

Université Libre de Bruxelles

INFO-H-415: Advanced Databases

Project

Deductive Databases and XSB

Authors:

Gonçalo MOREIRA, 479038

Kaoutar CHENNAF, 474861

Supervisor:

Prof. Esteban Zimanyi

December 17, 2018



Contents

Abstract	2
Introduction	2
Deductive Databases	3
Overview	3
Architecture	3
Applications	4
Datalog	5
Deductive databases vs. relational databases	5
Advantages and disadvantages	6
Deductive Databases Systems	12
XSB	9
About XBS	9
Specifications	9
Latest Release	9
Java and Graphical Interfaces	10
Integrated Development Environment	10
Major features	10
Case study - “Price Point”	11
Deductive Database Application - XSB	12
Our use-case	12
How to run a program in XSB	14
Another implementation	15
Technical information	15
Conclusion	16
Bibliography	17

Abstract

1. Introduction

The predominant systems for the majority of nowadays' database applications are based on the following three data models: relational, hierarchical, and network model. These are what we refer to as traditional or conventional data models in contrast with the new generation of databases that include semantics, real-time, object-oriented, spatial... All these models have proved to be very efficient in many fields of application. However, the problem that rises is the one of interpreting and understanding large amounts of information as soon as it is stored, especially when it belongs to domains that are complex like mineral exploration or financial analysis. They also all share a common feature which is the simplistic model of the data structure required to represent the stored information. Hence, the effectiveness of these models remains limited in areas such as expert systems or knowledge discovery and extraction where information does not have this simplicity, and where there is a huge need for consistency to be maintained. An example of such complex data structures are the ones used in AI fields such as computer-aided manufacturing and design. Deductive databases were born from the need to combine complex information structures with the capabilities of logic programming in order to draw appropriate conclusions from the stored data by means of inference.

A deductive DBMS is a system that includes the possibility of defining rules that can infer -or infer information from- facts stored in a database. In a deductive database management system, a declarative language is used to specify rules. An inference mechanism (or deduction mechanism) can then infer new facts from the database by interpreting the rules. The model used for deductive databases is similar to the relational data model and more particularly to the formalism of relational calculus. A subset of Prolog called Datalog is used to declaratively define rules. An interesting application example for deductive databases is where the data can be interpreted under different visions.

In this report, we cover the concepts of deductive databases, while shedding some light on a DDBMS that is called XSB. Chapter 2 gives an overview of deductive databases, their architecture, applications, advantages and disadvantages, and an introduction to Datalog, the language used in most DDBMS. Chapter 3 specifies the most famous DDBMS while specifying the differences between them. Chapter 4 gives a detailed description of XSB, a commonly used DDBMS. And finally Chapter 5 describes our use-case to study the power and efficiency of such systems compared to others.

2. Deductive Databases

2.1. Overview

A deductive database is an advanced database system that can make deductions (i.e., conclude additional facts) based on rules and facts stored in the database.

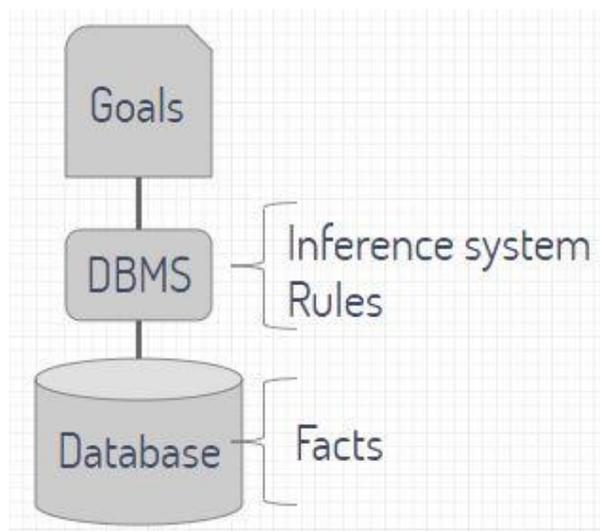
It uses an inference system that, by evaluating rules against facts, derives new facts which in turn can be used to answer queries.

Inference represents the derivation of new knowledge from existing knowledge and axioms (i.e., rules of derivation) within a single step, and can be one of many kinds, such as, induction, deduction and abduction. Inferences are formed from reasons and are basically conclusions drawn from propositions or assumptions that are supposed to be true.

