FOLRE:
un sistema para el tratamiento integrado de actualizaciones en bases de datos*

E. Teniente, C. Farré, C. Martín, E. Mayol, J. A. Pastor, J.Sistac, T. Urpí,

Universitat Politècnica de Catalunya
Dept. de LSI - Facultat d'Informàtica
Pau Gargallo, 5 E-08028 Barcelona
{ teniente | martin | mayol | pastor | sistac | urpi}@lsi.upc.es, farre@goliat.upc.es

Resumen

En este artículo se describe la arquitectura del sistema de gestión de bases de datos FOLRE, cuyo objetivo principal es el de proporcionar capacidades avanzadas de actualización de bases de datos en un marco integrado y teóricamente bien fundado. En la actualidad, el sistema FOLRE gestiona la actualización de vistas, la comprobación y el mantenimiento de restricciones de integridad, la materialización de vistas, la monitorización de condiciones y la generación y el procesamiento de transacciones predefinidas; además de las capacidades tradicionales de definición y de consulta de una base de datos.

El sistema está construido a partir de sistemas y resultados teóricos ajenos a nuestro grupo -para la semántica, consultas y almacenamiento y gestión de transacciones- y de nuestra investigación previa en la actualización avanzada de bases de datos. Esperamos que el sistema FOLRE permita acortar la distancia actual entre los productos y los prototipos de sistema de gestión de bases de datos deductivas.

PALABRAS CLAVE: sistemas de gestión de bases de datos; bases de datos deductivas; actualización de bases de datos; actualización de vistas, satisfacción de integridad.

1. Introducción

La emergente necesidad de incorporar más "inteligencia" en los sistemas de información actuales conlleva un incremento de la demanda en infraestructura tecnológica para su implementación y uso. Enmarcados en este contexto dominado por la demanda actual del mercado, los Sistemas de Gestión de Bases de Datos Deductivas (SGBDD) parecen empezar a evolucionar finalmente de herramientas de investigación a sistemas reales de bases de datos. La gran cantidad de investigación teórica dedicada al campo de las bases de datos deductivas o lógicas durante la última decada está materializándose actualmente en algunas implementaciones de SGBDD, ya sea al nivel de prototipo o de sistema comercial (véase por ejemplo los prototipos descritos en [VLDB94]).

La mayoría de los sistemas actuales se centran principalmente en proporcionar facilidades para el almacenamiento y la consulta eficientes de grandes cantidades de información. La semántica de las base de datos deductivas, el procesamiento de consultas y los sistemas de almacenamiento deductivo se encuentran entre los temas más estudiados en este área. Sin embargo, pocos SGBDD actuales incorporan algún mecanismo para la actualización de datos, con excepción de actualizaciones de relaciones básicas. En este sentido, estos sistemas no han extendido en demasía las capacidades ofrecidas actualmente por los sistemas de gestión de bases de datos relacionales.

Esto es cierto incluso cuando existen resultados teóricos satisfactorios para la mayoría de problemas relacionados con la actualización de bases de datos deductivas, como por ejemplo la *actualización* y *materialización de vistas*, la *comprobación* y *mantenimiento de restricciones de*

* Este proyecto de investigación está financiado parcialmente por la CICYT, proyecto TIC94-0512.

integridad y la *monitorización de condiciones*. Algunos métodos prometedores han aparecido con el objetivo de solucionar estos problemas (véase por ejemplo las referencias que se mencionan en el apartado 4 de este artículo). Sin embargo, la mayoría de estos métodos han tratado su respectivo problema de un modo aislado, sin aún tomar en consideración la fuerte relación entre todos estos problemas.

De este modo llegamos a las motivaciones originales de nuestra investigación en bases de datos deductivas y al objetivo principal del sistema FOLRE[1] presentado en este artículo: "el tratamiento integrado de problemas avanzados de actualización en bases de datos deductivas mediante un SGBDD especializado". Con este propósito, el sistema está construido a partir de sistemas y resultados teóricos ajenos a nuestro grupo -para la semántica, consultas y necesidades de almacenamiento y gestión de transacciones- y de nuestra investigación previa en la actualización avanzada de bases de datos.

Por lo que a investigaciones previas de nuestro grupo se refiere, se propusieron distintos métodos competitivos para resolver la mayoría de problemas mencionados anteriormente. Sin embargo, si bien fueron desarrollados en base a un cuerpo teórico común, nuestros métodos compartían con los métodos previos parte de la crítica de aislamiento mencionada anteriormente. Para empezar a resolver esta limitación, y ya dentro de las premisas del proyecto FOLRE, en [TU95] se estableció la base teórica para la especificación y la combinación de los problemas anteriores en un marco integrado. A partir de estos resultados y de los prototipos aislados de que ya disponíamos hemos construido la primera versión del prototipo de SGBDD FOLRE que se presenta en este artículo.

