

Un SGBDOO soportado por el Sistema Operativo Distribuido del Sistema Integral Oviedo3

A. B. Martínez Prieto, F. Álvarez García, F. Ortín Soler y J.M Cueva Lovelle

Universidad de Oviedo
Departamento de Informática.
C/Calvo Sotelo s/n, 33007.
email: (belen,falvarez,ortin,cueva)@pinon.ccu.uniovi.es

Resumen

Actualmente la mayoría de los sistemas de gestión de bases de datos orientadas a objetos distribuidos son arquitecturas cliente-servidor construidas sobre sistemas operativos tradicionales y que incorporan la posibilidad de replicar la información. Una alternativa a esto puede ser el aprovechamiento del sistema de distribución proporcionado por el sistema operativo de un sistema integral orientado a objetos para la construcción del propio SGBDOO distribuido. Todos los elementos antes mencionados comparten el mismo modelo de objetos. En este artículo se enumeran algunas de las ventajas de esta arquitectura.

Palabras Clave: Orientación a objetos, SGBDOO, distribución, reflectividad, máquina abstracta, sistema integral.

1. Introducción

El término de base de datos distribuida ha sido empleado con muchos significados diferentes en la literatura, sin embargo, se pueden resumir en dos los requerimientos exigibles clásicamente a una base de datos distribuida orientada a objetos: la posibilidad de acceder a datos localizados en otras máquinas y la ejecución remota de las operaciones, así como la capacidad de ejecutar consultas y programas que acceden a más de una base de datos.

Una posibilidad tradicional para conseguir distribución viene de la mano de los sistemas cliente servidor, de hecho, la mayoría de los SGBDOO son arquitecturas cliente-servidor en la que un servidor proporciona servicios a un número de clientes. En estas arquitecturas la comunicación entre los clientes y el servidor es realizada mediante llamadas a procedimientos remotos (RPC, *remote procedure call*) o paso de mensajes, y es básica la determinación de las funciones que van a ser realizadas por los clientes y el servidor, así como la unidad de comunicación entre ellos (páginas u objetos).

La mayoría de estos sistemas están contruidos sobre sistemas operativos tradicionales con las abstracciones limitadas que estos nos proporcionan, así como los problemas de interoperabilidad entre modelos de objetos que se generan. Sin embargo, otra posibilidad puede ser aprovecharse del sistema de distribución proporcionado por el sistema operativo de un sistema integral orientado a objetos para la construcción del propio SGBDOO distribuido. Todos los elementos antes mencionados comparten el mismo modelo de objetos lo que facilita la interoperabilidad y elimina la desadaptación de impedancias entre ellos.

El resto del artículo se organiza como sigue. En el apartado siguiente se define un SGBDOO distribuido, así como algunos de los problemas que se derivan de su construcción sobre sistemas operativos tradicionales. En el apartado 3 se describe el sistema integral antes mencionado. A continuación, se describen brevemente las características del sistema operativo desde el punto de vista de la distribución. Las características del SGBDOO así como el estado actual del mismo son descritos en el apartado 5. Finalmente, en el apartado 6 se enumeran las ventajas principales que obtiene el SGBDOO al contar con este sistema operativo distribuido subyacente.

2. Sistemas de Gestión de Bases de Datos Distribuidas

Una base de datos distribuida es definida en [12] como una *colección de múltiples bases de datos lógicamente interrelacionadas distribuidas sobre una red de ordenadores*. Un sistema de gestión de base de datos (SGDB) distribuidas *es el software que permite la gestión de la base de datos distribuida y hace la distribución transparente al usuario*. Y finalmente, un SGBDOO distribuido *permite la gestión transparente de los objetos que están distribuidos en diferentes máquinas*. Se puede deducir que la palabra clave aquí es la transparencia:

- *Transparencia de localización*, que oculta la existencia de la red y la distribución de los datos
- *Transparencia de replicación*, que enmascara la existencia de copias físicas de datos
- *Transparencia de fragmentación*, que evita que el usuario tenga que tratar con las particiones de las bases de datos de objetos.

