308-612A Database Programming Principles

Supplementary Questions and Exercises

Relational Algebra

1. In an electronic logic circuit, primitive gates are represented by their truth tables in the form of the following relations.

AND(*X*	*Y*	*Z*)	*OR*(*X*	*Y*	*Z*)	*NOT*(*W*	*Z*)
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

   Write two relational algebra statements, each of which gives the relation *NOR* (on three attributes), which is the truth table of the combined operation, *OR* followed by *NOT*. One staement should use the [<attr.list> <join> <attr.list>] syntax for joins and the other should use only <join>.

2. (a) Design a relation which can be used to represent text as a heirarchy (e.g., as a sequence of chapters, which are sequences of sections, which are sequences of paragraphs, which are sequences of sentences, etc.). Use as few attributes as possible, but be prepared to answer the following question.

   (b) Using the above relation, write recursive code to identify all the paragraphs that contain the word "Aldat".

3. The relations `IMPORTS(COUNTRY, YEAR, PRODUCT)` and `EXPORTS(COUNTRY, YEAR, PRODUCT)` give, respectively, the products imported and exported each year by each country.

   (a) A country is *badly managed* if, in any one year, it imports most of the products which it exports. Write Aldat statements to find all badly managed countries.

   (b) An exporting country *covers* an importing country if, in any one year, its exported products include all the imports of the latter. Write Aldat statements to find all pairs of countries such that the first covers the second.

   (c) Write Aldat statements to find the transitive closure of the relation you found in (b).

   (d) What is the relationship between the answers to (b) and (c)?

4. Can the *one* relation, *R*, below be natural-join-decomposed without loss using projection? Give *all* projections that can be joined to form the original relation and write the corresponding relational algebra expressions.

R(*A*	*B*	*C*)	*cont*(*A*	*B*	*C*)
Ann	Sue	21	Ann	Joe	33
Tom	Sue	21	Tom	Joe	33
Ann	Joe	27	Ann	Sue	33
Tom	Joe	27	Tom	Sue	33

5. Can we have a lossless σ-*join decomposition* analogous to the lossless natural-join decomposition? Show an example or explain why we cannot.

6. Given the relations $R(A, B, C)$ and $S(C, D)$, use only μ-joins, T-selectors and the domain algebra if necessary to write statements equivalent to the following. Use Aldat syntax.

   (a) $U <\!- R$ **sep** S;

   (b) $V <\!- [A]$ **where** $\{(\# = 3)B\}$ $C = 2$ **in** R;

7. A relation *PIC*(*SR, JR, X, Y*) describes pictures (*SR*) composed of subpictures (*JR*) centred at coordinates (X, Y). Using recursion, write Aldat statements which give *PIC**, the closure of *PIC*. For instance, given *PIC*, below, *PIC** would consist of the tuples of *PIC* plus the tuples (P, P11, 2, 2) and (P, P21, -2, 2).

PIC(*SR*	*JR*	*X*	*Y*)
P	P1	2	2
P	P2	-3	1
P1	P11	0	0
P2	P21	1	1

8. Relations FACTS(CONS) and RULES(RULE#, ANTE, CONS) represent, respectively the *facts* that Fido has fur, pointed teeth and was born, and the *rules* that mammals are born, carnivores have pointed teeth, and dogs are furry carnivorous mammals. Show the data for the relations and show the Aldat code that would deduce all consequences of given facts, such as that Fido is a dog.

9. Name three σ-joins on $R(A, B)$ and $S(B, C)$ that give the same result assuming the functional dependence $A \rightarrow B$.

10. Given the relation *Parts*(*partNo, colour*), a) find all pairs of parts such that the first part comes in all colours the second part comes in. b) What is significant about the transitive closure of the relation describing these pairs?

11. (a) Show all ways in which the following relation can be projected into pairs of binary relations such that the pairs can be **ijoin**ed to give back the original relation.

    (b) Show all ways in which the above can be done with one unary and one binary relation n each case.

