Una Perspectiva de las Bases de Datos Orientadas a Objetos con Respecto a las Relacionales

LAI. Eunice Echeverría Portillo.
Profesora del CIDETEC-IPN.
M. en C. Ivonne Rivera Santos.
LAI. María del Carmen Celis Ortega.
Profesoras del CIC-IPN.

Los actuales manejadores de Sistemas de Bases de Datos son inadecuados para el manejo de información inherentemente compleja, lo que ha generado la necesidad de nuevos tipos de sistemas, en respuesta al nuevo rango de requerimientos de las estructuras de información complejas, tales como: integración de código y datos, multimedia, versiones de datos, datos persistentes, etc. Las Bases de Datos Orientadas a Objetos (BDOO) ofrecen una arquitectura que soporta estas estructuras; aunque se encuentran actualmente en desarrollo, lo que representa algunas limitantes.

Introducción

Para adentrarse en el tema es conveniente analizar la definición de Sistema Manejador de Bases de Datos (SMBD), propuesta por Richard Cooper:

Es un software que debe permitir el manejo de grandes cantidades de datos estructurados, es decir, que cuenta con una forma previamente establecida por algún modelo.

Por otro lado, Kim W. refiere que una BDOO es una colección de objetos en los que su estado, comportamiento y relaciones se definen de acuerdo a un modelo de datos.

Ambos conceptos indican que los datos a manipular tienen una composición lógica, previamente diseñada, y que existen relaciones entre ellos.

Un SMBD no es un simple repositorio de datos, dado que posee una arquitectura, un modelo de datos y un conjunto de lenguajes especiales para definir, manipular y consultar los datos almacenados (ver figura 1); la finalidad del almacenamiento, es recuperar información valiosa, del contenido o de la infraestructura. El valor de un sistema de información radica en la potencia de sus consultas y el manejo de sus datos.

Con el rápido avance de la tecnología en computación, la definición anterior ha extendido su semántica a otros tipos de sistemas de información, de donde se derivan los siguientes conceptos:

Sistema Manejador de Bases de Datos (SMBD): Paquete de Software que permite manejar gran-

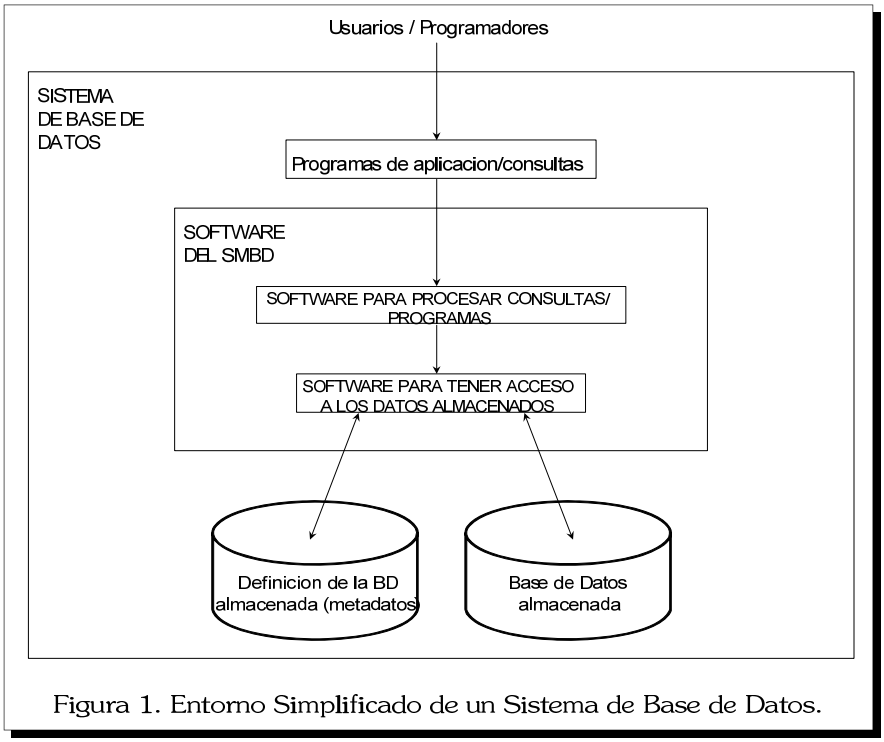

Figura 1. Entorno Simplificado de un Sistema de Base de Datos.

des cantidades de datos estructurados.

Sistema de Objetos o Sistema basado en Objetos (SO): Sistema que soporta el modelado de datos como entidades abstractas con identidad.

Sistema Orientado a Objetos (SOO): Sistema de objetos donde todos los datos se crean como instancias de clase.

Sistema Manejador de Bases de Datos Orientadas a Objetos (SMBDOO): Sistema de base de datos que permite la definición y manipulación de una base de datos orientada a objetos.

Bases de Datos Orientadas a Objetos (BDOO): Es una base de datos formada por objetos y manejada por un SMBDOO.

En todas estas definiciones se encuentra un concepto común: el modelo de datos de una aplicación. El Modelo de Datos es la estructura básica con que se construyen todas las entidades en el sistema (especificaciones denominadas esquema). Por ejemplo, el modelo relacional está basado en los conceptos registro, columna, llave, tabla y relación. Todas las construcciones de los sistemas basados en este modelo, emplean esta estructura para modelar un problema.

En el caso del modelo de objetos, son los conceptos objeto, clase, herencia y demás componentes del paradigma, los que estarán presentes en tal modelado.

Los Sistemas de Objetos a los que se refiere la definición anterior incluyen a los Sistemas de Objetos Complejos, Modelos Semánticos de Datos y Lenguajes de Programación Orientados a Objetos (LPOO).

