EVOLUCIÓN Y PERSPECTIVAS FUTURAS SOBRE BDOO

Saldaño, Viviana [1]

División Tecnología
Unidad Académica Caleta Olivia
Universidad Nacional de la Patagonia Austral [2]

RESUMEN

En este trabajo se describen los orígenes y evolución de las Bases de Datos de Objetos. Se detallan los distintos enfoques por los que a lo largo de estos años, los investigadores han llegado a la conclusión de querer construir una Base de Datos de Objetos.
Asimismo, se analiza la funcionalidad mínima que debe presentar cualquier producto que se considere OODBMS. Por lo tanto, este artículo brinda un panorama del estado del arte en cuanto a tecnologías de OODBMS.
Para concluir, se tomará la documentación técnica de un producto OODBMS en particular, y se analizarán las funcionalidades provistas por el mismo.

PALABRAS CLAVE:
OODBMS, ORDBMS, DBMS.

[1] e-mail: VIALE@SATLINK.COM, JTP con Dedicación Exclusiva - Universidad Nacional de la Patagonia Austral. Analista Programadora Universitaria. Alumna avanzada de la carrera Ingeniería en Sistemas. Universidad Nacional de la Patagonia Austral.
[2] Acceso Norte, Ruta 3. (9011) Caleta Olivia. Provincia de Santa Cruz. Tel. (0297) 4854888.

INTRODUCCIÓN

Los tiempos de la tecnología informática, enfrentan un proceso de cambio continuo. Surgen cotidianamente nuevos términos tales como CORBA, nuevos lenguajes como Java y nuevos enfoques como desarrollo basado en componentes. Si la evolución se da de esta manera, ¿por qué entonces al seleccionar una base de datos, elegir una tecnología con más de 20 años en el mercado?

Las bases de datos de objetos aparecen a mediados de los 80'. El objetivo entonces, era proveer una nueva clase de bases de datos, diseñada y optimizada para almacenar y administrar objetos, dada la amplia repercusión de los lenguajes y técnicas de modelado orientado a objetos, de ese entonces.

La principal ventaja de una base de datos de objetos es la naturalidad con que maneja modelos complejos con relaciones complejas. El manejo de objetos en cuanto a estructuras complejas es fundamental, pero lo más importante es la posibilidad de manejar relaciones, no solamente del tipo uno a uno ó uno a muchos, sino relaciones que incluyen semántica, tales como conjuntos (unicidad), listas (ordenamiento), diccionarios (búsqueda asociativa), etc..

Organizaciones de todos los tamaños, están paulatinamente incrementando la utilización de bases de datos orientadas a objetos para atender la creciente complejidad de los requerimientos de las aplicaciones. En algunos casos, los requerimientos pertenecen a aplicaciones de dominios específicos, tales como:

- Computer-Aided Design (CAD)
- Computer-Aided Manufacturing (CAM)
- Geographic Information Systems (GIS)
- Computer-Aided Software Engineering (CASE)
- Automatización de oficinas
- Aplicaciones multimediales

y en otros casos los requerimientos involucran aplicaciones comerciales más convencionales con necesidades específicas, tales como:

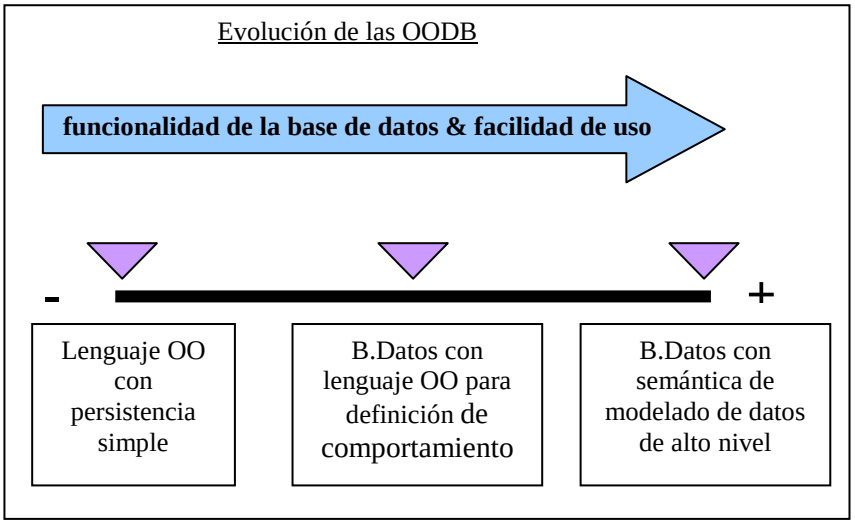
- Sistema de Mantenimiento Externo de Planta, con manejo de información referente a ubicación geográfica de los equipos, necesidades gráficas de información.
- Sistema de Reclamo de Seguros, con necesidad de almacenamiento de las fotografías asociadas a la propiedad dañada, que es objeto del reclamo.
- Sistema de Manufactura de aviones, requiriendo el control de millones de partes relacionadas que pueden ser ensambladas bajo diferentes configuraciones.

