CMPT 354
Database Systems

Simon Fraser University
Spring 2017

Instructor: Oliver Schulte

Assignment 2: Relational Queries, SQL.

Instructions: Check the instructions in the syllabus. The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. *Everything submitted should be your own writing or coding.* You must not let other students copy your work. On your assignment, put down your **name**, the number of the assignment and the number of the course. Spelling and grammar count.

Group Work: Discussions of the assignment is okay, for example to understand the concepts involved. If you work in a group, put down the name of all members of your group. There should be no group submissions. Each group member should write up their own solution to show their own understanding.

For the due date please see our course management server https://courses.cs.sfu.ca. Part of the assignment task is figuring out how to use the CSIL system. If your assignment is late because you did not figure this out soon enough, you will lose marks according to the syllabus policy.

Instructions for what to submit and how to obtain the required databases are posted appear at the end of this file.

The assignment asks you to write SQL equivalents for relational algebra queries. Since in the course we study SQL before relational algebra, you may want to start with the SQL queries first (parts IV and V). Then you should be able to make use of your SQL query logic to design the relational algebra queries.

Part I: Relational Algebra

Consider the following relational schema. An employee can work in more than one department; the *pct_time* field of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(*eid*: integer, *ename*: string, *age*: integer, *salary*: real)
Works(*eid*: integer, *did*: integer, *pct_time*: integer)
Dept(*did*: integer, dname: string, *budget*: real, *managerid*: integer)

Write the following queries in **relational algebra**. In another part you will write SQL equivalents for the relational algebra queries.

1. Print the names and ages of each employee who works in both the Hardware department and the Software department.
2. Print the name of each employee whose salary exceeds the budget of all of the departments that he or she works in. To illustrate, if the employee works in 3 departments, whose budgets are 1000, 2000 and 3000 respectively, then the maximum budget of all of the departments that he/she works in is 3000.
3. Find the *managerids* of managers who manage only departments with budgets greater than $1 million.
4. Find the *enames* of managers who manage the departments with the largest budgets. To illustrate, suppose there are three departments like this: department 1 with budget $1 million, department 2 with budget $2 million, department 3 with budget $2 million. Then departments 2 and 3 have the largest budgets. Write this query *without using aggregate functions*.

Marking Criteria:

Total Marks: 100 Marks
1. Technical Correctness (70 Marks)
2. Notational Correctness (20 Marks)
3. Presentation (10 Marks)

Part II: Relational Algebra

Consider the following relational schema.

Flight(number:integer, airline: string, from: string, to: string)
CanAirport(code: string, location: string)
USAirport(code: string, location: string)

The "from" and "to" fields in the Flight table point to the "code" fields in the airport table. For example, a typical entry in the Flight table might be

56, AIRC, YEG, YYV

and a typical entry in the CanAirport table might be

YEG, Edmonton.

Write a query in **relational algebra** for the following requests.

a. Find all airlines that fly to all Canadian airports.
b. Find all airlines that fly to a Canadian airport but not to any US Airport.
c. Find the locations of all airports with at least one flight to New York.

Marking Criteria:

Total Marks: 100 Marks
1. Technical Correctness (70 Marks)
2. Notational Correctness (20 Marks)
3. Presentation (10 Marks)

Part III: Relational Algebra

Introduction

In this assignment we use a Star Wars Trilogy database for science fiction fans. Your task will be to create queries that answer some of the often asked questions. The schema is as follows. (You may also want to look at the database.)

Tables

Characters: contains information about the character's Name (primary key), Race (if known), Homeworld (if known) and Affiliation (rebels/empire/neutral/free-lancer).

Planets: contains information about the planet's Name (primary key), its Type (gas/swamp/forest/handmade/ice/desert), and its Affiliation (rebels/empire/neutral)

TimeTable: contains CharacterName, PlanetName, Movie in which the character visited the planet and the time of arrival and departure from the planet. Movie 1 represents *The Star Wars*, Movie 2 represents *Empire Strikes Back***,** and Movie 3 represents *Return of the Jedi.* Each movie has been divided into 10 time chunks and these chunks are used to define time of arrival and departure. So if Darth Vader visited Bespin (Cloud City) in Empire Strikes Back from the middle of the movie till its end, the record of it will look like this:

CharacterName	*PlanetName*	*Movie*	*Time of Arrival*	*Time of Departure*
Darth Vader	Bespin	2	5	10

Write the following queries in **relational algebra**.

1. Find all characters that have been on all neutral planets.
2. Find names of the planets visited by humans who are affiliated with the empire.
3. On which planets and in which movies has Luke been at the same time on the planet as Darth Vader?
4. Find humans that visited desert planets and droids that visited swampy planets. List the movies when it happened and the names of the characters.