En los SOO la creación de toda entidad se realiza basándose en la instanciación de una clase, dando como resultado un objeto que encapsula estado y comportamiento, en donde dicha clase forma parte de una jerarquía de herencia.

Un SMBD además de definir, manipular y consultar grandes cantidades de datos, debe proporcionar ciertas facilidades identificadas en el cuadro sinóptico de la figura 2, las cuales se describen brevemente a continuación.

- Generalidad de Aplicación

  Implica que el Sistema debe ser capaz de manejar datos de diferentes áreas de aplicación de igual manera, empleando las mismas técnicas. Es decir, no importa la naturaleza de los datos ni el tipo de sistema de información que los emplea para desarrollar una aplicación en el manejador. Por ejemplo, se diseña el SMBD para que manipule un sistema de nómina de una empresa, y por igual, para manejar un sistema de registro y seguimiento para pacientes de un hospital.

- Acceso eficiente a datos

  Se refiere a que los datos se almacenarán de forma tal que puedan accederse de la manera más rápida posible. Esta velocidad se logra usando estructuras de datos diseñadas especialmente para indexamiento, las cuales aceleran la búsqueda de un dato y su acceso ordenado (familia de los árboles B, B+, B*). Esto es, búsquedas en un orden de O(lg n).

  Por otro lado, se emplean técnicas de almacenamiento de datos basándose en clusters, las cuales consisten en utilizar pequeñas áreas en disco (un conjunto de "x" bloques lógicamente contiguos) donde se almacenan datos relacionados. Esto garantiza que las transferencias de lectura y escritura para datos que se encuentran en la misma área se realicen en bloques, lo que acelera dichas operaciones.

  Como ejemplo de ello se tiene que, al leer la relación entre los registros de dos tablas relacionadas, almacenados en una misma área, existe la garantía de que los registros implicados se recupera-

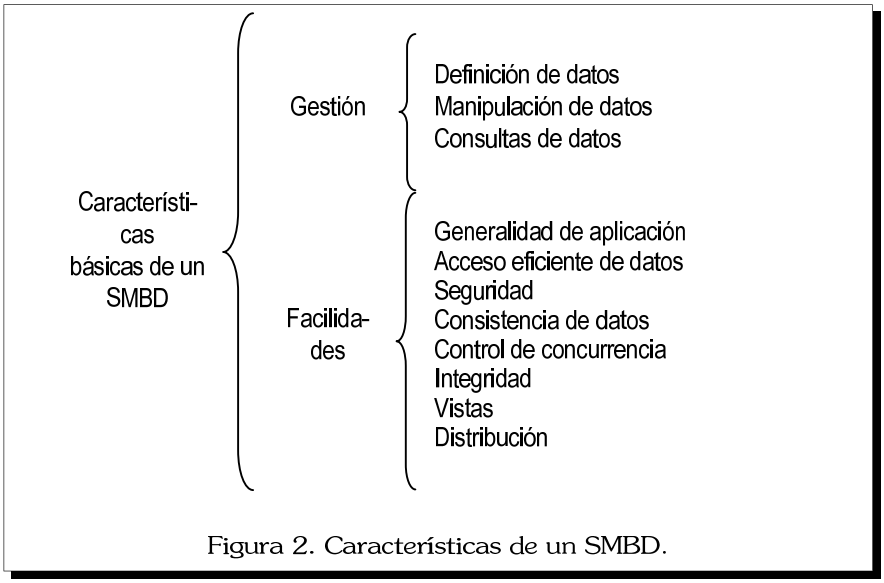

Figura 2. Características de un SMBD.

rán más rápidamente que si estuvieran dispersos en el disco. Por otro lado, el almacenamiento de todos los objetos de una clase en una sola área, permite optimizar los algoritmos de actualización del recolector de basura.

- SEGURIDAD

Para un SMBD, la seguridad implica la existencia de mecanismos de privacidad que garanticen la confidencialidad de los datos almacenados. No todos los usuarios deben tener acceso a todos los datos, de ahí que el empleo de dichos mecanismos incluya desde asignar niveles de acceso a una base de datos, permisos de lectura y escritura a tablas, hasta el empleo del principio de encapsulación de los objetos, donde sólo se acceden las propiedades que la interfaz del objeto muestra al exterior.

- CONSISTENCIA DE DATOS

Otro concepto importante empleado en los SMBD es la **transacción**, que consiste en uno o más cambios sobre los datos de la base, lo cual la cambia de un estado consistente a otro.

Mediante la consistencia de datos se garantiza que las transacciones a realizar se efectúen de forma tal que, en cualquier momento que se consulte la base de datos existirán datos consistentes. Esto está estrechamente relacionado con la resistencia o **tolerancia a fallas**, ya sea por parte del software o hardware, lo cual implica que, después de la falla, los datos regresarán al último estado consistente anterior a la modificación que provocó dicho fallo.
Uno de los mecanismos para asegurar la consistencia es organizar los accesos a la base de datos en transacciones, garantizándose la Atomicidad, Consistencia, Independencia y Durabilidad del manejo de transacciones (propiedades ACID), para los datos almacenados.

- CONTROL DE CONCURRENCIA

Si el Sistema cuenta con un mecanismo de bloqueo, entonces soporta un ambiente multiusuario; esto es, permite el acceso de más de un usuario a la misma base de datos y garantiza que no habrá interferencias en el procesamiento. Esto conlleva un nivel de granularidad de bloqueo en el SMBD, el cual puede ser a nivel de la base de datos, de tabla, de columna o de registro (Modelo Relacional) o de base de datos extensión u objeto (Modelo de Objetos).

- INTEGRIDAD

Existe la facilidad de imponer restricciones para mantener la coherencia de los datos almacenados, así como establecer rangos de dominios para los valores de las columnas de los registros o del estado de un objeto. Dichas restricciones son sentencias que se verifican en cada modificación a los datos almacenados; en el caso de infringirse, generan excepciones o disparan otros procedimientos almacenados.

