CSc 460 — Database Design
Spring 2023 (McCann)

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

# Homework #2

(100 points)

*Due Date: March 2nd, 2023, 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 so 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/spring23/leap/scripts/users/leapinstall`
2. When asked to supply the LEAP source directory, respond with this: `/home/cs460/spring23/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. To select our database, give leap this command: `use ebook`

To execute the sample query I've provided (in `leap/database/ebook/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/ebook/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!).

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

```
book (isbn,title,edition,category,price,copyright,pages,pcode)
encoding (isbne,format,drm)
writer (wid,surname,givenname,wcity,wstate,wzipcode,phone,email)
publisher (pid,name,address,pcity,pstate,pzipcode,url)
authorship (isbna,aid,percentage)
review (rid,isbnr,stars)
```

I've underlined the primary key fields. Foreign keys should be easy to identify, as they have names similar to the corresponding primary keys. Looking at the data helps, too!

Your task is to create relational algebra queries for LEAP that correctly answer the following questions:

1. What are the names of the publishers?
2. What are the full names (given and surnames) of the writers who live in California (CA)?
3. What is the Cartesian Product of the writers' cities and the publishers' cities?
4. (You must **not** use JOIN for this query) What are the titles of the ebooks encoded in azw3?
5. (You must use JOIN for this query) What are the titles of the ebooks encoded in azw3?
6. What are the surnames of the writers who have at least one of their books available as a PDF?
7. For each book available without DRM, show its title, author's email, and publisher URL.
8. What are the full names of the writers who have written one or more ebooks as a coauthor?
9. Display the titles of the books that have no encodings.
10. What are the titles of the ebooks that have a copyright more recent than 2015 or that have at least 500 pages (or both)?
11. What are the names of the publishers of the books written by book authors who are also reviewers?
12. What are the IDs of the reviewers who have reviewed all of Chousky's books? (Yes, this is the $\div$ query.)

**Hand In:** Submit your LEAP queries (as a `tar` file) using turnin. The submission folder is `cs460h2`.

**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: Don't run a text editor from within `script`!
- In LEAP, you can create a `.src` file (named `query1.src`, for example) in the `leap/database/ebook/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.
- 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, the odds of success improve.)
- LEAP may have problems dealing with temporary relations of high degree; if you have problems, remove extraneous attributes before performing joins.
- When LEAP crashes, it can't clean up its temporary files, 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' contents change.