raeval

Relational Algebra Evaluator

Version 0.2 (Beta), 20th April 2011

Basic Instructions

Contents

1. About raeval

Raeval is an interactive interpreter for relational algebra.

This means that you can load relations and then evaluate relational algebra expressions against them and see the results on-screen.

The aim is to penetrate some of the mystery of relational algebra by allowing experimentation.

The screenshot below shows the evaluation of the following commands:

1. relna **:= load** “/home/nick/Desktop/testreln.csv”

2. **project** relna **over** b

3. **project** (**select** relna **where** b < c) **over** b, c, d

Explanation:

Command 1 loads a relation from a csv file into a variable called relna.

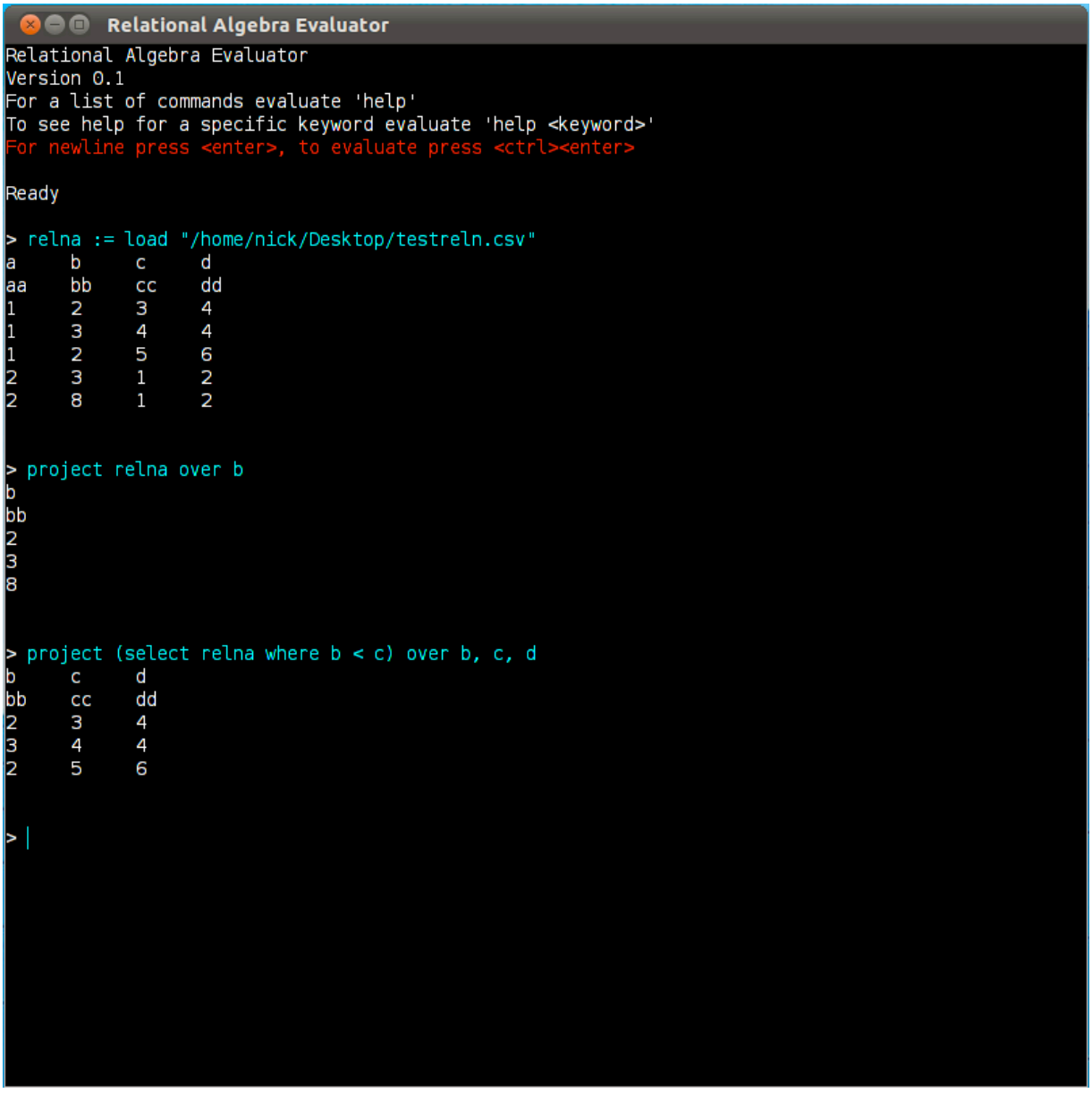
The result of the operation (i.e. the contents of the loaded relation) are displayed.

In the display, the first line shows the attribute names and the second line shows the domain names for those attributes. Subsequent lines show each tuple.

Command 2 projects the relation in variable relna over the attribute b.

The result of the projection is displayed.

Command 3 carries out a select operation and then projects the result which is then displayed.



