Course Project:

Government Directory Database System Project Phase 2 (15 Marks)

Due: *Week 12 (22:00hrs, Wednesday March 27, 2019) to eLearn ONLY*

Hours Expected: *16-20 hours per person*

Deliverables

1. **DDL Scripts**
   - **SQL creation script** that creates the tables in a MySQL database for the Logical Design that we have provided in the Appendix.
   - **SQL load script** that loads sample data into the tables created by your creation script. The sample data is provided in the zip file uploaded on eLearn. Please refer to *MySQL_LoadData_handout.pdf* for the details of loading data from .txt file into MySQL tables.
   - **You must include the** same **sample data in the deliverables** as mentioned in the Submission section.
     Note: You **ARE NOT** allowed to change the value of any sample data by inserting new records or deleting existing records from the sample data.

2. **DML / Stored Routine Scripts**
   a. Derive **DML SQL statements** to support the questions 1, 2 and 3. A **single SQL statement** is expected for each question.
   b. For **DML questions on Trigger and Stored Procedure (Questions 4 and 5)**, submit the script of your trigger and stored procedure (along with the script that calls it).
   c. The workload for teams with different number of members is different.
      - For those teams with only **3 members**, please complete **ALL 5 (i to v) DML questions**.
      - For those teams with only **2 members**, please complete the following **4 (ii, iii, iv and v) DML questions.**

Note:

(i) For all questions that expect inputs from the user, you are allowed to submit the SQL statement with a where clause containing a hardcoded value of the input. But we may be testing the correctness of your query with a different input. So, make sure that your SQL statement logic is correct that works for any input and it is easy for us to change the input.

(ii) All result outputs shown below are only for illustration purpose and may not represent the actual partial solution of the given questions.

DML Questions

1. Given a department name (eg. 'Prime Minister's Office'), display all job positions that exist under the department and all its immediate sub-departments (1 level down). For each job position listed, show the (a) Job ID (b) Job Title (c) Job Type (P for Political and NP Non-Political), (d) Dept ID of the Job position and (e) Number of assignments for the job position.

   Note: If the same staff is assigned to the same job position again, the number of assignments is taken as 2.

   The output should be sorted by "Number of assignments" in descending order.

   The following illustrates the partial output for query of department name **"Prime Minister's Office"**.

Job ID	Job Title	Job Type	Dept ID	Number of Assignments
…				
91398	Member	NP	90632	7
92274	PA to Deputy Secretary	NP	92211	…
…				
92120	Minister, Prime Minister's Office	P	88703	…
…				

2. Given a name in the where clause of the query, return all the matching identities of the staff member(s) who have the Staff Name that contains the queried name. We are only interested in staff(s) with multiple identities.
   Note that a staff member may have multiple staff records each having a possibly different Staff Name. As long as one of these records has Staff Name containing the queried name, the staff member should be included in the results.

   For example, for a staff member who has three staff records with (Staff ID 12575, Staff ID 1309951 and Staff ID 2682287) having "Vincent HO", "Vincent HO Hui Boon" and "Vincent HO" as Staff Names respectively, all the three should be returned. The pair of records (main and matched) should be returned only once with no duplication and the main Staff ID should be the smallest Staff ID (ie. 12575) among all its matching records.
   Note that the main Staff ID should not be the same as the matched Staff ID.

   For each staff member in the output, list the Staff ID, Staff Name, and Email of all the matched records. The output should be sorted in the ascending order of "Staff ID".

   The following illustrates the partial output for the query of name **"Vincent Ho".**

Main Staff ID	Main Staff Name	Main Email	Match Staff ID	Match Staff Name	Match Email
12575	Vincent HO	vincaft_ho@nrf.gov.sg	1309951	Vincent HO Hui Boon	vincaft_ho@nrf.gov.sg
12575	Vincent HO	vincaft_ho@nrf.gov.sg	2682287	Vincent HO	vincaft_ho@mhaw.gov.sg

3. Given a date in the where clause of the query, list all the non-political staffs that are assigned to a job position as of the given "Date". In addition, during his job assignment period return the number of speeches for which he is the contact person. The query should return (a) Staff ID, (b) Staff's Name, (c) Job ID, (d) Post Date, (e) End date, (f) Dept Name of the job assignment, and (g) Count of speeches as contact.

   Display the results in descending order of the Post Date followed by ascending order of Staff ID.

   The following illustrates the partial output for the query date **"2015-08-30".**

Staff ID	Staff Name	Job ID	Post Date	End Date	Dept Name	Contact Count
..						
25832	LIM Chang Choon Naomi	90852	2015-03-20	2018-10-22	Human Resource Development	3
..						

