INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES

TESIS

QUE PARA OBTENER EL GRADO DE

Maestro en Ciencias en Ingeniería de Sistemas

PRESENTA

Lic. Benina Velázquez Ordóñez

DIRECTORES DE TESIS

Dr. Jesús Manuel Olivares Ceja

M en C. Graciela Vázquez Álvarez

MÉXICO D.F 2006

SIP-14

INSTITUTO POLITECNICO NACIONAL
SECRETARIA DE INVESTIGACION Y POSGRADO

ACTA DE REVISION DE TESIS

En la Ciudad de México, D. F. siendo las 13:00 horas del día 9 del mes de octubre del 2006 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de la E. S. I. M. E. para examinar la tesis de grado titulada:

"SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES"

Presentada por el alumno:

Apellido paterno	Apellido materno	Nombre(s)
VELAZQUEZ	**ORDOÑEZ**	**BENINA**

Con registro: **B 0 3 1 5 8 6**

Aspirante al grado de:

MAESTRO EN CIENCIAS

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACION DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISION REVISORA

Director de tesis

M. EN C. GRACIELA VAZQUEZ ALVAREZ
DR. JESUS MANUEL OLIVARES CEJA

Presidente

DR. LUIS MANUEL HERNANDEZ SIMON

Segundo Vocal

DR. JESUS MANUEL OLIVARES CEJA

Tercer Vocal

M. EN C. EFRAIN JOSE MARTINEZ ORTIZ

Secretario

M. EN C. LEOPOLDO A. GALINDO SORIA

Suplente

M. EN C. IGNACIO E. PEON ESCALANTE

EL PRESIDENTE DEL COLEGIO

DR. JAIME ROBLES GARCIA

SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION

INSTITUTO POLITÉCNICO NACIONAL

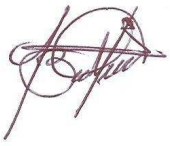
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESION DE DERECHOS

En la Ciudad de México, D.F., del día 07 del mes de noviembre del año 2006, la que suscribe Benina Velázquez Ordoñez alumna del Programa de Maestría en Ciencias con especialidad en Ingeniería de Sistemas con número de registro B031586, adscrito a la Sección de Estudios de Posgrado e Investigación de la ESIME Unidad Zacatenco, manifiesta que es el autor(a) intelectual del presente trabajo de Tesis bajo la dirección del Dr. Olivares Ceja Jesús Manuel y la M.en C. Graciela Vázquez Álvarez y cede los derechos del trabajo intitulado Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección jesus@cic.ipn.mx , chelita0423@yahoo.com.mx, bvelazquez@azul.bnct.ipn.mx Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Benina Velázquez Ordoñez



Dedico el esfuerzo de este trabajo a:

Mis padres: *Por haberme enseñado que la perseverancia, el esfuerzo y la dedicación día a día, son el camino para lograr los propósitos de la vida.*

Jaz. *Por su tiempo y apoyo incondicional, que me ofreció en todo momento.*

A mis asesores:

Jesús. O. C. *Por su apoyo, su tiempo, dedicación, al brindarme la oportunidad de recurrir a su capacidad y experiencia, en un marco de confianza y darme la orientación de culminar este trabajo.*

Graciela. V.A. *Por su tiempo y dedicación, al orientarme en la culminación de este trabajo.*

ÍNDICE

GLOSARIO DE TÉRMINOS

ACL	*Agent Communication Language*, el cual es un lenguaje de comunicación entre agentes, que implica la interacción a un nivel más semántico [45].
Agente	"Un sistema de hardware y/o software autónomo (independiente del usuario), el cual interactúa con su entorno (u otros agentes o humanos), guiado por uno o varios propósitos, su comportamiento es proactivo (reacciona a eventos y a veces se anticipa haciendo propuestas), adaptable (puede enfrentar situaciones novedosas), sociable (se comunica, coopera y/o negocia), su comportamiento es predecible en cierto contexto" [17]. "Un agente inteligente es una entidad de software que, basándose en su propio conocimiento, realiza un conjunto de operaciones para satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere" [3].
Agente de Interfaz de Usuario	"Programa que emplea las técnicas de la inteligencia artificial a fin de proveer asistencia al usuario en una aplicación particular. La metáfora es la de un asistente personal que colabora con el usuario en el mismo entorno de trabajo" [21].
Agentes de Consulta	Un sistema de agentes orientados a consulta origina uno o más agentes en respuesta a la pregunta formulada por un usuario. Estos agentes trabajan en representación del individuo mientras dura la consulta, recogiendo información de todas las bases de datos disponibles [33].
Agente Gestor de Bases de Datos	Es el encargado de diseñar el sistema de recuperación de información y de suministrar a los agentes de tipo funcional un agente facilitador para la correcta resolución de las subconsultas. Se genera cada vez que tienen que realizar una consulta para un individuo [33].
Agente de Usuario	Es una persona en concreto, los agentes de usuario siempre están activos, buscando información y suministrándosela a su creador [33].
APE	Sistema de Aprendizaje Automático [19].
API	(Application Program Interface). Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones [47].
Arquitectura ACO	Arquitectura que maneja Agentes, Memoria Caché y Ontología.
Atomicidad	"Se garantiza (desde un punto de vista lógico) que se ejecuten las operaciones de una transacción en su totalidad o que no se ejecuten en lo absoluto, aun cuando (por decir algo) el sistema fallara a la mitad del proceso" [3]. Consiste en que cada atributo debe contener un único valor del dominio [3].

Base de Datos	Es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización [5].
Bright	Es un proyecto (Brazilian Internet Guide in Hypertext) que evoluciona hacia un mecanismo de búsqueda llamado Radix [16].
Caché	Es un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, respecto a la copia en el caché [53]. Una caché es un sistema especial de almacenamiento de alta velocidad. Puede ser tanto un área reservada de la memoria principal como un dispositivo de almacenamiento de alta velocidad independiente [35].
Concurrencia	Se refiere al hecho de que los DBMSs (Sistema de Administración de Base de Datos) permiten que muchas transacciones accedan a una misma base de datos a la vez [3].
Control de Concurrencia	Cuando se ejecutan varias transacciones de manera simultánea en distintos procesos(o distintos procesadores), se necesita cierto mecanismo para mantener a cada uno lejos del camino del otro [2].
Cliente-Servidor	Es un sistema distribuido en el cual algunos son sitios clientes y algunos son servidores, los datos residen en los sitios servidores, las aplicaciones son ejecutadas en los sitios clientes [3].
Cliente	Es una estación de trabajo personal adaptada a las necesidades del usuario final y por lo tanto, capaz de soportar mejores interfaces [3]. Corresponde a la denominación de un programa (*software*) utilizado para contactar y obtener datos desde un software servidor que se encuentra generalmente en otro ordenador. En la arquitectura cliente-servidor, existe un software cliente corriendo en un ordenador y un software servidor corriendo en otro ordenador que interactúan entre ellos y ejecutan alguna tarea específica [36].
Data Warehouse	Es un conjunto de datos integrados orientados a una materia, que varían con el tiempo y que no son transitorios, los cuales soportan el proceso de toma de decisiones de la administración. (W.H. Inmon, se considera como el padre del data warehouse) [44].
Diagrama de Caso de Uso	Permite el modelado de una vista del negocio que se detalla con un escenario [40].
Diagrama de Clases	Es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema [40].
Diagrama de Secuencia	Muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores

	horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria [40].
DBMS	Acrónimo de Data Base Management System. (Sistema Manejador de Base de Datos. Es el software que maneja el acceso a la base de datos [3].
FIPA	(Foundation for Intelligent Physical Agents) Fundación para el desarrollo de agentes físicos inteligentes [51].
Gestionar Estímulos	Son acciones de entorno (el mundo físico, un usuario vía una interfaz gráfica, una colección de otros agentes, Internet o todos ellos combinados), a los cuales reacciona al sistema [34].
Groupware	Trabajo colaborativo [7].
JAVA	Lenguaje de programación orientada a objetos inventado por la firma Sun Microsystems, bajo la dirección de James Gosling similar al lenguaje C++ y que permite escribir programas que funcionan en diversas plataformas, independiente del sistema operativo que utilice [36].
JADE	(Java Agent Development Framework) Es un conjunto de bibliotecas en Java que nos permite crear sistemas multiagente que cumplen los estándares de la FIPA que es herramienta de desarrollo de sistemas multiagente [54].
JDK	Java Development Kit"(JDK),"Standard Development Kit" (SDK) y "Java 2 Standard Edition" (J2SE) son nombres para el mismo componente e incluyen: El API de Java, el JRE (JVM), compilador de Java y otras funcionalidades definidas por Sun [50].
KQML	Manejador de Consultas de un Lenguaje de Conocimiento (Knowledge Query Management) [45].
Latencia	"Es el lapso necesario para que un paquete de información viaje desde la fuente hasta su destino. La latencia y el ancho de banda, juntos, definen la capacidad y la velocidad de una red" [37].
Login	El nombre dado a una cuenta y que permite el acceso a un ordenador o bien la acción de entrar a un sistema de ordenadores [36].
Modelo de Datos	Es una representación de la aplicación que capture las propiedades estáticas y dinámicas requeridas para dar soporte a los procesos deseados (por ejemplo, transacciones y consultas). Además de capturar las necesidades dadas en el momento de la etapa de diseño, la representación debe ser capaz de dar cabida a eventuales futuros requerimientos [49].
Multiprocesamiento	Arquitectura informática en la que varios procesadores comparten la misma memoria, que contiene una copia del sistema operativo, una copia de todas las aplicaciones que están en uso y una copia de los datos. Como el sistema

	operativo divide la carga de trabajo en tareas y asigna esas tareas a los procesadores que estén disponibles, SMP (Sistema de Multiprocesamiento Simétrico) reduce el tiempo de las transacciones [48].
Negociación	Cuando dos agentes entran en conflicto, tratan de resolverlo mediante un acuerdo bilateral, esta línea de investigación ha sido seguida por Lesser (Conry, 1988) Durfee (Durfee, 1989) y Sycara (1989) [38].
Ontología	Es la formalización de una conceptualización, generalmente descriptiva. En este contexto, es considerada como una colección de conceptos con un árbol como la estructura, formando una taxonomía jerárquica bajo la relación de subconjunto de padres a hijos. Los conceptos son independientes de cualquier lenguaje de comunicación. Cada concepto plantea un significado único y es empleado en un nodo único en la ontología. Un agente no sabe (conoce) todos los conceptos, en otras palabras, la ontología de un agente es incompleta. Por lo general, dos agentes poseen conceptos diferentes, que quieren decir que ellos tienen los árboles diferentes de conocimiento. Sin embargo, con este trabajo consideramos que todos los agentes emplean una ontología común relacionada con el contenido de base de datos [17] [29].
Ontología -Tema	Es un árbol de conceptos.
Password	Es el código secreto utilizado para acceder a un sistema protegido [36].
Proceso	Es un programa en ejecución. Consta del programa ejecutable, sus datos y pilas, contador y otros registros, además de toda la información necesaria para ejecutar el programa [2].
RAM	Es un dispositivo de bloque con dos comandos: leer un bloque y escribir en un bloque. Estos bloques se almacenan en memorias rotativas, como los discos flexibles o discos duros. Tiene la ventaja de acceso instantáneo (no existe retraso rotacional), lo que lo hace adecuado para el almacenamiento de programas o datos con acceso frecuente [2].
RADIX	Es un algoritmo de ordenamiento, usado para ordenar elementos identificados por claves únicas [39].
RUP	Es un proceso iterativo e incremental de Ingeniería de Software la cual designa tareas y responsabilidades [52].
Servidor de Base de Datos	Un nodo de la red o estación, dedicada a almacenar y proporcionar acceso a una base de datos compartida [3].
Servidor	Soporta todas las funciones básicas del Sistema Manejador de la Base de Datos (DBMS) [3*]. Es un ordenador o un software que provee una clase especial de servicio al software clientes que están corriendo en otros ordenadores y al que acceden para realizar una función determinada. Un ordenador funcionando como servidor puede tener operando varios software servidores para prestar servicios, por ejemplo: servidor de WWW, servidor de Mail, etc. [36].

