Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

Gestión de Datos

Bases de Datos Orientadas a Objetos

Ing. Enrique Reinosa
Septiembre 2007

BASES DE DATOS ORIENTADAS A OBJETOS (BDOO)

INTRODUCCION

En este Capitulo hablaremos de la evolución de los diferentes tipos de Bases de Datos y por consiguiente del surgimiento de las Bases de Datos Orientadas a Objetos (BDOO). Las BDOO almacenan y manipulan información que puede ser digitalizada (representada) por objetos, proporcionan una estructura flexible con acceso ágil, rápido, con gran capacidad de modificación.

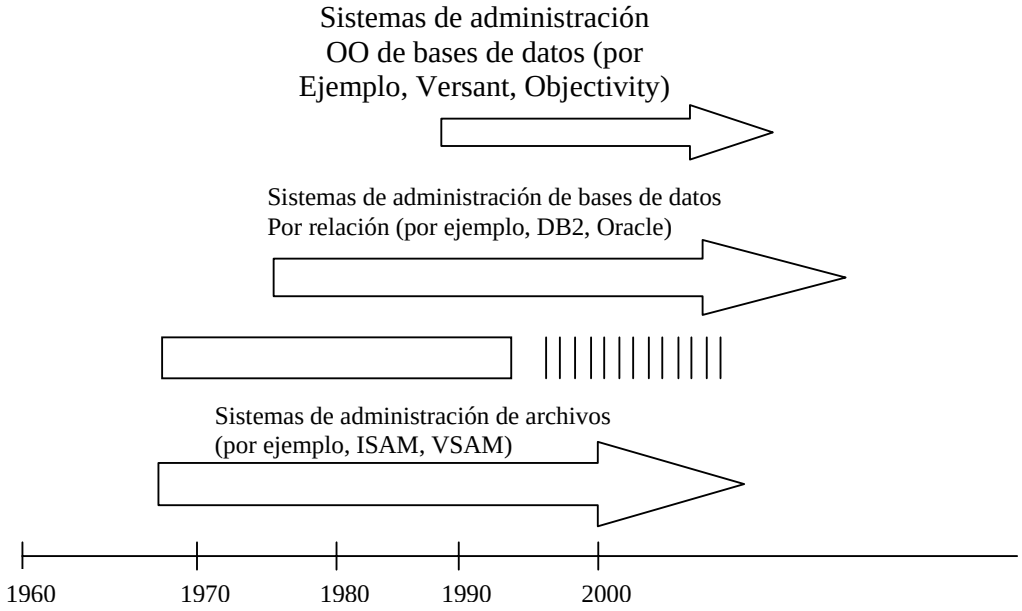
Además combina las mejores cualidades de los archivos planos, las bases jerárquicas y relacionales. Como veremos a continuación las BDOO representan el siguiente paso en la evolución de las Bases de Datos para soportar el análisis, diseño y programación Orientada a Objetos.

Estás permiten el desarrollo y mantenimiento de aplicaciones complejas ya que se puede utilizar un mismo modelo conceptual y así aplicarlo al análisis, diseño y programación, esto reduce el problema entre los diferentes modelos atrávez de todo el ciclo de vida, con un costo significativamente menor.

Como cualquier base de datos programable, una base de datos orientada a objetos (BDOO) da un ambiente para el desarrollo de aplicaciones con un depósito persistente listo para su explotación, por otra parte permiten que el mismo modelo conceptual se aplique al análisis, diseño, programación, definición y acceso a la base de datos. Esto reduce el problema del operador de traducción entre los diferentes modelos a través de todo el ciclo de vida. El modelo conceptual debe ser la base de las herramientas CASE OO totalmente integradas, las cuales ayudan a generar la estructura de datos y los métodos.

Además las BDOO ofrecen un mejor rendimiento de la máquina que las bases de datos por relación, para aplicaciones ó clases con estructuras complejas de datos. Sin embargo, las BDOO coexistirán con las bases de datos por relación como una forma de estructura de datos dentro de una BDOO.

(Figura No.1)
Breve Historia del Desarrollo de las Bases de Datos



Como se muestra en la (Figura No.1), cuatro generaciones de sistemas han manejado datos de computación. Al principio, los lenguajes y las instrucciones de máquina eran muy similares, lo que producía un modelo de programación orientado por procesos. Por ejemplo, los programas para la suma se organizaban en torno al proceso de suma de la máquina: los números se cargaban en registros, se ejecutaba la instrucción de suma y se trabajaban los posibles errores de desbordamiento superior ó inferior. Algunos resultados se almacenaban para su uso posterior. En principio los programas ejecutaban las tareas y nunca las escribían en un dispositivo de almacenamiento. En está etapa, uno de los pocos elementos que se almacenaban era el propio programa. Sin embargo, los programadores pronto se dieron cuenta del valor de registrar los resultados. La grabación de los resultados del programa aumentó con el advenimiento del almacenamiento en discos magnéticos rotatorios, lo que ofreció la posibilidad del acceso aleatorio a grandes cantidades de datos almacenados.

1.- CONCEPTOS FUNDAMENTALES

A efectos de estandarizar la terminología utilizada en resto del capítulo y capítulos posteriores definiremos a continuación términos básicos y conceptos fundamentales de la Programación Orientada a Objetos (POO) y su aplicación a BDOO.

Objeto: es cualquier cosa real ó abstracta acerca de la cual almacenamos datos y los métodos que controlan dichos datos. Por ejem. En una empresa EMPLEADO se aplica a los objetos que son personas empleadas por alguna organización alguna INSTANCIA podría ser Juan Pérez, María Sánchez etc.

Tipo de Objeto: es una categoría de objeto. Ejem: EMPLEADO. Un objeto es una Instancia de un tipo de objeto. PERSONA (Juan Pérez)

Encapsulado: es el resultado o acto de ocultar los detalles de implantación de un objeto respecto de su usuario.

Una Solicitud: invoca una operación específica, con uno ó más objetos como parámetros. Es decir, es para que se lleve acabo la operación indicada y que se produzca el resultado. En consecuencia las implantaciones se refieren a los objetos como solicitudes.

Clase: es una implantación de un tipo de objetos. Especifica una estructura de datos y los métodos operativos permisibles que se aplican a cada uno de sus objetos.

Herencia: Una clase implanta el tipo de objeto. Una Subclase hereda propiedades de su clase padre, una subclase puede heredar la estructura y los métodos ó algunos de los métodos.
En las BDOO los datos están encapsulados y se dice que estos son activos más que pasivos; debido a que por ejemplo: La clase mayor detecta si tiene un hijo (objeto) más o uno menos, es por esto que se dice que están activos ya que cuentan los hijos u objetos que tiene.
En el modelo de objetos existen cuatro características fundamentales:

Abstracción: denota las características esenciales de un objeto que lo distinguen de todos los demás tipos objeto, y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador". Una abstracción se centra en la visión externa de un objeto, y, por tanto sirve para separar el comportamiento esencial de un objeto de su implantación.

Modularidad: Se basa en el concepto de fragmentación de los programas en componentes individuales para reducir su complejidad en algún grado, y para crear además una serie de fronteras bien definidas y documentadas dentro del programa, dónde estas fronteras o interfaces tienen un incalculable valor cara a la comprensión del programa.

Jerarquía: una clasificación u ordenación de abstracciones.

Tipos: Es un conjunto de objetos que tienen un mismo comportamiento (comparten una misma funcionalidad) que se puede observar desde afuera.

Genericidad: permite construir clases genéricas para otras clases.

Objetos Complejos: Están construidos mediante algunos más simples ó mediante la aplicación de constructores a ellos. Los Objetos más simples son objetos como: Integer, Carácter, String de Bytes de cualquier longitud, booleanos ó punto flotante y algunos pueden ser de tipo atómico. Hay varios constructores de objetos complejos como son Listas y arreglos. De este modo el juego mínimo de constructores que el sistema debe tener son una lista y un Arreglo.
Las listas y arreglos son importantes porque, pueden capturar ordenes las cuales ocurren en el mundo real y también se pueden levantar en muchas especificaciones científicas donde las necesidades de la gente son matrices, series de tiempo de información ó datos. El objeto de constructores debe ser ortogonal cualquier constructor debe ser aplicado a cualquier objeto.

