Outer Joins in a Deductive Database System

Fernando Sáenz Pérez
Grupo de Programación Declarativa (GPD)
Dept. Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

What Outer Joins are All About

```
students(
  name:string,
  subject:string,
  mark:real
        )
```

students

name	*subject*	*mark*
anderson	programming	6
andrews	databases	5
arlingon	databases	3
arlingon	programming	7
norton	databases	6
smith	databases	NULL

```
SELECT name
FROM students
WHERE
  subject='databases'
  AND mark >= 5
```

answer

name
andrews
norton

```
conversion(
  mark:int,
  grade:string
          )
```

conversion

mark	*grade*
...	...
3	D
4	D+
5	C
6	C+
...	...

```
SELECT name,mark,grade
FROM students   AS s,
     conversion AS c
WHERE
  s.subject='databases'
  AND
  s.mark=c.mark
```

answer

name	*mark*	*grade*
andrews	5	C
arlingon	3	D
norton	6	C+

- Now, we recall a database student named Smith
- Where's Smith in the answer?

students

name	*subject*	*mark*
anderson	programming	6
andrews	databases	5
arlingon	databases	3
arlingon	programming	7
norton	databases	6
smith	databases	NULL

```
SELECT name,mark,grade
FROM students AS s
     LEFT OUTER JOIN
     conversion AS c
     ON s.mark=c.mark
WHERE
  s.subject='databases'
```

answer

name	*mark*	*grade*
andrews	5	C
arlingon	3	D
norton	6	C+
smith	NULL	NULL

But, beware ...

```
SELECT name
FROM students
WHERE
  subject='databases'
  and mark >= 5
```

```
SELECT name
FROM students
WHERE
  subject='databases'
  and mark < 5
```

students

name	*subject*	*mark*
anderson	programming	6
andrews	databases	5
arlingon	databases	3
arlingon	programming	7
norton	databases	6
smith	databases	**NULL**

answer

name
andrews
norton

And even worser ...

```
SELECT *
FROM students
WHERE mark IN
    (SELECT mark
     FROM students);
```

- Where's damned Smith in the answer?
- Hasn't Smith the same mark as itself eventually, even if it is unknown yet?

students

name	*subject*	*mark*
anderson	programming	6
andrews	databases	5
arlingon	databases	3
arlingon	programming	7
norton	databases	6
smith	databases	**NULL**

answer

name	*subject*	*mark*
anderson	programming	6
andrews	databases	5
arlingon	databases	3
arlingon	programming	7
norton	databases	6

So:

- NULL's, although subject of semantic flaws, are widely used in current database applications
- However, they are not usual in deductive databases (perhaps to above?)
  - SPARQL
  - 4QL
- This talk will describe a concrete implementation of tabling supporting NULL's and Outer Join operations

Contents

1. Introduction
2. DES, Datalog, and SQL
3. Null Semantics
4. Source-to-Source Transformations
5. Transfers to Other Systems
6. Conclusions

1. Introduction

- So, why deductive databases?
- Databases:
  From relational to deductive
- (Declarative) Query Languages:
  From SQL to Datalog
  - SQL: Extended Relational Algebra
  - Datalog: Predicate Logic

1. Introduction

- Recent commercial deductive systems:
  - DLV (Italy)
  - LogicBlox (USA)
  - Intellidimension (USA)
  - Semmle (UK)
- Recent academic deductive systems:
  - 4QL (Warsaw University)
  - bddbddb (Stanford University)
  - ConceptBase (Tilburg University)
  - DES (Complutense University)

2. DES, Datalog, and SQL
2.1. DES

- Interactive system targeted at teaching databases
- User-friendly (Installation, Usability)
- Free, Open-source, Multiplatform, Portable
- Query languages sharing EDB/IDB:
  - Datalog following ISO Prolog standard
  - (Recursive) SQL following ANSI/ISO standard
- Null value support *à la* SQL
- Outer joins for both SQL and Datalog
- Aggregates
- Stratified negation
- ... and many more

2.2. Datalog (1/3)

- A database query language stemming from Prolog

Prolog	Datalog
Predicate	Relation
Goal	Query

- Meaning of a predicate
  (Multi)set of derivable facts
  - Intensionally (Rules or Clauses)
  - Extensionally (Facts)

2.2. Datalog (2/3)

- Program: Set of rules.
- Rule:
  - `head :- body.`
  - `head.`
- Head: Positive atom.
- Body: Conjunctions (,) and disjunctions (;) of literals
- Literal: Atom, negated atom or built-in.
- Query:
  - Literal with variables or constants in arguments
  - Body (Conjunctive queries, …)
  - Temporary views

