

Deductive Databases and XSB

Advanced Database Project for IT4BI Masters 2013-2015

12/29/2013

Kofi Manful

Table of Contents

Introduction	3
Background	3
Building Blocks	3
Advantages.....	4
Disadvantages.....	5
Uses.....	5
XSB	6
Intro.....	6
Sample XSB Programs	6
Fibonacci	6
Shortest Path.....	7
Connection to Oracle DB.....	9
Puzzle/Problem Solving.....	9
Conclusion.....	12
Sources.....	13

Introduction

Relational Databases dominate the database market today. This can be attributed to their simple model and their efficient performance of everyday tasks. However deductive databases offer an interesting alternative to relational databases in areas such as recursion and information retrieval. In this paper, we will give a brief background and description of deductive databases and some of their main functionality. We will then illustrate these with an implementation of a deductive database called XSB Prolog, which is an open source deductive database being developed by several institutions (University of New York, XSB Inc, Universidade Nova de Lisboa among others).

Background

What we now call deductive databases started from research in two fields, namely Artificial Intelligence and Databases. While the former concentrated more on topics such as knowledge representation and language processing and dealt with main memory mainly, the latter focused more on the efficient storage and retrieval of data on secondary memory and the associated issues with data security and concurrent access. In the 1970s, deductive databases became a topic of their own and research in the following years has produced some foundational theories (Grant & Minker, 1989) (Zaniolo, 1990), some of which will be mentioned in this paper.

Building Blocks

Deductive databases are the combination of logic programming and database systems. This means that not only can they store data like traditional relational databases, but they make logical connections between the data provided to generate results for other queries. Information in a deductive database is specified through rules, predicates, facts and goals such as the ones below:

1. Employee(Name, Position, Salary, ReportsTo)
2. Employee('Jack Densu', 'Accounts Manager', 60000, 'Kwasi Mensah')
3. TopEarnings(Name) \leftarrow Employee(Name, Position, Salary, ReportsTo), Salary > 100000
4. Superior (Boss, Subordinate) \leftarrow Employee (Subordinate, Position, Salary, Boss).
Superior (Boss, Subordinate) \leftarrow Employee(EmpName, Position, Salary, Boss),
Superior (Empname, Subordinate)
5. ?TopEarnings('Kwasi Mensah')
6. ?TopEarnings(Name)

The example we see in number 1 is a specification is called a predicate. It specifies an entity and its attributes. If we were to draw a parallel between relational databases, we would call this the schema of the entity. The number of arguments/attributes the predicate takes is referred to as the arity and in this example for Employee, the arity is 4.

When we have a predicate in which all the attributes have taken on actual values, such as in example 2, we call it a fact. A fact specifies an instance of an entity and the values it contains. The equivalent in a relational database would be a tuple in a table. Therefore we can think of an Employee table as being made up of many facts.

Example 3 illustrates what is called a rule. A rule is made up of a head and a body separated by the \leftarrow symbol. One can think of the head of a rule as one would think of a view in a relational database. The body of the rule is made up of goals. These goals are separated by commas (,) which can be interpreted as the AND operator. The goals specify the criteria used to define the rule. In this case, the rule states that TopEarnings can be defined by every tuple in the Employee "table" which has a salary greater than 100,000.

Example 4 is also a rule but in this case the head of the rule is defined twice. Because the two definitions are connected by a period (.), this means any tuples that meet either the first set of goals or the second will be in the head of the rule. This kind of definition allows us to express transitive relationships more easily than a relational database would. In this example, we are specifying that an employee is a superior to another if the first employee is either the direct boss of the second employee, or if the first employee is the direct boss of someone who is a superior to the second employee.

Examples 5 and 6 show how queries are written when interacting with deductive databases. In example 5, because a constant has been specified as a parameter, the query will return true or false to indicate if that constant meets the requirements specified by that predicate (i.e. if there is a Kwasi Mensah who is an employee who earns more than 100,000). Example 6 has a variable as an argument so the result of the query will be all the employees who meet the requirements specified by the TopEarner rule.

Advantages

Deductive databases provide a few advantages over relational databases in a few areas. One of the major ones is the ability to do recursion and the ease with which such recursion can be defined. Although some commercial databases do allow you to define recursive relationships, this has only been a recent development. The SQL standard only added this in 1999 and it took a few years for some of the major relational databases to implement it (Microsoft SQL Server, PostgreSQL and a few others as of 2011).