- VISTAS

Son diferentes representaciones de los datos almacenados, las cuales se generan con la finalidad de restringir o seleccionar cierta información que se presentará a los diferentes tipos de usuario. Conocidas también como **interfaces de usuario**, son la cara que presenta el SMBD a los usuarios para el manejo de la aplicación y sus diferentes herramientas. Estas pueden ser paramétricas, de consulta, gráficas, de programación o administrativas.

- DISTRIBUCIÓN

Un SMBD que permite trabajar con bases de datos que se encuentran físicamente particionadas o replicadas en diferentes lugares, las cuales funcionan como una sola base de datos a través de una comunicación vía red.

En resumen, los SMBD poseen estas facilidades a fin de manejar los datos mediante una forma sencilla: el modelo de datos lógico.

Otro tipo de modelo de datos empleado actualmente para trabajar grandes cantidades de información es el Sistema basado en el Manejo de Registros.

Estos Sistemas se caracterizan por el uso del registro como entidad básica o estructura principal de datos, el cual está compuesto por campos o columnas que poseen un tipo de dato previamente definido y con naturaleza atómica.

Existen dos tipos principales de SMBD basadas en archivos: de Red y Relacionales.

Los SMBD de Red organizan las colecciones de registros del mismo tipo como listas ligadas, las cuales cuentan con una cabecera que describe la estructura del tipo de registro que mantiene.

Por su parte, en los SMBD **Relacionales**, donde las colecciones de registros del mismo tipo reciben el nombre de tabla, se maneja el concepto de llave primaria, cuyo fin es el de garantizar una identidad única a cada registro. Las relaciones entre colecciones de registros tienen dife-

rente representación según el tipo del SMBD. Por ejemplo, en los de tipo relacional se emplea la llave foránea y en los de tipo red se agrega un campo más al registro, el cual contiene un apuntador a la otra lista o listas con las que se relaciona.

Estos SMBD manejan enormes cantidades de datos eficientemente, gracias al almacenamiento simple en registros con campos atómicos. Sin embargo, el gran éxito de estos sistemas ha motivado a un amplio rango de nuevos clientes potenciales a buscar alternativas que satisfagan sus necesidades ante el manejo de datos más complejos, ya que en estos manejadores relacionales los datos no pueden transformarse fácilmente dentro de sus estructuras simples.

Hasta hace algunos años, el uso de los Manejadores Basados en Registros había sido suficiente para desarrollar aplicaciones comerciales robustas; no obstante, el desarrollo de nuevas líneas de investigación en software ha generado necesidades que no cubren los manejadores tradicionales. Su empleo en aplicaciones avanzadas ha significado que el diseño y su adaptación se torne extremadamente problemático.

A continuación se presentan brevemente algunas de estas aplicaciones avanzadas:

Diseño Asistido por Computadora (CAD), la Ingeniería Asistida por Computadora (CAE) y la Manufactura Integrada por Computadora (CIM), las cuales implican la creación y el mantenimiento de estructuras complejas que describen objetos técnicos que requieren un trabajo cooperativo, tales como los diagramas.

Igualmente, se encuentra la Ingeniería de Software Asistido por Computadora (CASE), la cual mejora el proceso de diseño y mantenimiento de software, proporcionando mecanismos para almacenar, buscar y recuperar componentes de software de forma sistemática, los cuales requieren estar organizados en estructuras complejas.

Estas dos áreas de aplicación son usualmente referidas como aplicaciones de diseño, por lo que comparten características comunes y manejan diseños complejos de información.

Por otra parte, están los Sistemas de Información de Oficina (OIS), los cuales permiten manejar la información tradicional de una oficina (documentos tales como: mensajes, listas, etc.), pero no cuentan con una estructura definida, y requieren de almacenar datos junto a los procesos que los manejan.

Los Sistemas de Información Geográfica (GIS), almacenan información que se deriva de tomas satelitales e inspecciones, con relaciones espaciales y temporales, así como complejas formas de visualización. Juegan un papel importante en los sistemas de transportación, en la organización de herramientas de telecomunicaciones, tierras de cultivo e irrigación.

El World Wide Web (WWW) maneja sistemas hipermedia con datos de texto integrado, numéricos, gráficos y sonido, empleando funciones que permiten recorrer tales estructuras. Utiliza técnicas de compresión para tipos grandes de datos. Sin embargo, la WWW se ha desarrollado con poca tecnología de bases de datos.

Las características de los SMBD relacionales aplicados con gran éxito en los negocios, dieron pie a que se les considerara con similar importancia en aplicaciones avanzadas o complejas; sin embargo, algunas de estas características necesitaban una transformación radical , mientras que otras realmente eran requerimientos completamente nuevos . No se considera al modelo relacional como la arquitectura idónea para el desarrollo de las nuevas áreas, debido a que presenta algunas carencias y desventajas para sus nuevas necesidades. A continuación se describen aquellas características más relevantes que requirieron de alguna modificación:

- TRANSACCIONES CORTAS

  Los manejadores relacionales están diseñados para trabajar con transacciones cortas y no soportan las de largo tiempo. Una transacción larga, como en las aplicaciones avanzadas, puede durar el tiempo necesario para actualizar una pieza o diseñar un componente de software (horas, días). El aislamiento es un aspecto vital de la transacción desde el momento en que se compite por los recursos. El Aislamiento y el Control de concurrencia de una transacción tradicional (ACID) son características válidas en una transacción de corto tiempo, pero no en una transacción de periodo largo. Las transacciones largas requieren un trabajo colaborativo y cooperativo más que competitivo. Cuando una transacción entra en ejecución y requiere de un largo tiempo en espera para finalizar, se cede la ejecución a otras transacciones. Entonces se requiere de una estrategia que permita envolver nuevos tipos y compartir el acceso.