La tendencia actual marca que las aplicaciones que se caracterizan por el manejo de información compleja, altamente relacionada, se inclinan a utilizar bases de datos orientadas a objetos. Esto sucede dado que la tecnología de bases de datos relacionales no satisface las necesidades de los sistemas complejos de información.

El problema con los sistemas de bases de datos relacionales, es que requieren que el desarrollador de la aplicación, transforme el modelo de la aplicación a tablas, donde las relaciones entre entidades se definen por valores. Al comparar las bases de datos relacionales con las orientadas a objetos, Mary Loomis, arquitecta del OODBMS de Versant dice: -"*El diseño de bases de datos relacionales es en realidad un proceso en el que se trata de resolver cómo representar objetos del mundo real dentro de los confines de las tablas, de tal manera que se obtenga buena performance y se preserve la integridad de los datos. En cambio, el diseño en las bases de datos orientadas a objetos, es totalmente diferente. El diseño de las bases de datos orientadas a objetos es una parte fundamental del proceso total de diseño de la aplicación. Las clases de objetos usadas por el lenguaje de programación son las clases usadas por el OODBMS. Dado que los modelos son consistentes, no hay necesidad de transformar el modelo de objetos del programa en algo específico para el administrador de bases de datos-*".

1. Evolución de las Bases de Datos Orientadas a Objetos.

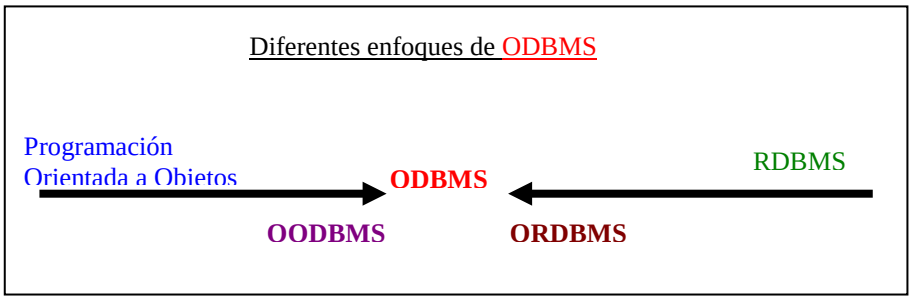
Las bases de datos orientadas a objetos están siguiendo una maduración similar a la de las bases de datos relacionales. Al comienzo, tenemos lenguajes orientados a objetos que fueron extendidos para brindar persistencia simple, permitiendo a los objetos de la aplicación persistir entre distintas sesiones de usuario. En este punto se provee mínimamente la funcionalidad de una base de datos, en términos de control de concurrencia, transacciones, recuperación, etc. Una fase más adelante, se logra soporte para muchas de las propiedades de las bases de datos, y las bases de datos de esta fase son suficientes para desarrollar aplicaciones de administración de datos razonablemente complejas. Los OODBMS actuales, se encuentran en este punto de la evolución, contando con semántica declarativa tal como restricciones, reglas de integridad referencial y seguridad. En la mayoría de los OODBMS, la mayor parte de la semántica de la base de datos es definida por los programadores que utilizan los servicios de bajo nivel provistos por la base de datos. La próxima fase de la evolución es más difícil. A medida que se tiende a este punto de la evolución, la base de datos hace más por el usuario, requiriendo menor esfuerzo para el desarrollo de aplicaciones. Una manera de visualizar la madurez de la base de datos es verificar hasta qué punto las operaciones tales como optimización, reglas de integridad, migración, resguardo y recuperación pueden ser ajustadas por el usuario usando comandos declarativos de alto nivel del OODBMS. Hoy en día, la mayoría de los productos de OODBMS, requieren que el desarrollador escriba código para realizar estas operaciones.



2. Distintos enfoques para Implementar Bases de Datos de Objetos.

Las primeras bases de datos de objetos, se diseñaron desde cero, sin ataduras al modelo relacional; basándose principalmente en los lenguajes de programación orientados a objetos, agregándoles características de bases de datos.