Secondly the syntax and structure for doing this with deductive databases is so much more intuitive and simple than the SQL equivalent. Compare example 4 above which defines the Superior relationship with the SQL equivalent below:

```
CREATE RECURSIVE VIEW Subordinates(Superior, subordinate) AS
```

```
SELECT ReportsTo, Name
```

```
FROM Employee
```

```
UNION
```

```
SELECT sub.Superior, emp.name
```

```
FROM Subordinates sub, Employee emp
```

```
WHERE sub.subordinate=emp.ReportsTo
```

The deductive database syntax is clearly simpler to construct and much easier to maintain.

Disadvantages

One big disadvantage of deductive database technology is the lack of widespread development and support. There are very few operational deductive databases on the market and most of them are still experimental. Because there isn't as large a community around deductive databases as there is around relational databases, there really isn't any system that has been extensively used. Therefore it would be difficult for any company/individual to switch to a deductive database from a relational one. Even if one was able to do this, the significant lack of extensive documentation is also another obstacle.

Another weakness is the inability to deal with unstructured data. As the data in a deductive database is specified using predicates and rules and goals, there does have to be a clear relationship between the information being stored in the database. Without this it is very difficult to store data in a deductive database.

Uses

Deductive databases and their concepts are used in artificial intelligence systems and expert systems. Their ability to make connections between pieces of data and the manner in which information is specified using rules makes deductive databases ideal for this. Here are some real world applications below:

The Cyc System:- An attempt to create a program with enough understanding to learn from books

PACADE:- Protein Atomic Coordinate Analyzer with Deductive Engine, a system for searching for and analyzing proteins

XSB:- a deductive database which is used in other projects such as Semantic Inferencing on Large Knowledge (SILK), OpenSHORE (a hypertext repository that stores data about and described by documents) among others.

XSB

Intro

XSB is a logic programming and deductive database system developed by research teams from the Computer Science Department of Stony Brook University, Universidade Nova de Lisboa, XSB, Inc, Katholieke Universiteit Leuven, and Uppsala Universitet. It runs on UNIX and Windows and the code is open source allowing anyone to contribute to its development. It is written in C++ and as such one needs to have a C++ compiler to build it before being able to run it. The system itself is rather lightweight and able to run on a computer of average specifications.

The system is operated with a command line interface. Documentation on the commands needed to setup and run XSB are included in the package that can be downloaded from the source cite. The package also includes some example databases and programs. We shall highlight some of them here to demonstrate some of the features of XSB.

Sample XSB Programs

Fibonacci

We start off with a simple program that calculates the n^{th} term in the Fibonacci series. Here is the XSB code below:

```
demo :-
    write('Enter N : '), read(N), nl,
    fib(N, Fib),
    write('Fib of '), write(N), write(' is '), writeln(Fib).

fib(N, X) :- fib0(N, X, _).

/* fib0(+Arg, -Result, -MinusOneRes) */

fib0(0, 1, 0).
fib0(1, 1, 1).
fib0(N, X, X1) :-
    N > 1,
    N2 is N - 2,
    fib0(N2, X2, X3),
    X1 is X2 + X3,
    X is X1+X2.
```

In this example we see the rule “demo” being defined as a small script that reads in a value and uses it to calculate the value of the Fibonacci number at that position before outputting it to the user. The Fibonacci function itself is defined recursively beneath that. Shown below is the output of running this program. For convenience sake, the XSB code was loaded as a file.

```

C:\Windows\system32\cmd.exe - "C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\config\x...
C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\examples>"C:\Users\Jazz\Dropb
ox\IT4BI\Advanced Databases\XSB\config\x64-pc-windows\bin\xsb.exe"
[xsb_configuration loaded]
[sysinitrc loaded]

XSB Version 3.4.0 (Soy mILK) of May 1, 2013
[x64-pc-windows; mode: debug; engine: slg-wam; scheduling: local]
[Patch date: 2013/05/02 17:42:32]

| ?- [fib].
[fib loaded]

yes
| ?- demo.
Enter N : 17.

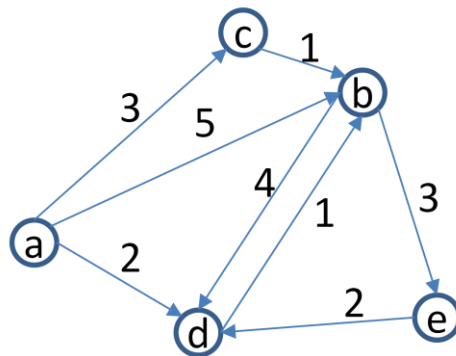
Fib of 17 is 2584

yes
| ?-

```

Shortest Path

The well known shortest path problem can also be solved using XSB. We show below the graph structure from which the code is based.



