csc343 winter 2021
assignment #1: relational algebra
due February 12th, 4 p.m.

Kristin Huang & Eric Zhu

goals

This assignment aims to help you learn to:

- read a relational scheme and analyze instances of the schema
- read and apply integrity constraints
- express queries and integrity constraints of your own
- think about the limits of what can be expressed in relational algebra

Your assignment must be typed to produce a PDF document **a1.pdf** (hand-written submissions are not acceptable). You may work on the assignment in groups of 1 or 2, and submit a single assignment for the entire group on MarkUs. You must establish your group well before the due date by submitting an incomplete, or even empty, submission.

background

You will be working on a schema for a database to track covid-19 vaccinations.[1] Vaccines batches are tracked from the factory that produces them. Their arrival in time Canada, and in the province or territory they are to be administered in, are recorded. Minimum and maximum intervals for follow-up doses are also recorded. There is a unique identifier for each vial.

Patients, vaccine administrators and attendants are each recorded, both to follow up on subsequent doses (where required by the manufacturer), and to track vaccine efficacy and safety. Each patient's covid status at the time of vaccination is recorded, and the time of the latest subsequent infection is recorded. Patients are observed by the attendants for at least 15 minutes after vaccination, and any bad reactions are treated and recorded.

relations

- Batch(<u>bID</u>, mID, productionDate, vialCount)
  Vaccine batch *bID*, manufacturer *mID*, was produced on *productionDate*, with *vialCount* vials in this batch.

[1] Details may not be accurate.

- Vial(vID, bID, thawTime, dose_count)
  Vial *vID* from batch *bID* removed from cold storage at *thawTime*, with *dose_ count* doses remaining.[2]
- Manufacturer(mID, name, thawMax, intervalMin, intervalMax)
  Manufacturer *mID*, with company *name*, *thawMax* maximum hours vaccine is usable after being removed from cold storage, *intervalMin* minimum days to second dose, *intervalMax* maximum days to second dose (both zero for a single-dose vaccine).
- Tracking(bID, canadaDate, locationDate, locationName)
  Batch *bID* arrived in Canada on *canadaDate*, shipped to province or territory *locationName* on *locationDate*.
- Vaccination(pID, date, vID, adID, atID, reaction, covidStatus)
  Patient *pID* vaccinated on *date* from vial *vID*. The dose was administered by *adID*, the patient was attended by *atID*. At vaccination time the patient had infection status *covidStatus* and reaction to vaccine *reaction*.
- Patient(pID, latestPositiveTest)
  Patient *pID* had most recent positive Covid-19 test on *latestPositiveTest* (00:00:00, January 1st, 1970 if this never happened).
- Staff(sID, pID, specialty)
  Medical staff *sID* is also patient *pID*, and has medical specialty *speciality*.

our constraints

For each of the following constraints give a one sentence explanation of what the constraint implies, and why it is required.

- $\Pi_{pID}Staff - \Pi_{pID}Patient = \emptyset$

  Explanation: All staff are patients (Staff is a subset of Patients). This is required because Patient is all-encompassing, since Staff need to receive vaccination as well.
- $(\Pi_{adID}Vaccination \cup \Pi_{atID}Vaccination) \subseteq \Pi_{sID}Staff$

  Explanation: All the attendants and administrators for vaccinations are staff. This is required to make sure only staff can administer or attend to patients; only trained medical staff should be involved with vaccinations.
- $\Pi_{specialty}Staff \subseteq \{'RN','RPN','MD','Pharmacist'\}$

  Explanation: All staff have either the specialty 'RN', 'RPN', 'MD', or a pharmacist. This is required to categorize all Staff into their respective specialties.
- $\Pi_{pID}Vaccination \subseteq \Pi_{pID}Patient$

  Explanation: Everyone who got a vaccination is a patient. This is required to make sure only patients get vaccinated, and getting vaccinated implies that you are a patient, even if just for the vaccination.
- $\Pi_{bID}Vial - \Pi_{bID}Batch = \emptyset$

  Explanation: Every vial is part of a batch. This is required so that the batches would contain all corresponding vials in the batch as vials manufactured in batches (as mentioned in the background section).

[2] A timestamp of 00:00:00, January 1st, 1970 is recorded for any events that have not happened (yet).

- $\Pi_{covidStatus} Vaccination \subseteq \{'positive', 'negative'\}$

  Explanation: The only covid status possible at the time of vaccination is positive or negative. This is required to reflect real life covid status - only negative and positive.

- $\Pi_{reaction} Vaccination \subseteq \{'true', 'false'\}$

  Explanation: The only possible vaccination reaction is true (bad reaction) or false (good reaction). This is required to reflect the real life situation: either a bad reaction or a good reaction to vaccinations.