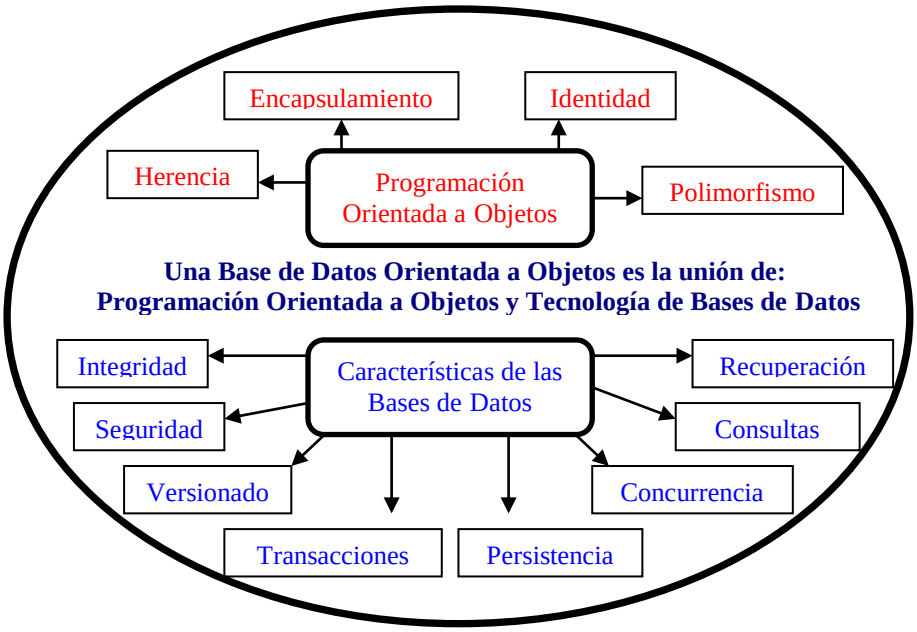
Otra corriente, posterior, tomó los fundamentos del modelo relacional, agregándole características de objetos. A los productos de la primera corriente se los denomina Bases de datos Orientadas a Objetos ú OODBMS, y a los de la última corriente se los llamó Bases de Datos Objeto Relacionales ú ORDBMS. Cuando se hace referencia a cualquiera de ellos se utiliza el nombre Bases de Datos de Objetos ú ODBMS.



Productos de Administración de Bases de Datos por Empresa			
Organización	**RDBMS**	**ORDBMS**	**OODBMS**
Oracle	Oracle 7.x	Oracle 8.x	
Sybase	System 10/11		
Informix	Dynamic Server	Universal Server (Illustra)	
IBM	DB/2	Universal Database (DB/2 Extenders)	
UniSQL		UniSQL/X	
Unisys		OSMOS	
Computer Associates	OpenIngres		Jasmine
Gemstone			Gemstone
O2			O2
Object Design			Object Store
Objectivity			Objectivity/DB
Versant			Versant ODBMS

3. Características de las Bases de Datos Orientadas a Objetos

La tecnología de bases de datos orientadas a objetos es la unión entre la programación orientada a objetos y las tecnologías de bases de datos. Quizás la característica más interesante de la tecnología de base de datos orientadas a objetos es que combina programación orientada a objetos con tecnología de bases de datos para proveer un sistema de desarrollo de aplicaciones integral. Un ODBMS provee integración de datos, control total y conceptos de objetos tales como herencia,



encapsulamiento y además las ventajas de un DBMS, como manejo de transacciones, control de concurrencia, etc.. La idea de usar un ODBMS es representar una entidad u objeto en el dominio de la aplicación modelándolo con un objeto correspondiente en la base de datos. Esto significa modelar el comportamiento de cada objeto como también la estructura del objeto y sus relaciones con los otros objetos definidos. Este mapeo uno a uno reduce el hueco semántico ó "impedance mismatch" entre el dominio de la aplicación y el modelado del dominio de esa base de datos. Además, cuando se acopla a programas orientados a objetos, se reduce la separación entre esos programas y sus datos en la base de datos. Entre las características mínimas que debe soportar un ODBMS se encuentran:

- Objetos Complejos e Identidad de Objetos
- Extensibilidad mediante uso de tipos abstractos o clases
- Jerarquías de Clase o Tipo
- Redefinición y Binding tardío.
- Persistencia y Administración de Almacenamiento Secundario.
- Concurrencia y Recuperación
- Manejo de Consultas

4. CRITERIOS DE FUNCIONALIDAD QUE DEBERÍA PROVEER UN OODBMS.

Se analizarán a continuación las capacidades funcionales que debería proveer un OODBMS. Entre las mismas se considerarán:

- Modelado básico Orientado a Objetos
- Tópicos avanzados de Bases de Datos Orientadas a Objetos
- Arquitectura de Bases de Datos
- Funcionalidad de Bases de Datos
- Interfaz de Programación de la Aplicación
- Capacidad para consultas.

4.1. Modelado básico Orientado a Objetos

Los tópicos de esta sección, cubren las características básicas del paradigma OO, típicamente soportado en cualquier tecnología OO. Estas características básicas, se espera que sean soportadas en todos los OODBMS comerciales. Entre ellas encontramos:

- Objetos complejos
- Identidad
- Clases
- Comportamiento
- Encapsulamiento
- Herencia
- Redefinición del comportamiento y binding tardío
- Persistencia

4.2. Tópicos avanzados de Bases de Datos Orientadas a Objetos

Las capacidades funcionales identificadas en esta sección son aquellas específicas de los sistemas de bases de datos OO.