- ESTRUCTURAS SIMPLES

  Para almacenar una estructura compleja en el modelo relacional es necesaria su conversión a una estructura simple (uso de las formas normales), donde sus campos están atomizados y el rango de valores para los tipos que se ma-

nejan utiliza texto y formato numérico. Por igual, la identificación única del registro se basa en el valor de uno o más campos del mismo. Los desarrollos para las nuevas áreas, por el contrario, requieren el uso de estructuras que no restrinjan su representación a simple texto o números, o por la atomicidad de sus campos, donde la identidad de las entidades que manejan no dependa de la información que contienen.

Las aplicaciones avanzadas requieren facilidades para trabajar con diferentes versiones de los datos y el manejo de restricciones no sólo en la descripción estructural, lo cual tendría que ser añadido a un manejador convencional.

- Información Adicional

Cuando un manejador relacional crea relaciones entre entidades, se agrega una gran cantidad de datos a la tabla, complicando su manejo. No existe una representación adecuada de la identidad de un registro, debido a que se basa en el valor de alguno de sus campos originales o en uno que se creó para dicho fin, lo cual se hace visible en el esquema.

- Sobrecarga Semántica

Esto se debe a que su estructura básica (tabla) tiene más de un sentido de creación. Una tabla sirve tanto para agrupar una colección de registros del mismo tipo como para crear una relación entre tablas diferentes. Las relaciones entre dos registros son el resultado de modelar las propiedades multivaluadas de una entidad o la unión de los componentes de un valor complejo.
La razón por la cual los valores de dos columnas se encuentran relacionados, se debe a que ambas columnas pertenecen al mismo registro o bien están de registros diferentes conectados por algún tipo de relación.

Este sobreuso de las estructuras básicas con más de un propósito, hace imposible descifrar la naturaleza de una aplicación con sólo observar su esquema relacional.

- Impedancia de Correspondencia

Esta se presenta cuando se requiere extraer datos del manejador para realizar cálculos complejos mediante un lenguaje de programación; su empleo se debe a la limitada cantidad de operaciones que el manejador puede realizar sobre los datos.

Por lo general, el uso de un lenguaje orientado a bases de datos (tal como SQL), sirve para recuperar los datos del manejador, procesarlos con las instrucciones del lenguaje de programación y que finalmente se regresen modificados al manejador o los resultados de las operaciones. Sin embargo, la transferencia de datos entre ambas aplicaciones (Base de Datos – Lenguaje de Programación) no siempre se realiza de forma directa, ¿cuál es la causa? Simplemente que ambas aplicaciones están basadas en un modelo de datos diferente y los tipos que maneja cada aplicación no siempre serán equivalentes o compatibles; así, se presenta una resistencia a la navegación directa de los datos de una aplicación a otra, dado que existen tipos desconocidos o con diferente representación.
Al suceder lo anterior, es necesario un mecanismo de conversión para tipos diferentes de datos. La estrategia para solucionar este problema seria unificar los tipos de datos empleados en ambas aplicaciones, es decir, el modelo de datos de la base y los tipos en el lenguaje de programación, permitiendo de esta forma que sean persistentes y realizar cálculos complejos sobre ellos.

- Almacenamiento de Código

Algunas veces, la complejidad y naturaleza dinámica de un dato puede ser tan grande, que podría ser preferible su representación mediante código; sin embargo, los manejadores relacionales (SMBDR) no pueden almacenar código.

A continuación se presentan tres posibles formas de solucionar esta carencia en los SMBDR:

? Agregar una Extensión a la estructura de la base de datos, incluyendo con esto formas de integrar el código y los datos empleados.

- Adicionar una Estructura de Almacenamiento de Aplicaciones al SMBD, a fin de que el código de la aplicación se encuentre dentro del mismo, pero separado de los datos. Por lo regular, ésta es la técnica empleada cuando se actualiza un SMBD prediseñado.

- Considerar al código como Otro Tipo de Datos, el cual pueda reconocer el modelo de datos del SMBD y almacenarlo tal cual.

El almacenamiento de código y datos sin algún tipo de relación, genera falta de integridad e inconsistencia, lo cual resulta de las alteraciones en alguna de las dos partes sin el conocimiento de la otra. Por tal motivo, en este tipo de aplicaciones se emplean extensivamente las restricciones de integridad, y en virtud de que la tarea de modelado de aplicaciones es más compleja, se requiere producir interfaces de usuario más amigables para

aquellos usuarios no especialistas en el tema.

De acuerdo a las necesidades que presentan las nuevas áreas de desarrollo surgen los SMBD avanzados, en donde destacan los siguientes:

1.–SMBD Históricos: Se caracterizan por proporcionar facilidades para conservar todos los estados pasados de una base de datos. Cada vez que se actualiza o elimina un dato, el valor anterior no se remueve de la base, sino que se marca con una estampa de tiempo, la cual presenta información tal como la fecha de creación y destrucción de ese dato, permitiendo así recuperar todas las entidades que han estado presentes en algún momento.

   Este almacenamiento de datos se hace con el fin de soportar la coexistencia de varias versiones de un dato, aunque sólo es válida una versión a la vez. Como ejemplo de este tipo de sistemas se encuentra Chronolog, que trabaja con Oracle.

2.–SMBD de Objetos Complejos: Conocidos también como relacionales sin primera forma normal, estos sistemas son relacionales cuyo valor en un campo de la tabla puede ser un registro completo; como ejemplo de estos está PostgreSQL.

