University of Toronto
CSC343, Fall 2019

Assignment 1

Due: Wednesday, October 9, before 8pm

Learning Goals

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

- Read a new relational schema and determine whether or not a particular instance is valid with respect to that schema.
- Apply the individual techniques for writing relational algebra queries and integrity constraints that we learned in class.
- Combine the individual techniques to solve complex problems.
- Identify problems that cannot be solved using relational algebra.

These skills will leave you well prepared to be a strong SQL programmer.

The domain

For this assignment, you will write queries and integrity constraints on a database for a museum.

The objects in a museum can be as small as an individual button or as large as a tank. Each one is identified by a unique "catalogue number".

Donors make donations to the museum. A donation can consist of a single object or multiple objects. The individual objects are given unique "catalogue numbers" when they are entered into the database. This process is called "cataloguing". Because a donation can have many items, the items are often catalogued on different days and by different people.

One step in cataloguing an object is to assign it to a category. In order for museums to share information about their collection, a standard classification system is used. In Canada and the United States, the standard is the "Chenhall" system, devised by Robert G. Chenhall. Like biological classification, the Chenhall system is based on a hierarchy. In this assignment, we will simplify the Chenhall system by limiting it to a three-level hierarchy: The most general level will be called "category", the next will be "primary term", and the most specific category will be called "secondary term". In our database, when an item is catalogued, it is assigned a secondary term. For example, a man's dress shoe would be catagorized with secondary term "footware", which falls under primary term "clothing", which falls under category "personal artifacts". To solve the problems in this assignment, you do not need to know any of the specific names for categories and terms in the Chenhall hierarchy. You just need to know that we will use the three levels called category, secondary term and primary term.

In addition to classifying objects, our database can record tags associated with an object. These are useful when searching in the museum's database.

Schema

Relations

- Object(CN, date, name, description, type, length, width, height, value, who)
  A tuple in this relation represents an object in the museum's collection. *CN* is the object's catalogue number, *date* is the date on which it was catalogued, *name* is a short name for the object, *description* is a paragraph-length description of the object, *type* is the Chenhall secondary term for the object, *length*, *width* and *height* give the object's dimensions, *value* is the estimated value of the object, and *who* is the ID of the staff member who catalogued the object.

- Tag(CN, phrase)
  A tuple in this relation represents the fact that object *CN* has tag *phrase*. If an object has multiple tags, each one is represented by a separate tuple. A single tag can also include more than one word. For example, an item could have two tags, "Fenian raids" and "battle".

- Donor(DID, surname, firstname, address, email)
  A tuple in this relation represents a person who has donated to the museum's collection. *DID* is the donor ID, *surname* and *firstname* are the donor's names, and *address* and *email* are the donor's surface mail and email addresses.

- Donation(NID, date, DID)
  A tuple in this relation represents a donation of one or more objects to the museum's collection. *NID* is the donation ID, *date* is the date on which the donation was made, and *DID* identifies the donor who made the donation.

- Contains(NID, CN)
  A tuple in this relation represents the fact that donation *NID* contains object *CN*.

- Staff(SID, surname, firstname, address, email, type, date)
  A tuple in this relation represents a person who is or has been on the museum's staff. *SID* is the staff member's ID, *surname* and *firstname* are the staff member's names, *address* and *email* are the staff member's surface mail and email addresses, *type* is the type of staff member he or she is, and date *date* is the start date of his or her position at the museum.

- Chenhall(category)
  A tuple in this relation represents the fact that *category* is a category in the Chenhall classification system.

- PrimaryTerm(primary, category)
  A tuple in this relation represents the fact that *primary* is a primary term within category *category* in the Chenhall classification system.

- SecondaryTerm(secondary, primary)
  A tuple in this relation represents the fact that *secondary* is a secondary term under primary term *primary* in the Chenhall classification system.

Integrity constraints

- Object[type] $\subseteq$ SecondaryTerm[secondary]
- Object[who] $\subseteq$ Staff[SID]
- Tag[CN] $\subseteq$ Object[CN]

- Donation[DID] $\subseteq$ Donor[DID]
- $\Pi_{NID} Donation - \Pi_{NID} Contains = \emptyset$
- Contains[NID] $\subseteq$ Donation[NID]
- Contains[CN] $\subseteq$ Object[CN]
- Π_{type}Staff $\subseteq$ {"permanent", "temp", "volunteer", "intern"}
- PrimaryTerm[category] $\subseteq$ Chenhall[category]
- SecondaryTerm[primary] $\subseteq$ PrimaryTerm[primary]

Warmup: Getting to know the schema

To get familiar with the schema, ask yourself questions like these (but don't hand in your answers):

1. What does the following integrity constraint mean in plain English?

$$\Pi_{NID} Donation - \Pi_{NID} Contains = \emptyset$$

2. Why didn't we include the items for a donation right in the *Donation* relation? Did we really need to have the *Contains* relation?
3. Does our schema allow for a staff member to change roles, for instance from a volunteer to permanent staff? If so, how would that work? If not, why not?
4. Does our Chenhall system (Chenhall categories, primary terms and secondary terms) form a tree, or could one element in it have multiple "parents"?

Part 1: Queries

Write the queries 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 this course. You may 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 (for example, do not use the division operator).

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, including the ones written in English. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute. For those of you who know some SQL, in other words, there are no null values.
- Remember that the condition on a select operation may only examine the values of the attributes in one tuple, not whole columns. In other words, to use a value (other than a literal value such as 100 or "Adele"), you must get that value into the tuples that your select will examine.
- 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.

