INTRODUCCIÓN

La programación de aplicaciones que tienen acceso a bases de datos es algo que existe ya desde hace muchos años atrás, pero ni con todo este tiempo recorrido se ha logrado encontrar una solución definitiva a uno de los problemas más serios en el ámbito de la programación como es el problema de la diferencia de impedancia de objetos. Este problema se presenta al momento de querer conectar nuestra aplicación orientada a objetos, con un manejador de bases de datos que trabaja de manera relacional ya que estos dos componentes que forman parte principal de la aplicación trabajan de formas diferentes.

Una de las soluciones que es muy usada en la actualidad por la mayoría de programadores comprende en trabajar la programación insertando cadenas de texto quemadas con las instrucciones SQL necesarias sin pensar seriamente en las consecuencias que podrían resultar a corto o largo plazo, sin embargo esta opción tiene fuertes limitantes como son por ejemplo elevar los costos de modificación en el código, o de clásicos ataques de seguridad producidos por inyección SQL en la aplicación.

A mediados de los años noventa comenzaron a darse origen a nuevas técnicas de mapeo objeto relacional y de persistencias de objetos que tratan en su mayoría de

trabajar de forma transparente presentándose como una solución atractiva a este problema. De forma paralela a inicio de los 90 comenzaron aparecer las primeras bases de datos orientadas a Objetos con la meta principal de la reutilización de código aplicada al proceso de producción, estas bases de datos se han ido desarrollando con el pasar de los años si tenemos por ejemplo Db4objects .

El propósito del siguiente texto es justamente ayudar a determinar cual es el mejor método existente en la actualidad para eliminar la diferencia de impedancia de objetos sea esta la utilización de motores de persistencia (técnica mapeo objeto-relacional) o el uso directo de bases de datos orientadas a objetos en nuestras aplicaciones.

CAPÍTULO I

EL PROBLEMA

¿Cual es la mejor metodología para evitar la impedancia de Objetos?

Actualmente la mayoría de Bases de Datos usan un modelo relacional mientras que los programas en su evolución han ido usando un modelo orientado a objetos, aquí es cuando se presenta un conflicto al usar estos dos componentes en una misma aplicación debido a que hablan idiomas diferentes y su comunicación se dificulta esto es lo que se conoce comúnmente como impedancia de objetos.

SITUACIÓN CONFLICTO NUDOS CRÍTICOS

El método más usado por los programadores y el más sencillo para hacer que estos dos elementos puedan comunicarse consiste en hacer que toda la aplicación use un mismo modelo teórico relacional, esto se logra introduciendo las sentencias SQL en las clases de negocio de la aplicación.

Esta opción no es la más recomendable debido a que se genera una gran dependencia entre las clases de negocio y el esquema de la base de datos. Además se pierde la facilidad de mantenimiento de la aplicación. Por esta razón caemos en la necesidad de buscar otras opciones que nos permitan llevar a cabo la construcción de una aplicación más eficiente y con mejor rendimiento.

CAUSAS Y CONSECUENCIAS DEL PROBLEMA

Si no se escoge el modelo correcto y simplemente se opta por lo más sencillo o por lo que normalmente se realiza por costumbre, se puede originar una dependencia entre las clases de negocio y el esquema de la base de datos, perder la facilidad de mantenimiento de la aplicación y entre otras muchas situaciones.

Así como también se puede dejar de aprovechar las grandes ventajas que pueden ofrecer otras metodologías.

DELIMITACIÓN DEL PROBLEMA

El proyecto surgió debido a la necesidad personal de poder determinar la mejor solución para la impedancia de objetos. Este problema se da en una gran cantidad de empresas que desarrollan software con tecnologías nuevas como .Net y donde la predominancia del conocimiento sobre el modelo Entidad-Relación que lleva a la búsqueda automática de SGBS relacionales tales como MySQL o el popular Oracle.

Las Bases de Datos Orientadas a objetos actualmente no son tan usadas en el mercado en comparación con la metodología ORM (Object/Relational Mapping) que es usada normalmente con las bases de datos Relacionales las mismas que han logrado una gran madurez e impacto muy grande en el mercado actualmente.

FORMULACIÓN DEL PROBLEMA

¿Qué pasa si queremos almacenar Objeto Creados por un lenguaje orientado a Objetos en una base de datos Relacional? Es aquí en donde se produce el problema de la "Inadaptación de impedancia" entre Objetos y datos relacionales.

El problema de la "impedancia de objetos" lleva a las empresas a tener que desarrollar y mantener una cantidad considerable de código fuente que no aporta al objetivo final del desarrollo en curso que es satisfacer la necesidad del cliente. Por ello el trabajo que implica desarrollar tal código podría ser mejor Aprovechado.

Como lo menciona **Ian Graham -1996 "La resolución del desacoplo de impedancias apoya las arquitecturas cliente/servidor, en las que se almacena una mayor parte de la semántica en la base de datos. Esto significa que se reduce el número de accesos, y que se mantiene mejor la integridad." (p. 211)**

En esta investigación trataremos de determinar, cual es la mejor opción que existe actualmente para no tener una diferencia de impedancia.

Actualmente existen opciones que se pueden manejar para mejorar la comunicación entre las aplicaciones y las Bases de datos como son:

- Una de las soluciones más conocidas es Hibernate, que es considerado el servicio de persistencia de objetos en bases de datos relacionales de mejor rendimiento o metodología ORM. Sin embargo, debe considerarse que el hecho de tener que realizar el mismo trabajo que efectúan los desarrolladores,

que hacen el mapeo entre sus objetos y las bases de datos relacionales consume más recursos comparado a trabajar directamente sobre las bases de datos relacionales como lo menciona **Ian Graham -**1996 **"Los sistemas de gestión de bases de datos relacionales (SGBDR) pueden almacenar objetos en forma de atributo o de tablas, pero la cantidad de tráfico entre la aplicación y tales bases de datos se ve incrementada como consecuencia de la necesidad de volver a ensamblar los objetos después de su recuperación, y a la inversa para su mantenimiento". (p. 212)**

- Otra solución que ha existido desde hace un tiempo son los SGBDOO, de los cuales poco se conoce pues en su gran mayoría son de costos elevados, por lo tanto de no tan fácil acceso. Sin embargo las BDOO son ideales para almacenar y recuperar datos complejos permitiendo a los usuarios su navegación directa (sin un mapeo entre distintas representaciones).

Los motores de Persistencia proporcionan grandes ventajas para un desarrollo pero el problema es que no son muy usados debido a que muchos de ellos son un poco complejos en cuanto a su utilización, por esta razón los desarrolladores tienden a escoger otros métodos para facilitar la comunicación entre sus programas orientados a objetos y sus bases de datos relacionales.

De la misma manera las Bases de Datos Orientadas a Objetos también proporcionan muchas ventajas para una aplicación, pero no son tan usadas como las relacionales debido a que estas últimas tienen una madurez muy significativa dentro del mercado.

Debido al paradigma existente de las Bases de datos Relacionales es que se han originado varias interrogantes que se convierten en un problema para los desarrolladores a la hora de diseñar la arquitectura de sus aplicaciones, entre estas preguntas están: **¿Es hora de Usar Bases de Datos Orientadas a Objetos para solucionar el problema de impedancia de objetos? ó sería mejor mantenerse con las Bases de datos Relacionales y usar motores de persistencia para facilitar la comunicación?.**

EVALUACIÓN DEL PROBLEMA

A continuación se presentarán los aspectos que nos permitirán la evaluación del Problema.

Evidente: Nuestro problema es evidente debido a que se manifiesta claramente en el rendimiento de las aplicaciones que se desarrollan, lo que nos permitirá medir y observar todos los aspectos positivos y negativos que lo rodean.

Delimitado: Es delimitado debido a que sólo podremos realizar la investigación en base a las metodologías existentes hasta la actualidad y en base a pronósticos futuros.

Claro: La investigación tiene aspectos concretos que nos proporcionan una visión más clara y comprensiva para poder proyectar ideas más concisas y de esta forma sacar conclusiones más eficientes y efectivas.

Relevante: Es relevante porque permitirá a toda la comunidad desarrolladores y a los diseñadores de software conocer cual es la metodología más eficiente y efectiva para aplicar al proceso de producción y solucionar el problema de impedancia de objetos lo cual se requiere resolver científicamente.

Original: En cierta parte es original debido a que para la investigación será tomada en cuenta una base de datos OO llamado DB4O sobre la cual no se conoce mucho aún y tiene muchos ámbitos novedosos sobre los cuales podemos investigar.

Factible: Es factible porque a través de esta investigación podremos encontrar soluciones a través del tiempo en base a las debilidades que tiene cada metodología.

OBJETIVOS

GENERAL

Determinar mediante un estudio comparativo cual es la mejor solución para impedancia de objetos y al mismo tiempo saber si es hora de que crezca la presencia de la metodología BDOO en el mercado.

ESPECÍFICOS

El objetivo principal, lo podemos descomponer en los siguientes objetivos específicos.

- Estudiar la factibilidad, estructura, características, fortalezas y debilidades tanto de la metodología **BDOO** como de la **ORM**.

- Realizar la evaluación y análisis que permita obtener los resultados que arrojan ambas metodologías.

- Ayudar a los Proveedores de Software a tomar decisiones con respecto a la metodología que deberían emplear para sus aplicaciones Orientadas a Objetos.

JUSTIFICACION E IMPORTANCIA

Muchas veces los desarrolladores caen en el paradigma de que si lo Orientado a Objetos con el Modelo Relacional son incompatibles o no, pero hay que tener muy en cuenta que el hecho de enviar sentencias SQL a algún manejador de BD Relacional, no significa que sean compatibles.

Por esta razón es necesario que exista algún estudio que nos ayude a darle una solución a este paradigma y que le permita tanto a los desarrolladores como a los arquitectos de software encontrar capacidades para la obtención y almacenamiento de los datos y que reduzcan el tiempo de desarrollo y de mantenimiento de las

aplicaciones y al mismo tiempo poder obtener resultados que permitieran tomar decisiones sobre que metodología aplicar en proyectos futuros para mantener una comunicación entre la aplicación y la base de datos de una manera más efectiva, ya sea con el uso de bases de datos orientadas a objetos o a través del uso de los motores de persistencia y bases de datos relacionales.

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.

CAPÍTULO II

MARCO TEÓRICO

ANTECEDENTES DEL ESTUDIO

Actualmente existen muchos artículos y benchmark que hacen comparaciones entre las Base De Datos Relacionales Vs. Base De Datos Objeto-Relacionales pero no existe en la actualidad algún estudio que profundice en las fortalezas, debilidades y oportunidades especialmente de las bases de datos Orientadas a Objetos que nos ayude a determinar mediante pruebas de evaluación si es hora de comenzar a utilizarlas con mayor frecuencia en las aplicaciones que utilizan las empresas o cuales son las probabilidades que tiene esta base de datos de madurar en el mercado y ponerse a la altura de las bases de datos relacionales.

FUNDAMENTACIÓN TEÓRICA

Iniciando con el estudio de la primera opción planteada que es la metodología ORM podemos identificar los siguientes conceptos fundamentales que nos servirán como guía durante todo el proceso de la investigación.

¿QUE ES UN ORM?

Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, por sobre la base de datos relacional. Esto posihibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM. **(Fuente Wikipedia)**

Las bases de datos relacionales normalmente sólo pueden guardar datos primitivos y no objetos de una aplicación por lo tanto lo que se logra con los ORM es convertir los datos de un objeto en datos primitivos y de la misma forma si la aplicación necesita el objeto se tiene que pasar del datos primitivo al objeto correspondiente para que este pueda ser identificado.

Este es el objetivo principal del ORM (mapeo objeto- relacional) ayudar a olvidarnos completamente de utilizar en nuestra programación la conversión de los objetos primitivos para almacenarlos y viceversa.

Entre las ventajas y desventajas claramente mencionadas en el artículo publicado por **Angel Carrero (Programación en Castellano)** tenemos:

VENTAJAS

- ✓ **Rapidez en el desarrollo.** La mayoría de las herramientas actuales permiten la creación del modelo por medio del esquema de la base de datos, leyendo el esquema, nos crea el modelo adecuado.

