UN MODELO DE VERSIONES PARA APLICACIONES CAD E HIPERMEDIA

*Lic. Ana María García Pérez, Dra. Luisa González González**

RESUMEN

Este artículo trata un modelo de versiones para objetos complejos de datos, los cuales pueden ser objetos de diseño o grafos que representen aplicaciones hipermedia.

El modelo añade otra forma de trabajo para estas aplicaciones: el *modo conceptual,* ya que los usuarios pueden representar un modelo para su actividad, y además de esto, pueden trabajar con versiones del modelo y de los datos en el mismo ambiente de trabajo.

Palabras Clave:
Versiones, Evolución de esquemas, Bases de Datos Orientadas a Objeto, Objetos Complejos.

INTRODUCCION

Las primeras y más importantes aplicaciones de los Sistemas de Gestión de Bases de Datos se produjeron en áreas administrativas, para las cuales el modelo relacional es y seguirá siendo apropiado. Recientemente, han emergido nuevas aplicaciones intensivas de datos tales como: el CAD/CAM, CASE, CIM y las aplicaciones multimedia (como los sistemas de información geográfica). Para estas aplicaciones, se requieren modelos de datos que permitan representar, de la manera más natural y directa posible, no sólo la estructura de los objetos individuales, sino también las interrelaciones entre estos objetos. En estos ambientes, la estructura de los objetos, así como las interrelaciones entre ellos, cambian con el tiempo, de manera que el modelo que se escoja debe ser también capaz de expresar el comportamiento de los objetos.

En los sistemas tradicionales, por el contrario, los esquemas de una base de datos tienden a mantenerse fijos por largo tiempo, y además, al modificar datos, éstos se pierden.

Para poder dar satisfacción a los requerimientos impuestos por las nuevas aplicaciones, la investigación y el desarrollo en las bases de datos han seguido diferentes líneas o tendencias (no necesariamente divergentes), las que frecuentemente implican la integración de la tecnología de bases de datos con la tecnología de los lenguajes de programación (Bertino y Martino, 1993):

- Extender los sistemas relacionales
- Sistemas de Bases de Datos Orientadas a Objeto
- Sistemas de Bases de Datos Deductivas
- Sistemas de Bases de Datos Inteligentes

Aunque aún no posee un modelo de datos común ni están formalizados sus fundamentos, se reconoce que la tecnología orientada a objetos resulta ser la más promisoria para la próxima generación de sistemas de gestión de bases de datos (Deutsch, 1991).

Entre otros conceptos básicos del paradigma, las bases de datos orientadas a objeto tratan

* Docentes Investigadoras del Departamento de Computación, Universidad Central de Las Villas, Cuba.

el concepto de *objeto complejo de datos,* donde el valor de un atributo puede ser otro objeto o conjunto de objetos. Esta característica permite que puedan definirse objetos complejos arbitrarios en términos de otros objetos. Para ciertas aplicaciones, por ejemplo hipertextos, es importante poder describir el hecho de que un objeto es parte de otro.

Un Sistema de Gestión de Bases de Datos Orientado a Objetos (en inglés «O-O DBMS») debe permitir las siguientes características:

- Manipular versiones de objetos que reflejen la evolución de éstos
- Controlar la consistencia
- Recuperar objetos como un todo
- Manipular cambios en los esquemas
- Evaluar reglas y restricciones

Presentamos aquí nuestras consideraciones acerca de un modelo general de versiones que puede ser útil en el desarrollo de aplicaciones en dos áreas: *diseño ingenieril* y *diseño de hipertextos*.

CONCEPTOS BASICOS

Se han propuesto varios modelos para el manejo de versiones; un excelente resumen sobre los mismos aparece publicado en (Katz, 1990).

Sin embargo se conocen pocas implementaciones hoy en día. La más importante resulta ser ORION (Kim et. al, 1989), una interfaz para el sistema experto PROTEUS que implementa conceptos básicos sobre versiones, evolución dinámica de esquemas, objetos complejos y soporte a datos de multimedia.

También se reportan varios productos para el control de configuraciones de software (PC Week, Nov 7, 1994).

