MODELACION DE VERSIONES DE OBJETOS COMPLEJOS DE DATOS

Lic. Ana María García Pérez, Dra. Luisa González González,
Lic. Abel Rodríguez Morffi, Ing. Ionel Muñoz Jiménez
Universidad Central de Las Villas

RESUMEN

Los requerimientos impuestos por las aplicaciones CAD/CAM, CASE, multimedia, etc. hacen aconsejable el diseño de sistemas que permitan modelar estas clases de problemas de una manera más natural que con los sistemas actuales para procesamiento de datos.

En este sentido existen diversas tendencias de investigación, entre las que podemos citar el desarrollo de nuevos modelos de datos y arquitecturas que den soporte a estos objetivos.

En particular, se usan frecuentemente los modelos basados en objetos complejos de datos, aún sin madurez suficiente. La mayoría de los sistemas actualmente disponibles resultan ser extensiones de los lenguajes de programación orientados a objetos para permitir la persistencia de estos objetos complejos.

En este trabajo se aborda:

1. Un sistema de ayuda al diseño ingenieril que implementa un modelo de datos que permite representar y manipular patrones de objetos de diseño (objetos complejos de datos) y un modelo de versiones que permite representar y manipular versiones de un patrón.

2. La generalización del modelo de versiones anterior, lo cual es útil para el desarrollo de sistemas donde el problema por resolver se adapte al uso de *patrones* y *versiones* de estos patrones, sujetos a un grupo de *restricciones*.

PALABRAS CLAVES:

Bases de datos orientadas a objetos
Versiones
Reglas y restricciones en bases de datos
Optimización de solicitudes

SUMARIO:

I. INTRODUCCION

Un *modelo de datos* [1] proporciona una notación para reflejar:

1. Propiedades estáticas de los datos (entidades, atributos, interrelaciones entre entidades).
2. Propiedades dinámicas: Operaciones permitidas sobre los datos
3. Reglas de integridad: Restricciones que reflejan un estado válido para los datos.

Un *Sistema de Gestión de Bases de Datos* permite describir el esquema conceptual de una base de datos en términos de un modelo de datos [2]

La mayoría de los Sistemas de Gestión de Bases de Datos comercialmente disponibles se basan en el modelo relacional de datos y por lo tanto se denominan "sistemas relacionales". El modelo relacional de datos se basa en el concepto matemático de relación y en el álgebra relacional o cálculo relacional para expresar las operaciones sobre relaciones.

Las primeras y más importantes aplicaciones de los sistemas de gestión de bases de datos se produjeron en áreas de gestión y administrativas, para las cuales el modelo relacional es apropiado. En la actualidad han emergido nuevas aplicaciones intensivas de datos, tales como el CAD/CAM y el CASE, CIM y los sistemas multimedia (como sistemas de información geográfica). Para éstas se requieren modelos de datos que expresen

de la forma más natural y directa posible tanto la estructura de los objetos individuales como las relaciones que existen entre los diferentes objetos. En estos ambientes de aplicaciones la estructura de los objetos, así como las relaciones entre ellos, cambia con el tiempo, de modo que el modelo también debe poseer capacidad de expresar el comportamiento de los objetos y las restricciones que éstos deben satisfacer.

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

Un ejemplo que ilustra la desventaja del modelo relacional para representar objetos compuestos, es el siguiente: Para representar un objeto de diseño y sus partes componentes, necesitamos asociar un esquema con cada componente del objeto, ya que tienen atributos diferentes. Para poder recuperar la información que permita evaluar el objeto como un todo, necesitamos ejecutar acoples mediante claves foráneas, operación que resulta más lenta a medida que la cantidad de componentes por manejar se incrementa:

TRASMISION (Polea 1_id, Polea 2_id, veloc_max, tipo_seccion, dist_centros, torque, relac_tras, N° version)

Polea 1 (Polea1_id, diam_ext, N° ranuras, diam_int, ancho, veloc_lineal)

Polea 2 (Polea 2_id, ...)