- *Relaciones e Integridad Referencial*

Las relaciones son un componente esencial del paradigma de modelado orientado a objetos. Las relaciones permiten a los objetos referenciarse entre ellos y formar redes de objetos interconectados. Las relaciones son los caminos seguidos para acceder a los datos mediante navegación.

La existencia de las relaciones implica la necesidad de integridad referencial. La integridad referencial asegura que los objetos no mantengan referencias a objetos eliminados. La integridad referencial puede ser automáticamente provista para relaciones bidireccionales, en cambio no puede asegurarse para relaciones unidireccionales.

- *Objetos Compuestos*

Los objetos compuestos son grupos de objetos interrelacionados que pueden ser vistos lógicamente como un objeto simple. Los objetos compuestos son usados típicamente para modelar relaciones que representan semánticamente a la relación “es parte de”. Los objetos compuestos

están conectados por los mecanimos de relación provistos por los OODBMS. Las operaciones que se aplican al objeto 'raiz' de un grupo, pueden ser propagados a todos los objetos del grupo.

- *Transparencia de Localización*

La transparencia de localización es el concepto por el cual un objeto puede ser referenciado sin importar la base de datos en que esté físicamente. Los objetos deben poder ser trasladados y continuar manteniendo las referencias intactas.

- *Versionado de Objetos*

Este concepto se refiere a que cualquier objeto puede contar con múltiples versiones de sí mismo. Existen las siguientes formas de versionado:

- Versionado lineal, donde se almacenan las versiones previas de los objetos a medida que éstos cambian.
- Versionado Ramificado, en que se manejan políticas de concurrencia, donde múltiples usuarios pueden actualizar los mismos datos concurrentemente. En este caso, se pueden crear múltiples versiones nuevas de un objeto.

Asociado con la idea de versionado está el concepto de configuración. Una configuración, es un conjunto de versiones de objetos que son consistentes entre ellas. Es decir, es un grupo de objetos, cuyas versiones combinan entre ellas. Un OODBMS debe implementar una política de versionado, este versionado puede referirse a versiones de objetos ó a versiones de Bases de Datos.

- *Soporte para Trabajo en Grupo*

Una de las formas de soporte para trabajo en grupo, es justamente la capacidad de versionado de los objetos. Pero además, es necesario contar con la posibilidad de designar a las bases de datos compartidas y/o privadas, conjuntamente con el concepto de check-in y check-out de la información de la base de datos, que no es ni más ni menos que permitir que algunos segmentos de la base de datos puedan quedar fuera de linea temporalmente, para ser levantados en una computadora portátil y luego devueltos a la base de datos de orígen.

- *Evolución del Esquema*

Evolución del esquema es el proceso por el cual los objetos de bases de datos existentes, son puestos en línea luego de haber efectuado cambios a las definiciones de las clases de esos objetos. La evolución de los esquemas no es esencial durante el desarrollo de sistemas, pero sí lo es durante el mantenimiento y/o actualización de aplicaciones.

Los cambios en esquemas implicarán modificación en las instancias de una clase modificada, así como también en las aplicaciones referenciadas por la clase modificada. Existen varias formas de realizar la actualización: automática ó manual y de manera compulsiva ó diferida.

- *Acceso/Definición/Modificación de Esquema en tiempo de ejecución*

Un OODBMS típicamente usa una representación residente del esquema de la base de datos, proveyendo así acceso directo a la información del esquema. Esta información es útil para herramientas que muestren la estructura y contenido de la base de datos. Si la modificación y definición del esquema puede ser efectuado en tiempo de ejecución, entonces se pueden desarrollar aplicaciones dinámicamente extensibles.

- *Integración con Bases de Datos y Aplicaciones existentes.*

Las nuevas aplicaciones orientadas a objetos que se desarrollen necesitarán acceder a las bases de datos relacionales existentes. La información almacenada en las bases de datos orientadas a objetos debe ser accesibles a las aplicaciones SQL existentes. Algunas aplicaciones requerirán

acceso tanto a las bases de datos relacionales como a las orientadas a objetos. Los proveedores de OODBMS están tendiendo a soportar SQL para acceder a sus datos. Esto se logra incorporando una interfaz SQL en la interfaz de programación de la aplicación (API) del OODBMS y proveer la capacidad de procesamiento de consulta de SQL.

- *OODBMS Activo vs. OODBMS Pasivo*

Un OODBMS pasivo es aquel que no almacena la implementación de los métodos definidos por las clases. La ejecución de una aplicación, resulta en que cada objeto que es accedido durante la ejecución, es transferido del servidor al cliente. Una vez que el objeto reside en el espacio de direccionamiento de la aplicación, se envía un mensaje al objeto resultando en la ejecución de uno de los métodos del mismo.