4. Write a trigger named '**before_speech_insert'** that executes before the insertion of a record to the SPEECH table.

   The trigger should check if the speech event date is made within post date and end date of the presenter's job assignment. The trigger should not allow any insertion of record into the SPEECH table if speech event date occurs outside the valid period of the presenter's job assignment.

   Throw an error with the message "Event occurs outside job valid period".

5. Write a stored procedure called **sp_get_dept_reorg** to list all the departmental reorganizations that have occurred. Note that a department could have only 3 types of reorganisation (i) a department cab be **split** into 2 or more (new) departments, (ii) a department could be **renamed** as another (new) department and (iii) 2 or more departments could be **merged** to form a new department.

   The stored procedure should list all three reorganizational changes that have occurred, if any. List the (a) new Department ID, (b) new Department name, (c) original Department ID, (d) original Department Name, (e) the date when the reorganization happened (which is the start date of the new Department), and (f) the type of reorganization (Merger, Split, Renamed).

   The output should be sorted in the ascending order of Reorganization Type and New Department ID.

   The following illustrates the partial output for the stored procedure**.**

New Dept ID	New Dept Name	Original Dept ID	Original Dept Name	Date Reorganized	Type
89276	ADMINISTRATION	90014	ADMINISTRATION	…	merger
89276	ADMINISTRATION	90048	PROCUREMENT	…	merger
…	…	…	…	…	…
90155	Engagement	90176	Strategy & Policy	2016-11-15	renamed
…	…	…	…	…	…
90959	IGP CORPORATE LEADERSHIP	91056	Igp Public Communications & Engagement	…	split
90990	IGP LEARNING & PROGRAMME DEVELOPMENT	91056	Igp Public Communications & Engagement	…	split
…	…	…	…	…	split

Submission

1. Submission is to be uploaded to the respective eLearn Assignment Submission folder GX-Phase 2, where X is the section (1-13).
2. The deliverable should be a folder named GXTY that is compressed as GX-TY.zip and contains the sub folders as shown below, where X is the section (1 – 13) and Y is your team number (01-14). Example if you belong to G3 and Team 05, your zip file should be named G3T05.zip.

   a. There should be only two folders, **"Data"** and **"Scripts".**
   b. The **"Data"** folder must contain all the data files provided to you in .txt files.
   c. The **"Scripts**" folder must contain the following **2 files**:
      - A file named **"ddl.sql"**

        This SQL file must contain **both** the CREATE and LOAD data statements such that it allows us to create tables and load your data with minimum effort using "execute all" command on MySQLWorkBench  .

        The first and second statements in this file should be "**create schema GXTY**;" and "**use GXTY**;" where X is your section (1 – 13) and Y is your team number (01-14).

        It should be followed by CREATE table and LOAD data statements in the proper table order. The path of your LOAD statements should be **D:\\GXTY\\data*.txt;** where X is your section (1 – 13) and Y is your team number (1-14)

        Note: Tester will use 'find and replace' function to change the path D:\\GXTY\\data to a path in his laptop for testing.
      - A file named **"script.sql"**

        This file must contain SQL statements to all the DML questions that your team is required to complete. Every SQL statement solution should have a comment tag that states the question number.

        Eg: #1:

        Select * from Table1;

        #2:

        Select * from Table2, Table3 etc

Grading Scheme

Project Phase 2
DDL Scripts: SQL Statements for database creation and data loading Marks will be deducted for - Missing data files - Missing create schema and use schema statements. - Incomplete or incorrect create or load script
DML / Stored Routine Scripts: SQL Statements for the required questions SQL Statements should be complete so that they can be used with different input values. Grades for individual questions might vary slightly based on the number of members in the team and the complexity of the query - Marks will NOT be awarded for misinterpreting the DML questions. - Clarification should be made online via the eLearn **Discussion Forum**. Topic: **Project Phase 2**
You will be penalized if your deliverables are not according to our instructions as stated in Submission section.

Appendix

MySQL supports functions and clauses that may come in handy.

IF control function

The **IF** function returns a value based on a condition.

IF(expr, if_true_expression, if_false_expression)

returns "if_true_expersssion" if the "expr" evaluates to **TRUE**, otherwise returns "if_false_expression". The function may be used to return a numeric or string.

For an example, given the following statement
IF(column1 IS NULL, 'N/A', column1)
the output will be N/A if column1 has NULL value; otherwise it will be column1 value

IFNULL control function

The **IFNULL** function accepts two arguments and returns the first argument if the argument is not null; otherwise it returns the second argument.

