Name: Rahul Purushottam Gaonkar

Homework-1

Problem Set #1

Problem 1: Consider the following relational schema for a website for fans of live music that keeps track of artists (individual musicians or groups), concerts by artists, and the times and venues (places such as bars, concert halls, etc.) where concerts take place:

Artist (aid, aname, adesc)
ArtistGenre (aid, genre)
Concert (cid, vid, cdate, cstarttime, cendtime, cdesc)
ConcertArtist (cid, aid)
Venue (vid, vname, vaddress, vcity, vphone)
TicketType (cid, ticketclass, ticketcost)

Thus, for each artist we have an aid, a name, and a short description of their music. An artist can also play one or more genres, such as Jazz, Americana, Soul, etc. For each concert, we store where it takes place (the venue), the date, the start and end time, and a short description. A concert may involve several artists. For each venue we store its name, address, and contact phone number. Finally, we store information about the prices of different categories of tickets for each event. Note that the site is not keeping track of individual tickets and who purchased them; it only provides information about upcoming concerts and their ticket prices (probably with hyperlinks to the actual sites selling tickets).

(a) Identify suitable foreign keys for this schema.

Ans: The different foreign keys for the above schema are as follows:

Table Name	Foreign Key
ArtistGenre	aid
Concert	vid
ConcertArtist	cid, aid
TicketType	cid

(b) In the Concert table, if we did not have the attribute “cid” (meaning we only have vid, cdate, cstarttime, cendtime, and cdesc), how would you choose a suitable primary key?

Ans: The suitable primary key would be (vid, cdate, cstarttime)

(c) Write statements in SQL for the following queries.

I. Output the names of all artists that have the genre “Jazz”

Ans: i. Select aname
from Artist, ArtistGenre
where Artist.aid = ArtistGenre.aid and genre = ‘Jazz’

ii. Select aname
from Artist natural join ArtistGenre;
where genre = ‘Jazz’

II. Output all the cities in which artist “Bruno Mars” played in 2016.

Ans: i. Select distinct vcity
from Concert natural join Venue natural join ConcertArtist natural join Artist;
where aname = ‘Bruno Mars’ and year(cdate) = ‘2016’

ii. Select distinct vcity
from Concert, Venue, ConcertArtist, Artist
where Concert.vid = Venue.vid and Concert.cid = ConcertArtist.cid and
ConcertArtist.aid = Artist.aid and aname = ‘Bruno Mars’ and year(cdate) = ‘2016’

III. For each venue, output the vid and name and the number of concerts it hosted in 2016.

Ans: Select vid, vname, count (cid) as number of concerts
from Venue natural join Concert
where year(cdate) = ‘2016’
group by vname, vid

IV. Output the cid, ticketclass, cost, and date for the most expensive tickets ever for a Jazz concert. (A Jazz concert is one where at least one artist has Jazz as a genre.)

Ans: Select cid, ticketclass, ticketcost, date
from TicketType natural join Concert natural join ConcertArtist natural join ArtistGenre
where genre = ‘Jazz’ and ticketcost = (Select max(ticketcost)
from TicketType natural join Concert natural join
ConcertArtist natural join ArtistGenre
where genre=’Jazz’)

V. Output the aids and names of all artists who have appeared together with “Bruno Mars” in at least two concerts during 2016.

Ans: Select aid, aname
from Artist natural join ConcertArtist natural join Concert
where aname != ‘Bruno Mars’ and cid in (Select cid
from Artist natural join ConcertArtist natural join
Concert
where year(cdate) = ‘2016’ and aname = ‘Bruno
Mars’)
group by aid,aname
having count(cid)>=2

VI. Output the aid and name of any artist who has never given a concert in New York City.

Ans: Select aid, aname
from Artist
where aid not in (Select aid
from Artist natural join ConcertArtist natural join Concert natural join venue
where vcity = 'New York')

VII. Output the cid, data, and lowest cost for any Jazz concert in Chicago in October 2017. (Lowest cost means the cost of the cheapest class of tickets over all Jazz concerts.)