Un OODBMS activo es aquel que almacena la implementación de los comportamientos de los objetos. Esto permite que los objetos ejecuten esos comportamientos en el servidor de base de datos. Las ventajas de un modelo de datos activo son:

- Los objetos pueden ser accedidos y manipulados por programas no orientados a objetos.
- Los comportamientos de los objetos son almacenados en una única ubicación, lo que hace más fácil de controlar que todas las aplicaciones tengan la última versión de esos comportamientos.
- Cada objeto accedido por la aplicación no necesita ser transferido al espacio de direccionamiento de la aplicación.
- Las reglas de consistencia pueden ser automáticamente mantenidas por la base de datos.

4.3. Arquitectura de Bases de Datos

4.3.1. Enfoque Cliente-Servidor distribuido

Los OODBMS típicamente ejecutan en un ambiente de multiprocesamiento distribuido. Los procesos del servidor proveen los servicios de bases de datos tales como administración de almacenamiento secundario y control de transacciones. Los procesos del cliente manejan las actividades específicas de la aplicación, tales como acceso y actualización de objetos individuales. Estos procesos pueden estar ubicados en la misma workstation o en diferentes workstations. Típicamente un único servidor estará interactuando con múltiples clientes , dando servicio a pedidos concurrentes para la información manejada por ese servidor. Un cliente puede interactuar con múltiples servidores para acceder a la información distribuida a través de la red.

Tres alternativas de arquitectura workstation-servidor se han propuesto para usar con OODBMS:

- Enfoque Servidor de Objetos:
- Enfoque Servidor de Páginas:
- Enfoque Servidor de Archivos:

4.3.2. Mecanismo de acceso a los datos

Se refiere al proceso necesario para mover los datos desde almacenamiento secundario a la aplicación del cliente.

4.3.3. Agrupación de Objetos

Los OODBMS que transfieren unidades más grandes que un objeto, lo hacen bajo la suposición que el acceso de una aplicación a un objeto determinado, implica una alta probabilidad que otros objetos asociados puedan también ser accedidos. Mediante la transferencia de grupos de objetos, no será necesaria la interacción con el server nuevamente para satisfacer los accesos adicionales a estos objetos.

4.3.4. Operación heterogénea

El rol de un OODBMS, podría definirse como una tecnología de integración de aplicaciones. Un OODBMS provee un mecanismo para que las aplicaciones cooperen, mediante el acceso compartido a un conjunto común de objetos. Un OODBMS típico, soportará múltiples aplicaciones concurrentes ejecutando sobre múltiples procesadores conectados via una red de área local. A menudo, los procesadores pueden ser de diferentes fabricantes, cada uno con sus propios formatos de representación de datos. Por lo tanto, para ser un mecanismo efectivo de integración, un OODBMS debe soportar el acceso a datos en un ambiente de procesamiento heterogeneo.

4.4. Funcionalidad de Base de Datos

La principal ventaja que una aplicación encuentra en la utilización de base de datos es que la información persiste entre ejecuciones de la aplicación. Otros beneficios ofrecidos por una base de datos son la capacidad de compartir datos entre aplicaciones, provisión de acceso concurrente a los datos por aplicaciones múltiples, y provisión a la aplicación de un espacio mayor que el asignado para su procesamiento. Los ítems que se consideran en esta sección son los que distinguen a un OODBMS de un lenguaje con persistencia (Smalltalk). Entre ellos encontramos:

- *Acceso ilimitado a datos*
- *Integridad*
- *Concurrencia*
- *Recuperación*
- *Transacciones*
- *Detección de Deadlocks*
- *Bloqueo*
- *Backup y Restore*
- *Dump y Load*
- *Restricciones*
- *Modelo de Notificación*
- *Indexado*
- *Reclamo de almacenamiento*
- *Seguridad*

4.5. Interfaz de Programación de la Aplicación.

Una gran diferencia entre una aplicación de base de datos orientada a objetos y una aplicación de base de datos relacional, es la relación entre el lenguaje de manipulación de datos (DML) y el lenguaje de programación de la aplicación. En una base de datos relacional, las aplicaciones son desarrolladas en un lenguaje de programación estándar (C, C++) y el DML usado es SQL. Por lo tanto una cantidad importante de esfuerzo es invertido en la transformación de datos entre los dos lenguajes. Esto no ocurre en las OODBMS, en que por ejemplo C++ es el DML y también el lenguaje de programación de la aplicación.

4.5.1. Lenguaje DDL/DML

Un OODBMS debe proveer uno o más lenguajes en que las definiciones de los datos puedan ser especificados y se puedan construir las aplicaciones.

4.5.2. Completitud Computacional

Los lenguajes de consulta de las bases de datos relacionales, tales como SQL, no son computacionalmente completos, por lo que las aplicaciones deben construirse insertando las sentencias del lenguaje de consulta en el lenguaje de programación estándar. En cambio los OODBMS, usan un lenguaje de programación estándar tanto para la definición de los datos como para el manejo de los mismos.

