CSC343 Winter 2023
Assignment #1: Relational Algebra

Part 0: Understanding Our Constraints

C1:
On a trip, the first driver cannot have an eid higher than the second driver, and the two drivers cannot be the same driver or have the same eid.

C2:
For each stop on a collection route, the resident can either choose yes or no for the assistance option.

C3:
An employee cannot be hired before the date of hire, maintenance cannot be completed before the date of implementation, and a trip cannot occur before the date of occurrence. These dates cannot be after today because today is the last day.

C4:
All trucks that can be driven to the trip have been maintained for longer than 100 days.

C5:
A employee cannot work on a trip or perform maintenance on a day earlier than he got hired.

C6:
All trucks that can be driven to the trip cannot have the volume of waste exceed the maximum capacity.

C7:
TruckWasteType(tID, wasteType) represents the tid of each truck and the type of waste they are capable to carry.
MatchingWasteType(rID,tID,date) represents a trip on a day and the route that this trip took by a specific truck with its tID and the waste they collected match with at least one facility that is able to collect them.

Every trip that happens can be matched with at least one route, one truck and one facility, there is no unmatched trip. Therefore the waste type along the way should be the same and one trip will only have one waste type.

C8:
There are two works needed for the trip, one of them need to be the driver.

C9:
MaintainedTruckType(eID, truckType) represents the employee id of a technician and the type of truck that he is responsible for.
Every type of truck has a technician to perform the maintenance.

C10:
Have exactly same two pair of trip.

Part 1: Violating Our Constraints

C8:

Trip						
rID	**tId**	**date**	**volume**	**eID1**	**eID2**	**fID**
1	**2**	**2020-03-03**	**10**	**2**	**3**	**10**

Truck		
tID	**truckType**	**capacity**
2	**B**	**10**

Driver	
eID	**truckType**
1	**B**
2	**A**

3	A

C9:

TruckType	
truckType	wasteType
A	compost
B	organic

Truck		
tID	truckType	capacity
1	B	10

Technician	
eID	truckType
1	A

Maintenance		
tID	eID	date
1	1	2021-01-15

$$MaintainedTruckType(eid,\ truckType) := \Pi_{eid,\ truckType}(Maintenance \bowtie Truck)$$

MaintainedTruckType	
eID	truckType
1	B

In this case, $MaintainedTruckType - Technician != \emptyset$. Since the technician here can only maintain a type B truck but the truck needs to be maintained is type A. Therefore, the constraint is violated.

C10:

Trip						
rID	**tId**	**date**	**volume**	**eID1**	**eID2**	**fID**
1	**2**	**2020-03-03**	**10**	**1**	**3**	**10**

Trip						
rID	**tId**	**date**	**volume**	**eID1**	**eID2**	**fID**
1	**3**	**2020-03-03**	**10**	**1**	**3**	**10**

Part 2: Queries

Q1:

-- Match eID to two drivers in the trip, select those who hired apart more than 1000 days, either first was hired earlier or second hired earlier.

$$TripDriversHiredApart(rID,\ eID1,\ eID2,\ date) :=$$

$$\Pi_{rID,\ eID1,\ eID2,\ date}$$

$$\sigma_{(Trip.eID1\ =\ e1.eID)\ \wedge\ (Trip.eid2\ =\ e2.eID)\ \wedge\ ((e1.hiredate\ -\ e2.hiredate\ \geq\ 1000)\ \vee\ (e1.hireDate\ -\ e2.hireDate\ \leq\ 1000))}$$

$$(Trip\ \times\ \rho_{e1}Employee\ \times\ \rho_{e2}Employee)$$

$$\Pi_{date,\ eID1,\ eID2}TripDriversHiredApart$$

Q2:

--All drivers on all trips(which include drivers have a trip on every route and drivers have a trip on part of routes)

$$DriverFromTrip(eID, rID) := (\Pi_{eID1,\, rID} Trip) \cup (\Pi_{eID2,\, rID} Trip)$$

--All drivers who went on all routes(ideal best case)

$$ShouldHaveBeen(eID, rID) := (\Pi_{rID} Route) \times (\Pi_{eID} Driver)$$

--Drivers who actually didn't go on all routes

$$WereNotAlways(eID, rID) := \Pi_{eID,\, rID} (ShouldHaveBeen - DriverFromTrip)$$

--Drivers who actually go on trips on all routes

$$DriversOnEveryRoute(eID) := (\Pi_{eID} DriverFromTrip) - (\Pi_{eID} WereNotAlways)$$