2.2. Datalog (3/3)
Outer Joins (1/2)

- Null values:
  - Cte.: `null`
  - Functions: `is_null(Var)`
    `is_not_null(Var)`
- Outer join built-ins:
  - Left: `lj(Left_Rel, Right_Rel, ON_Condition)`
  - Right: `rj(Left_Rel, Right_Rel, ON_Condition)`
  - Full: `fj(Left_Rel, Right_Rel, ON_Condition)`

2.2. Datalog (3/3)
Outer Joins (2/2)

```
lj(a(X), b(Y), X=Y)
SELECT * FROM a LEFT JOIN b ON x=y;

lj(a(X), b(X), true)
SELECT * FROM a LEFT JOIN b WHERE x=y;

lj(a(X), rj(b(Y), c(U,V), Y=U), X=Y)
SELECT * FROM a LEFT JOIN (b RIGHT JOIN c ON y=u) ON x=y;
```

2.2. Datalog (3/3)
Examples

```
SELECT name                          answer(N) :-
FROM   students                       students(N,databases,M),
WHERE  subject='databases'            M>=5
       AND mark >= 5
```

```
SELECT name,mark,grade               answer(N,M,G):-
FROM   students   AS s,               students(N,databases,M),
       conversion AS c                conversion(M,G)
WHERE
  s.subject='databases'
  AND s.mark=c.mark
  AND s.name=c.name
```

2.2. Datalog (3/3)
Examples

```
SELECT name,mark,grade
FROM students AS s
      LEFT OUTER JOIN
      conversion AS c
      ON s.mark=c.mark
WHERE s.subject='databases'
```

```
answer(N,SM,G):-
 lj(students(N,databases,SM),
    conversion(CM,G),
    SM=CM)
```

2.3. SQL

- Follows ISO Standard
- DQL:
  - `SELECT … FROM … WHERE`
  - `WITH RECURSIVE …`
- DML:
  - `INSERT …`
  - `UPDATE …`
  - `DELETE …`
- DDL:
  - `CREATE [OR REPLACE] TABLE …`
  - `CREATE [OR REPLACE] VIEW …`
  - `DROP …`

2.4. Datalog and SQL in DES

- Deductive engine (DE):
  - Tabling implementation
- Datalog programs are solved by DE
- Compilation of SQL views and queries to Datalog programs
- SQL queries are also solved by DE
- Interoperability is allowed: SQL and Datalog do share the deductive database!
  - Datalog queries ↔ SQL queries
  - Datalog typed relations ↔ SQL tables and views
- ODBC connections to external RDBMS's

3. Null Semantics: SQL

Comparison Operator	Cte.$_1$	NULL	Left Argument
Cte.$_2$	True/False **OK**	False ?	
NULL	False ?	False ?	
Right Argument			

- Incomplete 3VL
  - a=a is False for NULL
- False → Unknown

3. Null Semantics: DES

Comparison Operator	Cte.$_1$	NULL$_1$	NULL$_2$	Left Argument
Cte.$_2$	True/False **OK**	False ?	False ?	
NULL$_1$	False ?	True/False **OK**	False ?	
NULL$_2$	False ?	False ?	True/False **OK**	
Right Argument				

- NULL's are distinguishable:
  - `'$NULL'(`*`Id`*`)`
- But still, we are in a 2VL

5. Source-to-Source Transformations

- Recall that `lj(A,B,C)` is the union of:
  - Tuples from A matching C joined with B
  - Tuples from A not matching C joined with NULL's

- E.g.:

```
lj(s(X,U), t(V,Y), U>V)
```

```
s(1,4).
s(2,3).
t(3,5).
```

```
answer(X, Y, U, V) →
  { answer(1, 4, 3, 5),
    answer(2, 3, null, null) }
```

5. Source-to-Source Transformations

```
v(X,Y) :- lj(s(X,U),t(V,Y),U>V).
```

```
v(X,Y) :- lj('$p0'(X,U,V,Y)).
```

```
'$p0'(A,B,C,D) :-
    '$p1'(A,B,C,D).
'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :-
    s(A,B),
    not('$p1'(A,B,E,F)).
```

```
'$p1'(A,B,C,D) :-
    s(A,B), t(C,D), B > C.
```