4.5.3. Estilo de Integración del lenguaje

Se pueden utilizar varios mecanismos para proveer acceso a la base de datos desde un lenguaje de programación. Entre ellas se encuentran las interfaces a librerías, extensiones de lenguajes, ó para lenguajes OO puros definición de comportamientos para la construcción, destrucción y acceso por funciones miembro. Se define la integración débil del lenguaje como la programación explícita de las operaciones de bases de datos, por ejemplo las llamadas a librerías. Se define la integración estrecha del lenguaje, a la que hace a las operaciones de base de datos transparentes, típicamente a travéz de comportamientos heredados en la jerarquía de clases.

4.5.4. Independencia de Datos

Es la habilidad que tiene el esquema de la base de datos de modificarse sin impactar la vista externa de ese esquema. El concepto de atributos derivados también se involucra con la independencia de datos. Un atributo derivado es aquél que no se almacena, sino que se calcula cuando se necesita.

4.5.5. Standards

Las aplicaciones pueden transportarse entre distintos OODBMS, si todos ellos concuerdan en una interfaz de programación de la aplicación (API). El Grupo de Administración de las Bases de Datos de Objetos (ODMG) es un grupo de trabajo compuesto por empresas vendedoras de OODBMS, que tienen como tarea la definición de un conjunto de especificaciones de interfaz que aseguren la portabilidad de las aplicaciones e interoperabilidad.

4.6. Consultando una OODBMS

Es muy importante para un OODBMS proveer un lenguaje de consulta declarativo que esté estrechamente integrado con un lenguaje de programación OO.

4.6.1. Capacidad de consulta asociativa

Los proveedores de OODBMS soportan sus propias extensiones de SQL para proveer acceso razonable a sus bases de datos de objetos desde un lenguaje SQL. El rango de capacidad de consulta que puede proveer un OODBMS es el siguiente:

- Sin soporte para consultas, los datos deben accederse mediante navegación en la base de datos.
- Consultas basadas en "COLLECTIONS", en que las consultas operan sobre algunas collections de objetos predefinidas, seleccionando objetos de acuerdo a algún predicado, y dando como resultado una collection de objetos.
- Consultas generales, que pueden tener un resultado de cualquier tipo, como valor, objeto, collection, etc.

4.6.2. Independencia de Datos

La motivación principal para usar un lenguaje declarativo de consulta, es para soportar la independencia de los datos.

4.6.3. Separación Semántica ó "Impedance Mismatch"

Se presume que uno de los beneficios de usar un OODBMS es que el lenguaje de programación de la aplicación y el DML son uno y el mismo. Sin embargo, algunos entendidos dicen que esto elimina la independencia de datos. Las capacidades de las consultas declarativas, que se están agregando a los OODBMS, soportarán el concepto de independencia de datos. La manera en que estas capacidades de consulta se integren con el lenguaje de programación de la aplicación, determinarán el nivel de separación semántica entre el lenguaje de programación de la aplicación y el uso de consultas asociativas.

El rango de técnicas para integrar las consultas en un OODBMS es:

- Proveer un método SELECT en todas las clases persistentes
- Extender el lenguaje de programación de objetos para incluir predicados de SQL para filtrar las operaciones de selección
- Insertar la sentencia SQL SELECT en el lenguaje de programación OO, proveyendo un preprocesador que traduzca estas sentencias a un conjunto apropiado de llamadas en tiempo de ejecución.

4.6.4. Invocación de Consultas

Lo más importante, es proveer una capacidad de consulta ad hoc desde las aplicaciones. Un requerimiento adicional, es proveer un medio para la invocación de consultas ad hoc, posiblemente desde una herramienta browser de la base de datos.

4.6.5. Invocación de Comportamientos Programados.

El lenguaje de consulta debe ser capaz de invocar los comportamientos de objetos como parte de sus predicados.

5. REVISIÓN DE UN OODBMS DEL MERCADO.

En esta sección se efectuará el análisis de la documentación técnica ofrecida por un proveedor de OODBMS, comercialmente conocido como VERSANT. Esta documentación está disponible públicamente en el sitio: www.versant.com.

El análisis realizado no pretende ser exhaustivo en cuanto al cumplimiento o nó por parte del producto, de las características detalladas en la sección anterior. Sería más adecuado considerarlo una comparación de las características ofrecidas por este proveedor contra las características funcionales esperadas de un OODBMS, y además poder visualizar en un producto concreto, qué decisiones de diseño se tomaron al momento de lanzar el producto al mercado.