Este artículo se organiza de la siguiente forma. En la sección 2 se repasan los conceptos básicos de las bases de datos deductivas. La sección 3 describe los lenguajes de definición y de manipulación de datos de nuestro sistema. En la sección 4 se presenta la arquitectura del sistema FOLRE. Finalmente, en la sección 5 se presentan las conclusiones y se comenta posible trabajo futuro.

2. Bases de datos deductivas

Las bases de datos deductivas se basan en el concepto de reglas deductivas (vistas) que definen el conocimiento común compartido por diferentes usuarios. Las reglas deductivas permiten deducir nuevos hechos (hechos derivados o vista) a partir de los hechos almacenados físicamente en la base de datos y de otros hechos derivados.

Las bases de datos deductivas utilizan la programación lógica como lenguaje de base y generalizan las bases de datos relacionals al superar las limitaciones de los lenguajes relacionales en la definición de vistas y de restricciones complejas. Esta extensión del concepto de base de datos relacional proporciona múltiples ventajas como facilitar el desarrollo de aplicaciones, evitar el múltiple desarrollo de programas con funcionalidades similares o permitir capacidad de razonamiento.

Las bases de datos deductivas son útiles para gestionar la mayoría de aplicaciones actuales, siendo especialmente aconsejables para aplicaciones que se caracterizan por la necesidad de realizar consultas y modificaciones rápidas sobre los datos, com en el caso de las bases de datos relacionales, pero que necesitan realizar operaciones más complejas sobre los datos que las requeridas por las aplicacions previas.

3. Lenguajes de definición y de manipulación de datos

3.1 Lenguaje de definición de datos

Nuestro lenguaje de definición de datos permite especificar todos aquellos elementos que pueden aparecer en una base de datos deductiva. De este modo, el diseñador puede definir predicados básicos, predicados derivados, restricciones de integridad y las claves de los predicados básicos y de los derivados. Adicionalmente, el diseñador puede también definir conocimiento adicional sobre estos elementos.

---

[1] **FOLRE** es a la vez un acrónimo para "**F**acing **O**n towards **L**ogic database **R**ule **E**nforcement" y el nombre de la estructura recientemente recuperada para aumentar la altura de los típicos *castells* humanos catalanes.

Para predicados derivados, el diseñador puede especificar si quiere que sean materializados o no, o sea, si quiere almacenar explícitamente su extensión en la base de datos. Además, puede especificar también el tipo de actualizaciones (inserciones, supresiones y/o modificaciones) que el sistema debe monitorizar sobre estos predicados.

En restricciones de integridad el diseñador puede distinguir entre aquellas restricciones que deben ser comprobadas, de tal modo que toda transacción que las viole sea rechazada; y aquellas que deben ser mantenidas, o sea, aquellas para las que el sistema debe intentar buscar acciones compensatorias en el caso de que vayan a ser violadas por una transacción.

El diseñador puede especificar también transacciones predefinidas. Éstas son programas precompilados que corresponden a actualizaciones que serán requeridas a menudo y, de este modo, el sistema debe optimizarlas en tiempo de compilación.

En la sección 4 explicaremos estos aspectos con mayor detalle.

Ejemplo: definición de una base de datos para una "Oficina de Empleo" encargada de concertar entrevistas entre personas que buscan trabajo y compañías que buscan contratar trabajadores.

En esta base de datos se guarda físicamente la información de las personas que trabajan (mediante el predicado básico Emp(x)), de las personas que solicitan trabajo (App(x)), de las compañías que buscan trabajadores (Comp(y)) y de las entrevistas concertadas entre personas y compañías (Int(x,y)). Además, se conoce la información de las personas que son candidatas a un puesto de trabajo (mediante el predicado derivado Cand(x)), entendiéndose que una persona es candidata si tiene concertada una entrevista con alguna compañía. Esta información está materializada y el sistema debe notificar cuando se produce alguna supresión de la misma.

La base de datos debe satisfacer dos restricciones de integridad: que los empleados no pueden buscar trabajo (restricción Ic1) y que los candidatos deben solicitar trabajo (restricción Ic2). Para la primera de estas restricciones se aplicará el enfoque de comprobación y para la segunda el de mantenimiento. Finalmente, se ha definido una transacción predefinida de alta de empleado.