- ✓ **Abstracción de la base de datos**. Al utilizar un sistema ORM, lo que conseguidos es separarnos totalmente del sistema de Base de datos que utilicemos, y así si en un futuro debemos de cambiar de motor de bases de datos, tendremos la seguridad de que este cambio no nos afectará a nuestro sistema, siendo el cambio mas sencillo.

- ✓ **Reutilización.** Nos permite utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.

- ✓ **Seguridad.** Los ORM suelen implementar sistemas para evitar tipos de ataques como pueden ser los SQL injections.
- ✓ **Mantenimiento del código**. Nos facilita el mantenimiento del código debido a la correcta ordenación de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.

DESVENTAJAS

- ✓ **Tiempo utilizado en el aprendizaje.** Este tipo de herramientas suelen ser complejas por lo que su correcta utilización lleva un tiempo que hay que emplear en ver el funcionamiento correcto y ver todo el partido que se le puede sacar.

- ✓ **Aplicaciones algo más lentas.** Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá de transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos.

Angel Carrero (Programación en Castellano) 2010

A continuación un listado de los Framework de persistencia que existentes y que son de código abierto.

- NEXT
- Enterprise Objects Frameworks
- OpenStep
- WebObjects
- Java Data Objects(JDO)
- Enterprise Java Beans
- Hibernate
- NHibernate
- iPersist
- Linq
- EntilyFramework
- CastleActiveRecord
- Oracle TopLink

MAPEO OBJETO RELACIONAL

Que pasa si queremos almacenar objetos creados por un programa orientado a objetos en una base de datos Relacional?.

El Mapeo Objeto Relacional:

- Nos permite mapear los objetos a registros en las tablas de bases de datos

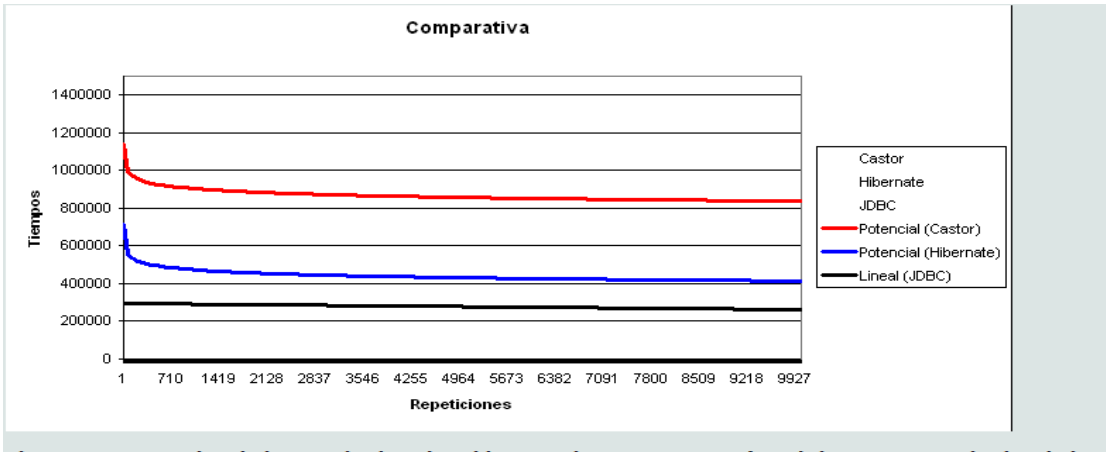
- Esto se realiza mediante una hoja de mapeo en la que se convierte cada clase en una tabla, cada objeto en u registro (fila) de la base de datos y cada atributo en una columna de la misma.

- Tenemos que manejar las relaciones entre clases (tablas) mediante las típicas claves principales primarias y o externas.

Para las pruebas que se realizarán en base a esta metodología se tomará como referencia el uso del Motor de Persistencia llamado Hibernet ya que este es considerado el motor de persistencia con mejor rendimiento en comparación con los demás existentes, esto de acuerdo al brechmark publicado por **Blanca Cubas Marzo 2009** en donde a través de pruebas de se demuestra el buen rendimiento del mismo.

Gráfico No. 1

Comparativa de los resultados obtenidos Hibernate Vs Castor



Fuente: Blanca Cubas Marzo 2009

Que es Hibernet?

Es una herramienta Objeto Relacional creada para plataformas Java que facilita el mapeo entre el modelo orientado a objetos de una aplicación y una base de datos relacional.

Características

- ✓ Buena documentación, comunidad amplia y activa
- ✓ Manejo de transacciones, Cache, polimorfismo, herencia, persistencia
- ✓ Potente lenguaje de consulta HQL
- ✓ Cada clase a persistir necesita un archivo de mapeo XML

A continuación examinaremos mediante conceptos fundamentales la segunda metodología que se basa en el uso de bases de datos Orientadas a Objetos como una

opción para eliminar la diferencia de impedancia de objetos en las aplicaciones, al mismo tiempo se revisará las variables que la componen.

¿QUÉ ES ORIENTACIÓN A OBJETOS?

El diseño de Orientación a objetos es un paradigma de programación en donde lo que más interesa es el comportamiento de cada uno de los objetos y cuyo propósito es dar una solución identificando las partes más relevantes de un problema, dentro del software OO, un objeto es cualquier cosa, real o abstracta, acerca de la cual podemos almacenar los datos y métodos que controlan a los mismos. 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" (Dan Ingalls de Smalltalk 1985).

BASE DE DATOS ORIENTADAS A OBJETOS

Las BDOO comienzan a aparecer a finales de los 80`s convirtiéndose en una verdadera sorpresa debido a que para este entonces las bases de datos relacionales estaban abarcando un gran número en el mercado, las BDOO están estructuradas para simplificar la programación OO, estas permiten almacenar los objetos

directamente en la Base de Datos, utilizando las mismas estructuras y relaciones que los lenguajes de POO.

Así como lo es mencionado en **ND 2009-02 – Año 200 "**Las principales características que las BDOO presentan son:

- Permiten trabajar de una manera transparente y eficiente en un entorno de programación basado en objetos, soportan todos los conceptos de la OO.

- Gestión de datos complejos como lo son por ejemplo los datos multimedia (imagen, video, entre otros).

- Permiten la persistencia transparente de los objetos.

- Presentan en muchos casos una arquitectura distribuida, Procesamiento transaccional que soporta la concurrencia.

Adicionalmente, en general soportan las siguientes características (aunque depende de cada Sistemas Manejadores de Base de Datos Orientados a Objetos (SMBDOO)) la Integridad de datos, versionamiento de objetos, indexación, seguridad, y tolerancia a fallos, entre otras."

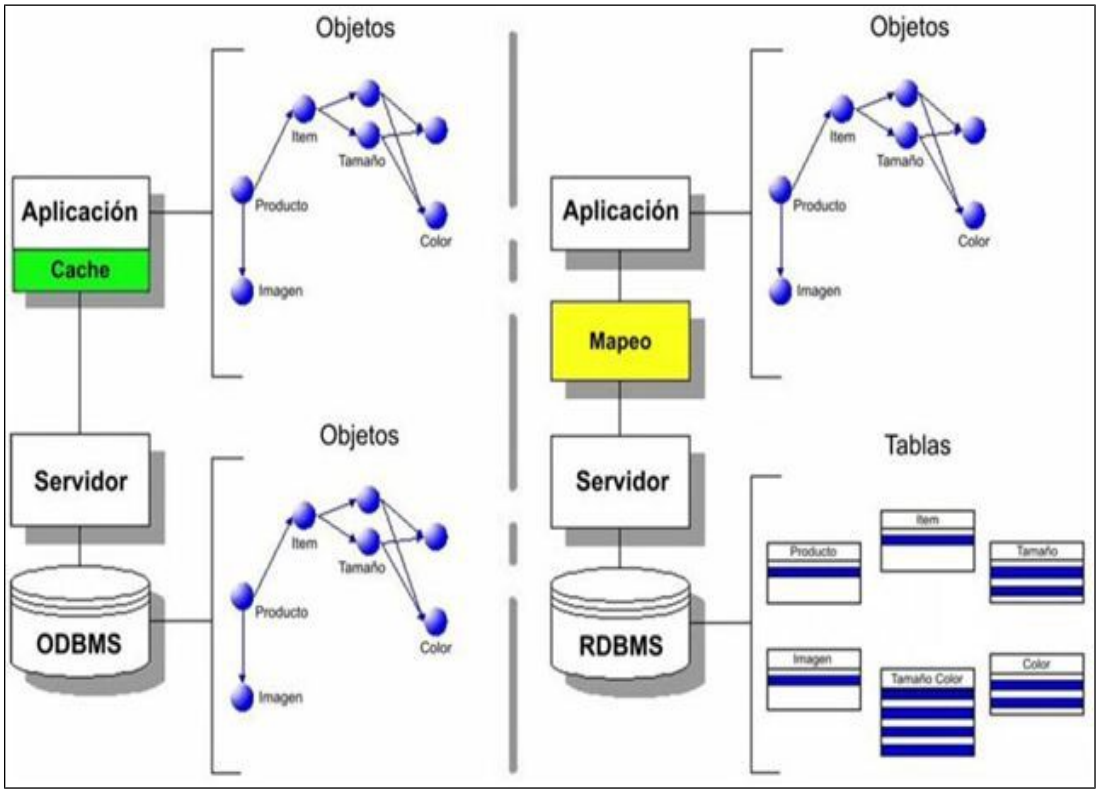
Los ODBMS además de todas estas características proporcionan los costes de desarrollo más bajos y el mejor rendimiento cuando se usan objetos gracias a que almacenan objetos en disco y tienen una integración transparente con el programa escrito en un lenguaje de programación orientado a objetos, al almacenar exactamente el modelo de objeto usado a nivel aplicativo, lo que reduce los costes de desarrollo y mantenimiento.

DIFERENCIAS ENTRE LOS SMBDR, SMBDOO y SMBDRO

También es de gran importancia mencionar los SMBDRO (Sistemas Manejadores de Bases de Datos Relacional Objeto) que equivale al modelo relacional pero en combinación con la orientación a objetos

Gráfico No 2

Diferencia Estructural entre SMBDOO Y SMBDR



Fuente de la imagen: German Viscuso.

Cuadro No 1

Comparación entre los SMBDR SMBDRO y SMBDOO

Sistemas Manejadores de Base de Datos Orientados a Objetos

Sistemas Manejadores de Bases de Datos Relacional Objeto

SMBDR	SMBDOO	SMBDOR
Utiliza la clave Primaria	Utiliza el OID	Utiliza un alisa para cada tabla el cual debe ser único
No se pueden crear nuevos tipos de Datos	Permite la creación de Nuevos Tipos de Datos	Permite la creación de Nuevos Tipos de Datos
Utiliza Tablas	Utiliza objetos	Igual a SMBDR
Utiliza solo tecnología de BDR	Utiliza las Técnicas de POO	Combina la Programación con la Tecnología de las BDR

En consultas utiliza las sentecnias SQL:Select, From ,Where	En consultas utiliza las sentencias SQL, pero el lenguaje es llamado OQL: Select, From, Where	Introduce un API Separado (basado en SQL) para manipular los datos almacenados, las definiciones de clase se de ben mapear a los tipos de datos soportados por el sistema. En desarrollo SQL3
No puede manipular datos complejos	Permite la manipulación de datos complejos	Igual a SMBDOO
Utiliza un Gestor de memoria interna.	La asignación de las posiciones de memoria de los objetos se realiza basada en las relaciones entre los objetos.	Igual a SMBDR

Fuente: ND 2009-02 -2009 Fundamentación Teórica de las Bases de Datos Orientadas a Objetos. Yosly Hernández, Antonio Silva (Octubre, 2009)

Un SMBDOO como bien lo hemos logrado observar es la combinación de las características o capacidades de una base de datos con la tecnología de orientación a objetos, para lo cual existe un tratamiento directo de los objetos sin ningún tipo de traducción para que de esta manera puedan ser gestionados.

Centraremos nuestra investigación en una de las Bases de Datos Orientadas a Objetos que actualmente esta siendo una de las soluciones más atractivas del momento.

DB4Objects

El surgimiento de db4o ha constituido una de las soluciones más atractivas del momento. Db4o se creó como un “Framework” de persistencia de objetos, sus siglas provienen de la expresión "DataBase 4 (for) Objects" que significa BD para objetos, y poco a poco ha ido siendo desarrollado y mejorado al punto de convertirse en la actualidad en un auténtico SGBDOO.