5.1. Algunas de las características soportadas por VERSANT OODBMS.

5.1.1. Modelado Básico Orientado a Objetos

Este producto soporta completamente el modelo de objetos del desarrollo, incluyendo conceptos tales como clases, herencia, encapsulamiento y polimorfismo. La ventaja de este producto de soportar modelos complejos es importante en la medida que las organizaciones definan sus estándares básicos con el paradigma Orientado a Objetos. Además, dado que el producto permite almacenar directamente los objetos de C++, Java y Smalltalk, el tiempo de desarrollo y mantenimiento de una aplicación disminuye, logrando aplicaciones más confiables y de mejor performance.

5.1.2. Tópicos Avanzados de Bases de Datos Orientadas a Objetos

Este producto provee altos niveles de integridad referencial. Debido a que soporta el modelo de desarrollo de Smalltalk, las reglas y definiciones se aseguran automáticamente. Las relaciones entre objetos en la base de datos son representadas con un tipo único de puntero persistente.

En cuanto al versionado, el producto soporta acceso explícito ó implícito a las versiones de los objetos. Cuenta con técnicas para el manejo de múltiples versiones, tales como referencia a la versión de padres e hijos y además la configuración para soportar los distintos estados, tales como: transitorio, en proceso, liberado, etc.

En cuanto al soporte para trabajo en grupo, este OODBMS soporta el uso de bases de datos personales, que le permiten al usuario reservar una porción de la base original, para su propio uso privado. En combinación con otras características para trabajo en grupo, las bases de datos personales pueden desconectarse de la base de datos principal, y luego reconectarse para actualización. El mecanismo que se utiliza para posibilitar este tipo de trabajo es el denominado CHECKIN/CHECKOUT, el cual consiste en una manera de establecer un bloqueo permanente sobre un conjunto de objetos, permitiendo al desarrollador trabajar sobre un objeto o grupos de objetos y denegar el acceso a los demás usuarios hasta que se haya completado la operación.

En cuanto al esquema de la base de datos, este producto lo genera automáticamente a partir de las clases Smalltalk, eliminando la necesidad de procesamiento adicional para la definición de la base de datos. Además, la evolución del esquema es dinámica y se realiza en línea, es decir la base de datos no necesita detenerse para actualizar el esquema ó los objetos. La forma en que implementa la actualización este producto, es perezosa, es decir que las instancias se actualizan en la medida que sean usadas. Esta táctica, se utiliza para amortizar el costo de la actualización con el tiempo y el uso.

5.1.3. Arquitectura de Bases de Datos

La arquitectura de este producto es cliente-servidor balanceados. Este OODBMS, permite compartir los objetos entre sistemas dispares, y permite el acceso a todas las bases de datos de objetos sin importar el hardware, plataforma y ambiente Smalltalk usado para crear un objeto dado, no es necesario realizar una recompilación ó reinicialización de la base de datos.

Por otra parte, las tareas de mantenimiento ó rutinarias tales como agrupamiento de objetos, son realizadas automáticamente por el OODBMS.

5.1.4. Funcionalidad de Bases de Datos

En cuanto al manejo de transacciones, el producto asegura la atomicidad mediante la utilización de la técnica two-phase commit. Se implementan las transacciones distribuidas de manera transparente, manteniendo la integridad sin importar la ubicación del objeto, el OODBMS se encarga de encontrar por sí mismo los objetos necesarios para concretar la transacción. Además, provee flexibilidad adicional al implementar mecanismos tales como commits parciales de cache, puntos de verificación y puntos de resguardo.

Este producto, brinda además soporte para transacciones largas, típicas en un OODBMS, las que son extremadamente útiles cuando se utilizan en conjunto con las técnicas para soportar trabajo en grupo, tales como checkin/checkout. Además de soportar transacción largas, este OODBMS provee otros tipos de transacción, entre los cuales figuran transacciones anidadas, sesiones compartidas y bloqueo personalizado. Si en algún momento se produjera una falla, este OODBMS asegura que automáticamente las transacciones involucradas serán deshechas (rolled back) y los bloqueos serán liberados.

Por otra parte, se asegura la transparencia en la detección de deadlocks potenciales, antes de que ocurran, liberando a los desarrolladores y usuarios de problemas de ciclos y sistemas congelados. Esto lo logra mediante la no otorgación de un bloqueo que pudiera causar un deadlock.

En cuanto al manejo de los bloqueos, este producto trabaja con bloqueo a nivel de objetos. Es decir los bloqueos pueden ser aplicados a clases, instancias ó versiones de objetos, soportando tanto bloqueos para transacciones cortas como bloqueos persistentes para transacciones largas. La política de bloqueo adoptada por este producto es optimista, es decir, permite a los usuarios trabajar con los objetos sin bloqueos hasta el momento del commit, garantizando un estado consistente del objeto para asegurar que los usuarios no se reescribirán unos a otros involuntariamente. También se soporta la flexibilidad adicional para que los usuarios puedan extender el modelo definiendo sus propias reglas de bloqueo.

Se provee también recuperación total de la base de datos, a travéz de backup incremental en línea, el cual asegura una vuelta rápida y segura a las condiciones previas de la falla. La recuperación de este producto, utiliza logging dual (físico y lógico) así como también log rollforward para garantizar la recuperación hasta la última transacción con commit en los logs.