- In your select conditions, you can compare dates using comparison operators such as $<$.
- Some relations in our schema have a date-time attribute. You may use comparison operators on such values. You may refer to the year component of a date-time attribute d using the notation d.year.
- You are encouraged to use assignment to define intermediate results.
- It's a good idea to add commentary explaining what you're doing. This way, even if your final answer is not completely correct, you may receive part marks.
- The order of the columns in the result doesn't matter.
- If there are ties, report all of them.

At least one of the queries and/or integrity constraints in this assignment cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed". Note: The queries are not in order according to difficulty.

1. Context: New staff need oversight, particularly volunteers, because learning the Chenhall system takes practise.

   Query: Find the latest object catalogued by the newest volunteer. Report the volunteer's ID number and first name, as well as the object's description, type, and date of cataloguing.

   The possible ties are interesting in this query. For example, two volunteers could be tied for newest. They *each* should be included in the result with their own latest-catalogued object(s), even if one volunteer's latest-catalogued object(s) were catalogued before the other volunteer's latest-catalogued object(s).

2. Context: We are curious about donors who've made very broad contributions.

   Query: For each donor who has made a donation in every Chenhall category in the database, report in a single tuple:

   - the value of the single most valuable object they have ever donated, and
   - the value of the single least valuable object they have ever donated.

3. Context: There is no good reason for this query. :-)

   Query: For each donation with three or more objects in it, report the second tallest object in the donation. Report the object's catalogue number, height, and width, as well as the ID of the donation it came from.

4. Context: We are looking for donations that may contain unusual objects.

   Query: Find any donation that was catalogued entirely by one staff person, where the same tag was used for at least 2 objects, but that tag was never used for any object catalogued by the same staff person in any other donation. For each such donation, report the donation ID and donor ID, and the staff member ID of the one person who catalogued it.

5. Context: We want to bring back volunteers who used to help consistently but no longer do.

   Query: Find all volunteers who have catalogued object(s) from at least two different donations every year up to and including in 2016, but have catalogued nothing since. By "every year" we mean every year that appears in the *Object* relation. For each of these volunteers, report their staff ID and email address.

6. Context: For efficiency, it makes sense that a donation with very similar objects be catalogued by a single person.

   Query: A "consistent donation" is one in which every object is from the same Chenhall category. Find all donations that were consistent, yet more that one person catalogued object(s) from it. For each qualifying donation, include in the result one tuple for each person who catalogued objects from it. Each tuple should contain the donation ID and person's staff member ID.

7. Context: Cataloguing a very large donation takes a great deal of diligence. We want to identify and reward staff who have done so.

   Query: The "largest donation" is the one with the most objects in it (there could be ties for largest). Find any donor who made a donation that qualifies as largest. Report the ID of each staff member who has catalogued an object that was part of any donation by any of these donors.

8. Context: We want to identify staff who are not using tags enough, so that we can provide additional training.

   Query: Find staff who have, two or more times, catalogued every object in a donation but gave none of them tags. Report only the SID of these staff members.

9. Context: We don't want staff isolating themselves.

   Query: Find all pairs of staff members who have catalogued together yet neither of them has catalogued with anyone else. (Two people have "catalogued together" if they have each catalogued object(s) from the same donation.) Put each pair into a tuple that includes their staff IDs and their email addresses. Don't include pseudo-duplicates, that is, don't report A, B and also B, A.

10. Context: We are exploring the categorization system.

    Query: A Chenhall category is "complete" if it has at least one primary term and each of its primary terms has at least one secondary term. Find all Chenhall categories that are not complete.

Part 2: Additional Integrity Constraints

Express the following integrity constraints with the notation $R = \emptyset$, where R is an expression of relational algebra. You are welcome to define intermediate results with assignment. The last step for each question must be a single assertion of the form expression $= \emptyset$.

Remember that at least one of the queries and/or integrity constraints in this assignment cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed".

1. No secondary term can be a primary term or a Chenhall category, and no primary term can be a Chenhall category.

2. A donation can be catalogued only if (a) it has one object, which is worth at least $100, or (b) it has two or more objects, which are worth at least $150 in total.

3. Each object catalogued before 2018 must have exactly three tags.

4. There are strict rules for who is allowed to catalogue what:

Staff type	Chenhall category		
	"personal artifacts"	"architectural"	any other category
temp	✗	✗	✗
volunteer	✓	✓	✗
intern	✓	✓	✗
permanent	✓	✓	✓

Formatting 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.

If you use software that lets you choose a font size, it must be at least 10. If you use LaTeX, the default font size (or larger) is acceptable.

Submission instructions

You must declare your team (whether it is a team of one or two students) and hand in your work electronically using the MarkUs online system. Instructions for doing so are 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 pair, only one of you should hand it in.

Check that you have submitted the correct version of your file by downloading it from MarkUs; new files will not be accepted after the due date.

How your assignment will be marked

Most of the marks will be for the correctness of your answers. However, there will be additional marks allocated to each of these:

- Comments:
  Does every assignment statement have a comment above it specifying clearly exactly what rows get to be in this relation? Comments should describe the data (*e.g.*, "The student number of every student who has passed at least 4 courses.") not how to find it (*e.g.*, "Find the student numbers by self-joining ..."). Put comments *before* the assignment, and two dashes on each line of your comment.
- Attribute names given on the LHS:
  Does every assignment statement name the attributes on the LHS? This should be done even if the names are not being changed. Together with the comments, it allows you to understand what a "subquery" is supposed to do without reading it. Think of this as analogous to good parameter names and good comments on a function.

- Relation and attribute names:
  Does every relation and every attribute have a name that will assist the reader in understanding the query quickly? Apply the same high standards you would when writing code.

- Formatting:
  Is the algebra formatted with appropriate line breaks and indentation to reveal the structure of the algebra for ease of understanding?