Ann	Joe	Que
Ann	Mac	Que
Sue	Joe	Que
Ann	Joe	Ont
Ann	Mac	Ont
Sue	Joe	Ont

12. In the relation *R*(*Women, Men, Province*), write relational algebra statements to find all women who relate to all men (in R) and men who relate to all women. Using as data the relation of the previous question, show the results of executing these statements.

13. (a) What is the computational complexity of the natural join? What basic operations must be counted in obtaining this complexity?

    (b) Outline an algorithm to implement the family of μ-joins and discuss the algorithm in the light of your answer about complexity, above.

14. The relations BC and GC describe the kinds of restaurants people like to go to. For the data shown, how many tuples will be in the natural join, BGC? Give *all* the ways in which BGC can be decomposed into separate relations which **ijoin** can put together to give back BGC.

BC		*GC*	
(*Boy*	*Cuisine*)	(*Girl*	*Cuisine*)
`Joe`	`French`	`Ann`	`French`
`Joe`	`Greek`	`Ann`	`Greek`
`Mac`	`French`	`Sue`	`French`
`Mac`	`Greek`		

15. If the functional dependence, $B \rightarrow A$, holds in the relation, $R(A, B, C)$, (*or* if the functional dependence, $B \rightarrow C$, holds) then

$$R = ([A, B] \textbf{ in } R) \textbf{ ijoin } [B, C] \textbf{ in } R \tag{1}$$

   (a) Prove that this is true. (A picture could suffice.)

   (b) Show an *example* of the smallest relation for which equation (1), above, is true, but neither of the functional dependences holds. (That is, the functional dependences are sufficient but not necessary.)

16. A catering organization, Caterwell, has employees who wait table and wash dishes, but who are not allowed, by their union, to do both at any one event. For the holiday season, there are a number of events, and Caterwell has used two relations for the schedule: *Tables*(*Event, Waiter*); and *Dishes*(*Event, Washer*). There is a crew of waiters and a crew of washers for each event.

   Write Aldat code in two ways to find all events that violate the union rule:

   (a) joining on the common attributes;

   (b) joining on the non-common attributes.

   (c) Supposing that there are five events, and in each event the table crew is six people and the dish crew is three, say which of the above two methods is cheaper and why.

17. (a) Give *all* possible ways the relation shown can be decomposed into two binary relations that can be natural joined to give back the original relation.

   (b) What other decomposition is there that can be natural joined to give back the original relation?

R(*Item*	*Colour*	*Floor*)
gizmo	red	1
gizmo	blue	1
gizmo	red	2
gizmo	blue	2
widgit	red	1
widgit	blue	1

18. Using R from the previous question, and I as shown, give the value of each of the seven σ-joins.

I(*Item*)
gizmo
widgit

I **icomp** R
I **sep** R
I **sup** R
I **gtjoin** R
I **sub** R
I **ltjoin** R
I **eqjoin** R

19. The relationship, $a = b + c$, which says that a is always the sum of b and c, can be thought of as the relation, $p(a, b, c)$. This in turn can be thought of as an infinite set of tuples (suppose a, b and c are integers) containing all triples of integers satisfying the sum.

Similarly, the relation $m(x, y, z)$ represents the relationship $x = y * z$.

(a) What relationship is given by $p[a$ **ijoin** $x]m$?

(b) What relationship is given by the above with **ijoin** replaced by each of the other μ-joins?

(c) What relationship is given by $p[c$ **ijoin** $x]m$?

(d) What relationship is given by the above with **ijoin** replaced by each of the other μ-joins?

(e) What relationship is given by $p[a$ **icomp** $x]m$?

(f) What relationship is given by $p[a$ **sup** $x]m$?

20. A tree can be represented as a binary relation, *Tree2*(*Sr, Jr*), or as a quaternary relation, *Tree4*(*self, parent, sibling, child*), as shown by the following example.

Tree2(*Sr*	*Jr*)	*Tree4*(*self*	*parent*	*sibling*	*child*)
CARS	FORD	0	5	-1	2
CARS	GM	1	-1	-1	4
FORD	FAIRLANE	2	0	-1	-1
FORD	MUSTANG	3	4	6	-1
GM	BUICK	4	1	-1	3
BUICK	CENTURY	5	1	-1	0
		6	4	-1	-1

The connection between the strings and the integers in these two representations is that the integers are synonyms for the strings and that the numerical order of the integers gives the alphabetical order of the strings, and vice-versa: (`BUICK, 0`), (`CARS, 1`), (`CENTURY, 2`), (`FAIRLANE, 3`), (`FORD, 4`), (`GM 5`), (`MUSTANG, 6`).