Usamos el término *manejo de versiones* para referirnos al conjunto de conceptos organizacionales y mecanismos de operación que se necesitan para organizar datos de diseño en agregados jerárquicos que cambian con el tiempo.

Un *modelo de versiones* proporciona extensiones semánticas a las facilidades primitivas de las bases de datos y está formado por una *organización física* y una *interfaz para realizar operaciones*.

Un *objeto de diseño* es una agregación de datos tratados como una unidad coherente por los diseñadores. Frecuentemente resulta ser un objeto complejo y a través del tiempo sus instancias significativas constituyen *versiones*. La modelación de versiones proporciona conceptos para organizar versiones individuales en *historias de versiones* (o sea, qué versión se deriva de cuál) y la composición de nuevas versiones del objeto complejo utilizando versiones de componentes (*configuraciones*).

Las operaciones relacionadas con las versiones incluyen la *herencia* (el mecanismo por el cual una nueva versión puede obtener datos de sus ancestros), *propagación del cambio* (que describe cómo la compleja estructura interconectada responde a cambios en los datos) y los *espacios de trabajo* (que proporcionan los mecanismos a través de los cuales se pueden hacer visibles las versiones a la comunidad de diseñadores).

UN MODELO DE VERSIONES PARA APLICACIONES CAD.

SIADI (González, et. al., 1994) es un sistema CAD desarrollado por la Universidad Central de Las Villas, Cuba.

Siadi usa un modelo de datos basado en el concepto de objeto complejo. Este trata al objeto de diseño como un patrón en el cual cada componente del objeto de diseño puede contener otro componente según una clase de membresía: obligatoria, alternativa u opcional.

Los componentes tienen atributos, y los valores de los atributos pueden obtenerse por medio de un método de cálculo, o seleccionando el valor en una tabla o de manera directa.

A continuación presentamos un ejemplo de un objeto de diseño (una trasmisión por poleas)

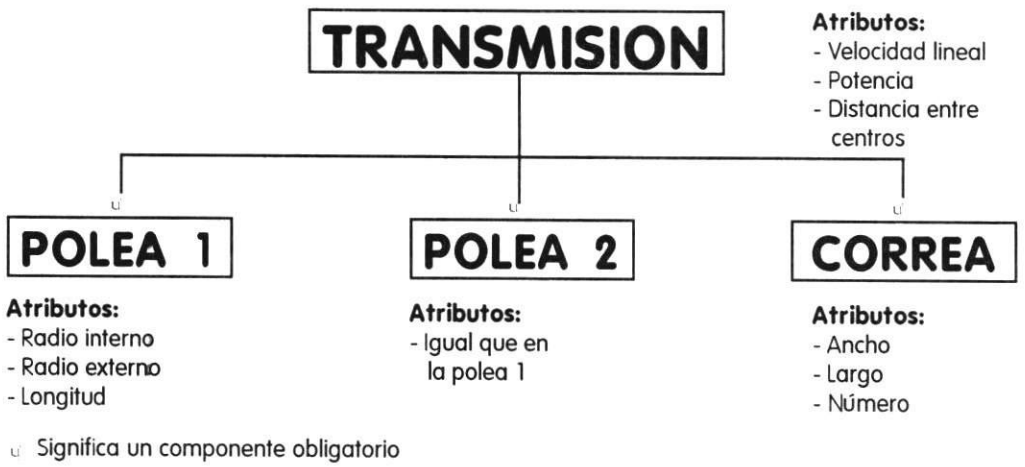

Fig. 1.- El objeto TRASMISION

Las versiones de la trasmisión pueden estar sujetas a restricciones definidas por el usuario, denominadas *restricciones de diseño*.

Otras restricciones (denominadas *restricciones implícitas*) se derivan de la semántica del patrón y se tratan de una forma transparente para los usuarios de manera que se pueda realizar la prueba de consistencia.

De esta forma, los diseñadores pueden usar un número pequeño de construcciones de modelación para hacer sus tareas.