Servlet	Son objetos que corren dentro del contexto de un servidor web (ejemplo: Tomcat) y extienden su funcionalidad. (Tomcat sólo es un contenedor de servlets) [38].
Sistema	Es una reunión o conjuntos de elementos relacionados que interaccionan entre si, para lograr un objetivo. Los elementos de un sistema pueden ser conceptos, en cuyo caso se trata de un sistema conceptual. Los elementos de un sistema pueden ser objetos y sujetos. Finalmente, un sistema puede estructurarse de conceptos, objetos y sujetos. Por tanto, un sistema es un agregado de entidades, viviente o no de otros sistemas a los que se le llaman subsistemas. En la mayoría de los casos, se puede pensar en sistemas más grandes, los cuales comprenden otros sistemas y que se llaman el sistema total y sistema integral [4].
Sociedad de Agentes Comunicantes	La comunicación es imprescindible para que exista cooperación. En sistemas de agentes con representación del conocimiento, esta comunicación se establece mediante mensajes. En sistemas de agentes reactivos la comunicación se establece mediante la observación de determinadas señales procedentes de otros agentes (posiciones, situaciones, etc.) [38].
Software	La suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo. Lleva a cabo las funciones de entrada /salida comunes a todos los dispositivos, además de proporcionar una interfaz uniforme del software a nivel usuario [2].
SHOE	(Simple HTML Ontology Extensions) Es un sistema que utiliza una extensión de una ontología simple HTML [12].
SQL	Siglas de Structured Query Language. Lenguaje Estructurado de Consultas. Es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos [3].
SMP	Multiprocesamiento simétrico (SMP) [48]
Tomcat	Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Se le considera un servidor de aplicaciones [38].
Transacciones	Es una colección de operaciones que se lleva a cabo como una función lógica simple en una aplicación de bases de datos [1].
UML	(Unified Modelling Language) El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación [40].

LISTA DE FIGURAS

LISTA DE TABLAS

Resumen

SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES

En esta tesis se desarrolla un Sistema en el que se mejora el acceso concurrente de usuarios a una base de datos con pocas operaciones de actualización y un gran número de consultas. Esto se hace mediante agentes que guardan el perfil del usuario y accesan una memoria caché. La memoria caché almacena las consultas más frecuentes que emiten los distintos usuarios conectados al sistema. La actualización de la caché se hace automáticamente mediante un algoritmo de eliminación del menos popular.

Este algoritmo garantiza que las consultas nuevas permanezcan un tiempo.

Las búsquedas que llevan a cabo los agentes se hacen cuando hay poca actividad en la base de datos y se le presentan al usuario a manera de un listado para facilitar su elección.

La metodología que se empleo para la elaboración de este trabajo, es la combinación de la metodología RUP (Racional Unified Process) y la metodología LGS propuesta para el Ciclo de Vida de un del Desarrollo de un Sistema de Información (CVDSI).

Con la arquitectura ACO se mejora el tiempo de respuesta para las consultas que se incluyen en la caché

Abstract

DATABASE CONCURRENCY CONTROL SYSTEM USING AGENTS

This theses shows a system development for improving users concurrent access in a database with few update operations and plenty of queries. The solution involves the use of agents that store a user profile and access a global cache memory. The cache stores the most frequent asked requests.

This algorithm guarantees that the new consultations remain a time.

The searches that carry out the agents do when there are few transactions in the data base and they appear to him to the user to way of a listing to facilitate his election.

The methodology that use for the elaboration of this work, the combination of methodology RUP (Racional Unified Process) and propose methodology LGS for Service Life the Development a System of Information (SLDSI).

With architecture ACO the response time for consultations improves that are included in the cache.

INTRODUCCIÓN

En aplicaciones a grandes bases de datos, por ejemplo, en una bodega de datos (datawarehouse) o en una biblioteca digital, se ha detectado que a menudo muchos usuarios solicitan una misma consulta, con las consecuentes respuestas similares. En este trabajo se aborda el caso de una biblioteca digital, la cual, recibe poca atención por parte de los investigadores de base de datos ya que no es una aplicación donde se maneje dinero. En la biblioteca digital, se realizan transacciones sencillas, los usuarios que acceden al sistema simultáneamente deben esperar por su respuesta debido al volumen de solicitudes que existen de manera concurrente y conforme aumenta el número de usuarios el tiempo de respuesta generalmente se hace más lento. Otra consecuencia del gran número de usuarios, es al momento de conectarse al sistema, ya que un usuario tiene que realizar varios intentos antes de ingresar.

En varias ocasiones para un usuario que ha logrado su conexión se presenta otro problema, la dificultad para expresar sus consultas debido a que en muchas ocasiones desconocen las palabras clave para formular sus preguntas y como se quedan "pensando" utilizan tiempo de conexión que en el modelo cliente/servidor impide a otro usuario conectarse.

Con lo anterior, se provoca pérdida de tiempo y estrés al usuario que trata de tener una conexión o bien de encontrar la información que requiere.

En esta tesis se propone realizar un sistema para dar una solución al problema del acceso a una base de datos concurrente a través de agentes y ayudarlo en la formulación de sus solicitudes al modelar en los agentes el perfil del usuario y la identificación de sus temas de interés mediante una ontología.

El perfil de un usuario contiene los datos de cada usuario y los conceptos de su interés. Estos conceptos se encuentran en una ontología, en la cual es posible navegar para generar posibles solicitudes de información que efectuaría un usuario. Una ontología en

este trabajo es una organización jerárquica de los conceptos que se encuentran en una biblioteca digital. Utilizando la ontología y un conjunto de patrones de solicitudes a la base de datos es posible generar consultas que pueden resultar de interés para un usuario, con lo cual, se intenta brindarle ayuda al usuario en la formulación de sus consultas.

Con el interés de encontrar una forma de mejorar el tiempo de respuesta para un conjunto de usuarios que acceden concurrentemente a un sistema de biblioteca digital, se propone en esta tesis el uso de una memoria caché donde se almacenan tanto las consultas más frecuentes emitidas por el conjunto de usuarios que accesan al sistema como su respuesta. Con la velocidad de acceso que se tiene a la memoria RAM y dado el tamaño de la caché, el acceso concurrente a la misma sí logra una mejora en el tiempo de acceso que se llevaría a cabo si la consulta se hiciera hasta la base de datos.

La tesis se encuentra estructurada en cuatro capítulos. En el capítulo 1, se presenta el marco conceptual describiendo los conceptos relacionados con base de datos, concurrencia, agentes, caché y ontología. También se describe la metodología aplicada en el desarrollo de esta tesis y las fases que se deben de cumplir para el desarrollo de una aplicación. Por último se comenta la propuesta de solución al problema, la justificación del trabajo de investigación, el objetivo general del trabajo y los objetivos específicos.

En el capítulo 2, se describen los trabajos relacionados con base de datos, agentes y ontologías que resultan de interés para esta tesis por ser algunas líneas que ya se han desarrollado y permiten enmarcar el presente trabajo.

En el capítulo 3, se describe como se desarrolló del sistema de control de concurrencia.

En el capítulo 4, se describe la implantación y las pruebas del sistema. Se comentan los resultados obtenidos y mediante una gráfica se visualiza el comparativo de un sistema que maneja agentes y otro que no lo maneja.

Finalmente, se mencionan las conclusiones, trabajos futuros, anexos, la bibliografía y las referencias.

CAPÍTULO 1
MARCO CONCEPTUAL Y METODOLÓGICO

En esta sección se describen los conceptos que fundamentan el desarrollo de esta tesis. Aquí se dan a conocer los conceptos utilizados en las bases de datos y la recuperación de información. Se continúa con los conceptos de la concurrencia y su problemática (acceso concurrente en las bases de datos) y el tiempo de respuesta. El tema de Agentes se estudia porque es una tecnología de reciente creación que se propone como una alternativa a las formas de mejorar el rendimiento en el acceso concurrente en las bases de datos. Los Agentes se apoyan en una estructura de conceptos llamada ontología, la cual se utiliza para generar preguntas en nombre del usuario y de esta forma tener precalculadas algunas solicitudes del usuario.

También se describe en este capítulo la metodología RUP (Rational Unified Process), que se usa para la creación de software y la metodología LGS, que se emplea para conceptualizar el ciclo de vida del desarrollo de un sistema de información. Generando de ambas la metodología propia para el desarrollo de esta tesis.

1.1 MARCO CONCEPTUAL

El marco conceptual de esta tesis está conformada por los temas de: base de datos, concurrencia, agentes y ontologías.

1.1.1 PIRAMIDE CONCEPTUAL

Los cimientos de la pirámide conceptual están constituidos por conceptos fundamentales sobre los que se va conformando la estructura requerida para desarrollar el Sistema de Control de Concurrencia utilizando Agentes para una Base de Datos, con lo que se permitirá lograr que el objetivo primordial representado en la parte más alta de la pirámide., ver figura 1.