Los principales componentes son: El motor db4 (archivo .jar), el API, **ObjectContainer y Object Manager.**

LICENCIAMIENTO

La Licencias con la que se presenta ante el mercado son:
La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés **GNU GPL**

ANTECEDENTES

Aunque esta base de datos constituye una de las mejores opciones ante el problema de “impedancia de objetos”, db4o debe afrontar la gran extensión y aceptación que tienen los SGBDR en el mercado, por lo que muchas empresas prefieren continuar trabajando con estos gestores en vez de realizar un cambio que puede llegar a ser demasiado tedioso y complicado. De aquí surge la idea del proyecto de estudio, pues db4o aún necesita ayuda para, por un lado, mejorar su aceptación por las empresas.

Esta es una base de datos que fue creada inicialmente para dispositivos móviles pero con el pasar del tiempo a comenzado a ser usada para bases de datos Web.

A Continuación se presenta un listado publicado por la página oficial de DB4O acerca de las empresas que utilizan en la actualidad esta Base de datos y su experiencia:

"INDRA, Sistema de Control de Trenes de Alta Velocidad de Misión Crítica - El Framework del Sistema de Control en Tiempo Real esta compuesto por más de 30.000 objetos en memoria y 30 clases, con 80 Terabytes de información fluyendo eventualmente a una base de datos relacional Oracle en el nivel corporativo. La velocidad de db4o permite al sistema procesar más de 200.000 objetos por segundo. Los beneficios de db4o van más allá de la velocidad; db4o está optimizado para correr en un espacio ínfimo y requiere administración cero. Adicionalmente db4o es nativo para ambos Framework de programación Java y .NET, permitiendo a los desarrolladores almacenar objetos directamente." 2000-2011 VERSANT CORP

"**BOSCH Sigpack:** db4o Controles Complejo, de Alta Velocidad de embalaje Robots. Principales criterios de selección para BOSCH Sigpack Systems AG fueron:

- Db4o de alto rendimiento, lo que permite la gestión de un gran número de objetos.
- Db4o la fiabilidad demostrada
- Acelerado tiempo de salida al mercado a través de db4o como la facilidad de uso y sencilla aplicación

Más allá de ofrecer la velocidad y la fiabilidad, db4o también está optimizado para funcionar en lo mínimo y requiere cero administración. Además, db4o es nativo de Java y ambas. NET marcos de programación, lo que permite a los desarrolladores almacenar objetos directamente." 2000-2011 VERSANT CORP

Clarity Medical, en Pleasanton, California, se basa en db4o para mejorar la salud de los ojos de los bebés con su equipo "Retcam II Wide-Field Pediatric" desprendimiento de la retina de imágenes de dispositivos médicos.

"Debido a que los datos se registran y almacenan precisamente hemos sido capaces de encontrar tumores que se han perdido con el método tradicional" **Clarity Medical**

Como lo hemos podido observar DB4O esta siendo utilizando actualmente en muchos sistemas orientados a diferentes funcionalidades teniendo resultados positivos. A continuación se detallará más sobre las ventajas y desventajas de este SMBDOO para lograr complementar las bases de nuestra investigación.

VENTAJAS

Db4o posee las siguientes ventajas:

- ✓ Alto rendimiento (sobre todo en modo embebido).}
- ✓ Soporta Aplicaciones Standalone así como Cliente/Servidor (Aplicaciones Distribuidas).
- ✓ Mínimo consumo de Recursos
- ✓ Capacidad para replicar sus datos en bases de datos Oracle, útil al implementar aplicaciones de Data Warehouse.
- ✓ Abandono completo del paradigma relacional de las bases de datos tradicionales.
- ✓ Licencia dual GPL/comercial.
- ✓ Soporte Al cambio de Versiones
- ✓ Portabilidad - Compatibilidad con plataformas Linux y Windows.
- ✓ Con DB4O se elimina el proceso de diseño, implementación y mantenimiento de la base de datos ya que el modelo de clases son el esquema de Base de Datos.
- ✓ Requiere cero Administración
- ✓ Fácil Implementación.
- ✓ Nativo para Framework de programación Java y .NET

- ✓ En cuanto al control de integridad de los datos de los objetos, sólo están disponibles las herramientas para realizarlo, tanto a nivel de lógica de negocios (Codificación en el lenguaje OO), como en db4o ("Callbacks", que son métodos que se ejecutan de manera similar a los "Triggers" en bases de datos relacionales, aunque deben ser declarados en la definición de cada clase).

DESVENTAJAS

- ✓ db4o no maneja integridad de datos.

ARQUITECTURA

Este posee dos métodos de Trabajo que son;

- ❖ Embebido y Cliente/Servidor

Cliente Servidor

Del lado del servidor es necesario tener corriendo una aplicación Java o .NET (según la versión de db4o) y en ella configurar un servidor db4o de la siguinete manera:

ObjectServer server = Db4o.openServer(YAPFILENAME, PORT);
server.grantAccess(USER,PASSWORD);

Del lado del cliente se necesita conectarse al servidor y así obtener un

ObjectContainer. Una vez obtenido este se trabaja sobre él de la misma forma que como si fuera un ObjectContainer obtenido de forma local:

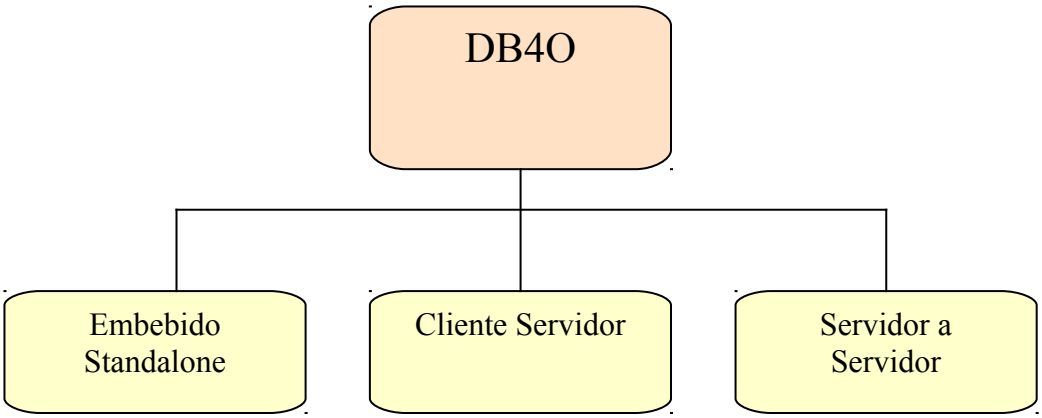
Conexión remota al servidor (uno o más clientes)

ObjectContainer client = Db4o.openClient(host, PORT, USER, PASSWORD);

client.close();

Cuadro No 2

Modos de Trabajo en DB4O



Elaborado: Estefanía Barahona

MOTOR

El motor de base de datos consiste en un archivo .jar menor a 2MB.

INSTALACION

La instalación consiste en agregar el motor de base de datos db4o.jar al CLASSPATH.

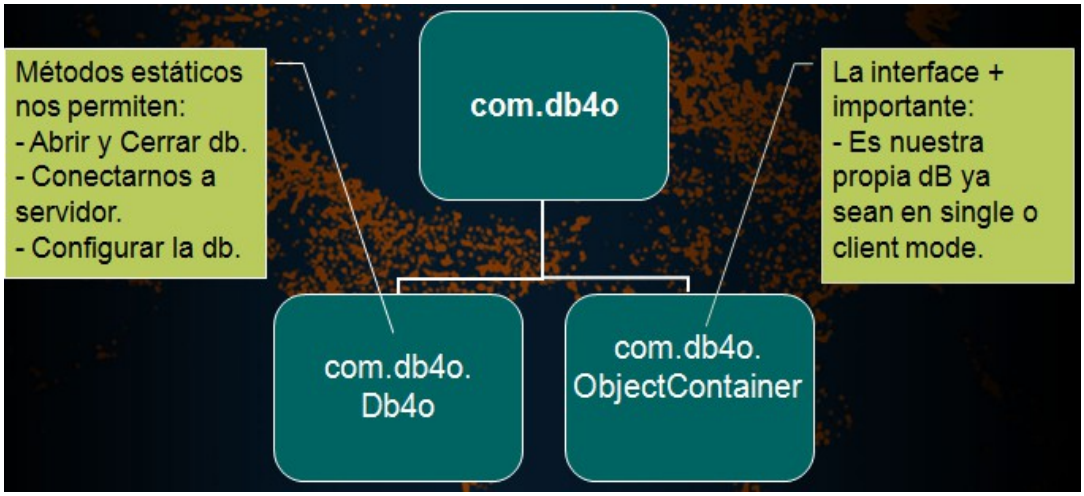
La API

Los paquetes que son necesarios para el correcto funcionamiento son los siguientes:

com.db4o y com .db4o.query

Grafico No 3

Esquema de API en DB4O



Fuente: E.T.S de Ingenieros de Telecomunicación 2005

Todas las transacciones son manejadas por objetos ObjectContainer.

LENGUAJES SOPORTADOS POR DB4Objects

La siguiente tabla presenta los lenguajes y entornos soportados por Db4o:

Lenguaje/Entorno
C#
Java (JRE)
Java (J#)
VisualBasic.Net

Delphi.NET
C++
Ruby (JRuby)
Python (IronPython)
Python (Jython)
Boo
ASP.NET
Compact Framework

El API de Db4o incluye la funcionalidad necesaria para que Db4o pueda ejecutarse como un servidor y permitir la definición de clientes que interactúen con el servidor, se puede implementar esta modalidad para aceptar conexiones desde otros computadores o desde PDAs, dispositivos o teléfonos celulares.

MANEJO DE MEMORIA

La memoria es manejada por la máquina virtual de Java así como lo describen **Yosly Hernández, Antonio Silva (Octubre, 2009)** en el siguiente texto.

"La responsabilidad de limpiar la memoria es cedida a la máquina virtual. En Java se utilizan referencias débiles para las instancias de objetos. Si un objeto no es referenciado por la aplicación durante un largo tiempo, el Garbage Colletor de la Máquina Virtual limpiará éstas referencias débiles. Con respecto a los objetos persistentes que no han sido referenciados, el Garbage Collection no tiene influencia sobre ellos, éstos continuarán siendo almacenados en la BD, incluso si no son referenciados por un largo tiempo". (P. 33)

MANEJO DE ÍNDICES

Los objetos correspondientes al API permiten índices asociados a los atributos de determinadas clases logrando de esta manera mejorar o maximizar el rendimiento de las consultas que se ejecuten en DB4O

A continuación el siguiente ejemplo:

Para indexar el atributo name de los objetos de la clase Pilot se hace lo siguiente:

Db4o.configure().objectClass(Pilot.class).objectField("name").indexed(true);

Db4O permite definir índices de clase, de atributos y de colecciones.

DICCIONARIO

“Db4o no posee un diccionario como tal, para mantener la información de las clases que conforman el esquema de la BD utiliza una clase especial llamada metadata, la cual contiene de cada una de las clases, el nombre, atributos, métodos, antecesores, y las relaciones” Hernández, Antonio Silva (Octubre, 2009)

SEGURIDAD

Db4o maneja la seguridad mediante protección por password y mecanismos de encriptación, como soluciones para una alta seguridad, permitiendo a cualquier

usuario escoger su propio mecanismo de encriptación. Es importante destacar que no proveen seguridad en la comunicación entre cliente y servidor.

Para respaldar la Base de Datos sólo es necesario copiar el archivo .yap para respaldarlo, existen también en la API métodos como los siguientes:

db.ext().backup("c:\xyzDB\xDB.yap")

server.ext().backup("c:\xyzDB\xDB.yap")

Otra medida de seguridad adicional es la encriptación de la base de datos utilizando el método de encriptación simple:

Db4o.configure().encrypt(true);

Db4o.configure().password("yourEncryptionPasswordHere");

Utilizando XTEA (eXtended Tiny Encryption Algorithm):

Db4o.configure ().io(new XTeaEncryptionFileAdapter(password));

INTEGRIDAD

Db4O garantiza integridad implícita, al manejar las identidades de los objetos, para lo que se debe mantener el ObjectContainer en ejecución para garantizarla.