La manipulación de las versiones en SIADI fue concebida originalmente como interacción con el usuario, que sigue un camino navegacional a través de los componentes del objeto de diseño valorizando atributos. Sin embargo una característica deseable es que se suministren dos tipos de acceso:

- el acceso a objetos simples por medio de la navegación
- el acceso a conjuntos de objetos por medio de un l*enguaje de solicitudes*.

SIADI fue escrito en C++, y su modelo de versiones se extendió para permitir no sólo la navegación sino el uso de un lenguaje declarativo tipo SQL para recuperar versiones.

UN MODELO GENERAL DE VERSIONES PARA APLICACIONES CAD E HIPERMEDIA

El objetivo de este modelo es permitir que se puedan manejar versiones de un objeto complejo de datos sin importar lo que éste represente. La representación lógica de una versión es dependiente de la aplicación. Sin embargo, la organización física y la interfaz son independientes del tipo de aplicación (CAD o hipermedia).

Nuestro modelo necesita la definición de tres aspectos:

(PATRON, VERSIONES, RESTRICCIONES)

donde:

- El PATRON representa un objeto complejo
- Las VERSIONES se consideran instancias del patrón y el usuario puede agregar atributos específicos de versiones que no estén presentes en el patrón, por ejemplo, fecha/hora de creación de la versión
- Las RESTRICCIONES no siempre se necesitan, pues dependen del problema objeto de la aplicación . En caso de existir, se usan para el control de la consistencia

Presentamos la interfaz principal de SIADI (ver fig. 2). A la izquierda se observa un menú de herramientas por medio de las cuales el usuario-diseñador puede definir la estructura de su objeto de diseño (patrón), en este caso un troquel de corte. Obsérvese que los componentes están interconectados por medio de la relación «parte de». Los componentes podrían representar partes mecánicas pero también pueden ser tópicos o «frames» de un hipertexto.
A la derecha, el sistema mantiene una historia de las versiones derivadas, no sólo de los datos, sino también de los esquemas de ellas. Con el mouse el usuario puede recuperar cualquier versión, aunque también se proporciona una interfaz parecida a la del lenguaje SQL para recuperar:

- Una versión con un número de versión específica
- La (o las) versiones que tienen algunas propiedades
- La ultima versión en una rama
- La última versión de todas.

Se proporcionan además botones para realizar las operaciones CUT y DEL. El primero posibilita desechar todas las versiones anteriores a una dada (la cual se convertiría en la «versión cero» u original). La segunda operación para permitir borrar una versión hoja, es decir, una versión sin descendientes.

La persistencia de las versiones se lleva a cabo manteniendo un conjunto de tablas para los valores de los atributos.

Solamente se guardan los valores que son diferentes entre la versión actual y su antecesora directa, a lo que se le denomina *delta entre versiones.*

Los identificadores de versiones se mantienen en tal forma que resulta posible definir para una versión, *la sucesora, antecesora y hermana.*

A nivel físico existe un archivo que registra los componentes que han sido eliminados en los esquemas.

A continuación se presentan dos algoritmos, escritos en SQL, el primero para recuperar una versión y el segundo para ejecutar un corte (CUT) en un identificador de versión.