Deductive databases are the combination of logic programming and database systems. In most common databases the goal is to store information and query over the data but in deductive databases, in addition to that, it is possible to make conclusions or retrieve other facts, i.e. facts that are implicitly true but are not explicitly represented in the database.

2.2. Architecture

A deductive database system consists of the following components:



- Basic **Facts** (Database);
- Non-procedural description of a system functionality (**Rules** of Inference);
- **Goals**.

Figure 1 - Deductive database system architecture

The information in a deductive database is specified through **predicates**, **facts**, **rules** and **goals**.

Example 1: Patient(Name, Ethnicity, Blood, Age, Disease)

Example 2: Patient('john', 'asian', 'a', 21, 'asthma')

Example 3: OlderPatients(Name) \leftarrow Patient(Name, Ethnicity, Blood, Age, Disease),
Age>40

The example 1 represents a **predicate**. It specifies an entity and its attributes. In a relational database, the same structure can be named as the schema of the entity. The number of arguments/attributes the predicate takes is referred to as the **arity** (for the example 1, the arity is 5).

The example 2 describes a **fact** - a predicate in which all the attributes have actual values. A fact specifies an instance of an entity and the values it contains. When the arguments are constant, the predicate states that a fact is true. When it has constants and variables, the predicate is considered as a **query** or as part of a rule. All predicate names and variables start with an upper-case letter. The constant values of character type must be identified with a lowercase letter. In relational databases we can represent a fact as a tuple in a table. Therefore we can think of a Patient table as being made up of many facts.

A **rule** (example 3) is the main construct of Datalog programs. It is made up of a head and a body separated by a symbol (\leftarrow). The head of the rule corresponds to a view in relational database. The body is made up of **goals**. 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 OlderPatients can be defined by every tuple in the Patients “table” that have Age greater than 40.

2.3. Applications

In recent years, deductive databases have found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

There are many areas of application for deductive database technology such as Artificial intelligence, Expert systems, Biology (Mining Industry), Planning systems, etc.

- **Decision support systems**

Nowadays, in many organisations, the exploitation of their resources requires not only a lot of information about the current and future status of the resources themselves, but also a way of reasoning effectively about plans for the future. Deductive database technology can help solve this problem.

- **Expert systems**

In areas such as Computing, the specific analysis of data made by deductive database technologies allows the performance of these data to be more efficiently, accurate and with a lower chance of error than by other methods.

For example, medical analysis can generate large amounts of data, and an error can have disastrous consequences. A tool to carefully monitor a patient's condition or to retrieve relevant cases during diagnosis reduces the risk of error in such circumstances.

Other examples: Cyc System, PACADE and XSB.

2.4. Datalog

The programming language for deductive databases is Datalog. It is a declarative logic-based language, syntactically very similar to Prolog. Typically, it is used to specify facts, rules and queries in deductive databases.

The most important differences between deductive databases and logic programming:

- **Order sensitivity and procedurality:** in Prolog, program execution depends on the order of rules and on the order of parts of rules; these properties are used by programmers to build efficient programs. In database languages (like SQL or Datalog), however, program execution is independent of the order of rules and facts.
- **Special predicates:** in Prolog, programmers can directly influence the procedural evaluation of the program with special predicates such as the cut, this has no correspondence in deductive databases.
- **Function symbols:** logic programming languages allow function symbols to build up complex symbols. This is not allowed in deductive databases.
- **Tuple-oriented processing:** Deductive databases use set-oriented processing while logic programming languages concentrate on one tuple at a time.

2.5. Deductive databases vs. relational databases

Deductive databases offer an interesting alternative to relational databases in areas such as recursion and information retrieval. Deductive databases are an extension of relational databases which support more complex data modeling.

In Table 1, we can compare the expressiveness between SQL and Datalog languages.

	Semantics	Recursion	OR
SQL	CWA*	No	Yes
Datalog	CWA*	Yes	No

Table 1: Comparison of expressiveness between SQL and Datalog

*The Closed-World Assumption (CWA) is the presumption that a statement that is true is also known to be true. Therefore, conversely, what is not currently known to be true, is false.

