University of Toronto
CSC343, Fall 2018

Assignment 1

Due: NEW DUE DATE Monday, October 22, 2018, 11:59pm

Learning Goals

By the end of this assignment you should be able to:

1. read a new relational schema, and determine whether or not a particular instance is valid with respect to that schema,
2. apply the individual techniques for writing relational algebra queries and integrity constraints that we learned in class,
3. combine the individual techniques to solve complex problems, and
4. identify problems that cannot be solved using relational algebra.

These skills will leave you well prepared to learn SQL. Later in the course, you will learn about how to develop your own schema based on knowledge of the domain. Even though developing a schema is really the first step in building a database, it is a more advanced task than querying an existing database; this is why you will be learning about it later.

Domain

For this assignment, you will operate on a database that stores aviation information. The database contains information about airlines, airports, flights as well as crew assignments.

Schema

Relations

Airport(apID, name, city, country)

AirportDistance(apID1, apID2, distance)

Airline(cID, callSign, headquarters, mainHub, website)

Aircraft(acID, type, avgSpeed, maxSpeed, range, serialNumber, firstFlightDate, manufacturer)

Employee(eID, name, airline, type, rank, primary_language, secondary_language, startDate)

Flight(cID, flightNumber, origin, destination, departureTime, arrivalTime)

FlightInstance(fID, cID, flightNumber, date, acID, actualDepartureTime, actualArrivalTime)

CrewAssignment(fID, eID, role)

PilotExpertise(eID, aircraftType)

- Airports: apID is the International Air Transport Association (IATA) airport identifier. The estimated distance (measured in km) between two airports is stored in relation AirportDistance.

- Aircrafts: acID is an aircraft's international registration number (for example, C-GBIP). The 'range' is measured in km.
- Airlines: each airline is identified by its IATA code, for example 'AC' for Air Canada, and the database stores information about the airlines international call sign, (e.g., 'Air Canada') as well as the locations of its main hub and headquarters ('YYZ' and 'Montreal', respectively, for Air Canada). 'YYZ' is Pearson Airport's IATA code.
- Employees: the database holds information about the employees of all airlines. All pilots, flight attendants and ground crew are employees. All pilots have a rank and expertise in flying different types of aircrafts. Only pilots have flight expertise, and their expertise is in the aircraft types listed in the database. For every employee, a primary language is recorded. In addition, each employee may have a second language in which he or she is fluent. All pilots have ranks, which must be one of Captain, Navigator, or First Officer.
- Flights: a flight is identified by a company code (for example, 'AC') and a flight number (for example, 456). For every flight, the database stores the origin and the destination airports (Toronto and Ottawa for AC456) and the scheduled departure and arrival times (3:10pm and 4:09pm for 'AC456'). Every flight instance is assigned a 'Captain' and a 'First Officer'.
- Flight instance: contains information about the implementation of a flight on a specific date, for example, flight AC 456 on the 1st of July 2018 was implemented using aircraft C-GBIP, departed at 3:20pm and arrived at 4:15pm.
- Crew assignment: the database stores information about the employees (the crew) assigned to each flight implementation. The employees must be present in the database.

Integrity constraints

- $AirportDistance[apID1] \subseteq Airport[apID]$
- $AirportDistance[apID2] \subseteq Airport[apID]$
- $FlightInstance[cID, flightNumber] \subseteq Flight[cID, flightNumber]$
- $FlightInstance[acID] \subseteq Aircraft[acID]$
- $CrewAssignment[fID] \subseteq FlightInstance[fID]$
- $CrewAssignment[eID] \subseteq Employee[eID]$
- $PilotExpertise[eID] \subseteq Employee[eID]$
- $PilotExpertise[aircraftType] \subseteq Aircraft[type]$
- $Employee[airline] \subseteq Airline[cID]$
- $Flight[origin] \subseteq Airport[acID]$
- $Flight[destination] \subseteq Airport[acID]$
- $PilotExpertise[aircraftType] \subseteq Aircraft[type]$

Write the queries and integrity constraints below in relational algebra. There are a number of variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in class and on the slides. You may only use assignment, and the operators we have used in class: $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$. Assume that all relations are sets (not bags), as we have done in class, and

do not use any of the extended relational algebra operations from Chapter 5 of the textbook (for example, do not use the division operator or aggregation).