2.1 Arquitecturas

Tradicionalmente existen diferentes posibilidades para la implementación de SGDB distribuidos sobre sistemas operativos convencionales:

- Arquitecturas cliente-servidor (*two tier*), donde múltiples clientes acceden a un servidor de base de datos. Los clientes gestionan el procesamiento de los datos y la presentación.
- Arquitecturas de tres capas (*Three Tier*) [8]. Evolución más actualizada de la anterior, en la que se distinguen tres capas: *presentación*, *lógica* y el *acceso a los datos*. La principal ventaja con relación a la anterior es que facilita la escalabilidad de las aplicaciones.

Pero realmente un SGDB distribuido no distingue entre máquinas clientes y servidoras. Idealmente, cada máquina puede realizar la funcionalidad de un cliente y un servidor. La base de datos puede distribuirse físicamente entre las diferentes máquinas fragmentando y replicando datos. El resto del artículo lo dedicaremos ya a una arquitectura de este tipo.

2.2 Problemática de construir un SGBDOO distribuido sobre un SO tradicional

En los apartados anteriores se ha estado definiendo un SGDB distribuido así como las arquitecturas empleadas tradicionalmente en la distribución de bases de datos. La mayoría de estos SGDB, orientados a objetos o no, están contruidos sobre sistemas operativos tradicionales.

Aunque esto es una realidad hoy en día, ya a finales de los 70 Gray y Stonebraker reconocieron los inconvenientes que tenía la construcción de un SGDB, en aquel momento relacional, sobre un SO tradicional [13]. A su modo de ver el principal problema se deriva de que el sistema operativo proporciona un conjunto de abstracciones con una implementación simple para cada una de ellas, de forma que si un SGDB requiere una implementación diferente para una abstracción del SO las opciones son bastante limitadas: bien se modifica el código fuente del SO si es que se posee, o bien el programador del SGDB tiene que volver a implementar la abstracción. En sistemas distribuidos esta situación es más complicada por las consideraciones para la comunicación en red (problemas para adaptar el sistema operativo a los distintos entornos de red).

En el caso de los SGBDOO estos inconvenientes se acentúan [9]. Además del empleo de abstracciones no adecuadas por parte del sistema operativo (ej. no permiten comunicar objetos en dos espacios de direcciones diferentes), se produce una desadaptación de interfaces (la mayoría están orientados al paradigma procedimental) lo que provoca una desadaptación de impedancias o salto semántico, a medida solucionada incorporando una capa de software adicional que enmascara dicho problema.

Además, en el caso de OO se produce a menudo un problema de interoperabilidad entre modelos de objetos ya que aunque diferentes elementos del sistema empleen el paradigma de OO es probable que existan

problemas de desadaptación entre ellos. De nuevo se recurre a la introducción de capas de software de adaptación para solucionar el problema (Ej. CORBA).

3. Sistema Integral Orientado a Objetos

Como se ha visto para solucionar los problemas anteriores los sistemas convencionales recurren a la introducción de capas de software adicionales, lo que ocasiona una disminución del rendimiento global del sistema, una pérdida de portabilidad y flexibilidad y un aumento de la complejidad.

Otra posibilidad sería contar con un sistema operativo distribuido orientado a objetos sobre el que se construiría el SGBDOO y en el que la única abstracción fuese el objeto. Para ello es necesario que el soporte para los objetos se dé desde las capas más bajas del mismo y de una forma homogénea para todos los elementos del sistema. Es entonces cuando surge la necesidad de un sistema integral orientado a objetos. Un sistema integral orientado a objetos *ofrece al usuario un entorno de computación que crea un mundo de objetos: un único espacio de objetos, virtualmente infinito, en el que un conjunto de objetos homogéneos coopera intercambiando mensajes independientemente de su localización, y donde los objetos residen indefinidamente hasta que ya no son necesarios.*

3.1 Sistema Integral Oviedo3

Oviedo3 [7] es un sistema integral orientado a objetos en el que todos los componentes: interfaces de usuario, aplicaciones, lenguajes, compiladores, bases de datos y el propio sistema operativo comparten el mismo paradigma de orientación a objetos. En el corazón del sistema integral se encuentra una máquina abstracta [3].

