Datalog: Bases de datos Deductivas

Norma Verónica Ramírez Pérez
Instituto Tecnológico de Celaya
norma.ramirez@itcelaya.edu.mx

Martin Laguna Estrada
Instituto Tecnológico de Celaya
martin.laguna@itcelaya.edu.mx

Resumen

Este artículo muestra un breve estudio sobre Datalog, el cual es considerado como una extensión de Prolog que es uno de los sofware más utilizados en la inteligencia artificial. Sistemas importantes como SWI-Prolog[22], Ciao Prolog[4], Sictus-Prolog[20], han sido compiladores utilizados para hacer uso de la funcionalidad del lenguaje lógico para bases de datos deductivas y han logrado la implementación de consultas recursivas sobre las bases de datos relacionales. En este estudio también se presentan conceptos básicos de Datalog, así como algunos sistemas que se han desarrollado para trabajar con este lenguaje.

Palabra(s) Clave(s): Programación lógica, Prolog, Datalog.

1. Introducción

Las bases de datos deductivas incrementan la capacidad expresiva de las bases de datos relacionales, incorporando reglas que permiten definir conocimiento en forma implícita, es decir, derivar información a partir de información almacenada explícitamente.

En el esquema de una base de datos deductiva se pueden distinguir dos tipos de relaciones: relaciones básicas cuyas tuplas se almacenan explícitamente y relaciones derivadas o vistas que se definen por reglas deductivas a partir de relaciones básicas y de otras relaciones derivadas. Un estado de base de datos está formado por un conjunto

de hechos, tuplas de las relaciones básicas y por un conjunto de reglas deductivas que definen las relaciones derivadas.

Las investigaciones sobre la relación entre la teoría de las bases de datos y la lógica se remontan a finales de la década de los 70's, pero el estímulo principal para el aumento de interés por el tema aparece en la publicación de Reiter en 1984 [14], que caracterizaba la percepción tradicional de los sistemas de bases de datos según la Teoría de Modelos, que en términos no muy formales expresa:

"Una base de datos contempla un conjunto de relaciones explícitas, cada una de las cuales contiene un conjunto de tuplas explícitas, y la ejecución de una consulta puede considerarse como la evaluación de una fórmula especificada sobre esas relaciones y tuplas explícitas. También argumentó que existía la posibilidad de una perspectiva según la Teoría de Demostraciones, preferible en algunos aspectos".

Desde esta perspectiva, la base de datos se contempla como un conjunto de axiomas y la ejecución de una consulta se considera como la demostración de que una fórmula especificada es una consecuencia lógica de esos axiomas. A la fecha se siguen explorando nuevas líneas de cooperación entre la inteligencia artificial y las bases de datos. Se puede considerar a la Inteligencia Artificial [15] como el área de la computación en donde una computadora puede simular formas del pensamiento humano obteniendo nueva información de la ya existente y el almacenar, recuperar y actualizar la información dentro de una base de datos de una forma eficiente. Al combinar las bases de datos y la inteligencia artificial, se obtienen los objetivos principales de una base de datos deductiva:

- Descubrir formalismos para la representación de datos.
- Capturar la información del mundo real en un modelo apropiado.
- Enfatizar el manejo de símbolos en la representación del conocimiento.
- Lograr la eficiencia del sistema, ya que es utilizado como fuente de conocimiento.
- Elaborar el diseño que permita el mantenimiento en forma fácil.

El modelo empleado en las bases de datos deductivas está relacionado con el campo de la programación lógica y el lenguaje Prolog, el cual derivó en un subconjunto de Prolog llamado Datalog [7].

2. Datalog

Tomando como referencia a los autores [5,7], Datalog es un lenguaje lógico desarrollado para el modelo relacional. Datalog sin recursión tiene el mismo poder expresivo que el álgebra relacional, sin embargo, a partir de 1999 SQL ha usado una solución para la recursión en Datalog para el desarrollo de consultas recursivas. Es similar a Prolog en su sintaxis, pero su semántica operacional es diferente. Una regla o cláusula en Datalog tiene la forma:

$$cabeza \leftarrow cuerpo$$

Donde la cabeza es un átomo y cuerpo es una lista de átomos que puede ser vacía; en este caso se habla de un hecho. Los hechos se definen:

$$P(t_1, \ldots., t_n)$$

Donde P es un símbolo de predicado y t_i son variables constantes. No se admiten símbolos de función en t_i, a diferencia de Prolog.

