CSC343 Winter 2023
Assignment #1: Relational Algebra
Due Thursday February 9 by 5:00pm

In this assignment you will be working with a schema for a database to manage waste collection for the City of Toronto. Waste collection for a large city is a complex task, involving collection routes, different types of trucks, waste processing facilities, and people to drive and maintain the equipment.

Learning Goals

This assignment aims to help you learn to:

- Read a relational schema and consider whether or not instances of the schema are valid
- Apply the techniques from class for writing queries and integrity constraints
- Combine techniques to solve complex problems
- Identify the limits of what can be expressed in relational algebra

Formatting Instructions

Your assignment must be typed to produce a PDF document named **a1.pdf** (hand-written submissions are not acceptable). You may use any word-processing software you like. Many academics use LaTeX, and some use a tool called Overleaf to work on LaTeX documents in the cloud, which makes collaborating with a partner easier. (Note that if you choose to use some kind of online storage of your assignment, make sure it is private to only you and your partner.)

Please use a font size of at least 10 (or larger). The default LaTeX font size is acceptable.

Regardless of which tool you use, make sure you leave yourself enough time to type up your answers. **Formatting may take longer than you expect!**

Submission Instructions

You may work on the assignment in groups of 1 or 2, and submit a single assignment for the entire group on MarkUs. Partners do not have to be in the same section, but must be in CSC343 at St. George this semester. You must establish your group well before the due date. Instructions on how to form a group can be found here. No changes may be made to groups after Monday, February 6.

You declare a group by submitting an empty, or partial, file, and this should be done well before the due date. You may always replace such a file with a better version, until the due date.

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. If you are working with a partner, only one of you needs to upload the file for your group.

Relations

- WasteType(wasteType)

  A tuple in this relation represents a type of collectable waste, specified by *wasteType*.

- TruckType(truckType, wasteType)

  A tuple in this relation represents a type of truck *truckType* and the waste type it can collect (*wasteType*).

- Truck(tID, truckType, capacity)

  A tuple in this relation represents a waste collection truck. *tID* is the truck ID, *truckType* is type of truck, and *capacity* is the maximum volume of waste that this truck can hold.

- Facility(fID, address, wasteType)

  A tuple in this relation represents a waste collection facility (i.e. where trucks empty their loads after a trip). Each facility has an ID *fID*, is located at *address*, and collects waste of type *wasteType*.

- Employee(eID, name, hireDate)

  A tuple in this relation represents an employee with employee id *eID*, named *name*, who was hired on *hireDate*.

- Driver(eID, truckType)

  A tuple in this relation represents a driver with employee id *eID* who can drive trucks of type *truckType*.

- Technician(eID, truckType)

  A tuple in this relation represents a technician with employee id *eID* who can maintain trucks of type *truckType*.

- Maintenance(tID, eID, date)

  A tuple in this relation represents maintenance on truck *tID* that was performed by the employee with employee ID *eID* on *date*.

- Route(rID, wasteType, length)

  A tuple in this relation represents a scheduled waste collection route. *rID* is the route ID, *wasteType* is the type of waste collected on this route, and *length* is the distance in kilometres that must be driven for this route.

- Stop(address, rID, assistance)

  A tuple in this relation represents a stop on a collection route (e.g. a house or apartment building). *address* is the street address of the stop, *rID* is the route the stop is on, and *assistance* is a boolean value that indicates whether or not the resident of that stop requires assistance with bringing their waste to the curb. (The City of Toronto has such an option for people with mobility challenges.)

- Trip(rID, tID, date, volume, eID1, eID2, fID)

  A tuple in this relation represents a collection trip made by a truck *tID* on route *rID* with drivers *eID1* and *eID2*, collecting *volume* amount of waste which was taken to facility *fID*, and occurring on *date*.

Constraints

Referential Integrity Constraints

Note that some of the constraints below are bi-directional. We have written $=$ instead of $\subseteq$ and $\supseteq$ for brevity.

- WasteType[wasteType] $=$ TruckType[wasteType]
- Truck[truckType] $\subseteq$ TruckType[truckType]
- WasteType[wasteType] $=$ Facility[wasteType]
- Driver[eID] $\subseteq$ Employee[eID]
- Driver[truckType] $\subseteq$ TruckType[truckType]
- Technician[eID] $\subseteq$ Employee[eID]
- Technician[truckType] $\subseteq$ TruckType[truckType]
- Maintenance[tID] $\subseteq$ Truck[tID]
- Maintenance[eID] $\subseteq$ Technician[eID]
- Route[wasteType] $\subseteq$ WasteType[wasteType]
- Route[rID] $=$ Stop[rID]
- Trip[rID] $\subseteq$ Route[rID]
- Trip[tID] $\subseteq$ Truck[tID]
- Trip[eID1] $\subseteq$ Driver[eID]
- Trip[eID2] $\subseteq$ Driver[eID]
- Trip[fID] $=$ Facility[fID]