Some additional points to keep in mind:

- Do not make any assumptions about the data that are not enforced by the original constraints given above. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute, in other words, there are no null values.
- The condition on a select operation can use comparison operators (such as $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). Simple arithmetic is also okay, `e.g.`, attribute1 $\leq$ attribute2 + 5000.
- You may assume that all attribute values are ordered so they can be compared with $=, <, >, \leq, \geq, etc.$
- You are encouraged to use the assignment operator to define intermediate results.
- Add commentary explaining what you're doing. This way, even if your final answer is not completely correct, you may receive partial marks.
- The order of the columns in the result doesn't matter.
- When asked for a maximum or minimum, if there are ties, report all of them.
- If you believe that a query cannot be expressed in the Relational Algebra language that you are using, then, simply write "cannot be expressed". Note: The queries are not ordered by difficulty.

Simplifying assumption: we consider that all flights start and finish on the same day. We assume that we can perform arithmetic operations on the values stored in the 'departureTime' and 'arrivalTime' attributes, e.g., 'departureTime' - 'arrivalTime' = 50 means that the difference between the two recorded dateTime values is 50 minutes.

Part 1: Queries

1. List for each airline the employees who have expertise in at least two different types of aircraft. Use the two-character airline code.
2. Find all the airports (airport code and name) that one can reach on the same day if one departs from Pearson airport (apID 'YYZ') and is willing to take at most 3 flights. Assume that one needs at least 120 minutes between consecutive flights.
3. Find all flight instances on which no crew member speaks French (as either a first or second language). Return the airline, flight number, and the date of the flight instance. (Flight instances with no crew assigned satisfy the condition and should be returned as well.)
4. List all airports with more than 3 departing flights a day.
5. Find all pairs of employees who only work together. For example, the query should return a pair of employees 1 and 2 if employee 1 only works on flight instances on which employee 2 also works and vice versa. Return only the employee identifiers. (The employees returned must work on at least one flight instance.)

6. Find all employees (just their identifiers) who have scheduling conflicts, i.e., are scheduled to work on at least two incompatible flight instances. We consider two flight instances to be incompatible is they (1) overlap, i.e., they are scheduled on the same date and their flying time overlaps or (2) the flights happen on the same date, they do not overlap, but the destination of the earlier flight is different from the departure of the later flight, or if the destination of the earlier flight and departure of the later flight are the same, but there is less than an hour between the scheduled arrival of the earlier flight and the departure of the later flight.

7. Find all pilots who have expertise in all the aircraft types in the database. Return the pilot's employee id and rank.

8. Find the airlines with the most intra-Canada flights (the origin and destination of the flight is within Canada). Return only the company codes.

9. Return the code of the airline with the newest aircraft.

10. For every company, return the flight instances with assignment violations, i.e. the assigned 'Captain' does not have expertise in flying the aircraft type of the plane or the distance between the departure and destination airports is greater than the range of the plane assigned to the flight instance. Return the two-character company code and the flight instance id.

11. Find all employees (return their employee ids) who with speak French as a second language or have expertise in an aircraft with maximum speed over 700km/h at typical cruise altitude.

12. Return the codes of the airlines that do not have any Boeing Dreamliner (aircraft type 787-9) in their registered fleet.

Part 2: Integrity constraints

Express in Relational Algebra the following integrity constraints:

1. No flight originates and ends in the same airport.

2. An employee cannot be assigned multiple times to the same flight implementation.

Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online. Whatever you choose to use, you need to produce a final document in pdf format.

You must declare your team and hand in your work electronically using the MarkUs online system. Instructions for doing so will be posted on the Assignments page of the course website. Well before the due date, you should declare your team and try submitting with MarkUs. You can submit an empty file as a placeholder, and then submit a new version of the file later (before the deadline, of course); look in the "Replace" column.

For this assignment, hand in just one file: A1.pdf. If you are working in a team, only one of you should hand it in. Check that you have submitted the correct version of your file by downloading it from MarkUs;