3.–SMBD Deductivos: Estos poseen la capacidad de definir reglas con las cuales deducir e inferir información adicional a partir de datos almacenados en la base. Centran su importancia en los datos derivados y las restricciones, así como en proporcionar mecanismos para describir los datos en términos de estas últimas. La mayoría de estos sistemas se desarrollan como formas persistentes del lenguaje Prolog/Datalog, el cual permite la descripción de los datos como simples hechos y reglas para generar otros hechos. Existe una amplia variedad de desarrollos tanto académicos como comerciales de estos sistemas; entre los primeros destaca el proyecto CORAL de la Universidad de Wisconsin.

4.–Lenguajes de Programación Persistentes: La persistencia de los datos es una de las características más relevantes en un SMBD. Dicha persistencia es la cualidad de algunos objetos de mantener su identidad y relaciones con otros objetos, con independencia del sistema o proceso que los creó. En la práctica, se implementa proporcionando a los objetos un mecanismo cuyo objetivo es tanto el almacenamiento en memoria como la recuperación posterior de éstos. Así, el papel que juegan los lenguajes de programación persistente consiste en eliminar el problema de impedancia de correspondencia, entre los datos persistentes y los lenguajes de programación. Este tipo de lenguajes parte de un lenguaje de programación normal y lo extiende con capacidades de base de datos, y se basan en dos principios:

- Los valores de cualquier tipo de datos del lenguaje de programación se pueden almacenar y recuperar posteriormente (persistentes), incluyendo valores procedurales y multimedia.

- Para almacenar los datos no es necesario cambiar su estructura física.

  Como principales ejemplos se encuentran Napier88 y el proyecto Forest.

5.–SMBD Activos: Implican que el manejador contiene algún componente de comportamiento, usualmente del tipo de restricciones; por ejemplo, actualización de la base de datos o una salida de esta por parte del usuario. Frecuentemente esto es una actividad almacenada que puede iniciarse por un cambio en la base de datos, o por un evento detectado automáticamente, como podría ser el transcurso de un lapso de tiempo. En este tipo de desarrollos sobresale el proyecto IDEA.

6.–Sistemas de Dominio Específico: Son una respuesta a las necesidades de áreas especificas de investigación, tales como las Bases de Datos Espaciales (sistemas de información geográfica), y las Bases de Datos Temporales. Estos tipos de SMBD intentan proporcionar nuevas estructuras dentro del modelo de datos, así como facilidades para emplearlas.

   La mayoría de las investigaciones realizadas en estas áreas, han dado la pauta para el empleo del paradigma orientado a objetos, esperando que en un futuro cercano emerjan componentes espaciales y temporales mediante los Sistemas Manejadores de Bases de Datos Orientadas a Objetos (SMBDOO).

Un SMBDOO soporta la creación y manipulación directa de los objetos desde un Lenguaje de Programación Orientado a Objetos (LPOO). Este tipo de manejador incluye toda la funcionalidad de un SMBD más las construcciones para la descripción de datos que le proporciona el paradigma de la Orientación a Objetos, como son: Identidad de objetos, Clasificación, Encapsulación, Herencia, Redefinición y Sobrecarga (figura 3). A continuación se explica brevemente cada uno de ellos:

Identidad de Objetos

Es el principio que establece que cada entidad del mundo real será representada por una estructura indivisible llamada objeto, el cual posee una identidad que es independiente de los valores que almacenan sus atributos. La identidad del objeto lo distingue de las demás entidades creadas por la aplicación, y es inherente al objeto por el resto de su vida. Existen varias formas de asignar el OID (Object IDentifier) al objeto, siendo obligatorio que la aplicación lo establezca en el momento de la creación y se emplee exclusivamente por el sistema, sin que el desarrollador pueda modificarlo.

Clasificación

Los SMBDOO soportan tipos de datos definidos por el usuario, de tal forma que los desarrolladores pueden crear el tipo de dato que requieran, empleando el principio de abstracción y modelándolo como un Tipo de Dato Abstracto (ADT), considerando a este como sinónimo de Clase o Tipo. Una vez que se define una clase, es posible hacer tantas instanciaciones de ésta como se requiera (crear objetos de dicha clase).

Encapsulación

Cada clase tiene definido un conjunto de atributos que representan su estado, y un conjunto de métodos (propiedades) con los que se manipulan los objetos de la clase, es decir, desarrollan un comportamiento. La descripción de la clase se encuentra dentro de una interfaz, la cual permite la interacción con el mundo exterior; en ella se enumeran los atributos y propiedades con sus respectivas firmas. Esta interfaz representa la división entre la abstracción de la clase y su implementación, comúnmente denominada encapsulación.

Mediante la encapsulación, algunos atributos y propiedades de la clase se tornan inaccesibles (poseen modificadores de acceso) para otras clases dentro de la aplicación. El establecimiento de estos modificadores tiene políticas propias para cada lenguaje de programación.

Herencia

Las clases definidas toman parte en una Jerarquía de Herencia, de tal forma que se puede definir una clase nueva a partir de una existente, la cual heredará los atributos y propiedades de su antecesora, siendo posible adicionar comportamiento y/o atributos a la clase recién creada.

Redefinición

Al utilizar el mecanismo de herencia para definir nuevas clases, es posible que se desee redefinir la funcionalidad de algún comportamiento heredado, proporcionando una nueva implementación a la firma de dicho método, con lo que se oculta o se Redefine la operación original. Así, dependiendo del objeto que invoque este comportamiento, se ejecutará el código indicado.

Sobrecarga

Otro uso del polimorfismo consiste en declarar una nueva funcionalidad bajo el nombre de un método existente, pero con diferencias en la firma; a esto se le denomina sobrecarga.