Other Constraints

C1. $\sigma_{\text{eID1} \geq \text{eID2}}\text{Trip} = \emptyset$

C2. $\Pi_{\text{assistance}}\text{Stop} \subseteq \{\text{'yes', 'no'}\}$

C3. $(\sigma_{\text{hireDate} > \text{today}}\text{Employee}) \cup (\sigma_{\text{date} > \text{today}}\text{Maintenance}) \cup (\sigma_{\text{date} > \text{today}}\text{Trip}) = \emptyset$

C4. TripsMaintenancePairs(rID, tID, date) :=

$$\Pi_{\text{rID,Trip.tID,Trip.date}}(\sigma_{(\text{Trip.date} - \text{Maintenance.date}) < 100}(\text{Trip} \bowtie_{\text{Trip.tID = Maintenance.tID}} \text{Maintenance}))$$

$(\Pi_{\text{rID, tID, date}}\text{Trips}) - (\Pi_{\text{rID, tID, date}}\text{TripsMaintenancePairs}) = \emptyset$

C5. $\Big(\sigma_{\text{hireDate} > \text{date}}(\text{Trip} \bowtie_{\text{Trip.eID1 = Employee.eID}} \text{Employee}) \cup$

$$\sigma_{\text{hireDate} > \text{date}}(\text{Trip} \bowtie_{\text{Trip.eID2 = Employee.eID}} \text{Employee}) \cup$$

$$\sigma_{\text{hireDate} > \text{date}}(\text{Maintenance} \bowtie \text{Employee})\Big) = \emptyset$$

C6. $\text{TripWithCapacity(tID, volume, capacity)} := \Pi_{\text{tID, volume, capacity}}(\text{Trip} \bowtie \text{Truck})$

$\sigma_{\text{volume}>\text{capacity}}\text{TripWithCapacity} = \emptyset$

C7 . $\text{TruckWasteType(tID, wasteType)} := \Pi_{\text{tID, wasteType}}(\text{Truck} \bowtie \text{TruckType})$

$\text{MatchingWasteType(rID, tID, date)} := \Pi_{\text{rID, tID, date}}\,(\text{Trip} \bowtie \text{Route} \bowtie \text{TruckWasteType} \bowtie \text{Facility})$

$(\Pi_{\text{rID, tID, date}}\text{Trip}) - \text{MatchingWasteType} = \emptyset$

C8. $\text{TripTruckType(rID, tID, date, eID1, eID2, truckType)} := \Pi_{\substack{\text{rID, tID, date,}\\ \text{eID, eID2, truckType}}}(\text{Trip} \bowtie \text{Truck})$

$\text{MatchingTruckType(rID, tID, date)} :=$

$\Pi_{\text{rID, tID, date}}(\sigma_{\text{eID1=eID} \vee \text{eID2=eID}}(\text{TripTruckType} \bowtie \text{Driver}))$

$(\Pi_{\text{rID, tID, date}}\text{Trip}) - \text{MatchingTruckType} = \emptyset$

C9. $\text{MaintainedTruckType(eID, truckType)} := \Pi_{\text{eID, truckType}}(\text{Maintenance} \bowtie \text{Truck})$

$\text{MaintainedTruckType - Technician} = \emptyset$

C10. $\text{TripPairs(tID1, date1, eID1\_1, eID2\_1, tID2, date2, eID1\_2, eID2\_2)} :=$

$$\Pi_{\substack{\text{T1.tID, T1.date, T1.eID1, T1.eID2,}\\ \text{T2.tID, T2.date, T2.eID1, T2.eID2}}}\sigma_{\substack{\text{(T1.date = T2.date)}\\ \wedge \\ \text{(T1.eID1 = T2.eID1} \vee \text{T1.eID1 = T2.eID2} \vee\\ \text{T1.eID2 = T2.eID1} \vee \text{T1.eID2 = T2.eID2)}}}(\rho_{T1}\text{Trip} \times \rho_{T2}\text{Trip})$$

$\text{TripPairs} - \left[\sigma_{\text{tID1 = tID2} \wedge \text{eID1\_1 = eID1\_2} \wedge \text{eID2\_1 = eID2\_2}}\text{TripPairs}\right] = \emptyset$

Part 0: Understanding Our Constraints (not for marks)

TASK: Write out the constraints C1 to C10 above in words. This is not for marks, but we strongly recommend you complete it to help you better understand the schema you will work with for the rest of the assignment.

Part 1: Violating Our Constraints

TASK: For each of the constraints C8, C9, and C10, give one instance of the relevant relations (i.e. the ones involved in the constraint) that would *violate* the constraint. All the relations in your instances must contain at least one tuple, and must satisfy the constraints of other relations that are involved in the constraint. (Exception: You do not need to satisfy the additional constraints listed in Part 3 of the assignment.)

For example, the constraint C7 involves the relations Trip, Route, Facility, Truck, and TruckType. Here is an instance of those relations that violates C7 (but that satisfies other constraints on those particular relations).

