**CS3431-A22 Wong**

**Assignment 3: Relational Algebra and More SQL**

**Solutions**

This assignment is to be typed. The following relational algebra symbols are provided for your use in the assignment:

σ, π, γ, **δ,** ⋈, 🡨, ∩, **∪**

**Part 1: Art Association (85 points, 17 points for each subpart)**

The relations given below are from Assignment 2. The characters before the table name are the table abbreviations to be used in your relational algebra answers below. Do not use the table abbreviations in your SQL code unless you create them in your code.

AR Artwork (year PK FK, artworkID PK, artistID FK, title, materialsID FK, price, chosen, purchased,

purchaseDate, gallery FK)

N NewArt (year PK FK, artworkID PK, artistID FK, title, materialsID FK, price, chosen, purchased,

purchaseDate, gallery FK)

A Artist (artistID PK, firstName, lastName, city, state, memberLevel FK)

MB Membership (memberLevel PK, yearlyFee, discount)

M Materials (materialsID, category, medium)

G Gallery (gallery PK, floor, building FK)

B Building (buildingName PK, street, city, state, zipcode)

T TicketPrice (year PK, price)

You may assume that there are no duplications in the combination of year and artworkIDs between Artwork and NewArt.

1. (17 points) Based on the primary keys and foreign keys in the relational schema above, specify below the foreign key relationships that exist.
   1. **Using the table abbreviations**, write the constraints using the following format:   
      Table1.ID1, Table1.ID2 References Table2.ID1, Table2.ID2

**AR.year References T.year**

**N.year References T.year**

**AR.artistID References A.artistID**

**N.artistID References A.artistID**

**AR.materialsID References M.materialsID**

**N.materialsID References M.materialsID**

**AR.gallery References G.gallery**

**N.gallery References G.gallery**

**A.memberLevel References MB.memberLevel**

**G.building References B.buildingName**

* 1. Assume the tables above have already been created but without any foreign key references. Create just the SQL constraints for the foreign keys using the ALTER TABLE command and the full table names instead of the abbreviations. Note that if an Artist leaves the association, all of the records in tables that reference the Artist table will have its value for artistID set to null.

alter table Artwork add constraint Artwork\_year\_FK foreign key (year) References TicketPrice (year);

alter table NewArt add constraint NewArt\_year\_FK foreign key (year) References TicketPrice (year);

alter table Artwork add constraint Artwork\_artistID\_FK foreign key (artistID) References Artist (artistID);

alter table NewArt add constraint NewArt\_artistID\_FK foreign key (artistID) References Artist (artistID);

alter table Artwork add constraint Artwork\_materialsID\_FK foreign key (materialsID) References Materials (materialsID);

alter table NewArt add constraint NewArt\_materialsID\_FK foreign key (materialsID) References Materials (materialsID);

alter table Artwork add constraint Artwork\_gallery\_FK foreign key (gallery) References Gallery (gallery);

alter table NewArt add constraint NewArt\_gallery\_FK foreign key (gallery) References Gallery (gallery);

alter table Artist add constraint Artist\_memberLevel\_FK foreign key (memberLevel) References Membership (memberLevel);

alter table Gallery add constraint Gallery\_building\_FK foreign key (building) References Buiding (buildingName);

1. (17 points) Display, just for artists who have a Student level membership, the artist’s full name in a column FullName (first name, a space, and then the last name), and the number of artworks that have titles which include the word ‘Beach’ in it. Use natural joins. Then make your relational algebra more efficient by filtering the records before joining the tables together. Remember to use the table abbreviations provided at the beginning of this assignment.
   1. Write the relational algebra

**γfirstName || ' ' || lastName as FullName, count(\*) (σtitle like '%Beach%' (AR) ⋈ σ memberLevel = 'Student' (A)**

A solution which includes cases of 0 counts is presented below:

**γfirstName || ' ' || lastName as FullName, count(\*) as CNT (σtitle like '%Beach%' (AR) ⋈ σ memberLevel = 'Student' (A)**

**∪**

π **firstName || ' ' || lastName as FullName, 0 as CNT ((δ**π **artistID AR - δ**π **artistID (σtitle like '%Beach%' AR)) ⋈ σ memberLevel = 'Student' (A))**

* 1. Translate the relational algebra in 2.a into SQL code. Note you will need to use subqueries in the natural joins in order to filter the records before the joins.

**select firstName || ' ' || lastName as ArtistName, count(\*)**

**from (select \* from Artwork where title like '%Beach%')**

**natural join (select \* from Artist where memberLevel = 'Student')**

**group by firstName || ' ' || lastName;**

The SQL code which includes cases of 0 counts is presented below. Note that select distinct can also be accomplished by using a group by command.