Supported relational algebra operations are:

select	**union**
project	**intersection**
join	**difference**
rename	**alias**
divide	
times	

Conditions may include ( ) + - * / and or not < > = <= >= <> as well as attribute names taken from the relation.

Non-relational operators are:

load	**:=**	(assignment)
show		

In addition, the interpreter supports:

help

quit

2. Installation

Download raeval.jar from the project home.

Double click on the downloaded raeval.jar file to start the interpreter.

Alternatively, from a command line interface, type “java -jar raeval.jar”.

3. Using the interpreter

Relational algebra expressions may be typed over several lines. To move to a new line, press the <enter> key on your keyboard.

To execute a completed expression, press <ctrl> and <enter> together.

To copy the last command, for example to correct an error, press <ctrl> and <up arrow> together. Infinite command history is can be accessed using <ctrl><up arrow> and <ctrl><down arrow>.

Mac users can also use the Command key in place of <ctrl>.

“help” and “quit”

To quit, type **quit** and press <ctrl><enter>.

To get help, type **help** and press <ctrl><enter>.

Help is also available for each relational operator, for example **help select** <ctrl><enter>

Nesting expressions

It is possible to nest relations using brackets.

For example, the following are valid expressions (assuming that the relations exist):

- **project** (**select** person **where** age > 100) **over** name, age
- **project** ((**select** person **where** age > 100) **join** address) **over** name, age, town
- **project** (**select** person **where** (age > 100) **or** (name = “zebedee”)) over name, age
- oldpeople **:= select** person **where** age > 100

4. Creating and loading relations -- the 'load' operator

It is not possible to edit the contents of relations in raeval (except by using relational algebra). Instead raeval uses the **load** command to bring relations in from a CSV (comma separated variable) file.

CSV files can be created using a text editor or most spreadsheets.

Rules for files:

- Use a comma to separate values
- Keep all values for a tuple on the same line
- Use quotes to enclose strings (optional)
- The first line must contain the attribute names for the relation
- The second line must contain the domain names for the attributes
- Any subsequent lines contain the tuples
- Whitespace may be included to improve readibility - it will be trimmed

Some example files are included in the testdata.zip file in the downloads section of the project homepage (http://code.google.com/p/relational-algebra/).

An example is given below for somerelation.csv:

```
a,      b,      c,      d
aa,     bb,     cc,     dd
1,      2,      3,      4
4,      8,      1,      2
1,      2,      1,      4
8,      9,      1,      8
5,      4,      7,      9
```

This relation has four attributes: a, b, c and d.

These attributes are defined over domains aa, bb, cc and dd respectively.

The relation has five tuples.

When loaded, relations should be assigned to a relation variable (otherwise they will not be available for subsequent operations).

To load a relation from a CSV file, evaluate

<relation> **:= load** “<path for file>”

For example,

relnb **:= load** “C:\relnb.csv”

5. More about assignment (:=)

The assignment operator is not part of relational algebra. But it allows calculated relations to be stored for further computation. This improves the range of experimentation that can be achieved.

To use assignment evaluate

<relation> **:=** <relation>

For example,

tom **:= select** harry **where** secretagents > 5

dick **:=** tom

harry **:= project** harry **over** codewords, secretagents

The assignment operator performs a deep copy so that the new relation that is stored is a separate and independent copy of the donor relation.

If the result of an expression is not assigned it will be evaluated, displayed and then discarded.

6. Relational operators.

The following example expressions give an indication of how to use the remaining relational operators. The examples assume the existence of two relations named relna and relnb. They may have been loaded from disk or created using assignment of an earlier result.

Each relation contains the same attributes (a, b, c and d) and the same domains - they are therefore 'union compatible'.

Examples of expressions involving relational operators:

select relna **where** (a > 1) **and** (b > 5)

select relnb **where** a >= 4 **and** d <> "aardvark"

select relna **where** (a > 1) **or** ((b > 5 **and** c <> 9))

result := project relna **over** a

project relna **over** a, b, c

result := select (**project** relna **over** a, b) **where** a >= b

relna **join** relnb

relna **rename** (a **as** newa)

relna **rename** (a **as** newa, b **as** newb)

result := relna **union** relnb

select (relna **union** relnb) **where** c > 5 **or** a < 3

relna **intersection** relnb

result := project (**select** (relna **intersection** relnb) **where** d = "wolf") **over** b, d

relna **difference** relnb

result := relna **difference** (**select** relna **where** a > 5)

relna **times** relnb

result := relna **times** relnb

(**select** relna **where** d <> "wolf" **and** d <> "aardvark") **times** relnb

divide relna **by** relnb

(relna **times** relnb) **divide** relnb

7. The show command

The show command can be used to check the contents of a relation variable.