Identidad de Objetos: Un modelo significa en un modelo una identidad de objeto. El objeto tiene una existencia la cual es independiente de su valor, esto es dos nociones de equivalencia del objeto. Dos objetos pueden ser idénticos, que tengan el mismo objeto ó pueden ser iguales, que tengan el mismo valor; este tiene dos implicaciones una es la compartición del objeto y la otra es la actualización del objeto.

Compartimiento de Objetos: Es un modelo basado en la identidad de dos objetos contener ó compartir un componente la representación pictórica de un objeto complejo es una gráfica mientras que están limitadas en un árbol sin identidad de objeto.
Considere el siguiente ejemplo: Una persona tiene un nombre y una edad y un juego de niños. Asumiendo que Pedro y Susana ambos tienen un niño de 15 años de edad llamado Juan en la vida real estas dos situaciones pueden levantarse para presentarse como: Susana y Pedro son padres del mismo niño ó dos niños están envueltos por:

(Pedro,40,{(Juan,15,{})}) Y Susana es representada por:(Susana,41,{(Juan,15,{})})

Es decir, no hay forma de expresar que Pedro y Susana son padres del mismo niño en un modelo basado en la identidad estas dos estructuras pueden compartir una parte común Juan 15 ó no. Esto es capturando ambas situaciones.
Asumiendo que Pedro y Susana son obviamente padres de un niño llamado Juan en este caso todas las actualizaciones al hijo de Juan van a ser aplicadas al objeto de Juan y consecuentemente también aplicadas al hijo de Pedro.

2.- BASES DE DATOS ORIENTADAS A OBJETOS (BDOO)

2.1 ¿Qué es Orientación a Objetos (OO)?

En esos mundos OO, el conocimiento se descentraliza en todos los objetos que lo componen, cada objeto sabe hacer lo suyo y no le interesa saber cómo el vecino hace su trabajo, pero sabe que lo hace y qué es lo que puede hacer. Como bien lo definió Dan Ingalls de Smalltalk con las siguientes palabras:

"La orientación a objetos proporciona una solución que conduce a un Universo de Objetos 'bien educados' que se piden de manera cortés, concederse mutuamente sus deseos".

2.2 ¿Por qué OO?

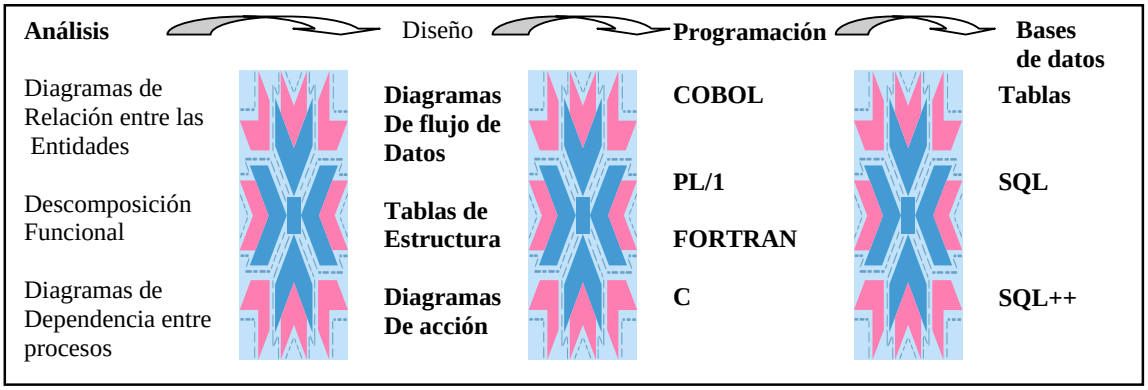
La meta es dejar la etapa en la que la construcción del software es una labor de artesanos, y pasar a la etapa en la que se pueda tener fábricas de software, con gran capacidad de reutilización de código y con metodologías eficientes y efectivas que se apliquen al proceso de producción.

2.3 ¿Qué es una BDOO?

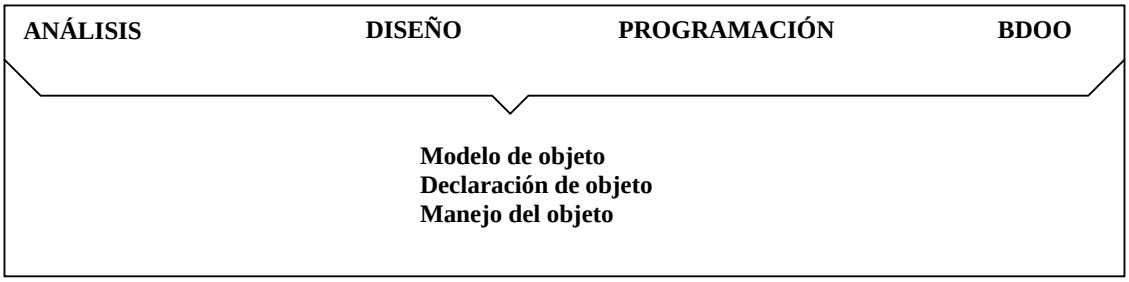
A finales de los 80's aparecieron las primeras BDOO, es una base de datos inteligente. Soporta el paradigma orientado a objetos almacenando datos y métodos, y no sólo datos. Está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Evita el acceso a los datos; esto es mediante los métodos almacenados en ella. Es más segura ya que no permite tener acceso a los datos (objetos); esto debido a que para poder entrar se tiene que hacer por los métodos que haya utilizado el programador.

2.4 Un Modelo Conceptual Unificado

Las técnicas OO utilizan los mismos modelos conceptuales para el análisis, diseño y construcción. La tecnología de las BDOO da un paso más hacia la unificación, el modelo conceptual de la base de datos OO es igual al del resto del mundo OO, en lugar de utilizar tablas por relación independientes como SQL.
El uso del mismo modelo conceptual para todos los aspectos del desarrollo simplifica éste, particularmente con las herramientas CASE OO; mejora la comunicación entre usuarios, analistas y programadores, además de que reduce las posibilidades de error.



(Figura No.2)
El desarrollo tradicional tiene cuatro modelos conceptuales.



(Figura No.3)
La Tecnología orientada a objetos utiliza un modelo consistente

2.5 Arquitectura de Una BDOO

En la siguiente imagen (Tabla Nro.: 1) se muestran algunos de los principales productos de BDOO y sus vendedores. Los primeros se diseñaron como una extensión de los lenguajes de programación como Smalltalk ó C++. El LMD (lenguaje para el manipulación de datos; también conocido como DML) y el LDD (lenguaje para la definición de los datos; también conocido como DDL) construían un lenguaje OO común.
El diseño de las BDOO actuales debe aprovechar al máximo él CASE e incorporar métodos creados con cualquier técnica poderosa, incluyendo enunciados declarativos, generadores de códigos e inferencias con base en reglas.

(Tabla No.1)

Seis Productos de BDOO y sus Proveedores.

Producto	Proveedor
Gemstone	Servio Corporation, Alameda,CA
Itasca	Itasca Systems,Inc.,Minneapolis,MN
Objectivity	Objectivity,Menlo Park,Ca
Object Store	Object Design,Inc.,Burlington,MA
Ontos	Ontos Inc.,Bellerica,MA
Versant	Versant Object Technology,Menlo Park,CA

Algunas de las siguientes características son independientes de la arquitectura fundamental de una BDOO pero son comunes a la mayoría de ellas:

- ***Versiones:*** La mayoría de los sistemas de bases de datos sólo permiten que exista una representación de un ente de la base de datos dentro de esta. Las versiones permiten que las representaciones alternas existan en forma simultánea.

