AY2019/2020 Semester 2: CS2102 Assignment 3 (SQL)

- The assignment consists of 10 one-mark questions and is due on **March 13 (Friday) at 6pm**.
- The assignment is to be submitted using Examplify. You can download this assignment only once.
- The password for this assignment is **cs2102sql**. The password is needed to start the assignment and to resume from a suspended session.
- The resume code for this assignment is **F01356**. This code is needed to resume from a reboot while Examplify was running.
- As the assignment will be auto-graded, it is **important** that your entered answers conform to the requirements specified under **Assignment Instructions**.
- The instructions given here can be viewed anytime either via **Exam Controls -> Exam Notice** or via **Exam Controls -> Exam Attachment** (to view as it as a pdf attachment).

Examplify Instructions

- Once you start the assignment with Examplify, you are given a time limit of **1000 minutes** (i.e., 16 hours 40 minutes) to complete and submit the assignment. This is the maximum time configurable using Examplify.
- **Suspending Assignment** You can suspend the assignment (as well as the timer) by clicking on 'Suspend Exam' from the 'Exam Controls' menu. To resume the suspended assignment, you need to enter the assignment password (Examplify refers to this as the Exam password). You can find screenshots on how to suspend and resume the assignment at https://wiki.nus.edu.sg/pages/viewpage.action?pageId=230556734. Please ensure that your laptop battery will not completely discharge while Examplify is being suspended. To resume from a suspended session, enter the exam/assignment password and press "Resume Exam".
- **Resuming from a reboot** If your laptop rebooted while using Examplify, Examplify will prompt for you for a resume code. This resume code is different from the assignment/exam password for resuming from a suspended Examplify session. The resume code for this assignment is **F01356**. To resume from the reboot, enter the resume code and press "Start Exam". Do not press "Close Exam"!
- **Submitting Assignment** It is important that you submit your assignment by the deadline (March 13 at 6pm) and before the timer times out. You will not be able to submit after your timer has timed out or after the deadline (even if your Examplify timer has not counted down to zero). Once you have submitted the assignment, you will not be able to make any changes to your answers. It is important that you ensure that you've answered all the questions and that the entered answers are indeed the

intended answers for the specific questions before you click the 'Submit' button. You can find screenshots on how to submit the assignment at https://wiki.nus.edu.sg/pages/viewpage.action?pageId=187598226.

- **Troubleshooting** If you encounter any technical problems while using Examplify, please refer to https://wiki.nus.edu.sg/display/DA/General+Support+and+Troubleshooting to see if the problem can be resolved following the guidelines there. If the problem can't be resolved, please send an email to CIT at **citbox25@nus.edu.sg** (and cc the lecturer in the email). You must not uninstall/reinstall Examplify after you'd downloaded the assignment.

Assignment Instructions

As this assignment will be **auto-graded**, it is important that your SQL answers conform to the requirements specified below. If the grading program detects that an entered answer does not conform to the specified requirements, that answer will receive 0 marks even if there is some correct solution embedded within the entered answer.

- For each question, you are to write a **single CREATE VIEW SQL statement** to answer the question. You MUST use the view schema provided for each question without changing any part of the view schema (i.e., view name, column names, or the order of the columns). For example, the provided view schema for Question 1 is "v1 (eid)", and if your answer to this question is the query "SELECT eid FROM Employees", then you must enter your answer in the answer box as follows:

```
CREATE VIEW v1 (eid) AS
SELECT eid
FROM Employees
;
```

- Each CREATE view statement must terminate with a semicolon.
- Each answer must be a syntactically valid SQL query without any typographical errors: a SQL answer that is syntactically invalid or contains very minor typographical error will receive 0 marks even if the answer is semantically correct.
- For each question, your answer view must not contain any duplicate records.
- You must not enter any extraneous text for your answer (e.g., "My answer is: CREATE VIEW v1 (eid) AS SELECT eid FROM Employees;") or enter multiple solutions (where your answer ends up being multiple SQL statements).
- Each question must be answered independently of other questions: the answer for a question must not refer to any other view that is created for another question.
- You are allowed to create the answer view using a single CTE statement (defining possibly multiple temporary tables). If your answer uses a CTE

statement to create temporary table(s), you **must not** use any of the following table names:
  - Names that are used by the application's schema (e.g., Offices, Departments)
  - Names that are used by the 10 answer views: v1, v2, . . . , v9, v10.
  - Any table name that begins with the prefix **zz**.
- Your answers must not use SQL constructs that are not covered in class (e.g., window functions).
- Your answers must be executable on PostgreSQL: your answers must not use constructs that are not supported in PostgreSQL.
- It is not necessary to write comments in your answer. If you choose to write comments, you should not use SQL comments that begin with two hyphen characters to avoid a potential copy-and-paste problem that results in a query being copied over as a single line with SQL code being commented out (e.g., "CREATE VIEW v1(eid) AS SELECT eid – yada yada FROM Employees;"). Using C-style SQL comments are fine (e.g., "CREATE VIEW v1(eid) AS SELECT eid /* yada yada */ FROM Employees;").

Database Schema

The questions are based on the following application about a company's operation. Its ER data model is shown below with the following constraints.

