

Generalidades sobre OODBMS

Cristian Rossel Moraga

crossel@inf.udec.cl

Resumen.

En este artículo se estudia el modelo objeto aplicado a Sistemas de Gestión de Bases de Datos.

La importancia de trabajar sobre la base de un estándar para éste paradigma, queda reflejado en los numerosos empeños que se han realizado desde el mundo comercial, como también del mundo académico. Estándares como ODMG-93 y SQL3 son el premio al esfuerzo y a la dedicación, los cuales se describen genéricamente.

Introducción

A los nuevos servicios de las empresas, le corresponden nuevas funciones de aplicación. Esas modificaciones de aplicación deben poder hacerse rápido, sin tener que reescribir todo.

Corría la segunda mitad de la década de los 80', cuando comienzan a generalizarse en múltiples aplicaciones los sistemas de gestión de bases de datos orientadas al objeto (OODBMS), los cuales toman las ventajas del enfoque orientado al objeto, ya probados y los sistemas de gestión de bases de datos, siendo su principal virtud el ofrecer un muy buen desempeño en la gestión de datos complejos y multimediales.

Desde el punto de vista del desarrollador las ventajas están dadas en ganancias de productividad, dado su modelamiento más simple que facilita el desarrollo de aplicaciones. El modelamiento de aplicaciones es mucho más sencillo gracias al concepto de objetos complejos y el modelo obtenido es fácilmente comprensible para el usuario. Este modelo puede ser validado directamente por el cliente de la aplicación. El enfoque Orientado al Objeto favorece la reutilización, porque gracias a la encapsulación y la herencia, es fácil especializar un componente existente para responder a las necesidades particulares de la aplicación.

Desde el punto de vista del usuario final de los OODBMS, el mayor aporte es la calidad en términos de ergonomía, fiabilidad, evolución y desempeño. La mayor parte de los productos disponibles en el mercado, son productos cerrados, difícilmente adaptables a las especificaciones de la empresa: la ergonomía de la aplicación es fija, las reglas de cálculo son difícilmente modificables, etc. la tecnología objeto, gracias en particular a la encapsulación y la herencia, da productos desarrollos con un OODBMS más abierto y fácilmente adaptables. El usuario puede obtener un costo menor de las modificaciones de sus procedimientos, que van a integrar más fácilmente su medio ambiente (entorno).

Un poco de historia

La primera aparición del concepto data de 1984 con la proposición de David Maier y George Copeland de construir un DBMS desde Smalltalk, Copeland et al. (1984) [5].

Se pueden citar dos grandes proyectos desde 1985, el proyecto ORION de MCC en Austin, Texas y en 1986 el proyecto Altair, en Rocquencourt, Francia. En 1988, la primera generación apareció, seguida por una segunda generación en 1990.

Después de esta intensa actividad, pareciera necesario definir de una manera precisa el concepto de OODBMS (Sistemas de Gestión de Bases de Datos Orientadas al Objeto). En efecto, contrariamente al caso de los sistemas relacionales que primero fueron definidos formalmente en el artículo original de Codd, después se generaron prototipos y finalmente transformados en producto, no hubo un principio de especificación precisa de lo que debía ser un OODBMS.

En 1989, el paper "*The Object-Oriented Database System Manifesto*"^[1] aunque con un enfoque demasiado limitado en temas de administración, Stonebraker et al. (1990), propone una definición compuesta de tres tipos de reglas que deben respetar un OODBMS:

- **Reglas Obligatorias:** Las cuales el sistema debe imperativamente seguir para merecer la calidad de OODBMS.
- **Reglas Facultativas:** Lineamientos suplementarios del sistema, pero que no son indispensables.
- **Reglas Abiertas:** Propiedades alternativas del sistema que puede ejercer.

El conjunto de reglas es rápidamente admitido por la comunidad científica y comercial como un consenso mínimo.