Marking Criteria:

Total Marks: 100 Marks
1. Technical Correctness (70 Marks)
2. Notational Correctness (20 Marks)
3. Presentation (10 Marks)

Part IV: SQL Queries

In this part we use a database called cmpt354_company.
Consider again the following relational schema.

Emp(*eid*: integer, *ename*: string, *age*: integer, *salary*: real)
Works(*eid*: integer, *did*: integer, *pct_time*: integer)
Dept(*did*: integer, dname: string, *budget*: real, *managerid*: integer)

Write the following queries in **SQL**.

1. Print the names and ages of each employee who works in both the Hardware department and the Software department.
2. Print the name of each employee whose salary exceeds the budget of all of the departments that he or she works in. To illustrate, if the employee works in 3 departments, whose budgets are 1000, 2000 and 3000 respectively, then the maximum budget of all of the departments that he/she works in is 3000.
3. Find the *managerids* of managers who manage only departments with budgets greater than $1 million.
4. Find the *enames* of managers who manage the departments with the largest budgets.

Marking Criteria

Total Marks 100

1. Technical Correctness (80 Marks)
2. Correctness of Results from Queries (10 Marks)
3. Presentation, Style (10 Marks)

Part V: SQL Queries

Introduction

In this part we use a Star Wars Trilogy database, called cmpt354_starwars, for science fiction fans. Your task will be to create SQL queries that answer some of the often-asked questions. You may have to use brackets for column names with spaces. For example, to project the Time of Arrival column from the TimeTable relation, you can write

SELECT [Time of Arrival] from TimeTable;

Write **SQL queries** for computing the following queries.

1. Find all characters that have been on all neutral planets.
2. Find distinct names of the planets visited by humans affiliated with the empire.
3. On which planets and in which movies has Luke been at the same time on the planet as Darth Vader?
4. Find humans that visited desert planets and droids that visited swampy planets. List the movies when it happened and the names of the characters.
5. For each character and for each neutral planet, how much time total did the character spend on the planet? The column that contains the total times should be labelled "TotalTime".

Marking Criteria

Total Marks 100

1. Technical Correctness (80 Marks)
2. Correctness of Results from Queries (10 Marks)
3. Presentation, Style (10 Marks)

Installation and Submission

Installing the Databases

This assignment requires you to execute SQL queries on SQL Server. You will need the two databases "Starwars" and "company". You have two options for importing these to SQL Server.

Option 1: DIY. We have posted .bak files for the two databases on-line. You can import these to your personal SQL Server installation using the same procedure as with the AdventureWorks database.

Option 2: Use the CSIL setup. The databases have been imported into the CSIL SQL Server instance. You have read access to them, which should be sufficient for this assignment. If you want to import the tables into your CSIL database, you can use the same procedure as with the AdventureWorks database.

Writing Up Your Solutions

Part I. Relational Algebra. Submit four relational algebra expressions. You can handwrite relational algebra expressions and scan them into a .pdf. You will lose marks if your writing is hard to read. This method has the problem that handwriting is difficult to fix, if you want to revise your query. Experience shows that investing a bit of time into figuring out how to typeset RA expressions leads to better solutions. You can use keywords like UNION for $\cup$, NATURAL JOIN for $\bowtie$, but make sure it's clear what algebra operators your keywords refer to.

Part II. Relational Algebra. Submit three relational algebra expressions.

Part III. Relational Algebra. Submit four relational algebra expressions.

All queries should be submitted together as a single pdf file called algebra.pdf. (Write the solution of Part I, Part II and Part III into one pdf file named algebra.pdf)

Part IV: SQL queries. Submit four queries Please submit your solutions two ways: 1) as an executable .sql script, 2) with screenshots. (Submit both, not just one).

Part V: SQL queries. Submit five queries. Please submit your solutions two ways: 1) as an executable .sql script, 2) with screenshots. (Submit both, not just one).

Submit a single file called "script.sql" for your solution to both parts IV and V. For the screenshots, submit a single file sql-screen.pdf for both parts IV and V. (Write the solution of Part IV and Part V into sql-screen.pdf)

Do not change the column names from the original database. This means that you can use the AS alias construct for table names, but not column names. For example:

GOOD: SELECT [Time of Arrival] from TimeTable;

BAD: SELECT [Time of Arrival] AS MyNewName from TimeTable;

An exception is query 5, where you introduce a new type of column for the total time spent by each character on each planet. The column that contains the total times should be labelled "TotalTime".

The purpose for having all solutions use the same column names is that it makes it easier for us to check if your solutions are correct.