Descrito mediante el lenguaje de definición de datos de FOLRE el ejemplo es:

base predicate Emp (x: string)
key: x
description: x es un empleado

base predicate Comp (y: string)
key: y
description: y es una compañía

base predicate App (x: string)
key: x
description: x solicita trabajo

base predicate Int (x: string, y: string)
key: x
description: x tiene entrevista con y

derived predicate
Cand (x) ← Int (x,y) ∧ Comp (y)
key: x
type: materialized
monitor: deletions
display x ha sido suprimido como candidato
description: x es un candidato a puesto de trabajo si tiene concertada alguna entrevista con una compañía

integrity constraint
ic1 (x) ← Emp (x) ∧ App (x)
type: checking
description: los empleados no pueden solicitar trabajo

integrity constraint
ic2 (x) ← Cand (x) ∧ ¬ App (x)
type: maintenance
description: los candidatos deben solicitar trabajo

predefined transaction Alta-empleado (Per)
is {conjunto de actualizaciones especificadas en el lenguaje de manipulación de datos con parámetros formales}

3.2 Lenguaje de manipulación de datos

En FOLRE los usuarios pueden realizar inserciones, supresiones y modificaciones de hechos básicos y de hechos derivados. Estas actualizaciones se especifican de la forma siguiente:

ιP(**K,X**): denota una inserción del hecho P(**K,X**).

δP(**K,X**): denota una supresión del hecho P(**K,X**).

μP(**K,X,Y**): denota una modificación del hecho P(**K,X**) por P(**K,Y**), donde **X** ≠ **Y**.

El usuario puede realizar también actualizaciones que consisten de un conjunto de inserciones y/o supresiones y/o modificaciones de hechos básicos y derivados. Véase [UO92] para una definición más formal de nuestro lenguaje de manipulación.

Ejemplo1: la actualización consistente en la inserción del hecho derivado Cand(Juan) y en la supresión del hecho básico App(María) se expresaría como { ιCand(Juan), δApp(Maria) }.

Ejemplo2: la petición de transacción predefinida explicada en la sección 3.1 es:

predefined transaction Alta-empleado(Per) **is** { ιEmp(Per) }

4. Arquitectura del sistema FOLRE

Dado que el énfasis principal de FOLRE concierne el tratamiento de las actualizaciones de una base de datos, nuestro sistema se compone de dos partes claramente diferenciadas. Primero, un *front-end* que actúa como interfaz con el usuario o el diseñador y cuyo objetivo principal es el de gestionar todos aquellos aspectos relacionados con la definición y la actualización de una base de datos deductiva. Segundo, una plataforma deductiva (en nuestro caso el sistema ECLiPSe [ECL94]) que actúa como *back-end* y que incluye el soporte de almacenamiento en disco y facilidades de gestión de consultas y de transacciones. Esta idea se ilustra en la figura 1.

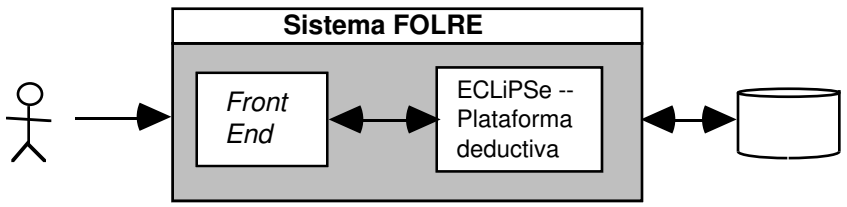

Figura 1 - Arquitectura del sistema

Debe mencionarse que, aunque la plataforma deductiva sobre la que está construido actualmente nuestro sistema sea ECLiPSe, el módulo front-end se ha construido de tal manera que FOLRE podría ser transportado fácilmente a otras plataformas deductivas que satisfacieran nuestros requisitos.

El front-end se divide en tres componentes, tal y como se muestra en la figura 2: Gestor de la Definición, Gestor de Consultas y Gestor de Actualizaciones. Estos componentes se explican detalladamente en lo que resta de esta sección.

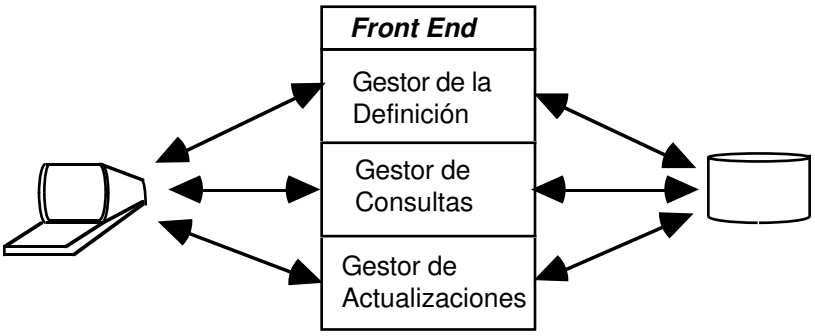

Figura 2 - Arquitectura del Front-End

4.1 Gestor de la Definición

El Gestor de la Definición permite al diseñador especificar todos los aspectos del esquema de una base de datos deductiva descritos en la sección 3.1 y efectuar, en tiempo de compilación, todas las actividades necesarias para garanitzar un comportamiento satisfactorio de la base de datos en tiempo de ejecución.