The point of this question is to generate *Tree4* from *Tree2*. Note that it has four parts, equally marked. Steps b, c, d may be done in any order.

(a) Put the two attributes of *Tree2* together and generate an attribute which is the position of the string in sorted order.

(b) Generate attributes *self, parent, sibling, child* for the internal nodes of the tree.

(c) Generate attributes *self, parent, sibling, child* for the root node(s) of the tree.

(d) Generate attributes *self, parent, sibling, child* for the leaf nodes of the tree.

And, of course, put them all together to generate *Tree4*.

Hint: draw *Tree4* with corresponding strings replacing the integers, in order to understand exactly what its attributes mean, particularly *child* and *sibling*.

21. *FemDem* gives a projection for women's demographics for the year 2000. The *age*s are the bottom values of 5-year ranges, and the *pop* and *birth* numbers are in millions for the year 2000. ("*birth* numbers" 21. can be taken to mean the number of women in each age category having a baby that year.) Write code to find the percentage growth in the population (assuming there are an equal number of men) and the percentage of girls (ages 10–19) giving birth that year.

FemDem

(*age*	*pop*	*births*)	(*age*	*pop*	*births*)
95	1	0	45	160	0
90	2	0	40	190	1
85	5	0	35	210	4
80	10	0	30	230	14
75	20	0	25	240	21
70	40	0	20	250	22
65	60	0	15	255	22
60	80	0	10	255	5
55	100	0	5	250	0
50	130	0	0	240	0

22. In a relation, *Names*(*SIN, family, given*), a person (identified by Social Insurance Number) has one family name and possibly many given names. Write code to find all people who have all the given names of anyone else in their families. (Assume same family name means same family.)

23. Using relational and domain algebras, write a recursive view which computes factorial. Make sure it stops at, say, 4! You may need to create an initial value for your factorial relation, and use the **initial** syntax to start the recursion.

    factorial **initial** *factorial0* **is** ..

24. Given the relation, $R(A, B, C)$, write Aldat code to determine whether

    (a) the functional dependence $A \rightarrow B$ holds;

    (b) the join dependence holds that permits R to be reconstructed from its projections on $\{A, B\}$ and $\{B, C\}$.

    (c) Give example data for R in which both of the above hold but neither $B \rightarrow A$ nor $B \rightarrow C$.

25. A catering organization, Caterwell, has employees who wait table and wash dishes. Members of the crew who wait tables may also wash dishes, subject to the rule that at least one of their number is always on duty in the dining room (and so not washing dishes). For the holiday season, there are a number of events, and Caterwell has used two relations for the schedule: *Tables*(*Event, Waiter*); and *Dishes*(*Event, Washer*). There is a crew of waiters and a crew of washers for each event.

    Caterwell is using the new (Java) implementation of Aldat, which does not yet have σ-joins or QT-selectors, although it implements enough of Aldat to solve the problem: how do they determine which events violate the rule?

26. Use Aldat to add up the *prices* by *product* in a datacube and include the result as one tuple for
26. each product. The example shows a particular case, but your code should handle any number of tuples and any number of attributes.

DataCube

(*product*	*store*	*date*	*price*)
shampoo	downtown	981203	2.25
shampoo	downtown	981204	2.35
shampoo	shopctr	981203	2.20
shampoo	shopctr	981204	2.30
soap	downtown	981203	1.25
soap	downtown	981204	1.35
soap	shopctr	981204	1.30

Tuples to be included:

shampoo	DC	DC	9.10
soap	DC	DC	3.90

27. The *datacube* shown has the three attributes, *product, store* and *date* as key, and may be thought of as a three-dimensional box containing *price*s. Write Aldat code to complete the cube by summing the
27. *price*s over all *product*s; over all *store*s; over all *date*s; over all *product*s and *store*s; over all *product*s and *date*s; over all *store*s and *date*s; and over all *product*s, *store*s and *date*s (that is, $7 = 2^3 - 1$ kinds of sums).