2.1 Reglas lógicas

Una regla se escribe: $P \leftarrow Q_1, \ldots, Q_n$

Dónde: "Si $Q_1, Q_2, \ldots y\ Q_n$ son ciertos, entonces P es cierto". Si $n = 0$ "P es cierto", y se escribe: P.

Para definir el significado de las reglas, existen diferentes alternativas que se mencionan a continuación.

Interpretación de la teoría de pruebas. Es el conjunto de todos los hechos que se pueden probar a partir de las reglas del programa usándolas de todas las formas posibles.

Interpretación de la teoría de modelos. La interpretación de una colección de predicados asigna cierto o falso a cada posible instancia de los predicados, donde los argumentos se escogen de un conjunto infinito de constantes. La interpretación se representa habitualmente por el conjunto de instancias verdaderas.

Definición computacional. La última forma de definir el significado de las reglas lógicas es proporcionar un algoritmo para ejecutarlas para determinar si un hecho es cierto o falso. Prolog define su semántica de esta forma, solo que tiene un inconveniente, hay hechos que no se pueden probar de esta forma (ramas infinitas).

Datalog es una versión de Prolog, que fue adecuada para dar respuesta a las bases de datos, pero se diferencia en:

- Datalog no admite símbolos de función en los argumentos.
- El significado de los programas Datalog sigue el punto de vista de la teoría de modelos, en cambio, Prolog se basa en un significado computacional que se desvía de los significados de la teoría de modelos y la teoría de pruebas.

2.2 El modelo de datos de Datalog es similar al relacional

Una relación se representa por un predicado.

Sin embargo, sus argumentos siguen una notación posicional no explícita como el modelo relacional.

Ejemplo 1: Una instancia r de la relación r(A,B) en el modelo relacional definida por r, sería de la siguiente manera:

r

A	B
1	2
3	4

Sin embargo, representada en Datalog por los hechos: r(1,2) y r(3,4).

En Datalog, el primer argumento de R corresponde con el atributo A y el segundo a B.

El significado de la relación en ambos modelos de datos es el mismo, el conjunto de tuplas {(1,2),(3,4)}. Es decir, hay una relación de tipo R entre 1 y 2 y entre 3 y 4.

2.3 Predicados intencionales y extensionales

Esta diferencia entre Datalog y las bases de datos relacionales. Un predicado cuya relación se almacena explícitamente en la base de datos, se denomina extensional.

Un predicado que se define en términos de reglas se denomina intencional.

En el modelo relacional, todas las relaciones se definen extensionalmente. Las vistas permiten una forma limitada de definición intencional, más limitada que en Datalog.

Ejemplo 2: En una base de datos genealógica:

madre,padre, mujer, nombre: extensional
madre(ana,pedro).
madre(ana,juan).
…
padre(jose,julia)
padre(luis,jose).
…..

progenitor, antepasado: intencional
progenitor(X,Y):-madre(X,Y).
progenitor(X,Y):-madre(X,Y).
antepasado(X,Y):-progenitor(X,Y).
antepasado(X,Y):-progenitor(X,Y), antepasado(Z,Y).

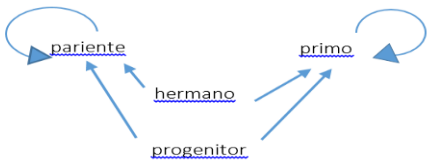
2.4 Recursividad y grafos de dependencia

Las reglas de Datalog son en general recursivas. Para determinar si un determinado predicado es recursivo, se construye un grafo de dependencias, sus nodos son predicados:

- Hay un arco de p a q si hay una regla con un subjetivo, cuyo predicado sea q.
- Un programa lógico es recursivo si hay uno o más ciclos en el grafo.
- Un predicado es recursivo si forma parte de un ciclo.
- Nótese que p y q pueden ser el mismo predicado (de hecho, es el caso más habitual).