Los componentes del Gestor de la Definición se muestran en la figura 3.

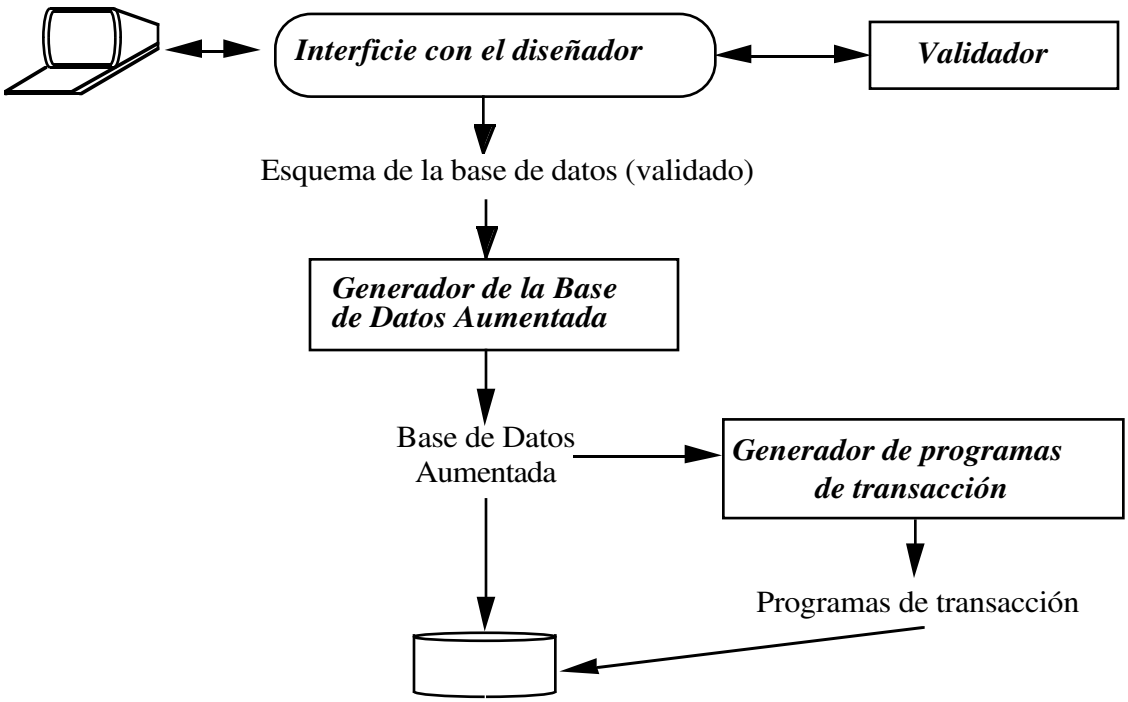

Figura 3 - Arquitectura del Gestor de la Definición

El módulo Interficie con el Diseñador gestiona toda la interacción de entrada/salida y permite al diseñador especificar todos los aspectos de una base de datos mencionados anteriormente. Esta especificación se realiza con el lenguaje de definición de datos ilustrado en la sección 3.1.

El Validador es una herramienta que permite al diseñador validar la definición del esquema de una base de datos deductiva. Entre otros, permite validar la satisfactibilidad del esquema, la vivacidad de una vista, la redundancia de una restricción de integridad, etc. En el momento actual se está trabajando en la implementación de este módulo en base a los resultados teóricos obtenidos recientemente en [CTUF96, DTU96].

Dado un esquema validado, el Generador de la Base de Datos Aumentada obtiene automáticamente la base de datos aumentada correspondiente a dicho esquema. Es decir, obtiene un modelo que especifica los cambios que ocurren durante la transición de un estado de la base

de datos a un nuevo estado motivada por la aplicación de una transacción, de la forma descrita en los trabajos previos de nuestro grupo [Oli91, UO92, UO94]. La base de datos aumentada tiene un papel muy importante en nuestro sistema puesto que todos los métodos propuestos por nuestro grupo para tratar problemas relacionados con la actualización de una base de datos se basan en este concepto.