DataCube

(*product*	*store*	*date*	*price*)
shampoo	downtown	981203	2.25
shampoo	downtown	981204	2.35
shampoo	shopctr	981203	2.20
shampoo	shopctr	981204	2.30
soap	downtown	981203	1.25
soap	downtown	981204	1.35
soap	shopctr	981204	1.30

Hint. Code the first sum, incorporating the answer in *DataCube* with null values for *product*. Then notice that this has built a face on the datacube, turning it into an extended datacube.

28. (a) Decompose the relation *dps*, shown below, into two ternary relations, such that the original is the natural join of the components. Show the matrix form of the original relation, and discuss.

(b) For the relation *dps*, give two ways of writing the query *find values of p associated with all values of d*. What is the result?

(c) For the decomposition you got in part a, give two ways of writing the query *find values of p associated with all values of d*. (The result must be the same as part b.) Discuss.

(d) For the relation *dps*, write code to give the sum of Q for each pair of values for p and s, and the sum of Q for each value of d.

(e) For the decomposition you got in part a, write code to give the *same* sum of Q for each value of d.

(f) Repeat the last two parts, but finding the averages instead of the sums. Discuss any simplifications you can make.

dps(*d*	*p*	*s*	*Q*)
t	1	a	3
t	1	b	3
y	1	a	2
y	2	a	2

29. (A *recommender system* promotes goods to a potential customer by finding items he has not tried and which have succeeded with other customers of like tastes. This question works through a particular recommender system for books—or, to keep the data short, authors of books. Read all parts of this question before starting.)

In the following, you should write general Aldat code. Given the data

read

reader	*author*
Joe	Steinbeck
Joe	Forrester
Joe	Niven
Sam	Steinbeck
Sam	Hemingway
Sam	Forrester
Sam	Ignatieff
Sue	Niven
Tom	Hemingway
Tom	Clarke
Tom	Niven
Tom	Forrester

a) find reader(s) who have the largest author overlap with Joe (i.e., who have read the most authors that Joe has read);
b) find author(s) most often read by these readers, but not read by Joe. (These would be the authors that the system would recommend to Joe.)
c) Now suppose a third attribute, *rating*, is added to *read*, giving a number from 0 to 1 to indicate how much the reader likes the author: modify (b) (and (a) if necessary) to recommend the most highly rated author read by the reader(s) in (a). (If more than one reader recommends an author, take the highest rating.)
d) For this three-attribute relation, find both: the average rating per reader; and authors all of whose ratings are above 0.9.

30. Show the smallest projections from which each of the following relations can be reconstructed by **ijoin**s. If a relation cannot be so reconstructed, say so. If it can be reconstructed in more than one way, show all ways.

R

A	*B*	*C*
Joe	Can	Pres
Joe	Eng	Pres
Sue	Can	Pres
Sue	Eng	Pres
Joe	Can	Fut
Joe	Eng	Fut

S

A	*B*	*C*
Joe	Can	Past
Ann	Can	Pres
Joe	Can	Pres
Sue	Can	Pres
Joe	Aus	Pres
Joe	Eng	Pres
Joe	Can	Fut

T

A	*B*	*C*
Joe	Can	Pres
Joe	Eng	Pres
Sue	Can	Pres
Sue	Eng	Pres
Joe	Can	Fut
Joe	Eng	Fut
Sue	Can	Fut
Sue	Eng	Fut

31. Write an expression to find all colours in a relation $PC(P, C)$ except those in which parts (P1 and P2) or P3 come.

32. 5. Supposing the relation *Fool*, shown, contains all possible people (*Who*) and all possible times (*When*), write expressions giving people who can be fooled all of the time, times when all the people can be fooled, and the predicate "you can fool all of the people some of the time, some of the people all of the time, but not all of the people all of the time".

Fool

Who	*When*
Ann	2004
Joe	2000
Joe	2004
Joe	2008
Sue	2000
Ted	2000
Ted	2004

The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.