- ***Transacciones compartidas:*** Las transacciones compartidas soportan grupos de usuarios en estaciones de trabajo, los cuales desean coordinar sus esfuerzos en tiempo real, los usuarios pueden compartir los resultados intermedios de una base de datos. La transacción compartida permite que varias personas intervengan en una sola transacción

2.6 Desarrollo con Bases de Datos OO

Las BDOO se desarrollan al describir en primer lugar los tipos de objetos importantes del dominio de aquellos tipos de objetos. Estos tipos de objetos determinan las clases que conformarán la definición de la BDOO.
Por ejemplo: Una base de datos diseñada para almacenar la geometría de ciertas partes mecánicas incluiría clases como CILINDRO, ESFERA Y CUBO. El comportamiento de CILINDRO podría incluir información relativa a sus dimensiones, volumen área superficial:

Clase de CILINDRO{
ALTURA FLOTANTE ();
RADIO FLOTANTE ();
VOLUMEN FLOTANTE ();
AREA DE SUPERFICIE FLOTANTE ();
};

Se puede llegar a definiciones similares para el cubo y la esfera. En la definición anterior, ALTURA,RADIO y ÁREA representan los mensajes que se pueden enviar a un objeto CILINDRO. La Implementación se lleva a cabo en el mismo lenguaje, escribiendo funciones correspondientes a las solicitudes OO:

CILINDRO::ALTURA () {RETORNA CILINDRO_ALTURA;}
CILINDRO::VOLUMEN () {RETORNA PI*RADIO ()*ALTURA ();}

En este caso, la Altura se almacena como un elemento de los datos, mientras que volume se calcula mediante la fórmula apropiada. Observe que la implantación interna de volume utiliza solicitudes para obtener altura y radio. Sin embargo, el aspecto más importante es la sencillez y uniformidad que experimentan los usuarios de CILINDRO. Sólo necesitan conocer la forma de enviar una solicitud y las solicitudes disponibles.

2.7 Tres Enfoques de Construcción de Bases de Datos OO

Las BDOO se pueden construir mediante alguno de los tres enfoques siguientes:

- ***El Primero:*** se puede utilizar el código actual altamente complejo de los sistemas de administración de las bases de datos, de modo que una BDOO se implante más rápido sin tener que iniciar de cero. Las técnicas orientadas a objetos se pueden utilizar como medios para el diseño sencillo de sistemas complejos. Los sistemas se construyen a partir de componentes ya probados con un formato definido para las solicitudes de las operaciones del componente.

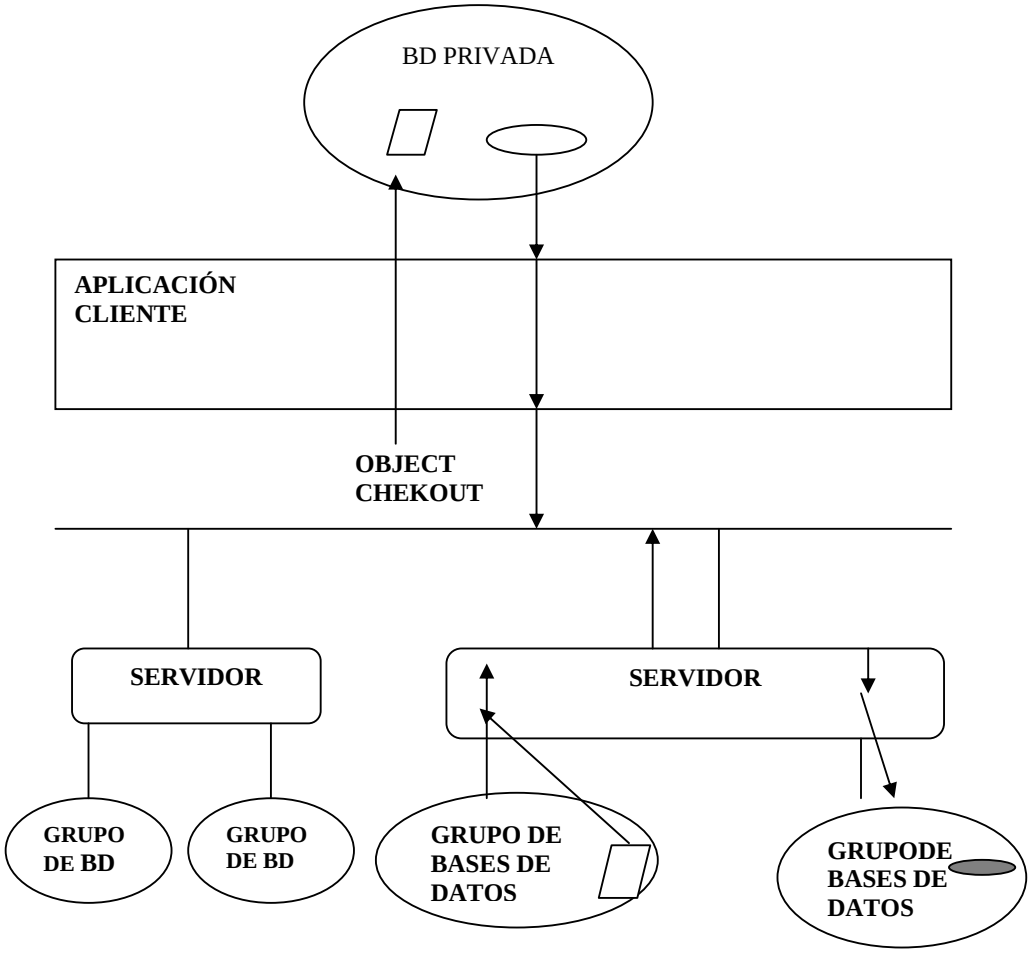
- ***El Segundo:*** considera a la BDOO como una extensión de la tecnología de las bases de datos por relación. De este modo, las herramientas, técnicas, y vasta experiencia de la tecnología por relación se utilizan para construir un nuevo SABD. Se pueden añadir apuntadores a las tablas de relación para ligarlas con objetos binarios de gran tamaño (BLOB). La base de datos también debe proporcionar a las aplicaciones clientes un acceso aleatorio y por partes a grandes objetos, con el fin de que sólo sea necesario recuperar a través de la red la parte solicitada de los datos.

- ***El Tercero:*** reflexiona sobre la arquitectura de los sistemas de bases de datos y produce una nueva arquitectura optimizada, que cumple las necesidades de la tecnología OO. Las compañías como Versant, Objectivity, Itasca, etc. Utilizan esté enfoque y afirman que la tecnología de relación es un subconjunto de una capacidad más general. Además que las BDOO no de relación son aproximadamente dos veces más rápidas que las bases de datos por relación para almacenar y recuperar la información compleja. Por lo tanto, son esenciales en aplicaciones como CAD y permitirían que un depósito CASE fuera una facilidad de tiempo real en vez de una facilidad por lotes.

La Arquitectura de Versant está designada al soporte Cliente/Servidor con acercamiento a la computación distribuida; cualquier aplicación de Cliente el servidor la procesa, usa las EDT y las máquinas servidoras que pueden cooperar en una BD distribuida de Versant. Las BD pueden estar levantadas como un sistema m-Cliente/n-Servidor.

Un servidor en el medioambiente de Versant es una máquina que está corriendo los procesos del servidor, esta soporta accesos concurrentes por usuarios múltiples de una o más BD. Un cliente es un proceso de aplicación este tiene acceso a espacios de trabajo de BD persistentes privadas y en adición puede accesar diversas BD sobre servidores concurrentes con otras aplicaciones de cliente.

Fig. No. 4
Arquitectura de Versant



2.8 Impacto de la Orientación a Objetos en la Ingeniería del Software.

En las BDOO, la organización “Gestión Manejadora de Datos Objeto (ODMG)” representa el 100% de las BDOO industriales y ha establecido un estándar de definición (ODL - Lenguaje de Definición de datos) y manipulación (OQL - Lenguaje de consulta) de bases de datos equivalente a SQL.

Respecto a las relacionales, todas (Oracle, Informix, etc.) están añadiendo en mayor o menor grado algunos aspectos de la orientación a objetos. ANSI(Instituto Nacional Estadounidense de Estándar), por su parte, está definiendo un SQL-3 que incorpora muchos aspectos de la orientación a objetos. El futuro del SQL-3 es sin embargo incierto, ya que ODMG ha ofrecido a ANSI su estándar para que sirva de base para un nuevo SQL, con lo que solo habría un único estándar de base de datos.

El grupo ODMG (Grupo Manejador de Datos Objeto) nació de un grupo más grande, llamado “Grupo Manejador de Objetos (OMG)”, donde están representados todas las cosas con alguna influencia en el sector. Este grupo esta definiendo un estándar universal por objetos. Este estándar permitirá que un objeto sea programado en cualquier lenguaje y sistema operativo. Esto facilitará enormemente el desarrollo de sistemas abiertos cliente-servidor.

