Assignment 1

February 12, 2021

Part 1, our constraints

- $\Pi_{pID}Staff - \Pi_{pID}Patient = \emptyset$
  The constraint implies that attribute patient pID in relation Staff is a subset of the attribute patient pID in relation Patient.
  It's required because it indicates that the medical staffs are a part of people who have got Covid-19 test, same as the other people who got test but are not medical staffs in relation Patient. We can do the natural join on the ralations Staff and Patient.

- $(\Pi_{adID}Vaccination \cup \Pi_{atID}Vaccination) \subseteq \Pi_{sID}Staff$
  The constraint implies that the union of people who administer doses and people who attend patients is the subset of attribute sID in relation Staff.
  It's required because we need to know that the sID in relation Staff contains adID and atID in relation Vaccination. We can do select join on the relations Vaccination and Staff.

- $\Pi_{specialty}Staff \subseteq \{'RN', 'RPN', 'MD', 'Pharmacist'\}$
  The constraint implies that the medical specialty each medical staff has is one of the string in the set of {'RN', 'RPN', 'MD', 'Pharmacist'}.
  It's required because we are able to know that the attribute specialty in relation Staff is a string tuple, and also the possible values of specialty.

- $\Pi_{pID}Vaccination \subseteq \Pi_{pID}Patient$
  The constraint implies that the attribute pID in relation Vaccination is a subset of the attribute pID in relation Patient.
  It's required because we are able to know that not all patients in relation Patient have vaccinated. We can do natural join on the relations Vaccination and Patient.

- $\Pi_{bID}Vial - \Pi_{bID}Batch = \emptyset$
  The constraint implies that the attribute bID in relation Vial is a subset of the attribute bID in relation Batch.
  It's required because we are able to know that the vials in relation Vial come from the batches in relation Batch. We can do the natural join on the relations Vial and Batch.

- $\Pi_{covidStatus}Vaccination \subseteq \{'positive', 'negative'\}$
  The constraint implies that at vaccination time the patient had infection status either is positive or negative.
  It's required because we are able to know that the attribute covidStatus in relation Vaccination is a string tuple, and also the possible values of covidStatus.

- $\Pi_{reaction}Vaccination \subseteq \{'true', 'false'\}$
  The constraint implies that the reaction of patient to vaccine is either true or false.
  It's required because we are able to know that the attribute reaction in relation Vaccination is a string tuple, and also the possible values of reaction.

- $\Pi_{mID}Batch \subseteq \Pi_{mID}Manufacturer$
  The constraint implies that the attribute mID in relation Batch is a subset of the attribute mID in relation Manufacturer.
  It's required because we are able to know that the natural join is available on relations Batch and Manufacturer.

- $\Pi_{bID} Tracking - \Pi_{bID} Batch = \emptyset$
  The constraint implies that the attribute bID in relation Tracking is a subset of the attribute bID in relation Batch.
  It's required because we are able to know that not all vaccine batches have arrived in Canada. We can do the natural join on the relations Batch and Tracking.

- $\Pi_{vID} Vaccination - \Pi_{vID} Vial = \emptyset$
  The constraint implies that the attribute vID in relation Vaccination is a subset of the attribute vID in relation Vial.
  It's required because we are able to know that all vaccines got administered are part of the relation Vial. We can do the natural join on the relations Vaccination and Vial.

Part 2, queries

1. (a) # It can not be expressed, Because everyone can be injected with multiple vaccines, so we cannot tell whether the patient continues to receive another vaccine or not.

   (b) # First we are going to find all the vial match to the mID
   $FactorOfVial := \Pi_{vID,\ mID}\ (Batch\ \bowtie\ Vial)$

   # Then we combine with Tracking to find location for each used vID
   $LoactionVial := \Pi_{vID,\ mID,\ locationName}\ (Tracking\ \bowtie\ FactorOfVial)$
   $UsedVial := (\Pi_{vID}\ (Vaccination)) \bowtie\ LoactionVial$

   # Last step we will use the all mID times locationName and minus the loaction that doesn't use all vaccine.
   $All := (\Pi_{mID}\ (Manufacture))\ \times\ (\Pi_{locationName}\ (Tracking))$
   $LoactionNotUsedAll := \Pi_{loactionName}\ (All - \Pi_{mID,\ loactionName}\ (UsedVial))$

   # Answer: User all location in Tracking minus the location that don't used all vaccine
   $(\Pi_{locationName}\ (Tracking)) - LoactionNotUsedAll$;

2. (a) # It can not be expressed, same as 1a, we cant tell whether the patient continues to receive another vaccine or not.