En FOLRE, el diseñador puede también especificar en tiempo de definición peticiones de transacción predefinida que consisten en un conjunto de actualizaciones propuestas sobre hechos básicos y derivados, definidas mediante parámetros formales. El Generador de Programas de Transacción genera automáticamente, en tiempo de compilación, los programas de transacción correspondientes tomando en consideración, bajo el control de instrucciones específicas de control, la información proporcionada por el esquema de la base de datos y la base de datos aumentada.

Estos programas de transacción se almacenan físicamente en la base de datos para que, en tiempo de ejecución, los usuarios puedan instanciar los parámetros formales con parámetros reales y las actualizaciones proporcionadas por la ejecución de los programas de transacción sean aplicadas a la base de datos, tal y como explicaremos en la sección 4.3. Este módulo se ha implementado según los procedimientos formales descritos en [Pas94, PO94, PO95], que contemplan el tratamiento de los problemas de actualización de vistas y de comprobación y mantenimiento de restricciones de integridad.

4.2 Gestor de Consultas

Durante los últimos años se ha dedicado mucha investigación al campo de procesamiento de consultas (véase por ejemplo [BR86, Ull88]). Puesto que la mayoría de plataformas deductivas actuales, y en particular ECLiPSe, implementan las técnicas más recientes y eficientes, nuestro sistema utiliza las facilidades proporcionadas por ECLiPSe para gestionar las consultas. Por consiguiente, el Gestor de Consultas se encarga únicamente de traducir la consulta efectuada por el usuario al lenguaje de la plataforma ECLiPSe. Ésta gestiona eficientemente la consulta y, a continuación, el Gestor de Consultas interpreta debidamente los resultados proporcionados por la plataforma deductiva.

4.3. Gestor de Actualizaciones

La finalidad del Gestor de Actualizaciones es la de permitir al usuario actualizar satisfactoriamente hechos básicos y derivados. En aras de satisfacer esta necesidad, este componente debe controlar las restricciones de integridad, las vistas a materializar y las condiciones a monitorizar. Sin lugar a dudas, éste es el componente más característico, importate e innovador de FOLRE ya que es el encargado de gestionar las actualizaciones de un modo integrado, objetivo principal de nuestro proyecto.

El sistema FOLRE permite al usuario efectuar dos tipos de actualizaciones distintas: *on-line* y predefinidas. Puesto que ambos tipos de actualizaciones deben procesarse de forma distinta, en la arquitectura del sistema hemos distinguido entre dos componentes distintos: el Gestor de Actualizaciones *On-line* y el Gestor de Actualizaciones Predefinidas.

4.3.1. Gestor de Actualizaciones *On-line*

El Gestor de Actualizaciones *On-line* es el componente del sistema que permite al usuario solicitar en tiempo de ejecución la aplicación de un conjunto de actualizaciones sobre una base de datos deductiva. Dada una petición de actualización del usuario, este componente es el encargado de determinar los cambios de contenido de la base de datos que garantizen que la actualización se realize efectivamente, que se continúan cumpliendo las restricciones de integridad, que las vistas a materializar son actualizadas correctamente y que todas las condiciones sean monitorizadas.

La figura 5 muestra la arquitectura del Gestor de Actualizaciones *On-line*, que se compone de tres módulos: el Módulo de Actualización de Vistas y Mantenimiento de Restricciones de Integridad; el Módulo de Comprobación de Restricciones de Integridad; y el Módulo de Mantenimiento de Vistas Materializadas y Monitorización de Condiciones.