3.2 Máquina Abstracta

La máquina abstracta, llamada Carbayonia, soporta un modelo de objetos [4] básico con las características *de identidad de objetos y abstracción, encapsulación, herencia, genericidad, relaciones de agregación entre objetos, y polimorfismo o paso de mensajes.* Este modelo de objetos incorpora los conceptos del paradigma de orientación a objetos más aceptados y permite, por tanto, representar los modelos de los lenguajes de programación más utilizados.

La arquitectura de la máquina está estructurada en cuatro áreas (Figura 1): área de clases, área de referencias, área de instancias y área de referencias del sistema. Cada área puede ser considerada como un objeto encargado de la gestión de sus datos.

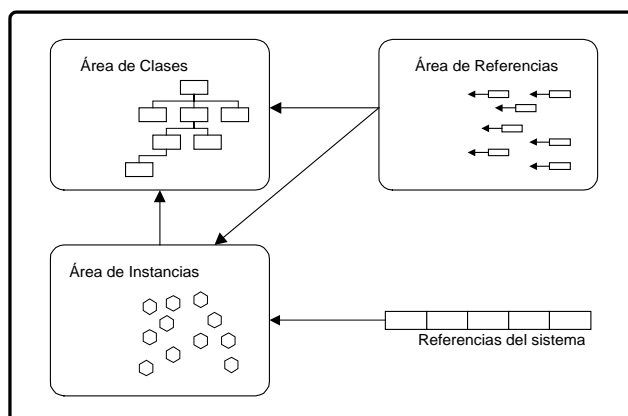


Figura 1. Arquitectura de referencia de la máquina abstracta

El lenguaje de la máquina (Carbayón) es un lenguaje OO de bajo nivel, que permite declaración de clases, definición de métodos y manejo de excepciones.

3.2.1 Reflectividad de la máquina abstracta

La reflectividad fue definida por [14] como *la propiedad que tiene un sistema de razonar acerca de sí mismo y actuar sobre sí mismo cambiando su propio comportamiento*. Es decir, la propiedad por la que el sistema base puede acceder a su meta-sistema, conocer información acerca de sí mismo, e incluso modificar el meta-sistema (cambiando así su propio comportamiento).

La arquitectura reflectiva de la máquina abstracta [15] divide el espacio de objetos en dos niveles; el nivel-base y el meta-nivel. Los objetos en el nivel base implementan aplicaciones de usuario. El meta-nivel está descompuesto en un conjunto de objetos llamados meta-objetos, que extienden, modifican o complementan el comportamiento por defecto de la máquina. El sistema operativo está básicamente implementado en el meta-nivel.

3.3 Sistema Operativo

El sistema operativo, llamado SO4, ofrece la abstracción de un espacio de objetos simple en el que los objetos existen indefinidamente, y dónde los objetos colocados en diferentes máquinas cooperan transparentemente usando mensajes. Además, el sistema operativo proporciona de una forma transparente un conjunto de características como son: seguridad, persistencia, distribución y concurrencia, con las ventajas que eso supone para la construcción del SGBDOO [1].

4. Características del sistema operativo desde el punto de vista de la distribución

Para proporcionar distribución al sistema integral se ha optado por aprovechar la reflectividad de la máquina de forma que el paradigma de objetos no se rompe y los servicios pueden adaptarse fácilmente por modificación o reemplazamiento de los objetos.

El sistema operativo distribuido aquí presentado está construido como un conjunto de objetos de forma que la solicitud tradicional de servicios al sistema operativo se realiza invocando métodos de objetos que proporcionan esas funciones. La invocación de esos métodos se realiza de una forma transparente e independiente de la ubicación de los objetos (local o remota). El sistema operativo solucionará todos los problemas que surjan debido a las diferentes situaciones en que se puedan encontrar los objetos, sin necesidad de que el programador haya tenido que escribir código para su tratamiento. Además, este sistema operativo proporciona mecanismos para la migración de objetos entre los distintos nodos del sistema distribuido [2].

4.1 Características de la distribución

Las interacciones entre objetos están basadas en las invocaciones de métodos. Cada objeto posee un identificador único dentro del sistema e independiente de la ubicación del mismo.