Redes de Objetos

Este tipo de manejadores también soporta el almacenamiento y navegación de estructuras tipo red, las cuales se constituyen cuando los objetos mantienen referencias a otros objetos (posiblemente creados en diferentes sesiones de la aplicación) estableciendo relaciones entre ellos. Cada objeto en la red debe tener una referencia a los identificadores de sus objetos vecinos. El LPOO en que se basa el manejador debe soportar el acceso a los atributos de un objeto, así como a sus métodos, a través de

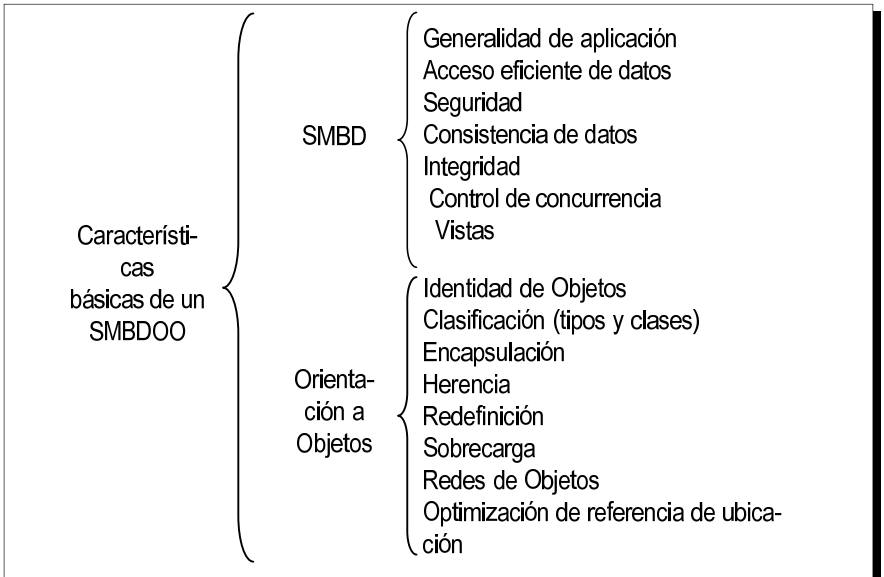

Figura 3. Características de un SMBDOO

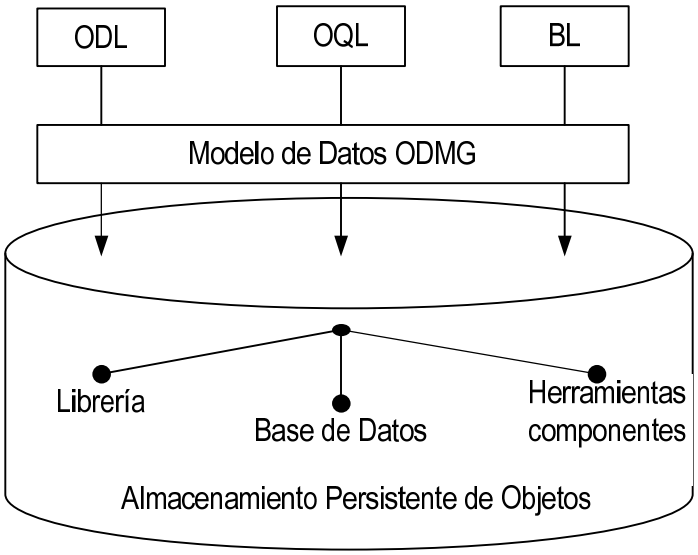

Figura 4. Arquitectura del estándar ODMG.

estas referencias. Comúnmente este soporte incluye el manejo de asociaciones con diferente cardinalidad (uno a uno, uno a muchos y muchos a muchos) basadas en los OID de los objetos implicados.

UBICACIÓN

Esta característica se refiere a la mejora en el desempeño de las tareas de almacenamiento y recuperación de objetos, conocida como optimización, la cual aprovecha las estructuras formadas de red. Esta optimización se desarrolla mediante tres técnicas, la de **generación de clusters en disco**, para almacenar objetos relacionados, la **técnica de bloqueo** (bloques de objetos), en cuanto al manejo en memoria, y la técnica del **manejo de cachés** de objetos, basada en las relaciones en que éstos se encuentran implicados.

En el campo de los SBDOO comerciales existen varias diferencias, que van desde la definición del concepto de Base de Datos Orientada a Objetos, hasta las características que ésta debe poseer. Ante esto, surge el Object Data Management Group (ODMG), organización de empresas desarrolladoras de SMBDOO, dedicada al establecimiento de estándares para BDOO.

Dado que los desarrollos comerciales propietarios no soportaban la portabilidad entre plataformas, se planteó el establecimiento de un estándar que creará un ambiente donde los SMBDOO compartiesen muchas características, similar a las interfaces SQL de los manejadores relacionales. Esta organización se ideó basándose en grupos de trabajo para cada capítulo del estándar a desarrollar, y las especificaciones adoptadas en cada grupo de trabajo debían pasar por un consenso final.

Algunos de los miembros de ODMG son: Sun Microsystems, Excellon Corp., Winward Solutions, Barry & Associates, American Management Systems, Wichita State University, in Line Software, Ardent Software, NEC, Hitachi, Lucent Technologies, Unidata, POET Software, Lockheed Martin, UniSQL, IBEX ObjectSystems, Object Design, GemStone Systems, Versant Object Technology, y MATISSE Software.

El primer resultado de este comité fue el estándar ODMG-93, cuya segunda versión se liberó en 1997, y la tercera versión, 3.0, se publicó en enero del 2000.

ARQUITECTURA DE UN SMBDOO

El estándar ODMG versión 3.0 establece una arquitectura, proporcionando algo más que un lenguaje de alto nivel como SQL para la definición, manipulación restringida y consulta de datos. Esta arquitectura en particular establece que un SMBDOO integra transparentemente las capacidades de una base de datos con las de un lenguaje de programación, dentro de un ambiente portable (figura 4). Esto es, la **definición de datos** está implícita en la construcción de la aplicación orientada a objetos, y la **manipulación** de datos se realiza a través de las operaciones del tipo.