A principios de la década de los 90', la comercialización efectiva de sistemas progresó rápidamente y fueron desarrolladas varias aplicaciones.

En 1992, los principales distribuidores se juntan para definir un estándar que asegure la portabilidad de las aplicaciones de un sistema a otro, y en Octubre de 1993 es publicado el estándar del ODMG (Grupo de Administración de Bases de Datos Orientadas al Objeto), Catell et al. (1993) ^[4]. Se cuenta con una tecnología madura y adaptada al mercado, una oferta rica y diversificada, una demanda para este tipo de producto y un estándar realista (*ODMG-93* es un estándar para bases de datos al objeto puras, en contraste con el estándar *SQL3* nacido para bases de datos relaciones extendidas basadas en objetos, siendo ésta compatible con el modelo relacional clásico).

El estándar ODMG-93: *Un estándar para bases de datos orientadas al objeto puras*

Es el resultado de trabajos que duraron 18 meses por los 5 principales distribuidores de OODBMS. Su objetivo fue asegurar la portabilidad de las aplicaciones de un sistema a otro. En este objetivo son definidas tres interfaz:

1) ODL (Lenguaje de Definición de Objetos)

El lenguaje de definición del objeto permite definir el modelo de datos. Es compatible con IDL, el lenguaje del OMG (Grupo de Administración de Objetos). Permite la definición de objetos complejos, de relación entre esos objetos y de métodos asociados a dichos objetos.

2) OQL (Lenguaje de Consulta al Objeto)

El lenguaje de requerimientos permite consultar los objetos de estructuras complejas, de enviar mensajes a objetos, efectuar join y otras operaciones de tipo asociativo. Su sintaxis es del tipo SQL.

3) Conexión vía C++ y Smalltalk.

Esta interfaz ("bindings", enlazamientos), especifica como se debe hacer la programación en C++ o Smalltalk de una aplicación sobre una base de datos que ha sido declarada en ODL. La conexión es basada sobre la noción de "puntero inteligente" que permite manejar los objetos persistentes como objetos ordinarios vía punteros persistentes.

Definiendo las OODBMS

Un OODBMS ofrece todas la funcionalidad de un sistema de gestión de bases de datos, al igual que todas las de un sistema objeto.

Conceptos DBMS

La tecnología de gestión de bases de datos (DBMS), nació a fines de los años 60. Las tecnologías de implementación de esos sistemas han evolucionado, su arquitectura ha emigrado hacia una arquitectura Cliente/Servidor, pero los servicios ofrecidos (En particular a su nivel físico) son los mismos. La figura 1 muestra un DBMS tradicional considerando sus aspectos más característicos.

Un DBMS ofrece facilidades de almacenamiento, de acceso, manipulación y de compartimento de grandes volúmenes de datos. Esos datos pueden ser muy abultados, ellos no están ni en memoria principal ni en memoria virtual. Es el DBMS quien asegura la gestión de diferentes niveles de jerarquía de memoria, el programador no hace la diferencia entre un dato en memoria y un dato en disco. La gestión del disco asegurada por el DBMS debe ser transparente y ofrece buenos desempeños. Ella debe ofrecer mecanismos tales como gestión oculta de indexación, reagrupamiento de objetos sobre los discos (debe proveer una forma adecuada de reagrupar los objetos de un nivel físico, a un nivel superior), etc.



Figura .1: DBMS Tradicional, Manola (1994) [6].

El DBMS garantiza igualmente la coherencia de los datos en casos de actualización simultánea por varios usuarios (mecanismo de transacciones), su integridad en caso de operaciones incorrectas por un programa o un usuario, su fiabilidad en caso de rollback o commit y su confidencialidad en caso de accesos mal hechos o accidentales (mecanismos de seguridad).

Los lenguajes de consultas nacieron con las DBMS Relacionales, permitiendo simplemente interrogar a la base de datos sin tener que escribir un programa específico. Un lenguaje de consulta es un elemento indispensable que uno debe encontrar en todo SGDB.