Se emplea un modelo de objetos activo, de forma que cada objeto encapsula el estado de su computación. Esto facilita la migración de objetos ya que cuando se mueve un objeto no sólo se mueve su estado si no también su computación.

4.2 Migración de objetos

En este sistema la unidad de migración es el objeto. El movimiento del objeto se realiza de forma transparente, lo que se traduce en que el objeto continuará su computación en el nodo de destino como si el movimiento no hubiese tenido lugar, y las invocaciones hechas cuando el objeto se está moviendo se redireccionarán a la nueva localización.

La migración de objetos se soluciona en el meta-nivel de una forma transparente. Existen un conjunto de meta-objetos que son responsables de interrumpir cualquier actividad interna en el objeto, obteniendo su estado encapsulado en una cadena, enviando la cadena en un mensaje al destino elegido y reactivando el objeto allí. Permite emplear diferentes políticas para decidir cuándo, dónde y cómo moverlos.

4.3 Invocación de métodos

El mecanismo de invocación de métodos proporcionado por la máquina abstracta es reescrito al meta-nivel para hacer posible la invocación de métodos remota. Un conjunto de meta objetos implementan un nuevo mecanismo de invocación de métodos que reduce las dificultades introducidas por la posible dispersión de objetos, clases y referencias implicadas.

5. Sistema de Gestión de Bases de Datos del Sistema Integral

Sobre este sistema operativo se está construyendo un SGBDOO que comparte el mismo modelo de objetos y que se caracteriza por aplicar los principios de orientación a objetos al propio SGBD. El sistema se diseña como una jerarquía de clases, en la que cada una tiene una funcionalidad específica. Esto dota al sistema de una estructura modular de sencilla identificación, y que permite fácilmente la configuración aprovechándose para ello de mecanismos como la herencia.

5.1 Características del SGBDOO

El sistema aquí descrito, como todo SGBDOO, debe proporcionar las características de un modelo de objetos y todas las funcionalidades de un SGBD. Entre sus características destacan las siguientes:

- *Modelo de objetos de la máquina abstracta.* El modelo de objetos del SGBDOO es el modelo de objetos de la máquina abstracta adoptado también por el resto de elementos del sistema.
- *Algunas capacidades del SGBD proporcionadas por el sistema operativo.* La persistencia, distribución, concurrencia y seguridad son proporcionadas por el SO.
- *SGBDOO puro.* Los objetos modelados con las herramientas de análisis y diseño orientadas a objetos son traducidos en objetos en el lenguaje de programación y finalmente almacenados directamente en la base de datos empleando la persistencia ofrecida por el sistema.
- *Extensibilidad y Configurabilidad.* Son dos características fundamentales en este sistema. Este sistema ha de permitir fácilmente cambiar los mecanismos de indexación o las estrategias de estimación de costos que emplea el motor. Para ello se acude inicialmente a las facilidades que para ello da el paradigma de OO y en última instancia a la característica reflectiva de la propia máquina abstracta [11].
- *Plataforma de benchmarking.* Basándose en las dos características anteriores este SGBDOO es especialmente apropiado como plataforma de benchmarking entre por ejemplo, diferentes técnicas de indexación.

5.2 Arquitectura del SGBDOO

El SGBDOO se ha estructurado para su construcción en tres apartados (Figura 2):

- **Motor.** En este módulo se destacan dos funciones principales: procesamiento de consultas y gestión del almacenamiento. El motor es realmente una jerarquía de clases en las que se aprovechan los servicios del sistema operativo como la persistencia o la distribución, se implementan algunos nuevos como la indexación, el procesamiento de consultas, etc. y todos ellos se ponen en forma de métodos a disposición de las restantes capas del SGBDOO o bien de un usuario experto si lo desea.