Ejemplo 3:

Hermano(X,Y):-progenitor(Z,X), progenitor(Z,Y), X≠Y
primo(X,Y):-progenitor(PX,X), progenitor(PY,Y), hermano(PX,X), hermano(PY,Y).
primo(X,Y):-progenitor(PX,X), progenitor(PY,Y), primo(PX,PY).
pariente(X,Y):- Hermano(X,Y).
pariente(X,Y):- pariente(X,Z), progenitor(Z,Y).
pariente(X,Y):- pariente(Z,Y), progenitor(Z,X).



2.5 Reglas seguras

Los predicados predefinidos representan en general relaciones infinitas. Considérese los hechos de forma infija, es el cierre transitivo de:

$$\{\dots, -2 < -1, -1 < 0{,}0 < 1{,}1 < 2{,}2 < 3, \dots.\}$$

Para evitar conjuntos infinitos, se debe asegurar que el rango de las variables implicadas en estos predicados esté acotado. Se debe asegurar en general que las reglas Datalog se activen (como operaciones sobre relaciones en una base de datos relacional). Además de los predicados predefinidos, aparecen problemas cuando hay variables que solo aparecen en la cabeza de la cláusula:

Ejemplo 4: Todo el mundo come comida:
come(X,Y):-comida(Y).

Esta regla define un conjunto infinito de pares como(X,Y), aún y cuando la relación comida sea finita.
Condiciones para asegurar reglas seguras:

- Las relaciones del cuerpo deben ser finitas.
- Los valores de las variables que hacen cierta la regla deben prevenir de estas relaciones finitas.

Reglas para la definición de variables limitadas:

- Cualquier variable que aparezca como argumento de un predicado normal (finito) del cuerpo está limitado.
- Cualquier variable que aparezca en un subjetivo X=constante o constante=X, está limitada.
- La variable X está limitada si aparece en un subjetivo X=Y o Y=X, si Y también está limitada.

Ejemplo 5:

come(X,Y):-comida(Y).
X no está limitada
mayor(X,Y):-X>Y.
ni X ni Y están limitadas (no forman parte de un predicado finito).
p(X,Y):-q(X,Z), W=a, y =W
X y Z están limitadas por la regla (1), W está limitada por la regla (2).
Y está limitada por la regla (3), dado que W está limitada.

2.6 Negación

En ocasiones es útil expresar una negación en el cuerpo de una regla. Sin embargo, tal regla así escrita no es una cláusula Horn.

El problema se traslada al cálculo del complemento: el complemento de una relación no es un operador bien definido (a diferencia del complemento de una condición simple). Para calcularlo es necesario conocer el dominio de las variables implicadas en el átomo negado para calcular el resultado por diferencia de conjuntos.

Aun si esto se puede hacer así, en general nos encontramos con una relación infinita que no es computable en Datalog.

Casos

Todas las variables de un átomo negado aparecen como argumentos de una relación finita.

Ejemplo 6:

- La regla se_lleva_bien(X,Y):-familiares(X,Y), ¬negocios_entre(X,Y), es equivalente a la expresión del algebra relacional S=F-N. Donde S representa a se_llevan_bien. F a familiares y N a negocios_entre. (Esta expresión se puede formular en Datalog).
- Alguna variable aparece solo como argumento de un átomo negado.

  soltero(X):-hombre(X), ¬casado(X,Y).

  No se puede calcular como en el caso anterior con el complemento porque obtendría todos los hombres X que no estén casados con una cierta Y.

El complemento de casado (X,Y), son todas las tuplas tales que X no está casado con Y, al reunirlo con hombre se obtienen todos los pares tales que X es hombre e Y no está casada con X, pero hay muchas Y que no están casadas con X, aunque con X, esté casado con alguien en concreto.

