University of Toronto
CSC343, Fall 2020

# Assignment 1: SIWEI TANG 1004024548 ZHENYU WANG

## Part 1: Queries

1. Context: Bus trips have been falling in popularity. We are reviewing our pricing for these, as well as frequent traveller discounts, and arrangements with resellers like TravelZoo and Groupon.

   Query: For each instance of a trip that includes transportation by bus, find the highest price paid by any traveller for that instance, and find the lowest price paid by any traveller for that instance. Report the trip instance ID and advertised price, the highest price paid, and the lowest price paid.

   – All the trips includes bus transportation.
   $BusTrip(tripID) := \pi_{tripID}\sigma_{transportationType="bus"}Itineary$

   – All the trip instances and advertised price for those tripIDs.
   $BusInstances(instID, price) := \pi_{instID,price}(BusTrip \bowtie TripInstance)$

   – all the paid prices for the trip instances except the highest price
   $NotTop(instID, pricePaid) :=$

   $\pi_{instID,pricePaid}\sigma_{T_1.pricePaid<T_2.pricePaid \wedge T_1.instID=T_2.instID}[(\rho_{T_1}BusInstances \bowtie Booking) \times (\rho_{T_2}BusInstances \bowtie Booking)]$

   – all the paid prices for the trip instances except the lowest price
   $NotBottom(instID, pricePaid) :=$

   $\pi_{instID,pricePaid}\sigma_{T_1.pricePaid>T_2.pricePaid \wedge T_1.instID=T_2.instID}[(\rho_{T_1}BusInstances \bowtie Booking) \times (\rho_{T_2}BusInstances \bowtie Booking)]$

   – Find all price paid for the trip instances
   $AllPaid(instID, pricePaid) := \pi_{instID,pricePaid}(BustInstances \bowtie Booking)$

   – Find the highest price paid for every trip instance
   $HighestPrices(instID, highestPrice) := \rho_{instID,highestPrice}(AllPaid - NotTop)$

   – Find the lowest price paid for every trip instance
   $LowestPrices(instID, lowestPrice) := \rho_{instID,lowestPrice}(AllPaid - NotBottom)$

   – Combine the tables together
   $Answer(instID, price, highestPrice, lowestPrice) :=$

   $BusIntances \bowtie HighestPrices \bowtie LowestPrices$

2. Context: We are reviewing trips that haven't been offered recently, with a plan to rethink whether they might be more appealing if they visited more cities.

   Query: For each trip that has had at least 2 instances but none in 2019 or since, report the trip id and name, and the number of stops on its itinerary.

   – Cannot be expressed

3. Context: The company is planning a special promotion for people who have taken extensive trips to Paris.

   Query: Find all travellers who have booked at least one instance of each trip that starts in Paris and includes 3 or more cities (the start city counts towards this total). Report the traveller's first and last name, and email address.

   Solution:

   –First, find cityID of Paris.
   $ParisID(cityID) := \Pi_{cityID}\sigma_{name='Paris' \wedge region='le-de-France' \wedge country='France'}City$

   –Then, find trips that start with Paris.
   $StartWithParis(tripID) := \Pi_{tripID}(Trip \bowtie ParisID)$

   –Then, find cities that travelled by each trip.
   $CitiesofTrips(tripID, cityID) := \Pi_{tripID,cityID}Itinerary$

   –Find trips that visit at least three cities.
   $AtLeastThreeCities(tripID) :=$

   $\Pi_{D1.tripID}\sigma_{D1.tripID=D2.tripID=D3.tripID \wedge D1.cityID>D2.cityID>D3.cityID}(\rho_{D1}CitiesofTrips \times \rho_{D2}CitiesofTrips \times \rho_{D3}CitiesofTrips)$

   –Find trip that starts with Paris and visit at least three cities.
   $SatifiedTrips := AtLeastThreeCities \cap StartParis$

   –Then, find all trip instances of those trips.
   $SatisfiedTripInstances(tripID, instID) := \Pi_{tripID,instID}(SatifiedTrips \bowtie TripInstance)$

   –Now, find all travellers of these instances.
   $ActualTravellers(travID, tripID) := \Pi_{travID,tripID}(SatisfiedTripInstances \bowtie Booking)$