Model	Data Element Organization	Relationship Organization	Identity	Access Language
Hierarchical	Files, Records	Logical Proximity in a Linearized tree	Record based	Procedural
Network	Files, Records	Intersecting Networks	Record based	Procedural
Relational	Tables	Identifiers of rows in one table are embedded as attribute values in another table	Value based	Non-procedural
Object-Oriented	Objects	Logical Containment - Related objects are found within a given object by recursively examining attributes of an object that are themselves objects	Record based	Procedural
Object-Relational	Object- infrastructure to the database system itself—user-defined data types, functions, and rules	Relational extenders that support specialized applications such as image retrieval, advanced text searching, and geographic applications.	Value based	Non-procedural
Deductive	Facts, Rules	Inference rules that permit related facts to be generated on demand.	Value based	Non-procedural

Figure 2 - DB Models comparison

2.6. Advantages and disadvantages

Advantages

- The major advantage is the ability to do recursion and the ease with which such recursion can be defined. This can be seen as an additional expressive power to relational databases.
- Syntax and structure more intuitive and simpler (to construct), easier to maintain than SQL equivalent.
- Draw conclusions while handling large amounts of data.

Disadvantages

- Lack of widespread development and support.
- Small community.
- Lack of documentation.
- Inability to deal with unstructured data / difficult to store data (it is necessary to have a clear relationship between the information).
- Less expressive than logic programming systems.

3. Deductive Databases Systems

Deductive databases have seen the creation of several prototypes that specify the way inference is being done. Many of these models have been implemented in projects ranging from military, to biology, or finance...

In this section, we discuss few examples of such systems and their uses. These systems have received significant attention and development effort by the deductive database community.

- **RDL/C**

RDL/C was developed in order to integrate a rule-based language in addition to the programming language C. It was derived from RDL1, hence it supports rules and abstract data types and therefore allows the user to program at a higher level than it would be possible by combining SQL and C. It also manages temporary relations instead of leaving this task to the user.

- **MegaLog**

MegaLog has been developed at the ECRC (European Computer Research Centre). This is a system that was designed especially to support manipulating large amounts of data while providing some Prolog features. One of its major contributions is the support of a multi-dimensional grid called “Balanced And Nested Grid file” (BANG). Another important one is the support of garbage collection and excellent facilities for dictionary management.

- **EKS**

Like most deductive database systems, the goal of this system is to demonstrate the validity of deductive database technology for real-world applications. The language of EKS is Datalog. Therefore, it does not support functions. One of its main advantages is the way the system handles negation in a top-down setting. This makes negation more simple to implement than in bottom-up methods by using the magic set transformation.

- **LDL**

This system was developed at the Micro Computer Corporation. Its main feature is supporting sets in the language. It was built based on the bottom-up model, and uses several optimization techniques like magic sets. This system is a single user system, and all relations are memory-resident. The deductive part of this system is memory-resident.

- **LOLA**

LOLA was developed at the Technical University of Munich. The system compiles Horn clauses, which may or may not contain lists, into a Relational Lisp program with embedded SQL statements. The system does optimizations to minimise the calls to the underlying SQL database system. The system supports multiple users and transactions. Its deductive part is memory-resident. Therefore, it is not scalable for large databases.

- **CORAL**

This system was developed at the University of Wisconsin at Madison. It uses a bottom-up evaluation and a variety of optimization techniques that should be specified and specialized by the programmer. It's important features include support for non-ground terms. The system is also a single-user system and again, memory-resident.

- **Glue-Nail**

Glue-Nail was developed at Stanford University. It's most important feature is its provision of two languages: one for declarative statements that are based on Horn clauses, and another one for I/O, control constructs and updates, that is procedural. This system supports a form of higher-order syntax for managing relation names. It is yet another single-user and memory-resident system.

- **Aditi**

Aditi was developed at the University of Melbourne. It uses a bottom-up approach by relying on relational technology. Its main contribution is the fact that permanent and temporary relations can be disk-resident. Therefore, it is one of the only systems that are scalable to large databases. It also supports function symbols, negation, and aggregates. Its architecture is based on the client-server model, and hence supports parallel query processing and multi-users.

Name	Evaluation	Syntactic Restrictions	Negation	Data Requirements
Aditi	Magic Sets	Datalog	Mod. Strat	Disk-Resident
CORAL	Magic Templates	First-order	Mod. Strat.	Main-Memory
LDL	Magic Sets	First-order	Mod. Strat.	Main-Memory
Glue-Nail	Magic Rewriting	F-O with Restricted HiLog extensions	Well-founded	Main-Memory
Sylog	Backchain Inferencing	Datalog	Stratified	Disk-Resident
XSB	SLG	HiLog with F-O optimisations	Well-founded	Main-memory

Figure 3 - Comparative table of DDBMS

Before deductive database technology is generally accepted in the global database community, these systems will have to standardise database facilities for crash recovery, transaction processing, integrity constraints, multi-users access, and triggers. Unfortunately, several of these prototype systems do not have such capabilities. It is also promising to witness the rise of some commercial deductive database systems, which will include the standard database features.