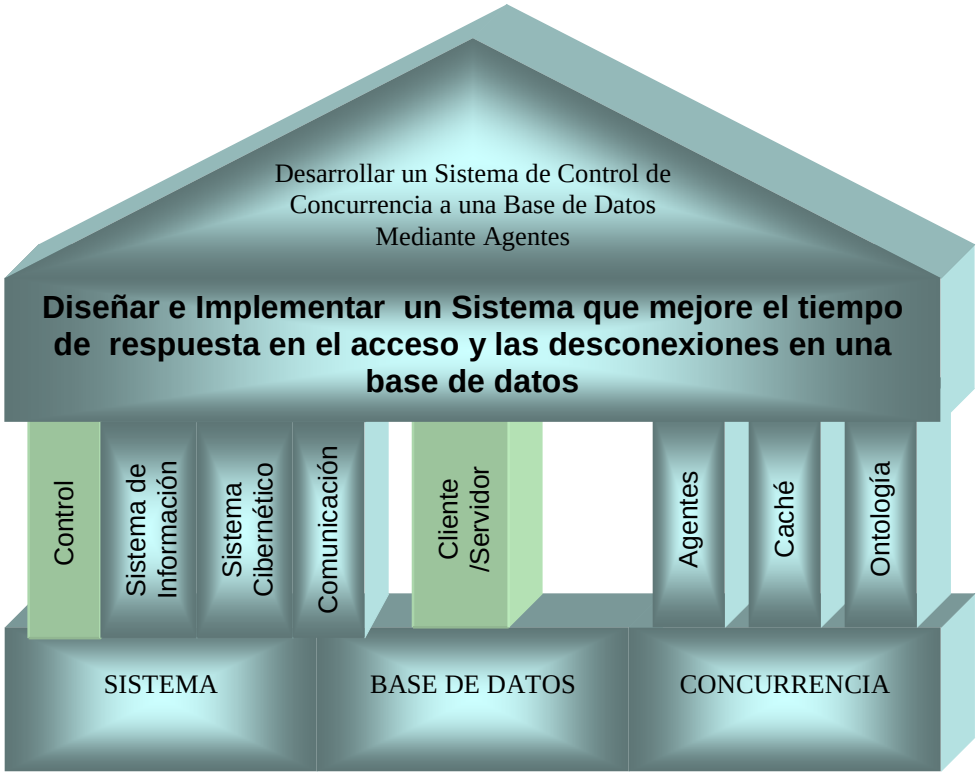


Figura 1, Pirámide Conceptual.

1.1.2 SISTEMA

Un *Sistema [4],* es una reunión o conjunto de elementos relacionados que interaccionan entre sí, para lograr un objetivo. Los elementos de un sistema pueden ser conceptos, en cuyo caso se trata de un sistema conceptual. Los elementos de un sistema pueden ser objetos y sujetos. Finalmente, un sistema es un agregado de entidades, vivientes no vivientes o ambas. Por tanto, es suficiente visualizar que los sistemas se componen de otros sistemas a los que se llaman subsistemas. En la mayoría de los casos, se puede pensar en sistemas más grandes o súper ordinales, los cuales comprenden otros sistemas y que se llaman el *sistema total* o *sistema integral*.

1.1.3 SISTEMAS DE INFORMACIÓN

El *sistema de información* [16], es un conjunto integrado de programas de computadoras, equipos y servicios de cómputo, cuyo propósito fundamental es: obtener y proporcionar información de apoyo a las funciones de la institución.

1.1.4 PROCESO CIBERNÉTICO

La *cibernética* [58], es la ciencia de la comunicación y el control ya sea en el animal o en la máquina. La comunicación integra y de coherencia a los sistemas; el control regula su comportamiento. La cibernética comprende los procesos y sistemas de transformación y su concreción en procesos físicos, fisiológicos, psicológicos, etc., de transformación de la información.

1.1.5 BASE DE DATOS

El auge de las bases de datos se da en la década de los 80's, desde entones se ha llevado acabo estudios sobre este tema. Una *Bases de Datos* [5], "Es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización", los sistemas de bases de datos están diseñados para almacenar grandes cantidades de información. Los sistemas de bases de datos deben proporcionar seguridad de la información además de que van cambiando conforme se van guardando datos y eliminando las mismas.

El *diseño de bases de datos* se descompone en tres etapas: diseño conceptual, diseño lógico y diseño físico. El diseño conceptual es el proceso por el cual se construye un modelo de la información que se utiliza en una empresa u organización, independientemente del Sistema Manejador de la Base de Datos (SMBD) que se vaya a utilizar para implementar el sistema y de los equipos informáticos o cualquier otra consideración física.

Un modelo conceptual de una base de datos es un conjunto de conceptos que permiten describir la realidad mediante representaciones lingüísticas y gráficas. Los

modelos conceptuales poseen las propiedades de expresividad, simplicidad, minimalidad y formalidad.

El modelo conceptual más utilizado es el modelo *entidad-relación*, que posee los conceptos de entidad, relación, atributo, dominio de atributos, identificadores y jerarquías de generalización.

El término *entidad* se refiere a cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc. Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual. Hay dos tipos de entidades: fuertes y débiles. Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil.

Una *relación* es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria, si son tres las entidades participantes, la relación es ternaria, etc.

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

El diseño de una Base de Datos sigue una *metodología que conecta tres etapas* se construye un esquema conceptual local para la vista de cada usuario o grupo de usuarios. En el diseño lógico se obtiene un esquema lógico local para cada esquema conceptual local. Estos esquemas lógicos se integran después para formar un esquema lógico global que representa las vistas de los distintos usuarios de la organización. Por último, en el diseño físico, se construye la implementación de la base de datos sobre un SMBD

determinado. Ya que este diseño debe adaptarse al SMBD, es posible que haya que introducir cambios en el esquema lógico para mejorar las prestaciones a nivel físico.

El *acceso a las bases de datos* se lleva a cabo por uno o varios usuarios para recuperar información que satisface sus requerimientos de información. Cuando son varios los usuarios se dice que es un *acceso concurrente* a la misma base de datos. El objetivo principal en los sistemas multiusuarios es permitirle a cada usuario que el sistema se comporte como si un usuario estuviera trabajando él solo. Para lograr este objetivo se han creado las transacciones y el control de la concurrencia.

La operación de un sistema de base de datos puede ser visto como un sistema que tiene una estructura de dos partes que consisten en un servidor y conjunto de clientes. El *servidor* es el SMBD el cual se encarga del acceso a la base de datos procesando las peticiones de información utilizando algún lenguaje de consulta de datos, por lo regular es el Lenguaje Estructurado de Consultas (Structered Query Language, SQL). Por otro lado, los *Clientes* son las aplicaciones que emiten las peticiones de información sobre el DBMS.

Los clientes y servidores operan en maquinas independientes dando lugar al procesamiento distribuido. En donde *procesamiento distribuido* significa que distintas maquinas pueden conectarse en una red de comunicación como el Internet de tal manera que una sola tarea de procesamiento se pueda extender a varias maquinas. La comunicación entre las diversas maquinas se maneja mediante un software de administración de redes.

El acceso a las bases de datos por parte de los diferentes clientes puede causar algunos problemas como la inconsistencia en la información de la base de datos, por ejemplo, si un cliente esta modificando una tupla y otro cliente la accesa en ese momento y la modifica también, entonces los datos finales son inconsistentes. La forma de solucionar estos problemas se han estudiado por diferentes autores, en el siguiente capítulo se describen algunos de estos trabajos.

1.1.6 CONCURRENCIA

Cuando varios procesos acceden a datos comunes, se dice que es un acceso concurrente. El término *concurrencia* [3], se refiere al hecho de que los Sistemas de Manejadores de Base de Datos permiten que muchos clientes accedan a una misma base de datos a la vez. En un sistema de éstos se necesitan mecanismos de control de concurrencia con el fin de asegurar que los clientes no interfieran entre si. Uno de estos mecanismos son las transacciones.

Una *Transacción* [3], es una unidad de trabajo lógica que comprende varias operaciones de base de datos. Se crearon con el propósito de manejar una secuencia de varias operaciones de la base de datos, la importancia principal consiste en que la base de datos mantenga sus datos consistentes al final de la ejecución de una transacción.

Una transacción inicia cuando se ejecuta la instrucción (begin transaction) es decir iniciar transacción y termina cuando se ejecuta la instrucción (commit) significa ejecutar con éxito la transacción, o se da la instrucción (rollback) lo que es lo mismo terminar la transacción anormalmente, se cancela las actualizaciones hechas por la transacción, como si nunca se hubiera realizado la transacción. La única forma de que una transacción aborte es a través de la presencia de un bloqueo. Cuando una transacción aborta su ejecución se reinicia con el mismo conjunto de datos antes de que ocurriera tal evento.

Las transacciones tienen cuatro propiedades importantes: atomicidad, consistencia, aislamiento y durabilidad, también conocidas como ACID.

a) *Atomicidad:* Consiste en que cada atributo debe contener un único valor del dominio.

b) *Consistencia:* Las transacciones conservan la integridad de la base de datos.

c) *Aislamiento:* Las actualizaciones de una transacción dada están ocultas ante las demás, hasta que esa transacción sea confirmada.

d) *Durabilidad:* Una vez que una transacción es confirmada, sus actualizaciones sobreviven en las bases de datos aun cuando hay una caída posterior del sistema.

Existen diferentes mecanismos para el control de la concurrencia de transacciones los cuales se clasifican en enfoque pesimista y optimista. El enfoque optimista considera

las mejores condiciones en que puede ocurrir una transacción, mientras que el pesimista lo hace tomando en cuenta las peores condiciones.

Algunos mecanismos de control de concurrencia son:

- *Cerrojos.*
- *Marcas Temporales.*
  - *Algoritmo de Cierre.*
  - *Algoritmos basados en marcas temporales.*
- *Votaciones y Quórum.*
- *Votación Estática.*

Los *cerrojos,* se han utilizado tradicionalmente para proteger cada una de las tablas o rangos de registros en una determinada base de datos relacional; aunque en general, controla el acceso sobre cualquier recurso que pueda utilizar una transacción. Para ello, existen dos modos de utilización de los cerrojos: *exclusivo* o *de escritura*, cuando la transacción debe modificar el contenido de la relación, o *compartido* o *de lectura*, cuando la transacción únicamente quiere consultar el valor de los registros existentes. El objetivo del control de concurrencia basado en cerrojos es gestionar los recursos que las transacciones utilizan de manera que cuando estas finalicen el resultado de sus acciones sobre dichos recursos sea el mismo que el de cualquiera de las ejecuciones en serie de las transacciones. El control de concurrencia mediante cerrojos presenta la característica de que la probabilidad de que una transacción quede suspendida depende del número de cerrojos que necesite adquirir (esto es, del número de recursos a los que deba acceder para completar su trabajo) y de la probabilidad de que otras transacciones en curso hayan adquirido previamente dichos cerrojos en un modo no compatible con el que necesitaba la transacción analizada.

