CMPT 354 Assignment 3: SQL Bank Queries

You are to write, and run, SQL queries to answer a series of questions about a bank database. The database is on the Cypress server and you can access it from the CS labs in Burnaby and Surrey. The database is named (imaginatively) *cmpt354_bank*. This assignment is worth 6% of your final grade.

Bank Schema

The schema for the bank database is as follows. Primary key attributes are underlined and foreign keys are noted in superscript.

- *Customer* = {customerID, firstName, lastName, income, birthDate }
- *Account* = {accNumber, type, balance, branchNumber$^{FK\text{-}Branch}$}
- *Owns* = {customerID$^{FK\text{-}Customer}$, accNumber$^{FK\text{-}Account}$}
- *Transactions* = {transNumber, accNumber$^{FK\text{-}Account}$, amount}
- *Employee* = {sin, firstName, lastName, salary, branchNumber$^{FK\text{-}Branch}$}
- *Branch* = {branchNumber, branchName, managerSIN$^{FK\text{-}Employee}$, budget}

Notes

- The *customerID* attribute (*Customer*) is a unique number that represents a customer, it is *not* a customer's SIN
- The *accNumber* attribute (*Account*) represents the account number
- The *balance* (*Account*) attribute represents the total amount in an account
- The *type* (*Account*) attribute represents the type an account: chequing, saving, or business
- The *Owns* relation represents a many-to-many relationship (between *Customer* and *Account*)
- The *transNumber* attribute (*Transactions*) represents a transaction number, combined with account number it uniquely identify a transaction
- The *branchNumber* attribute (*Customer*)uniquely identifies a branch
- The *managerSIN* attribute (*Customer*) represents the SIN of the branch manager

Questions

Write *SQL* queries to return the data specified in questions 1 to 20.

Query Requirements

- The answer to each question should be a single SQL query
- You must order each query as described in the question (noted in green), order is always ascending unless specified otherwise
- Every column in the result should be named, so if the query asks you to return something like *income minus salary* make sure that you include an *AS* statement to name the column
- While your queries will *not* be assessed on their efficiency, marks may be deducted if unnecessary tables are included in the query (for example including *Owns* and *Customer* when you only require the *customerID* of customers who own accounts)

Queries

1)*First name* and last *name* of customers whose income is over $80,000, order by last name, then first name.

2)*Branch name, account number* and *balance* of accounts with balances over $115,000 held at branches with budgets greater than $2,000,000, order by branch name, then account number.

3)*First name*, *last name*, and *income* of customers whose income is at least twice the income of any customer named *Charles Smith*, order by last name then first name.

4)*Customer ID*, *income, account numbers* and *branch numbers* of customers with *income* greater than 90,000 who own an account at either the *London* or *Latveria* branches, order by *customer ID then account number*. The result should contain all the account numbers of customers who meet the criteria, even if the account itself is not held at *London* or *Latveria*.

5)*Customer ID*, *types*, *account numbers* and *balances* of business (type *bus*) and savings (type *sav*) accounts owned by customers who own at least one business account and at least one savings account, order by *customer ID, then type, then account number*.

6)*SIN*, *branch name*, *salary* and *salary* – *manager's salary* (that is, salary of the employee minus the salary of the employee's manager) of all employees, order by descending (salary – manager salary).

7)*Customer ID* of customers who have an account at the *Berlin* branch, who do *not* own an account at the *London* branch and who do not co-own an account with another customer who owns an account at the *London* branch,

order by customer ID. The result should not contain duplicate customer IDs.

8)*SIN*, *last name*, and *salary* of employees who earn more than $80,000, if they are managers show the *branch name* of their branch in a fourth column (which should be NULL for most employees), order by salary in decreasing order. You must use an outer join in your solution (which is the easiest way to do it).

9)Exactly as question eight, except that your query *cannot* include any join operation.