Ans: Select cid, cdate, ticketcost
from TicketType natural join Concert natural join ConcertArtist natural join ArtistGenre natural join Venue
where genre = 'Jazz' and vcity = 'Chicago' and year(cdate) = '2017' and month(cdate) = '10' and (cid, ticketcost) = (Select cid, min(ticketcost)
from TicketType natural join Concert natural join ConcertArtist natural join ArtistGenre natural join Venue
where genre = 'Jazz' and vcity = 'Chicago' and year(cdate) = '2017' and month(cdate) = '10'
group by cid)

(d) Write expressions in Relational Algebra for the above queries.

I. Output the names of all artists that have the genre "Jazz".

Ans: i. πaname(σ Artist.aid = ArtistGenre.aid $\wedge$ genre = 'Jazz'(Artist **x** ArtistGenre))

ii. πaname(σ genre='Jazz'(Artist $\bowtie$ ArtistGenre))

II. Output all the cities in which artist "Bruno Mars" played in 2016.

Ans: i. πvcity (σaname = 'Bruno Mars' $\wedge$ year(cdate) = '2016'(Concert $\bowtie$ Venue $\bowtie$ ConcertArtist $\bowtie$ Artist))

ii. πvcity (σ Concert.vid=Venue.vid $\wedge$ Concert.cid=ConcertArtist.cid $\wedge$ ConcertArtist.aid=Artist.aid $\wedge$ aname = 'Bruno Mars' $\wedge$ year(cdate) = '2016' (Concert **x** Venue **x** ConcertArtist **x** Artist))

III. For each venue, output the vid and name and the number of concerts it hosted in 2016.

Ans: vid,vname **G** count(cid) as number of concerts**(**σ year(cdate)='2016'**(**Venue $\bowtie$ Concert))

IV. Output the cid, ticketclass, cost, and date for the most expensive tickets ever for a Jazz concert. (A Jazz concert is one where at least one artist has Jazz as a genre.)

Ans: πcid,ticketclass,ticketcost,date **(**σgenre = 'Jazz'$\wedge$ ticketcost**= (G** max(ticketcost) **(**σgenre='Jazz' (TicketType $\bowtie$ Concert $\bowtie$ ConcertArtist $\bowtie$ ArtistGenre)))(TicketType $\bowtie$ Concert $\bowtie$ ConcertArtist $\bowtie$ ArtistGenre))

V. Output the aids and names of all artists who have appeared together with “Bruno Mars” in at least two concerts during 2016.

Ans:

$T1 \leftarrow \pi_{cid}(\sigma_{year(cdate) = '2016' \wedge aname = 'Bruno\ Mars'}(Artist \bowtie ConcertArtist \bowtie Concert))$

$T2 \leftarrow \pi_{cid,aid,aname}(\sigma_{aname \neq 'Bruno\ Mars'}(Artist \bowtie ConcertArtist \bowtie Concert))$

$\pi_{aid,aname}(aid,aname\ G_{count(cid)>=2}(T1 \bowtie T2))$

VI. Output the aid and name of any artist who has never given a concert in New York City.

Ans: $T1 \leftarrow \pi_{aid}(\sigma_{vcity='New\ York'}(Artist \bowtie ConcertArtist \bowtie Concert \bowtie Venue))$

$T2 \leftarrow Artist.aid - T1$

$\pi_{aid,aname}(Artist \bowtie T2)$

VII. Output the cid, data, and lowest cost for any Jazz concert in Chicago in October 2017.

Ans: $\pi_{cid,cdate,ticketcost}(\sigma_{genre='Jazz' \wedge vcity='Chicago' \wedge year(cdate)='2017' \wedge month(cdate)='10' \wedge (cid,ticketcost) = (cid\ G_{min(ticketcost)}(\sigma_{genre='Jazz' \wedge vcity='Chicago' \wedge year(cdate)='2017' \wedge month(cdate) = '10'}(TicketType \bowtie Concert \bowtie ConcertArtist \bowtie ArtistGenre \bowtie Venue)))(TicketType \bowtie Concert \bowtie ConcertArtist \bowtie ArtistGenre \bowtie Venue))$

(e) Write either DRC or TRC queries for the above queries. Or explain the reason why you think a particular query cannot be done in DRC or TRC.

