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  
)  

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

<table>
<thead>
<tr>
<th>name</th>
<th>subject</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>anderson</td>
<td>programming</td>
<td>6</td>
</tr>
<tr>
<td>andrews</td>
<td>databases</td>
<td>5</td>
</tr>
<tr>
<td>arlingon</td>
<td>databases</td>
<td>3</td>
</tr>
<tr>
<td>arlingon</td>
<td>programming</td>
<td>7</td>
</tr>
<tr>
<td>norton</td>
<td>databases</td>
<td>6</td>
</tr>
<tr>
<td>smith</td>
<td>databases</td>
<td>NULL</td>
</tr>
</tbody>
</table>

answer

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>andrews</td>
</tr>
<tr>
<td>norton</td>
</tr>
</tbody>
</table>

PROLE 2011  7/9/2011
conversion(
    mark:int,
    grade:string
)

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

<table>
<thead>
<tr>
<th>mark</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>D+</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>6</td>
<td>C+</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>mark</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>andrews</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>arlington</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>norton</td>
<td>6</td>
<td>C+</td>
</tr>
</tbody>
</table>
Now, we recall a database student named Smith

Where's Smith in the answer?

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

<table>
<thead>
<tr>
<th>name</th>
<th>subject</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>anderson</td>
<td>programming</td>
<td>6</td>
</tr>
<tr>
<td>andrews</td>
<td>databases</td>
<td>5</td>
</tr>
<tr>
<td>arlingon</td>
<td>databases</td>
<td>3</td>
</tr>
<tr>
<td>arlingon</td>
<td>programming</td>
<td>7</td>
</tr>
<tr>
<td>norton</td>
<td>databases</td>
<td>6</td>
</tr>
<tr>
<td>smith</td>
<td>databases</td>
<td>NULL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>mark</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>andrews</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>arlingon</td>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>norton</td>
<td>6</td>
<td>C+</td>
</tr>
<tr>
<td>smith</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>
But, beware ...