Para evitar estos problemas, se impide que aparezcan variables, sólo en los átomos negados y así, el ejemplo anterior se reescribe:

marido(X):-casado(X,Y).

soltero(X):-hombre(X), ¬marido(X)

- La condición (1) se puede relajar y permitir que la relación finita contenga más atributos que los del átomo negado, como en:

  puede_comprar(X,Y):-quiere(X,Y), ¬sin_blanca(X)

  ¬sin_blanca(X) es infinito en general, pero no impide calcular puede_comprar (X,Y) por que se pueden ir tomando las tuplas quiere(X,Y) y comprobar que cada una tiene componente X que es miembro de no_sin_blanca, o, lo que es lo mismo, que X no es miembro de sin_blanca(X).

  En álgebra relacional se puede expresar:

  puede_comprar(X,Y).

Por tanto, podemos escribir siempre en Datalog reglas que cumplan (1) a partir de la condición relajada (3) como en el ejemplo anterior.

2.7 Existencia de varios mínimos puntos fijos

Hay otro problema además de los considerados anteriormente al usar la negación. Es posible encontrar más de un mínimo punto fijo (es decir, no se sabe cuál es el significado del programa).

Ejemplo 7:

p(X):-r(X), ¬q(X).≡P=R-Q

q(X):-r(X), ¬p(X).≡Q=R-P

Si R={1}, S1 la solución P={} y Q={1} y S2 la solución P={1} y Q{}.

Las dos son soluciones a las ecuaciones algebraicas pero no se puede decir que S2<S1 ni S1<S2 y tampoco se puede encontrar un S menor que S1 o S2.

Para resolver este problema, solo se permite la negación estratificada que, aunque no garantiza un punto fijo, si permite una selección de uno de ellos de forma determinista, de manera que se acepte como el significado del programa.

Se dice que las reglas están estratificadas si para cada regla p:-…., ¬q,.. no hay un camino en el grafo de dependencias de p a q.

2.8 El álgebra relacional y Datalog

Sin considerar la negación, el álgebra relacional, y Datalog no son comparables porque en cada caso se pueden expresar relaciones que en el otro no. Con una versión limitada de la negación (negación estratificada), se puede demostrar que Datalog es más expresivo que el álgebra relacional.

2.9 Expresión en Datalog de las operaciones del algebra relacional

Para simplificar la demostración de que todas las operaciones del álgebra relacional se pueden representar en Datalog y, en particular las secciones complejas, se plantean los siguientes lemas.

Lema 1. Dado σ_F(E), con F conteniendo ¬, se pueden trasladar todas las negaciones dentro de los operadores $\wedge$ y $\vee$ con las leyes de Morgan:

$$\neg(\mathrm{F} \wedge \mathrm{G}) = (\neg\mathrm{F}) \vee (\neg\mathrm{G})$$

$$\neg(\mathrm{F} \wedge \mathrm{G}) = (\neg\mathrm{F}) \wedge (\neg\mathrm{G})$$

Aplicando reiteradamente estas leyes, se pueden alcanzar condiciones atómicas de la forma:

$$X\theta Y \text{ y } \neg X\theta Y \ ,$$

Donde X e Y son atributos y θ es una condición $=, \leq, \geq, \neq, <, >, \neg X\theta Y$, siempre se puede reescribir como $\mathrm{X}\theta\mathrm{Y}$,siendo θ la condición opuesta a θ.

Lema 2. Toda expresión del álgebra relacional produce secciones de la forma $X\theta Y$, con X e Y atributos o constantes y θ una condición simple.

Demostración: Se parte de una selección compleja sin negaciones y que contiene condiciones simples. Se demuestra por inducción y solo se da la idea.

Caso base (no hay operaciones lógicos): no hay que hacer nada.

Teorema: Toda la función expresable en el álgebra relacional es expresable como un programa Datalog no recursivo.

Demostración: Se parte de una expresión con i operadores, también por inducción caso base (i=0, no hay operadores, un único operando):

Si el operando es una relación R existente, es una relación extensional y no es necesario ninguna otra regla para expresarla.

Si el operando es un conjunto de tuplas, se crea una nueva relación P formada de hecho para expresar este conjunto.