TRC queries are as follows:

I. Output the names of all artists that have the genre “Jazz”.

Ans: $\{t \mid \exists r \in Artist(t[aid]=r[aid] \wedge \exists s \in ArtistGenre(s[genre] = 'Jazz' \wedge s[aid]=r[aid]))\}$

II. Output all the cities in which artist “Bruno Mars” played in 2016.

Ans: $\{t \mid \exists r \in Venue(t[vcity]=r[vcity] \wedge \exists s \in Concert(year(s[cdate])='2016' \wedge s[vid]=r[vid] \wedge \exists u \in ConcertArtist(u[cid]=s[cid] \wedge \exists v \in Artist(v[aname]='Bruno\ Mars' \wedge v[aid]=u[aid])))))\}$

III. For each venue, output the vid and name and the number of concerts it hosted in 2016.

Ans: This query requires group by clause and aggregate function which cannot be represented using Tuple Relational Calculus.

IV. Output the cid, ticketclass, cost, and date for the most expensive tickets ever for a Jazz concert. (A Jazz concert is one where at least one artist has Jazz as a genre.)

Ans: {t|∃r ∈ Concert ∧ ∃s ∈ TicketType(t[cid]=r[cid]∧t[ticketclass]=s[ticketclass]∧t[ticketcost]=s[ticketcost] ∧t[cdate]=r[cdate]∧s[cid]=r[cid]∧ ∃u ∈ ConcertArtist(s[cid]=u[cid] ∧ ∃v ∈ ArtistGenre(v[aid]=u[aid]∧v[genre]='Jazz')))∧∀z ∈ {m|∃n ∈ Concert ∧ ∃o ∈ TicketType(m[cid]=n[cid]∧ m[ticketclass] =o[ticketclass]∧m[ticketcost]=o[ticketcost]∧m[cdate]=n[cdate]∧n[cid]=o[cid]∧∃p ∈ ConcertArtist (o[cid] = p[cid]∧ ∃q ∈ ArtistGenre(q[aid]=p[aid]∧q[genre]='Jazz')) ⇒ z[ticketcost] < = t[ticketcost])})}

V.Output the aids and names of all artists who have appeared together with "Bruno Mars" in at least two concerts during 2016.

Ans: {t|∃r ∈ Artist (t[aid]=r[aid]∧t[aname]=r[aname]∧ ∃s ∈ ConcertArtist(s[aid]=r[aid]∧ ∃u ∈ Concert(u[cid]=s[cid]∧u[year(cdate)]='2016')∧∃v ∈ Artist(v[aid]=s[aid] ∧ v[aname]='Bruno Mars')∧ ∃n ∈ ConcertArtist(n[aid]=v[aid]∧ ∃o ∈ Concert(o[cid]=n[cid]∧year(o[cdate])='2016')∧∃q ∈ Artist(q[aid]=n[aid]∧q[aname]='Bruno Mars')∧s[aid]=n[aid]))∧r[aname] ≠ 'Bruno Mars')}

VI. Output the aid and name of any artist who has never given a concert in New York City.

Ans: {t|∃r ∈ Artist(t[aid]=r[aid]∧t[aname]=r[aname]∧¬ ∃s ∈ Artist(s[aid]=r[aid]∧ ∃u ∈ ConcertArtist (u[aid]=s[aid]∧ ∃v ∈ Concert(v[cid]=u[cid]∧ ∃w ∈ Venue(w[vid]=v[vid]∧w[vcity]='New York'))))))}

VII. Output the cid, data, and lowest cost for any Jazz concert in Chicago in October 2017. (Lowest cost means the cost of the cheapest class of tickets over all Jazz concerts.)

Ans: This query requires group by clause and aggregate function which cannot be represented using Tuple Relational Calculus.

Problem 2: In this problem, you need to design a relational schema for a babysitting agency website. This website is used to connect parents and babysitter, and to help parents book babysitters on-demand.

For each babysitter, you need to store detailed information, including a unique id, her/his name, age, gender, address, phone number, languages spoken, and a short text intro. For each customer (usually a parent needing a babysitter), you need to store a unique id, her/his name, address, and phone number. You also need to store information about each child, such as the parent/guardian, name, age, gender, and languages spoken.

Each babysitter has a schedule on the website, so that it is possible to search which babysitters are available at a particular time. For simplicity, you may assume that the schedule consists of slots lasting one hour. Parents may also upload babysitting requests, specifying the time and date, and which children need to be supervised.