Los conceptos de Objetos

El objeto reagrupa en una misma entidad una parte estática: los datos del objeto y una parte dinámica: el conjunto de operaciones, también llamados métodos, que pueden ser aplicados a cada objeto y son conocidos en el mundo exterior a través de una interfaz pública (ver figura 2) constituida por algunas operaciones (métodos). La parte estática así como la implementación de operaciones están ocultos; es un principio conocido como "encapsulación". Este mecanismo garantiza la independencia entre la vista externa de un objeto (su parte pública) y su vista interna (su implementación). La encapsulación impone un buen nivel de modularidad.

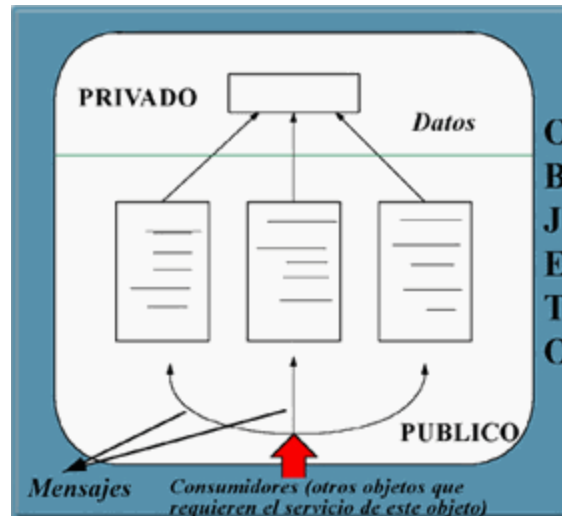


Figura 2: Composición de un Objeto.

La parte estática del objeto puede reagrupar informaciones alfanuméricas, gráficas, textuales, sonoras, etc. Ella puede ser atómica (los objetos atómicos son del tipo de base del sistema: enteros, reales, caracteres, strings, booleanos) o compleja (constituida a partir de otros objetos, atómicos o complejos).

La parte dinámica del objeto puede estar constituida de funciones de archivos, cálculos, de búsqueda de información, etc. Contrariamente a lo que existe en la programación clásica, las operaciones son subordinadas a los objetos; el objeto no es solamente caracterizado por lo que es, sino también por lo que hace, ocultando la estructura interna de un objeto y la implementación de las operaciones.

Cada objeto tiene una identidad propia (ver figura 3), independiente de su valor. Podemos actualizar el valor de un objeto sin alterar su identidad. El sistema maneja su identidad, atribuyendo a cada objeto un identificador que asegura unicidad.

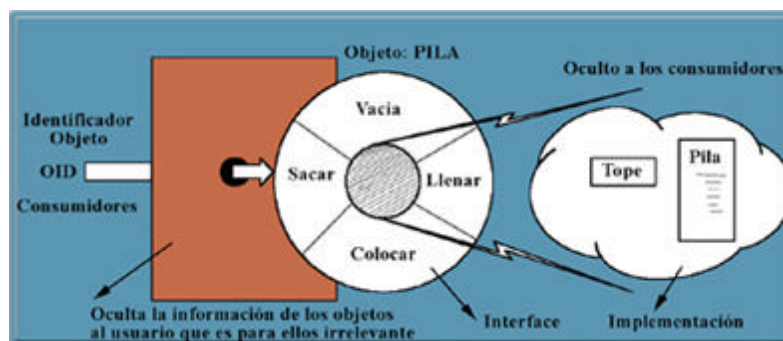


Figura 3: La identidad objeto.

La noción de identidad de objeto guarda relación con la composición del objeto, diferenciándose de otros objetos.

