tables and “skillcode” with “skillcode” in the second two tables. Only the attributes from the orignal “job_description” table have been selected in the returned tuple.

Procedure 07 – “07_find_job_descriptions_sorted_by_university”

This is a query that selects a join between the tables “job_description” and “universities” correlating “university_id” with “un_id”. The attributes “universityname”, “jobtype”, “num_positions” and “interview_levels” table have been selected in the returned tuple and the data is ordered by “universityname”

Procedure 08 – “08_applicants_offered_job”

This is a query that counts the number of instances of “app_id” in the “interviews” table that have ‘Offer Job’ in the “interview_outcome” attribute.

[I didn’t use “distinct” in this procedure as each applicant can only be offered a particular job description once. If an applicant applies for two different job descriptions, I would consider that to be technically two applicants.]

Procedure 09 – “09_job_descriptions_requiring_research”

This is a query that counts the number of instances of “descid” in the “skills_for_job_description” table that have “XRE” in the “skillcode” attribute.

Procedure 10 – “10_interviews_on_date”

This is a simple parametric query that takes “date” as a parameter and returns the full tuple associated with that instance of the “date” attribute from the table “interviews”.

Procedure 11 – “11_applicants_interviewed_twiceormore”

This is a query that selects a join between the tables “interviews” and “applicants” and returns the attributes “app_id”, “firstname” and “lastname” for applicants whose “app_id” occurs 2 or more times.

Procedure 12 - “table_applicants_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “applicants”

Procedure 13 – “table_applicants_with_skils_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “applicants_with_skills”

Procedure 14 – “table_interiews_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “interviews”

Procedure 15 – “table_job_description_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “job_description”

Procedure 16 – “table_positions_available_insert_row”

This is a parametric query taking all attributes as parameters to add a tuple to the table “positions_available”

[I’ve set this procedure to only take parameters to enter data in the first two attributes as the applicant attribute will only be filled later when the position has been filled after a successful eound of interviews]

Procedure 17 – “table_skills_for_job_description_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “skills_for_job_description”

Procedure 18 – “table_skils_insert_row”

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “skills”

Procedure 19 - table_universities_insert_row

This is a standard parametric query taking all attributes as parameters to add a tuple to the table “universities”

Section 3 – ER Diagram