The following code will find the shortest path from “a” to every other node using the ability of deductive databases to discover transitive relationships between elements. Once again a recursive definition is used for the shortest path rule. Below that, the facts which make up the database are stated. Further below is the result of querying this database.

```

demo :-
    sp(a,X,C),
    write('Best cost so far from a to '),write(X),write(' is
'),writeln(C),fail.

```

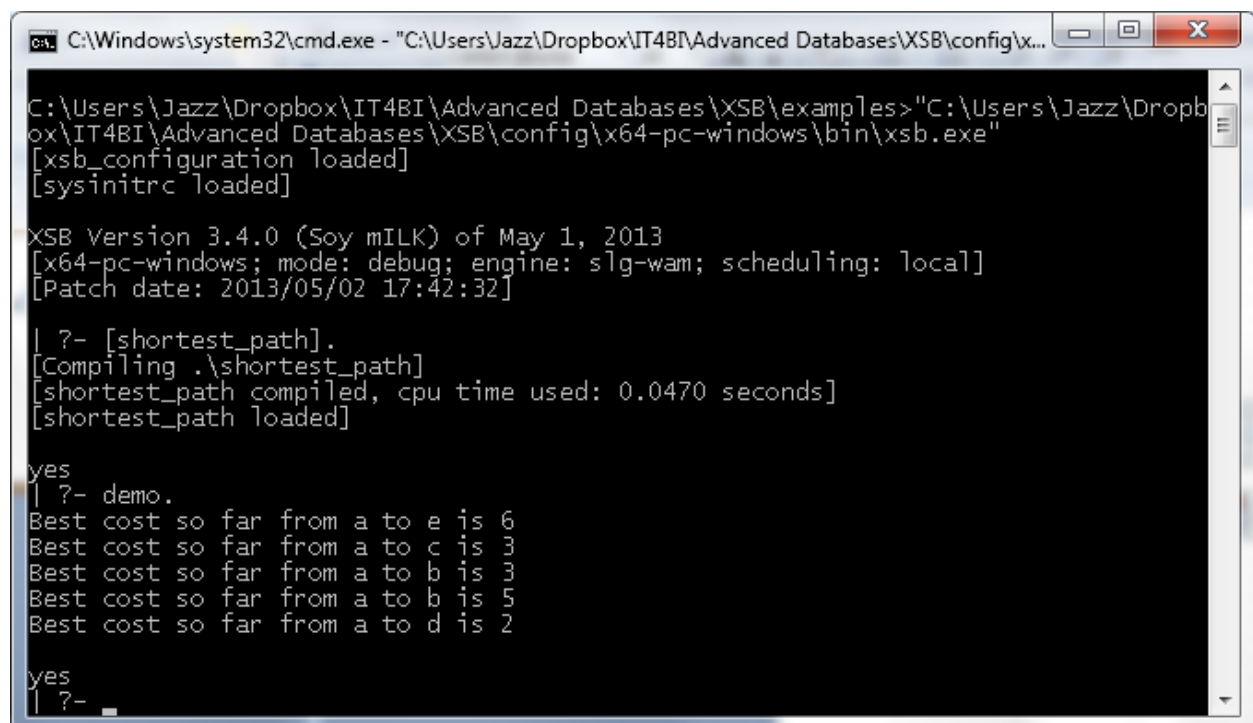
demo.

```
sp(X,Y,C) :-  
    dist(X,Y,C),  
    none_better(X,Y,C).
```

```
sp(X,Y,C) :-  
    sp(X,Z,C1),  
    none_better(X,Z,C1),  
    dist(Z,Y,C2),  
    C is C1+C2,  
    none_better(X,Y,C).
```

```
none_better(X,Y,C) :-  
    get_calls_for_table(sp(X,_,_),Call),  
    subsumes_chk(Call,sp(X,Y,_)),  
    !,  
    \+ ( get_returns_for_call(Call,Ret),  
        Ret = sp(X,Y,C1),  
        C1 < C  
    ).
```

```
dist(a,d,2).  
dist(a,b,5).  
dist(a,c,3).  
dist(c,b,1).  
dist(b,e,3).  
dist(b,d,4).  
dist(e,d,2).  
dist(d,b,1).
```



```
C:\Windows\system32\cmd.exe - "C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\config\x...  
C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\examples>"C:\Users\Jazz\Dropb  
ox\IT4BI\Advanced Databases\XSB\config\x64-pc-windows\bin\xsb.exe"  
[xsb_configuration loaded]  
[sysinitrc loaded]  
  
XSB Version 3.4.0 (Soy mILK) of May 1, 2013  
[x64-pc-windows; mode: debug; engine: slg-wam; scheduling: local]  
[Patch date: 2013/05/02 17:42:32]  
  
| ?- [shortest_path].  
[Compiling .\shortest_path]  
[shortest_path compiled, cpu time used: 0.0470 seconds]  
[shortest_path loaded]  
  
yes  
| ?- demo.  
Best cost so far from a to e is 6  
Best cost so far from a to c is 3  
Best cost so far from a to b is 3  
Best cost so far from a to b is 5  
Best cost so far from a to d is 2  
  
yes  
| ?-
```