4. Source-to-Source Transformations

- But, we get an *unsafe* rule because of *floundering*:

```
'$p0'(A,B,'$NULL'(C),'$NULL'(D))
```

- However, such NULL specifications are otherwise treated as *null providers*
- A null provider returns a unique identifier for a given tuple of ground values

4. Source-to-Source Transformations

```
'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :-
    s(A,B),
    not('$p1'(A,B,E,F)).
```

```
s(1,4).
s(2,3).

t(3,5).
```

```
'$p0'(2,3,'$NULL'(1),'$NULL'(2))
```

```
'$p1'(A,B,C,D) :-
    s(A,B), t(C,D), B > C.
```

```
'$p1'(1,4,3,5)
```

5. Transfers to Other Systems

- The transformation includes a floundering rule, as E and F are not *range restricted* :

```
'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :-
    s(A,B),
    not('$p1'(A,B,E,F)).
```

- The meaning of `not('$p1')` is unsafe, as it contains unbounded arguments:

```
not('$p1'(2,3,A,B))
```

5. Transfers to Other Systems

- Usually, floundering is not allowed as in DLV
- However, some floundering programs can be translated into non-floundering [Ullman]
- This time we are lucky

5. Transfers: DLV

```
v(X,Y) :- lj(s(X,U),t(V,Y),U>V).
```

```
v(X,Y) :- '$p0'(X,U,V,Y).
```

```
'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :-
    s(A,B), not('$p1'(B)).
'$p0'(A,B,C,D) :-
    s(A,B), t(C,D), B > C.
```

- Solved at an extra cost

```
'$p1'(B) :-
    s(A,B), t(C,D), B > C.
```

5. Transfers to Other Systems

- Another state-of-the-art system is XSB
- Here, built-in `sk_not/1` allows floundering by program transformation

5. Transfers: XSB

```
v(X,Y) :- lj(s(X,U),t(V,Y),U>V).
```

```
:- table('$p0'/4), table('$p1'/4).
:- table(s/2), table(t/2).
main(Vs) :- findall(v(X,Y),v(X,Y),Vs).
v(X,Y) :- '$p0'(X,U,V,Y).
'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :-
   get_id(C), get_id(D), s(A,B),
   sk_not('$p1'(A,B,E,F)).
'$p0'(A,B,C,D) :- '$p1'(A,B,C,D).
'$p1'(A,B,C,D) :- s(A,B), t(C,D), B > C.
:- dynamic id/1.
id(0).
get_id(X) :-
 id(X), retractall(id(X)), Y is X+1, assertz(id(Y)).
```

6. Conclusions

- SQL NULL values in DDB's
  - 2-Valued Logic
  - Similar behavior, but for comparing the same NULL
  - Easily modeled in DDB's: `'$NULL'`(*Id*)
- SQL Outer Joins as Program Transformations
  - It can be done, but, better, native support for NULL providers
  - DES natively implements them, but transfers to other systems have been highlighted

6. Conclusions (2/2)

- DES:
  - Successful implementation guided by need
  - Widely used, both for teaching and research
    - More than 35,000 downloads since 2004
    - Referenced by ACM SIGMOD Group
    - Companies: XLOG Technologies, CaseLab: Applied Operations Research, Ideacube, LogicBlox
    - Wikipedia

DES Facts

Canada

- Efficient Integrity Checking for Databases with Recursive Views
  Davide Martinenghi and Henning Christiansen
  In Advances in Databases and Information Systems: 9th East European Conference, ADBIS 2005, Tallinn, Estonia, September 12-15, 2005 : Proceedings
  Autor Johann Eder, Hele-Mai Haav, Ahto Kalja, Jaan Penjam
  ISBN 3540285857, 9783540285854
- PhD
  Computer Science and Engineering Department
  University of Nebraska - Lincoln, USA
- PhD
  University of Texas at San Antonio, USA

Industry:

- XLOG Technologies GmbH, Zürich
- CaseLab : Applied Operations Research
- Ideacube

Links to DES:

- ACM SIGMOD Online Publicly Available Database Software from Nonprofit Organizations
- The ALP Newsletter. vol. 21 n. 1
- Datalog Wikipedia German
- Datalog Wikipedia English
- Wapedia
- SWI-Prolog. Related Web Resources
- SICStus Prolog. Third Party Software. Other Research Systems
- SOFTPEDIA. Datalog Educational System 1.7.0
- Famouswhy
- DBpedia
- BDD-Based Deductive DataBase (bddbddb)
  Other implementations of Datalog/Prolog
- Reach Information
- Ask a Word
- Acronym finder
- Acronym Geek
- .