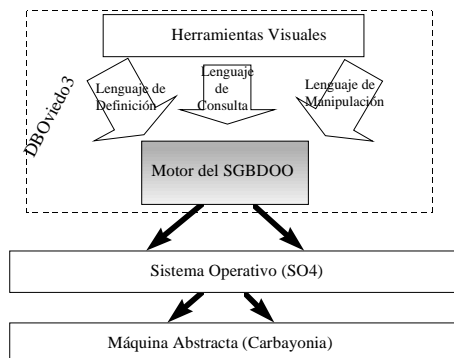


Figura 2. Arquitectura del SGBDOO de Oviedo

- Lenguajes.** En este módulo es dónde se definen y se especifican formalmente los lenguajes de base de datos que van a ser utilizados en el sistema. Los lenguajes empleados siguen el estándar ODMG [5,6], en el que se incluye la especificación para un lenguaje de definición de datos (ODL, Figura 3) y para un lenguaje de consulta (OQL). Para el lenguaje de manipulación (OML) se descarta la idea de una sintaxis única, proponiendo modificaciones para tratar las bases de datos a lenguajes de programación OO ya existentes como C++, Smalltalk o Java.

<pre> Class Professor { attribute string FirstName; attribute string LastName; attribute short Age; short GetAge(); relationship set<Section> Teaches inverse Section::Is-Taught_By; }; </pre>	<pre> Class Section { attribute short Number; short GetNumber(); relationship Professor Is-Taught_By inverse Professor::Teaches; }; </pre>
--	--

Figure 3. Ejemplo de especificación ODL

- Herramientas Visuales.** Este módulo tiene como misión la construcción de un entorno de desarrollo visual para interactuar con la base de datos, y cuyo objetivo principal es facilitar a todo tipo de usuario el trabajo con la misma. Se incluyen *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, y *herramientas visuales para la administración de la base de datos*, que faciliten las tareas de administración de la base de datos y de los usuarios.

5.3 Situación Actual

Actualmente existe un primer prototipo de este SGBDOO que incluye las funcionalidades expuestas en los dos primeros apartados, y en estos momentos se están diseñando las herramientas visuales para el desarrollo.

5.3.1 Motor

En este primer prototipo el motor proporciona las siguientes funcionalidades:

- Gestión del almacenamiento, aprovechando la persistencia del sistema operativo
- Procesamiento de consultas elemental (sin optimización)
- Mecanismo de indexación configurable y extensible

Características del mecanismo de indexación

El procesamiento de consultas es básico en cualquier motor de base de datos, y el rendimiento de este procesamiento depende en gran medida del mecanismo de indexación empleado.

En OO la existencia de jerarquías de herencia y de agregación, así como la posible invocación de métodos en los lenguajes de consulta orientados a objetos exigen la existencia de mecanismos de indexación que

permitan un procesamiento eficiente de las consultas bajo estas condiciones. Son muchas las técnicas de indexación en orientación a objetos que se han propuesto [10], sin embargo, no se puede concluir que una de ellas sea mejor que las demás en todas las situaciones o al menos bajo las condiciones en las que han sido probadas. Sin embargo, en la mayoría de los SGBDOO existentes en el mercado la técnica de indexación empleada es una o un conjunto fijo de ellas, y el diseñador no tiene posibilidad de seleccionar el esquema de indexación, ni por supuesto añadir nuevos esquemas.

Es por ello que el mecanismo de indexación para este SGBDOO debe incluir una serie de características (incluidas ya en este primer prototipo):

- *Distintos esquemas de indexación.* El sistema permitirá el uso de distintos esquemas de indexación. Inicialmente permite Single Class, CH-Trees y Path Index.
- *Selección del esquema de indexación.* Permitirá seleccionar el esquema de indexación que se considere más apropiado en función del tipo de consulta de que va a ser objeto la clase (o jerarquía de clases) que se está tratando. Gracias a la herencia y polimorfismo esta selección puede llevarse a cabo fácilmente.
- *Independencia del tipo de dato.* Implica que el índice puede ser ejecutado sobre cualquier tipo de datos (no simplemente sobre los tipos simples). Para conseguir esto, el mecanismo permite al usuario definir sus propios operadores de comparación.