Una *marca temporal* o *timestamp,* es un valor numérico único que se asigna a una transacción o un recurso y que es elegido de entre una secuencia monótonamente creciente. Una forma de generar estas marcas es utilizar los *relojes lógicos* de Lamport aunque puede utilizarse cualquier otra. Utilizando marcas temporales se pueden dar dos

tipos de control de concurrencia: algoritmos de cierre y algoritmos basados en marcas temporales.

Los *algoritmos de cierre*, se emplean cuando una transacción de inicio recibe una marca temporal generada por el nodo donde ha empezado. Estas marcas temporales servirán para dar cierto tratamiento en caso de que dos transacciones presenten conflictos. Estos conflictos se dan cuando una transacción (a la que vamos a llamar *peticionaria*) intenta leer un recurso para el que actualmente otra transacción está realizando una escritura o bien cuando una transacción intenta escribir sobre un recurso en el que otra transacción está realizando una escritura o una lectura. Para resolver los conflictos entre dos transacciones se podrían emplear las siguientes acciones:

- *Espera*: La transacción peticionaria debe esperar hasta que la transacción con la que está en conflicto termine o aborte.
- *Reinicio*: Bien la transacción peticionaria o bien la que está en conflicto con ella es abortada e iniciada de nuevo. Esta estrategia de reinicio se puede realizar de dos maneras distintas:

  * *Muerte*. La transacción peticionaria es abortada e iniciada de nuevo.

  * *Rebobinado.* La transacción en conflicto con la peticionaria es marcada con la etiqueta de rebobinada" y un *mensaje de rebobinado* es enviado a todos los nodos que mantengan recursos utilizados por tal transacción. Si ese mensaje es recibido antes de que la transacción haya terminado en ese nodo entonces la transacción es abortada. En caso contrario, el mensaje no es tomado en cuenta. Si fuera abortada, la transacción seria iniciada posteriormente. La transacción peticionaria conseguirá el recurso una vez la transacción en conflicto haya terminado o haya sido abortada.

Algoritmos basados en marcas temporales: Aunque existen múltiples algoritmos basados en marcas temporales para realizar la serialización de las transacciones. En este algoritmo, el *gestor de datos* de cada nodo mantiene para cada uno de sus objetos la marca temporal más reciente con la que se han realizado por una parte las operaciones de escritura y por otra las de lectura. En este algoritmo las marcas temporales asociadas a las transacciones se

toman como una guía para realizar su serialización. Aquellas transacciones con marcas más antiguas deberán terminar antes que las que lleven marcas temporales más grandes. Además, se exige que una transacción no pueda consultar ni modificar las escrituras realizadas por una transacción con una marca temporal posterior. Tanto los algoritmos de cierre y las marcas temporales resuelven los conflictos entre transacciones abortando en algunos casos a una de ellas. Esto tiene la ventaja de que se evitaría la aparición de interbloqueos.

Existen dos algoritmos basados en marcas temporales y son los siguientes:

- ✧ *Algoritmo espera-muerte*. Es un *algoritmo no expulsivo* puesto que la transacción peticionaria nunca podría forzar a la transacción con la que esté en conflicto por abortar.
- ✧ *Algoritmo rebobinado-espera*. Este es un *algoritmo expulsivo* en el que la transacción peticionaria si podrá abortar a la que esté en conflicto con ella.

Las *votaciones y quórum*: Son técnicas de control de concurrencia basada en votaciones en las cuales se asume que los objetos a proteger están replicados. Dentro de las técnicas de votación se pueden distinguir las siguientes alternativas: votación estática, votación dinámica por mayoría y votación dinámica con reasignación de votos.

La técnica de *votación estática*, asume que el objeto a proteger ha sido replicado y que para poder acceder a él debe obtenerse un cerrojo del modo apropiado (bien lectura o bien escritura).

1.1.7 AGENTES

En muchas ocasiones cuando los usuarios accesan aplicaciones que usan base de datos en ambientes de redes, ellos disponen de poco tiempo y a veces desconocen como formular las consultas apropiadas para satisfacer sus requerimientos de información. Para ayudar a los usuarios en la atención de sus requerimientos se han creado los sistemas de agentes.

Un *agente* [17], es un sistema de hardware y/o software autónomo (independiente del usuario), el cual interactúa con su entorno (u otros agentes o humanos), guiado por

uno o varios propósitos, su comportamiento es proactivo (reacciona a eventos y a veces se anticipa haciendo propuestas), adaptable (puede enfrentar situaciones novedosas), sociable (se comunica, coopera y/o negocia), su comportamiento es predecible en cierto contexto".

Desde la perspectiva de un agente el ambiente es el conjunto de recursos, propiedades y agentes que se encuentran en el sistema. Wooldridge [59], caracteriza a los ambientes de los agentes con los siguientes criterios:

- *Accesibilidad*, cuando el agente tiene el control completo sobre el ambiente, esto es, que puede accesar todos los componentes que lo forman, se dice que el ambiente es accesible al agente. El dual son ambientes inaccesibles, donde existe al menos un elemento que el agente no puede accesar. Internet es un ambiente no-accesible.
- *Determinismo*, si en un ambiente una acción tiene un efecto único y específico, es decir, no hay incertidumbre acerca del mismo, se dice que el ambiente es determinístico. El caso contrario es cuando alguna acción realizada tiene un resultado que puede ser imprevisto, por ejemplo, al modelar una mesa donde se tienen figuras geométricas que un agente puede mover, al colocar una esfera sobre una mesa, el resultado final puede ser que la esfera caiga al suelo y que no se encuentre en el lugar donde fue colocada originalmente.
- *Ocurrencia de los eventos*, cuando las acciones de un agente se consideran como un episodio sin relación con otros escenarios o tiempos se le llama episódico, por ejemplo un sistema para ordenar una lista de correo electrónico es un ambiente episódico, mientras que uno no episódico es aquel donde se tenga que razonar hacia episodios futuros como la compra de una camioneta, la cual requerirá de aceite, gasolina, pago de tenencia, entre otros, en un tiempo futuro y por lo tanto el costo de mantenimiento debe considerarse al momento de la compra, porque si quien la compra se acaba su dinero en la compra inicial será muy difícil que pueda darle el mantenimiento mínimo requerido.
- *Forma de los cambios*, Pueden ser estático o dinámico. Cuando el ambiente permanece sin cambio excepto por las acciones que realiza el agente, se dice que es *estático*. Es *dinámico* aquel en el que ocurren procesos y operaciones que provocan variaciones fuera del control del agente. El mundo físico es altamente dinámico.

Un agente puede ser implementado en cualquier lenguaje de programación orientada a objetos, por ejemplo java, C++, etc. Es importante notar que existen algunas plataformas que son extensiones de lenguajes orientados a objetos como Java y que implementan algunos modelos de agencia o de agentes, como el modelo creencia-deseo-intención, el cual fue propuesto por Rao/Gorgeff [59]. Una de estas plataformas que ha tenido gran difusión es el *Java Agent Development Framework* (JADE).

Las plataformas [47], se han desarrollado como una necesidad del intercambio de mensajes y conocimiento entre agentes algunos lenguajes y protocolos de comunicación, como el KQML (Knowledge Query Management Language), Manejador de Consultas de un Lenguaje de Conocimiento, el cual incluye protocolos de comunicación entre agentes y que además es posible enriquecerlo con agentes especiales que proveen funciones que se adicionan como servicios. Uno de los resultados más importantes en la estandarización de la teoría de agentes es el desarrollo del *Agent Communication Language* (ACL), el cual es un lenguaje de comunicación entre agentes, que implica la interacción a un nivel más semántico. Las construcciones que estructuran las expresiones de los mensajes ACL se verían muy limitadas por la capacidad de procesamiento automático del lenguaje natural necesaria, de no ser por el uso de ontologías de dominio, como medio para describir semánticamente las relaciones entre las entidades que conforman entornos de aplicación específicos.

El desarrollo de plataformas de agentes ha tenido un gran auge en la última década, encontrando entre otras a las siguientes plataformas:

- *Aglets*, maneja agentes móviles los cuales cuentan con un itinerario de su movilidad y llevan a cabo ejecuciones autónomas y dinámicas sobre sus itinerarios.
- *D'Agents*, es una plataforma simple independiente del sistema de agente móvil. El modelo de navegación esta basado en un simple comando *agente_salta*, este comando puede aparecer en un agente y provocar que su estado y contexto de ejecución sea congelado y transportado a un nodo específico. El Modelo de Comunicación tiene tres comandos: *agente_enviar, agente_recibir* y *agente_reunir.* Cuando un agente quiere migrar a una nueva maquina, éste manda llamar a una simple función *agente_salta*, el cual automáticamente captura la completa

información del estado del agente y la envía al servidor en una máquina destino. El servidor destino empieza apropiándose del ambiente de ejecución, cargando la información del estado y retornándolo a su lugar de origen. Esta plataforma es segura, transparente en la movilidad y maneja la comunicación entre los agentes.

❋ *Voyager*, es una aplicación realizada en Java, su meta es proveer al programador de un espacio para crear programas distribuidos rápidamente, fácilmente, con mucha flexibilidad y extensibilidad. Una de las grandes ventajas de este sistema es que soporta ambas arquitecturas: cliente-servidor y basada en agentes.

❋ *Odyssey*, es una biblioteca de clases Java que permiten al usuario crear sus propias aplicaciones de agentes móviles. Maneja el Protocolo de Transferencia de Agentes Simples.

1.1.8 ONTOLOGIAS

Una *ontología* [17] [29], es una representación de una conceptualización, generalmente descriptiva. En este contexto, es considerada como una colección de conceptos con un árbol como la estructura, formando una taxonomía jerárquica bajo la relación de subconjunto de padres a hijos. Los conceptos son independientes de cualquier lenguaje de comunicación. Cada concepto plantea un significado único y es empleado en un nodo único en la ontología. Un agente no sabe (conoce) todos los conceptos, en otras palabras, la ontología de un agente es incompleta. Por lo general, dos agentes poseen conceptos diferentes, es decir que ellos tienen los árboles diferentes de conocimiento. Sin embargo, con este trabajo se considera que todos los agentes emplean una ontología común relacionada con el contenido de base de datos.

Las ontologías son formas de la representación del conocimiento el cual esta basado en taxonomías, permiten estructurar en forma clara los conceptos de un dominio específico y así determinar su validez en cierto dominio de conocimiento como: electrónica, medicina, mecánica, comercio electrónico, la industria.