Connection to Oracle DB

XSB is also capable of connecting to an Oracle database and performing transactions to retrieve or store data. Some of the commands used to do that are listed below:

db_open():- uses the values given as parameters to connect to the Oracle database.

db_sql():- executes whatever sql code is passed as a parameter.

db_create_table():- a command for creating a table in the Oracle database. Instead of writing the whole SQL create table syntax using the db_sql() command, one can just specify the fields and data types and this command will create table.

db_transaction(): - used to start, commit or abort Oracle transactions depending on the parameter that is passed

db_show_schema():- used to display the schema of a table in the Oracle database

db_import ():- a command for loading content from an oracle database into a set of facts in XSB

Puzzle/Problem Solving

Since XSB is a deductive database, it can be used to deduce information when presented with facts. In the example below, by cleverly setting up goals and rules, we use XSB to find the solution to the following puzzle:

In a street, there are five houses. Each is painted with a different colour. In each house lives somebody coming from a different country. Each has a favorite pet, a favorite drink and a favorite cigarette brand. We know the following facts:

The English lives in the red house. The dog belongs to the Spanish. One uses to drink coffee in the green house. The Ukrainian drinks tea. The green house is on the right of the white one. The Old Gold smoker breeds snails. The occupant of the yellow house smokes Kool. The occupant of the middle house drinks milk. The Norwegian lives in the first house on the left. The smoker of Chesterfield lives beside the owner of the fox. The smoker of Kool lives beside the owner of the horse. The Gitanes smoker drinks wine. The Japanese smokes Craven. The Norwegian lives beside the blue house.

Question: Who breeds the zebra? Who drinks water?

Shown below is the XSB code that states the problem in terms of rules and goals. As can be seen, it does require some ingenuity and experience to be able structure the problem statement properly. However when this done, the solution is easy to deduce as seen in the subsequent image.

```

demo :- bagof( X, go3( X ), L ), write(L), nl.

houses_iter(0) :- !.
houses_iter(_ ) :- bagof( X, go3( X ), _ ), fail.
houses_iter(N) :- M is N-1,
                 houses_iter(M).

%% memb/2
%%-----
memb( X, [_|Y] ) :- memb( X, Y ).
memb( X, [X|_] ).

%% testp/4
%%-----
testp( X, Y, [X|_] , [Y|_] ).
testp( X, Y, [_|R] , [_|S] ) :- testp( X, Y, R, S ).

testp_seq( X, Y, [X|_] , [Y|_] ).
testp_seq( X, Y, [_|R] , [_|S] ) :- testp_seq( X, Y, R, S ).

%% on_right/4
%%-----
on_right( X, Y, [X|_] , [_, Y|_] ).
on_right( X, Y, [_|R] , [_|S] ) :- on_right( X, Y, R, S ).

on_right_seq( X, Y, [X|_] , [_, Y|_] ).
on_right_seq( X, Y, [_|R] , [_|S] ) :- on_right_seq( X, Y, R, S ).

%% on_left/4
%%-----
on_left( X, Y, [_, X|_] , [Y|_] ).
on_left( X, Y, [_|R] , [_|S] ) :- on_left( X, Y, R, S ).

on_left_seq( X, Y, [_, X|_] , [Y|_] ).
on_left_seq( X, Y, [_|R] , [_|S] ) :- on_left_seq( X, Y, R, S ).

%% beside/4
%%-----
beside( X, Y, Xs, Ys ) :- on_right( X, Y, Xs, Ys ).
beside( X, Y, Xs, Ys ) :- on_left( X, Y, Xs, Ys ).

beside_seq( X, Y, Xs, Ys ) :- on_right_seq( X, Y, Xs, Ys ).
beside_seq( X, Y, Xs, Ys ) :- on_left_seq( X, Y, Xs, Ys ).

%% go3/1
%%-----
go3( config( Countries, Colours, Animals, Drinks, Cigarettes ) ) :-
    Countries = [norway, _, _, _, _],           % const 9
    Colours = [_, _, _, _, _],
    Animals = [_, _, _, _, _],