El estándar coloca como componentes principales de esta arquitectura los siguientes: especificación de un repositorio de objetos, el modelo de objetos (Object Model) y los lenguajes de bases de datos tales como: lenguaje de especificación o definición de objetos (Object Definition Language, ODL), lenguaje de consulta (Object Query Language, OQL) y lenguaje de enlace o ligado (Language Binding).

A continuación se describen brevemente el modelo de objetos y los lenguajes de bases de datos.

MODELO DE OBJETOS (OM)

Es el componente central de la arquitectura, en virtud de que es la

jerarquía de clases en la que se basa el desarrollo de los esquemas de las aplicaciones, y la estructura de organización con la cual se almacena toda la información. ODMG soporta un Modelo de Objetos unificado para compartir datos entre diferentes LPOO. Todos los datos que maneja el SMBDOO están estructurados en términos de las construcciones del Modelo de Datos.

Lenguaje de Especificación o Definición de Objetos (ODL)

El programador debe especificar los esquemas de la aplicación y de la implementación de las entidades definidas. Los esquemas pueden desarrollarse con una extensión de la sintaxis del Lenguaje de Ligado con el que se esté trabajando (Language Binding), o con un ODL independiente del Lenguaje de Ligado, por ejemplo UML. Los esquemas se describen en términos del ODL, como un conjunto de interfaces de objetos, es decir, las propiedades y las relaciones de los tipos junto con las firmas de sus métodos. El Lenguaje de Especificación no es completamente un lenguaje de programación, dado que las implementaciones de los tipos definidos deben realizarse con el Lenguaje de Ligado.

Por lo tanto, se dice que el ODL es un lenguaje virtual, el cual no está implementado obligatoriamente en los productos basados en el estándar ODMG; en su lugar, es posible que tengan lenguajes de definición equivalentes, los que poseen la funcionalidad que ofrece un ODL, pero desarrollados a la medida de las especificaciones de un SMBDOO en particular.

Lenguaje de Ligado (Binding)

La implementación de los métodos definidos en los esquemas y las aplicaciones se lleva a cabo con un LPOO. Sin embargo, estos Lenguajes de Programación no son persistentes por naturaleza, así que es necesario extenderlos para que puedan interactuar con datos persistentes. Es entonces cuando se interactúa con el Lenguaje de Ligado, que es el puente de integración entre un LPOO y el Modelo de Objetos de la arquitectura del SMBDOO.

El tipo de extensiones que permiten que un Lenguaje de Programación Orientado a Objetos trabaje con Objetos Persistentes son construcciones adicionales al lenguaje o bibliotecas, las cuales proporciona el Lenguaje de Ligado específico para ese lenguaje de programación. El SMBDOO puede proporcionar mas de un Lenguaje de Ligado con el fin de usar diferentes LPOO como Lenguajes de Manipulación de sus datos. El estándar ODMG 3.0, establece el Lenguaje de Ligado para Smalltalk, C++ y Java. Un SMBDOO sólo puede trabajar con un Lenguaje de Ligado a la vez; dicho lenguaje está diseñado a la medida de un LPOO específico, con el fin de proporcionar un ambiente de programación único e integrado para la manipulación de datos. Los esquemas y la implementación se compilan y ligan junto al SMBDOO para producir la aplicación ejecutable. Esta aplicación accesa una base de datos, los cuales corresponden a los tipos declarados en los esquemas, y los manipula mediante el paso de mensajes a los métodos definidos e implementados para cada clase.

El acceso a las bases de datos puede ser compartido entre varias aplicaciones, debido a que el SMBDOO proporciona un servicio concurrente con base en transacciones y manejo de bloqueos.

Lenguaje de Consulta de Objetos (OQL)

OQL es el Lenguaje de Consulta que forma parte del estándar ODMG. Es similar al SQL en el sentido de que puede utilizarse como un Lenguaje de Consulta por si solo o puede estar integrado en un LPOO, pero a diferencia de SQL, OQL sólo contiene comandos de consulta y carece de comandos para definición o manipulación de datos. Este lenguaje es de uso común en los diferentes lenguajes de programación con los que puede trabajar un SMBDOO.

A pesar del potencial de los SMBDOO ante las aplicaciones complejas o nuevas, aún presentan varios problemas, los cuales abarcan diversas líneas de investigación:

La Migración de Datos

Un gran número de empresas en el mundo mantienen actualmente cantidades considerables invertidas en Sistemas de Bases de Datos, Sistemas Expertos y Sistemas de Cómputo, principalmente en Datos Heredados (Data Legacy), siendo éstos últimos los datos sobre los que corren las aplicaciones ya mencionadas; esto es, los datos históricos de la empresa que han sido almacenados y procesados en el transcurso de los años y donde reside el valor de una institución: su información. Por lo tanto, es natural que a pesar de que la compañía decida migrar hacia una nueva plataforma, será necesario tomar en cuenta los costos de transferencia o compatibilidad de tal plataforma con los datos existentes; esta problemática generó la línea de investigación denominada Migración de Datos. De entre los resultados hasta ahora obtenidos para dar solución a esta

problemática se encuentran las envolturas (wrappers), para manejadores relacionales, y el mapeo objeto relacional, entre otros.

Complejidad del Modelo Orientado a Objetos

Mas de 20 años de investigación en los SMBD basados en registros les han otorgado una alta eficiencia, a diferencia de los SMBDOO, los cuales reflejan una tecnología inmadura. A pesar de que los SMBDOO hacen uso de técnicas desarrolladas especialmente para los SMBD basados en registros, es necesario desarrollar líneas de investigación especiales para cubrir las necesidades particulares de esta tecnología.