   –Now, assume that all actual travellers that would have been to all of those trips.
   $WouldHaveBeen(travID, tripID) :=$

   $\Pi_{ActualTravellers.travID,SatifiedTrips.tripID}(ActualTravellers \times SatifiedTrips)$

   –Find travellers that not in all those required trips.
   $WereNotAlways(travID, tripID) := \Pi_{travID,tripID}(WouldHaveBeen - ActualTravellers)$

   – Then, find all travellers that satisifed the requirement.
   $TravellersSatisfied(travID) := \Pi_{travID}(ActualTravellers - WereNotAlways)$

   – Lastly, find the required information for the selected travellers.
   $Answer := \Pi_{fname,lname,email}(TravellersSatisfied \bowtie TravellersSatisfied)$[10pt]

4. Context: We are looking for travellers who like a lot of perqs. We might design a special trip aimed at this market.

   Query: Let's define a plush trip as one with the most base activities. (There could be exactly one plush trip, or perhaps there are several that are tied.) Report the name and email address of all travellers who have taken one or more of these plush trips, have never taken a non-plush trip, and have never booked an add-on activity.

   – Cannot be expressed

5. Context: The company is planning a staff retreat. Some of the activities will be done in pairs, and we'd like to group people who have experiences in common.

   Query: Find all pairs of staff who've both been trip manager for different instances of the same trip. For each pair, report both people's email address, and the start date of the very first trip instance they were manager for. (It might not be an instance of the/a trip they have in common with the other person.) Use attribute names staff1, start1, staff2, start2 and make the person represented by the first two attributes be the one from the pair with the most seniority (that is, the one whose employment date comes first). If there is a tie, it doesn't matter whose data is represented by the first two attributes.
   Solution:
   –First, find all staffs that have been trip manager for different instances of the same trip.
   $ExperiencedManagers(staffID, tripID) :=$

   $\Pi_{T1.tripManager,T1.tripID}\sigma_{T1.instID \neq T2.instID \wedge T1.tripID=T2.tripID \wedge T1.tripManager=T2.tripManager}$
   $(\rho_{T1}TripInstance \times \rho_{T2}TripInstance)$

   –Find all startTime of these staffs that have.
   $AllStartTime(staffID, startDate) := \Pi_{staffID,startDate}(TripInstance \bowtie ExperiencedManagers)$

   – Find trip instances that are not the earliest for each stuff.
   $NotEarliest(staffID, startDate) :=:=$

   $\Pi_{E2.staffID,E2.startDate}\sigma_{E1.startDate<E2.startDate \wedge E1.staffID=E2.staffID}(\rho_{E1}AllStartTime \times$
   $\rho_{E2}AllStartTime)$

   – Then, find earliest start date of those staffs.
   $Earliest(staffID, startDate) := (AllStartTime - NotEarliest)$

   –Find employment date of those pairs.
   $EmploymentDate(staffID, employDate) := \Pi_{staffID,employDate}(Staff \bowtie ExperiencedManager)$

   –Then, find all pairs of those managers that have been on the same trip.
   $ExperiencedManager(staffID, tripID, employDate, startDate) := Earliest \bowtie EmploymentDate \bowtie$
   $ExperiencedManagers$

   – Find All pairs of managers with same EmploymentDate.
   $EqualPair(staff1, start1, staff2, start2) :=$

   $\Pi_{E1.staffID,E1.startDate,E2.staffID,E2.startDate}$
   $\sigma_{E1.staffID>E2.staffID \wedge E1.employDate=E2.employDate \wedge E1.tripID=E2.tripID}(\rho_{E1}ExperiencedManager \times$
   $\rho_{E2}ExperiencedManager)$

   –Find All pairs of managers with different EmploymentDate.
   $DifferentPair(staff1, start1, staff2, start2) :=$

   $\Pi_{E1.staffID,E1.startDate,E2.staffID,E2.startDate}$
   $\sigma_{E1.staffID>E2.staffID \wedge E1.employDate<E2.employDate \wedge E1.tripID=E2.tripID}(\rho_{E1}ExperiencedManager \times$
   $\rho_{E2}ExperiencedManager)$

   –union equal and less than to get the final manager result.
   $Answer(staff1, start1, staff2, start2) := EqualPair \cup DifferentPair$

6. Context: We are designing a new trip for very adventurous travellers and are targeting it towards people who are willing to spend money on add-on activities.