**select firstName || ' ' || lastName as ArtistName, count(\*) as CNT**

**from (select \* from Artwork where title like '%Beach%')**

**natural join (select \* from Artist where memberLevel = 'Student')**

**group by firstName || ' ' || lastName**

**UNION**

**select firstName || ' ' || lastName as ArtistName, 0 as CNT**

**from (**

**(select distinct artistID from Artwork**

**MINUS**

**select distinct artistID from Artwork**

**where title like '%Beach%')**

**natural join (select \* from Artist where memberLevel = 'Student'));**

1. (17 points) Write the relational algebra to display for each city, the number of artworks in each gallery located in that city. Only include artworks from the years 2009, 2013, and 2019. Use theta joins and make your relational algebra more efficient by filtering the records before joining the tables together.

**R1 🡨 (σ year in (2009, 2013, 2019) (AR)) ⋈AR.gallery = G.gallery (G)**

**Result 🡨 γcity, G.gallery, count(\*) (R1 ⋈R1.building = B.buildingName (B))**

1. (17 points) Write the relational algebra to display the following information for the artist who has donated the greatest total value of artworks: the first name, last name, and total value of all artwork donated with a heading of MaxDonatedValue. You will want to use the assignment operator ( 🡨 ) to make the relational algebra easier to do. Use theta joins.

**R1 🡨** πsum **γ artistID, sum(price) as SUM (AR)**

**R2 🡨 γ max (SUM) as MaxSum (R1)**

**Result 🡨 σ sum(price) = R2 γfirstName, lastName, sum(price) as MaxDonatedValue** **(AR ⋈AR.artistID = A.artistID A)**

1. (17 points) Write the relational algebra to display for each membership level, the number of members from the state of Rhode Island and the total of yearly fees. Only display the results for those membership levels where the total of yearly fees is greater than $5000.
   1. Write the relational algebra

**σ sum(yearlyFee) > 5000 γ M.memberLevel, count(artistID) as NumberOfMembers, sum(yearlyFees) as TotalFees** **(M ⋈M.memberLevel = A.memberLevel (σ state=’RI’ A))**

* 1. Translate the relational algebra in Part 5.a into SQL code  
       
     select M.memberLevel, count(artistID) as NumberOfMembers, sum(yearlyFee) as TotalFees

from Membership M

join (select \* from Artist where state = 'RI') A on M.memberLevel = A.memberLevel

group by M.memberLevel

having sum(yearlyFee) > 5000;

**Part 2: Relational Algebra (15 points, 5 points for each subpart)**

Provide the resulting tables with attribute headings for the following problems.

1. N ⋈B=T ( π B as T M – π A as T N)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | | | |  | **N** | | |
| **A** | **B** | **Y** | **Z** |  | **A** | **B** | **Q** |
| 3 | 4 | Tom | Jane |  | 1 | 8 | David |
| 6 | 2 | Mary | Susan |  | 7 | 10 | Jane |
| 10 | 8 | David | Paul |  | 4 | 7 | Paul |
| 4 | 3 | Mark | Helen |  | 4 | 10 | Tom |
| 1 | 1 | Lisa | Brian |  | 2 | 3 | Susan |

Answer:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | Q | T |
| 1 | 8 | David | 8 |
| 2 | 3 | Susan | 3 |

1. πB,Y,Z(M) ⋈ πB,Q, A+B as Z (N)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | | | |  | **N** | | |
| **A** | **B** | **Y** | **Z** |  | **A** | **B** | **Q** |
| 3 | 4 | Tom | 2 |  | 1 | 1 | David |
| 6 | 2 | Mary | 9 |  | 7 | 2 | Jane |
| 4 | 3 | Mark | 5 |  | 4 | 7 | Paul |
| 1 | 1 | Lisa | 4 |  | 4 | 10 | Tom |
|  |  |  |  |  | 2 | 3 | Susan |

Answer:

|  |  |  |  |
| --- | --- | --- | --- |
| B | Z | Y | Q |
| 2 | 9 | Mary | Jane |
| 3 | 5 | Mark | Susan |

1. σA=1 or B=3 (πA,B,Z as Q (M) **∪** N)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **M** | | | |  | **N** | | |
| **A** | **B** | **Y** | **Z** |  | **A** | **B** | **Q** |
| 3 | 4 | Tom | Jane |  | 1 | 8 | David |
| 6 | 2 | Mary | Susan |  | 7 | 10 | Jane |
| 10 | 8 | David | Paul |  | 4 | 7 | Paul |
| 4 | 3 | Mark | Helen |  | 4 | 10 | Tom |
| 1 | 1 | Lisa | Brian |  | 2 | 3 | Susan |

Answer:

In order to take the union of M and N, they must be union-compatible. So M.Z and N.Q must be the same datatype and domain. This also applies to matching columns A, and matching columns B.

|  |  |  |
| --- | --- | --- |
| A | B | Q |
| 1 | 1 | Brian |
| 1 | 8 | David |
| 4 | 3 | Helen |
| 2 | 3 | Susan |

If your answer is the empty set (table), then you must state why the tables are not union compatible.