4. XSB

XSB is a Logic Programming and Deductive Database system for Unix and Windows. It is being developed at a number of institutions, including the Computer Science Department of Stony Brook University, Universidade Nova de Lisboa, XSB Inc., and Coherent Knowledge Systems, Inc.

4.1. About XBS

XSB has similar goals to the CORAL system in supporting non-ground terms and negation. However, its main distinction is the model of computation used in XSB, which is a top-down one with memoing. Like most other systems, it is a memory-based, single user system.

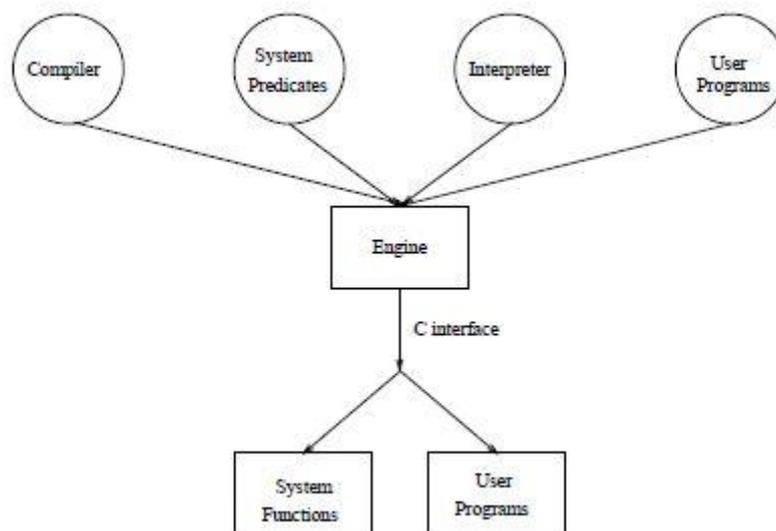


Figure 4 - XSB Architecture

4.2. Specifications

- Latest Release

The latest stable version of XSB 3.8 (available since October 29, 2017).

- Documentation:

The XSB download includes the manuals in pdf format, but they may also be downloaded separately: Volume 1 and Volume 2

The xsbdoc Documentation Generator can be used for literate XSB programming (PDF).

- Java and Graphical Interfaces

InterProlog distributed by InterProlog Consulting, provides a means of interfacing XSB with Java programming.

- Connecting to the source code

The ``.P'` suffix indicates to XSB that this is a file containing source code.

`[user].` opens editor

`end_of_file.` closes editor

`%` comments

`halt.` exits XSB

- Integrated Development Environment

Prolog Studio is a new IDE specifically written for Prolog programmers, and distributed by InterProlog Consulting. It arose out of a Coherent Knowledge Systems software project, so it has many XSB-specific features.

4.3. Major features

Tabled Resolution

Tabled resolution is mainly useful for recursive queries computation, thus, allowing programs to terminate correctly in most cases when Prolog cannot. Users interested in Parsing, Program Analysis, Model-checking, Data Mining, Diagnosis and Temporal Reasoning will benefit from XSB.

Support for Multi-Threading

XSB has started supporting a configuration allowing multiple threads of computation within a single process under the Posix model.

Indexing and Dynamic Code

XSB contains a variety of features to support memory-based & data-oriented applications. By using these features, knowledge bases with (let's say) millions of clauses can be quickly and efficiently loaded and efficiently indexed.

Interfaces

XSB supports interfaces. All source code is available so that the interfaces can be tuned as needed or ported to other Prologs.

HiLog Compilation

HiLog supports a type of higher-order programming in which predicate symbols can be variable or structured. This allows unification to be performed on the predicate symbols themselves in addition to the arguments of the predicates.

Packages

- Flora: a highly sophisticated and efficient implementation of F-logic
- XJ: a system that allows XSB programs to create graphical user interfaces based on Java's Swing package.
- PITA: is a library that supports probabilistics and fuzzy reasoning with tabling.
- XMC: a temporal-logic model checker for verifying properties of concurrent programs. and containing a graphical user interface.
- xsbdoc: a mechanism for generating manuals for XSB libraries and applications using principles of literate programming. Manuals can be generated in a variety of formats, including html, pdf, ps, and even ASCII text.