10) *Customer ID*, *last name* and *birth dates* of customers who own accounts in all of the branches that *Adam Rivera* owns accounts in, order by customer ID.

11) *SIN*, *first name*, *last name* and *salary* of the highest paid employee (or employees) of the *Berlin* branch, order by sin.

12) *Sum* of the employee salaries (a single number) at the *Latveria* branch.

13) *Count* of the number of different first names of employees working at the *London* branch and a *count* of the number of employees working at the *London* branch (two numbers in a single row).

14) *Branch name*, and *minimum*, *maximum* and *average salary* of the employees at each branch, order by branch name.

15) *Customer ID*, *first name* and *last name* of customers who own accounts at a minimum of two different branches, order by customer ID.

16) *Average income* of customers older than 50 and *average income* of customers younger than 50, the result must have two named columns, with one row, in one result set (hint: look up T-SQL time and date functions).

17) *Customer ID*, *last name*, *first name*, *income,* and *average account balance* of customers who have at least three accounts, and whose *last names* begin with *Jo* and contain an *s* (e.g. **Jo**hn**s**on) **or** whose *first names* begin with A and have a vowel as the letter just before the last letter (e.g. **A**ar**o**n), order by customer ID. Note that this will be much easier if you look up LIKE wildcards in the MSDN T-SQL documentation. Also note - to appear in the result customers must have at least three accounts and satisfy one (or both) of the name conditions.

18) *Account number*, *balance*, *sum of transaction amounts*, and *balance - transaction sum* for accounts in the *New York* branch that have at least ten transactions, order by account number.

19) *Branch name*, *account type*, and *average transaction amount* of each account type for each branch for branches that have at least 50 accounts of any type, order by branch name, then account type.

20) *Branch name, account type, account number, transaction number and amount* of transactions of accounts where the average transaction amount is greater than three times the (overall) average transaction amount of accounts of that type. For example, if the average transaction amount of all business accounts is $2,000 then return transactions from business accounts where the average transaction amount for that account is greater than $6,000. Order by branch name, then account type, account number and finally transaction number. Note that all transactions of qualifying accounts should be returned even if they are less than the average amount of the account type.

Accessing the Database from Home

If you find it inconvenient to work in one of our computer labs and want to complete the assignment from home, there are a couple of ways of doing this.

The first (and simplest) is to use Remote Desktop to connect to the CSIL lab servers. You received an email about your database on the Cypress server and this email contains a link that takes you to a help page that describes how to use Remote Desktop. The page is here, the CSIL Windows Terminal server name is *leto.csil.sfu.ca*. Once you have connected run *SQL Server Management Studio* under *Microsoft SQL Server 2012* from the Start menu. The *SQL Server Database* name is *CYPRESS.csil.sfu.ca*. Then navigate to the *cmpt354_bank* database and write and run your queries

The second method is to install SQL Server on your own PC, create a database called cmpt354_bank and then run this script to create and populate the database.

Note that I am not particularly willing to try to troubleshoot any problems you run into trying to do this by email (or on the discussion forum). I am willing to help in person (i.e. if you bring in a laptop so that I can actually see what you are doing).

Finally, and importantly, if you decide to complete the assignment at the last minute from home and are unable to either connect via Remote Desktop or get the database running on your computer I will not accept this as an excuse to submit the assignment late.

Assessment

The assignment is out of 20.

Submission

You should submit your assignment online to the CoursSys submission server. You should submit the following two files:

1) A single .sql file with your solutions to all twenty questions, with the solution to each question preceded by a title in comments (e.g. -- question 1)
2) A single .pdf file containing the queries and output for all of the questions. The document should contain your queries and, after each query, the output for that query. Copy the output from SQL Server when you send the Query results to Text. You can select this in the Query menu in SQL Server (Query : Results to Text) or by pressing Ctrl T when focus is in the query window.

The assignment is due by 11:59pm on Wednesday the 1st of November.

CMPT 354 Home

John Edgar (johnwill@sfu.ca)