- $\Pi_{mID} Batch \subseteq \Pi_{mID} Manufacturer$

  Explanation: Every batch is manufactured by a manufacturer. This is required so that we can trace every batch to the manufacturer that manufactured the batch.

- $\Pi_{bID} Tracking - \Pi_{bID} Batch = \emptyset$

  Explanation: Every tracked batch arriving in Canada can be traced back in total batches. This is required so we can find every tracked batch in Canada back to the original batch data, which can be useful for a variety of tasks like tracking the arrival of vaccines with bad reactions.

- **UPDATED 6/2/21** $\Pi_{vID} Vaccination - \Pi_{vID} Vial = \emptyset$

  Explanation: Every vaccination came from a vial. This is required so we can trace each vaccination back to the vial.

queries

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:

- All relations are sets, and you may only use relational algebra operators covered in Chapter 2 of the course text.
- Do not make assumptions that are not enforced by our constraints above, so your queries should work correctly for any database that obeys our schema and constraints.
- Other than constants such as 23 or "Moderna", a select operation only examines values contained in a tuple, not aggregated over an entire column.
- Your selection conditions can use arithmetic operators, such as $+, \leq, \neq, \geq, >, <$. 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.
- Use good variable names and provide lots of comments to explain your intentions.
- Allow the return of multiple tuples if that is appropriate for your query.

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.” The queries below are not in any particular order.

Answer Note: All date objects used (date, productionDate, thawTime, thawMax etc.) are specified to the second and automatically converted for algebraic purposes.

1. Rationale: Let's see how well we're doing.

   Query: Find pID of all patients who have received all required doses since the beginning of December 2020.

   Query 1.1 Answer:

   Cannot be expressed.

   Query 1.1 cannot be expressed because a patient can receive an unlimited number of vaccines. Hence, a patient can receive 1, 3, 5, 7, 9,... etc. number of 2-dose vaccines, so they have not 'received all required doses' since they need an even number of doses to complete 2-dose vaccination. Because relational algebra has no way of aggregating the number of vaccines and detecting even or odd, we cannot express this query.

   Query: Find the names of all provinces or territories that have used vaccine from every manufacturer in their vaccinations.

   Query 1.2 answer:
   $locationManu := \pi_{mID,locationName}(Tracking \bowtie \pi_{mID,bID}(Batch \bowtie \pi_{bID}(Vial \bowtie \pi_{vID}Vaccination)))$
   – Table of location and manufacturer id of all vaccines used in vaccinations
   $allCombos := \pi_{mID,locationName}(\pi_{locationName}Tracking \times \pi_{mID}Manufacturer)$
   – Table of location and manufacturer id of all possible vaccines delivered but not necessarily used
   $allLocations := \pi_{locationName}Tracking$
   – Table of all locations
   $allLocations - (\pi_{locationName}(allCombos - locationManu))$
   – Returns the difference of all locations and the locations that have not used vaccines from all manufacturers

2. Rationale: Let's see how badly we're doing.

   Query: Find pID of all patients who are still waiting for a subsequent dose more than the maximum number of days recommended by the manufacturer.

   Query 2.1 Answer:

   Cannot be expressed.

   Query 2.1 cannot be expressed because a patient can receive an unlimited number of vaccines. Hence, a patient can receive 1, 3, 5, 7, 9,... etc. number of 2-dose vaccines, which would require them to 'wait for a subsequent dose' from the manufacturer. Since relational algebra has no way of aggregating the number of vaccines and detecting even or odd, we cannot express this query.

   Query: Find sID of all staff who administered a vaccination from a vial that had thawed longer than recommended by the manufacturer.

   Query 2.2 Answer:

   $VialThawData(thawTime, thawMax, vID) := \Pi_{thawTime,thawMax,vID}((Manufacturer \bowtie Batch) \bowtie Vial)$

   – Table of each vial and their thawTime and thawMax data

   $AdminID(adID) := \Pi_{adID}(\sigma_{date>thawTime+thawMax}((VialThawData \bowtie Vaccination))$

   $\Pi_{sID}(AdminID \bowtie_{AdminID.adID=Staff.sID} Staff)$

– Return the sID of the staff who administered a vaccination from a vial that thawed longer than recommended

Query: Find vID of all vials with 4 or fewer doses used by the time they had exceeded the maximum time recommended by the manufacturer after thawing.

Query 2.3 Answer:

VialUsed(vID) := $\Pi_{vID}(\sigma_{thawTime>00:00:00,January1st,1970}Vial)$

– Vial ID of all vials thawed. We use the date 00:00:00, January 1st, 1970 in accordance with footnote 2.