Conceptos como la herencia permiten definir una clase (la clase define una estructura y un comportamiento común, a varios objetos) de objetos a partir de una clase ya existente (herencia simple) o de varias otras clases (herencia múltiple). La clase creada recupera no solamente su estructura sino además sus métodos de su(s) clase(s) padre(s). La herencia trae una descripción compacta bien estructurada del esquema de aplicación. Es una técnica de clasificación de objetos que evita la duplicación de código facilitando la reutilización de propiedades de estructuras y comportamientos ya definidas.

Los objetos se comunican entre ellos por envío de mensajes, lo que se quiere decir, es que transmiten órdenes a otros objetos. Cuando se recibe un mensaje por un objeto, éste lo ejecuta. Él puede crear nuevos objetos y enviar nuevos mensajes.

Los métodos asociados a clases diferentes, pueden tener el mismo nombre: la elección del método que sea efectivamente llamado es aplazada hasta el momento de la ejecución (enlazamiento dinámico), dependiendo de la persistencia de la clase del objeto del cual el método es llamado en tiempo de ejecución. Ese mecanismo de enlace dinámico aumenta la independencia entre los métodos y los objetos y permite sumar nuevas clases sin modificar ni recompilar los programas existentes. Por ejemplo, la aplicación de gestión de pago maneja diferentes tipos de contratos, cada uno con su método de cálculo de sueldo, basta con crear una nueva subclase de la clase "contrato" con un método de cálculo de sueldo correspondiente a ese algoritmo. Esos nuevos contratos son entonces inmediatamente tomados en cuenta por los programas existentes, sin modificación particular del código.

La herencia y el enlazamiento dinámico permiten aliviar los programas y el trabajo del programador (por efecto de la reutilización) cuando se activan los módulos de la aplicación.

Se hace notar que el modelamiento de objetos se puede realizar bajo una forma gráfica, pudiendo tener ciclos.

Conceptos específicos de OODBMS

La noción de objetos complejos optimiza la representación de las estructuras complejas y acortando la distancia que separa el modelamiento de un escenario del mundo real y su implementación. Los objetos complejos permiten así, un modelamiento más intuitivo de datos de una aplicación, su presentación en la base de datos está mas cerca de la realidad. La estructura compleja de los objetos es manejada por punteros lógicos. Esos punteros reemplazan la utilización de uniones relacionales, e inducen así una ganancia de desempeño, particularmente cuando la cantidad de datos es importante. La siguiente figura muestra un esquema típico de una OODBMS:

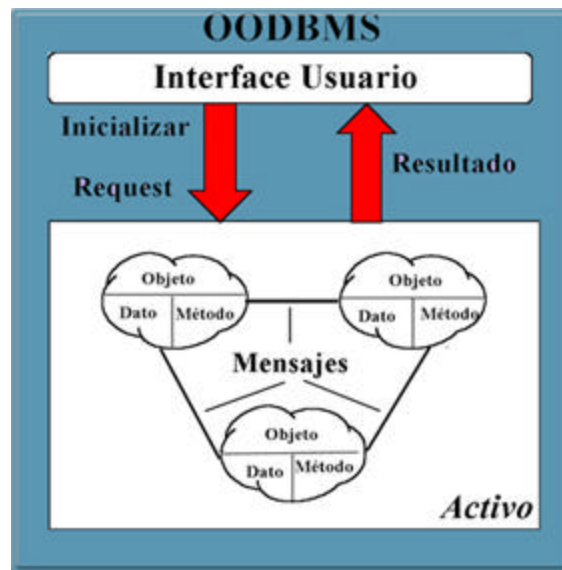


Figura 4: Una OODBMS tradicional, Manola (1994) [6].

El concepto de identidad de objeto, tiene repercusiones particularmente interesantes en el contexto de un DBMS. La distribución de objetos permite limitar la tabla de la base de datos. Permite una mejor gestión de actualización, y es más rápida, ya que tiene menos objetos que modificar. Ofrece igualmente una gestión automática de integridad referencial (desaparición de referencias a objetos destruidos), problema crucial de las bases de datos.

