# Una Alternativa Eficiente para Evaluación de Restricciones de Integridad en Bases de Datos Lógicas

Alonso Márquez

Leonid Tineo

Departamento de Computación,

Universidad Simón Bolívar,

Apartado Postal 89000, Caracas 1080A, VENEZUELA

## Resumen

El garantizar que un conjunto de restricciones sea satisfecho por una Base de Datos Lógica en todo momento obliga a que cada vez que se hagan Modificaciones, deba chequearse que ésta satisfaga las restricciones. La solución debe minimizar el cómputo aprovechando el supuesto de integridad (la base de datos ante de cualquier modificación está en un estado consistente con las restricciones), y evaluando sólo las restricciones que son relevantes a la modificación. Nuestra solución se basa en la consideración del conjunto de Restricciones de Integridad como la definición de un predicado especial "contradicción" y la generación incremental del modelo de la base de datos. Para lo cual definimos una transformación sintáctica de la Base de Datos Lógica, llamada la Completación Horn, que elimina el uso del Operador de Negación implícita. Esta transformación aplica a Bases de Datos Lógicas jerarquizada, lo cual resuelve la mayoría de los casos prácticos.

**Palabras Clave:** bases de datos lógicas, restricciones de integridad, evaluación incremental, completacion Horn, modelo perfecto, negación, modificación

# 1 INTRODUCCION

La semántica más ampliamente aceptada actualmente para Bases de Datos Lógicas es la semántica de Modelo Perfecto [NICOLAS78] [CHANDRA82] . Esta coincide con la semántica de Punto Fijo [APT82] [CHANDRA82], para el caso de Bases de Datos Lógicas sin Negación.

Como es sabido, el operador de Punto Fijo es un operador monótono [LLOYD87], luego su evaluación puede hacerse de manera incremental. Así, la Evaluación Incremental provee una forma sencilla y eficiente para generar el modelo de Bases de Datos Lógicas sin Negación con Modificaciones de tipo Inserción.

En el caso de Bases de Datos Lógicas con Negación, el mecanismo de generación del modelo que se utiliza es el Punto Fijo Estratificado [CHANDRA82] , caso particular de los mecanismos de prueba por encadenamiento hacia adelante, que es no monótono. Hasta ahora no se conoce ningún mecanismo de Evaluación Incremental de este operador.

Una extensión que se ha hecho a las Bases de Datos Lógicas es la posibilidad de establecer Restricciones de Integridad [LLOYD87], esto es: un conjunto de reglas lógicas que sirven para restringir el conjunto de modelos válidos para la base de datos. No se conoce una manera eficiente de hacer el chequeo de la satisfacción de las Restricciones de Integridad y asegurar que se generen modelos permitidos.

En este trabajo proponemos una solución eficiente al problema de garantizar la satisfacción de las Restricciones de Integridad por el Modelo Perfecto de Bases de Datos Lógicas. Esta solución se basa en la Evaluación Incremental de Bases de Datos Lógicas con Negación. Mediante una transformación denominada Completación Horn, basada en la completación [CLARK78], que consiste básicamente en reemplazar el uso del operador de Negación por un manejo explícito de la Negación.

Asumimos que el lector está familiarizado con los conceptos básicos del área de Bases de Datos Lógicas [GALLAIRE84] [ULLMAN88] [CERI89].

# 2 COMPLETACION HORN

La conocida Teoría de Pruebas nos da una manera sencilla de inferir las nuevas consecuencias que forman parte del modelo de la Base de Datos Lógica resultante de una operación de Modificación, esto debido a que se tienen reglas que definen a los predicados por comprensión.

En la Base de Datos Lógica no hay manera, de acuerdo a la Teoría de Pruebas, de inferir cuáles son las consecuencias que forman parte del modelo de la Base de Datos Lógica antes de una Modificación y no forman parte del modelo de la Base de Datos Lógica resultante de la Modificación. En consecuencia, si se está aplicando una estrategias de encadenamiento hacia adelante, habría que calcular en cada modificación todo el modelo de la Base de Datos Lógica.

## 2.1 Definición de la Completación Horn.

La Completación Horn de un Programa Lógico se basa principalmente en eliminar el uso del Operador de Negación y definir de manera explícita la Negación de los predicados del Programa Lógico. Para ello se hace uso de la completación de programas.