5.3.2 Lenguajes

En este primer prototipo está implementado el estándar ODMG 2.0 ligado al lenguaje Java. En este apartado fue necesario la creación de un traductor de Java a Carbayón (lenguaje de la máquina abstracta), así como la traducción de las instrucciones de manipulación de la base de datos convirtiéndolas en las correspondientes llamadas a los métodos proporcionados por el objeto que representa el motor de la base de datos.

<pre> Class Professor AGGREGATION /* atributos de clase */ LastName:String; FirstName:String; Age:Integer; /* atributos relación */ Teaches:Tset; METHODS FieldbyName(ParamName:String):Object; REFS Bres:Bool; INSTANCES Param1str:String('LastName'); Param2str:String('FirstName'); Param3str:String('Age'); CODE ParamName.Equal(Param1str):bRes; JFD bRes, Label1; Assign rr, LastName; Jmp Fend; </pre>	<pre> Label1: ParamName.Equal(Param2str):bRes; JFD bRes, Label2; Assign rr, FirstName; Jmp Fend; Label2: ParamName.Equal(Param3str):bRes; JFD bRes, Label3; Assign rr, Age; Jmp Fend; Label3: Fend: Exit; ENDCODE /* métodos de clase */ GetAge():Integer CODE ENDCODE ENDCLASS </pre>
---	--

Figura 4. Código Carbayón generado para la especificación en ODL de la clase Professor

A este primer prototipo sería interesante incorporarle la capacidad de distribución aprovechando la distribución del SO, y en el apartado siguiente se comentan brevemente algunas de las ventajas que de ello se derivan.

6. Ventajas de la existencia de un SO distribuido para el SGBDOO

La arquitectura distribuida para el sistema integral se obtendrá cuando se disponga de un sistema operativo que nos proporcione distribución más un conjunto de máquinas abstractas conectadas en red. El SGBDOO expuesto anteriormente se podría aprovechar de una serie de características ofrecidas por el sistema de distribución, cara a convertirse en un SGBDOO distribuido.

6.1 Procesamiento distribuido de consultas

El procesamiento de consultas en bases de datos distribuidas se basa en generar y ejecutar estrategias para descomponer la consulta en operaciones que son enviadas a varios nodos, ordenando estas operaciones y ensamblando los resultados para construir el resultado de la consulta. Pero además hay que considerar que los objetos pueden estar formados por otros objetos que se encuentran distribuidos sobre diferentes nodos.

El procesamiento distribuido sería posible en el SGBDOO gracias a la posibilidad de invocación remota de métodos que nos proporciona ya el sistema de distribución del SO, así como a la transparencia de localización de los objetos favorecida en gran medida por la existencia de un identificador de objetos independiente de la ubicación física de los mismos.

6.2 Duplicidad de los objetos

Una de las características de un SGBDOO distribuido es la posibilidad de que los objetos aparezcan duplicados en más de un sitio pero de forma totalmente transparente para el usuario. Esta posibilidad incrementa la confiabilidad del sistema de forma que si los ordenadores de un determinado nodo se quedan fuera de servicio el resto de la red puede seguir funcionando. Los usuarios no dependen de la disponibilidad de una sola fuente para sus datos. Esto además permite colocar los datos cerca del punto de su utilización, de forma que el tiempo de comunicación sea más corto reduciéndose el tráfico de la red.

En el SGBDOO sería posible duplicar los objetos, incluyendo no solo el estado de los objetos sino también la computación. Para ello se empleará un meta-objeto, similar al que proporciona la migración, pero a diferencia de este no elimina la copia original del objeto.

6.3 Escalabilidad

Los sistemas distribuidos permiten variar su tamaño de un modo sencillo. Se pueden agregar ordenadores adicionales a la red conforme aumentan por ejemplo, el número de usuarios y la carga de procesamiento.

La escalabilidad que nos podría proporcionar el SGBDOO en función de las posibilidades ofrecidas por el SO distribuido son:

- Estática. Proporciona un procesamiento distribuido de los objetos, basándose en una configuración inicial escalada en función del número de usuarios, el volumen de información y la futura carga inicial.
- Dinámica. Posibilita la migración de la información de forma dinámica a otros gestores de bases de datos en función de un meta-objeto (System) que indica la carga del sistema.