Query: Find travellers who have booked more than one trip instance and on every trip instance they've booked, they booked all of the add on activities available. Report simply the email address of each such traveller.
Solution:
–First, find all travellers that booked more than one trip.
$MoreThanOne(travID) := \Pi_{B1.travID}\sigma_{B1.instId \neq B1.instId \wedge B1.travID = B2.travID}(Booking)$

–Then, find those travellers with add on activities.
$TravelersAddOn(travID, instID, addID) := \Pi_{travId,instID,addID}(AddOnBooking \bowtie MoreThanOne)$

–Then find trips of those actual booked instance, travelers.
$TripsAddOn(travID, instID, addID, tripID) :=$

$\Pi_{travID,instID,addID,tripID}(TripInstance \bowtie TravelersAddOn)$

–Then find trips that would have been if all travelers booked their instances of activities.
$WouldHaveBeen(travID, instID, addID) :=$

$\Pi_{AddOnActivity.travID,AddOnActivity.instID,TripsAddOn.addID}(AddOnActivity \times TripsAddOn)$

– Find travellers that do not book all for some add on activities.
$WereNotAlways(travID, instID, addI) := WouldHaveBeen - TravelersAddOn$

–Find all satisfied travelers.
$SatisfiedTravelers(travID) := \Pi_{travID}WereNotAlways$

–Get required information of those satisfied travelers.
$Answer := \Pi_{email}(SatisfiedTravelers \bowtie Traveller)$

7. Context: We think some of our reviews may be fake.

Query: Let's say a low-ball rating for a trip is one that (a) is at most 2 stars, (b) is the lowest rating (lowest number of stars) for that trip, and (c) is unique for that trip: no one else gave that trip the same number of stars. Find travellers who've given at least one review and whose every review either gives a low-ball rating or has text consisting of just the word "terrible". For each such traveller, report just their email address.
–First, find all reviews with at most 2 stars.
$AtMostTwo(tripID, travID) := \Pi_{tripID,travID}\sigma stars \leq 2Review$

–Then, find not the lowest rating for all trips.
$NotLowestRate(tripID, travID) :=$

$\Pi_{R1.tripID,R1.travID}\sigma_{R1.tripID=R2.tripID \wedge R1.travID \neq R2.travID \wedge R1.stars > R2.stars}(\rho_{R1}Review \times \rho_{R2}Review)$

–Then, find not the lowest rating for all trips.
$LowestRate(tripID, travID) := \Pi_{tripID,travID}(Review) - NotLowestRate$

–Find rating that appears at least twice for each trip.
$AtleastTwice(tripID, travID) :=$

$\Pi_{R1.tripID,R2.travID}\sigma_{R1.stars=R2.stars \wedge R1.tripID=R2.tripID \wedge R1.travID \neq R2.travID}(\rho_{R1}Review \times \rho_{R2}Review)$

–Find unique ratings.
$UniqueRating(tripID, travID) := \Pi_{tripID,travID}(Review) - AtleastTwice$

–Find Low-Ball Ratings.
$LowBallRatings(tripID, travID) := LowestRate \cap AtleastTwice \cap UniqueRating$

–Find reviews only contains Terrible.
$TerribleReview(tripID, travID) := \Pi_{tripID,travID}\sigma_{text='Terrible'}(Review)$

–Find all potential fake reviews.
$FakeReview(tripID, travID) := TerribleReview \cup LowBallRatings$

–Find all reviewers that gives at least one real review.
$RealReviewers(travID) := \Pi_{travID}(\Pi_{tripID,travID}(Review) - FakeReview)$

–Find all reviews always gives fake reviews.
$FakeReviewers(travID) := \Pi_{travID}(Review) - RealReviewers$

–Find all required email address for those reviewers.
$Answer(email) := \Pi_{email}(Traveller \bowtie FakeReviewers)$.

8. Context: We are looking for some social media influencers to expand our sales of trips to South America, so we want to hire a young traveller who has had a positive outlook on our trips to Brazil.

   Query: Find all travellers who've rated a trip that includes Rio de Janeiro and Sao Paulo, and have given that trip a rating that is above the average rating for that trip. Report the traveller email, and first and last name. Include only travellers whose birth year is between 1995 and 2002 inclusive.