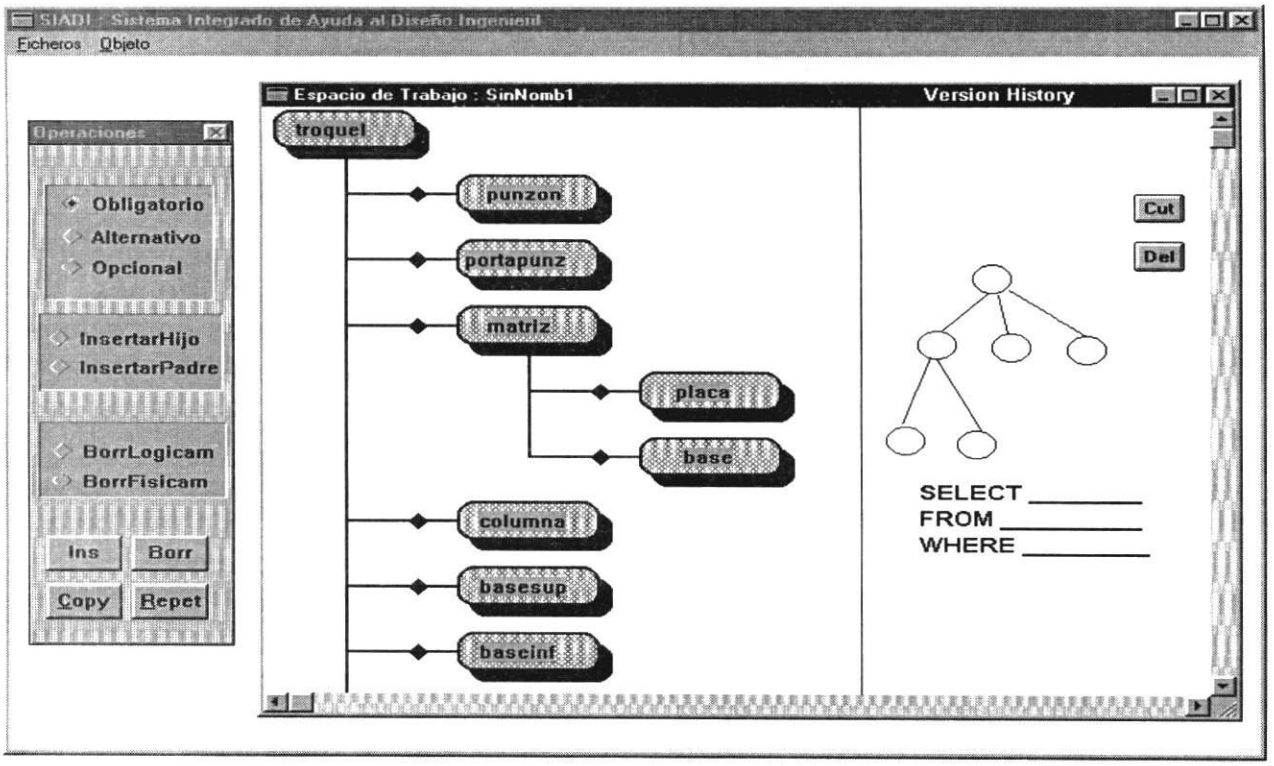

Fig. 2.- Interfaz de SIADI

ALGORITMO SQL PARA RECUPERAR UNA VERSION

Entradas necesarias:

1) La tabla genérica
   ATTRIBUTE (component-id, version-id,value)
2) La tabla
   DELETED (component-id, version-id)
3) La tabla
   PATTERN (component-id, nombre del componente, id del hijo, clase de membresía, version-id)
4) Current version = Id de la version a recuperar.

Paso 1.- «Iteración cero»

```
CV = Current version;
CREATE RET-VER
AS SELECT *
        FROM ATTRIBUTE
        WHERE version-id =:CV;
CREATE DEL-V
AS SELECT component-id
        FROM DELETED
        WHERE version-id=:CV;
```

Paso 2.- Iteraciones para añadir registros en la version recuperada.

```
WHILE CV <> 0 DO
BEGIN
        CV= ANCESTOR(CV);
        INSERT INTO RET-VER
        (SELECT * FROM ATTRIBUTE
        WHERE component-id NOT IN
(SELECT component-id FROM RET-VER)
        AND component-id NOT IN
DEL-V AND version-id =:CV);
        IF CV<> 0 THEN
                INSERT INTO DEL-V
                (SELECT component-id
                FROM DELETED
                WHERE version-id =:CV)
END;
```

ALGORITMO SQL PARA PRODUCIR UN CORTE EN EL ARBOL DE DERIVACION

Entradas necesarias:

Las mismas que en el algoritmo anterior, pero Current version = Id de la versión que se desea poner como versión «cero»

Paso 1.- Recuperar la versión que se desea poner como cero.