2.9 Ventajas en BDOO’s

Está su flexibilidad, y soporte para el manejo de tipos de datos complejos. Por ejemplo, en una base de datos convencional, si una empresa adquiere varios clientes por referencia de clientes servicio, pero la base de datos existente, que mantiene la información de clientes y sus compras, no tiene un campo para registrar quién proporcionó la referencia, de qué manera fue dicho contacto, o si debe compensarse con una comisión, sería necesario reestructurar la base de datos para añadir este tipo de modificaciones. Por el contrario, en una BDOO, el usuario puede añadir una "subclase" de la clase de clientes para manejar las modificaciones que representan los clientes por referencia.

La subclase heredará todos los atributos, características de la definición original, además se especializará en especificar los nuevos campos que se requieren así como los métodos para manipular solamente estos campos. Naturalmente se generan los espacios para almacenar la información adicional de los nuevos campos. Esto presenta la ventaja adicional que una BDOO puede ajustarse a usar siempre el espacio de los campos que son necesarios, eliminando espacio desperdiciado en registros con campos que nunca usan.
La segunda ventaja de una BDOO, es que manipula datos complejos en forma rápida y ágilmente. La estructura de la base de datos está dada por referencias (o apuntadores lógicos) entre objetos.

2.10 Posibles Desventajas

Al considerar la adopción de la tecnología orientada a objetos, la inmadurez del mercado de BDOO constituye una posible fuente de problemas por lo que debe analizarse con detalle la presencia en el mercado del proveedor para adoptar su producto en una línea de producción sustantiva. Por eso, en este apartado se propone que se explore esta tecnología en un proyecto piloto.

El segundo problema es la falta de estándares en la industria orientada a objetos. Sin embargo, el "Grupo Manejador de Objetos" (OMG), es una organización Internacional de proveedores de sistemas de información y usuarios dedicada a promover estándares para el desarrollo de aplicaciones y sistemas orientados a objetos en ambientes de cómputo en red. La implantación de una nueva tecnología requiere que los usuarios iniciales acepten cierto riesgo. Aquellos que esperan resultados a corto plazo y con un costo reducido quedarán desilusionados. Sin embargo, para aquellos usuarios que planean a un futuro intermedio con una visión tecnológica avanzada, el uso de tecnología avanzada, el uso de tecnología orientada a objetos, paulatinamente compensará todos los riesgos.

2.11 Rendimiento

- Las BDOO permiten que los objetos hagan referencia directamente a otro mediante ***apuntadores suaves***. Esto hace que las BDOO pasen más rápido del objeto A al objeto B que las BDR, las cuales deben utilizar comandos JOIN para lograr esto. Incluso el JOIN optimizado es más lento que un recorrido de los objetos. Así, incluso sin alguna afinación especial, una BDOO es en general más rápida en esta mecánica de caza-apuntadores.

- Las BDOO hacen que el agrupamiento sea más eficiente. La mayoría de los sistemas de bases de datos permiten que el operador coloque cerca las estructuras relacionadas entre sí, en el espacio de almacenamiento en disco. Esto reduce en forma radical el tiempo de recuperación de los datos relacionados, puesto que todos los datos se leen con una lectura de disco en vez de varias. Sin embargo, en una BDR, los objetos de la implantación se traducen en representaciones tabulares que generalmente se dispersan en varias tablas. Así, en una BDR, estos renglones relacionados deben quedar agrupados, de modo que todo el objeto se pueda recuperar mediante una única lectura del disco. Esto es automático en una BDOO. Además, el agrupamiento de los datos relacionados, como todas las subpartes de un ensamble, puede afectar radicalmente el rendimiento general de una aplicación. Esto es relativamente directo en una BDOO, puesto que representa el primer nivel de agrupamiento. Por el contrario, el agrupamiento físico es imposible en una BDR, puesto que esto requiere un segundo nivel de agrupamiento: un nivel para agrupar las hileras que representan a los objetos individuales y un segundo para los grupos de hileras que representan a los objetos relacionados.

2.12 Características mandatorias ó reglas de oro

Un sistema de BDOO debe satisfacer dos criterios:

- ***Debe tener un BDMS***
- ***Debe ser un sistema OO***

Por ejemplo: para la extensión posible este debe ser consistente en los actuales cortes de lenguajes de programación OO.

El primer criterio se traduce en 5 características como son: Persistencia, Manejador de almacenamiento secundario, Concurrencia, Recuperación, y Facilidad de Query.

La Segunda se traduce en 8 características: Objetos Complejos, Identidad del objeto, Encapsulamiento, Tipos ó Clases, Sobre paso con combinación retrasada, Extensibilidad y Completado Computacional.

2.12.1 Manifiesto de sistema de gestión de BDOO

Esta publicación intenta definir un sistema de BDOO y describe las principales características. Hemos separado estas características en 3 grupos:

- ***Mandatarias:*** son las que el Sistema debe satisfacer a orden de tener un sistema de BDOO y estos son: Objetos complejos, Identidad de objetos, Encapsulación, Tipos ó Clases, Sobre paso combinado con unión retardada, Extensibilidad, Completación Computacional, Persistencia y Manejador de almacenamiento secundario, Concurrencia, Recuperación y Facilidad de Query.

- ***Opcionales:*** son las que pueden ser añadidas para hacer el sistema mejor pero que no son Mandatarias estas son de: herencia múltiple, chequeo de tipos e inferencia distribución y diseño de transacciones y versiones.

- ***Abiertas:*** son los puntos donde el diseñador puede hacer un número de opciones y estas son el paradigma de la programación la representación del sistema ó el tipo de sistema y su uniformidad. Hemos tomado una posición no muy a la expectativa para tener una palabra final más bien para proveer un punto de orientación para un debate futuro.

2.12.2 Características obligatorias

Este es un punto que no debe faltar en una BD.

- ***Predominancia combinada con enlace retardado:*** se puede definir que sea Excel, Autocad, etc. desde la programación.

- ***Extensibilidad:*** proporciona los tipos de datos como: Caracter, booleano, String, etc.

- ***Concurrencia:*** permite que varios usuarios tengan acceso a una BD al mismo tiempo.

- ***Recuperación:*** cuando se hace una transacción pero no se puede realizar y se regresa al mismo estado.

- ***Facilidad de "Consultas a Modo":*** esto es que se tienen diferentes estándares.

2.12.3 Características opcionales

Esta depende del producto que se vaya a realizar.

- ***Herencia Múltiple:*** tienen características de padres diferentes y proporcionan mecanismos para saber de 2 o más opciones cual conviene.

  - ✓ Verificación de tipos de inferencia.
  - ✓ Distribución: que se puede tener parte de una BD en un servidor y otra parte en otro.
  - ✓ Sistemas de Representación: forma en como se presentan los esquemas.
  - ✓ Uniformidad: todo debe ser igual. Diseño de ventanas, etc.
  - ✓ Asociaciones y Cardinalidad de Asociaciones: Cardinalidad: 1:1 (Uno a Uno), 1:M (Uno a Muchos), M:1 (Muchos a Uno), M:M (Muchos a Muchos).

2.12.4 Características abiertas

Es como si fuera una especialización con cierta marca de software.

2.12.5 Características generales

- ***Control de Concurrencia***

  - ✓ Modo Pesimista.- Tomo el dato no dejo que nadie lo tome para que no accedan al mismo dato.
  - ✓ Modo Optimista.- Es el que dice yo le saco una copia y piensa nadie lo va a modificar.
  - ✓ Modo Mixto.- Combinación del Pesimista y el Optimista.
  - ✓ Modo Semioptimista.- Toma las virtudes del Optimista.

2.12.6 Los sistemas de BDOO te proporciona el bloqueo.

- ***Bloqueos***

  - ✓ Bloqueos de Lectura.- Leer una dato y que no quieres que nadie lo modifique mientras los estas usando.
  - ✓ Bloqueos de Escritura.- Bloquear el Objeto mientras yo estoy escribiendo(nadie más puede escribir).
  - ✓ Bloqueos Nulos.- Es para Sincronización.(ejem. ”papel Out”notificación de una impresora).
  - ✓ Bloqueos de Notificación.- Es para Sincronización.(ejem.”papel Out” notificación de una impresora).