CORREAS (Correa_id, Longitud, área, veloc_límite, ancho, tensión, cant)†

Claves foráneas:

Para aceptar los requerimientos impuestos por las nuevas aplicaciones, las investigaciones en bases de datos siguen diferentes líneas o tendencias (no necesariamente divergentes) orientadas a incrementar la potencia de expresión (captar más semántica) de los modelos de datos:

1. Extender sistemas relacionales para permitir reflejar relaciones con anidación (objetos de naturaleza compuesta).
2. Sistemas de gestión de bases de datos orientados a objeto (OO-DBMS).
3. Sistemas de gestión de bases de datos deductivas (Deductive Databases).
4. Sistemas de gestión de bases de datos Inteligentes.

Aunque todavía no poseen un modelo de datos común ni están formalizados sus fundamentos (como sí lo están en el modelo relacional) se reconocen los OO-DBMS como la tecnología más promisoria para la próxima generación de Sistemas de Bases de Datos [3].

Entre otros conceptos básicos del paradigma orientado a objetos, las bases de datos O-O incorporan el concepto de *Objeto Complejo*, donde el valor de un atributo puede ser un objeto o conjunto de objetos. Esta jerarquía permite que objetos complejos arbitrarios puedan definirse en términos de otros objetos.

Para ciertas aplicaciones, hipertextos por ejemplo, es importante describir el hecho de que un objeto es parte de otro objeto. El concepto de objeto complejo o compuesto ha sido introducido tanto en algunos OO-DBMS, p.ej. ORION (Kim, et. al. 1987) y algunos lenguajes de programación, para permitir que algunas aplicaciones modelen el hecho de que varios objetos (conocidos como objetos componentes) constituyan una entidad lógica.

Un DBMS-OO debe contener mecanismos para:

1. Manejar versiones de objetos que reflejen los diferentes estados de evolución de éstos.
2. Controlar la consistencia.
3. Recuperar objetos como un todo.
4. Manejar cambios en los esquemas de los objetos complejos (debido a la naturaleza evolutiva de las aplicaciones).
5. Evaluar reglas y restricciones.

No obstante, se señalan algunas críticas al modelo O-O, entre las que podemos citar:

- La forma navegacional de cálculo representa una «vuelta atrás» a los tiempos de las bases de datos jerárquicas y de redes
- El modelo no está basado en una teoría matemática coherente
- No ofrece características estándar de los sistemas relacionales tales como las solicitudes declarativas y otras.

II. CONCEPTOS BASICOS DE UN PROBLEMA DE VERSIONES.

Usamos el término *manejo de versiones* para referirnos al conjunto de conceptos organizativos y mecanismos operacionales usados para organizar datos de diseño ingenieril en agregados jerárquicos que cambian con el tiempo.

Un *modelo de versiones* se define sobre organizaciones de datos existentes para proveer extensiones semánticas que permitan soportar organizaciones de datos de ingeniería basadas en facilidades más primitivas de bases de datos.

Un *objeto de diseño* es una agregación de datos de diseño tratados como una unidad coherente por los diseñadores.. A través del tiempo, sus instantáneas significativas forman *versiones*.

La modelación de versiones proporciona conceptos de estructuración de datos para organizar versiones individuales en *historias de versiones* (qué versión se deriva de cuál), componiendo objetos compuestos a partir de versiones de sus componentes (*configuraciones*), y verificando *versiones equivalentes* entre representaciones.

Operaciones relacionadas con las versiones son:

- La *herencia*, mecanismo por el cual una nueva versión puede obtener datos de sus ancestros.
- La *propagación de cambios* que describe cómo la estructura interconectada, responde a los cambios en los datos
- Los *espacios de trabajo* que proporcionan los mecanismos a través de los cuales nuevas versiones pueden hacerse visibles para la comunidad de diseñadores.

Han sido propuestos varios modelos para el manejo de versiones; un excelente panorama de éstos aparece en el artículo de (Katz, 1990).
Sin embargo se conocen pocas implementaciones hoy en día. La más importante resulta ser ORION [3], una interfaz para el sistema experto PROTEUS que implementa los conceptos básicos sobre versiones, evolución de esquemas, objetos compuestos y soporte a datos de multimedia. También se reportan diversos productos para el control de versiones de software (*version control software*) [9] como herramienta de ayuda a la gestión de configuraciones de software.