2.10 Inducción

Hay cinco casos de operador raíz en la expresión: unión, diferencia, selección, proyección y producto (el resto de operadores, como la reunión, se pueden reescribir en términos de éstos):

Caso 1 E=E1∪ E2. Por la hipótesis de inducción se supone que hay predicados e_1 y e_2 definidos por reglas Datalog no recursivas, entonces E se define por:

$$E(X_1,\ldots,X_n):\text{-}e1(X_1,\ldots X_n).$$

$$E(X_1,\ldots,X_n):\text{-}e2(X_1,\ldots X_n).$$

Caso 2: E=E1-E2

$$E(X1,\ldots,Xn):\text{-}e1(X1,\ldots Xn),\ \neg(X1,\ldots Xn).$$

Caso 3: E=πi1,….ik(E_1)

$$E(X_{i1},\ldots X_{ik}):\text{-}e1(X_1,\ldots X_n).$$

Caso 4: E = E1 x E2

$$e(X_1,\ldots\ldots X_{n+m}):\text{-}e1(X_1,\ldots X_n),\ e2(X_{n+1},..X_m)$$

Caso 5: E=σ_F(E1). Por el lema 2 se asume que F es una selección simple AθB

$$E(X_1,\ldots X_n):\text{-}e1(X_1,\ldots X_n),\ X_A\theta X_B.$$

Donde XA, XB son variables o constantes para los atributos A y B respectivamente.
También se puede demostrar que en cualquier programa Datalog no recursivo, se puede encontrar una expresión equivalente del álgebra relacional para cada relación intencional Datalog.

2.11 La potencia de la recursividad en Datalog

Datalog permite expresar a partir de una base de datos de hechos padre(X,Y) y madre (X,Y) los progenitores de un determinado X con la relación de antepasado(X,Y) que se describió anteriormente.

antepasado(X,Y):-progenitor(X,Y).

antepasado(X,Y):-progenitor(X,Y), antepasado(Z,Y).

2.12 Evaluación de Datalog sin recursividad

El objetivo, a diferencia de Prolog, es calcular el conjunto respuesta de una relación.
Es un cálculo necesariamente descendente como es en Prolog. En lugar de partir de un objetivo, se parte de una relación de la cual hay que hallar su significado. Es decir, el conjunto de posibles tuplas pertenecientes a esa relación.
Al tratar con reglas no recursivas, no hay ciclos en el grado de dependencias y se puede encontrar un nodo p1 sin arcos de entrada.
Se calcula p1, después se puede calcular p2 ,…pn conectadas secuencialmente en el grafo p2 a partir de p1, p3 a partir de p2 y así recursivamente .

2.13 Evaluación de Datalog con recursividad

No se puede seguir la solución anterior porque no se dispone de ningún p sin arcos de entrada (es decir, cada regla de en un ciclo depende del resto de reglas del ciclo).
La idea es comenzar infiriendo algunos hechos y, por aplicaciones sucesivas de las reglas del ciclo, derivar más arcos siguiendo un procedimiento de búsqueda de punto fijo. El punto fijo encontrado es precisamente el significado del programa.

3. Implementaciones relacionadas con Datalog

En los últimos años, algunos investigadores se han dedicado a indagar sobre bases de datos deductivas, que han derivado en el desarrollo de sistemas capaces de trabajar con este tipo de base de datos. Datalog es un lenguaje declarativo, y aunque no ha tenido mucha demanda a nivel comercial, si es usado por investigadores de diversas universidades, algunos de ellos proporciona un entorno fácil de manejar y con multiplataforma. En este estudio se comentan algunos de estos sistemas, los cuales se describen en los siguientes apartados.

3.1 Sistema DES (DATALOG EDUCATIONAL SYSTEM) [16,17]

Este sistema es una implementación libre de código abierto, con multiplataforma. Su implementación está basado en Prolog, y ya se han realizado varias versiones a partir de su creación en el año del 2003. Actualmente fue lanzada la versión DES 3.7 que contiene: álgebra de registro de datos, relaciones y lenguajes de consulta SQL.