Se define la Eliminación de la Negación de una fórmula F (elimneg(F)) como:

- ri(t1,..,tai) si F= ri(t1,..,tai)
- no_ri(t1,..,tai) si F= not ri(t1,..,tai)
- elimneg(A) $\wedge$ elimneg(B) si F= (A$\wedge$B)
- elimneg(A) v elimneg(B) si F= (AvB)
- elimneg(A) $\leftarrow$ elimneg(B) si F= (A$\leftarrow$B)
- $\forall$X elimneg(A) si F= $\forall$X A
- $\exists$X elimneg(A) si F= $\exists$X A

- elimneg(not A) v elimneg(not B) si F= not(A∧B)
- elimneg(not A)∧elimneg(not B) si F= not(A∨B)
- ∃X elimneg(not A) si F= not ∀X A
- ∀X elimneg(not A) si F= not ∃X A

La Completación Horn de un predicado r (comp_horn(r)) es:

- Si r es un predicado extensional

  {r(t1,..,ta) / r(t1,..,ta) ∈ BDE} U

  {no_r(t1,..,ta) / r(ti,..,ta) ∉ BDE}

- Si r es un predicado intensional y la completación Clark de r es

  ∀X1.. ∀Xa (ri(X1,..,Xa) ↔ F)

  {elimneg( ∀X1.. ∀Xa (ri(X1,..,Xai) ← F)),

  elimneg( ∀X1.. ∀Xa ( not ri(X1,..,Xai) ← not F))

  }

La Completación Horn del programa PL (comp_horn(PL)) es:

- comp_horn(PL) = { R / R ∈ comp_horn(r), r predicado }

Por ejemplo, consideremos la Base de Datos Lógica:

BDL = { usa(X1,X2) :- X2=sparc, dicta(X1,bases_de_datos)., dicta(esther,bases_de_datos).}

La Completación Horn es:

comp_horn(BDL) = { ∀X1 ∀X2 ( usa(X1,X2) ← X2=sparc, dicta(X1,bases_de_datos) ).,
∀X1 ∀X2 ( no_usa(X1,X2) ← X2≠ sparc; no_dicta(X1,bases_de_datos) )., dicta(esther,bases_de_datos).
no_dicta(esther,esther)., no_dicta(bases_de_datos,bases_de_datos)., no_dicta(bases_de_datos,esther).}

## 2.2 Justificación de la Completación Horn.

Debemos asegurar que el Modelo Mínimo de la Base de Datos Lógica resultante de la Completación Horn restringida a los predicados del programa original coincida con el Modelo Perfecto de la Base de Datos Lógica original y que la parte correspondiente a los predicados "negativos" coincida con el complemento del Modelo preferencial del programa original con respecto a la Base de Herbrand del lenguaje del programa original.

Completitud de la Completación Horn: si BDL es una Base de Datos Lógica jerárquica y F es una fórmula, entonces comp_horn(BDL) $\models$ elimneg(F) si y sólo si comp_horn(BDL) $\not\models$ elimneg(not F).

Teorema Fundamental: si BDL es una Base de Datos Lógica jerárquica y F es una fórmula, entonces BDL $\models$ F si y sólo si comp_horn(BDL) $\models$ elimneg(F).

Estos resultados se demuestran por inducción sobre la jerarquización de los predicados de la Base de Datos Lógica e inducción sobre la estructura de la fórmula F. No presentamos estas demostraciones aquí por razones de espacio, la demostración formal puede conseguirse en [TINEO92].

# 3 EVALUACION INCREMENTAL DE BASES DE DATOS LOGICAS.

En general, si U es una operación de Modificación, al aplicarla sobre una Base de Datos Lógica, el resultado es otra Base de Datos Lógica cuyo Modelo Perfecto viene dado por el Modelo Perfecto de la Base de Datos Lógica original más las consecuencias nuevas que pueden alcanzarse en la nueva Base de Datos Lógica menos las consecuencias que eran alcanzables en la Base de Datos Lógica original pero no lo son en la nueva [MACHANDA88] [ULLMAN88].

Por la propiedad de monotonicidad del operador de Punto Fijo se obtiene que el modelo

resultante de la Modificación en el caso de inserciones en bases de datos sin negación, es el Punto Fijo (PF) del operador de consecuencia directa (cd) aplicado al modelo previo a la Modificación con el nuevo hecho.

Mediante la Completación Horn hemos dado una definición explícita de la Negación; más aún, el Teorema Fundamental garantiza que la semántica de la Completación Horn es equivalente (vía la Eliminación de la Negación "elimneg") a la semántica del Modelo Perfecto.

De esta manera la nueva parte del modelo que se infiere ante una Inserción para los predicados "negativos" de la Completación Horn corresponden a la parte del modelo que deja de ser consecuencia luego de la actualización en al Base de Datos Lógica.

El problema de propagar las consecuencias de la Eliminación es análogo al hacer Inserciones en una Base de Datos Lógica con Negación, es decir se quiere inferir la parte positiva (nuevas consecuencias) así como la negativa (consecuencias que dejan de poder alcanzarse).