3. UN MODELO DE VERSIONES PARA APLICACIONES CAD

SIADI [14] es un sistema de ayuda al diseño ingenieril desarrollado en la Universidad Central de Las Villas.

El modelo de datos que sustenta a SIADI consta de una notación para representar:

1. Un objeto de diseño como una jerarquía de composición de objetos (representa un objeto complejo).
2. Restricciones para la obtención de versiones del objeto de diseño.

Los objetos del modelo de datos son agregaciones de información que tienen un nombre, pues tienen su contrapartida en el mundo real, y tienen propiedades o atributos . La asignación de valores a los atributos puede ser mediante la evaluación del método de cálculo, en caso de que lo tenga definido, o de forma explícita mediante otros mecanismos.

Un atributo también puede ser valorizado mediante la obtención de su valor en una tabla, situación muy frecuente en aplicaciones ingenieriles. Esto justifica el desarrollo en el sistema de una interfaz a una base de datos relacional que facilite este tipo de operación.

Los atributos de un objeto pueden reutilizar la definición de atributos de otros objetos o de alguna tabla definida como parte de la base de datos relacional. Esto constituye una herencia de representación.

Cada componente del objeto de diseño puede ser considerada imprescindible u *obligatoria* en la obtención de una instancia, o puede ser considerada *opcional*; de igual forma hay componentes para las que no existe una sola variante y se pueden presentar de manera *alternativa*. De forma recursiva cada componente del objeto de diseño es en sí también un objeto y se puede definir de manera similar.

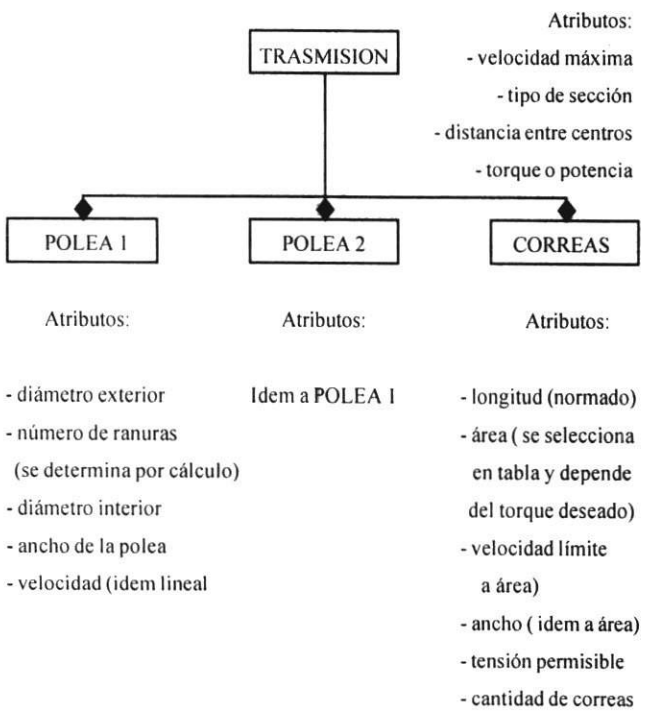
Restricciones de integridad: En adición a los tipos convencionales de restricción de integridad, el modelo tiene un grupo de restricciones implícitas que permiten

obtener instancias consistentes con los requerimientos del modelo y un grupo de restricciones explícitas de gran utilidad en aplicaciones ingenieriles que reflejan interrelaciones válidas entre objetos y que en general se conocen como *restricciones de diseño.*

Estos tipos de restricciones pueden definirse por el usuario como expresiones relacionales y reglas del tipo IF expresión1 THEN expresión 2.

Un ejemplo de «patrón de objeto de diseño» y sus restricciones, tal como quedan formulados en SIADI, se presenta a continuación para un mecanismo de

TRASMISION POR CORREAS:



Los rectángulos representan las piezas (objetos o entidades)

Las líneas representan la relación "es parte de", que relaciona objetos de más alto nivel con otros de nivel inferior.

Esta representa una conexión opcional.

Esta representa una conexión obligatoria.

Esta representa una conexión alternativa.