--eid and hire dates of all drivers

$$DriverEmployeePair(eID, hireDate) := \Pi_{\text{Drive.eID,hireDate}} (DriversOnEveryRoute \bowtie Employee)$$

--Drivers that are not serving longest

$$NotLongest(eID) := \Pi_{\text{d1.eID}} \sigma_{(d1.eID = d2.eID) \wedge (d1.hireDate < d2.hireDate)} [(\rho_{d1} DriverEmployeePair) \times (\rho_{d2} DriverEmployeePair)]$$

--Drivers that are serving longest

$$Longest(eID) := \Pi_{\text{eID}} (DriversOnEveryRoute - NotLongest)$$

Q3:

-- First find drivers' eID who have a trip with at least one assistance.

$$AtLeastOneAssistance(eID) :=$$

$\Pi_{Driver.eID}$

$\sigma_{((Driver.eID = Trip.eID1) \vee (Driver.eID = Trip.eID2)) \wedge (Stop.assistance = 'yes')}$
$(Driver \times (Trip \bowtie Stop))$

-- The difference between all drives and drivers with at least once to get those never had a trip required assistance.
$\Pi_{eID} Driver - \Pi_{eID} AtLeastOneAssistance$

Q4:
--The route has 1 or more stops that do not need assistance
$NoNeed(rID) := \Pi_{rID} \sigma_{\text{assistance = 'no'}}(Route \bowtie Stop)$

--All stops on routes need assistance
$AllNeed(rID) := \Pi_{rID} Route - \Pi_{rID} NoNeed$

--Drivers of all stops on routes need assistance
$AllNeedDriver(rID, eID) := \Pi_{rID, \text{eID}}$

$\sigma_{(Driver.eID = Trip.eID1) \vee (Driver.eID = Trip.eID2)}$
$(AllNeed \bowtie Driver \bowtie Trip)$

--Drivers of all stops need assistance on at least two trips
$AtLeastTwo(rID, eID) := \Pi_{a1.rID,\ a1.eID}$

$\sigma_{(a1.eID = a2.eID) \wedge (a1.rID \neq a2.rID) \vee (a1.date \neq a2.date)}$
$(\rho_{a1} AllNeedDriver \times \rho_{a2} AllNeedDriver)$

Q5:
-- Get all the information of trips have been on 40 St George Street and collect recycling
$TripPassedStGeorge(rID,\ eID1,\ eID2,\ date) :=$

$\Pi_{Trip.rID,\ eID1,\ eID2,\ date}$

$\sigma_{address = '40\ St\ George\ Street' \wedge wasteType = 'recycling''}$

$(Trip \bowtie Stop \bowtie Route)$

-- Get all drivers who have been on 40 St George Street

$AllDriversInTrip(eID, rID, date) :=$
$(\Pi_{eID, rID, date}$
$(\rho_{t1(rID, eID, eID2, date)} TripPassedStGeorge)) \cup$
$(\Pi_{eID, rID, date}$
$(\rho_{t1(rID,, eID1, eID, date)} TripPassedStGeorge))$

-- Get every possible trip that can happen with different drivers

$EveryPossible(eID, rID, date) := \Pi_{eID} Driver \times$
$\Pi_{rID, date} TripPassedStGeorge$

-- Get the trips that do not appears in actual world

$Missing(eID, rID, date) := EveryPossible - AllDriversInTrip$

-- Get all drivers who have been on every trip collecting recycling at 40 St George Street.

$DriversOnEvery(eID) := \Pi_{eID} AllDriversInTrip - \Pi_{eID} Missing$

Q6:

-- Cannot be expressed.

Q7:

-- Get TruckWasteType from other constraints C7, matches wasteType with trucks.

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

-- Then get all trucks that have a run in the last 7 days and the wasteType they carried in the trip.

$TruckRunInPastSeven(tID, wasteType) :=$
$\Pi_{Trip.tID, TruckWasteType.wasteType}$

$$\sigma_{(today\ -\ Trip.date\ \leq\ 7)}$$
$$(Trip \bowtie\ Route \bowtie TruckWasteType)$$

-- Get all trucks that have a run in the last 7 days and collect at least two different types of waste

$$TruckCollectAtLeastTwo(tID) :=$$
$$\Pi_{t1.tID}$$
$$\sigma_{(t1.wasteType\ \neq\ t2.wastetype)\ \wedge\ (t1.tID\ =\ t2.tID)}$$
$$(\rho_{t1}TruckRunInPastSeven\ \times\ \rho_{t2}TruckRunInPastSeven)$$

-- Finally the difference between trucks that run in the past 7 days and those trucks that collect at least two types of waste in the past 7 days gives the truck that collects only one type of waste in past 7 days.

$$TruckCollectOne(tID) :=$$
$$\Pi_{tID}TruckRunInPastSeven\ -\ \Pi_{tID}TruckCollectAtLeastTwo$$

$$\Pi_{tID}TruckCollectOne$$

Q8: cannot be expressed

Q9:

-- Cannot be expressed. In order to get the total waster collected to date, it requires aggregate function SUM().