Del desarrollo de las ontologías se benefician tanto la comunidad de agentes como la de servicios web (servicios web semánticos) y es uno de los pilares donde se apoya el desarrollo de la Web Semántica. El desarrollo de ontologías es un problema de

representación de conocimiento, orientado fundamentalmente a facilitar la comunicación entre agentes computacionales, entre estos y humanos.

Los trabajos que se han realizado sobre ontologías se clasifican como a continuación se describen [17]:

- *Especificación de ontologías*, se hacen procesos de ingeniería de conocimiento para encontrar los conceptos y la forma de estructurarlos. Algunos trabajos sobre esto son los del proyecto *CyC* (Guha & Lenat 1990), que está orientado a la construcción de una base de conocimiento que contenga el conocimiento humano necesario para hacer inferencias y las especificaciones de FIPA (Foundation for Intelligent Physical Agents).
- *Integración de ontologías*, en donde se pretende integrar ontologías mediante la unión de dos o más ontologías. Sobre esto existe el trabajo de Ontolingua y *Observer*, ontolingua provee diversos tipos de servicios en Internet y sirve para crear, modificar y usar una ontología.
- *Mapeo de ontologías*, se refiere a encontrar equivalencias entre ontologías utilizando información que puede resultar ambigua y que se propone en este trabajo como alternativa de solución al tratamiento de ontologías.
- *Aplicaciones de ontologías*, se tienen en diversos lugares, por ejemplo en la estructuración de conceptos en una biblioteca digital como el caso de la Biblioteca Digital de la Universidad de Michigan o el aprovechamiento de intercambio entre bases de datos realizadas por diferentes grupos de personas.

El uso de ontologías en el desarrollo de sistemas contribuye en una mejora en la calidad del producto final, ya que ellas pueden ayudar a evitar problemas como:

- Inconsistencia entre ontologías implícitas.
- Conflictos entre conceptos ontológicos e implementaciones.
- Conflictos entre ontología de sentido común y conceptos básicos no incluidos en el software.

Una ontología especifica una forma de ver el mundo. Por lo cual, cada ontología incorpora un punto de vista. Una ontología pues, contiene definiciones que provienen del

vocabulario para referirse a un dominio y éstas dependen del lenguaje que se usa para describirlas.

En consecuencia, se pueden señalar algunas características de la ontología típicas:

❖ Pueden existir ontologías múltiples:

El propósito de una ontología es hacer explícito algún punto de vista. Por ello, a veces, se necesita combinar dos o más ontologías. Cada una de ellas introduce conceptualizaciones específicas.

❖ Se puede identificar niveles de abstracción de las ontologías: Estos niveles de generalización proporcionan una topología de ontologías. La idea es caracterizar una red de ontologías usando multiplicidad y abstracción. Como no se puede aspirar a tener una descripción completa del mundo, se puede pensar en una estrategia de construcción gradual de abajo hacia arriba.

Clasificación de las ontologías de acuerdo con su dependencia y relación con una tarea específica:

◆ *Ontologías de Alto Nivel o Genéricas*: Describen conceptos más generales. En relación con los Sistemas de Información, estas ontologías describirían conceptos básicos. Por ejemplo: una teoría describiría partes y todas sus relaciones con la topología.

◆ *Ontologías de Dominio*: Describen un vocabulario relacionado con un dominio genérico. Por ejemplo, podría ser una descripción de datos y entidades relacionados con la sensorización remota con un ambiente urbano.

◆ Ontologías *de Tareas o de Técnicas básicas*: Describen una tarea, actividad o artefacto. Por ejemplo la evaluación de la contaminación sonora en ambientes urbanos o la descripción de características generales de componentes, procesos o funciones.

◆ *Ontologías de Aplicación*: Describen conceptos que dependen tanto de un dominio específico como de una tarea específica y, generalmente son una especialización de ambas.

Otras ontologías son propuestas por:

Federico Fonseca [10], propone que este tipo de ontologías nazcan a partir de una combinación de ontologías de niveles superiores. Ellas representan las necesidades de los usuarios relacionados con una aplicación específica como, por ejemplo, una evaluación de disponibilidad de camarones en la costa de Espíritu Santo (Brasil).

Van Heist [10], propone lo siguiente:

- *Ontologías terminológicas*: Especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado.
- *Ontologías de información*: Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.
- *Ontologías de modelado de conocimiento*: Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

El uso de las ontologías puede aplicarse a:

- *Comunicación*: En modelos normativos, crea la semántica de un sistema y el modelo para extenderlo y transformarlo entre diferentes contextos.
- *Interoperabilidad*: usa ontologías como una inter-lengua y la Ingeniería de sistemas.

El uso de ontologías explícitas en el desarrollo y uso de sistemas de información lleva a los que son llamados Sistemas de Información basados en ontologías [10*].

De igual forma, el conocimiento almacenado en un sistema documental se puede organizar definiendo cada concepto como, por ejemplo: Revista, Artículo, Libro, Autor, etc. También se pueden organizar las relaciones que existen entre un consenso tanto en el significado de cada término, como en una unificación formal de almacenamiento de esa información. Las ontologías se construyen siguiendo esta filosofía, por lo que pueden ser reutilizadas en diferentes dominios y con diferentes fines [10].

1.1.9 CACHE

Una memoria *caché* [35], es un sistema especial de almacenamiento de alta velocidad. Puede ser tanto un área reservada de la memoria principal como un dispositivo de almacenamiento de alta velocidad independiente.

Una memoria *caché* [53], es un conjunto de datos duplicados de otros originales, con la propiedad de que los datos originales son costosos de acceder, normalmente en tiempo, respecto a la copia en el caché.

Cuando se accede por primera vez a un dato, se hace una copia en la caché; los accesos siguientes se realizan a dicha copia, haciendo que el tiempo de acceso aparente al dato sea menor.

La caché es una nueva forma de almacenar datos para obtener así un alto rendimiento conocido como "postrelacional". Como base de datos postrelacional, combina una base de datos de objetos, SQL de alto rendimiento y un potente acceso a datos multidimensionales; estos pueden acceder simultáneamente a los mismos datos. Los datos sólo se describen una vez en un único diccionario de datos integrado que está disponible instantáneamente utilizando todos los métodos de acceso. La caché proporciona mejora en los niveles de rendimiento, escalabilidad, programación rápida y facilidad de uso respecto a la etnología relacional clásica.

La caché también va más allá de las bases de datos tradicionales con la incorporación de un entorno productivo para el desarrollo de sofisticadas aplicaciones basadas en navegador (Web). Por ejemplo la tecnología Caché Server Pages (CSP) permite desarrollar y ejecutar rápidamente páginas Web generadas dinámicamente. Miles de usuarios Web simultáneos pueden acceder a las aplicaciones de bases de datos, incluso con hardware de bajo costo

Las memorias utilizadas como fundamento de los distintos procesos computacionales no se han quedado atrás. Los esfuerzos por desarrollar memorias cada vez más rápidas y de mayor capacidad a un menor costo es una tarea evolutiva con un difícil final a corto plazo, la aplicación de programas (sistemas operativos de aplicación general y específica) y la reciente demanda de estas en el medio computacional.

Una memoria Caché tiene varias ventajas, entre las cuales se encuentran [46]:

- Flexibilidad; los modos de acceso a los datos de Caché (Objetos, SQL y Directo) se pueden utilizar concurrentemente sobre los mismos datos.
- Menos trabajo; la Arquitectura de Datos Unificada de Caché describe automáticamente los datos como objetos y como tablas con una sola definición.
- Rendimiento; utilizando un modelo de datos multidimensional eficaz con técnicas de almacenamiento mediante arreglos en lugar de un laberinto inmanejable de tablas bidimensionales, el acceso a los datos y las actualizaciones se realizan con menos operaciones E/S en disco. La E/S reducida significa que las aplicaciones se ejecutarán más rápidamente.
- Escalabilidad; el modelo de datos multidimensional transaccional permite a las aplicaciones basadas en Caché escalarse a muchos miles de usuarios sin sacrificar el alto rendimiento.
- El uso del bloqueo lógico en la Caché; para las actualizaciones en lugar de bloquear las páginas físicas contribuye también de forma importante a la concurrencia, como lo es su sofisticado almacenamiento de datos en memoria intermedia caché a través de las redes.
- Desarrollo rápido de la Caché; permite desarrollar más rápidamente porque la estructura de datos proporciona un almacenamiento natural y de fácil comprensión de datos complejos y no requiere gran cantidad de declaraciones ni definiciones largas o complicadas.
- Rentabilidad; comparadas con las aplicaciones relacionales de tamaño similar, las aplicaciones basadas en la Caché requieren mucho menos hardware y ningún administrador de bases de datos.
- Consultas radicalmente más rápidas; mediante la utilización de técnicas de algoritmo de mapeo (bitmapping) transaccional, los usuarios pueden conseguir búsquedas extremadamente rápidas sobre grandes bases de datos (buscar en millones de registros en una fracción de segundo) en sistemas usados principalmente para proceso transaccional.

En sistemas con miles de usuarios, la reducción de conflictos entre procesos concurrentes es crítica para conseguir alto rendimiento. Uno de los mayores conflictos se

da entre transacciones que intentan acceder a los mismos datos. Los procesos de la Caché no bloquean páginas enteras de datos al ejecutar actualizaciones. En su lugar, dado que las transacciones requieren acceso frecuente o cambios a pequeñas cantidades de datos, en la Caché el bloqueo de la base de datos se realiza a nivel lógico. Los conflictos de la base de datos se reducen aún más utilizando operaciones individuales de suma y resta, que no requieren bloqueo.

Con la Caché, las transacciones individuales se ejecutan más rápidamente, y se pueden ejecutar más transacciones simultáneamente.

Como los datos de la Caché tienen de forma inherente una longitud variable, y se almacenan en un arreglo de datos, la Caché suele necesitar menos de la mitad del espacio que necesita una base de datos relacional. Además, de reducir los requerimientos de disco, el almacenamiento de datos compacto mejora el rendimiento porque se pueden leer y escribir más datos con una sóla operación de E/S, y los datos se pueden almacenar en memoria intermedia caché de forma más eficiente.

En el diseño de la memoria caché se deben considerar varios factores que influyen directamente en el rendimiento de la memoria y por lo tanto su objetivo de aumentar la velocidad de respuesta. Estos factores son de: ubicación, extracción, reemplazo, escritura y el tamaño de la caché y de sus bloques.