Surge una vez más como solución el uso de la Completación Horn, con la cual se le da una definición explícita a las consecuencias negativas. En este caso Eliminar A es equivalente a Insertar el átomo correspondiente a la negación de A en la Completación Horn.

# 4 EVALUACION DE LAS RESTRICCIONES DE INTEGRIDAD

Proponemos extender la base de datos intensional con un nuevo predicado, al cual llamaremos "contradicción", cuya definición viene dada por un conjunto de reglas, cuyos cuerpos son los cuerpos de las cláusulas Horn negativas que especifican las Restricciones de Integridad. Bajo la citada transformación, la base de datos es consistente si y solo si "contradicción" no es parte del modelo.

Dado que "contradicción" es un nuevo predicado de la Base de Datos Intensional, podemos conseguir siempre una Jerarquización donde el predicado contradicción se encuentra en el último nivel, de esta manera, evaluamos el predicado contradicción, una vez que se ha

generado el nuevo modelo de la base de datos.

Ahora bien, una vez hecha la generación incremental del modelo, si se viola una Restricción de Integridad, la base de datos ha quedado en un estado inconsistente. Hay que restaurar la consistencia de la base de datos, la solución es deshacer la Modificación realizada. Esto se logra aplicando la Modificación inversa.

# 5 CONCLUSIONES

Como la Completación Horn da una definición explícita de la Negación de los predicados, al aplicar operaciones de Modificación es posible deducir no sólo el conjunto de consecuencias que se agregan, sino, también el conjunto de consecuencias que eran alcanzables en la Base de Datos Lógica previa a la Modificación y que no lo son luego.

La solución que hemos propuesto para la generación de modelo de Bases de Datos Lógicas con Negación es mucho más eficiente que la solución que deberíamos hacer comúnmente de genera todo el modelo cada vez que hay una Modificación, pues aprovecha el modelo calculado previamente haciendo uso del operador de Punto Fijo que sólo hará los cálculos necesarios para obtener las nuevas consecuencias.

Para el manejo de Restricciones de Integridad definimos la Base de Datos Lógica Aumentada con el predicado "contradicción" como la Base de Datos Lógica donde se ha creado un nuevo predicado, llamado "contradicción", cuyas reglas tienen como cuerpo cada una de las Restricciones de Integridad.

Para garantizar que la Base de Datos Lógica es consistente en todo momento, rechazamos las modificaciones que hagan que "contradicción" sea parte del modelo. Debido a las propiedades de la Completación Horn, el rechazar una Modificación es equivalente a aplicar la operación inversa sobre resultado de dicha Modificación.

Nuestro trabajo fue centrado en el problema de evaluación de Bases de Datos Lógicas Jerarquizadas, principalmente teniendo en cuenta que este es un caso interesante, de com-

plejidad no elevada y que resuelve la mayoría de los problema prácticos; sin embargo, puede ser extendido a Bases de Datos Lógicas Estratificadas.

# 6 REFERENCIAS


[APT82] Apt, K. R. and M. H. Van Emden, "Contributions to the theory of logic programming", J. ACM 1982, 29:3.

[CERI89] Ceri, S., G. Gottlob and L. Tanca, "What you always wanted to know about Datalog (and never dared to ask)", IEEE Transactions on knowdledge and data engineering, 1:1, 1989.

[CHANDRA82] Chandra, A. K. and D. Harel, "Structure and complexity of relational queries", J. Computer and System Sciences 1982, 25:1.

[CLARK78] Clark, K. L., "Negation as Failure", in Logic and Data Bases, Gallaire, H. and J. Minker (eds), Plenum Press, New York, 1978.

[GALLAIRE84] Gallaire, H., J. Minker and J. M. Nicolas, "Logic and Databases: A Deductive Approach", Computing Surveys 1984, 16:2.

[LLOYD87] Lloyd, J. W., Foundations of Logic Programming. Springer-Verlag, Berlin, 1987.

[NICOLAS78] Nicolas, J. M. and H. Gallaire, "Data Base: Theory vs Interpretation", in Logic and Data Bases, Gallaire, H. and J. Minker (eds), Plenum Press, New York, 1978.

[SADRI88] Sadri, F. and R. Kowalski, "A theorem-proving approach to database integrity", in Foundations of Deductive Databases and Logic Programming, Minker, J. (ed) Morgan Kaufmann Publisher, 1988.

[TINEO92] Tineo, L., "Evaluación incremental de restricciones de integridad en bases de datos logicas", MSc. thesis, Universidad Simón Bolívar, 1992.

[ULLMAN88] Ullman, J. D., Principles of Database and Knowledge Base Systems Vollume I, Computer Science Press, 1988.