

# Tutorial – Datalog Practice

## Setting up Datalog:

- You can download a copy of the Datalog Engine from the following link :  
<http://www.ugrad.cs.ubc.ca/~cs304/2006W2/tutorials/datalog/des.zip>
- To set it up, unzip the above file and run des.exe. The folder also has another file deswin.exe. I suggest you don't use it as it seems rather unstable.
- To run a command: use /[command name]. The key commands are:
  - /consult - loads a data file
  - /assert - creates a new rule  
[rulename with variables]. - runs a query - note the "." at the end (it's mandatory )
- The main difference between the datalog covered in class, and the datalog in DES:

In DES inside an atom, everything that starts with an uppercase letter or an \_ is a variable and lowercase letters are constants, and in class strings in quotes are variables. So, for example, in DES: father (tom, amy) and father(X, Y) would both be predicates using variables. In DES, the first is a set of constants, and the second is a set of variables.

## Practice Script:

The following template script should get you familiar with using DES. Comments are in red color and console I/O is in blue.

```
*****  
*****
```

DES>

*The general prompt. First, find out what commands are available using the "/help" command:*

DES> /help

Available commands:

- /consult filename Consults a Datalog file, abolishing previous rules
- /c filename Shorthand for /consult filename

- /[filenames] Consults Datalog files, abolishing previous rules
- /reconsult filename Consults a Datalog file, keeping previous rules
- /[+filenames] Consults Datalog files, keeping previous rules
- /assert head:-body Asserts a rule
- /retract head:-body Retracts a rule
- /abolish Abolishes all Datalog rules
- /listing Lists Datalog rules
- /listing name Lists Datalog rules matching a name
- /listing name/arity Lists Datalog rules matching the pattern
- /list\_et Lists contents of extension table
- /list\_et name Lists contents of extension table matching a name
- /list\_et name/arity Lists contents of extension table matching the pattern
- /clear\_et Clears the extension table
- /builtins Lists builtin operators
- /prolog goal Triggers Prolog evaluation for goal
- /cd path Sets the current directory
- /pwd Displays the current directory
- /ls Displays the contents of the current directory
- /ls path Displays the contents of the given path
- /halt Quits DES
- /shell command Submits command to the OS shell
- /help Shows this help

Any other expression is evaluated as a Datalog query

Type des. if you get out of DES from a Prolog interpreter

*The next step is to load a file of rules and atoms using the "/consult" command. In this case, we'll use the example about family trees in the example directory:*

DES> /consult examples/family.dl

Consulting examples/family.dl...

```

father(tom,amy).
father(jack,fred).
father(tony,carolIII).
father(fred,carolIII).
mother(graceI,amy).
mother(amy,fred).
mother(carolI,carolIII).
mother(carolII,carolIII).
parent(_6173,_6193):-father(_6173,_6193).
parent(_6173,_6193):-mother(_6173,_6193).
ancestor(_6173,_6193):-parent(_6173,_6193).
ancestor(_6173,_6193):-parent(_6173,_6252),ancestor(_6252,_6193).

```

yes

*At this point, it's added in all of those facts. Note that it's renamed all existing variables with anonymous variable names (those starting with an \_). Otherwise, it just reprints the facts –remember, every alphanumeric string in a predicate that starts with a lower case letter is a constant.*

*Next, query the database to see what the rule "father" returns. You must give it two variables (upper case letters) to return. Also, don't forget the trailing "."*

```
DES> father(X,Y).
```

```
{  
  father(fred,carolIII),  
  father(jack,fred),  
  father(tom,amy),  
  father(tony,carolIII)  
}
```

```
yes
```

*As we can see, we've retrieved the fathers according to the facts in the database.*

*Next, create a new rule using the "/assert" command. Again, don't forget the trailing "."  
In this case, we'll define a grandmother:*

```
DES> /assert grandmother(X,Y):-mother(X,Z), ancestor(Z,Y).
```

```
yes
```

*now we can query for grandmother, just as we did for father*

```
DES> grandmother(X,Y).
```

```
{  
  grandmother(amy,carolIII),  
  grandmother(carolI,carolIII),  
  grandmother(graceI,carolIII),  
  grandmother(graceI,fred)  
}
```

```
yes
```

*finally, exit the program*

```
DES> /exit
```

```
*****
```

\*\*\*\*\*

**Exercise:**

Load the file Data.dl (/consult Data.dl) . The file is in the same directory as des.exe.

**Practice Queries:**

Consider the *Flights* relation:

*Flights* (*flno*: integer, *from*: string, *to*: string, *distance*: integer, *departs*: time, *arrives*: time)

Write the following queries in datalog:

1. Find the *flno* of all the flights that depart from Madison.
2. Find the *flno* of all flights leaving from Madison that cost less than \$200.
3. Find the *flno* of all flights that leave Chicago after Flight 4884 arrives in Chicago.
4. Find the *flno* of all flights that do not depart from Madison.
5. Find all cities reachable from Madison through a series of one or more connecting flights.
6. Find the *flno* of all flights that do not depart from Madison or a city that is reachable from Madison through a chain of flights.