En informática [42], se utilizan sistemas caché, que almacenan temporalmente la información utilizada, porque está comprobado que esa misma información suele volver a necesitarse. Este truco tiene cierto sentido cuando el sistema de almacenamiento temporal es mucho más ágil (rápido de consultar) que la fuente original que contenía la información. Por ejemplo, muchos sistemas operativos almacenan en memoria RAM la información obtenida del disco duro; hay bastantes posibilidades de que posteriormente vuelva a necesitarse esa misma información, y ya no será necesario volver a consultar el disco duro: podrá obtenerse directamente de la "copia" que se guarda en memoria RAM, con la ventaja de que es mucho más rápido consultar la memoria RAM que el disco.

Otro ejemplo: quizá hayas visto que algunos microprocesadores anuncian como una de sus características que disponen de memoria "caché". En esa pequeña memoria se va guardando una copia de la información recientemente obtenida de la memoria RAM, que -proporcionalmente- es más lenta. De esa forma se agiliza el trabajo: puesto que

muchas veces el microprocesador necesita volver a consultar los mismos datos de la memoria RAM, así puede obtenerlos más rápidamente de sus propios circuitos.

En Internet ocurre algo muy parecido. Muchas páginas son consultadas repetidamente al cabo del día; el proveedor de conexión puede instalar un sistema que actúe como memoria temporal intermedia para guardar una copia de las páginas que van visitando los usuarios. Así, cuando otro usuario (o el mismo usuario) quiera volver a consultar esa página, el proveedor de Internet ya no necesita solicitarla al servidor que la aloja en otro ordenador remoto: puede pasarle al usuario los datos (una "copia") que están almacenados en alguno de sus propios ordenadores. A este sistema se le suele denominar Proxy-caché o Proxy-transparente.

Existen otras páginas que el servidor remoto "construye" en el momento de servirlas. Por ejemplo, cuando consultas un buscador, la página de resultados se prepara en el momento, dependiendo de la consulta concreta. También hay otros sistemas de páginas no estáticas, como por ejemplo las páginas de los periódicos que pueden cambiar su contenido varias veces al día. Existen páginas que en todo momento ofrecen la situación de la bolsa, y esas cambian continuamente (minuto a minuto) a lo largo de la jornada bursátil.

1.1.10 METODOLOGIA RUP (Rational Unified Process)

En este apartado se describe la metodología RUP (Rational Unified Process) la cual sirve para guiar el desarrollo de sistemas de información.

El *Proceso Unificado Racional (RUP)* [52]: es un proceso iterativo e incremental de Ingeniería de Software el cual designa tareas y responsabilidades.

La metodología RUP, al igual que otras, cuenta con el objetivo de asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con un costo y un calendario predecible.

Las características de RUP son:

❄ *Guiado o Manejado por Casos de Uso*: Remplazan la antigua especificación racional tradicional.

❄ *Apoyado en una arquitectura principal*: Es como una radiografía del sistema que se está desarrollando suficientemente completa para que todos los involucrados tengan idea de lo que se está construyendo. Iterativo e Incremental. Divide al proceso en cuatro fases dentro de las cuales se realiza varias iteraciones.

❄ *Se basa en componentes definidas:* Divide el sistema en componentes con interfaces bien definidas que posteriormente serán ensamblados para generar el sistema.

❄ *Utilización del lenguaje modelado UML* para todos los modelos de diagramas que se realicen.

❄ *Proceso integrado:* Soporta flujos individuales de trabajo, secuencia de actividades realizadas por los diferentes roles, así como la relación entre los mismos, que nos producen unos resultados observables.

Dentro de toda metodología, es importante no perder de vista los elementos, que ayudan a una mejor fundamentación del trabajo, estos son:

a) *Roles* ¿Quién?
b) *Actividades* ¿Cómo?
c) *Productos* ¿Qué?
d) *Flujos de trabajo* ¿cuando?, hay dos tipos: proceso y apoyo.

RUP divide el proceso de desarrollo en ciclos, donde se obtiene un producto. Cada ciclo se divide en cuatro Fases: Concepción, Elaboración, Construcción, y Transición. Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones.

✂ *Fase de concepción:* nos preguntamos ¿Cuál es el objetivo? ¿Es factible? ¿Lo construimos o lo compramos? ¿Cuánto va a costar?
En esta fase se establece la oportunidad y alcance del proyecto. Se identifican todas las entidades externas con las que se trata (actores) y se define la interacción en un alto nivel de abstracción: se deben identificar todos los casos de uso, y se deben describir algunos en detalle. La oportunidad del negocio incluye: definir los

criterios de éxito, identificación de riesgos, estimación de recursos necesarios, y plan de las fases.

- *Fase de elaboración:* Es analizar el dominio del problema, en esta fase se construye un prototipo de la arquitectura. Definir y validar una arquitectura estable. Se hace un refinamiento de la revisión del sistema, basándose en nueva información obtenida durante esta fase, se establece una sólida comprensión de los casos de uso más críticos que definen las decisiones arquitectónicas y de planificación. Creación de los planes de desarrollo detallados para las iteraciones de la fase de construcción.

- *Fase de construcción:* Todos los componentes, características y requisitos que no hayan sido hechos hasta ahora, han de ser implantadas, integradas.

- *Fase de transición:* es poner el producto en manos del usuario.

En la ejecución de los planes de implantación, se finalizan los manuales de usuario y mantenimiento.

Para llevar a cabo las pruebas del sistema en el entorno de explotaciones toman en cuenta los siguientes criterios:

- Creación de una revisión del sistema.
- Validación del sistema por los usuarios.
- Ajuste fino del sistema según la validación con el usuario.
- Se facilita la transición del sistema al personal de mantenimiento.
- Se pone el producto a disposición del usuario final.

En la siguiente figura 2, se describe de manera general la metodología RUP:

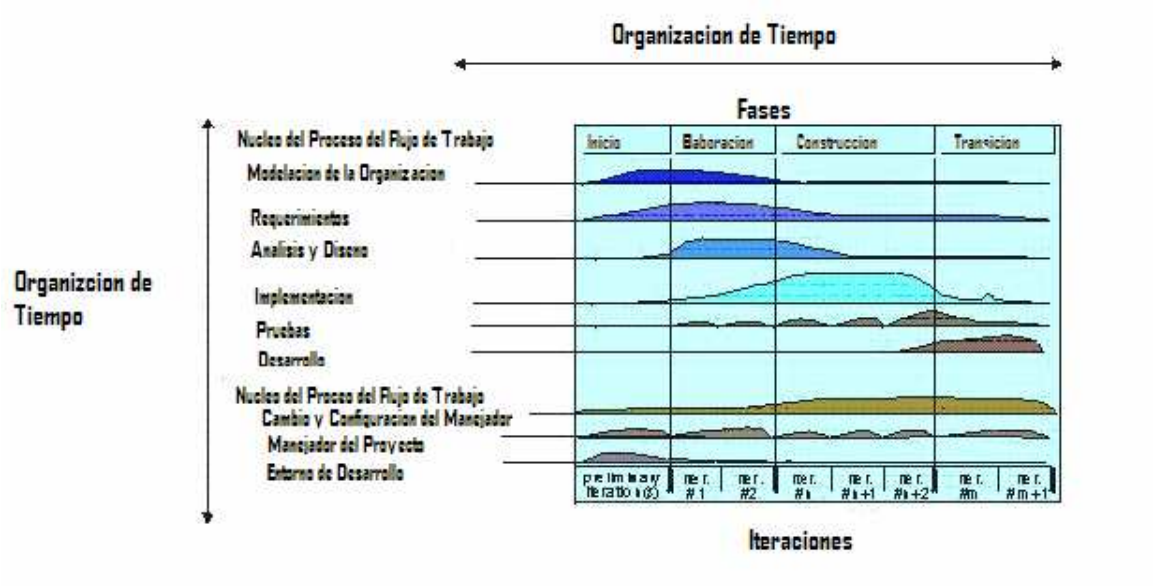


Figura 2. Metodología RUP.

La metodología RUP por ser un proceso iterativo como incremental, permite tener mini proyectos antes de terminar todo el software, con esto es posible presentar al cliente un avance de proyecto para que éste pueda evaluar como se está trabajando.

Es necesario que las aplicaciones cumplan con normas de calidad ya que también se maneja baja esta modalidad, como por ejemplo se pueden mencionar las transacciones bancarias y todas estas transacciones requieren que sean seguras y confiables.

RUP pretende implementar las mejores prácticas en ingeniería de software, con el objetivo de asegurar la calidad del software de producción, dentro de plazos y presupuestos predecibles, tomando en cuenta los criterios que se describen a continuación:

- *Desarrollo iterativo*: Permite una comprensión creciente de los requerimientos, a la vez que se va haciendo crecer el sistema. RUP sigue un modelo iterativo que aborda primero las tareas más riesgosas. Así se logra reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente.

- *Administración de requerimientos*: RUP describe cómo obtener los requerimientos, cómo organizarlos, cómo documentar los requerimientos de funcionalidad y

restricciones, cómo rastrear y documentar las decisiones, y cómo captar y comunicar los requerimientos de la organización.

- *Arquitecturas basadas en componentes*: El proceso se basa en diseñar antes una arquitectura como base ejecutable del sistema. Esta arquitectura debe ser: flexible, fácil de modificar, intuitivamente comprensible, y debe promover la reutilización de componentes.

- *Modelamiento visual*: RUP propone un modelamiento visual de la estructura y el comportamiento de la arquitectura y los componentes. En este esquema, los bloques de construcción deben ocultar detalles, permitir la comunicación con el equipo de desarrollo, y permitir analizar la consistencia entre los componentes, el diseño y la implementación.

- *Verificación de la calidad del software*: No sólo la funcionalidad es esencial, también el rendimiento y la confiabilidad. RUP ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifiquen estas cualidades.

- *Control de cambios*: Los cambios son inevitables, pero es necesario evaluar si se justifican y también hay que rastrear su impacto. RUP indica como controlar, rastrear y monitorear los cambios dentro del proceso iterativo de desarrollo.

1.1.11 METODOLOGIA LGS (CVDSI)

Esta metodología tiene por objetivo enlazar las actividades que intervienen en el Ciclo de Vida del Desarrollo de un Sistema de Información (CVDSI). En la figura 3, se representa la relación entre las fases del CVDSI [12]. En el esquema se muestra cómo la metodología va enlazando las actividades, donde algunas pueden ser parte de la fase de análisis, diseño construcción e implantación, además establece una continúa retroalimentación entre cada etapa, puesto que esto es fundamental para hacer las debidas

correcciones y obtener el mejor resultado, así mismo, nunca se consideran como actividades separadas sino como complemento para lograr un objetivo en común.