7. Conclusiones

La distribución en la mayoría de los SGBDOO actuales se consigue en base a una arquitectura cliente servidor en la que es básica la determinación de las funciones que van a ser realizadas por ambos (cliente y servidor), así como la unidad de comunicación entre ellos.

Además, estos SGBDOO están contruidos sobre sistemas operativos tradicionales con los inconvenientes que ello supone: abstracciones no adecuadas del SO para afrontar determinadas necesidades de los SGBD,

interfaces no adecuadas de los SO en forma de llamadas a procedimientos, y problemas de interoperabilidad entre modelos de objetos.

En este artículo se presenta un SGBDOO construido sobre un sistema integral OO, cuyo corazón es una máquina abstracta con arquitectura reflectiva que proporciona un modelo de objetos que comparten el resto de elementos del sistema. El sistema operativo proporciona un sistema de distribución que permite invocación remota de métodos, migración y movilidad de objetos, y que el gestor podría aprovechar con las consecuentes ventajas que esto supone para su conversión a un SGBDOO distribuido. Entre esas ventajas se encuentran: *la capacidad de procesamiento distribuido* de consultas derivada de la transparencia de localización y de la posibilidad de invocación remota de métodos, *la duplicidad transparente de objetos* basada en la capacidad de movilidad de los objetos, y *la escalabilidad dinámica* facilitada por la capacidad de migración proporcionada por el sistema de distribución.

Referencias

- [1] Álvarez D., Martínez A.B., Cueva J.M. *Persistencia en Sistemas Operativos Orientados a Objetos. Ventajas para los Sistemas de Gestión de Bases de Datos*. I Jornadas de Investigación y Docencia en Bases de Datos. La Coruña, 1996.
- [2] Álvarez F., Tajés L., Díaz M.A., Álvarez D. *Introducing Distribution in an OS Environment of Reflective OO Abstract Machines*. 14th European Conference on Object-Oriented Programming (Workshop), 2000.
- [3] Álvarez D., Tajés L. et. al. *An Object-Oriented Abstract Machine as the Substrate for an Object-Oriented Operating System*. 11th European Conference on Object-Oriented Programming (Workshop). Jyväskylä 1997.
- [4] Booch G. *Object Oriented Analysis and Design with Applications*. Addison-Wesley, 1994.
- [5] Cattell R., Atwood T., Duhl J. et. al. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, 1994.
- [6] Cattell R., Barry D., Bartels D. et al. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.
- [7] Cueva J.M et. al. *El Sistema Integral Orientado a Objetos: Oviedo3*. II Jornadas sobre Tecnologías Orientadas a Objetos. Oviedo, 1996.
- [8] Mary Kirtland. *Designing Component-Based Applications*. Microsoft Press, 1999.
- [9] Martínez Ana B., Álvarez D., Cueva J.M., Ortín F. y Pérez J.A. *Incorporating an Object-Oriented DBMS into an Integral Object-Oriented System*. SCI/ISAS , Orlando 1998.
- [10] Martínez A.B., Cueva J.M, Álvarez D, y García M. *Técnicas de Indexación para BDOviedo3*. IV Jornadas sobre Tecnologías Orientadas a Objetos. Bilbao, 1998.
- [11] Ortín F., Martínez A.B, Álvarez D y Cueva J.M. *A Reflective Persistence Middleware over an Object Oriented Database Engine*. XIV Simposio Brasileiro de Banco de Dados (SBBD'99). Brasil, 1999.
- [12] Ozsu M. y Valduriez P. *Principles of Distributed Database Systems*. Prentice-Hall, Englewood Cliffs, NJ.
- [13] Ozsu T., Dayal U, Valduriez P. *Distributed Object Management*. Morgan Kaufmann, 1994.
- [14] Brian Smith. *Reflection and Semantics in a Procedural Language*. Tesis Doctoral, Massachusetts Institute of Technology, EE.UU. 1982.
- [15] Tajés L., Álvarez F., Díaz M.A, Álvarez D. *A Computational model for a Distributed Object-Oriented Operating System Based on a Reflective Abstract Machine*. 12th European Conference on Object-Oriented Programming (Workshop), 1998.