XSB was used to construct large-scale systems for institutions such as the U.S. Customs Service, the U.S. Defense Logistics Agency, the National Security Agency, Medicine Rules, Inc, MdLogix Inc., and many others.

4.4. Case study - “Price Point”

In the paper “Ensuring Good Value: New Methods for Price Evaluation”, we have an example of a real use case in which the XSB implementation solved the problems and difficulties of a company. The case study describes the problem and how it was solved.

In general, traditional methods of evaluating fair prices for catalogued products placed on government contract are not scalable. The volume of examinable Stock Keeping Units (SKU) grows as suppliers add new products. This evaluation of prices can be done with data mining solutions.

Problem: validation of prices for all 10,000 products consume an unreasonable amount of time - when done manually; in fact, prices may change by the time this process is completed. Manufacturer name/part number representations differ: DEWDW235G and BLADW235G are different ways to say “DW235G”. Another layer of complexity is added when vendors start using their internal catalog numbers in the fields designated for a Manufacturer part number.

Solution: an automated system that can manage the process, rapidly examine a large number of products and identify suspicious prices that require further inspection by the Contract Officer.

The companies ended up licensing “Price Point”, a Web based automated price evaluation tool that collects data on tens of millions of products and their pricing and quickly identifies potentially problematic supplier pricing. This solution uses data mining technology coupled with a large knowledge base of company brands and abbreviations to overcome data quality issues that complicate item comparison and evaluation. Price Point’s analysis of the first set of catalogs resulted in more than a 2000% productivity increase (which is not surprising for process automation), the coverage of products evaluated increased eight-fold, with a ten-fold increase in the number of over-priced products identified. The result has been a tremendous improvement in efficiency and a reduction in total acquisition costs.

5. Deductive Database Application - XSB

For the implementation, we build a program in datalog based on the Einstein's puzzle - projectXSB.P. The goal of the program is to resolve the puzzle based on the rules and facts that we established.

5.1. Our use-case

In an hospital, there are five special patients. Each one has a specific disease. That's why they are distributed in separate rooms because they all have different backgrounds. Each has a specific ethnicity, blood type and age.

We know that:

- 1- Tony is African.
- 2- Paul has blood type O-.
- 3- The Aussie patient is 50 years old.
- 4- Amy is 42 years old.
- 5- The Aussie's room is on the right of the European one.
- 6- The Tuberculous patient has blood type AB.
- 7- The Asian patient is Asthmatic.
- 8- The 19 years old patient's room is in the middle.
- 9- John is in the first room on the left.
- 10- The Amnesic patient's room is beside of the patient with blood type A.
- 11- The Asthmatic patient's room is beside of the patient with blood type B.
- 12- The Obese patient is 31 years old.
- 13- Mary has diabetes.
- 14- John's room is beside the latino patient.

Questions:

Who has the blood type O?

Who is 27?

In Tables 2 we can see the full dataset of the dimension - "Patient".

Name	Ethnicity	Blood	Age	Disease
John	Asian	A	27	Asthma
Amy	Latino	B	42	Anemia
Tony	Africa	AB	19	Tuberculosis
Paul	European	O-	31	Overweight
Mary	Aussie	O	50	Diabetes

Table 2 - Patient's dataset represented in a table

The same information can also be represented in Datalog as follows:

```
patients.P x
1 patient(john, asian, a, 21, asthma).
2 patient(amy, latino, b, 42, anemia).
3 patient(tony, african, ab, 19, tuberculosis).
4 patient(paul, european, o-, 31, obesity).
5 patient(mary, aussie, o, 50, diabetes).
```

Figure 5 - File patients.P

In SQL, we can reproduce the same example by creating a table and inserting the values.

```
USE [projectXSB]
CREATE TABLE [Patient] (
    [Name] nvarchar(50) NOT NULL,
    [Ethnicity] char(50) NOT NULL,
    [Blood] nvarchar(10) NOT NULL,
    [Age] int NOT NULL,
    [Disease] nvarchar(50) NOT NULL
)
```

Figure 6 - CREATE TABLE Patient

```
USE [projectXSB]

INSERT Patient(Name, Ethnicity, Blood, Age, Disease)
VALUES('John', 'Asian', 'A', 21, 'Asthma')

INSERT Patient(Name, Ethnicity, Blood, Age, Disease)
VALUES('Amy', 'Latino', 'B', 42, 'Anemia')

INSERT Patient(Name, Ethnicity, Blood, Age, Disease)
VALUES('Tony', 'African', 'AB', 19, 'Tuberculosis')

INSERT Patient(Name, Ethnicity, Blood, Age, Disease)
VALUES('Paul', 'European', 'O-', 31, 'Obesity')

INSERT Patient(Name, Ethnicity, Blood, Age, Disease)
VALUES('Mary', 'Aussie', 'O', 50, 'Diabetes')
```

Figure 7 - INSERT Patient

The result should be a table “Patient” as seen in Figure 8.