Q10:

-- Each trip with its corresponding facility

$$FacilityTripPair(address, date, fID):=\ \Pi_{address,\ date, fID}$$
$$(Facility \bowtie\ Trip)$$

-- Date of Facilities that are not opened

$$Not\ Opened(address, date, fID):=\ \Pi_{f1.address,\ f1.date, f1.fID}$$
$$\sigma_{(f1.fID\ =\ f2.fID)\ \wedge\ (f1.date\ >\ f2.date)}$$
$$(\rho_{f1}FacilityTripPair\ \times\ \rho_{f2}FacilityTripPair)$$

-- Date of Facilities that are opening

$$Opening(address, date) := \Pi_{address,\ date}(FacilityTripPair - Not\ Opened)$$

Q11:

-- First get the trucks which have not been maintained in the past 100 days.

$$NeedMaintenance(tID,\ truckType) := \Pi_{tID,\ truckType} Truck - \Pi_{tID,\ truckType}\, \sigma_{(today - date \leq 100)}(Maintenance \bowtie Truck)$$

-- Get the technicians who are able to maintain those trucks.

$$\Pi_{Technician.eID}(NeedMaintenance \bowtie Technician)$$

Q12:

-- tID and eID of the maintainer

$$Maintainer(tID, eID) := \Pi_{tID, eID}\, Maintenance$$

-- Trucks that are driven and maintained by the same employee and driven to trip

$$SameTruckSame\ Employee(tID, eID) := \Pi_{tID, eID}\, \sigma_{((eID1 = eID)\ \vee\ (eID2 = eID))\ \wedge (Maintainer.tID = Trip.tID)}(Maintainer \times Trip)$$

Q13:

-- Cannot be expressed. Count the most frequent routes that have been visited in the last 7 days require to use the aggregate function COUNT().

Q14:

-- ID and the waste type they have collected before of all trucks.

$$TripTruckWasteType(tID, wasteType) := \Pi_{tID, wasteType}(Trip \bowtie Route)$$

-- Trucks have collected at least four types of waste.

$$AtLeastFour(tID, truckType) := \Pi_{t1.tId}$$

$$\sigma_{(t1.tID = t2.tID = t3.tID = t4.tID) \wedge}$$

$$(t1.wasteType < t2.wasteType < t3.wasteType < t4.wasteType)$$

$$[(\rho_{t1} TripTruckWasteType) \times (\rho_{t2} TripTruckWasteType) \times (\rho_{t3} TripTruckWasteType) \times (\rho_{t4} TripTruckWasteType)]$$

-- Trucks have collected at least three types of waste.

$$AtLeastThree(tID, truckType) := \Pi_{t1.tId}$$

$$\sigma_{(t1.tID = t2.tID = t3.tID) \wedge (t1.wasteType < t2.wasteType < t3.wasteType)}$$

$$[(\rho_{t1} TripTruckWasteType) \times (\rho_{t2} TripTruckWasteType) \times (\rho_{t3} TripTruckWasteType)]$$

-- Trucks have collected exactly three types of waste.

$$ExactlyThree(tID, truckType) := \Pi_{tId}$$

$$(AtLeastThree - AtLeastFour)$$

-- Id and type of Trucks have collected exactly three types of waste.

$$Answer(tID, truckType) := \Pi_{Truck.tId, truckType}$$

$$(ExactlyThree \bowtie Truck)$$

Q15:

-- Get all stops that trip can arrive.

$$AllStops(rID, tID, date, address, wasteType) :=$$

$$\Pi_{Trip.rID, tID, date, address, wasteType}$$

$$(Trip \bowtie Stop \bowtie Route)$$

-- Get all route pairs that have a stop with the same address and same wasteType

$$TripSameStops(rID1, rID2, tID1, tID2, date1, date2, wasteType, address)$$

$$:= \Pi_{s1.rID1, s2.rID, s1.tId, s2.tID, s1.date, s2.date, s1.wasteType, s1.address}$$

$$\sigma_{(s1.wasteType = s2.wasteType) \wedge (s1.address = s2.address)}$$

$(\rho_{s1} AllStops \times \rho_{s2} AllStops)$

-- Create all possible common stops for trip2 where trip1 have been there and have same wasteType.