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

```
SELECT name
FROM students
WHERE
    subject = 'databases'
    AND mark < 5
```
And even worser ...

```sql
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?
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)
  - bdddbdd (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
A database query language stemming from Prolog

<table>
<thead>
<tr>
<th>Prolog</th>
<th>Datalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate</td>
<td>Relation</td>
</tr>
<tr>
<td>Goal</td>
<td>Query</td>
</tr>
</tbody>
</table>

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)`
outer joins (2/2)
2.2. Datalog (3/3)
Examples

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

SELECT name, mark, grade
FROM   students   AS s,
        conversion AS c
WHERE  s.subject='databases'
AND    s.mark=c.mark
AND    s.name=c.name
answer(N,M,G) :-
students(N,databases,M),
conversion(M,G)
```
2.2. Datalog (3/3) Examples

```
SELECT name, mark, grade
FROM students AS s
  LEFT OUTER JOIN conversion AS c
  ON s.marks.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 \(\leftrightarrow\) SQL queries
  - Datalog typed relations \(\leftrightarrow\) SQL tables and views
- ODBC connections to external RDBMS's
### 3. Null Semantics: SQL

<table>
<thead>
<tr>
<th>Comparison Operator</th>
<th>Cte.₁</th>
<th>NULL</th>
<th>Left Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cte.₂</td>
<td>True/False</td>
<td>False</td>
<td>?</td>
</tr>
<tr>
<td>NULL</td>
<td>False</td>
<td>False</td>
<td>?</td>
</tr>
<tr>
<td>Right Argument</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key Points:**
- Incomplete 3VL
- a=a is False for NULL
- False → Unknown
## 3. Null Semantics: DES

<table>
<thead>
<tr>
<th>Comparison Operator</th>
<th>Cte.₁</th>
<th>NULL₁</th>
<th>NULL₂</th>
<th>Left Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cte.₂</td>
<td>True/False OK</td>
<td>False ?</td>
<td>False ?</td>
<td></td>
</tr>
<tr>
<td>NULL₁</td>
<td>False ?</td>
<td>True/False OK</td>
<td>False ?</td>
<td></td>
</tr>
<tr>
<td>NULL₂</td>
<td>False ?</td>
<td>False ?</td>
<td>True/False OK</td>
<td></td>
</tr>
<tr>
<td>Right Argument</td>
<td>NULL₁</td>
<td>NULL₂</td>
<td>NULL₁</td>
<td></td>
</tr>
</tbody>
</table>

- **NULL's are distinguishable:**
  - '\$NULL' (Id)

- But still, we are in a 2VL
5. Source-to-Source Transformations

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

- E.g.:

  \[
  l_j(s(X, U), t(V, Y), U>V)
  \]

  \[
  \begin{align*}
  s(1,4) & .
  
  s(2,3) & .
  
  t(3,5) & .
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{answer}(X, Y, U, V) \rightarrow 
  
  \{ \text{answer}(1, 4, 3, 5), 
  
  \text{answer}(2, 3, \text{null}, \text{null}) \} 
  \end{align*}
  \]
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

\[
'sp0'(A, B, 'NULL'(C), 'NULL'(D)) :-
\]
\[
s(A, B),
\]
\[
\textbf{not}('sp1'(A, B, E, F)).
\]

\[
'sp0'(1, 4, 'NULL'(1), 'NULL'(2))
\]

\[
'sp1'(A, B, C, D) :-
\]
\[
s(A, B), \ t(C, D), \ B > C.
\]

\[
'sp1'(1, 4, 3, 5)
\]
5. Transfers to Other Systems

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

\[
\text{'$p0'(A,B,'$NULL'(C),'$NULL'(D)) :- s(A,B), not('$p1'(A,B,E,F)).}
\]

- The meaning of \text{not('$p1') is unsafe, as it contains unbounded arguments:}

\[
\text{not('$p1'(2,3,A,B))}
\]
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

\[ v(X, Y) :- \text{lj}(s(X, U), t(V, Y), U > V). \]

\[ v(X, Y) :- '\$p0'(X, U, V, Y). \]

\[ '\$p0'(A, B, '\$NULL'(C), '\$NULL'(D)) :- s(A, B), \text{not}( '\$p1'(B)). \]

\[ '\$p0'(A, B, C, D) :- s(A, B), t(C, D), B > C. \]  
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

\[ v(X, Y) :\leftarrow l^j(s(X, U), t(V, Y), U>V). \]

\[ :- \text{table}('p0'/4), \text{table}('p1'/4). \]
\[ :- \text{table}(s/2), \text{table}(t/2). \]

\[ \text{main}(V_s) :\leftarrow \text{findall}(v(X, Y), v(X, Y), V_s). \]
\[ v(X, Y) :\leftarrow 'p0'(X, U, V, Y). \]
\[ 'p0'(A, B, 'NULL'(C), 'NULL'(D)) :\leftarrow \]
\[ \text{get}_\text{id}(C), \text{get}_\text{id}(D), s(A, B), \]
\[ \text{sk}_\text{not}('p1'(A, B, E, F)). \]
\[ 'p0'(A, B, C, D) :\leftarrow 'p1'(A, B, C, D). \]
\[ 'p1'(A, B, C, D) :\leftarrow s(A, B), t(C, D), B > C. \]

\[ :- \text{dynamic id/1}. \]
\[ \text{id}(0). \]
\[ \text{get}_\text{id}(X) :\leftarrow \]
\[ \text{id}(X), \text{retractall(id}(X)), Y \text{ is } X+1, \text{assertz(id}(Y)). \]
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
  - Wikipedia
Efficient Integrity Checking for Databases with Recursive Views
Davide Martinenghi and Henning Christiansen
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 (bddbddd)
- Other implementations of Datalog/Prolog
- Reach Information
- Ask a Word
- Acronym finder
- Acronym Geek

Australia:
- INFO2820: Database Systems 1 (Advanced) (2010 - Semester 1)
Engineering and Information Technologies
The University of Sydney
Tab “Resources”

Tutorial 3

Africa:
- Faculty of Sciences and Technologies of Mohammeda (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

**SWI-Prolog (Multi-threaded, version 5.10.0)**

```
*  
* 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>**
```
DES running in a Linux terminal

```
fernан@fernан-ubuntu:~/Escritorio/des$
.

fernан@fernан-ubuntu:~/Escritorio/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 Sánchez-Pérez (c) 2004-2010             |
| DISIA UCM                                       |
| Please send comments, questions, etc. to:       |
| fernan@sip.ucm.es                               |
| Web site:                                        |
| http://des.sourceforge.net/                     |
+--------------------------------------------------+

DES-Datalog>
```
PROLE 2011   7/9/2011

DES running under ACIDE

---

1 % Switch to SQL interpreter
2 /sql
3 % Creating tables
4 create or replace table a(a string);
5 create or replace table b(b string);
6 create or replace table c(a string, b string);
7 % Listing the database schema
8 /dbschema
9 % Inserting values into tables
10 insert into a values ('a1');
11 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 Grammer: DES Lexicon Configuration: des 04:02:46
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)

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{\`u}enz-P{\`e}rez (c) 2004-2010
DISIA UCM
Please send comments, questions, etc. to:
fernansip.ucm.es
Web site:
http://des.sourceforge.net/
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
- 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, …)