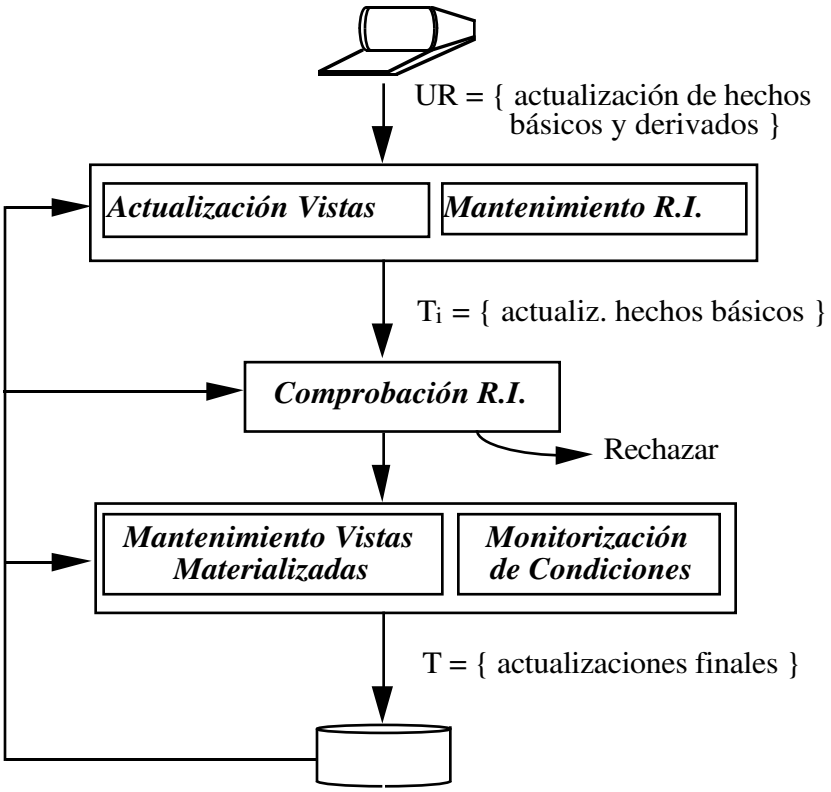

Figura 5- Arquitectura del Gestor de Actualizaciones *On-line*

a) Módulo de Actualización de Vistas y Mantenimiento de Restricciones de Integridad

El Módulo de Actualización de Vistas y Mantenimiento de Restricciones de Integridad trata dos problemas distintos pero íntimamente relacionados: la actualización de vistas y el mantenimiento de restricciones de integridad.

El problema de la actualización de vistas [KM90, GL91, AT92, CHM95] consiste en determinar como una petición de actualización de un hecho de una vista debe ser traducido apropiadamente en actualizaciones de los hechos básicos subyacientes. El enfoque que sigue nuestro sistema es el de obtener todas las traducciones que satisfacen una actualización de vista.

El problema del mantenimiento de las restricciones de integridad [ML91, Wüt93, Dec96] consiste en, dada una base de datos consistente y una transacción que viola alguna restricción de integridad, determinar un conjunto de actualizaciones de hechos básicos adicionales a añadir a la transacción original de tal manera que la transacción resultante satisfaga todas las restricciones de integridad.

En nuestro sistema, estos dos problemas son tratados conjuntamente de la forma propuesta en [TO92, MT93, MT95, TO95]. Dada una petición de actualización del usuario que consiste en un conjunto de actualización de hechos básicos y derivados, el Módulo de Actualización de Vistas y Mantenimiento de Restricciones de Integridad traduce automáticamente la petición en transacciones que consisten únicamente en actualizaciones de hechos básicos y que satisfacen tanto la actualización solicitada como las restricciones de integridad a mantener. Puesto que, en general, pueden obtenerse distintas transacciones resultantes, el usuario puede guiar el proceso de traducción y, eventualmente, elegir entre las soluciones obtenidas.

b) Módulo de Comprobación de Restricciones de Integridad

Una vez obtenidas las transacciones anteriores, debe verificarse que éstas no violan las restricciones de integridad especificadas como de comprobación. Este problema es conocido con el nombre de comprobación de restricciones de integridad (véase [GCM+94] y su lista de referencias). Si una transacción viola alguna de estas restricciones de integridad, debe ser

rechazada. En caso contrario, es aceptada y el sistema procede a la determinación de vistas a materializar y de condiciones a monitorizar.

Nuestras contribuciones en este área utilizan la base de datos aumentada para comprobar de manera eficiente si una transacción viola alguna restricción de integridad [Oli91, UO92, UO94].

c) Módulo de Mantenimiento de Vistas Materializadas y Monitorización de Condiciones

Una vista está materializada si su extensión se almacena explícitamente en la base de datos. Dada una transacción, el problema de mantenimiento de vistas materializadas consiste en determinar incrementalmente los cambios que deben ser realizados en la extensión de la vista materializada para que esta sea modificada acorde a la transacción [CW91, GMS93]. El sistema debe determinar también los cambios provocados por la transacción en las condiciones especificadas por el diseñador. El problema de determinar incrementalmente estos cambios se conoce con el nombre de monitorización de condiciones [RCB+89, QW91].

En nuestro grupo se ha propuesto también un método para tratar estos dos problemas [UO92]. Este método utiliza la base de datos aumentada para determinar las actualizaciones que deben realizarse sobre las vistas materializadas y los cambios en las condiciones a monitorizar.

Una vez el sistema FOLRE ha determinado ya las vistas a materializar y las condiciones a monitorizar, se obtienen un conjunto de transacciones globales resultantes. En este momento, el usuario debe elegir cual de estas transacciones debe ser aplicada físicamente a la base de datos y FOLRE se encargará de aplicar esta transacción.