IFNULL(expr1, expr2)
returns expr1 if expr1 is not NULL, otherwise it returns expr2.

For an example;
IFNULL(column1, column2)
The output will be column1 if it is not NULL, otherwise it will be the column2 value.

UNION

The **UNION** operator is used to combine the result-set of two or more SELECT statements.

Each SELECT statement within UNION must have the same number of columns

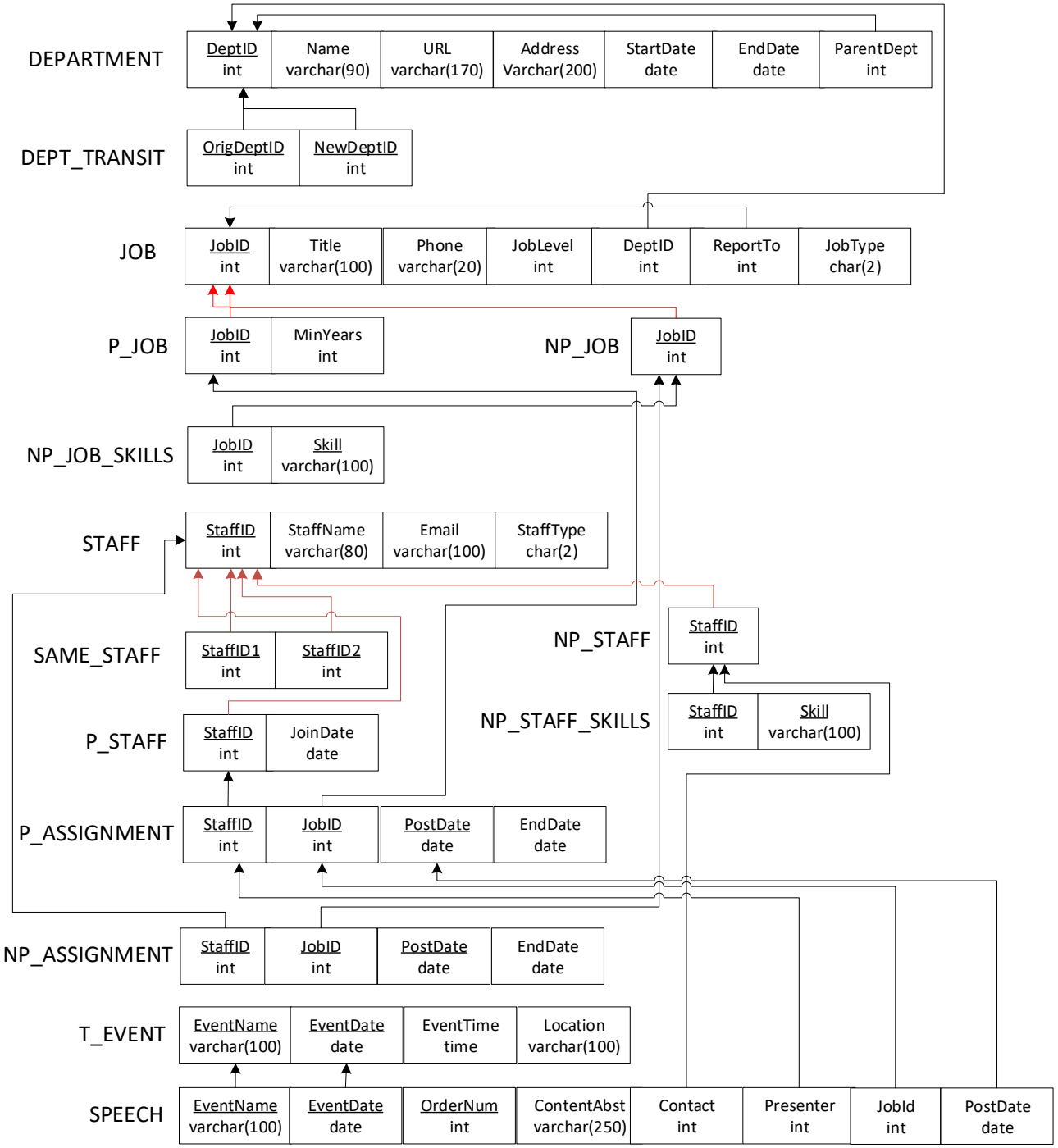
The columns must also have same data types

The columns in each SELECT statement must also be in the same order

For an example;

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

Project Logical Design



~~ END of Document