Presenta de igual manera, las siguientes características: evaluación completa recursiva con técnicas de memorización, eliminación de negación estratificada, restricciones de integridad, conexiones ODBC a sistemas externos de gestión de base de datos relaciones(RDBMS), registro de datos SQL y marcadores, texto API para aplicaciones externas, nuevos enfoques para consultas hipotéticas SQL, depuración declarativa de consultas Datalog y las vistas SQL, generación de casos de prueba para las vistas de SQL, apoyo a los valores nulos, combinación externa y global de predicados. Este sistema puede ser utilizado con diferentes intérpretes de Prolog.

Su principal objetivo de desarrollo, es tener un sistema sencillo, interactivo, multiplataforma y accesible para que los estudiantes obtengan conceptos fundamentales de una base de datos deductiva, con registro de datos, algebra relacional y consultas de SQL.

3.2 Sistema XSB

Este sistema fue implementado por DS. Warren [13], quien desarrolló un sistema que soporta negación estratificada y agregación (además de un meta-intérprete para

programas bien fundamentados), tuplas no cerradas y relaciones residentes en disco. Es un sistema Prolog extendido es caracterizado por implementar una semántica bien fundamentada para reglas con literales negativos en su cuerpo e implementar mecanismos de tablas de extensión, es multiplaforma, trabaja con Unix/Linux y Windows.

3.3 Sistema LOLA

El Sistema LOLA [24] permite la utilización de cláusulas declarativas recursivas, además de términos compuestos, negación, agregación y predicados definidos. Trabaja con un motor deductivo basado en el paradigma que evalúa ascendentemente operaciones relacionales y ofrece una interfaz en línea para clientes web.

3.4 Proyecto LDL en MCC

Este sistema originó el prototipo LDL++ [23,2], como un sistema de bases de datos deductivas con características como negación tanto estratificada como no estratificada, términos de tipo conjunto y agregados. Actualmente se puede usar vía Internet con un cliente con capacidades Java.

3.5 Sistema Coral

El Sistema Coral [12] trabaja con un lenguaje de consulta declarativo y soporta cláusulas generales de Horn con términos compuestos, agrupación por conjuntos negación, agregación y relaciones con tuplas donde sus variables son cuantificadas universales, no es multiplaforma, pues solo trabaja con Unix. Este sistema cuenta con una versión de programación orientada a objetos [20].

3.6 Sistema FLORID

El sistema FLORID [10] está orientado a objetos, trabaja con F-logic como un lenguaje de consulta además de definir los datos, debido a que sus autores tenían un interés por los datos semiestructurados. Este sistema ha sido extendido para tratar este tipo de datos, para integrarlos en la Web semántica.

3.7 Sistema NAIL

El sistema NAIL [11] trabajó con negación estratificada, negación bien fundamentada y negación estratificada modularmente. El lenguaje Glue [6], se añadió posteriormente para utilizar reglas lógicas con instrucciones SQL que se incorporaron en un lenguaje imperativo convencional. Este sistema es muy parecido a Coral, debido a que usa C++, pero tiene la limitante de que no trabaja en Windows.

3.8 Sistema ADITI 2

El sistema ADITI 2 [22,3], trabaja con bases de datos deductivas usando el lenguaje de programación lógica-funcional Mercury. Su limitación es que no trabaja con Windows y todas las operaciones relacionales son tratadas como residentes de disco. Se puede ver una visión general de este sistema en [8].

Conclusiones

De acuerdo a este estudio, podemos concluir que existen algunos sistemas que ofrecen las funcionalidades de Datalog de forma parcial, pues todavía no tienen todas las operaciones que se pueden trabajar en este Lenguaje. Otra de las desventajas es que muchos de ellos no son multiplataforma, ya que trabajan con Unix y otros más no lo hacen con Windows. Al realizar este estudio, se cumplen las expectativas de darle al lector herramientas que pueden ayudarle a manejar base de datos deductivas y puedan elegir cuál de los sistemas es pertinente para trabajar de una manera confiable y amigable, pues muchas veces el no contar con una implementación de alto nivel, desalientan a los usuarios a manejar dichos sistemas por no contar con una interfaz que sea de fácil manejo.

Bibliografía

[1] Álvarez ,I. Fundamentos de Inteligencia artificial , Universidad de Murcia, 1994.

[2] Derr, M. , Morishita, S. and Phipps ,G. “Design and Implementation of the Glue–NAIL Database System”, In Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 147–167, 1993.