Con las generaciones previas de DBMS, los desarrolladores tuvieron un nivel bajo de productividad. Esto es en particular dado por un problema de funcionalidad entre aquellas DBMS y los lenguajes de programación utilizados. En efecto, los tipos de datos de esas categorías de herramientas son incompatibles, el programador debe asegurarse por si mismo de la conversión de datos entre la base de datos y el lenguaje: Esto impone el desarrollo suplementario que hay que realizar para mantener la consistencia de la aplicación en su ejecución.

La falta de coincidencia entre lenguajes orientados a tablas, tales como SQL, y los lenguajes comunes, significa que se necesita un lenguaje separador para la manipulación de datos (DML), existiendo impedimentos de acoplar estilos declarativos y basados en procedimientos entre los tipos de sistemas del lenguaje de aplicación y de bases de datos, dando lugar a una pérdida de información en la interfaz, y obstaculizando la comprobación automática de tipos.

Para solucionar este problema los OODBMS ofrecen el concepto de completitud; que es la percepción de escribir completamente una aplicación utilizando el DBMS como un único lenguaje (tal como los lenguajes estándar según la norma ODMG; C++, Smalltalk o Java), sin tener que recurrir a los recursos de los lenguajes externos. Este concepto suprime el problema de funcionalidad, ya que los datos son manipulados de la misma manera en la base de datos que por el lenguaje de programación. En efecto, es el mismo modelo de datos, que es soportado por el lenguaje objeto de desarrollo de la aplicación y por el OODBMS. Los OODBMS ofrecen todo un lenguaje de programación completo integrando los accesos a las bases de datos.

Características de las ODBMS

- Un completo modelo de bases de datos, con rasgos tales como; la manipulación fija y el acceso de asociación.
- Un modelo orientado al objeto que respalda objetos complejos, identidad del objeto,

encapsulamiento, clases, caracteres, extensibilidad e integridad computacional.

- Un DDL (Data Definition language, define tipos de objetos, y además el usuario puede declarar procedimientos y variables asociadas con los tipos de objetos) real con rasgos de manipulación de esquemas.
- La capacidad de almacenar y manipular tanto los datos (objetos) como los metadatos (clases y métodos) en el propio sistemas.
- Un DML (Data Manipulation Lenguaje, por lo general contiene un lenguaje de consultas y un componente de lenguaje de programación), a través de un lenguaje de consulta completo, declarativo.
- Independencia de datos lógica y física (esto es, un DDL físico y uno lógico y la capacidad para modificar el esquema físico sin cambiar la aplicación).
- La capacidad de manipular cantidades de datos enormes.
- La capacidad de desarrollar aplicaciones completas en un medio único.

Los Desempeños

El problema de los desempeños es esencial para los DBMS. El mercado de sistemas orientados al objeto se desarrolla porque esos sistemas ofrecen desempeños mejorados con respecto a los sistemas relacionales. Existen tres tipos de benchmarking ("*pruebas de rendimiento*") que permiten medir estos sistemas.

- Benchmarking 001, Rubenstein et al. (1987) [7], Catell et al. (1992) [3], está orientado a aplicaciones que tienen por objetivo describir la aplicación en cuanto a su tipo de concepción (refiriéndose a su forma de generación, formación). Manipula los objetos complejos entre el servidor y una estación de trabajo.
- Benchmarking de Hypermodel, fue hecho por un grupo de navegadores de conceptos en sistemas. Se basa sobre la aplicación del tipo hipertexto y mide el tiempo de acceso a los objetos.
- Benchmarking 007, Carey et al. (1993) [2], hecho en 1992 para las limitaciones del Benchmarking 001, abarcando un gran número de operaciones, transacciones, requerimientos y el control de versiones. El 007 denota tres diferentes tamaños:
 - Pequeño: La base de datos en memoria principal.
 - Mediano: La base de datos está contenida en memoria virtual y una parte en la memoria principal.
 - Grande: La base de datos está en memoria virtual.