Si se cerrara este y los objetos de manipulan en memoria, se corre el riesgo de duplicidades al guardar objetos en Db4o

Al indagar más sobre db4o y sus prestaciones que lo convierten en un SGBDOO, se encuentra el importante tema del manejo de los datos. Una característica típica es la "integridad referencial"; esto es manejado transparentemente por db4o por lo que su comparación con los SGBDR no tiene mucho sentido, ambos funcionan. Sin embargo al hablar del manejo de los datos se debe pensar también en la fiabilidad de los datos almacenados, más conocido como "integridad de datos", expuesto en [Evaltech2000].

Más que importancia para los desarrolladores, esto es vital para los usuarios y clientes (Empresas), por lo que se constituye en un punto importante de calidad del software. El control que se realiza para asegurar la calidad de los datos no es realizado de forma nativa en los SGBDR, pues utilizan los conocidos Triggers o Desencadenadores para realizar las comprobaciones correspondientes a las reglas de negocio (dominio de la aplicación), que se permiten definir en las utilidades de diseño de la base de datos. Igualmente db4o no maneja esto de forma nativa, y que en el caso de los objetos se podría Denominar "integridad de objetos".

Lo que sí provee db4o son los denominados "Callbacks", que son métodos que se ejecutan ante eventos específicos, al igual que los Triggers. Esto permite que se implemente una característica deseable para este SGBDOO; el control de integridad de datos (Objetos en este caso).

MECANISMOS DE RESPALDO Y RECUPERACIÓN

El Object Container de Db4o usa una memoria caché de objetos donde se guardan los objetos y las transacciones que se están usando y todas las operaciones son realizadas allí en la caché y sólo son escritos a la BD en el momento en que se cierra la conexión. Si ocurre un error de sistema no se ve comprometida la BD sino sólo los objetos que se encuentran en la memoria caché.

Los métodos para realizar transacciones y que son proporcionados por DB4O son:

- ✓ Commit (Finaliza la transacción)
- ✓ Rollback (Deshace la transacción)

Db4o también provee como mecanismo de recuperación la opción de crear respaldos (backups) de la BD está activa y existen aplicaciones corriendo sobre ella.

FUNDAMENTACIÓN LEGAL

Según las leyes de la constitución del Ecuador tenemos los siguientes:

Art. 349.- El sistema de educación superior tiene como finalidad la formación académica y **profesional** con visión científica y humanista; la investigación científica y tecnológica; la innovación, promoción, desarrollo y difusión de los saberes y las culturas; la construcción de soluciones para los problemas del país, en relación con los objetivos del régimen de desarrollo.

Art. 350.- El sistema de educación superior está articulado al sistema nacional de educación y al plan nacional de desarrollo y se regirá por los principios de autonomía responsable, cogobierno, igualdad de oportunidades, calidad, pertinencia, integralidad, autodeterminación para la producción de pensamiento y conocimiento, en el marco del diálogo de saberes, pensamiento universal y producción científica tecnológica global.

DECRETO PRESIDENCIAL NO. 1014 DICTADO POR Econ. RAFAEL CORREA EL 10 DE ABRIL DEL 2008

Artículo 1.- Establecer como política pública para las Entidades de la Administración Pública Central la utilización de Software Libre en sus sistemas y equipamientos informáticos.

Artículo 2.- Se entiende por Software Libre, a los programas de computación que se pueden utilizar y distribuir sin restricción alguna, que permita su acceso a los códigos fuentes y que sus aplicaciones pueden ser mejoradas. Estos programas de computación tienen las siguientes libertadas:

a) Utilización del programa con cualquier propósito de uso común

b) Distribución de copias sin restricción alguna.

c) Estudio y modificación del programa (Requisito: código fuente disponible)

d) Publicación del programa mejorado (Requisito: código fuente disponible).

Artículo 3.- Las entidades de la Administración Pública Central previa a la instalación del software libre en sus equipos, deberán verificar la existencia de la capacidad técnica que brinde el soporte necesario para el uso de este tipo de software.

Artículo 4.- Se faculta la utilización de software propietario (no libre) únicamente cuando no exista una solución de Software Libre que supla las necesidades requeridas, o cuando esté en riesgo la seguridad nacional, o cuando el proyecto informático se encuentre en un punto de no retomo.

Para efectos de este decreto se comprende como seguridad nacional, las garantías para la supervivencia de la colectividad y la defensa del patrimonio nacional.

Para efectos de este decreto se entiende por un punto de no retorno, cuando el sistema o proyecto informático se encuentre en cualquier de estas condiciones:

a) Sistema en producción funcionando satisfactoriamente y que un análisis de costo beneficio muestre que no es razonable ni conveniente una migración a Software Libre.

b) Proyecto en estado de desarrollo y que un análisis de costo – beneficio muestre que no es conveniente modificar el proyecto y utilizar Software Libre.

Periódicamente se evaluarán los sistemas informáticos que utilizan software propietario con la finalidad de migrarlos a Software Libre.

Artículo 5.- Tanto para software libre como software propietario, siempre y cuando se satisfagan los requerimientos, se debe preferir las soluciones en este orden:

a) Nacionales que permitan autonomía y soberanía tecnológica.

b) Regionales con componente nacional.

c) Regionales con proveedores nacionales.

d) Internacionales con componente nacional.

e) Internacionales con proveedores nacionales.

f) Internacionales

HIPÓTESIS DE LA INVESTIGACIÓN DOCUMENTAL

PREGUNTAS A CONTESTARSE

Con esta investigación se podrá determinar cual es la mejor metodología que nos puede ayudar con la impedancia de objetos y al mismo tiempo saber si la metodología brindada por las bases de datos orientadas nos ayudaría o no a romper el paradigma de las bases de datos Relacionales mediante el uso de la metodología ORM. Respondiendo las siguientes preguntas:

- **¿Es hora de Usar Bases de Datos Orientadas a Objetos para solucionar los problemas de impedancia?**
- **¿Cómo se diseña con las bases de Datos Orientadas a Objetos?**
- **¿Cómo se dimensiona en comparación con las bases de datos relacionales?**

Todas estas preguntas y muchas otras serán respondidas con la evaluación que se realizará, la misma que permitirá profundizar en el estudio de las mismas y poder determinar bajo que parámetros de utilización se la puede tomar en cuenta.

VARIABLES DE LA INVESTIGACIÓN

- Una de las variables a estudiar mediante esta investigación son las bases de datos Orientadas a Objetos que serán representadas por la base de datos llamada DB4O la cual es un novedoso motor de base de datos orientada a objetos, las mismas que generan las siguientes consecuencias directas resultantes de este nuevo paradigma:

- ❖ Deja de existir un lenguaje SQL de consultas/modificaciones para pasar a crearse sistemas de consulta por métodos delegados y actualización/creación/borrado automático de entidades mediante código compilable.

- ❖ Se elimina la necesidad de representar el modelo de datos de la aplicación en dos tipos de esquemas: modelo de objetos y modelo relacional. Ahora el esquema de datos del dominio viene representado por la implementación que se realice del diagrama de clases.

- ❖ Se consigue evitar el problema del Object-Relational Impedance Mismatch sin sacrificar el rendimiento que los mapeadores objeto-relacionales sufren actualmente para llevar a cabo el mismo objetivo.

- Una segunda Variable a analizar es el uso de la metodología ORM (Object/Relational Mapping), la cual es una técnica de programación que permite vincular los objetos usados en nuestro modelo de la aplicación con una base de datos relacional.

  El principal problema en la actualidad surge porque prácticamente todas las aplicaciones están diseñadas para usar la Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional. Es aquí cuando esta metodología cobra importancia ya que permite traducir de forma automática los

objetos en registros y viceversa permitiendo una mejor comunicación entre ambos elementos.

Esta variable estará representadaza por el uso del motor de persistencia llamado Hibernate y la base de datos relacional llamada Oracle 11g.

DEFINICIONES CONCEPTUALES

A continuación se describen los conceptos básicos importantes que se manejarán durante el estudio.

Base de Objetos Embebida (o empotrada): una base de datos embebida es una base que es invisible para el usuario final.

GNU GPL: General Public License o licencia pública general, es una licencia creada por la Free Software Fundation a mediados de los 80, y esta orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre.

Persistencia.- es el almacenamiento de los datos en memoria para una posterior recuperación de los mismos.

Clases: abstracción conceptual que permite describir un conjunto de objetos que tienen el mismo tipo **(Marín, 2001).** Una clase no es más que un patrón en el que se basan aquellos objetos que tienen propiedades similares. Por ejemplo, la clase Persona donde los atributos que lo componen son nombre, apellido y edad.

Instanciación: mecanismo que permite crear objetos de una clase determinada. Por ejemplo, si se tiene la clase Persona a partir de este mecanismo se crea el objeto Juan, Marta, entre otros.

Objetos: corresponden a todos los elementos que se manipulan dentro de una BDOO. Un Objeto es una representación abstracta del mundo real, el cual está compuesto por un estado (las propiedades y sus respectivos valores), y un comportamiento (las operaciones asociadas que permiten interactuar con otros objetos y consigo mismo). **"Un Objeto no es más que Fundamentación Teórica de las Bases de Datos Orientadas a Objetos". Yosly Hernández, Antonio Silva (Octubre, 2009)**

Variable de Instancia: según **(Marín, 2001)** son cada uno de los atributos que caracterizan el estado de un objeto. Por ejemplo, si se tiene el atributo edad, la variable de instancia será el valor que tenga asignado.

Identidad de un Objeto: se implementa a través de un identificador único OID (Object Identifier), generado por el sistema. El valor de un OID no es visible para el usuario externo, el sistema lo emplea internamente para identificar cada objeto de manera única, así como también, crear y manejar referencias entre objetos. El concepto de identidad hace que sea necesario distinguir:

• Igualdad de identidad: dos objetos son iguales si tienen el mismo OID. En este caso se suele decir que son el mismo objeto.

Normalmente se representa por el símbolo '='.

• Igualdad de valor: dos objetos son iguales si los valores de sus atributos son iguales. Normalmente se representa por el símbolo

'=='.

Método: es un procedimiento algorítmico a través del cual se realiza una determinada operación sobre el comportamiento de un objeto, un método se caracteriza por tener su nombre, sus parámetros formales y su valor de retorno (si es el caso).Los métodos de los objetos pueden ser de tres tipos: observadores, que devuelven información acerca del estado de un objeto modificadores, que cambian un objeto de un estado válido a otro; y constructores, que permiten la creación de objetos**. (Marín, 2001).**

Herencia: mecanismo mediante el cual una clase puede ser definida sobre la base de la definición de de otra clase. Por este mecanismo, la subclase hereda los atributos que definen la estructura de la superclase y los métodos que caracterizan su comportamiento. Además, la subclase puede añadir nuevos atributos y métodos para completar su definición.

Los tipos de herencia son: simple (una clase hereda de una única superclase) y múltiple (una clase hereda de más de una superclase). **(Bertino & Martino, 1995).**

Polimorfismo: mecanismo que permite definir e invocar funciones que comparten la misma interfaz pero tienen una implementación diferente **(Marín, 2001).** Por ejemplo, si se tiene el método CalcularArea() depende del objeto que lo implemente se ejecutará, porque calcular el área de un cuadrado no es igual a calcular el área de un círculo.

Persistencia: capacidad del programador para que sus datos se conserven al finalizar la ejecución de un proceso, de forma que se puedan reutilizar en otros procesos. Los datos tienen que persistir sin necesidad de que el usuario explícitamente los copie **(Marín, 2001).**

Un Objeto persistente: es un objeto que sobrevive a la ejecución del proceso que lo creó. Mientras que un Objeto transitorio es un objeto que deja de existir cuando el proceso que lo creó termina su ejecución.

Encapsulamiento: ocultar la implementación de un método, dejando visible la especificación. Puede tener niveles de: privada, público y protegida

CAPÍTULO III

METODOLOGÍA

DISEÑO DE LA INVESTIGACIÓN

MODALIDAD DE LA INVESTIGACIÓN

El diseño de la exploración que se ha encontrado como el más adecuado para esta investigación de este proyecto de grado es la investigación de campo, ya que esta nos ayudará a determinar cuando, como y bajo que condiciones podremos obtener los datos.