Trip

rID	tID	date	volume	eID1	eID2	fID
1	11	2021-02-02	10	111	123	9

Route

rID	wasteType	length
1	recycling	100

Truck

tID	truckType	capacity
11	A	10

Facility

fID	address	wasteType
9	123 Main St	recycling

TruckType

truckType	wasteType
A	compost

Part 2: Queries

TASK: Write relational algebra expressions for each of the queries below. You must use notations from this course and operators:

$$\pi, \sigma, \rho, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, :=$$

You may also use constants:

today (for current date) $\emptyset$ (for the empty set)

In your queries pay attention to the following:

- Do not make assumptions that are not enforced by our constraints above in Part 1. In particular, do not assume the constraints from Part 3. Your queries should work correctly for any database that obeys our schema and constraints.
- Your selection conditions can use arithmetic operators, such as $+, -, \times, /, \leq, =, \neq, \geq, >, <$. You can use logical operators such as $\vee, \wedge$, and $\neg$, and treat dates and numeric attributes as numbers that you can perform arithmetic on. Difference of two dates produces an integer number of days, and adding/subtracting an integer to a date produces a new date the appropriate number of days later/earlier. Dates are comparable.

- Provide good comments to explain your intentions. Use meaningful variable names that include attributes in brackets, (e.g. RouteID(rID) := ...). See the section on marking for more details on query formatting requirements.
- There may be a query or queries that cannot be expressed in the relational algebra you have been taught so far, in which case just write "cannot be expressed". In this case, you will not write the query (since it is not possible).

The queries below are not in any particular order.

- **Q1**: Find the trip(s) where the two drivers were hired at least 1000 days apart. Report the dates and both eIDs from those trips.
- **Q2**: Find the longest serving driver(s) from the ones that have a trip on every route. Report the eID of those drivers.
- **Q3**: Find all drivers who have never had a trip on a route with stops requiring assistance. Report their eIDs.
- **Q4**: A *Supported Route* is a route where all stops on the route require assistance. Find all drivers who have been on at least two trips on all Supported Routes. Report the eIDs of the drivers and the rIDs of the routes.
- **Q5**: Find all drivers who have been on every trip collecting recycling at 40 St George Street. Report their eIDs and names.
- **Q6**: Find the route(s) that have the maximum number of truck trips on any single day. Report the rIDs of these routes.
- **Q7**: Find all trucks that collected exactly one waste type in the past 7 days (i.e. $(\text{today} - \text{date}) \leq 7$). Report the tIDs of those trucks.
- **Q8**: Find the average collection volume for each truck over all of its trips. Report the tID of the truck(s) whose average is closest to its capacity.
- **Q9**: Find the facility that collected the most waste total to date. Report the fID of that facility.
- **Q10**: Find the opening date for each facility (we'll assume the first trip to that facility is when it opened). Report the address and opening date for each facility.
- **Q11**: Find the trucks that are currently in need of maintenance (i.e. it has been > 100 days since their last maintenance). Report the eIDs of the technicians who can maintain them.
- **Q12**: Find all trucks that have been maintained by an employee who has driven that same truck on a trip. Report the tIDs and eIDs.
- **Q13**: Find the truck(s) that were on the most routes (but not necessarily the most trips) in the last 7 days. Report tIDs, rIDs, and waste types for those routes.
- **Q14**: Find all trucks that have collected exactly 3 different waste types. Report the tIDs and truckTypes of these trucks.
- **Q15**: Two routes are considered equivalent if they have the exact same stop addresses (ignore assistance for this) and collect the same type of waste. Report the rIDs, tIDs, and dates for all pairs of trips on equivalent routes. Do not report pseudo-duplicates.

Part 3: Your Constraints

TASK: For each constraint below, derive a relational algebra expression of the form $R = \emptyset$, where R may be derived in several steps, by assigning intermediate results to a variable. For each constraint, indicate if adding that constraint to the schema would change the result of one of the queries above. e.g. "This constraint would change the result of Q1" or "This constraint would not change any queries."

If the constraint cannot be expressed in the relational algebra you have been taught, write "cannot be expressed". (It is implied that constraints that cannot be expressed will not change any queries.)

1. No route has more than 2 stops requiring assistance.
2. An employee is either a driver or a technician, and not both.
3. Every stop address is on at least one route for every waste type.
4. All employees who are drivers can drive at least two truck types.
5. No route has a total volume of more than 1000 on any given date across all trips on that route.
6. No facility has the same address as a stop.

Marking

We will be marking your submissions for correctness, which will make up the majority of the marks. However, we will also be looking for clarity and readability, considering the following:

- **Comments**: Does every assignment statement include a comment above it specifying exactly what is in this relation? Comments should describe the *data*, rather than *technique*, of your queries. Comments should appear *above* each assignment, and start with two dashes.
- **Attribute names on LHS**: Are attribute names given on the left-hand side (LHS)? This should be done even when the names are not changed. You can consider this like choosing good variable names in 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?

A clear and easy to read submission means we will have an easier time finding ways to give you part marks if you have correctness issues, and you will have an easier time checking over and fixing your work if you keep it clear as you go.