CSc 460 — Database Design
Fall 2020 (McCann)

http://u.arizona.edu/~mccann/classes/460

Homework #2

(100 points)

Due Date: October 15th, 2020, at the beginning of class

Overview: Becoming proficient in formulating useful database queries takes practice. Knowing how to use pure Relational Algebra is useful background for learning SQL. To use SQL, you may not need to call the Relational Algebra operators directly, but you do need to specify the critical parts of them. It helps to know how those parts fit in to the query.

Software: Richard Leyton, then a student at Oxford Brookes University, wrote a simple DBMS called LEAP as a project. The current version of LEAP is 1.2.6 and runs reasonably well under the LINUX operating system. The syntax of its relational algebra commands differs a bit from what we use in class, but converting between the notations is not hard. There does exist a version of LEAP for Windows, but I have never used it and, given its age, cannot recommend it.

To install LEAP into your lab account, here's what you need to do:

1. From your home directory, type this: `/home/cs460/fall20/leap/scripts/users/leapinstall`
2. When asked to supply the LEAP source directory, respond with this: `/home/cs460/fall20/leap`
3. When asked to supply the target directory, just press Enter. It will default to a subdirectory named `leap` in your account.

To run leap, here's what you need to do:

1. Change directory to your local LEAP bin subdirectory: `cd leap/bin`
2. Run leap: `./leap`
3. Give leap the command `use aquarium` to select our sample database.

To execute the sample query I've provided (in `leap/database/aquarium/source/sample.src` in your account), run leap (see the three steps above) and give this LEAP command: `@ sample` Please note that for this command to work, your source files must be in that directory (`leap/database/aquarium/source`).

Here's a brief list of useful LEAP commands and their syntax:

Operation	**General Format**	**Example of Use**
Use	use *database*	use rental
Select	select (*relation*) (*condition*)	r1=select (borrow) (amount='1000')
Project	project (*relation*) (*attr. list*)	r2=project (spj) (sno,qty)
Join	join (*rel1*) (*rel2*) (*condition*)	j=join (spj) (p) (spj.pno=p.pno)
Union	(*relation*) union (*relation*)	u=(employee) union (manager)
Intersection	(*relation*) intersect (*relation*)	int=(employee) intersect (manager)
Difference	(*relation*) difference (*relation*)	m=(employee) difference (manager)
Cartesian Product	(*relation*) product (*relation*)	prod=(s) product (spj)
Display a Relation	display *relation*	display prod
Copy a Relation	duplicate (*relation*)	copyofs = duplicate (s)
Execute a Source File	@ *filename*	@ sample
Quit LEAP	quit	quit

(Unfortunately (for you!), LEAP does not have the division operator.) Other LEAP commands can be learned by reading the on–line help (type: `help` from within LEAP to get started) or the documentation files in the doc and help subdirectories. Be aware that every attribute is of type STRING or INTEGER, and all constants have to be specified in single quotes (yes, including integers!).

(Continued . . .)

Assignment: The database already contains some relations for an aquarium database in LEAP format. Here are their schemas.

```
Species (sno, sname, sfood)
Tank (tno, tname, tcolor, tvolume)
Fish (fno, fname, fcolor, fweight, t#, s#)
Event (eno, f#, edate, enote)
```

I've underlined the primary key fields. Foreign keys should be easy to identify, as they have names similar to the corresponding primary keys.

Your task is to create relational algebra queries that correctly answer the following questions. We expect that you will split your queries into sequences of multiple steps (for clarity, when appropriate) and will add explanatory comments to help the reader understand how your query works. If you believe that a query is impossible to answer using LEAP, explain why.

1. What are the names of all of the red fish?
2. What are the colors of all tanks named "lagoon"?
3. What is the Cartesian Product of the sname field from Species with the tname field from Tank?
4. (You must **not** use JOIN for this query) What are the colors of the sharks?
5. (You must use JOIN for this query) What are the colors of the sharks?
6. What are the names of the fish that are sharks and live in cesspools?
7. The database contains names of species, tanks, & fish. Create a relation containing all of these names in one column (field).
8. What are the names of species found in puddles?
9. What are the names of species that are found in the same tank with a shark?
10. What are the names of the fish that have been born and are swimming?
11. What are the names of the fish that have been born but are NOT swimming?
12. What are the names of the species that eat herring that have a representative in all green tanks?

Hand In: Create and submit, using turnin on lectura, the following items. The submission folder is `cs460h2`.

1. A text file named `netid.txt` (replace 'netid' with your NetID) containing (a) your name, (b) your Relational Algebra queries (from your `.src` files), and (c) the answers LEAP produces when it runs them. Please show your queries in ascending numerical order (1 to 12), and produce the answers in the same order.

   Creating this text file is easy if you use the script utility program; see below. Start script, use cat to display your .src files, run LEAP to produce your query results, exit script, and rename the resulting file.

2. Your LEAP query .src files (as a `tar` file).

Want to Learn More About LEAP? Visit `http://leap.sourceforge.net/` (on–line help files!)

Other Requirements and Hints:

- You can easily capture LEAP's output to a file by running LEAP within the script command. First type `script`, then run LEAP, then type `exit`. Everything you saw on the screen is saved in a file named `typescript`. Helpful hint: **Never** run a text editor from within `script`!
- In LEAP, you can create a `.src` file (named `query1.src`, for example) in the `leap/database/aquarium/source` directory in your account, and type into it the sequence of operations needed to answer question #1, above. To execute a file of LEAP commands, type @ followed by the name of the .src file you want to execute. (For example: `@ query1`) Do this while running LEAP, of course. This is a convenient mechanism for storing your queries and for easily creating your final output for submission on the due date.

(Continued . . .)

- I have attempted to write solutions to all of the assigned queries, and believe them to be possible to answer using LEAP. Feel free to help each other out with workarounds, etc., to LEAP's quirks, but write your own queries. (If you pattern your script file(s) after the `sample.src` file I've provided, things should work fine.)
- LEAP may have problems dealing with temporary relations of high degree; if you have problems, remove extra attributes before performing joins.
- When LEAP has a core dump, it doesn't clean up after itself, and as a result it can fail to restart. A simple solution: Save copies of your `.src` files, and reinstall LEAP.
- And finally: Please remember that a correct answer is a query that produces the correct result *in a logically correct way*! Write queries that will work even if the relations' content changes.