CS34800, Fall 2016
Project 3
Transactions and Integrity Constraints

Due 11:59pm 07 November, 2016

Introduction

In this project, you will design and implement a database system that provides data updates with enforcing integrity constraints.

The following relation '***TA***' is given:

Email	FirstName	LastName	Course	Salary	Standing	Enrollment
dave@abcd.edu	Dave	Smith	CS240	1000	Excellent	50
dave@abcd.edu	Dave	Smith	CS290	1050	Excellent	70
sara@abcd.edu	Sara	Page	CS240	800	Satisfactory	50
sara@abcd.edu	Sara	Page	CS290	800	Satisfactory	70
eda@abcd.edu	Eda	Taylor	CS240	700	Unsatisfactory	50
david@abcd.edu	David	Evans	CS290	900	Good	70
eda@abcd.edu	Eda	Taylor	CS320	700	Unsatisfactory	120
david@abcd.edu	David	Evans	CS340	900	Good	130
anthony@abcd.edu	Anthony	Hare	CS240	900	Good	50
bill@abcd.edu	Bill	Jenkins	CS290	800	Satisfactory	70
chris@abcd.edu	Chris	Garner	CS240	1000	Excellent	50
harry@abcd.edu	Harry	Mess	CS290	700	Poor	70
barry@abcd.edu	Barry	Neesham	CS380	700	Good	110
garry@abcd.edu	Garry	Guptill	CS390	800	Satisfactory	150
jerry@abcd.edu	Jerry	Latham	CS380	900	Good	110
mary@abcd.edu	Mary	Boult	CS390	100	Excellent	150

Set of the functional dependencies defined on TA relation:
email → FirstName, LastName, Standing
Course → Enrollment
Email, Course → Salary

Question 1. (15 pts). Given functional dependencies and the schema for relation TA, do a 3NF-normalization. Create schema for normalized relations that preserves the given functional dependencies. Explain it.

Question 2. (10 pts). Create (in SQL) the normalized tables (from Question 1). Populate your normalized tables from the relation TA (hosted on one of the TAs' account).

Question 3. (5 pts). Enforce the following constraint:

a) ***Standing*** **= {'Excellent', 'Good', 'Satisfactory', 'Unsatisfactory', 'N/A'}**

b) Fix integrity violations (for Standing) in data, if necessary. If you find violation in standing, set standing to 'N/A'

Note: (3a) and (3b) can be done in any order

Question 4. (15 pts). Create a view that gives the following:

a) First and Last names of TAs, courses TAs are assigned to and number of students enrolled per course.

b) The original single table
Note: this view will not have integrity errors that the original table might have

Question 5. (15 pts). Implement triggers to enforce the following integrity constraints:

a) (10 pts) TA's salary can only be raised if TA's standing is 'Excellent' or 'Good'

b) (5 pts) No TAs can make more than $2,000 per month
Note: for (b) you could use compound trigger – see
http://docs.oracle.com/database/121/LNPLS/triggers.htm#CHDFEBFJ

Question 6. (20 pts). Write Java application to access your database using JDBC API. Your application should take course number (e.g. CS348) as an input and output the following:

a) First and Last names of TAs assigned to the course specified in the input

b) Total salary of TAs assigned to the course specified in the input

Note: a simple "hello world" level JDBC program can be found at:
https://www.cs.purdue.edu/homes/clifton/cs348/JDBCTest.java

Question 7. (20 pts). Extend functionality of your Java application to give TAs assigned to the course specified in the input a 5% salary raise if (both the conditions are met):

a) TA is in a 'Good' or 'Excellent' standing

b) Course Enrollment per TA is greater than 50 students.

Document design decisions which you use in your implementation. E.g. what do you do if after 5% salary raise the new salary violates constraints specified in question 5.

Question 8. (Extra-credit) (10 pts). While populating your normalized tables from the given table 'TA' in question 2, deal with tuples that violate integrity constraints specified in question 5. It is up to you how you fix integrity violations in data. You need to document design decisions you made.

Note 1. In this project, you will have READ-ONLY access to the database 'TA' (hosted on one of the TAs' account). The relation TA may contain data that violate integrity constraints. READ-ONLY means you can only run SELECT queries on the database and will not be able to modify the data.

For grading, your implementation will be tested on the TA table with different dataset.

What to submit

You are required to submit 7 files. For extra-credit question submit 8-th file.
1. In a pdf, named '**Design_*your_career_login*.pdf** ', submit answer for question 1 and document all the design decisions you made. Give necessary comments on how to compile, run and use your Java application
2. For question 2, submit a sql script named *q2_your_career_login.sql*.
3. For question 3, submit a sql script named *q3_your_career_login.sql*.
4. For question 4, submit a sql script named *q4_your_career_login.sql*.
5. For question 5, submit a sql script named *q5_your_career_login.sql*.
6. For question 6, submit a Java program named *q6_your_career_login.java*. Include comments on how to compile, run and use your program. Specify JVM version you are using
7. For question 7, submit a Java program named *q7_your_career_login.java*. Include comments on how to compile, run and use your program. Specify JVM version you are using
8. For extra-credit question 8, submit a script named *q8_your_career_login.sql*. Document your design decisions

Turning in your files

Please create a single zip file that contains the seven (or eight if you did extra-credit question) files as mentioned. Please turn in the project by uploading the zip file in Blackboard (click on Project 3 and Attach File).

We prefer a typed/typeset answer for question 1 and for document *Design_your_career_login.pdf*. If you (clearly and readably) handwrite your answer, then turn in a (clear, readable) scan as a PDF. If you don't know of a way to generate a PDF, bring it up in PSO. Don't forget to write your name on your assignment document, otherwise you may lose points.