En cuanto al modelo de notificación, este OODBMS, provee un política de manejo de eventos basada en el servidor. Esta herramienta es asincrónica y se basa en la técnica “publicar y suscribir”, funcionando de la siguiente manera: los eventos se registran en un servidor de base de datos cualquiera, el cual corre un proceso background para capturar los eventos que ocurran, y se realiza la notificación a la aplicación cliente. Los eventos registrados pueden ser actualizaciones ó eliminaciones de objetos ó clases específicas, así como también creación, modificación y eliminación de cualquier instancia de una clase. Dado que el manejo de eventos es basado en el servidor, esto permite que se modifiquen los eventos registrados sin cambiar el código existente, de la aplicación. En el caso alternativo de manejo de eventos basado en el cliente, cada aplicación debe reescribirse para tener en cuenta cada nuevo evento.

Otro punto a considerar es el mantenimiento de los índices. Este proceso se efectúa automáticamente de manera contínua, tal que se libera al desarrollador de rehacer los índices cada vez que se modifique una clase.

Otra característica importante, es la de reclamo de almacenamiento, la cual es posible en este producto, dado que usa identificadores lógicos para cada objeto, y cuando se elimina alguno de ellos, el espacio es automáticamente reclamado. Al realizarse de esta manera se minimiza la fragmentación de la base de datos y se mejora la performance.

5.1.5. Interfaz de Programación de la Aplicación

Este producto soporta los lenguajes Java, Smalltalk y C++, y además permite interoperabilidad entre esos lenguajes, es decir que objetos creados en Smalltalk son accesibles desde Java y C++ y viceversa.

Las interfaces del lenguaje de este OODBMS son fáciles de usar, debido a que las operaciones primitivas de bases de datos son transparentes al desarrollador, eliminándose así la necesidad de mapear código. Los desarrolladores no necesitan cambiar su ambiente de programación a otro lenguaje como SQL, para almacenar y/o extraer objetos de su base de datos.

Asimismo, este producto ofrece un conjunto de liberías de clases y herramientas de integración para asistir en el diseño y desarrollo de aplicaciones.

En cuanto a los estándares, este producto además de soportar TCP/IP y OSI, es un miembro fundador del grupo ODMG y miembro del comité X3J16 ANSI C++.

5.1.6. Consultando un OODBMS

Este OODBMS, brinda acceso para consultas y reportes mediante concectividad ODBC a la base de datos. Los desarrolladores de Smalltalk, por ejemplo, pueden crear modelos de objetos de complejidad arbitraria y estos pueden ser automáticamente mapeados a un esquema compatible con SQL. Por lo tanto, se pueden usar herramientas estándar, para consultas y reportes, ya que toda la información del modelo Smalltalk, está disponible para usuarios no-Smalltalk.

6. CONCLUSIONES

La principal ventaja de utilizar un OODBMS es que el desarrollador debe desenvolverse solamente en un ambiente de objetos, preocupándose por diseñar correctamente el modelo y cumplir con las restricciones del mismo. Las preocupaciones no deben estar centradas en traducir esta u otra parte del diseño a otro modelo

Por otra parte, los RDBMS, se encuentran muy afianzados en el mercado actual, con múltiples aplicaciones desarrolladas, y lo que es más importante, mucho contenido de información. Por lo tanto, es muy importante que los nuevos productos de bases de datos provean acceso a los datos de un RDBMS, via SQL o algún equivalente (tipo sql3), y viceversa, dado que deberán coexistir largo tiempo.

Las bases de datos de Objetos constituyen la tendencia más notable en cuanto a tecnología de bases de datos se refiere. Esto se infiere a partir de la aparición de ambas líneas, OODBMS y ORDBMS, que indica una fuerte convergencia al concepto de objetos como un mecanismo clave para extender las capacidades de las Bases de datos, aunque todavía queda mucho por definir para contar con un producto maduro.

Como vimos en la evolución de los OODBMS, actualmente esta tecnología se encuentra desarrollada en un 50%, así que habrá que seguir muy de cerca los acontecimientos futuros.

7. BIBLIOGRAFÍA

- Elmasri-Navathe. “Fundamentals of Database Systems”, Addison-Wesley – 3°Ed. 2000.
- Loomis, Mary. “Object Database Essentials”, DBMS Interview – December 1994.
- Maier, David. “Comments on the Third-Generation Data Base System Manifesto” – 04/1991.-
- Manola, Frank. “An Evaluation of Object-Oriented DBMS Developments” – 08/1994.-
- McFarland, Rudmik, Lange. “Object-Oriented Database Management Systems Revisited” – 01/1999.-
- Stajano, Frank. “Object Oriented Databases: An Overview” – 11/1995.-
- http://www.versant.com