RESTRICCIONES DE DISEÑO:

1. Diámetros de la POLEA 2 mayores a los de la POLEA 1
2. Velocidad lineal menor que la velocidad máxima
3. Número de ranuras igual que número de correas.

Las restricciones sobre el objeto TRANSMISION POR CORREAS se escriben:

1) POLEA2. diam-ext > POLEA1. diam-ext & POLEA2. diam-int > POLEA1. diam-int
2) POLEA1.veloc-lineal < VMAX & POLEA 2.veloc-lineal < VMAX
3) POLEA1.num-ranuras = CORREAS.cantidad & POLEA2.num-ranuras = CORREAS.cantidad

SIADI cuenta con interfaces para:

1. Definir TABLAS, que constituyen para el diseñador normas para el trabajo:
2. Definir OBJETOS DE DISEÑO que constituyen el patrón del objeto a diseñar en forma de una jerarquía de composición de objetos, además de las restricciones de diseño.
3. Manipular VERSIONES del objeto de diseño, que constituyen instancias del patrón creado.

La opción Versiones trata las abstracciones de las instancias del objeto complejo con facilidades para:
- Crear una versión.
- Mostrar una versión.
- Componer una versión, a partir de versiones de componentes ya creadas.
- Operaciones de evaluación sobre versiones.

Hasta el momento se encuentran definidos un conjunto de funciones para la evaluación de versiones consistentes ya creadas, que de manera general se expresan como sigue:

Valor: [A, O, V] → R
V: Conjunto de versiones calculadas.
O: Conjunto de los objetos del patrón.
A: Conjunto de los atributos del patrón.

Valor(Ai, Oj, Vk) devuelve el valor del atributo Ai del objeto Oj en la versión Vk.

Las funciones operan sobre una o varias versiones, y este conjunto de funciones puede ampliarse en sucesivas ediciones de SIADI.

El trabajo con versiones en SIADI fue originalmente concebido para la interacción [15], siguiendo un mecanismo navegacional por los objetos componentes del patrón.

Sin embargo, se considera que una característica deseable es que los sistemas deben soportar los dos tipos de acceso:

- acceso a objetos simples por medio de la navegación.
- acceso a conjuntos de objetos por medio de query languages.

IV. GENERALIZACION DEL MODELO DE VERSIONES PARA OTRAS APLICACIONES.

Pretendemos a continuación exponer el diseño de un modelo de versiones que consiste en una **organización física** que permite la **representación** de las versiones y una **interfaz** para especificar las **operaciones** sobre las mismas.

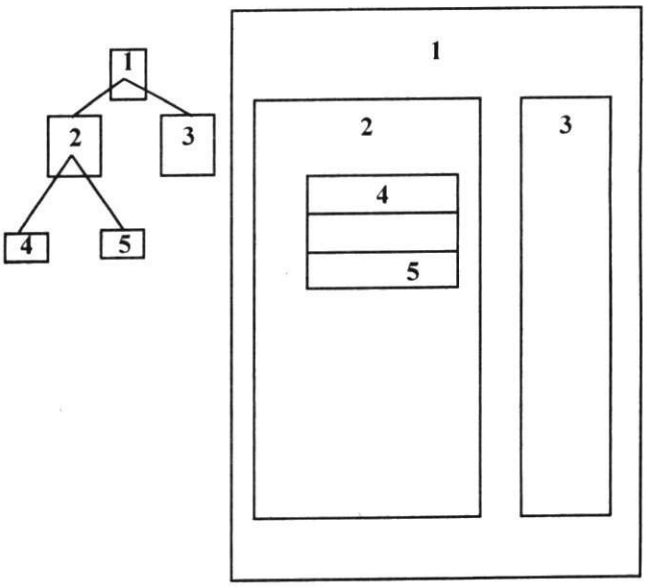
Se hace uso del concepto de «patrón» que se viene usando en modelación computacional de problemas [4].