Mediante esta investigación podremos manipular las variables con el fin de encontrar las causas o razones por las que no se ha logrado determinar aún el mejor método para controlar la impedancia de objetos

La modalidad de la investigación realizada en este proyecto se desglosa en:
30% investigación
70% de campo.

Mediante este diseño de investigación se lograrán obtener los resultados e informe final del proyecto.

Tipo de investigación

Según Vélez S. (2001), "La investigación experimental está orientada a la utilización del conocimiento básico y aplicado, previo control de los resultados

mediante el diseño, construcción y prueba de modelos, prototipos e instalaciones experimentales".

Por esta razón se realizará un tipo de investigación experimental debido a que el proyecto contiene las siguientes características:

- Manipulación de variable (Desempeño, estabilidad, Facilidad de Utilización, Portabilidad, flexibilidad)
- Comparación de resultados entre la metodología ORM y BDOO

POBLACIÓN Y MUESTRA

Población:

La población sobre la cual se desarrolló la investigación equivale a todos los desarrolladores e ingenieros de Sistemas, con experiencia en desarrollo y creación de sistemas computacionales de la ciudad de Guayaquil.

Fue indispensable que en este grupo existan personas con mínimo un año de experiencia laborando con aplicaciones orientadas a objetos. También fueron considerados 5 especialistas en el diseño de la arquitectura de software con más de 2 años de experiencia en el área.

Muestra:

Debido a que la población seleccionada para la investigación sobrepasa las 150 personas aproximadamente se procedió a utilizar la técnica de muestreo.

Para poder determinar un subconjunto de la población se escogió a 50 personas de diferentes empresas desarrolladoras de software que cumplen con lo indispensable para poder realizar la investigación por medio de los cuales se obtendrá la información y datos necesarios.

Cuadro No4
MUESTRA

MUESTRA	**50**
TOTAL	**50**

Entre las personas encuestadas se encuentran 5 especialistas en diseño de software los cuales nos ayudaron con su experiencia profesional formando parte de la encuesta.

OPERACIONALIZACIÓN DE VARIABLES

Las variables principales que formaron parte de la investigación son:

- Desempeño

- Estabilidad
- Tolerancia a Fallos
- Flexibilidad
- Facilidad de Utilización
- Escalabilidad
- Portabilidad
- Integridad
- Seguridad

Algunas de estas variables se lograron medir de forma complementaria con la encuesta realizada tanto para la metodología ORM como BDOO. El detalle respectivo de cómo se trabajo con cada una de ellas se lo detalla en el siguiente cuadro.

CUADRO No.5
MATRIZ DE OPERACIONALIZACIÓN DE VARIABLES

Variables	Dimensiones	Indicadores	Técnicas y/o Instrumentos

Evaluación del desempeño de la metodología ORM tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Tiempo de respuesta	Tiempo en Segundos que demora en responder o en ejecutarse las transacciones realizadas	Encuesta 50% Pruebas experimentales 50%
	Consumo de Recursos	Nivel de Uso de memoria y procesador de la maquina que emplea la metodología	Encuesta 50% Pruebas experimentales 50%
Evaluación de la estabilidad de la metodología ORM tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Conexiones Simultáneas	Hasta cuantas conexiones o sesiones abiertas es soportada por esta metodología	Bibliografía especializada 50% Pruebas experimentales 50%
Evaluación de la tolerancia a fallos de la metodología ORM tomando como base	Fallos o caídas de servidor	Que sucede con las transacciones que se encuentran en procesamiento y las ya realizadas	Bibliografía especializada 50% Pruebas experimentales

Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales		anteriormente cuando se cae el servidor	50%
Evaluación de la flexibilidad de la metodología ORM tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Disponibilidad a Cambios	Que sucede con un cambio de versión de la metodología o con cambios estructurales en la aplicación	Bibliografía especializada 50% Pruebas experimentales 50%
Evaluación de la facilidad de utilización de la metodología ORM tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando	Uso	Nivel de complejidad de la metodología y grado de utilización	Bibliografía especializada 10% Pruebas experimentales 20% Encuesta 70%

sistemas computacionales			
Evaluación de la calidad de la metodología ORM tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Seguridad	Facilidad de obtención de datos	Bibliografía especializada 50% Pruebas experimentales 50%
	Portabilidad	Si se puede usar en ambientes diferentes	Bibliografía especializada 50% Pruebas experimentales 50%
	Integridad de datos	Que tan coherente se mantiene la información	Encuesta 50% Pruebas experimentales 50%
Evaluación del desempeño de la metodología BDOO tomando como base DB4O	Tiempo de respuesta	Tiempo en Segundos que demora en responder o en ejecutarse las	Encuesta 50% Pruebas experimentales 50%

a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales		transacciones realizadas	
Evaluación del desempeño de la metodología DB4O tomando como base Hibernate a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Tiempo de respuesta	Tiempo en Segundos que demora en responder o en ejecutarse las transacciones realizadas	Encuesta 50% Pruebas experimentales 50%
	Consumo de Recursos	Nivel de Uso de memoria y procesador de la maquina que emplea la metodología	Encuesta 50% Pruebas experimentales 50%
Evaluación de la estabilidad de la metodología BDOO tomando como base DB4O a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y	Conexiones Simultáneas	Hasta cuantas conexiones o sesiones abiertas es soportada por esta metodología	Bibliografía especializada 50% Pruebas experimentales 50%

diseñando sistemas computacionales			
Evaluación de la tolerancia a fallos de la metodología BDOO usando como base DB4O a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Fallos o caídas de servidor	Que sucede con las transacciones que se encuentran en procesamiento y las ya realizadas anteriormente cuando se cae el servidor	Bibliografía especializada 50% Pruebas experimentales 50%
Evaluación de la flexibilidad de la metodología BDOO tomando como base DB4O a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Disponibilidad a Cambios	Que sucede con un cambio de versión de la metodología o con cambios estructurales en la aplicación	Bibliografía especializada 50% Pruebas experimentales 50%

Evaluación de la calidad de la metodología DBOO tomando como base DB4O a desarrolladores e ingenieros de Sistemas que tienen experiencia desarrollando y diseñando sistemas computacionales	Seguridad	Facilidad de obtención de datos	Bibliografía especializada 50% Pruebas experimentales 50%
	Portabilidad	Si se puede usar en ambientes diferentes	Bibliografía especializada 50% Pruebas experimentales 50%
	Integridad de datos	Que tan coherente se mantiene la información	Encuesta 50% Pruebas experimentales 50%

Elaboración: Estefanía Barahona León

INSTRUMENTOS DE RECOLECCIÓN DE DATOS

La técnica de campo que se utilizará para la recolección de datos necesarios para la investigación serán:

Documentales	**De Campo**
- Lectura Científica	- Observación
- Análisis de Contenido	- Encuesta
- De redacción y estilo	

CUADRO No.6
INSTRUMENTOS DE LA INVESTIGACIÓN.

Técnica	Instrumento
Observación	Registro de Observación y de las pruebas realizadas
Encuesta	Cuestionario de preguntas Correo Electrónico
Lectura Científica	Internet
De redacción y estilo	Elaboración de Informes

Elaborado por: Estefanía Barahona

LA ENCUESTA Y EL CUESTIONARIO

FORMATO DE LA ENCUESTA

UNIVERSIDAD DE GUAYAQUIL
ENCUESTA SOBRE LA METODOLOGÍA ORM Y BDOO
AL 28 DE NOV DEL 2010

Por favor dedique unos minutos a completar esta encuesta, la misma que nos ayudará a determinar la mejor manera de resolver la impedancia de objetos.

Marque con una X su respuesta

1.- Ha trabajado alguna vez con motores de persistencia o Framework que ayuden a persistir los objetos dentro de sus aplicaciones orientadas a objetos.

***(En el caso de seleccionar si pase a la siguiente pregunta de lo contrario continúe en la pregunta 5)**

Si ( ) ☐
No ( )
No se ( )

2.- Seleccione el motor de persistencia que usted más ha utilizado?

***(Debe de escoger sólo una opción)**

Hibernet	( )	Cayenne	( )	OJB	( )
Torque	( )	Ibatis	( )	Otros	( )
JDBCPersistence	( )	Castor	( )	Ninguno	( )

3.-Califique la facilidad de Uso del Framework seleccionado

Sencillo ( )

Manejable ( )
Complejo ( )
Demasiado complejo ( )

4.- Que opina del desempeño (Tiempo de respuesta – Consumo de recursos etc) que este ha tenido en sus aplicaciones cuando usted lo ha usado.

A) Bueno ( )
B) Regular ( )
C) Malo ( )
D) Pésimo ( )

5.- Ha trabajado alguna vez con bases de datos Orientadas a Objetos.
*(En el caso de seleccionar si pase a la siguiente pregunta de lo contrario la encuesta ha terminado)

A) Si ( )
B) No ( )
C) No se ( )

6.- Seleccione las Bases de Datos Orientadas a objetos que usted más ha utilizado?
***(Debe de escoger sólo una opción)**

DB4Objects	( )	ODABA	( )	ConceptBase	( )
Objectivity/DB	( )	PostgreSQL	( )	Perst	( )
Versant	( )	ObjectStore	( )	Otros	( )

7.- Califique la facilidad de Uso de la base de datos Orientada a Objetos seleccionada.

Sencillo ( )

Manejable ( )
Complejo ( )
Demasiado complejo ( )

4.- Que opina del desempeño (Tiempo de respuesta – Consumo de recursos etc) que ha tenido esta base de datos frente a sus aplicaciones

Bueno ( )
Regular ( )
Malo ( )
Pésimo ( )

Encuestadora: Estefanía Solange Barahona León

PROCEDIMIENTOS DE LA INVESTIGACIÓN

El problema:

Planteamiento del problema

Interrogantes de la investigación

Objetivos de la Investigación

Justificación o importancia de la investigación

Marco teórico:

Fundamentación teórica

Fundamentación legal

Preguntas a contestarse

Definición de términos

Metodología:

Diseño de Investigación

Población y Muestra

Instrumentos de recolección de datos

Operacionalización de variables, dimensiones e indicadores

Procedimiento de la Investigación

Criterios para la elaboración de la propuesta

PROCESAMIENTO Y ANÁLISIS

Las variables que se lograron complementar con la encuesta son:

- Uso
- Desempeño
- Facilidad de Uso

A continuación se describe los resultados obtenidos a través de las encuestas realizadas.

CUADRO No.7

1.- Ha trabajado alguna vez con motores de persistencia o Framework que ayuden a persistir los objetos dentro de sus aplicaciones orientadas a objetos.

Si	20
No	27
No se	3

Gráfico No.4
Uso de Motores de Persistencia

ANALISIS:

Según los resultados obtenidos de la muestra seleccionada que equivalen a 50 personas entre desarrolladores e Ingenieros, podemos decir que un 54% de los desarrolladores en el Ecuador no conocen sobre frameworks o motores de persistencia, quedando un 40% que si los conoce y un 6% que no saben de que se trata.

Pregunta No 2.- Seleccione el motor de persistencia que usted más ha utilizado

CUADRO No.8
Motores de Persistencia más usados

Hibernate	12
Torque	1
Jdbc	4
Ibatis	1
Odbc	2
Cayenne	0
Castor	0
Otros	0
Ninguno	0

Elaborado Por: Estefanía Barahona

Gráfico No.5
Uso de Motores de Persistencia

Elaborado Por: Estefanía Barahona

ANALISIS:

Con este resultado podemos observar a través de resultados consolidados cuales son los frameworks más usados, estando entre ellos de mayos a menor Hibernate con un 60 %, jdbc con un 20 %, Odbc con un 10%, Torque con un 5% , Ibatis con un 5% , lo que nos deja ver con mayor claridad que la mayoría de los desarrolladores en Ecuador que utilizan frameworks prefieren a Hibernate dentro de sus desarrollos frente a otros motores de persistencia ya existentes.