	Name	Ethnicity	Blood	Age	Disease
1	John	Asian	A	21	Asthma
2	Amy	Latino	B	42	Anemia
3	Tony	African	AB	19	Tuberculosis
4	Paul	European	O-	31	Obesity
5	Mary	Aussie	O	50	Diabetes

Figure 8 - Results

5.2. How to run a program in XSB

For this tutorial, we use the file - projectXSB.P.

1. Run XSB (in command line or using the executable file)
2. Compile/load the file
consult('C:/Users/GTM/BDMA/BRU_ULB/Advanced Databases/Project/projectXSB.P').
3. Run the program
demo.

```

XSB-Win64
[xsb_configuration loaded]
[sysinitrc loaded]
[xsbbbrat loaded]

XSB Version 3.8.0 (Three-Buck Chuck) of October 28, 2017
[x86-pc-windows; mode: optimal; engine: slg-wam; scheduling: local]
[Build date: 2017-10-31]

| ?- consult('C:/Users/GTM/BDMA/BRU_ULB/Advanced Databases/Project/projectXSB.P').
[Compiling C:\Users\GTM\BDMA\BRU_ULB\Advanced Databases\Project\projectXSB]
% Specialising partially instantiated calls to rooms_iter/1
[projectXSB compiled, cpu time used: 0.4060 seconds]
[projectXSB loaded]

yes
| ?- demo.
[config([john, amy, tony, paul, mary], [asian, latino, african, european, aussie], [a, b, ab, o_, o], [27, 42, 19, 31, 50],
[asthma, anemia, tuberculosis, obesity, diabetes])]

yes
| ?- _

```

Figure 9 - Run projectXSB.P

After running our program (demo.), we get the results as shown in Figure 6.

The results are expressed as 5 sets representing each patient's attributes (Name, Ethnicity, Blood, Age, Disease).

If we analyse the output, we can see that we have the answers to our initial questions:

Who has the blood type O?

- Mary.

Who is 27?

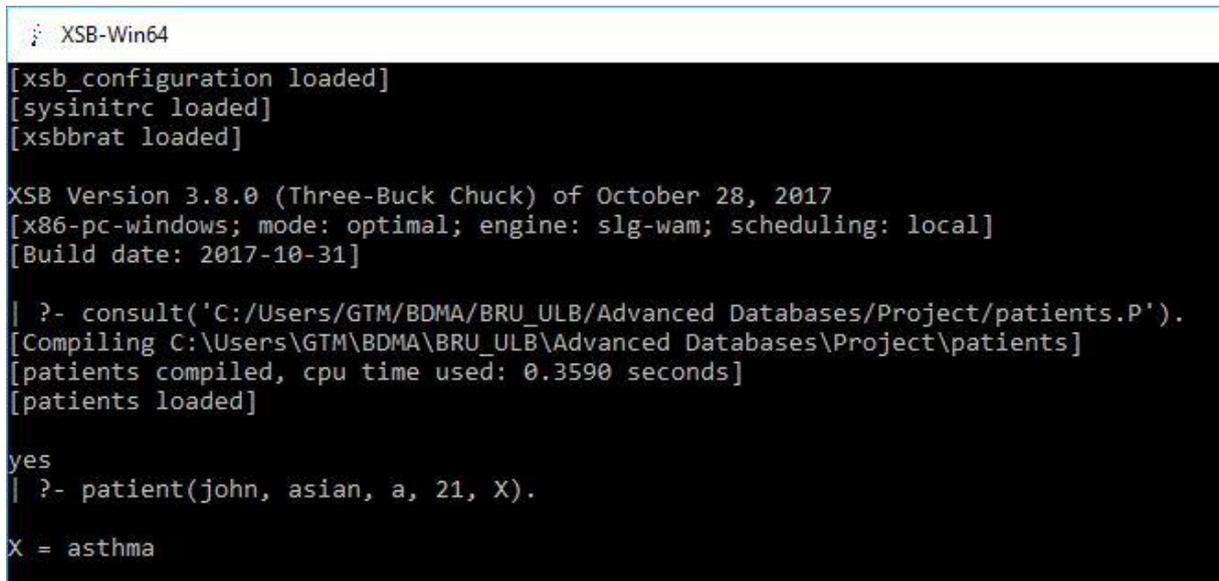
- John.