La interfaz está orientada tanto a la manipulación directa (navegación) como al uso de un lenguaje declarativo (para el acceso a conjuntos de objetos). Para este último se ha escogido un dialecto de SQL, tomando en cuenta que su se uso está expandiendo a la tecnología O-O [8] El modelo distingue dos formas de representación para las versiones:

- la representación física
- la representación lógica

La representación lógica es dependiente de la aplicación. Para una aplicación CAD una versión «lógica» puede ser un simple listado con las propiedades de los componentes valorizadas; también pudiera ser un plano donde las componentes se dibujan en una ubicación dada. Por ejemplo, dado el objeto 1 cuyas componentes son 2, 3, 4 y 5:

Una versión del objeto 1 puede ser el plano_



(Otras versiones se originan al cambiar de ubicación las componentes).

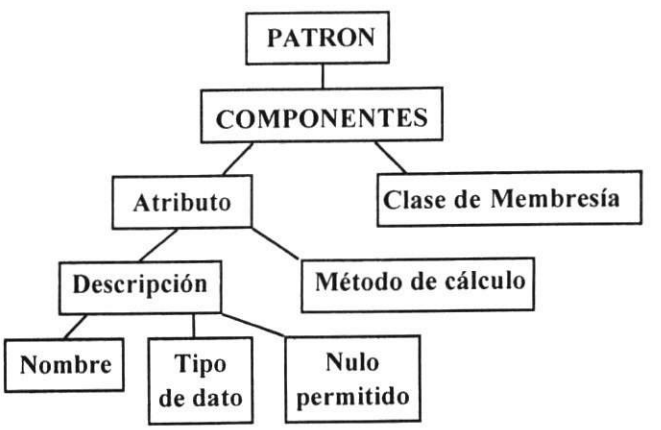
Sin embargo, la organización física y el lenguaje (por medio de los cuales nos referimos a la representación física) son independientes de la aplicación, capaces de permitir modelar las versiones, es decir, representarlas y operar con ellas.

«Ver» una versión del objeto complejo significa trasladarla de su representación física a una representación lógica adecuada a la aplicación.

El modelo necesita como entrada una tríada:

(PATRON, VERSIONES, RESTRICCIONES) , donde:

- El PATRON representa un objeto complejo de datos cuya composición ilustramos con la notación de diccionario de datos:



- VERSIONES, a las cuales se le definen opcionalmente ATRIBUTOS GLOBALES y ATRIBUTOS LOCALES (éstos últimos se añaden a los ya definidos en el patrón para las componentes). Un atributo global se refiere a una instancia del objeto de diseño, un atributo local a una instancia de componente.

- RESTRICCIONES
  Implícitas del patrón (relativas a la semántica de los tipos de membresía permitidos para las componentes)
  Explícitas (relativas a la validez semántica de las versiones).

Enfatizamos que una versión (instancia de un objeto complejo de datos) se considera como un Tipo de Datos Abstracto (ADT), al que se le pueden asociar atributos específicos debido a las necesidades que en la práctica reclaman las aplicaciones.

Ejemplos de atributos de versiones:

Globales:
1) Fecha-hora de creación de la versión
2) Identificación de la versión
3) Identificadores de las versiones de las cuales ésta se deriva

Locales: (de componentes)
1) Ubicación en el plano
2) Font que se debe utilizar para la visualización

Queda como responsabilidad del diseñador que tanto en el PATRON como en las VERSIONES se definan los atributos que hagan luego posible una representación lógica de acuerdo con la aplicación.

Programar una aplicación consistirá en:
1. Definir un modelo de PATRON.
2. Definir las restricciones implícitas del modelo de patrón.
3. Programar los mecanismos para obtener la representación lógica de las versiones.

Una vez que se construya la aplicación, los diseñadores en una rama específica dispondrán de un sistema con interfaces para:

1. Definir su patrón específico (p. ej. qué pieza va a diseñar).
2. Definir las restricciones de diseño (restricciones explícitas).
3. Definir las propiedades de las versiones.
4. Operar con los patrones y con las versiones, éstas últimas por manipulación directa en los componentes o a través del lenguaje declarativo.

El lenguaje declarativo permitirá la formulación de solicitudes sobre versiones, con facilidades para crear nuevas versiones a partir de versiones-fuente, por medio de:

- **Proyecciones** (obtención de Componentes).
- **Selecciones** (las versiones con los componentes que satisfagan ciertas condiciones).
- **Composiciones** (esto es, la obtención de versiones que mezclan versiones de componentes o configuraciones).