```

```

Drinks = [_ , _ , milk, _ , _],           % const 8
Cigarettes = [_ , _ , _ , _ , _],
testp(england, red, Countries, Colours), % const 1
testp(japan, craven, Countries, Cigarettes), % const 13
beside(norway, blue, Countries, Colours), % const 14
on_right(white, green, Colours, Colours), % const 5
testp_seq(yellow, kool, Colours, Cigarettes), % const 7
testp_seq(spain, dog, Countries, Animals), % const 2
testp_seq(old_gold, snail, Cigarettes, Animals), % const 6
testp_seq(gitane, wine, Cigarettes, Drinks), % const 12
testp_seq(ukr, tea, Countries, Drinks), % const 4
testp_seq(green, coffe, Colours, Drinks), % const 3
beside_seq(chesterfield, fox, Cigarettes, Animals), % const 10
beside_seq(kool, horse, Cigarettes, Animals), % const 11
memb(zebra, Animals), % query 1
memb(water, Drinks). % query 2

```

```

C:\Windows\system32\cmd.exe - ..\config\x64-pc-windows\bin\xsb.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Jazz>cd "C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\examples"
C:\Users\Jazz\Dropbox\IT4BI\Advanced Databases\XSB\examples>..\config\x64-pc-windows\bin\xsb.exe
[xsb_configuration loaded]
[sysinitrc loaded]

XSB Version 3.4.0 (Soy mILK) of May 1, 2013
[x64-pc-windows; mode: debug; engine: slg-wam; scheduling: local]
[Patch date: 2013/05/02 17:42:32]

| ?- [houses].
[houses loaded]

yes
| ?- demo.
[config([norway,ukr,england,spain,japan],[yellow,blue,red,white,green],[fox,horse,snail,dog,zebra],[water,tea,milk,wine,coffe],[kool,chesterfield,old_gold,gitane,craven]])]

yes
| ?-

```

From the results it is clear that the Japanese breeds the zebra and the Norwegian drinks water.

Conclusion

Deductive databases are an interesting alternative to relational databases for data storage and processing. Their structure and mode of data representation make them inherently better at expressing certain features (e.g. transitive closures) and calculating things like generalized aggregates.

There are a few challenges with deductive databases though. Most implementations are still experimental and not widely used enough to have a wide repository of support resources. Secondly setting up and managing a deductive database is generally a very technical task. For most deductive databases, some knowledge of a programming language is required.

In conclusion, for everyday relational database tasks, a deductive database might not be worth the effort. However for more specialized applications such as knowledge/expert systems and artificial intelligence systems, deductive databases make an excellent backbone on which to build.

Sources

Benedikt M, Snellart P, (2011), "Databases". In Blum, Edward K.; Aho, Alfred V. *Computer Science. The Hardware, Software and Heart of It*. pp. 169–229

Grant, J & Minker, J, (1989), "Deductive Database Theories", *The Knowledge Engineering Review*, Vol 4:4, pp 267-304

Lofi C, "Knowledge-Based Systems and Deductive Databases", Institut für Informationssysteme, Technische Universität Braunschweig, http://www.ifis.cs.tu-bs.de/webfm_send/1060

Tsukamoto Y et al, (1997), "Application of a deductive database system to search for topological and similar three-dimensional structures in protein", *CABIOS*, (pp183 -190)

Zaniolo, C, (1990), "Deductive Databases-Theory Meets Practice", *Lecture Notes in Computer Science Volume 416*, pp 1-15

Zaniolo C et al, (1997), "Advanced Database Systems", *Morgan Kaufmann Publishers Inc.*, California, 1997