5.3. Another implementation

Using the same example of Patients, we created the file patients.P (Figure 2) and tried to find a specific value “X”.

What is the disease of John?

- Asthma.



```

XSB-Win64
[xsb_configuration loaded]
[sysinitrc loaded]
[xsabbrat loaded]

XSB Version 3.8.0 (Three-Buck Chuck) of October 28, 2017
[x86-pc-windows; mode: optimal; engine: slg-wam; scheduling: local]
[Build date: 2017-10-31]

| ?- consult('C:/Users/GTM/BDMA/BRU_ULB/Advanced Databases/Project/patients.P').
[Compiling C:\Users\GTM\BDMA\BRU_ULB\Advanced Databases\Project\patients]
[patients compiled, cpu time used: 0.3590 seconds]
[patients loaded]

yes
| ?- patient(john, asian, a, 21, X).

X = asthma
```

Figure 10 - Find value “X”

XSB, as a deductive database system, is able to not only store the data but it also can deduct facts from established rules.

5.4. Technical information

- CPU: Intel i5-6200 CPU @ 2,30 GHz
- RAM: 8,00 GB
- OS; Windows 10, 64-bit
- XSE 3.8 - the last stable version (October 29, 2017) for Windows 64-bit;
- SQL Server 2014.

6. Conclusion

Deductive databases are really powerful and can be useful for rule-based applications. They can draw conclusions while handling with large amounts of data, and they have major features like recursion and negation. This database system technology was created during the 1970's and, since then, it has been applied in many different areas such as biology, data mining and expert systems. From our research, we conclude that the use of this technology system in real-world applications is progressing and there are possibilities to improve, for example by combining the query language capability of deductive databases with features from object-oriented systems.

This technology has reached now a level of maturity that allows the commercial development of deductive database systems. The systems shown above demonstrate the possibility of building real world-applications and the potential efficient performance to perform as efficiently as conventional models that has seen extensive efforts to be developed.

7. Bibliography

Sagonas, K., Swift T., Warren D. S. (1994). *XSB as an Efficient Deductive Database Engine*.

Ramakrishnan R., Ullman J. D. (1993). *A Survey of Research on Deductive Database Systems*.

(2011). *Deductive Data Model*.

https://datubaze.files.wordpress.com/2011/10/deductivedb_model.pdf

XSB, Inc., *Ensuring Good Value - New Methods for Price Evaluation*.

<https://www.xsb.com/sites/default/files/papers/Methods%20for%20Price%20Evaluation%20that%20Ensure%20Good%20Value.pdf>

Rosati, R. (2015). *SQL, DLs, Datalog, and ASP: comparison*.

<http://www.dis.uniroma1.it/~rosati/krst-1516/comparison.pdf>

Mohania, M. K., Sarda N. L. (1993). *An Architecture for a Distributed Deductive Database System*. <http://dspace.library.iitb.ac.in/xmlui/bitstream/handle/10054/1289/7689.pdf>

Popoola, J., Wasii Y. (2017). *Deductive databases* <https://pt.slideshare.net/pbimbs/deductive-databases-78658392>

(2015). *Deductive databases*. <https://www.youtube.com/watch?v=VH2g5RAz2bE>

(2018) *What is Deductive Database?*. <https://www.youtube.com/watch?v=LZdLLFyh5g>

Warren, S. D. (1999). *Executing Programs in XSB*

<https://www3.cs.stonybrook.edu/~warren/xsbbook/node8.html>

Matthew, *XSB Tutorial: Basic Command Line Examples*

https://www.isi.edu/~argos/matthew/xsb_command_line_tutorial.html

Genesereth, M., *A Brief Introduction to Deductive Databases*

<http://ggp.stanford.edu/notes/ddb.html>

<https://www.xsb.com/>

<http://xsb.sourceforge.net/index.html>

<https://sourceforge.net/projects/xsb/>