Se considera como versión fuente por defecto el propio patrón.

Para optimizar estas solicitudes se definen «aceleradores» (el uso de aceleradores aparece referido en [5],[13]) que no son más que índices especiales.

La sistematización de este modelo de versiones permite:

1. La adición de un formalismo de operadores para el tratamiento de la transformación clases-instancias, al tomar en cuenta lo señalado por Date en [7] sobre el acercamiento entre la teoría relacional y la orientada a objetos.
2. La obtención de una biblioteca de clases reutilizables para el desarrollo de aplicaciones donde se requiera modelar problemas con el uso de objetos complejos de datos, versiones y restricciones.

V. CONCLUSIONES

En este trabajo se analizan los requerimientos de nuevas aplicaciones intensivas de datos y la tecnología orientada a objetos como una vía para la producción de sistemas que hagan posible la satisfacción de estos requerimientos.

Se relacionan los conceptos básicos asociados con el problema del versionamiento de objetos complejos de datos, se ejemplifica con un sistema de ayuda al diseño ingenieril (SIADI) que se sustenta en un modelo de datos semántico con orientación a objetos y un modelo de versiones orientado a la interacción navegacional.

Por último se exponen los principios de un modelo general de versiones que posibilitará enfrentar el desarrollo de aplicaciones similares a SIADI, entre las que se encuentra una aplicación para el manejo de hipertextos que se acomete en nuestro grupo de trabajo.

VI. REFERENCIAS:

[1] Batini, C., Ceri, S. and Navathe, S. (1991). Conceptual Database Design. Benjamin/Cummings.

[2] Ullman, J. (1989). Principles of Database and Knowledge-Based Systems, Vol. 1. Rockville MD: Computer Science Press.

[3] Bertino, E. and Martino, L. Object-Oriented Database Systems: concepts and architectures. (1993). Addison-Wesley Publishing Co.

[4] James O. Coplien (1994). Software design: the emerging patterns. C++ REPORT, July-August, 1994. Vol. 6 No. 6 pp 18-22;66-67

[5] S. Koshafian, M.J. Franklin. M.J. Carey (1990). Storage Management for Persistent Complex Objects. INFORMATION SYSTEMS, Vol. 15 No. 3 pp. 303-320.

[6] M. Borhany, J.P. Barthes. P. Anota, F. Gaillard (1993). A synthesis of the versioning problems in object-oriented engineering systems. In Proceedings of the IFIP, Compiegne, 1993.

[7] C.J. Date (1994). Oh, Oh relational: toward an O-O/Relational rapprochement. Database Programming & Design. Oct. 1994 Vol 7 No. 10 pp 23-27

[8] Joe Celko (1994). SQL in the city of steel. DBMS, July 1994, Vol. 7 No. 8 pp 17-21

[9] C. Comaford (1994). Version Control is not optional, it's required. PC WEEK. Nov. 7, 1994 Vol 11 No. 44 p. 24

[10] Pehong Chen, Michael A. Harrison (1988). Multiple representation document development. IEEE COMPUTER. January 1988 pp 15-31

[11] C.J. Date (1993). Introducción a los Sistemas de Bases de Datos Vol I 5a. ed, Addison-Wesley Iberoamericana, 1993.

[12] John G. Hughes (1991). Object-Oriented Databases. Prentice-Hall International, C.A.R. Hoare series ed. 1991

[13] Dave S. Straube, M. Tamer Ozsu (1990). Queries and query processing in O-O Database Systems. ACM TRANSACTIONS ON INFORMATION SYSTEMS. Vol. 8 No. 4 Oct. 1990 pp 387-430

[14] Luisa M. González (1994). SIADI: Sistema Integrado de Ayuda al Diseño. Tesis en Opción al Grado de Doctor en Ciencias Técnicas. Universidad Central de Las Villas.

[15] Ana M. García Pérez (1995). Diseño de una solución para el manejo de versiones del sistema SIADI. UCLV.

[16] Randy H. Katz (1990). Toward a unified framework for version modelling in engineering databases. ACM COMPUTING SURVEYS, Vol. 22 No. 4 Dec. 1990, pp. 375-408.