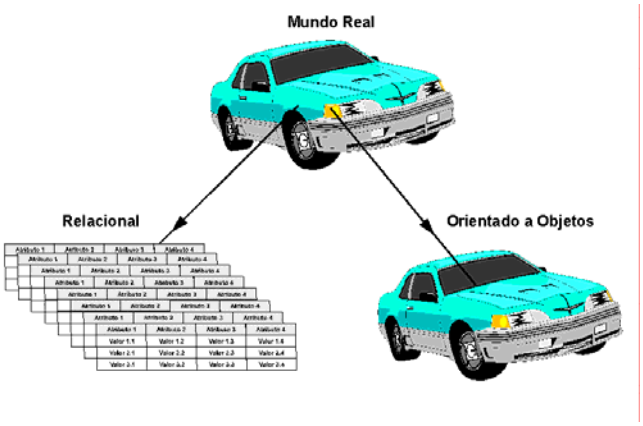
3. SISTEMA DE GESTION DE BDOO (SGBDOO)

Las bases de datos de objetos están diseñadas para simplificar la programación orientada a objetos. Almacenan los objetos directamente en la base de datos, y emplean las mismas estructuras y relaciones que los lenguajes de programación orientados a objetos.

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de datos relacionados entre sí y un grupo de programas para tener acceso a esos datos.

Un Sistema de Gestión de Bases de Datos Orientadas a Objetos (SGBDOO) se puede decir que es un SGBD que almacena objetos, permitiendo concurrencia, recuperación. Para los usuarios tradicionales de bases de datos, esto quiere decir que pueden tratar directamente con objetos, no teniendo que hacer la traducción a registros o tablas.

Fig. No. 5



Las bases de datos tradicionales almacenan sólo datos, mientras que las bases de datos orientadas a objetos almacenan objetos, con una estructura arbitraria y un comportamiento. Una simple metáfora (Esther Dyson) ayuda a ilustrar la diferencia entre ambos modelos. Consideremos el problema de almacenar un coche en un garaje al final del día. En un sistema de objetos el coche es un objeto, el garaje es un objeto, y hay una operación simple que es almacenar-coche-en-garaje. En un sistema relacional, todos los datos deben ser traducidos a tablas, de esta forma el coche debe ser desarmado, y todos los pistones almacenados en una tabla, todas las ruedas en otra, etc. Por la mañana, antes de irse a trabajar hay que componer de nuevo el coche para poder conducir (problema: al componer piezas puede salir una moto en vez de un coche).

3.1 Características de los SGBDOO

Un SGBDOO debe satisfacer dos criterios: Ser un sistema orientado a objetos, y ser un sistema de gestión de bases de datos. El primer criterio se traduce en ocho características generales [BOO94]: abstracción, encapsulación, modularidad, jerarquía, control de tipos, concurrencia, persistencia y genericidad. El segundo criterio se traduce en cinco características principales: persistencia, concurrencia, recuperación ante fallos del sistema, gestión del almacenamiento secundario y facilidad de consultas.

Fig. No.6

Características de SGBDOO

Como se puede apreciar en el esquema (Fig.No.6) la persistencia, al igual que la concurrencia son características del SGBDOO heredadas tanto del SGBD como del modelo de objetos. La persistencia en el caso del SGBD hace referencia a la conservación de los datos después de la finalización del proceso que los creó. En el caso del modelo de objetos, se refiere no sólo a la conservación del estado de un objeto, si no también a la conservación de la clase, que debe trascender a cualquier programa individual, de forma que todos los programas interpreten de la misma manera el estado almacenado. La concurrencia heredada del SGBD se refiere a la capacidad del sistema para gestionar a múltiples usuarios interactuando concurrentemente sobre el mismo, mientras que la concurrencia heredada del modelo de objetos hace referencia a la capacidad de distinguir a un objeto activo de otro que no lo está.

- ***Persistencia:*** Es la capacidad que tiene el programador para que sus datos se conserven al finalizar la ejecución de un proceso, de forma que se puedan reutilizar en otros procesos.

- ***Concurrencia:*** se relaciona con la existencia de muchos usuarios interactuando concurrentemente en el sistema. Este debe controlar la interacción entre las transacciones concurrentes para evitar que se destruya la consistencia de la base de datos.

- ***Recuperación:*** Proporcionar como mínimo el mismo nivel de recuperación que los sistemas de bases de datos actuales. De forma que, tanto en caso de fallo de hardware como de fallo de software, el sistema pueda retroceder hasta un estado coherente de los datos.

- ***Gestión del almacenamiento secundario:*** es soportada por un conjunto de mecanismos que no son visibles al usuario, tales como gestión de índices, agrupación de datos, selección del camino de acceso, optimización de consultas, etc. Estos mecanismos evitan que los programadores tengan que escribir programas para mantener índices, asignar el almacenamiento en disco, o trasladar los datos entre el disco y la memoria principal, creándose de esta forma una independencia entre los niveles lógicos y físicos del sistema.

- ***Facilidad de Consultas:*** permitir al usuario hacer cuestiones sencillas a la base de datos. Este tipo de consultas tienen como misión proporcionar la información solicitada por el usuario de una forma correcta y rápida.

3.2 Primer intento de Estandarización: ODMG-93.

La mayor limitación de las bases de datos orientadas a objetos es la carencia de un estándar. ODMG-93 (Object-Oriented Database Management Group) es un punto de partida muy importante para ello .Adopta una arquitectura que consta de un sistema de gestión que soporta un lenguaje de bases de datos orientado a objetos, con una sintaxis similar a un lenguaje de programación también orientado a objetos como puede ser C++ o Smalltalk. El lenguaje de bases de datos es especificado mediante un lenguaje de definición

de datos (ODL), un lenguaje de manipulación de datos (OML), y un lenguaje de consulta (OQL), siendo todos ellos portables a otros sistemas con el fin de conseguir la portabilidad de la aplicación completa.

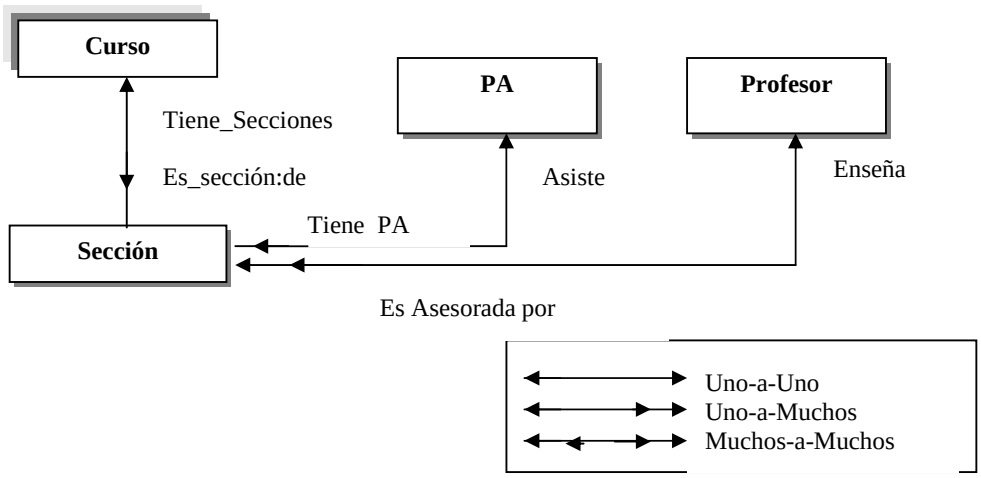
En definitiva, ODMG-93 intenta definir un SGBDOO que integre las capacidades de las bases de datos con las capacidades de los lenguajes de programación, de forma que los objetos de la base de datos aparezcan como objetos del lenguaje de programación, intentando de esta manera eliminar la falta de correspondencia existente entre los sistemas de tipos de ambos lenguajes .El SGBDOO extiende el lenguaje con persistencia, concurrencia, recuperación de datos, consultas asociativas, etc.

3.2.1 Lenguaje ODL

El lenguaje de definición de datos (ODL) en un SGBDOO es empleado facilitar la portabilidad de los esquemas de las bases de datos. Este ODL no es un lenguaje de programación completo, define las propiedades y los prototipos de las operaciones de los tipos, pero no los métodos que implementan esas operaciones.

(Fig.No.7*)*

Representación gráfica de un esquema de una base de datos.



El ODL intenta definir tipos que puedan implementarse en diversos lenguajes de programación; no está por tanto ligado a la sintaxis concreta de un lenguaje de programación particular. De esta forma un esquema especificado en ODL puede ser soportado por cualquier SGBDOO que sea compatible con ODMG-93.