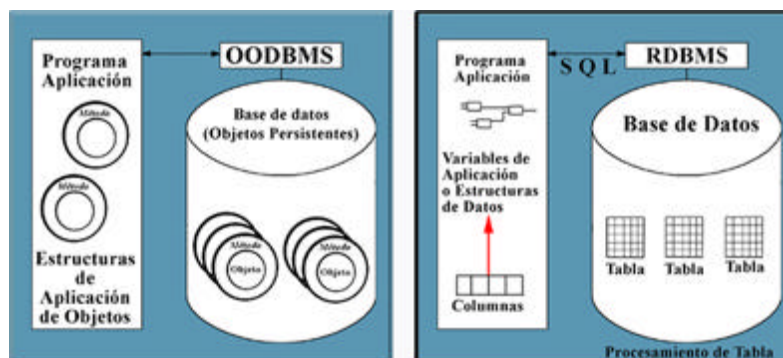


Figura 5: Comparación Arquitectura OODBMS vs RDBMS, Manola (1994, pág. 3,19) [6].

Los OODBMS dominan en desempeño con respecto a los sistemas relacionales sobre las aplicaciones manipulando objetos complejos. Esto es por una sencilla razón; que los sistemas relacionales fueron hechos y diseñados para efectuar algunas operaciones simples (selección, proyección, etc.), y los

OODBMS tienen por finalidad manipular objetos estructurados y complejos. En cuanto a una comparación arquitectónica entre el modelo relacional y el modelo objeto aplicado a DBMS, la figura 5 muestra su correspondiente composición habitual.

Los aportes a la tecnología

Los OODBMS permiten llegar a nuevos dominios para los cuales las bases de datos tradicionales son renuentes a ser aceptadas.

Su fuerte es en ambientes donde hay una necesidad de datos no estándar, es decir, de aquellos que uno manipula textos estructurados o no estructurados, imágenes, gráficos, sonidos, videos, documentos o programas. Se trata entonces de ambientes donde la estructura de los datos es tan compleja que representarla en un modelo tradicional es ineficaz.

Se trata de dominios o tipos de datos que al usarlos no permanecen fijos desde el inicio y son variables en el tiempo.

Son dominios comunes, por ejemplo:

- CAD
- Gestión de datos técnicos
- Cartografía
- Multimedia.
- Sistemas distribuidos y cliente/servidor.
- Bases de datos multimedia.
- Correo por voz.
- GIS

En todos estos dominios, la tecnología de OODBMS aporta los desempeños mejores y un desarrollo más eficaz de aplicaciones.

Los OODBMS disminuyen los costos lógicos de desarrollo.

Esta baja de costos es obtenida en opinión de connotadas figuras ligadas a DBMS (Por citar algunos: Atkinson et al.^[1], Bancilhon, Graham), por un acortamiento del ciclo de análisis, concepción, codificación, depuración, testeo, mantención y evolución. Mejoramiento obtenido por:

- La reutilización del componente lógico.
- La disminución de código.
- La capacidad de modelamiento directo de información compleja.
- La mejor integración a los lenguajes de programación.
- Desarrollo rápido de prototipos de aplicaciones.
- Utilización de medio ambiente gráfico de desarrollo.
- Utilización de herramientas de generación de interfaz gráfica de realización.

Los OODBMS permiten producir aplicaciones gráficas de mejor calidad.

Las aplicaciones son mejoradas en dos puntos esenciales:

- Los desempeños en el caso de la manipulación de datos complejos.
- La calidad "industrial" de aplicaciones para la facilidad de mantención, su capacidad de crecimiento y posibilidad de ser personalizadas en dominios específicos.

Los OODBMS permiten recuperar y mejorar las aplicaciones existentes.