The company has at least one office. Each office (identified by oid with location specified by address) consists of one or more departments. Each department (identified by did with a budget dbudget) is located in one office and has one or more employees. Each employee (identified by eid) must belong to a department. The application focuses on two subclasses of employees: engineers and managers. An employee can be neither an engineer nor a manager, and no employee can be both an engineer and a manager. Each employee can specialize in 0 or more areas (identified by aid). Each department must be managed by exactly one manager, and each manager can manage 0 or more departments. Each engineer can work in 0 or more projects, and there must be at least one engineer working in each project. For each project P that an engineer E works on, the number of hours per week that E spends on P is given by hours. Each project (identified by pid with a budget pbudget) must be supervised by exactly one manager. A manager can supervise 0 or more projects. The attributes hours, dbudget and pbudget have non-null values.

The above image showing the ER model and an overview of the relational schema is also viewable as a pdf attachment for each question.

Relational Schema

The following is the relational schema for this application.

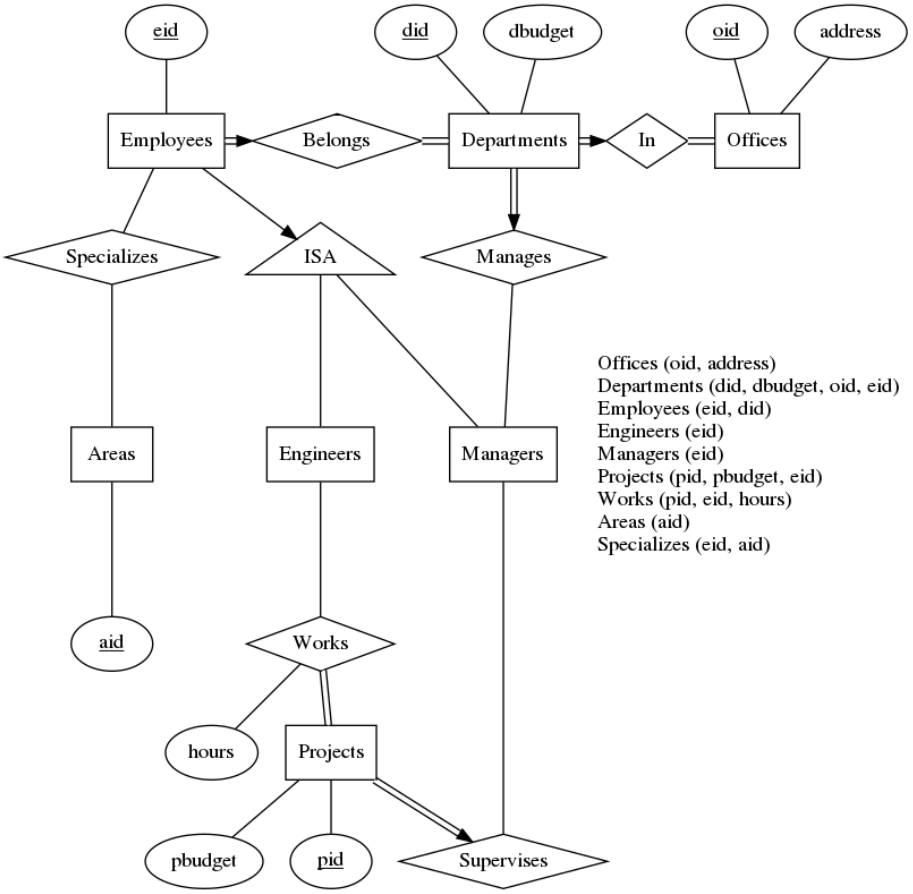

Figure 1: ER Model

```
CREATE TABLE Offices (
    oid             INTEGER,
    address         VARCHAR(60),
    PRIMARY KEY (oid)
);



/* eid = eid of department's manager */
CREATE TABLE Departments (
    did             INTEGER,
    dbudget         INTEGER NOT NULL,
    oid             INTEGER NOT NULL,
    eid             INTEGER NOT NULL,
    PRIMARY KEY (did),
    FOREIGN KEY (oid) REFERENCES Offices
);

CREATE TABLE Employees (
    eid             INTEGER,
    did             INTEGER NOT NULL,
    PRIMARY KEY (eid),
    FOREIGN KEY (did) REFERENCES Departments
);

CREATE TABLE Engineers (
    eid             INTEGER,
    PRIMARY KEY (eid),
    FOREIGN KEY (eid) REFERENCES Employees
);

CREATE TABLE Managers (
    eid             INTEGER,
    PRIMARY KEY (eid),
    FOREIGN KEY (eid) REFERENCES Employees
);


/* eid = eid of project's supervisor  */
CREATE TABLE Projects (
    pid             INTEGER,
    pbudget         INTEGER NOT NULL,
    eid             INTEGER NOT NULL,
    PRIMARY KEY (pid),
    FOREIGN KEY (eid) REFERENCES Managers
);
```

```
CREATE TABLE Works (
    pid             INTEGER,
    eid             INTEGER,
    hours           INTEGER NOT NULL,
    PRIMARY KEY (pid,eid),
    FOREIGN KEY (eid) REFERENCES Engineers,
    FOREIGN KEY (pid) REFERENCES Projects
);

CREATE TABLE Areas (
    aid             VARCHAR(5),
    PRIMARY KEY (aid)
);

CREATE TABLE Specializes (
    eid             INTEGER,
    aid             VARCHAR(5),
    PRIMARY KEY (eid,aid),
    FOREIGN KEY (eid) REFERENCES Employees,
    FOREIGN KEY (aid) REFERENCES Areas
);
```