   (b) #Get the adID from relation Vaccination who administered a vaccination from a vial that had thawed longer than recommanded by the manufacturer
   $adIDFitWithQuery\ :=\ \Pi_{adID}\ \sigma_{Vial.thawTime\ \neq\ '1970-01-01'\ and\ (Vaccination.date\ -\ Vial.thawTime)\ >\ Manufacturer.thawMax}\ (Vaccination\ \bowtie\ Vial\ \bowtie\ Batch\ \bowtie\ Manufacturer)$

   #Get the sID from relation Staff that is the same as adID from relation adIDFitWithQuery
   $\Pi_{sID}\ \sigma_{Staff.sID\ =\ adIDFitWithQuery.adID}\ (adIDFitWithQuery\ \bowtie\ Staff)$

   (c) # First we find all vID that have used before the time they had exceeded the maximum time recommended by the manufacturer after thawing
   $exceedVid := \Pi_{vID,thawMax,thawTime,date}\ \sigma_{Vial.thawTime\ \neq\ '1970-01-01'\ and\ (today\ >\ (thawMax+thawTime))}$
   $(Vaccination\ \bowtie\ Vial\ \bowtie\ Batch\ \bowtie\ Manufacturer)$

   # Then we find the vID of vials that have been administered at least 5 doses before exceed.
   (In order to save space, assume we can do the comparison in selection like v1.vID = v2.vID = v3.vID = v4.vID = v5.vID)
   $atLeast5\ :=\ \Pi_{v1.vID}\ \sigma_{v1.vID\ =\ v2.vID\ =\ v3.vID\ =\ v4.vID\ =\ v5.vID\ and\ v1.date\ \neq\ v2.date\ \neq\ v3.date\ \neq\ v4.date\ \neq\ v5.date\ and\ v1.pID\ \neq\ v2.pID\ \neq\ v3.pID\ \neq\ v4.pID\ \neq\ v5.pID}\ and(v1.date\ <\ (v1.thawMax\ +\ thawTime))and(v2.date\ <\ (v2.thawMax+v2.thawTime))\ and\ (v3.date < (v3.thawMax+v3.thawTime))\ and\ (v4.date < (v4.thawMax+ v4.thawTime))and(v5.date < (v5.thawMax+v5.thawTime))(\rho_{v1}\ exceedVid \times \rho_{v2}\ exceedVid \times \rho_{v3}\ exceedVid \times \rho_{v4}\ exceedVid\ \times \rho_{v5}\ exceedVid)$

Then we use all exceed vaccine minus the vaccine that had been used at lesat 5 time before it exceed to get our answer.
$(\Pi_{vID}\ exceedVid)\ -\ (\Pi_{vID}\ atLeast5)$

3. (a) # It can not be expressed, since condition (c) will cause a recursion.

4. (a) # First we project all adID that administer vaccines
$administerVacc := \Pi_{adID}\ Vaccination$
Then we porject all atID that attend patients
$attendPati := \Pi_{atID}\ Vaccination$
Finally we get the intersect ID between adID and atID
$(\Pi_{adID}\ administerVacc)\ \cap\ (\Pi_{atID}\ attendPati)$

5. (a) # First the vID that name is moderna and project vID and production date.
$ModernaVial := \Pi_{vID,productionDate}\ \sigma_{Manufacturer.name="Moderna"}\ ((Vial \bowtie\ Batch) \bowtie\ Manufacturer)$

Then we find vID that is not most recent Moderna, and use all moderna minus not most recent get most recent moderna vaccine
$notMostResent := \Pi_{T1.vID}\ \sigma_{T1.producionDate\ <\ T2.productionDate}\ (\rho_{T1}ModernaVial\ \times\ \rho_{T2}ModernaVial)$
$MostResent := (\Pi_{vID}(ModernaVial))$ - notMostResent

last step, find all patient used most recent Moderna vaccine with reaction is true, then use Staff - all patient with moderna get set of staff with not have recent Moderna or reaction = false, finally use Staff minus this set we get all staff id with recent Moderna and true reaction.
$patientModerna := \Pi_{pID}\ \sigma_{reaction="true"}\ (Vaccination \bowtie\ MostResent)$

Answer := $(\Pi_{sID}$Staff) - $((\Pi_{sID}(Staff))$ - patientModerna)