La sintaxis de ODL es una extensión de la del IDL (Interface Definition Language) desarrollado por OMG como parte de CORBA (Common Object Request Broker Architecture).

3.2.2 Lenguaje OML

El lenguaje de manipulación es empleado para la elaboración de programas que permitan crear, modificar y borrar datos que constituyen la base de datos. ODMG-93 sugiere que este lenguaje sea la extensión de un lenguaje de programación, de forma que se pueden realizar entre otras las siguientes operaciones sobre la base de datos: Creación, Borrado, Modificación e Identificación de un objeto

3.2.3 Lenguaje OQL

El lenguaje de consulta propuesto por ODMG-93, presenta las siguientes características:

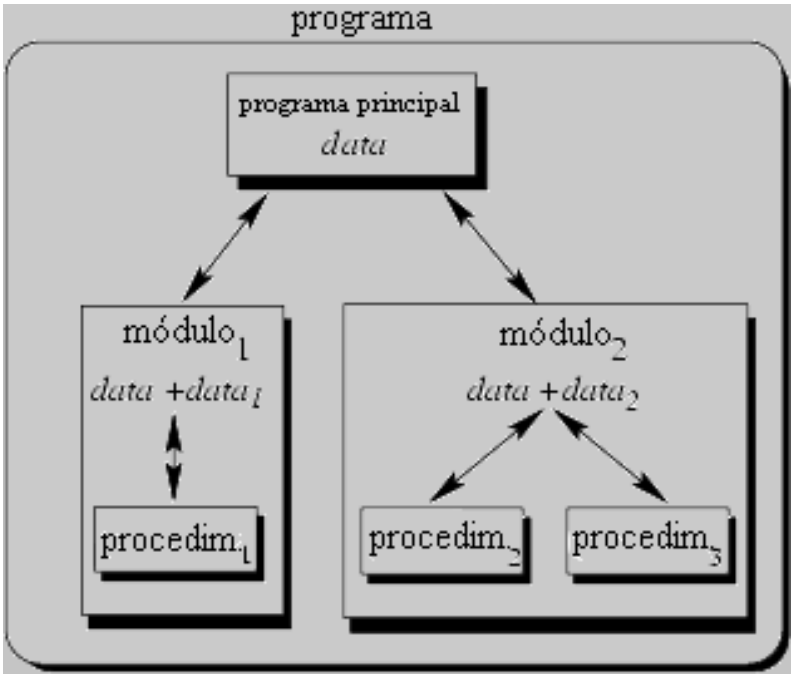
- ✓ No es computacionalmente completo. Sin embargo, las consultas pueden invocar métodos, e inversamente los métodos escritos en cualquier lenguaje de programación pueden incluir consultas.
- ✓ Tiene una sintaxis abstracta.
- ✓ Su semántica formal puede definirse fácilmente.
- ✓ Proporciona un acceso declarativo a los objetos.
- ✓ Se basa en el modelo de objetos de ODMG-93.
- ✓ Tiene una sintaxis concreta al estilo SQL, pero puede cambiarse con facilidad.
- ✓ Puede optimizarse fácilmente.
- ✓ No proporciona operadores explícitos para la modificación, se basa en las operaciones definidas sobre los objetos para ese fin.
- ✓ Proporciona primitivas de alto nivel para tratar con conjuntos de objetos, pero no restringe su utilización con otros constructores de colecciones.

Existen dos posibilidades para asociar un sublenguaje de consulta a un lenguaje de programación: fuerte y débilmente.

- ✓ El primer caso consiste en una extensión de la gramática del lenguaje asociado.
- ✓ En el segundo caso, las funciones query tienen unos argumentos String que contienen las preguntas.

3.3 Programación Modular

En la programación modular, los procedimientos con una funcionalidad común son agrupados en ***módulos*** separados. Un programa por consiguiente, ya no consiste solamente de una sección. Ahora está dividido en varias secciones más pequeñas que interactúan a través de llamadas a procedimientos y que integran el programa en su totalidad.



(Figura No.8): Programación Modular.

El programa principal coordina las llamadas a procedimientos en módulos separados y pasa los datos apropiados en forma de parámetros. **Cada módulo puede contener sus propios datos. Esto permite que cada módulo maneje un estado interno que es modificado por las llamadas a procedimientos de ese módulo. Sin embargo, solamente hay un estado por módulo y cada módulo existe cuando más una vez en todo el programa.**

3.4 Objetivos

El desarrollo del SGBDOO tiene como finalidad principal la verificación de las hipótesis que se plantean a continuación:

- Desarrollo más sencillo del propio SGBDOO. Esto parece lógico ya que algunas de las funciones que debería implementar el SGBD (ej. Persistencia) ya están disponibles dentro del propio sistema operativo. Al tratarse de un sistema integral orientado a objetos, se obtienen las ventajas de la orientación a objetos: así por ejemplo, es posible reutilizar el código de persistencia ya existente, o extenderlo añadiendo únicamente la funcionalidad adicional necesaria para el SGBDOO. Esta funcionalidad puede proporcionarse mediante un motor de base de datos adecuado que complemente las características proporcionadas por el sistema operativo.

- Mayor integración en el sistema. Es decir, los objetos de la base de datos son simplemente unos objetos más dentro de los objetos del sistema operativo que proporcionan servicios. Es más, puede pensarse en el SGBDOO como el elemento que cumpla el papel de los sistemas de ficheros en los sistemas operativos tradicionales. El SGBDOO no sería utilizado como un sistema independiente del sistema operativo, si no que el usuario podría utilizarlo como sistema de gestión de los objetos del sistema operativo, haciendo consulta sobre los mismos.

- Mayor rendimiento. Dado que el propio sistema integral ya es orientado a objetos, no existe la necesidad de desarrollar capas superpuestas a un sistema operativo tradicional para salvar el espacio existente entre el paradigma del sistema operativo y el de la base de datos.

- Mayor productividad. La programación de aplicaciones de bases de datos es más productiva ya que no es necesario que el programador cambie constantemente de filosofías: una para trabajar con la base de datos y otra para manejar el sistema operativo. Ambos elementos utilizan ahora el mismo paradigma de orientación a objetos.

3.5 Otros aspectos a destacar

Es un SGBDOO que se construye sobre una máquina abstracta persistente totalmente orientada a objetos. El hecho de basarse en una máquina abstracta permite que cualquier aplicación sobre este SGBDOO sea portable a cualquier plataforma, sin necesidad obviamente de sobrescribir el código. El SGBDOO ha de ser configurable y modular, de forma que permita la elección del mecanismo de indexación, de la estrategia de estimación de costos, etc. Aunque inicialmente el primer prototipo se implemente con una técnica de indexación o con una estrategia de estimación de costos concreta, eso no impide que se puedan ir incorporando a dicho sistema nuevas técnicas ya existentes o algunas innovadoras, de forma que se puedan realizar comparativas para ver cual de ellas se comporta mejor y en que situaciones.

3.6 Trabajos relacionados

En la actualidad existen sistemas de gestión de bases de datos orientadas a objetos así como simples gestores de almacenamiento persistente, tanto a nivel comercial como a nivel de investigación en proyectos realizados por diferentes universidades. Lo que no se han encontrado son SGBDOO integrados con máquinas abstractas persistentes y sistemas operativos orientados a objetos. A continuación se exponen brevemente las características de algunos de los sistemas encontrados.

Shore [Shore 97] es un sistema persistente de objetos, sucesor de Exodus, que está siendo desarrollado en la Universidad de Wisconsin en USA. Representa una mezcla entre las tecnologías de bases de datos orientadas a objetos y los sistemas de ficheros En Shore todo dato persistente es un objeto, y todo objeto tiene una identidad especificada por un identificador de objeto único. Todos los datos persistentes son descritos mediante el lenguaje SDL (Shore Data Language) parecido al ODL propuesto por ODMG. Shore permite el control de la concurrencia, la recuperación de caídas, el manejo de transacciones y las consultas optimizadas sobre ciertos tipos. Además, implementa un espacio de nombres, un modelo de control de acceso similar a los de Unix, y los servicios tradicionales de toda base de datos como el acceso asociativo a los datos, la indexación y el agrupamiento.