[3] Arni, F., Ong, K., Tsur, S., Wang, H., and Zaniolo, C. The Deductive Database System LDL++. Theory and Practice of Logic Programming, pages 61-94, 2003.

[4] Beeri, C. And Ramakrishnan R., "On the Power of Magic", Journal of Logic Programming, 10(3,4):255-299, 1991.

[5] Ciao. Disponible en http://www.ciaohome.org.

[6] Date, C.J. Introducción a los sistemas de bases de datos, Person, Prentice hall, 2001.

[7] Gallaire, H. and Minker J., editors. Logic and Data Bases, Advances in Data Base Theory. Plemum Press, 1978.

[8] Jayen Vaghani, Kotagiri Ramamohanarao, David B. Kemp, Zoltan Somogyi, and Peter J. Stuckey. Design overview of the aditi deductive database system. In Proceedings of the Seventh International Conference on Data Engineering, pages 240-247, Washington, DC, USA, 1991. IEEE Computer Society.

[9] Kiessling,W. Schmidt,H. , Strauss, W. ,and Dünzinger,G. “DECLARE and SDS: Early Efforts to Commercialize Deductive Database Technology”, VLDB Journal, 3, 1994.

[10] Kifer,M. Lausen,G., Wu, J. "Logical Foundations of Object Oriented and Frame Based Languages", Journal of the ACM, vol. 42, p. 741-843, 1995.

[11] Phipps,G. Derr, M. A. and Ross K.A. , “Glue–NAIL!: A Deductive Database System”. In Proc. of the ACM SIGMOD Conf. on Management of Data, pp. 308–317, 1991.

[12] Ramakrishnan,R. Srivastava,D. Sudarshan, S. and Seshadri,P. “Implementation of the Coral Database System”, In Proc. of the ACM SIGMOD Conference on Management of Data, 1993.

[13] Rao ,P. Sagonas K, Swift, T, Warren, D, and Freire, J, “XSB: A System for Efficiently Computing WFS", Logic Programming and Non– monotonic Reasoning, 1997.

[14] Reiter, R. Towards a Logical Reconstruction of Relational Database Theory; in M. L. Brodie, J. L. Mylopoulus and J. W. Schmit(eds.). "On Conceptual Mode-lling", Springer- Verlag, 1984.

[15] Russell Stuart & Norving Peter. Artificial Intelligence (A Modern Approach). Prentice-Hall, 1995, ISBN: 0-13-103805-2. pp. 304.

[16] Saenz, F. Tabling with Support for Relational Features in a Deductive Database, Electronic Communications of the EASST Volume 55 ,2012

[17] Sáenz, P. F. Datalog Educational System V1.8.1, March 2010. Disponible en http://des.sourceforge.net/.

[18] Sagonas K., Swift,T., and Warren, D. S.. XSB as an efficient deductive database engine. In SIGMOD '94: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, ACM, pages 442-453, 1994.

[19] SICS AB. Sictus Prolog. Disponible en http://www.sictus.se/sictus.

[20] Srivastava, D, Ramakrishnan,R, Sudarshan, S, and Seshadri P., "Coral++: Adding Object– Orientation to a Logic Database Language", Proceedings of the International Conference on Very Large Databases, 1993

[21] SWI-Prolog. Disponible en http://www.swi-prolog.org.

[22] Vaghani, K. Ramamohanarao, D.B. Kemp, Z. Somogyi, and P.J. Stuckey, "Design Overview of the Aditi Deductive Database System", In Proc. of the 7th Intl. Conf. on Data Engineering, pp. 240–247, 1991.

[23] Zaniolo, C, Arni, N, and Ong, K."Negation and Aggregates in Recursive Rules: the LDL++ Approach", In Proc. Intl. Conf. On Deductive and Object Oriented DBs, 1993.

[24] Zukowski, U.and Freitag, B. , "The Deductive Database System LOLA", In: J. Dix and U. Furbach and Nonmonotonic Reasoning.and A. Nerode LNAI 1265, pp. 375–386. Springer, 1997.(Eds.). Logic Programming