Pregunta No 3.- Califique la facilidad de Uso del Framework seleccionado

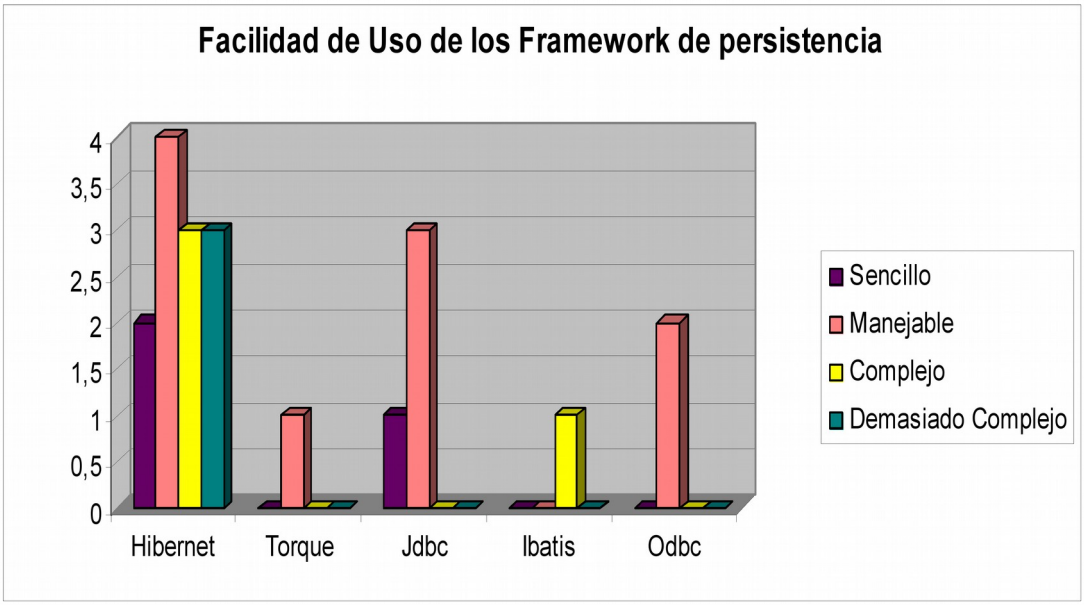
CUADRO No.9

Facilidad de uso de los frameworks de Persistencia

	Sencillo	Manejable	Complejo	Demasiado Complejo
Hibernet	2	4	3	3
Torque	0	1	0	0
Jdbc	1	3	0	0
Ibatis	0	0	1	0
Odbc	0	2	0	0

Elaborado Por: Estefanía Barahona

Gráfico No.6



Elaborado Por: Estefanía Barahona

ANALISIS

Según estos resultados podemos ver el manejo de la variable facilidad de Uso de los frameworks de persistencia ya que podemos observar que la mayoría de estos son manejables sin embargo unos de los frameworks más usados de acuerdo al resultado de la pregunta anterior como es Hibernate tiene a casi un 50% que opinan que es sencillo o manejable y el otro 50% que opinan que es muy complejo.

Pregunta No 4.- Que opina del desempeño (Tiempo de respuesta – Consumo de recursos etc) que este ha tenido en sus aplicaciones cuando usted lo ha usado

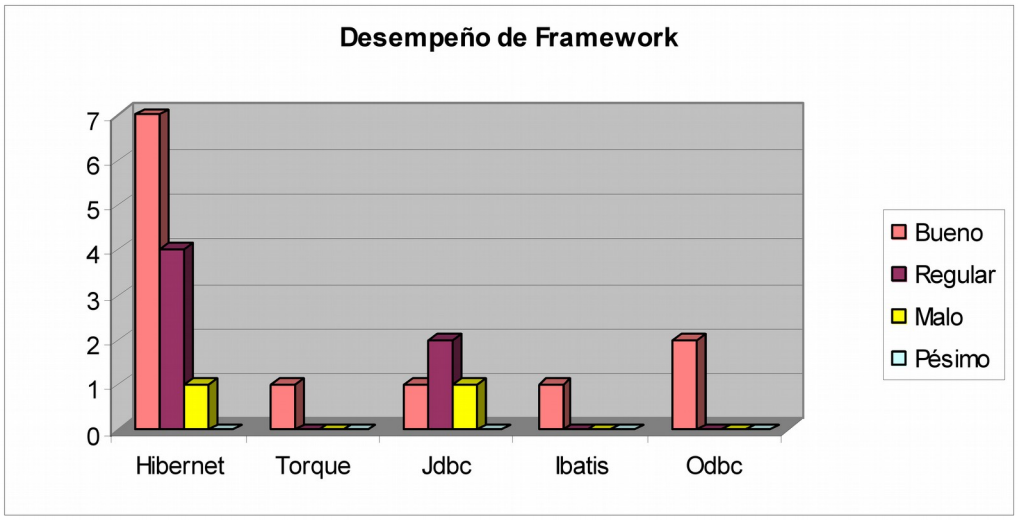
CUADRO No.10

Desempeño de los Motores de Persistencia

	Bueno	Regular	Malo	Pésimo
Hibernet	7	4	1	0
Torque	1	0	0	0
Jdbc	1	2	1	0
Ibatis	1	0	0	0
Odbc	2	0	0	0

Elaborado Por: Estefanía Barahona

Gráfico No.7



Elaborado Por: Estefanía Barahona

ANALISIS.

Según los resultados obtenidos en esta gráfica nos damos cuenta que Hibernate tiene un rendimiento mejor frente a los demás frameworks en cuanto a rendimiento o desempeño.

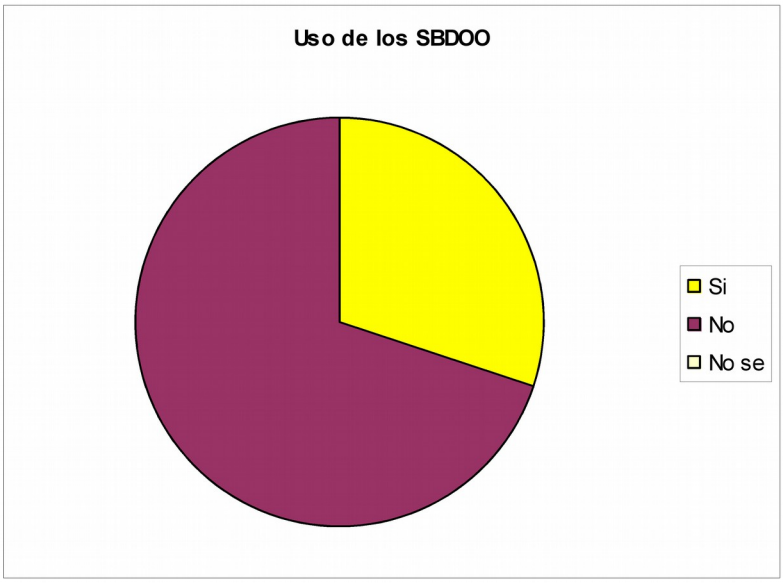
Pregunta No 5. Ha trabajado alguna vez con bases de datos Orientadas a Objetos.

CUADRO No.11
Uso de los SBDOO

Si	15
No	35
No se	0

Elaborado Por: Estefanía Barahona

Gráfico No.8



Elaborado Por: Estefanía Barahona

ANALISIS

Según los resultados obtenidos de la muestra seleccionada que equivalen a 50 personas entre desarrolladores e Ingenieros, podemos decir que un 70% de los desarrolladores de Ecuador en Guayaquil no conocen sobre Bases de datos Orientadas a Objetos, quedando un 30% que si los conocen.

En correlaión con los resultados de la misma variable pero para la metodología de Motores de Persistencia podemos encontrar la siguiente diferencia:

CUADRO No.12
Correlación entre la metodología BDOO y ORM

	Hibernate	BDOO	%Hibernate	%BDOO	Diferencia
Si	20	15	57,14%	42,86%	14,29%
No	27	35	43,55%	56,45%	-12,90%
No se	3	0	100,00%	0,00%	100,00%

Elaborado Por: Estefanía Barahona

La metodología más usada por los desarrolladores de Ecuador para eliminar la impedancia de objetos es la de los Motores de Persistencia o ORM con un 14.29 % de diferencia sobre las BDOO.

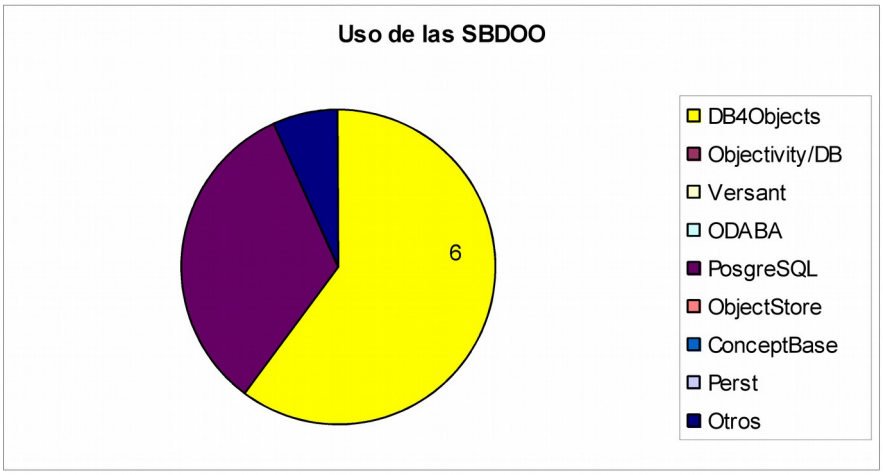
Pregunta No 6.- Seleccione las Bases de Datos Orientadas a objetos que usted más ha utilizado?

CUADRO No.13 BASES DE DATOS ORIENTADAS A OB4JETOS MAS USADAS

DB4Objects	9
Objectivity/DB	0
Versant	0
ODABA	0
PosgreSQL	5
ObjectStore	0
ConceptBase	0
Perst	0
Otros	1

Elaborado Por: Estefanía Barahona

Gráfico No.9



Elaborado Por: Estefanía Barahona

ANALISIS

Con este resultado podemos observar a través de resultados consolidados cuales son las bases de datos Orientadas a objetos más usadas, estando entre ellos de mayor a menor DB4O con un 60 %, PosgreSQL con un 33 % ,y otros con un 7%, lo que nos deja ver con mayor claridad que la mayoría de los desarrolladores en Ecuador qu utilizan BDOO prefieren a DB4O dentro de sus desarrollos frente a otras Basis de Datos Orientadas a Objetos ya existentes.

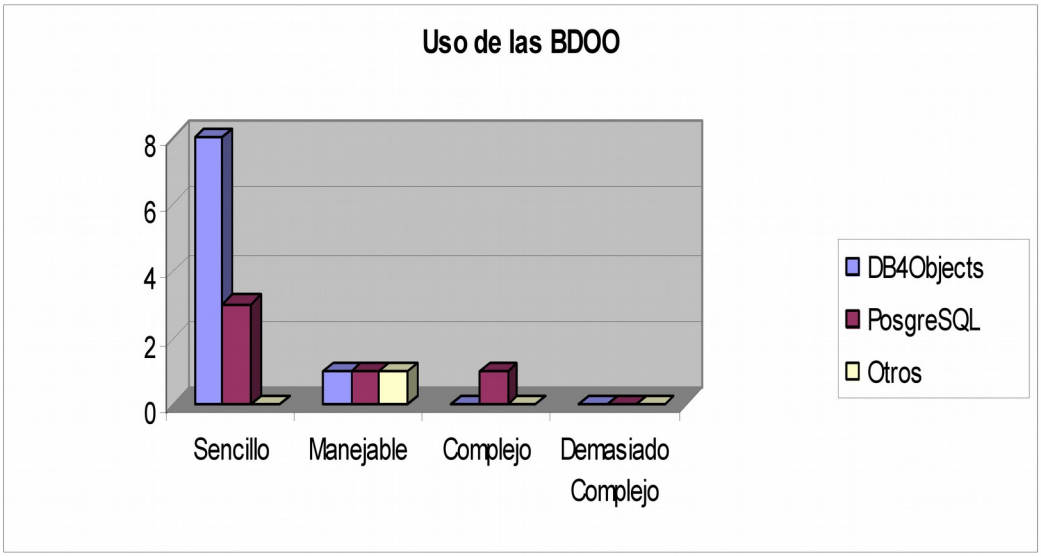
Pregunta No 7.- Califique la facilidad de Uso de la base de datos Orientada a Objetos seleccionada.