Paso 2.- Poner «cero» como id de versión en los artículos de esta versión recuperada.

```
UPDATE RET-VER
SET version-id = 0;
```

Paso 3.-En la tabla genérica original ATTRIBUTE, eliminar todos los registros con id de versión igual a cero y sus sucesoras y hermanas

```
DELETE FROM ATTRIBUTE
WHERE version-id NOT LIKE «current version %»
```

Nota: Id de versiones se crean de manera similar a la numeración de los capítulos y epígrafes de un libro (con notación decimal Dewey)

Paso 4.- En la tabla genérica original ATTRIBUTE eliminar (en todos los campos id de versión) la subcadena a la izquierda que tenga la misma longitud del identificador de la versión actual (current version)

```
UPDATE ATTRIBUTE
SET version-id = SUBSTR ( version-id,
length(current version) + 1)
```

Paso 5.- Añadir la versión recuperada a la tabla original.

```
INSERT
INTO ATTRIBUTE
(SELECT *
FROM RET-VER ).
```

CONCLUSIONES

En este artículo analizamos los requerimientos de nuevas aplicaciones intensivas de datos y la tecnología orientada a objetos como una vía para satisfacer estos requerimientos.

Utilizamos la orientación a objeto para la representación de los esquemas.

Mencionamos algunos conceptos básicos que sustentan la modelación de versiones de objetos complejos de datos.

Explicamos un modelo implementado en SIADI, un sistema de ayuda al diseño ingenieril.

Por último, mostramos los principios de un modelo general de versiones en un ambiente que necesite la modelación conceptual (usamos el término «patrón» para el objeto complejo) además de trabajo en modo-instancia.

Demostramos la factibilidad de este modelo describiendo algoritmos en SQL que se ejecutan sobre tablas que registran valores-delta entre versiones.

Por medio de este modelo podemos implementar tanto la evolución de esquemas como de instancias que se deriven de la modificación en el tiempo de un objeto complejo.

BIBLIOGRAFIA

ANDANY, J., M. LEONARD, C. PALISSER (1991) Management of schema evolution in Databases. In Proc. 17th. International Conference on VLDB. Barcelona. Spain. Lohman, G.M., Sernadas, A. and Camps, R. (eds) Morgan Kaufmann, San Mateo, CA. 161-170.

BERTINO, E. y L. MARTINO (1993) Object-Oriented Database Systems: concepts and architectures. Addison-Wesley Publishing Co.

BORHANY, M., J. P. BARTHES, P. ANOTA, F. GAILLARD (1993) A synthesis of the versioning problems in object-oriented engineering systems. Proceedings of the IFIP, Compiegne.

COMMAFORD, C. (1994) Version control is not optional, it's required. PC Week, Nov. 7, 1994., 11(44).

DATE, C.J. (1993) Introduction to Database Systems, Vol. 1 5th. ed. Addison-Wesley.

DATE, C.J. (1994) Oh relational: toward an OO/Relational Rapprochment. Database Programming & Design. Oct. 1994, 7(10) 23-27.

DEUTSCH, P.L. (1991) Object-Oriented Software Technology. IEEE Computer 24(9), 112-113.

GARCIA, A. (1995) Diseño de una solución para el manejo de versiones en SIADI. Reporte Técnico, UCLV, Cuba.

GONZALEZ, L. (1994) SIADI: Sistema Integrado de Ayuda al Diseño. Tesis en Opción al grado de Doctor en Ciencias Técnicas. Universidad Central de Las Villas, Cuba.

HUGHES, J.G. (1991) Object-Oriented Databases. Prentice-Hall. C.A.R. Hoare series ed.

KATZ, R. (1990) Toward a unified framework for version modelling in engineering databases. ACM Computing Surveys, 22(4), 375-408.

KIM, W., N. BALLOU, H. T. CHOU, J. GARZA y D. WOELK (1989) Features of the ORION object-oriented database system. In Object-Orient Concepts, Databases and Applications. (Kim,W. and Lochovsky, F., eds.) pp 251-282. Reading, MA: Addison-Wesley.

KOSHAFIAN, S., M. J. FRANKLIN y M. J. CAREY (1990) Storage management for persistent complex objects. Information Systems 15(3) 303-320.

PEHONG, CHEN y M. A. HARRISON (1988) Multiple representation document development. IEEE Computer. January 1988, 15-31.

STRAUBE, D. y M. ÖZSU (1990) Queries and query processing in O-O database systems. ACM Transactions on Information Systems. 8 (4) 387-430.