$TripOneAllStops(rID1, rID2, tID1, tID2, date1, date2, wasteType, address)$
$:= \Pi_{s1.rID, s2.rID, s1.tId, s2.tID, s1.date, s2.date, s1.wasteType, s1.address}$
$\sigma_{s1.wasteType = s2.wasteType}$
$(\rho_{s1} AllStops \times \rho_{s2} AllStops)$

-- Create all possible common stops for trip1 where trip2 have been there and have same wasteType.

$TripTwoAllStops(rID1, rID2, tID1, tID2, date1, date2, wasteType, address)$
$\Pi_{s1.rID1, s2.rID, s1.tId, s2.tID, s1.date, s2.date, s1.wasteType, s2.address}$
$\sigma_{s1.wasteType = s2.wasteType}$
$(\rho_{s1} AllStops \times \rho_{s2} AllStops)$

-- Get the all possible pairs that have a same stop of trip1 and trip2 and have same wasteType

$EveryPossible(rID1, rID2, tID1, tID2, date1, date2, wasteType, address)$
$:= TripOneAllStops \cup TripTwoAllStops$

-- Get the pairs that do not appear in actual world

$Missing(rID1, rID2, tID1, tID2, date1, date2, wasteType, address) :=$
$EveryPossible - TripSameStops$

-- Get all pairs of the equivalent trip

$Equivalent(rID1, rID2, tID1, tID2, date1, date2) :=$
$\Pi_{rID1, rID2, tID1, tID2, date1, date2} TripSameStops -$
$\Pi_{rID1, rID2, tID1, tID2, date1, date2} Missing$

-- Remove pseudo-duplicates, project all pairs.

$\Pi_{rID1, rID2, tID1, tID2, date1, date2} \sigma_{rID1 \leq rID2} Equivalent$

Part 3: Your Constraints

Q1: No route has more than 2 stops requiring assistance.
-- First, get routes which require at least thrice assistance, then get routes that require at most two assistance.

$$AtLeastThriceAssitance(rid) :=$$
$$\Pi_{s1.rID}$$
$$\sigma_{(s1.rID = s2.rID \wedge s1.rId = s3.rID \wedge s1.assitance = 'yes' \wedge s2.assitance = 'yes' \wedge}$$
$$s3.assitance = 'yes' \wedge s1.address \neq s2.address \wedge s1.address \neq s3.address \wedge s2.address \neq s3.address)$$
$$(\rho_{s1}Stop \times \rho_{s2}Stop \times \rho_{s3}Stop)$$

$$AtMostTwiceAssitance(rid) :=$$
$$\Pi_{rID}\, Route - \Pi_{rID}\, AtLeastThriceAssitance$$

$$\Pi_{rID}\, Route - \Pi_{rID} AtMostTwiceAssitance = \emptyset$$

This constraint would not change any queries.

Q2: An employee is either a driver or a technician, and not both.

$$\Pi_{eID}\, Employee - \Pi_{eID}\, Driver - \Pi_{eID}\, Technician = \emptyset$$
$$\Pi_{eID}\, Driver \cup \Pi_{eID}\, Technician = \emptyset$$

This constraint would change the result of Q12.

Q3: Every stop address is on at least one route for every waste type.

$$\Pi_{rID,\, wasteType}\, (Stop \times wasteType) - \Pi_{rID,\, wasteType}\, Route = \emptyset$$

This constraint would not change any queries.

Q4: All employees who are drivers can drive at least two truck types.

$AtLeastTwo(eID) =: \Pi_{d1.eID}\, \sigma_{(d1.eID = d2.eID \wedge d1.truckType \neq d2.truckType)}$
$[(\rho_{D1}Driver) \times (\rho_{D2}Driver)]$
$\Pi_{eID}Driver - AtLeastTwo = \emptyset$
This constraint would not change any queries.

Q5: No route has a total volume of more than 1000 on any given date across all trips on that route.

This constraint would not change any queries. Cannot be expressed.

Q6: No facility has the same address as a stop.
$(\Pi_{address}Stop) \cap (\Pi_{address}Facility) = \emptyset$
This constraint would not change any queries.