VacThawTime(vID, date, thawTime) := $\Pi_{Vial.vID,Vaccination.date,Vial.thawTime,Manufacturer.thawMax}(Vial \bowtie (Vaccination \bowtie (Batch \bowtie Manufacturer)))$

– Relation of each vaccination's vial ID, date, vial thaw time, and max time vial usable before thawing

AtLeast5Times(vID, latestDate, thawTime, thawMax) := $\Pi_{v1.vID,v5.date,v1.thawTime,v1.thawMax}(\sigma_{v1.vID=v2.vID\wedge v2.vID=v3.vID\wedge v3.vID=v4.vID\wedge v4.vID=v5.vID)\wedge(v1.date<v2.date\wedge v2.date<v3.date\wedge v3.date<v4.date\wedge v4.date<v5.date)}(\rho_{v1}VacThawTime \times \rho_{v2}VacThawTime \times \rho_{v3}VacThawTime \times \rho_{v4}VacThawTime \times \rho_{v5}VacThawTime)$

– Relation of vials that have been used at least 5 times, containing attributes of vial ID, the 5th time the vial is used, the vial thaw time, and max time vial usable before thawing

AtLeast5BeforeExpire(vID) := $\Pi_{vID}(\sigma_{latestDate<thawTime+thawMax}(AtLeast5Times))$

– Vial IDs that have been used at least 5 times before the vial exceed the maximum thawing time recommended by manufacturer

VialUsed - AtLeast5BeforeExpire

– Returns all the vials that were used, but were not used at least 5 times before the expiration. This is equivalent to all vials with 4 or fewer doses used by time they exceed the maximum

3. Rationale: Trace exposures.

   Query: Staff sID_1 is exposed to covid-positive staff sID_2 if:

   (a) staff sID_2 administered or attended staff sID_1's vaccination,

   (b) staff sID_1 administered or attended staff sID_2's vaccination,

   (c) or if some staff exposed to sID_2 administered or attended sID_1's, or had a vaccination administered or attended by sID_1. vaccination.

   Find sID of all staff exposed to covid-positive staff sID 42.

   Query 3 answer:
   Cannot be expressed. The 3rd condition is recursive, i.e., if we think of exposure as a graph, we must know/query those who have already been exposed in order to find the other connections of exposed staff. This is not possible with what we know about relational algebra.

4. Rationale: Find versatile staff.

   Query: Find all staff who have worked to both administer vaccines and attend patients (not necessarily at the same vaccination).

   Query 4 answer:
   $admin(sID) := \pi_{adID}Vaccination$
   – A table of the staff who have administered vaccines, all staff ids are unique

$attend(sID) := \pi_{atID} Vaccination$

– A table of the staff who have attended vaccines, all staff ids are unique

$admin \bowtie attend$

– Returns a table of the staff whose ids have been used to both administer and attend vaccinations, joined on matching sID

5. Rationale: Quality control.

   Query: Find the staff who gave the most recent Moderna vaccine that had a bad ('true') reaction. Keep ties.

   Query 5.1 Answer:

   ModernaVaccine(date, adID) := $\Pi_{productionDate,adID}(\sigma_{Manufacturer.name='Moderna' \wedge reaction=true}(((Manufacturer \bowtie Batch) \bowtie Vial) \bowtie Vaccination)$

   – All Moderna vaccinations with a bad('true') reaction

   NotMaxDate(adID) := $\Pi_{M1.adID}(\sigma_{M1.date<M2.date}(\rho_{M1} ModernaVaccine) \times (\rho_{M2} ModernaVaccine))$

   – All Moderna vaccine with a bad('true') reaction that is not the most recently produced vaccine

   $\Pi_{ModernaVaccine.adID}(ModernaVaccine) - NotMaxDate$

   – Subtracts the non-recently produced vaccines from all vaccines. Returns the most recently produced Moderna vaccine with a bad ('true') reaction.

   Query: Find all patients who did not have a positive covid status when they were vaccinated in Ontario, but did have a positive test at some later date (possibly in a different province or territory).[3]

   Query 5.2 Answer:

   $vialLocation(vID, locationName) := \Pi_{vID,locationName}(Tracking \bowtie Vial)$

   – a table of all vials and the name of the Canadian province they arrived at

   $vaccinatedOntario(date, pID, covidStatus) := \Pi_{date,pID,covidStatus}(Vaccination \bowtie (\sigma_{locationName='Ontario'} vialLocation)$

   – all the patients who were vaccinated in Ontario, the date of their vaccination, and their covid status

   $negCovidOntario(date, pID) := \Pi_{date,pID}(\sigma_{covidStatus='negative'} vaccinatedOntario)$

   – all the patients who did not have a positive covid status when they were vaccinated in Ontario, and the date of their vaccination

   $\Pi_{pID}(\sigma_{(Patient.latestPositiveTest>N1.date) \wedge (Patient.pID=N1.pID)}(\rho_{N1} negCovidOntario) \times Patient)$

   – return all the patients that had a positive test after their vaccination in Ontario

your constraints

For each of these constraints you should 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. If the constraint cannot be expressed in the relational algebra you have been taught, write "cannot be expressed."