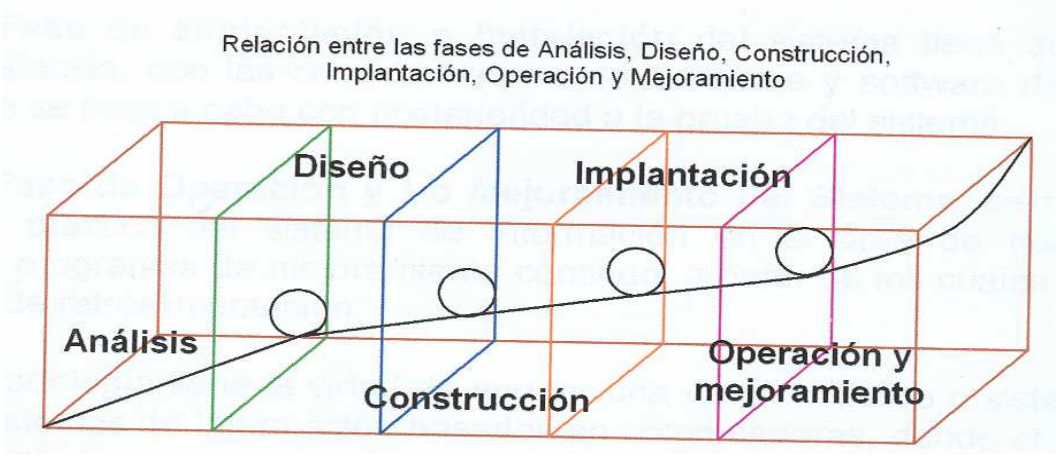


Figura 3. Esquema con la relación entre las fases para el desarrollo del CVDSI.

Ahora se describen brevemente cada una de las fases que conforman la metodología.

La fase de *Análisis*, es la etapa que sirve para recopilar e interpretar los procesos y sus datos, identificar y diagnosticar problemas y utilizar esta información, con el fin de proponer una solución a la problemática. Representa la etapa fundamental para conocer e interpretar adecuadamente los requisitos de los usuarios finales, quienes en muchas ocasiones no son tomados en cuenta y el sistema no responde a sus necesidades o solo responde parcialmente, así como también conocer el contexto o medio ambiente en el que se desenvuelven para una mejor competitividad.

La fase de *Diseño*, es el proceso de conceptualización y estructura detallado de un nuevo sistema en una empresa o institución para reemplazar o complementar a uno existente con algún problema. La fase de diseño, es también una parte fundamental para lograr un beneficio óptimo ya que antes de empezar a escribir programas se planea la estructura, los procesos, las entradas y las salidas para el sistema de información.

La fase de *Construcción o Programación*, es donde se realiza la escritura o desarrollo de programas o procesos en la computadora y es normalmente la actividad individual más operativa en el desarrollo de un sistema de información basado en computadoras.

La fase de *Implantación o Instalación,* tiene que ver, en primera instancia con las consideraciones de hardware y software del mismo, usualmente, se lleva a cabo con posterioridad a las pruebas del sistema.

La fase de *Operación y /o Mejoramiento,* se refiere a la puesta en práctica del sistema de información en el área de trabajo y se establecen programas de mejoramiento continuo, a partir de los cuales se genera el proceso de retroalimentación.

1.1.12 METODOLOGÍA PROPUESTA

Como resultado del estudio de estas dos metodologías, antes expuestas, que me aportaron una visión holística. A continuación se presenta el marco metodológico utilizado para el desarrollo del proyecto de esta tesis. El marco metodológico describe las actividades que será necesario desarrollar y el orden en que deben llevarse a cabo para lograr el fin propuesto. En cada una de las actividades se hace uso de alguna técnica y herramienta que permite la mejor obtención de resultados para lograr un objetivo específico ó meta, esta se muestra en la siguiente tabla 1.

Actividades ¿Qué hacer?	Técnicas ¿Cómo hacer?	Herramientas ¿Con qué hacer?	Metas ¿Qué obtener?
Definir: Marco Conceptual	Pirámide Conceptual	Procesador de palabras	Pirámide Conceptual
Analizar la situación actual	Acopio de Información Observación	Procesador de palabras, Internet	Estudiar la Concurrencia de los Datos
Definir Objetivo General, Objetivos Específicos y la Justificación	Organizarla para su análisis	Procesador de palabras	Objetivo General, Objetivos Específicos y la Justificación del Proyecto de Tesis
Diseño	Diagramas de caso de uso, de secuencias, flujo de datos, clases.	Procesador de textos, Erwin, Enterprise Architect.ure	Diseño conceptual del Sistema de Control de Concurrencia a través de agentes.
Construcción del modelo. Metodología : Análisis Diseño Construcción	Diagrama de Flujo de Datos. Casos de Uso. Diseño de entradas, procesos de salidas. Modelo E-R	Procesador de textos. Lenguajes de programación: Java. Servidor: Tomcat	Construcción del Sistema de Control de Concurrencia
Implantación del Prototipo del sistema.	Instalarlo en un equipo. Realizar pruebas al sistema. Operación del Sistema.	Sistema con XP	Condiciones para operar el sistema.
Pruebas y Verificación	Planear tres casos	Con una Pentium IV	Resultados de las pruebas
Conclusiones	Comparar objetivos propuestos con resultados obtenidos	Interfase del sistema. Graficador.	Identificación de trabajos futuros. Conclusiones.
Redacción de la tesis	Investigación, Recopilación de Información, Técnicas de Redacción de documentos	Resultado del Sistema de Control de Concurrencia. Procesador de texto.	Documento de Tesis.

Tabla 1. Marco Metodológico para el desarrollo del proyecto de Tesis.

A continuación se presenta la figura 4, que muestra de manera esquemática la metodología propuesta, indicando al pie el capítulo en el que se desarrolla cada una de las etapas.

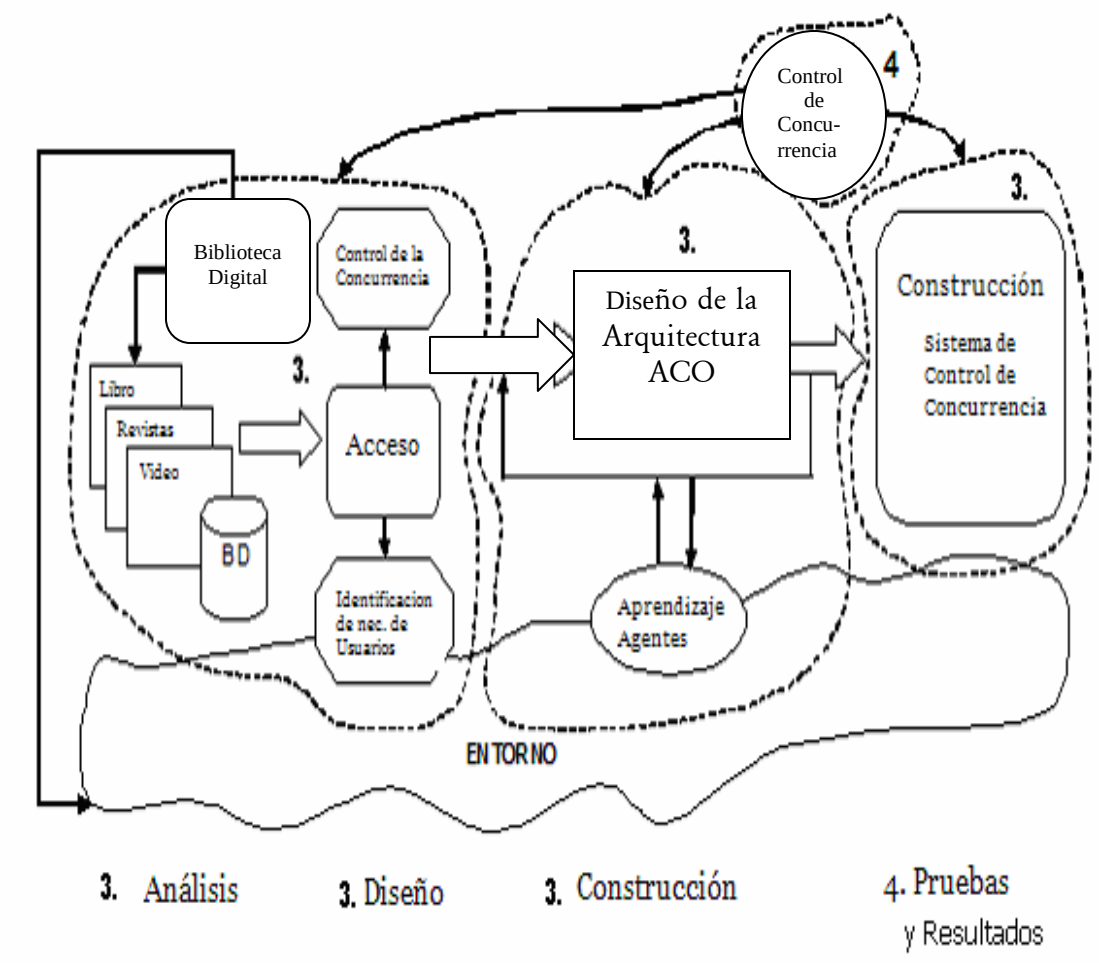


Figura 4. Metodología propuesta.

Análisis, consiste en estudiar y analizar la problemática que existe en los sistemas de Biblioteca Cliente/Servidor a través de una conexión constante, y en donde se tienen conectados muchos usuarios a la vez, y por otro lado analizar la problemática que existe con arquitectura Cliente/Servidor a través de la tecnología de Servlets, en donde existe una cantidad de usuarios conectados.

En la segunda etapa *Diseño,* en base al estudio realizado con respecto a las arquitecturas Cliente/Servidor a través de una conexión constante y con el uso de la tecnología de servlets; se crea una propuesta para mejorar el tiempo de respuesta la cual es aplicado a una arquitectura Cliente/servidor a través de agentes, una memoria caché y una ontología ACO.

Construcción, en esta etapa se lleva a cabo la construcción del Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes, con el análisis y el diseño hecho.

Pruebas, en esta etapa se realizan pruebas a la arquitectura propuesta llamada ACO (Agentes, Caché, Ontología), y se realiza un comparativo mediante la propuesta de tres casos de estudio.

1.2 ANTECEDENTES

Con el auge del uso de redes de computadoras se crea el modelo Cliente / Servidor, el cual desarrolla sistemas de información, en donde se ejecutan dos sistemas diferentes uno hace el cliente y el otro el servidor, es posible que tanto el cliente como el servidor se encuentren en un único sistema. Generalmente un servidor proporciona servicios a varios sistemas clientes, aunque puede haber un único sistema. A menudo el cliente es un sistema de escritorio conectado en red. Cada vez que el usuario necesite recuperar o almacenar información, la parte cliente de la aplicación ejecuta una solicitud, que se envía (generalmente por una red) al servidor. El servidor ejecuta entonces la solicitud y devuelve la información al cliente.