   – Cannot be expressed

## Part 2: Additional Integrity Constraints

1. Context: We don't want to book a traveller on an instance of a trip if it's clear that they will not have their accommodation needs met.

   Constraint: A traveller cannot book an instance of a trip unless there is, in every city on its itinerary (including the start city), an accommodation for that trip instance that offers either a room of the size they requested in their booking or a larger room.

   Solution:

   –Room in a city offered by a trip instance (offered).
   –Room in a city offered by a trip instances and satisfiy condition (condition satisfied).
   –Trip instance offered by a travellers.(includeing tripID, travID, instID, startCity)
   –Start city of a trip manager.
   –Find City visited
   –Find City vistied with big rooms.

2. Context: The company is concerned about the legitimacy of reviews.

   Constraint: You can only review a trip if you've booked at least one instance of it. (However, nothing stops you from reviewing a trip before you go on that trip.)
   – All people who take review.
   $TakeReview(tripID, travID) := \pi_{tripID,travID}Review$

   – The trip instances of those people who took review

$Instances(instID) := \pi_{instID}(Booking \bowtie TakeReview)$

– The trip IDs that correspond to the trip instances
$Trips(tripID) := \pi_{tripID}(Instances \bowtie TripInstance)$

–if the travellers who took all not fake review, there should be no difference between the actual trip IDs and review trip IDs.
$Answer := \pi_{tripID} Review - Trips = \emptyset$

3. Context: The company needs to recoup the lost income from discounts by making money on add-on activities.

   Constraint: If a traveller does not book either the most expensive or second-most expensive add-on activity on a trip instance, they can't get a discount price for that trip instance (that is, they can't pay less than the advertised price). This constraint applies only to instances of trips that have two or more add-on activities.
   –get the trip instance IDs that have travellers book add on activities.
   $TripIDs(instID) := \pi_{instID} AddOnBooking$

   – find all trip IDs corresponds to Trip instance IDs.
   $CorreInstances(tripID) := \pi_{tripID}(TripIDs \bowtie TripInstance)$

   –find all add on activities and its cost corresponds to trip IDs.
   $AddCost(addID, tripID, cost) := \pi_{addID,tripID,cost}(CorreInstances \bowtie AddOnActivity)$

   – find the not highest cost add on activities
   $NotTop(addID, tripID, cost) :=$

   $$\pi_{T_1.addID,T_1.tripID,T_1.cost} \sigma_{T_1.cost<T_2.cost \wedge T_1.tripID=T_2.tripID}[(\rho_{T_1} AddCost) \times (\rho_{T_2} AddCost)]$$

   –find the add on activities that are not highest and second highest cost
   $SecondNotTop(addID, tripID, cost) :=$

   $$\pi_{T_1.addID,T_1.tripID,T_1.cost} \sigma_{T_1.cost<T_2.cost \wedge T_1.tripID=T_2.tripID}[(\rho_{T_1} NotTop) \times (\rho_{T_2} NotTop)]$$

   – find all travellers who booked these activities in which instance.
   $BookerInstances(instID, travID, addID) := \pi_{instID,travID}[(\pi_{addID} SecondNotTop) \bowtie AddOnBooking]$

   – Booked at least twice the add on in one instance.
   $AtLeastTwiceBook(instID, travID, addID) :=$

   $$\pi_{T_1.instID,T_1.travID,T_1.addID} \sigma_{T_1.travID=T_2.travID \wedge T_1.addID \neq T_2.addID \wedge T_1.instID=T_2.instID}$$
   $(\rho_{T_1} BookerInstances) \times \rho_{T_2} BookerInstances$

   – based on traveller IDs and trip instance IDs we can find the price paid.
   $Paid(instID, travID, pricePaid) := \pi_{instID,travID,pricePaid}(AtLeastTwiceBook \bowtie Booking)$

   – find all advertised price for those trip instances.
   $AdvertisedPrice(instID, price) := \pi_{instID,price}(TripIDs \bowtie TripInstance)$

   – Compare the price paid and advertised price
   $Answer(travID) = \pi_{travID} \sigma_{pricePaid<price}(Paid \bowtie AdvertisedPrice) = \emptyset$