CUADRO No.14
Facilidad de Uso de BDOO

	Sencillo	Manejable	Complejo	Demasiado Complejo
DB4Objects	8	1	0	0
PosgreSQL	3	1	1	0
Otros	0	1	0	0

Elaborado Por: Estefanía Barahona

Gráfico No.10



Elaborado Por: Estefanía Barahona

ANÁLISIS

Según estos resultados podemos ver el manejo de la variable facilidad de Uso de las Bases de Datos Orientadas a Objetos ya que podemos observar que la mayoría de estos son sencillas sin embargo unos de las más usadas de acuerdo al resultado de la pregunta anterior como es DB4O tiene a casi un 90% que opinan que es sencillo o manejable y ninguno opina que es complejo.

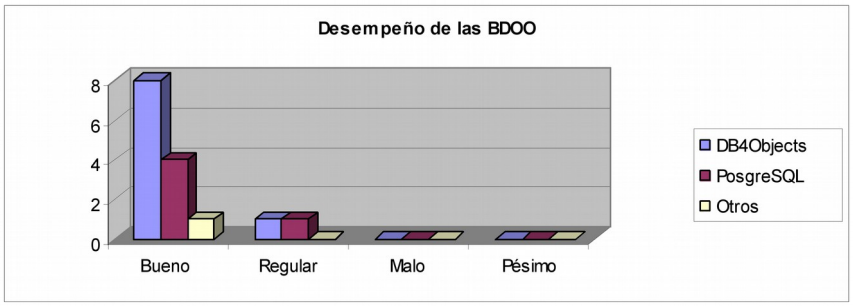
En correlación con el cuadro anterior podemos observar que las personas que usan esta metodología lo hacen debido a la facilidad de uso que esta presta a los que utilizan esta metodología.

Pregunta No 8.- Que opina del desempeño (Tiempo de respuesta – Consumo de recursos etc.) que ha tenido esta base de datos frente a sus aplicaciones.

CUADRO No.15
Desempeño BDOO

	Bueno	Regular	Malo	Pésimo
DB4Objects	8	1	0	0
PosgreSQL	4	1	0	0
Otros	1	0	0	0

Gráfico No.11



Elaborado Por: Estefanía Barahona

ANALISIS

En cuanto a la Variable Desempeño podemos decir que como resultado de esta encuesta se tiene que DB4O tiene entre 90% y 100% de personas que dicen que el desempeño de este es bueno lo que podría ser la justificación necesaria para preferencia frente a otras bases de datos orientadas a objetos.

INVESTIGACION EXPERIMENTAL

Definiendo el proyecto y sus requerimientos

Al momento de ahondar en db4o **(Metodología BDOO)** surge la necesidad de acercarlo a los tradicionales SGBDR para así comparar sus diferencias.

Además de la comparación se necesita medir desempeño, estabilidad, tolerancia a fallos, flexibilidad y características que ofrecen tanto la metodología ORM como el uso de las bases de datos Orientadas a Objetos para de esta manera lograr determinar cual es la metodología que sirve como solución para la impedancia de objetos.

Preparación del entorno de desarrollo

Con los antecedentes expuestos se realizó la tarea de preparar el ambiente de pruebas, es decir, buscar, seleccionar y mantener disponibles las herramientas y utilidades necesarias para complementar el desarrollo del aplicativo de TEST y las pruebas necesarias para proceder con la evaluación de cada una de las variables investigadas.

Recursos

Hardware

El hardware y arquitectura utilizada en las pruebas de evaluación tanto para el modelo Orientado a Objetos como el modelo Relacional es el siguiente:

CUADRO No. 16

Descripción Hardware utilizado.

Equipo	Características	Uso
Workstation 1	Core2 Duo HP Pavillion dv4,4 GB RAM , 300 GB HDD , Vista Home Premium 64 bits	Pruebas Servidor Base de Datos DB4O, Servidor Base de Datos Oracle
Workstation 2	Core2 Duo HP Pavillion dv4,2 GB RAM , 500 GB HDD , Windows 7 32 bits	Pruebas a Conexión Remota y concurrencia a Servidores de Base de Datos
Workstation 3	Máquina Virtual Core2 Duo HP Pavillion dv4,2 GB RAM , 35 GB HDD , Red Hat Enterprise Linux	Pruebas Servidor Base de Datos DB4O (Linux), Servidor Base de Datos Oracle (Linux).

Elaborado por: Estefanía Barahona

Las pruebas realizadas no implicaron mayores requerimientos, en cuanto a hardware, aunque hubiese sido excelente realizar las mismas pruebas en arquitecturas más robustas pero por cuestiones de disponibilidad de las mismas no formaron parte del alcance de este proyecto, sin embargo lo que si formo parte del alcance fue realizarlas como mínimo en dos ambientes diferentes como son:

- Plataforma Linux
- Plataforma Windows

Por lo demás los usuarios de estas metodologías deberán afrontar la elección correcta de un servidor de acuerdo a requerimientos de rendimiento de sus propios sistemas.

Software

La elección del software se basó en el conocimiento sobre herramientas de desarrollo utilizadas con anterioridad, además se prefirió el uso de software libre o de código abierto (Open Source).

Sin embargo fue necesario utilizar una versión de prueba de la herramienta licenciada "Oracle 11g", la cual nos permitió levantar un servidor de Base de Datos Relacional, así como la versión de prueba de VMware Workstation 6.0 para levantar el servidor Linux.

A continuación el detalle del software utilizado.

CUADRO No. 17

Descripción software utilizado

Software	Rol
db4o	DBMSOO (Sistema Manejador de Base de Datos Orientado a Objetos)
Oracle 11g	RDBMS Sistema Manejador de Bases de Datos Relacionales)
Hibernet	Framework para persistir los objetos
Jcreator	Editor de programación
Java	Motor de Programación Orientada a Objeto
Dreamweaver	Herramienta que permite crear los archivos xml del mapeo MOR.
Plsql Deveoper	Herramienta que permite gestionar
VMware Workstation 6.0	Software para la creación de Maquinas Virtuales.

Elaborado por: Estefanía Barahona L.

Modelado inicial

El modelado efectuado en la etapa inicial correspondió al diagrama de clases de la base de datos OO que utilizarán las dos versiones del aplicativo de prueba y al diagrama de entidad relación que utilizaría la base de Datos Relacional.

A Continuación el detalle del modelado.

Diseño de la base de datos OO

El diseño de la base de datos está dado por su diagrama de clases. El modelo diseñado representa una base de datos OO, manteniendo información acerca de sus usuarios, clases, atributos y reglas.

A diferencia del modelado Entidad-Relación de las bases de datos relacionales, el diagrama de una base de datos OO no necesita pasar por los modelos conceptual, lógico y físico, esto representa una ventaja al momento de dar mantenimiento a un módulo determinado, puesto que significa menor esfuerzo. Esto también significa que no se deben realizar operaciones especiales para representar relaciones o recursivas

Lo que sí implica es el desarrollo de lógica destinada a controlar reglas de negocio, pues por ejemplo no se definen claves primarias.

CUADRO No. 18
Clases Usadas del aplicativo de TEST.

Nombre de Clase
Persona
Empleado
Usuario
Organización
Cliente

Elaborado por: Estefanía Barahona L.

Diseño de la aplicación de Prueba utilizando Base de Datos orientadas a Objeto

- El modelo de clases constituye el esquema de las bases de datos lo que se identifica en las bases de datos relacionales como diagrama Entidad/Relación.
- El código que se genera a partir del modelo o esquema de la base de datos se debe mantener dentro del proyecto de la aplicación en Desarrollo.

En Windows y Linux

Para cada base de datos manejada con db4o se creó un servidor que recibe peticiones en un puerto TCP específico, en nuestro caso la clase que hace de servidor de la base de datos OO se llama run_db4o.java, la misma que crea el archivo llamado **db4o.yap** (nuestra base de datos).

Esta parte se denominó “db4o Server” y se concibió como un servicio que realizará sus tareas en “background”.

Diseño de la aplicación de Prueba utilizando Hibernate como motor de Persistencia usando una base de datos Oracle.

- Inicialmente se procedió a crear el diseño de la base de datos relacional.
- Como segundo Punto se creó el archivo de configuración hibernet.cfg que es en donde a través del driver de conexión nos podemos comunicar con la base de datos Oracle 11g.
- Luego se procedió a crear los archivos de mapeo por ejemplo Persona.xml en donde se hace la relación respectiva, atributos de cada clase vs. Campos de la tabla.
- Una vez terminado el mapeo se procede con la creación de las clases necesarias para las pruebas a realizar.

Generación del plan de pruebas

Como se mencionó anteriormente las pruebas para ambas metodologías serán realizadas mediante un aplicativo de Prueba (TEST). Estas pruebas tienen una doble finalidad , por una parte simular un proceso completo que pueda ocurrir sobre una base de datos, ante situaciones normales de ingreso y por otro lado ayudar a detectar desventajas ante cambios en el código de la aplicación.

Para llevar a cabo esto, se deberá implementar el código necesario, en la etapa de construcción, y poder representar lo establecido en el siguiente plan de pruebas:

Plan de pruebas para aplicación de TEST

- **Portabilidad**

  - ✓ Las pruebas serán realizadas en Plataformas diferentes (Linux y Windows) para ambas metodologías.

- **Desempeño**

  - ✓ Realización de operaciones de Inserción, consulta, eliminación de forma masiva, evaluando el tiempo de respuesta que arroja la aplicación de acuerdo al uso de cada metodología.

  - ✓ Medición del tiempo de respuesta frente a la ejecución de operaciones Combinadas para tratar de simular las transacciones normales realizadas por un sistema transaccional definiendo para esta prueba (30% Ingreso, 40% Consulta, 20% modificación, 10% eliminación)

- ✓ Medición de Consumo de Recursos durante las transacciones realizadas por el aplicativo. (Nivel de Uso de memoria y procesador de la maquina que emplea la metodología).

- **Estabilidad en Plataformas (Linux y Windows)**

  - ✓ Probar conexiones simultáneas (soporte en base a conexiones o sesiones abiertas por cada metodología)

- **Tolerancia a fallos en Plataformas (Linux y Windows)**

  - ✓ Respaldo de información en caso de fallos
  - ✓ Que sucede con las transacciones que se encuentran en procesamiento y las ya realizadas anteriormente cuando se cae el servidor)

- **Flexibilidad**

  - ✓ Que pasa en un Cambio de Versión?

  - ✓ Que sucede si existieran cambios estructurales en la aplicación.

- **Facilidad de Utilización**
  - ✓ Nivel de complejidad de la metodología y grado de utilización

- **Pruebas conexión remota**

- ✓ Esta prueba tiene la finalidad de comprobar que los servidores para las bases e datos están disponibles a través de una red de computadores.

Resultados de las pruebas

PRUEBA # 1: Realización de operaciones de Inserción, consulta, eliminación de forma masiva, evaluando el tiempo que arroja la aplicación de acuerdo al uso de cada metodología

Gráfico No.12 - RESULTADO DE OPERACIONES METODOLOGIA BDOO

Elaborado por: Estefanía Barahona

Gráfico No.13 - RESULTADO DE OPERACIONES METODOLOGIA ORM

Elaborado por: Estefanía Barahona

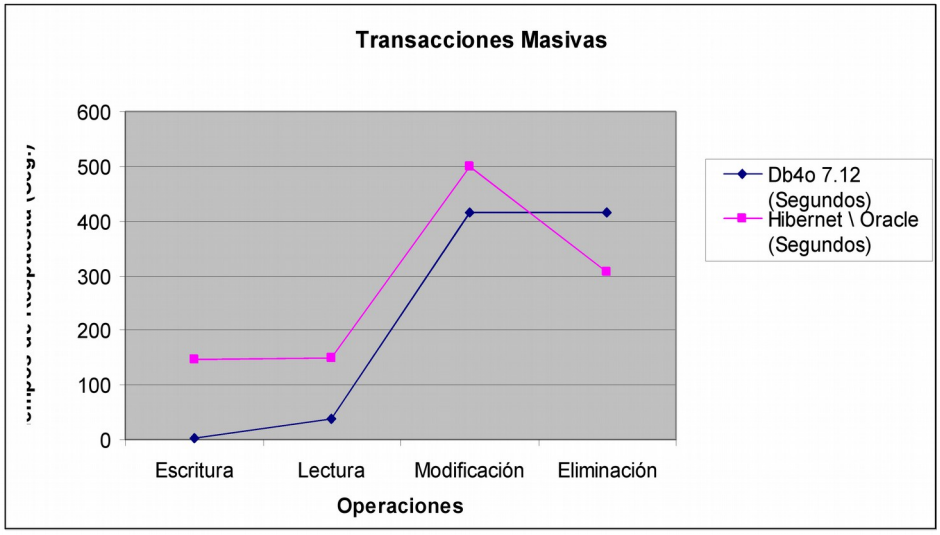
ANALISIS

Esta prueba fue realizada con un número máximo de 3000 registros y se obtuvieron los siguientes resultados.