University of California, at Los Angeles

- CS240A: Databases and Knowledge Bases
- The University of Arizona
  CsC372
- The State University of New York
  University at Buffalo
  CSE 636: Data Integration
- The University of British Columbia
  CS304: Introduction to Relational Databases
  Datalog Tutorial
- Master's of Information Technology in Arkansas Tech University, Russellville
- The University of Texas at Austin
  CS2

Australia:

- INFO2820: Database Systems 1 (Advanced) (2010 - Semester 1)
  Engineering and Information Technologies
  The University of Sydney
  Tab "Resources"
- INFO2120/2820: Database Systems 1 (2009 - Semester 1)
  School of Information Technologies
  The University of Sydney
  Tutorial 3
- Allan Hancock College >> INFO >> 2120 Fall, 2009
  Description: School of Information Technologies
  INFO2120/2820: Database Systems I 1.Sem./2009
  Tutorial 3: SQL and Relational Algebra 23.03.2009

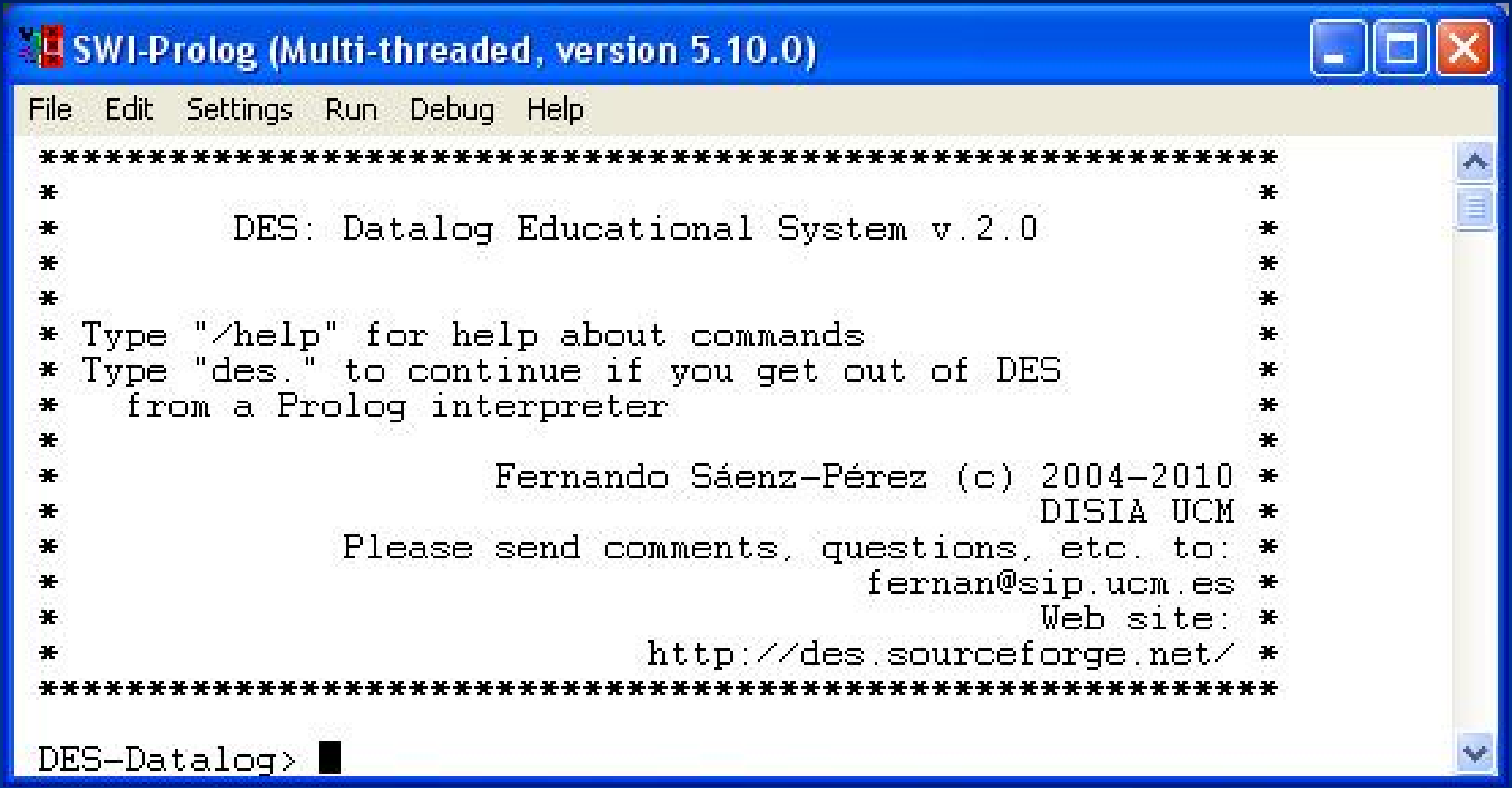
Africa:

- Faculty of Sciences and Technologies of Mohammedia (FSTM) - Morocco

Installing DES

- Distro under GPL in Sourceforge:
  - Sources
  - Portable Executables (Windows, Linux)
  - Portable Bundle including Java IDE (Windows)
- Starting the system. Either:
  - From a Prolog interpreter (Ciao, GNU, Sicstus, SWI)
  - Simply execute the binary
  - Start the Java application

DES running as a Windows application

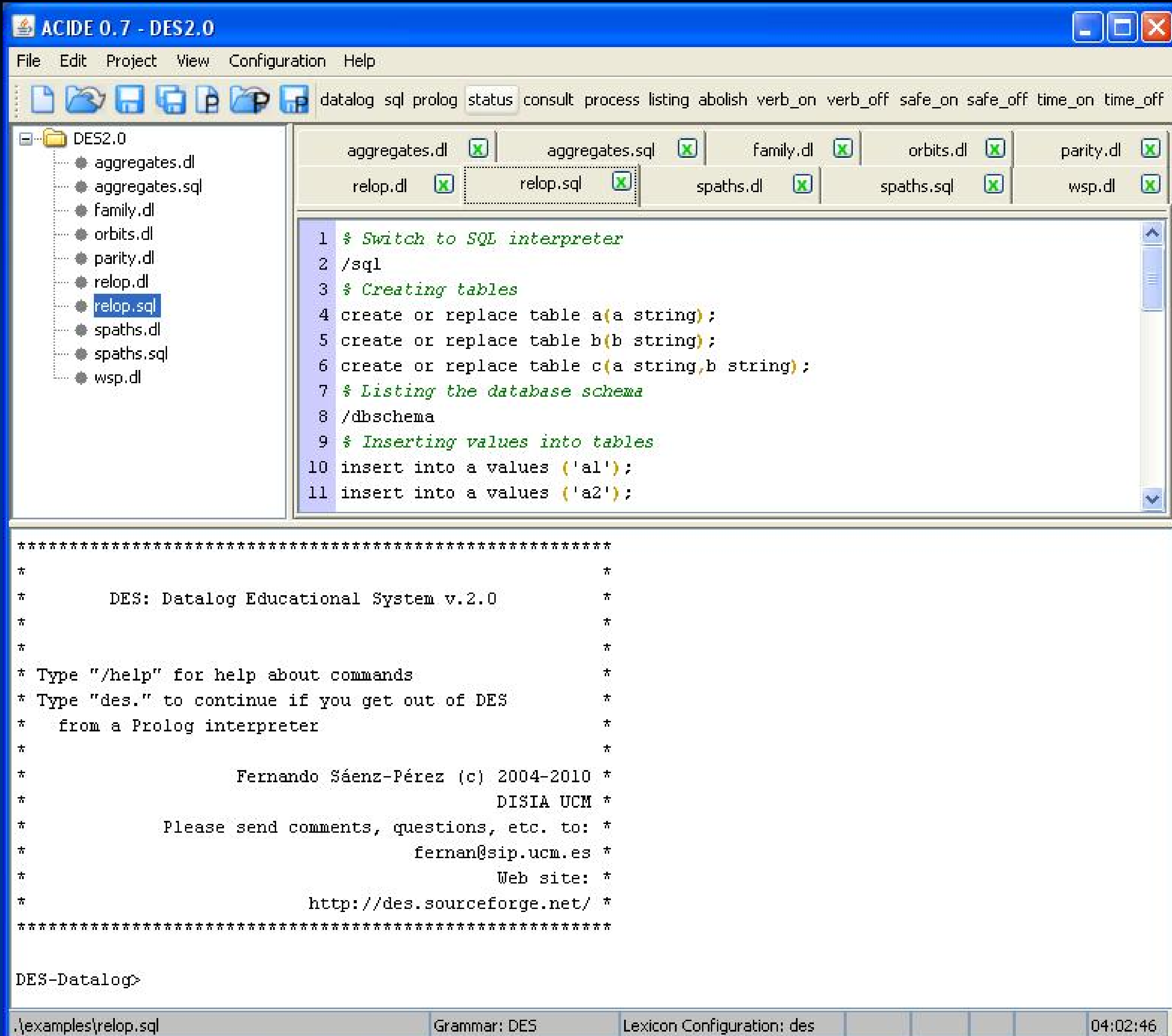


DES running in a Linux terminal

```
fernan@fernan-ubuntu: ~/Escritorio/des
Archivo  Editar  Ver  Terminal  Ayuda

fernan@fernan-ubuntu:~/Escritorio/des$ ./des
*********************************************************
*                                                       *
*         DES: Datalog Educational System v.2.0         *
*                                                       *
*                                                       *
* Type "/help" for help about commands                  *
* Type "des." to continue if you get out of DES         *
*   from a Prolog interpreter                           *
*                                                       *
*                 Fernando S�enz-P�rez (c) 2004-2010    *
*                                             DISIA UCM *
*              Please send comments, questions, etc. to: *
*                                        fernan@sip.ucm.es *
*                                             Web site: *
*                            http://des.sourceforge.net/ *
*********************************************************

DES-Datalog>
```


ACIDE 0.7 - DES2.0
File Edit Project View Configuration Help
datalog sql prolog status consult process listing abolish verb_on verb_off safe_on safe_off time_on time_off
DES2.0
aggregates.dl
aggregates.sql
family.dl
orbits.dl
parity.dl
relop.dl
relop.sql
spaths.dl
spaths.sql
wsp.dl
% Switch to SQL interpreter
/sql
% Creating tables
create or replace table a(a string);
create or replace table b(b string);
create or replace table c(a string,b string);
% Listing the database schema
/dbschema
% Inserting values into tables
insert into a values ('a1');
insert into a values ('a2');
DES: Datalog Educational System v.2.0
Type "/help" for help about commands
Type "des." to continue if you get out of DES
from a Prolog interpreter
Fernando Sáenz-Pérez (c) 2004-2010
DISIA UCM
Please send comments, questions, etc. to:
fernan@sip.ucm.es
Web site:
http://des.sourceforge.net/
DES-Datalog>
.\examples\relop.sql
Grammar: DES
Lexicon Configuration: des
04:02:46

emacs@fernan-ubuntu

File Edit Options Buffers Tools Complete In/Out Signals Help

```
father(tom,amy).
father(jack,fred).
father(tony,carolII).
father(fred,carolIII).
mother(grace,amy).
mother(amy,fred).
mother(carolI,carolII).
mother(carolII,carolIII).

parent(X,Y) :-
  father(X,Y)
--(DOS)---  family.dl      Top L1     (DES)------------------------------------
*********************************************************
*                                                       *
*        DES: Datalog Educational System v.2.0          *
*                                                       *
*                                                       *
* Type "/help" for help about commands                  *
* Type "des." to continue if you get out of DES         *
*   from a Prolog interpreter                           *
*                                                       *
*                    Fernando S�enz-P�rez (c) 2004-2010 *
*                                            DISIA UCM *
*             Please send comments, questions, etc. to: *
*                                      fernan@sip.ucm.es *
*                                            Web site: *
*                            http://des.sourceforge.net/ *
*********************************************************

DES-Datalog>
-U:**-  *des*        66% L61    (Comint:run)------------------------------------
```

Implementation

- DES command-line interpreter: Prolog
  - Tabling (Bottom-up Top-down driven)
  - Computation by strata saturations (negation and aggregates)
- Datalog Debugger: Prolog + Java
  - [CGS07] R. Caballero, Y. García-Ruiz, and F. Sáenz-Pérez, A new proposal for debugging Datalog programs. WFLP'07
- SQL Debugger: Prolog
  - [CGS11] R. Caballero, Y. García-Ruiz, and F. Sáenz-Pérez, Algorithmic Debugging of SQL Views, PSI'11
- Test Case Generator: Prolog + FD constraints
  - [CGS10a] R. Caballero, Y. García-Ruiz, and F. Sáenz-Pérez, Applying Constraint Logic Programming to SQL Test Case Generation, FLOPS 2010
- ACIDE: Java
  - A Configurable IDE (LaTeX, SQL, Prolog, Datalog, …)
  - [Sae07b] F. Sáenz-Pérez, "ACIDE: An Integrated Development Environment Configurable for LaTeX", The PracTeX Journal, 2007.