1. No vial is in two different batches.

   $\rho_{v1}(Vial) \bowtie_{v1.vID=v2.vID \wedge v1.bID>v2.bID} \rho_{v2}(Vial) = \emptyset$

   – Theta joining the "Vial" table on itself on matching vIDs, where the all those vials coming from different batches via bID leads to the empty set.

[3] Not that we are advocating inter-provincial travel at this poine.

2. No patient receives vaccines from two different manufacturers.

   $patientVaccBatch := (Batch \bowtie (\pi_{vID,bID} Vial \bowtie \pi_{pID,vID} Vaccination))$

   – A table of consisting of attributes of batches, manufacturer ids, vials ids, and patient ids that have been used in vaccinations

   $patientVaccManufac := \pi_{pID,mID,vID} patientVaccBatch$

   – A cleaner table consisting of only patient ids, manufacturer ids, and vial ids from $patientVaccBatch$

   $\rho_{p1}(patientVaccManufac) \bowtie_{p1.pID=p2.pID \wedge p1.mID>p2.mID} \rho_{p2}(patientVaccManufac) = \emptyset$

   – Theta joining $patientVaccManufac$ on itself to create a set of patients having vaccines from two different manufacturers. We let it equal the empty set to show that this is not possible with our constraint.

3. No patient is vaccinated with more than two doses.

   $\sigma_{v1.pid=v2.pid=v3.pid \wedge (v1.date>v2.date) \wedge (v2.date>v3.date)}(\rho_{v1} Vaccination \times \rho_{v2} Vaccination \times \rho_{v3} Vaccination) = \emptyset$

   – Theta joining the "Vial" table on itself on matching $pIDs$, so if the same patient got three vaccinations from the different dates, it leads to an empty set.

4. All staff receive at least one vaccination dose before they either administer, or attend, vaccinations.

   $AdministerStaff(giveDate, sID) := \Pi_{date,adID} Vaccination$
   $AttendStaff(giveDate, sID) := \Pi_{date,atID} Vaccination$
   $StaffGiveVaccine(giveDate, sID) := AttendStaff \cup AdministerStaff$

   – a table of all staffs that have attended or administered vaccines, and the corresponding dates

   $StaffVaccinated(givedate, receivedate, sID) := \Pi_{givedate,date,sID}(Vaccination \bowtie (StaffGiveVaccine \bowtie Staff))$

   – a table of the staff's dates when they administered/attended vaccinations, dates of when they received vaccinations, and the staff's sID

   $NotMinVacDate(givedate, receivedate, sID) := \Pi_{givedate,S1.receivedate,sID}(\sigma_{S1.receivedate>S2.receivedate}(\rho_{S1} StaffVaccinated \times \rho_{S2} StaffVaccinated))$

   MinVacReceived(givedate, receivedate, sID) := StaffVaccinated - NotMinVacDate

   – a table of all staffs, the dates they gave vaccine, and the earliest date they received vaccines

   $\sigma_{receivedate>givedate} MinVacReceived = \varnothing$

   – The empty set applies when staff received no vaccination (so MinVacReceived would be empty) or when staff first receives vaccination after they administer or attend to patients. Setting it to the empty set constraints staff to receive at least one vaccination before administering/attending to patients.

5. No vaccine is administered before it arrives in some Canadian territory or province.

   $AllVaccineKeys(pID, date) := \Pi_{pID,date} Vaccination$

   – All the vaccines administered and their dates

   $GoodVaccineKeys(pID, date) := \Pi_{pID,Vaccination.date}(\sigma_{Tracking.locationDate<Vaccination.date}(Vaccination \bowtie (Vial \bowtie Tracking)))$

   – The set of good vaccines that are administered after they arrive in a Canadian territory or province. This is a subset of all the vaccines, so it must be a subset of AllVaccineKeys

$AllVaccineKeys - GoodVaccineKeys = \emptyset$

– This implies AllVaccineKeys is a subset of GoodVaccineKeys. Since earlier we said GoodVaccineKeys is a subset of AllVaccineKeys, then AllVaccineKeys must be equivalent to GoodVaccineKeys. This constraints all vaccines to be administered after arriving to a Canadian province.

submissions

Submit **a1.pdf** on MarkUs. One submission per group, whether a group is one or two people. 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.

Double check that you have submitted the correct version of your file by downloading it from MarkUs.

marking

We mark your submission for correctness, but also for good form:

- For full marks you should add comments to describe the *data*, rather than *technique*, of your queries. These may help you get part marks if there is a flaw in your query.
- Please use the assignment operator, ":=" for intermediate results.
- Name relations and attributes in a manner that helps the reader remember their intended meaning.
- Format the algebraic expressions with line breaks and formatting that help make the meaning clear.