4.3.2. Gestor de Actualizaciones Predefinidas

Las actualizaciones predefinidas corresponden a las peticiones de transacción predefinidas especificadas por el diseñador de la base de datos en tiempo de compilación. El Gestor de Actualizaciones Predefinidas es el componente del sistema FOLRE que permite al usuario ejecutar las actualizaciones predefinidas mediante la ejecución de sus correspondientes programas de transacción. Este componente se compone de un único módulo: el Procesador de Transacciones, como se muestra en la figura 6.

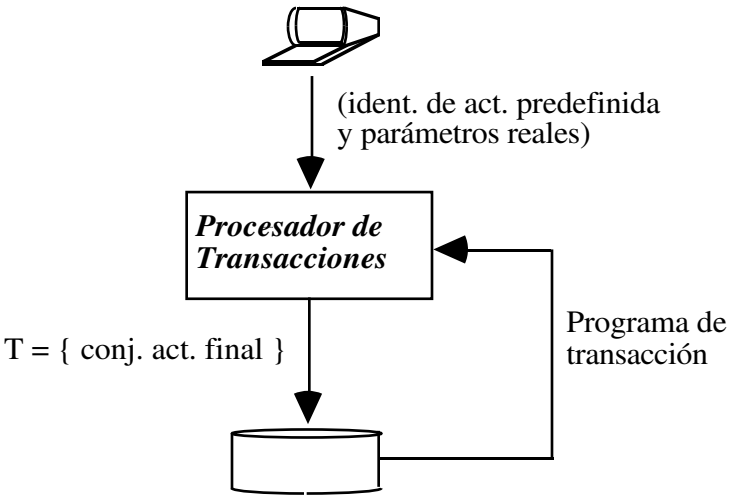

Figura 6 - Arquitectura del Gestor de Actualizaciones Predefinidas

Para actualizar una base de datos deductiva mediante una transacción predefinida parametrizada, el usuario debe seleccionar la correspondiente petición de transacción predefinida e instanciar sus parámetros con valores reales. El Procesador de Transacciones usa estos parámetros para llamar al programa de transacción correspondiente a la petición del usuario, ejecuta dicho programa y obtiene como resultado un conjunto de actualizaciones de hechos básicos a aplicar a la base de datos deductiva.

5. Conclusiones y trabajo futuro

En este artículo hemos descrito la arquitectura del sistema FOLRE. El objetivo principal de este sistema es el de extender los sistemas de bases de datos deductivas actuales para proporcionar capacidades de actualización más avanzadas como actualización de vistas, comprobación y mantenimiento de restricciones de integridad, mantenimiento de vistas materializadas y monitorización de condiciones.

En el momento actual, disponemos de una primera versión de FOLRE que permite ya una gestión integrada de actualizaciones avanzadas. La versión actual de FOLRE incorpora todos los módulos descritos en este artículo, con la excepción del módulo validador (véase la sección 4.1) Como trabajo futuro, nos planteamos integrar en el sistema aquellos módulos de los que ya hemos obtenido algunos resultados teóricos nuevos y aumentar la eficiencia de algunos de los módulos actuales lo que implicará probablemente el rediseño de algunos de los componentes actuales del sistema.

Referencias

[AT92] Atzeni, P.; Torlone, R. "Updating Intensional Predicates in Datalog", *Data & Knowledge Engineering* (8), 1992, pp. 1-17.

[BR86] Bancilhon, F; Ramakrishnan, R. “An Amateur’s Introduction to Recursive Query Processing Strategies”. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Washington D.C., 1986, pp. 16-152.

[CHM95] Chen, I.M.; Hull, R.; McLeod, D. "An Execution Model for Limited Ambiguity Rules and Its Application to Derived Data Update", ACM Transactions on Database Systems, Vol. 20, No. 4, 1996, pp. 365-413.

[CTUF96] D.Costal, E.Teniente, A.Urpí, C.Farré. "Handling Conceptual Model Validation by Planning", to appear in*Int. Conf. on Advanced Information Systems Engineering (CAiSE'96)*, Springer-Verlag, Creta, 1996.

[CW91] Ceri, S.; Widom, J. "Deriving production rules for incremental view maintenance", *Proc. of the 17th. VLDB Conf.*, Barcelona, 1991, pp 577-589.

[Dec96] H.Decker. "An Extension of SLD by Abduction and Integrity Maintenance for View Updating in Deductive Databases", *to appear in Joint Int. Conf. and Symposium on Logic Programming (JICSLP'96)*, 1996.

[DTU96] H.Decker, E.Teniente, A.Urpí. "How to Tackle Schema Validation by View Updating", *Fifth Int. Conf. on Extending Data Base Technology (EDBT'96)*, Springer-Verlag, Avignon, 1996, pp. 535-549.