Complejidad del Modelo Orientado a Objetos con Respecto al de Registros

Una jerarquía de clases diseñada apropiadamente implica un bajo costo de mantenimiento. Sin embargo, esta jerarquía no se puede modificar fácilmente, sobre todo si ya se han creado objetos persistentes. Es por ello que se debe tener un gran cuidado al identificar las clases y las relaciones entre ellas, de tal forma que todas ellas se identifiquen plenamente.

El tiempo dedicado a las fases de análisis y diseño en una BDOO es mucho mayor que el de una BD relacional, aunque se ve compensado por los costos de mantenimiento mucho menores. Otro aspecto de la complejidad es encontrar personal capacitado y experimentado para estos desarrollos.

Falta de Estándares

Es provocada por la motivación natural de los vendedores para mantener productos propietarios, situación muy común en la industria del software, donde los vendedores de bases de datos se esfuerzan por asegurar su clientela de tal forma que no le sea fácil migrar el procesamiento de datos a otros manejadores.

El término Modelo de Datos Orientado a Objetos no implica una definición de estructuras universalmente aceptadas. La Tecnología de Objetos cuenta con diferentes metodologías de soporte para las etapas de análisis y diseño de sistemas de cómputo; quizá las cuatro más populares son las de Rumbaugh, Booch, Coard-Yourdan y la de Schaer-Mellor. En la mayoría de los casos, estas metodologías son muy similares entre sí, teniendo cada una su manera de representar gráficamente las entidades y aplicar los principios del paradigma. Sin embargo, es posible integrar conceptos de un método con conceptos de otra técnica, creando una metodología mixta; ante esta problemática, surge el Unified Modeling Language (UML) que es un lenguaje para especificar, visualizar, construir y documentar los componentes de sistemas de software.

El UML representa una colección de las mejores prácticas de ingeniería (Booch, OMT y OOSE) que han probado su éxito en el modelado de sistemas grandes y complejos, adoptándose como un estándar OMG. Debido a la falta de estándares para arquitecturas de BDOO, la mayoría de los vendedores de SMBDOO han creado productos propietarios, los cuales difieren bastante entre si, principalmente en lo referente a sus estructuras internas.

Conclusiones

El empleo de los SMBD relacionales ha tenido sin duda un gran éxito en el manejo de datos sencillos, de sistemas tales como: nómina, compras, ventas, almacén, renta de automóviles, reservaciones de vuelos, etc. Por otro lado, el adaptar este modelo relacional al manejo de estructuras complejas de datos, resultado del avance tecnológico en la computación y de las nuevas líneas de investigación en Bases de Datos, ha sido problemático e inadecuado; debido principalmente a que su modelo simple de datos carece de un lenguaje de programación completamente computacional, que permita tanto el acceso directo a los datos como la integración de código y datos dentro del control de almacenamiento del SMBD.

Esto ha dado pauta a la investigación de nuevos paradigmas, modelos, técnicas, estándares de modelado y lenguajes, etc., para aplicaciones avanzadas que permitan manipular datos o información compleja, tales como: datos multimedia, requerimientos para múltiples versiones de datos, bases de datos distribuidas en ambiente de redes e Internet, datos espaciales y temporales, habilidad para manejar tipos de restricciones más poderosas, al igual que la integración de persistencia con tipos de datos complejos; es decir, la necesidad apremiante de integrar de una mejor forma código y datos, siendo esto último lo mas importante.

Por lo mismo, las Bases de Datos Orientadas a Objetos (BDOO) representan el paso siguiente en esta evolución, para integrar el análisis, diseño y programación Orientada a Objetos; además, permiten el desarrollo y mantenimiento de aplicaciones complejas con un costo menor, haciendo posible que el mismo modelo concep-

tual se aplique al análisis, diseño, programación, definición y acceso a la base de datos, reduciendo el problema de traducción entre diferentes modelos a través de todo el ciclo de vida de los sistemas.

Todo este potencial, brindado por el Modelo de Datos Orientado a Objetos, el cual extiende el modelo semántico para integrar código con la manipulación de datos en una sola estructura, es empleado por los Sistemas Manejadores de Bases de Datos Orientados a Objetos (SMBDOO), como un modelo lógico para el SMBD o como un LPOO persistente.

Por otro lado, este modelo aun joven está teniendo éxito con resultados satisfactorios en la atención de las nuevas necesidades, aún cuando presenta algunas limitantes tales como: inmadurez, la necesidad de innovación de las técnicas y aspectos del manejo de datos complejos, la escasez de diseñadores y desarrolladores con experiencia, la falta de apego a los estándares por parte de fabricantes y desarrolladores de software y el rendimiento limitado de la computadora en aplicaciones de tiempo real, entre otros.

Las Bases Orientadas a Objetos coexistirán con las Bases de Datos Relacionales durante los próximos años, puesto que se utiliza un modelo relacional como forma de estructura de datos dentro de las BDOO.

Bibliografía

[1] Cooper, Richard. "Object Databases An ODMG Approach". Thomson Computers Press. ITP An International Thomson Publishing Company. 1997.

[2] Catell, R.G.G.. Barry K. Douglas. "The Object Data Standard: ODMG 3.0". Morgan Kaufmann Publishers, an Imprint of Academic Press. 2000.

[3] Eckel, Bruce. "Thinking in Java". Second edition. Prentice Hall PTR. 2000.

[4] Martínez Prieto, Ana Belén. Tesis para obtener el grado de Doctor en Informática. "Un sistema de Gestión de Bases de Datos Orientadas a Objetos Sobre una Máquina Abstracta Persistente". Universidad de Oviedo, Oviedo. Mayo de 2001.

[5] Elmasri Rarmez, Navathe Shamkant B. "Sistemas de Bases de Datos Conceptos Fundamentales". Segunda edición. Addison-Wesley Iberoamericana. 1998.