For example,

result **:= selec**t relna **where** c > 2

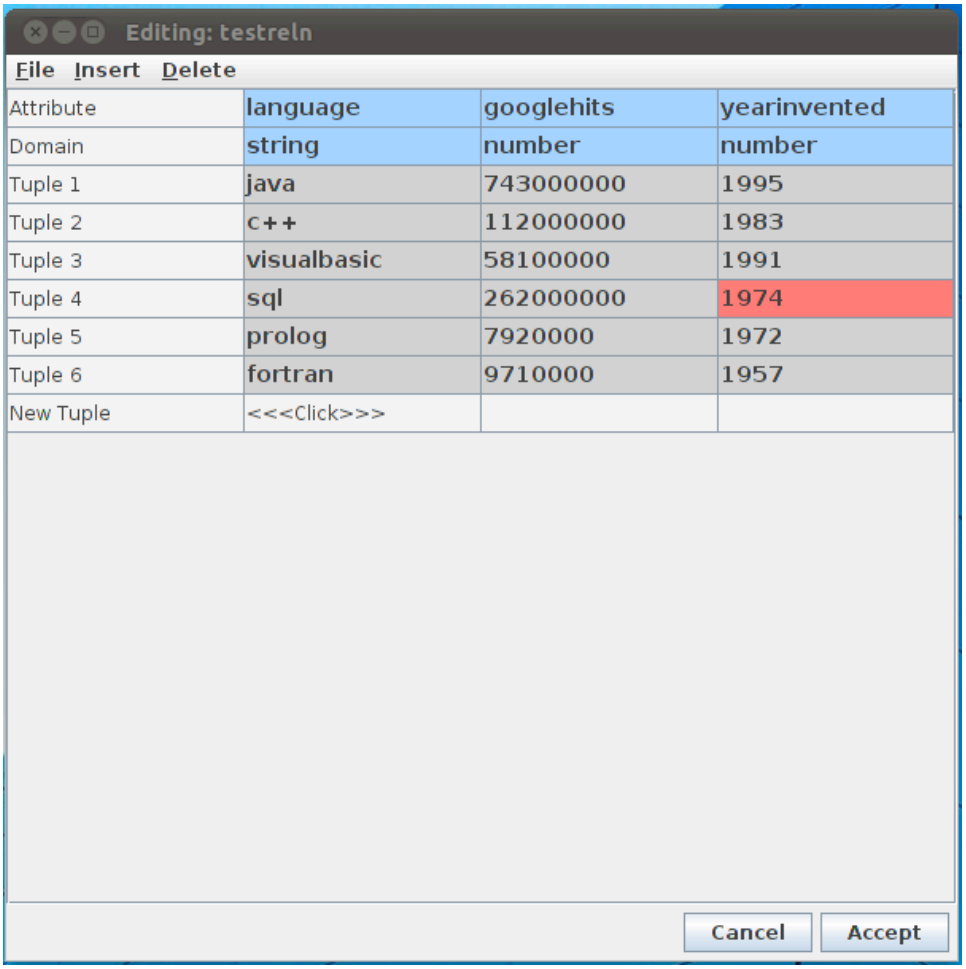
Then,

show result

8. Using the editor

The editor can be used to create and amend relations without the need to use CSV files. However, the editor does have the ability to load and save files to disk using CSV if you wish.

The editor window showing a relation being amended:



To begin an editor session evaluate:

edit <relation> (a relation name must be provided)

For example,

edit myrelation

To **create a new relation**, provide a relation name that does not already exist.

The edited (or created) relation will only be brought back to the when the Accept button is clicked. Clicking Cancel will discard all changes and return to the interpreter. This is the case even if the relation has been loaded or saved during the edit session.

The editor will not allow a relation to be accepted if it has blank values for any attribute name, domain name or tuple value. If this is an attempted then a warning dialog will be displayed and the editor session will continue. The editor will accept duplicate tuples, but the duplicate will be removed from the relation that is returned to the interpreter.

The following menu options are available within the editor:

File - Open File: opens a dialog box to identify a CSV file to load.
File - Save: save to CSV file where file name already known.
File - Save As: opens a dialog box to identify a CSV file to save (will be overwritten).
File - Close: cancels the edit and returns to the interpreter.

Insert - Insert Attribute Left - create a new column for an attribute
Insert - Insert Attribute Right - as above
Insert - Insert Tuple Above - create a new row for a tuple
Insert - Insert Tuple Below - as above

Insert commands operate relative to the current selected value (shown in red). Click a value to select.

A ‘New Tuple’ row is provided at the bottom of the relation. Clicking in the indicated value will auto create a new tuple at the bottom as a shortcut to insert below.

Delete - Delete Attribute - removes the attribute column which is currently highlighted.
Delete - Delete Tuple - removes the tuple that is currently highlighted.

It is not possible to use the interpreter whilst the editor window is open.

9. Some facts about raeval

Raeval is open source and released under the permissive Apache 2.0 license.

It is distributed as an executable JAR (Java Archive) file.

Basic testing has been completed successfully on Windows, Mac and Linux based machines.

The project home is code.google.com/p/relational-algebra/

All of the functionality exposed by the interactive interpreter is coded in the Relation class. This class can be re-used and the source code is available at the project home.

The author of raeval is Nick Everitt, nicholaseveritt@gmail.com -- all feedback gratefully received.