(b) # First we find all pID and dates that the patients who did not have a positive covid status when they were vaccinated in Ontario
$negativeVaccinated := \Pi_{pID,\ Vaccination.date}\ \sigma_{covidStatus\ =\ 'negative'\ and\ locationName\ =\ 'Ontario'}\ (Vaccination \bowtie Vial \bowtie Tracking)$
Finally we find all pID that had a positive test at some later date
$\Pi_{pID}\ \sigma_{negativeVaccinated.date\ <\ Patient.latestPositiveTest}\ (negativeVaccinated \bowtie\ Patient)$

Part 3, your constraints

1. $(\sigma_{v1.vID\ =\ v2.vID\ and\ v1.bID\ \neq\ v2.bID}\ (\rho_{v1}\ Vial\ \times\ \rho_{v2}\ Vial))\ =\ \emptyset$

2. # We first find all pID and manufacturer's names
$pIDAndManufact := \Pi_{Vaccination.pID,\ Manufacturer.name}\ (Vaccination \bowtie Vial \bowtie Batch \bowtie Manufacturer)$
Then we get empty set that the same pID has two different manufacturer's names by product two same relations
$(\sigma_{p1.pID\ =\ p2.pID\ and\ p1.name\ \neq\ p2.name}\ (\rho_{p1}\ pIDAndManufact\ \times\ \rho_{p2}\ pIDAndManufact))\ =\ \emptyset$

3. # First we project all pID and date from relation Vaccine and assign it with name allID
$allID := \Pi_{pID,\ date}\ Vaccination$
Then we cross product three relations allID together and select tuples that have the same pID

$sameID := \sigma_{v1.pID\ =\ v2.pID\ and\ v1.pID\ =\ v3.pID\ and\ v2.pID\ =\ v3.pID}\ (\rho_{v1}\ pIDDate \times \rho_{v2}\ pIDDate \times \rho_{v3}\ pIDDate)$
Finally we get the empty set that no patient is vaccinated with more than two doses by selecting tuples that have three different dates from relation sameID
$(\sigma_{v1.date\ \neq\ v2.date\ and\ v1.date\ \neq\ v3.date\ and\ v2.date\ \neq\ v3.date}\ sameID)\ =\ \emptyset$

4. # First we find vaccination dates and sID of all staff that got vaccinated
$gotVaccinated := \Pi_{sID,\ date}\ (Vaccination\ \bowtie\ Staff)$
Then we find the tuples of multiple times vaccination of each staff in relation gotVaccinated
$multipleTimeVaccinated := \Pi_{g1.sID,\ g1.date}\ \sigma_{g1.sID=g2.sID\ and\ g1.date\ >\ g2.date}$
$(\rho_{g1}\ gotVaccinated\ \times\ \rho_{g2}\ gotVaccinated)$
Then we find the tuples of first time vaccination of each staff in relation gotVaccinated by subtracting multipleTimeVaccinated from gotVaccinated
$firstTimeVaccinated := (\Pi_{sID,\ date}\ gotVaccinated)\ -\ (\Pi_{sID,\ date}\ multipleTimeVaccinated)$
Then we select adID, atID and vaccination dates that the staff administer or attend vaccinations
$gaveVaccination := \Pi_{adID,\ atID,\ date}\ Vaccination$
Then we select all tuples that sID = adID or sID = atID in the cross product between firstTimeVaccinated and gaveVaccination (Here we assume that the attribute date in firstTimeVaccinated and the attribute date in gaveVaccination have different column names so that we can do cross product on two relations)
$commonID := \sigma_{sID\ =\ adID\ \vee\ sID\ =\ atID}\ (firstTimeVaccinated\ \times\ gaveVaccination)$
Finally we get the empty set by comparing the date that the staff received at least one vaccination dose with the date that the staff either administer or attend vaccination
$(\sigma_{firstTimeVaccinated.date\ >\ gaveVaccination.date}\ commonID)\ =\ \emptyset$

5. # First we get vID and vaccinated date of all administered vaccines
$administeredVacc := \Pi_{vID,\ date}\ Vaccination$
Then we get vID and location date of all vaccines that arrives in some Candian territory or province
$arrivedVacc := \Pi_{vID,\ locationDate}\ (Tracking\ \bowtie\ Vial)$
Finally we get the empty set by comparing the arrived location date with the administered date
$(\sigma_{administeredVacc.date\ <\ arrivedVacc.locationDate}\ (administeredVacc\ \bowtie\ arrivedVacc))\ =\ \emptyset$