Para los sistemas que disponen de una interfaz C++ estándar, el mecanismo consiste en tomar una aplicación existente en la cual la persistencia es asegurada por un sistema de archivos que al migrarlos al OODBMS se reemplaza el acceso a los archivos por el almacenamiento en el OODBMS. Así, el costo de migración es mínimo y la aplicación es bastante mejorada. En efecto, resultando en:

- Una simplificación de código (el acceso a los objetos de la base de datos es inmediata y sin traducción).
- La capacidad de fiabilidad y compatibilidad de los datos.

Los Mercados**1. Aplicación en Sistemas de información geográficos.**

Para los sistemas de información geográficos o para toda aplicación en la cual hay una dimensión espacial o geográfica (la cartografía de una región, la topología de una zona o el plano de un edificio), los desarrolladores de estas aplicaciones necesitan la tecnología de objetos; ella ofrece un mayor desarrollo y mejores desempeños.

2. Gestión de datos técnicos.

Porque permiten almacenar los datos de naturaleza variada y de tipo extensible, los OODBMS son elegibles como sistemas de almacenamiento para este tipo de aplicaciones, que incluyen la gestión de datos científicos experimentales, la gestión de datos asistidos por computador (CAD) y la documentación técnica.

3. Aplicaciones Multimedia.

Para toda aplicación que manipula gráficos, imágenes, animación y voz, los OODBMS son los primeros en la elección de los desarrolladores.

 **Conclusión**

Las OODBMS representan una tecnología innovadora y un mercado en pleno desarrollo. Después de una fase extensiva de evaluación de la tecnología y de experimentación vía prototipos, varios desarrolladores generan aplicaciones a un nivel real de explotación. Estos desarrolladores son los primeros en aprovechar las ventajas de esta tecnología, beneficiándose de una ventaja competitiva sobre las demás organizaciones.

Los beneficios de estos sistemas para aplicaciones convencionales, son un mejoramiento del proceso de desarrollo (diseño, codificación, mantención y evolución se desarrollan mas fácilmente y a menor costo) y de la aplicación resultante (las aplicaciones están mejor estructuradas, más confiables y fáciles de construir y modificar). Para nuevas aplicaciones sus beneficios radican en la capacidad para tratar con nuevos tipos de datos (texto, imágenes, figuras, sonidos, etc.).

Se señala a juicio personal del autor que las virtudes para el desarrollador están dadas por ganancias en productividad y para el usuario final en cambio es la disposición de más desempeño y mejor calidad, faltando solamente un acelerador para aumentar su madurez y credibilidad.

 **Referencias**

- [1] Malcom P. Atkinson, Francois Bancilhon, David DeWitt, Klaus Dittrich, David Maier, Stanley Zdonik,: "*The Object-Oriented Database System Manifesto*", Proc. First Intl. Conf. On Deductive and Object Oriented Databases, Elsevier Science Publishers, Amsterdam, 1989.
- [2] M.J. Carey, D.J. Dewitt, J.F. Naughton,: "*The OO7 Benchmark*", Proceeding of the ACM SIGMOD conference, Washington DC, 1993.
- [3] R.G.G Catell, J. Skeen,: "*Object Operations Benchmark*", ACM Transaction on Database Systems, Vol 17(3), 1992.
- [4] Rick Catell, Tom Atwood, Josh Duhl, Guy Ferran, Mary Loomis, Drew Wade,: "*The ODMG-93 Estándar*", Morgan Kaufman, 1993.
- [5] G. Copeland and D. Maier,: "*Making Smalltalk a Database System*", proc. SIGMOD, 1982.
- [6] Frank Manola: "*An Evaluation of Object-Oriented DBMS Development 1994 Edition*", GTE Laboratories Incorporated, 1994.
- [7] W. B. Rubenstein, M.S. Kubicar and R.G.G Catell,: "*Benchmarking Simple Database Operations*", Proc. SIGMOD conference, San Francisco, 1997.