[ECL94] "ECLiPSe User Manual", Technical Report, ECRC, 1994.

[GCM+94] García, C; Celma, M.; Mota, L.; Decker, H. "Comparing and Synthesizing Integrity Checking Methods for Deductive Databases", *Proc. of the 10th ICDE*, Houston, USA, 1994, pp. 214-222.

[GL91] Guessoum, A.; Lloyd, J.W. "Updating Knowledge Bases II", *New Generation Computing*, Vol. 10, 1991, pp. 73-100.

[GMS93] Gupta, A; Mumick, I.S.; Subrahmanian, V.S. “Maintaining Views Incrementally”, *ACM SIGMOD*, Washington, 1993, pp. 157-166.

[KM90] Kakas, A.; Mancarella, P. "Database Updates through Abduction", *Proc. of the 13th Int. Conf. on Very Large Data Bases (VLDB)*, Brighton, 1987, pp. 61-70.

[ML91] Moerkotte, G; Lockemann, P.C. "Reactive Consistency Control in Deductive Databases", *ACM Transactions on Database Systems*, Vol. 16, No. 4, December 1991, pp. 670-702.

[MT93] Mayol, E.; Teniente, E. "Incorporating Modification Requests in Updating Consistent Knowledge Bases", 4th Int. Workshop on the Deductive Approach to Information Systems and Databases, 1993, pp. 335-360.

[MT95] Mayol, E.; Teniente, E. "Towards an Efficient Method for Updating Consistent Deductive Databases", *Basque International Conference on Information Technology (BIWIT'95)*, IEEE Computer Society Press, Donostia, 1995, pp. 113-123.

[Oli91] Olivé, A. "Integrity Checking in Deductive Databases", Proc. 17th Int. Conf. on Very Large Data Bases (VLDB), Barcelona, 1991, pp. 513-523.

[Pas94] Pastor, J.A. "Extending the Synthesis of Update Transaction Programs to handle Existential Rules in Deductive Databases", 5th Int. Workshop on the Deductive Approach to Information Systems and Databases, Aiguablava, 1994, pp. 189-218.

[PO94] Pastor, J.A.; Olivé,A. "An Approach to the Synthesis of Update Transactions in Deductive Databases", Proc. of the CISMOD'94 (Fifth International Conference on Information Systems and Management of Data), Madras, India, October 1994, pp. 93-110.

[PO95] Pastor, J.A.; Olivé,A. "Supporting Transaction Design in Conceptual Modelling of Information Systems", *Proc. of the CAiSE-95 Conference*, 40-53, Jyväskylä, 1995, pp. 40-53.

[QW91] Qian, X.; Wiederhold, G. "Incremental recomputation of active relational expressions", *IEEE Trans. on knowledge and data engineering*, Vol. 3, No. 3, September 1991, pp. 337-341.

[RCB+89] Rosenthal, A.; Chakravarthy,S.; Blaustein, B.; Blakeley, J. "Situation monitoring for active databases", *Proc. of the 15th VLDB Conf.*, Amsterdam, 1989, pp. 455-464.

[TO92] Teniente, E.; Olivé. A. "The Events Method for View Updating in Deductive Databases", Proc. Int. Conf. on Extending Database Technology (EDBT'92), Vienna, 1992, pp. 245-260.

[TO95] Teniente, E.; Olivé. A. "Updating Knowledge Bases while Maintaining their Consistency", The VLDB Journal, Vol. 4, Núm. 2, 1995, pp. 193-241.

[TU95] Teniente, E.; Urpí, T. "A Common Framework for Classifying and Specifying Deductive Database Updating Problems", 11th ICDE, Taiwan, 1995, pp. 173-183.

[Ull88] Ullman, J.D. *Principles of Database and Knowledge-Base Systems*, Computer Science Press, New York, 1988.

[UO92] Urpí, T.; Olivé, A. "A Method for Change Computation in Deductive Databases", Proc. of the 18th Int. Conf. on Very Large Data Bases (VLDB), Vancouver, 1992, pp. 225-237.

[UO94] Urpí, T.; Olivé, A. "Semantic Change Computation Optimization in Active Databases", Proc. of the 4th Int. Workshop on Research Issues on Data Engineering- Active Database Systems (RIDE -ADS'94), Houston, USA, 1994, pp. 19-27.

[VLDB94] The VLDB Journal "Special issue on prototypes of deductive database systems". Vol. 3, Num. 2, April 1994.

[Wüt93] Wüthrich, B. "On Updates and Inconsistency Repairing in Knowledge Bases", *Int. Conf. on Data Engineering*, Vienna, 1993, pp. 608 - 615.