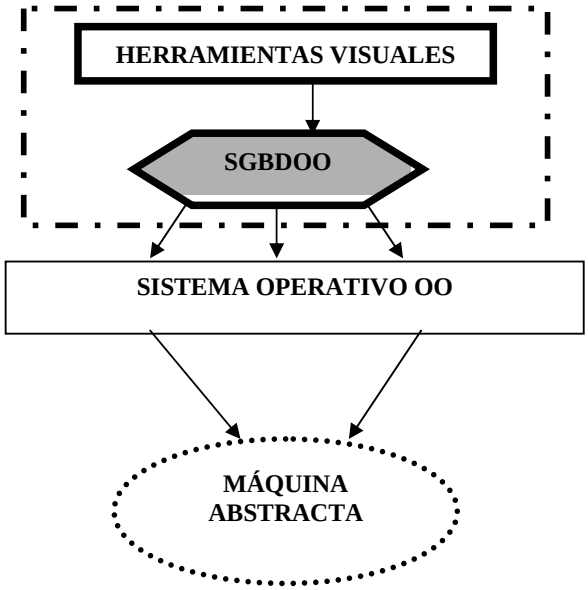
Texas [Singhal 92] es un sistema de almacenamiento persistente orientado a objetos desarrollado en la Universidad de Texas en Austin. Ha sido implementado como una librería de C++, de tal manera que cualquier aplicación enlazada con dicha librería podrá crear y manipular tanto objetos persistentes como transitorios. Al igual que la base de datos orientada a objetos comercial ObjectStore emplea un mecanismo de traducción de punteros (pointer swizzling) cuando se produce el fallo de página. Los objetos almacenan las referencias a otros objetos como identificadores de objetos persistentes sobre disco, pero cuando se trae una página a memoria todas las referencias sobre esa página son traducidas a punteros en memoria virtual. Todo esto es realizado de una forma transparente al programa cliente, y una vez que las referencias a los objetos han sido traducidas, los programas clientes pueden acceder rápidamente a los objetos en memoria.

Thor [Thor 97] es un sistema de base de datos distribuido orientado a objetos desarrollado en el Instituto Tecnológico de Massachusetts. Proporciona un sistema de almacenamiento persistente de objetos, permitiendo el acceso a los mismos mediante transacciones. En Thor un objeto se convierte en persistente cuando es alcanzado por el objeto raíz. La arquitectura de Thor se organiza en torno a un repositorio de objetos residente en el servidor y a unos pares front-end/cliente almacenados en los clientes. Los procesos front-end son los que hacen las veces de intermediarios entre el programa cliente y el repositorio de objetos. Además, en el cliente para cada lenguaje soportado se proporciona una librería de código que implementa la interfaz entre el cliente y el front-end. Los objetos dentro de Thor se especifican e implementan usando un nuevo lenguaje procedural de programación llamado Theta. Este lenguaje es orientado a objetos, extensible y fuertemente tipado, permitiendo tanto la especificación de la interfaz de un tipo como la implementación del mismo

3.7 Situación del SGBDOO

La ideal inicial de este subproyecto dentro del Proyecto Oviedo3 es la construcción de un SGBDOO (Fig.No.9) que esté totalmente integrado con las características de la máquina abstracta y del sistema operativo de la misma, aprovechando al máximo las posibilidades que ofrecen.

(Fig.No.9)
Idea Inicial



3.8 Prototipo I

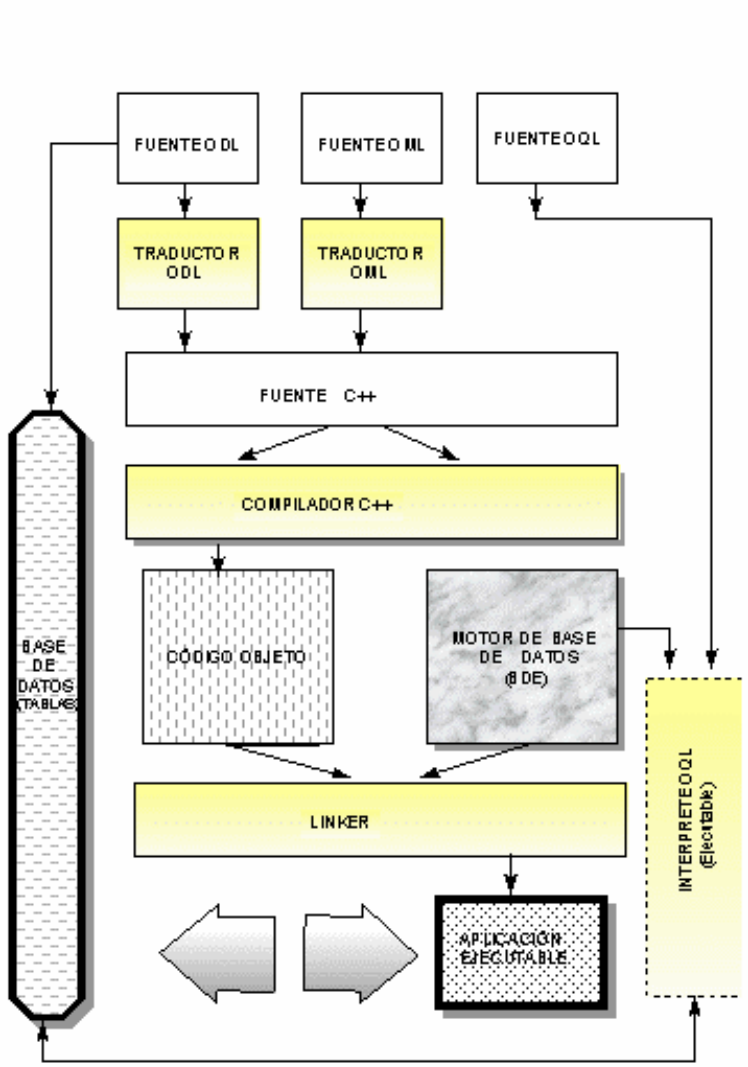
El primer paso en el desarrollo de este prototipo (FigNo.10), y del sistema en general, consistirá en el diseño del lenguaje de definición de datos (ODL), el lenguaje de manipulación de datos (OML) y el lenguaje de consulta (OQL). Este diseño intentará seguir, en la medida de lo posible, las pautas impuestas por el estándar ODMG-93.

El paso siguiente se basa en conocer el lenguaje de programación que se va a emplear como base para los lenguajes de bases de datos anteriormente mencionados. Se intentará que estos lenguajes sean lo menos dependientes posible del lenguaje de programación. En este primer prototipo (FigNo.10) el lenguaje de programación seleccionado será un C++ comercial.

A continuación, el siguiente módulo consistirá en la traducción de estos lenguajes de bases de datos (ODL y OML), al lenguaje de programación seleccionado, en este caso C++, de manera que el programa objeto así obtenido se pueda compilar con un compilador comercial, para

posteriormente disponer de todas las capacidades de los sistemas de gestión de bases de datos mediante el "linkado" (enlazado) con un motor de base de datos. En este caso, el motor de base de datos empleado es el de Borland (BDE). En el caso del lenguaje de consulta (OQL), en vez de un traductor se puede emplear un intérprete avanzado de OQL (Fig.No.10) que permita obtener las respuestas a las consultas de forma interactiva. El "linkado"(enlazado) con el motor de base de datos se realizaría durante la construcción del intérprete.

Prototipo I



(Fig.No.10)
Prototipo I del SGBDOO

En esta primera fase se realiza una traducción de los objetos a tablas, ya que el motor de base de datos sigue trabajando sobre un sistema operativo tradicional orientado a ficheros. Es decir, los datos al final estarán almacenados en tablas del estilo Dbase, Paradox, Quattro Pro, etc. Como se puede apreciar, en esta fase se emplearán una serie de herramientas comerciales

(compilador de C++, BDE,..) que serán sustituidas en etapas posteriores por herramientas que tendrá el propio sistema Oviedo3, pero que todavía no están implementadas. Se acude a estas herramientas comerciales en este prototipo, porque el objetivo del mismo es avanzar en el diseño y prueba de los lenguajes de bases de datos mientras se construyen el sistema operativo y la máquina abstracta de Oviedo3.