CUADRO No. 19

Gráficos Estadísticos de Resultados

	Db4o 7.12 (Segundos)	Hibernate \ Oracle (Segundos)
Escritura	3.14	146.39
Lectura	38.5	150.38
Modificación	416.69	500.45
Eliminación	416.29	305.51



Elaborado por: Estefanía Barahona

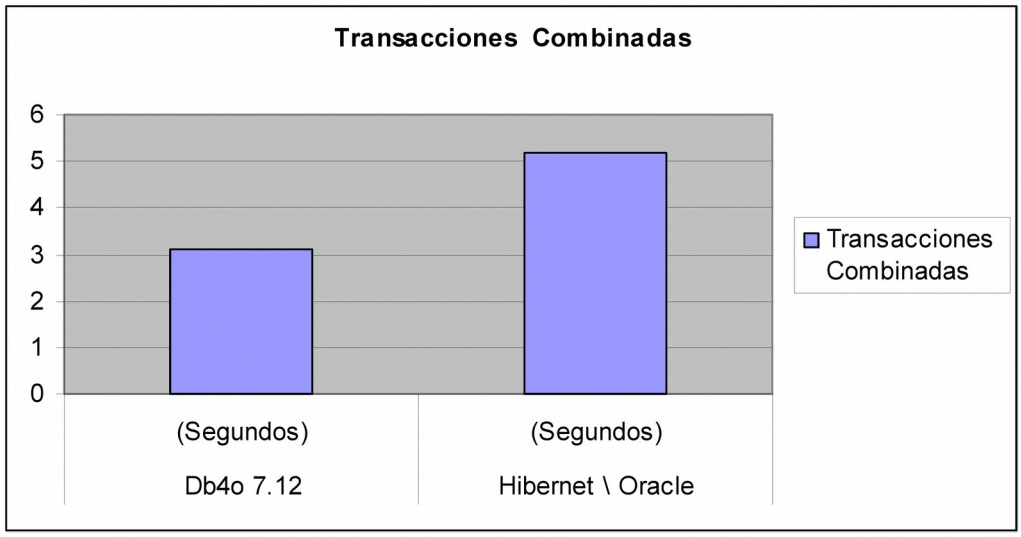
Estos datos nos demuestran que Db4o es mucho más rápido que Hibernate tanto en transacciones de lectura, escritura y modificación, siendo lo contrario para las transacciones de eliminación.

PRUEBA # 2: Realización de operaciones Combinadas para tratar de simular las transacciones normales realizadas por un sistema transaccional en ambiente Windows definiendo para esta prueba (30% Ingreso, 40% Consulta, 20% modificación, 10% eliminación).

CUADRO No. 20
RESULTADO DE OPERACIONES COMBINADAS PARA LA METODOLOGIA BDOO Y METODOLOGIA ORM

	Db4o 7.12 (Segundos)	Hibernet \ Oracle (Segundos)

Transacciones Combinadas	3.12	5.20



Elaborado por: Estefanía Barahona

PRUEBA # 3: Medición de Consumo de Recursos durante las transacciones realizadas por el aplicativo.

ANALISIS

Los recursos de memoria y procesador aumentaron considerablemente un 10% al momento de realizarse las pruebas con Hibernate, mientras que con DB4O fue casi transparente la utilización de recursos.

PRUEBA # 4

Pruebas conexión remota

Esta prueba tiene la finalidad de comprobar que los servidores para las bases e datos están disponibles a través de una red de computadores.

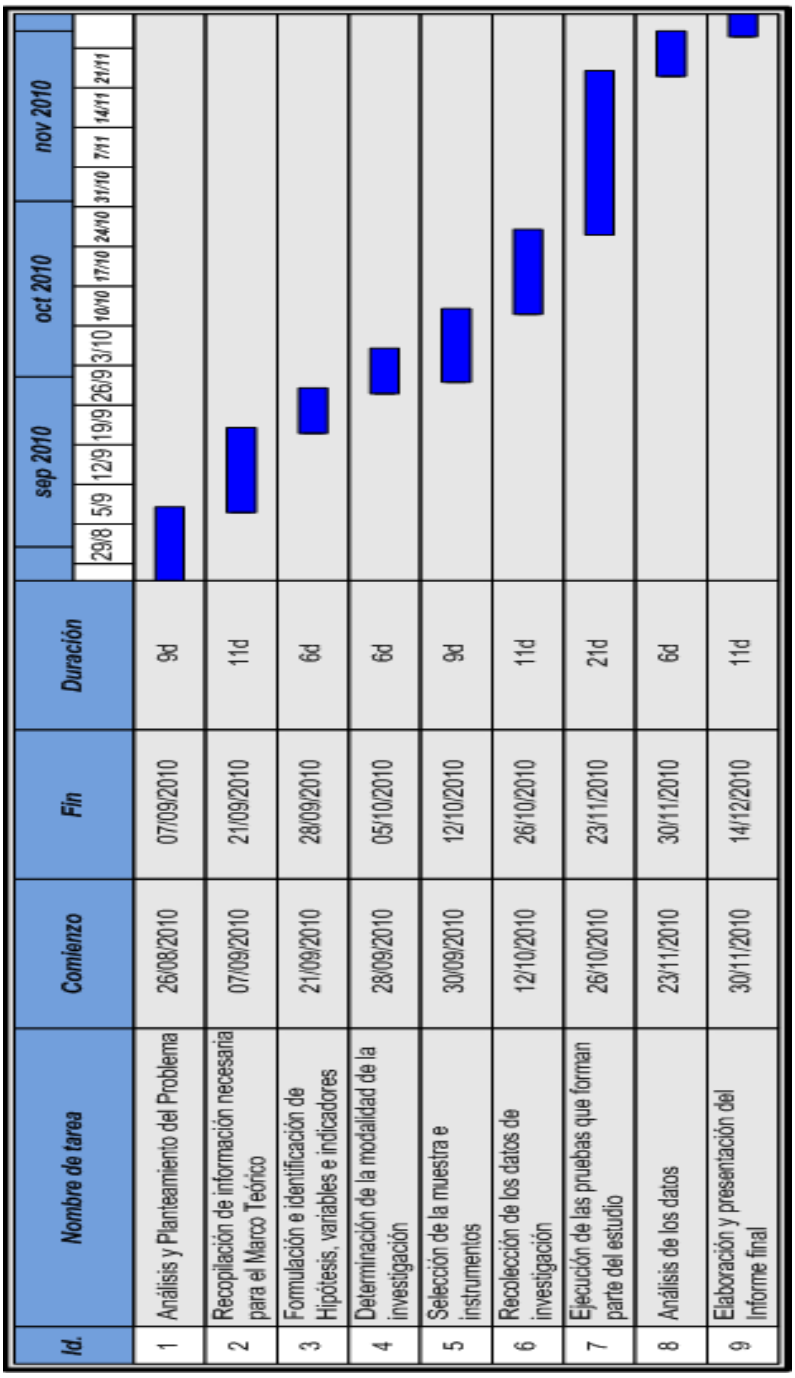
Las pruebas realizadas fueron hechas en ambiente Cliente Servidor accesando desde otra máquina a la aplicación, para este caso las pruebas resultaron exitosas tanto para la metodología ORM usando el motor de persistencia Hibernate como DB4DOO con la base DB4O.

Con esto logramos probar que la base DB4Objects trabaja perfectamente en ambientes Cliente Servidor soportando conexiones simultáneas.

CAPÍTULO IV

MARCO ADMINISTRATIVO

CRONOGRAMA



Id.	Nombre de tarea	Comienzo	Fin	Duración
1	Análisis y Planteamiento del Problema	26/08/2010	07/09/2010	9d
2	Recopilación de información necesaria para el Marco Teórico	07/09/2010	21/09/2010	11d
3	Formulación e identificación de Hipótesis, variables e indicadores	21/09/2010	28/09/2010	6d
4	Determinación de la modalidad de la investigación	28/09/2010	05/10/2010	6d
5	Selección de la muestra e instrumentos	30/09/2010	12/10/2010	9d
6	Recolección de los datos de investigación	12/10/2010	26/10/2010	11d
7	Ejecución de las pruebas que forman parte del estudio	26/10/2010	23/11/2010	21d
8	Análisis de los datos	23/11/2010	30/11/2010	6d
9	Elaboración y presentación del Informe final	30/11/2010	14/12/2010	11d

PRESUPUESTO

A Continuación se detallan los gastos que se llevaron a efecto, para el cumplimiento del objetivo del proceso de la investigación.

Cuadro # 21

Detalle de egresos del proyecto

EGRESOS	DÓLARES
Suministros de oficina y computación	$ 200.00
Fotocopias	$ 20.00
Libros y documentos	$ 150.00
Computadora y servicios de Internet	$ 100.00
Transporte	$ 100.00
Refrigerio	$ 55.00
Empastado, anillado de tesis de grado	$ 50.00
TOTAL……	$ **675.00**

Elaborado por: Estefanía Barahona

Ingresos

Los recursos que se emplearon durante el desarrollo del trabajo de grado son los siguientes:

Sueldo Empleado para recursos	$ 600.00
Otro	$ 75.00
TOTAL DE INGRESOS	$ 675.0

Egresos

Los materiales que se utilizaron son los siguientes:

- Impresiones $ 150.00
- Hojas A4 $ 50.00

• Fotocopias	$ 20.00
• Servicio de Internet durante 4 meses	$ 100.00
• Transporte	$ 100.00
• Anillado de tesis	$ 50.00
• Libros	$150.00
• Almuerzos y Refrigerios	$ 55.00
TOTAL DE EGRESOS	$675.00

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Una vez realizado el análisis de cada una de las respuestas del Instrumento aplicado, se enuncian las conclusiones.

(Pueden utilizarse cuatro o cinco páginas)

RECOMENDACIONES

De todas las conclusiones se toman dos grandes grupos para unificar la apreciación crítica y señalar los aspectos que sean generadores de actitudes y/o procedimientos.

BIBLIOGRAFÍA

LIBROS

Hernández, Y., Montilla J (2005). Estudio comparativo del manejo de objetos multimedia en los sistemas manejadores de bases de datos orientado a objetos (SMBDOO) FastObjects y Jasmine II. Caracas: Universidad Central de Venezuela, Facultad de Ciencias, Escuela de Computación.

Ian Graham – 2008. Métodos orientados a Objetos

Alejandro Alberca Manzaneque (2008). - Bases de datos Orientadas a Objetos y Bases de Datos Objeto-Relacionales

Jesús Galvez Díaz-Tendero (2008). - Tendero Modelos Avanzados de Bases de Datos

Yosly Hernández Bieliukas y Prof. Antonio Silva Sprock -ND 2009-02 - (2009) Fundamentación Teórica de las Bases de Datos Orientadas a Objetos

Marín,N.(2001). Tesis Doctoral "Estudio de la Vaguedad en los Sistemas de Bases de Datos Orientados a Objetos: Tipos Difusos y sus Aplicaciones". Universidad de Granada.

PUBLICACIONES

Blanca Cubas Marzo 2009
http://www.cesnavarra.net/cesdigital/Lists/Noticias%20CESDigital/Ao2008.aspx

Persistencia-de-objetos-con-hibernate
http://www.slideshare.net/ingeniods/persistencia-de-objetos-con-hibernate

DIRECCIONES WEB

Db4Objects. http://www.db4o.com/
http://www.db4o.com/espanol/

http://www.programacion.com/articulo/conceptos_basicos_de_orm_object_relational_mapping_349