When a babysitter and a customer come to an agreement on a babysitting job, (which will usually happen after some direct negotiation on the phone or in person outside this system), a booking is created and stored that specifies the customer, the babysitter, the children that will be supervised, the fee the babysitter charges for this job, a status (booked, cancelled, or completed), and a rating from one to five stars that the customer can assign after the job is completed. Note that each booking is for one babysitting session, so if the customer and babysitter agree on a job that takes place every Wednesday

evening, this would result in one booking for each week. Also, the children that should be supervised need to be specified for each booking; in some cases, a customer with two kids may require babysitting only for one of the kids, or a neighbor's child might join so the children might not all have the same legal guardian, and some babysitters might charge more for jobs involving more than one child.

(a) Design a relational database schema for this application scenario that supports the above functionality. Specify all primary and foreign key constraints, and state any assumptions you are making. You can decide which exact attributes make sense for this schema.

Ans: The relational database schema for the above application is as follows:

Babysitter (bid, bname, bage, bgender, baddress, bphn, bdesc)
Customer (cuid, cuname, cuaddress, cuphn)
Child (cid, cuid, cname, cage, cgender)
Babysitter_Schedule (bsid, bid, bsdate, bstime)
Babysitting_req (baid, cuid, bid, badate, batime, cid)
Booking (boid, baid, bofees, bostatus, botype, rating)
Languages_Babysitter (bid, blang)
Languages_Child (cid, clang)

Table	Primary Key
Babysitter	bid
Customer	cuid
Child	cid
Babysitter_Schedule	bsid
Babysitting_req	baid
Booking	boid
Languages_Babysitter	bid, blang
Languages_Child	cid, clang

Table	Foreign Key
Child	cuid
Babysitter_Schedule	bid
Babysitting_req	cuid, bid, cid
Booking	baid
Languages_Babysitter	bid
Languages_Child	cid

Assumptions made:

1. The Babysitter_Schedule table will store records for each available slot (i.e of one hour) of a babysitter in a day where bstime will store the start time of the slot.
2. The batime attribute of Babysitting_req table stores the start time of the babysitting session slot requested for booking.
3. The rating for all the bookings other than the ones with bostatus = 'completed' in the Booking table will be null
4. The booking table will store records for babysitting session of one child for one day only and we should make multiple bookings if we want to continue the session every week or if we want to schedule babysitting sessions for more than one child a day.
5. Assuming Babysitter and child speak multiple languages, I have created separate language table for both to reduce complexity of the system while storing data.

(b) Write SQL statements for the following queries. If your schema does not support these, you need to modify it appropriately. (For this first homework, you may use informal expressions such as Timestamp(ts) ='2017-09-08T11:00:00', where ts is a timestamp, to check if the datetime is 2017-09-08 11:00:00, or use year(ts) or month(ts) to get the year or month of a timestamp, etc.)

i. Output the names of all babysitters who are available to work from 2017-09-20T15:00:00 to 2017-09-20T18:00:00. (They should be available all three hours, not just part of the time.)

Ans:
```
Select bname
    from Babysitter
    where bid in (Select bid
                  from Babysitter natural join Babysitter_Schedule
                  where bsdate = ' 2017-09-20' and bstime between '15:00:00' and '17:00:00'
                   group by bid
                   having count(bid)>=3)
```

ii. For each babysitter, output their ids, and the number of distinct children they have babysat in 2016.

Ans:
```
Select bid, count (distinct cid) as number of distinct children
    from Babysitting_req natural join Booking
    where bostatus = 'completed' and year(badate) = '2016'
    group by bid
```

iii. Output the babysitter(s) who earned the most money overall during 2016.

Ans:
```
with overall_fees (bid, value) as
    (Select bid, sum(bofees)
     from Babysitting_req natural join Booking
     where year(badate) = '2016'
```

```
group by bid)
Select bid, bname
from Babysitter natural join overall_fees
where value = (Select max(value)
                from overall_fees)
```

iv. Output the names of customers who have booked at least 5 completed jobs, but who have never given a rating of 4 stars or higher.

Ans:

```
Select cuname
from Customer
where cuid in (Select cuid
                from Customer natural join Babysitting_req natural join Booking
                where bostatus =’completed’
                group by cuid
                having count(cuid)>=5 and max(rating) = 3)
```