3.9 Prototipo II

Los contenidos de este prototipo (FigNo.11) son mucho más ambiciosos, ya que aprovecharán plenamente las características ofrecidas por la máquina abstracta y por el sistema operativo de Oviedo3. A pesar de esto, los módulos a desarrollar seguirán siendo los mismos, pero con algunas notables diferencias:

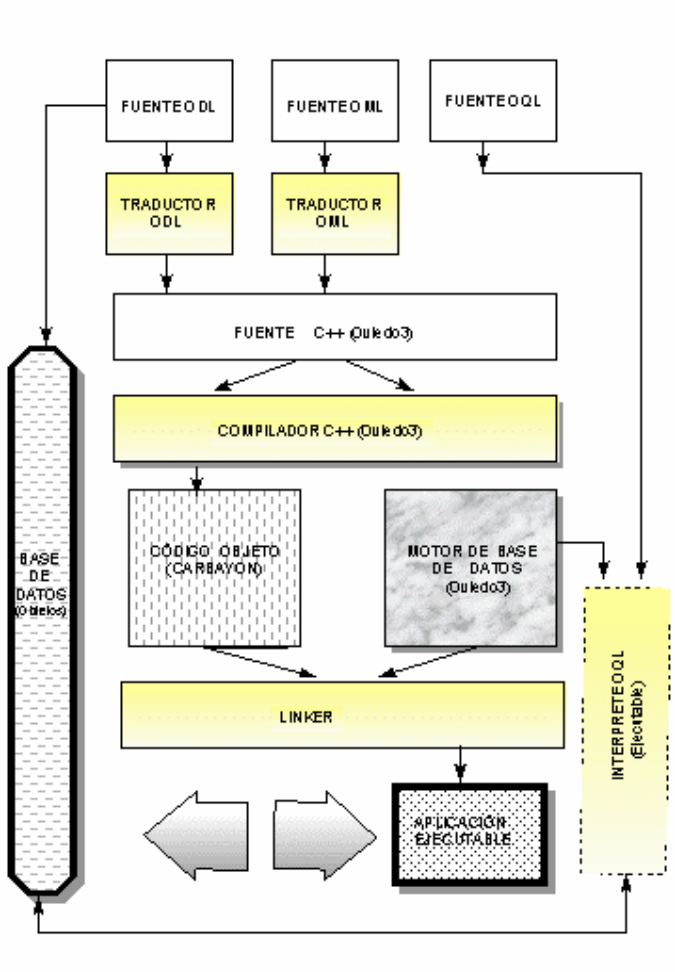
- En vez de utilizar como lenguaje de programación C++, se empleará otro lenguaje muy similar, pero dónde el código objeto que se genera al compilar un programa con este lenguaje, es el lenguaje (CARBAYON) de la máquina abstracta.

- Los traductores y el intérprete para los lenguajes de definición, de manipulación y de consulta, tendrán que generar un código objeto en este nuevo lenguaje, en vez de en el C++ comercial empleado en el prototipo anterior.

- El motor de base de datos, ya no será el BDE si no que será propio y desarrollado en función de las características que nos proporcione el sistema operativo. Aquí habrá que prestar especial atención a temas como la persistencia, seguridad, distribución, concurrencia, generación y mantenimiento de índices,…

- La información, es decir, los objetos se almacenarán ya como tales al proporcionar el sistema operativo la persistencia de los mismos, a diferencia del prototipo anterior en el que se almacenaban en tablas.

(Fig.No.11)
Prototipo II



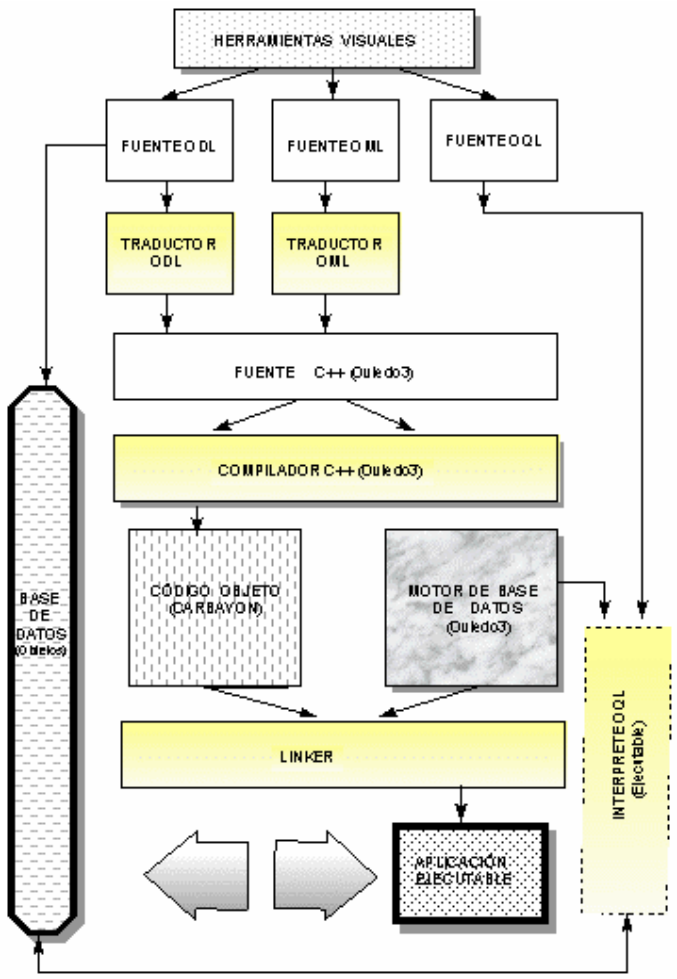
3.10 Prototipo III

Este prototipo (FigNo.12) constituye una ampliación del anterior, cara a facilitar al usuario su trabajo con este sistema de gestión de bases de datos. Con esta finalidad se ha incorporado un nuevo componente (Herramientas Visuales) a los ya existentes en el prototipo anterior. Este nuevo elemento puede descomponerse en dos módulos principales.

El primer módulo consiste en un conjunto de herramientas visuales para el desarrollo, que permitirán tanto definir el esquema de la base de datos, como manipular o consultar los datos en ella almacenados de forma totalmente visual. El objetivo de estas herramientas en este sistema es que la cantidad de código que tenga que escribir el usuario sea la mínima posible.

El segundo módulo consiste en un conjunto de herramientas también visuales, que faciliten las tareas de administración de la base de datos, del servidor y de los usuarios. Este módulo va a exigir la existencia de un lenguaje de control que permita la protección de la base de datos, la restricción de los accesos tanto a objetos como a métodos, etc.

(Fig. No. 12)
Prototipo III



3.11 Qué viene después de OO

La orientación del objeto es uno de los queridos de la tecnología hoy. La IBM está integrando agresivamente tecnología de OO en cada aspecto de su estrategia. Un ejemplo de esto es iniciativa de IBM's San Francisco. Como sabemos, tecnologías du jour es los fenómenos efímeros. Tan qué viene después de OO. Esta sesión presentará la vista de la

tecnología futura de un departamento principal del R&D de las herramientas de desarrollo de la aplicación de OO. Discute el potencial y las limitaciones prácticos de la tecnología actual de OO. Presenta una vista del potencial futuro de OO y de las tecnologías posibles que crezcan fuera de OO de hoy.

3.12 Conclusiones

En Conclusión sabemos que las BDOO representan el siguiente paso en la evolución de las bases de datos, para soportar el Análisis, Diseño y Programación OO. Las BDOO permiten el desarrollo y mantenimiento de aplicaciones complejas con un costo Significativamente menor. Permiten que el mismo modelo conceptual se aplique al Análisis, diseño, programación, definición y acceso a la base de datos. Esto reduce el problema del operador de traducción entre los diferentes modelos a través de todo el ciclo de vida. El modelo conceptual debe ser la base de las herramientas CASE OO totalmente integradas, las cuales ayudan a generar la estructura de datos y los métodos.

Las BDOO ofrecen un mucho mejor rendimiento de la máquina que las bases de datos por relación, para aplicaciones o clases con estructuras complejas de datos. Sin embargo, Las BDOO coexistirán con las bases de datos por relación durante los próximos años, puesto que a menudo se utilizará un modelo por relación como una forma de estructura de datos dentro de una BDOO.
Podemos decir que en conclusión con el caso de Oracle ha aumentado la demanda de una representación de objetos complejos en las actuales aplicaciones convencionales.