A continuación se citan algunos ejemplos de sistemas con el modelo cliente/servidor.

Se encuentran puntos de venta en supermercados, ATM= Cajeros automáticos, donde ha habido inversiones económicas para buscar la mejora.

Control Escolar, Bibliotecas Digitales a estos sistemas no se le han dado recursos para atenderlos y por tanto en algunos casos la ejecución local de las aplicaciones puede mitigar los cuellos de botella de la red.

Por mencionar otro ejemplo de estos sistemas es UNICORN (Sistema ínter bibliotecario) el cual opera en la Biblioteca Nacional de Ciencia y Tecnología del Instituto Politécnico Nacional; este sistema almacena el material bibliográfico, así como también las Tesis, Videos, Revistas, Mapas de la Biblioteca Nacional Víctor Bravo Ahuja. Trabajando con este sistema se detectó que tiene algunas dificultades en la conexión de los usuarios, ya que a medida que se van conectando una cantidad de clientes, llega un momento en que el sistema ya no permite la conexión de otros usuarios, lo cual provoca que un usuario con deseos de conectarse tiene que realizar una cantidad de intentos para tener una conexión en el sistema, y esto es molesto para los usuarios, ya que provoca perdida de tiempo. Otro problema que se detectó, fue que entre mayor sea la cantidad de personas conectados al sistema, el rendimiento disminuye, esto quiere decir, que el tiempo de respuesta es muy tardado, por lo que un usuario tiene que esperar una determinada cantidad de tiempo para obtener su respuesta, lo cuál hace que el usuario se impaciente y pierda tiempo.

Independientemente de cual sea el caso, en los sistemas cliente servidor, se han detectado los siguientes problemas:

En el Acceso Concurrente:

a) Cuando el *número de usuarios es grande* el tiempo de respuesta es lento, además que impide la conexión de usuarios "nuevos".

b) Hay problemas de conexión de algunos usuarios debido a que otros *usuarios conectados se encuentran "pensando"* su consulta, lo cual hace que consuman tiempo que bien podría ser aprovechado para que otro usuario se conectará al sistema.

c) El usuario al quedarse pensando en lo que hará en el sistema de la base de datos, provoca que consuma tiempo y hace que los otros usuarios esperen más tiempo para obtener la respuesta de sus tareas.

d) Falta de conocimiento o expresividad para realizar las consultas.

e) Muchas veces las *mismas consultas* se repiten entre diferentes usuarios.

1.3 DESCRIPCIÓN DEL PROBLEMA

El problema a solucionar en esta tesis se enfoca al control de acceso concurrente a una Base de Datos en la cual se tienen muchas operaciones de consulta, generadas por el acceso de cientos de usuarios concurrentemente, por ejemplo este tipo de tareas se ven más reflejados en Bibliotecas Digitales, Datawarehouse, grandes bancos de información.

En este trabajo se aborda el caso de una biblioteca digital, la cual, recibe poca atención por parte de los investigadores de base de datos ya que no es una aplicación donde se maneje dinero. En la biblioteca digital, se realizan transacciones sencillas, los usuarios que acceden al sistema simultáneamente deben esperar por su respuesta debido al volumen de solicitudes que existen de manera concurrente y conforme aumenta el número de usuarios el tiempo de respuesta generalmente se hace más lento. Otra consecuencia del gran número de usuarios, es al momento de conectarse al sistema, ya que un usuario tiene que realizar varios intentos antes de ingresar.

En varias ocasiones para un usuario que ha logrado su conexión se presenta otro problema, la dificultad para expresar sus consultas debido a que en muchas ocasiones desconocen las palabras clave para formular sus preguntas y como se quedan "pensando" utilizan tiempo de conexión que en el modelo cliente/servidor impide a otro usuario conectarse.

1.4 OBJETIVO GENERAL

Desarrollar un Sistema de Control de Concurrencia a una base de datos Mediante Agentes, en donde se reciban varias solicitudes de los usuarios simultáneamente.

1.5 OBJETIVOS ESPECIFICOS

- Investigar el uso y aplicaciones de los Agentes
- Analizar el problema de la concurrencia de los datos.
- Estudiar los problemas de las desconexiones en un Sistema Manejador de Bases de Datos.

- Detectar las solicitudes de los usuarios, es decir se examinan y solamente se envían al Sistema Manejador de Bases de Datos aquellas solicitudes novedosas para minimizar el tiempo de respuesta a la Base de Datos.
- Investigar el manejo de la caché y el uso de las ontologías.
- Probar el sistema propuesto en base a casos de uso.

1.6 JUSTIFICACIÓN

Las bases de datos de hoy en día y las redes computacionales luchan por la necesidad de contar con una mejor utilización de sus recursos. La información es el recurso más importante de ambos, por eso es necesario involucrarle cierto grado de interés.

La concurrencia de los datos permite que los usuarios accesen a la base de datos en forma sincronizada. Mientras mejor sea el control de la concurrencia de los datos será más eficiente su acceso.

El control de la concurrencia ha sido tratado por otras alternativas. Sin embargo, en el estudio hecho en este trabajo no existe evidencia de que se hayan utilizado agentes.

El uso de Agentes permite que la sincronización de los datos sea eficiente al momento de ser accesado por los usuarios, además de no permitir que dos o más transacciones se bloqueen entre ellas.

La razón más importante por la cuál se decidió enfocar el proyecto de tesis en resolver este tipo de problemas fue el hecho de utilizar las características de los agentes junto con la caché, para mejorar el control de la concurrencia de los datos permitiendo que los procesos puedan sincronizarse, y para ello es necesario disponer de servicios los cuales puedan permitir bloquear o suspender la ejecución de un proceso.

Los sistemas operativos para evitar el problema de la concurrencia utilizan los mecanismos que son: Señales, Tuberías, Semáforos, Mutex y variables condicionales además del Paso de mensajes.

También se incluye el deseo de aplicar los conocimientos existentes sobre Agentes, Caché y Ontología para el control de la concurrencia de los datos, tratando de proporcionar con este proyecto nuevos resultados que pudieran ser relevantes al área, los Agentes se han utilizado para darle solución a numerosos problemas, pero hasta hoy no se han aplicado a bases de datos que manejan un volumen considerado de datos.

Los beneficios que se pretenden brindar con este sistema se enumeran a continuación:

- Mejorar los Conflictos de la concurrencia de varios usuarios cuando accesan a una Base de Datos.
- Disminuir la latencia de Sincronización de los Datos.
- Disminuir la latencia de Comunicación de Mensajes, entre el Servidor y el Usuario.
- Tener un mejor aprovechamiento de la CPU y la memoria Caché

1.7 PROPUESTA

El trabajo se enfoca al control de acceso concurrente a una Base de Datos en la cual se tienen muchas operaciones de consulta, generadas por el acceso de cientos de usuarios concurrentemente, para ello se analiza una Base de Datos de Biblioteca Digital.

Se propone desarrollar un Sistema de Control de Concurrencia mediante agentes, el cual cuenta con la siguiente arquitectura:

a) Un conjunto de Agentes que reciben las solicitudes y hacen sugerencias en base al perfil del usuario. Estos agentes tienen la función de dar alternativas de respuestas a los usuarios de acuerdo al tema que ellos desean consultar y los cuales ya están determinados en el perfil que cada usuario ha determinado. En este caso el agente en base al perfil tomará las consultas y procede a realizar la búsqueda de la información que se desea o en su caso sugiere las posibles respuestas que se

acerquen lo más posible al tema que el usuario desea obtener. Cada agente-usuario mantiene una conexión continua con el usuario y solamente envía al SMBD las consultas "nuevas" de las cuales se guarda la respuesta (cuando el tamaño es conveniente, en este caso se estableció que sea un peso de 14 KB). El agente-usuario únicamente mandará guardar aquellas consultas actualizadas hechas por los usuarios y que además no tengan un peso mayor a lo que se mencionó anteriormente, ya que esto permitirá no saturar la caché de datos que tengan poca importancia para los que realizan una consulta al sistema.

b) Una memoria caché de las solicitudes frecuentemente realizadas al sistema por el usuario. En esta memoria el agente-usuario buscará las consultas preferentes que los usuarios deseen consultar y obtener en un momento dado sus resultados. Una vez que las consultas se van almacenando en la "caché", estas se van marcando. Para esta aplicación se empleará una caché para 10000 registros, al momento en que la caché se encuentre llena, será posible eliminar las consultas que fueron las menos frecuentadas para poder liberar espacio y seguir almacenando otros datos.

c) La utilización de la Ontología la cual sirve para la organización jerárquica de los conceptos que se encuentran en el sistema. Al utilizar la ontología y el conjunto de patrones de solicitudes a la base de datos es posible generar consultas que pueden resultar de interés para un usuario, con lo cual, se intenta brindarle ayuda al usuario en la formulación de sus consultas. A través de la ontología tanto el agente como el perfil del usuario, se nutren de los nuevos datos que sean de importancia para ellos, y de esta manera los agentes proporcionan sugerencias a los usuarios en referencia a los temas que sean de su interés.

En este capítulo, se presentó el marco conceptual, donde se desataca la importancia de los conceptos fundamentales involucrados en la tesis, los cuales están representados en la

pirámide conceptual, y también se expresa el problema a resolver en esté trabajo, dando la justificación del mismo, para ello se establecieron el objetivo general y los objetivos específicos

En el capítulo dos, se exponen los trabajos relacionados con la propuesta de esta tesis.

CAPÍTULO 2
TRABAJOS RELACIONADOS

En este capítulo se presentan trabajos relacionados con el acceso concurrente a una base de datos. El interés de este estudio radica en encontrar una solución más a la concurrencia de base de datos para mejorar el tiempo de acceso a los sistemas mediante la aplicación de agentes de software, ontologías y memoria caché. En primer lugar se mencionan algunos trabajos relacionados con las Bases de Datos, así como también las diferentes tecnologías que se están aplicando sobre ellas, tales como las Bases de Datos Semántica y las Bases de Datos Inteligentes. En donde las Bases de Datos deben ser semánticas para así poder decir que es una Base de datos Inteligente, aquí los Agentes juegan un papel importante debido a la cantidad de información que se maneja en ellas y el gran número de usuarios que accesan al sistema simultáneamente, el tiempo de respuesta de los sistemas aumenta a medida que se conectan más usuarios. En esta situación los resultados que obtienen los usuarios son poco relevantes para sus consultas y las omiten, por tanto para evitar este tipo de situaciones el agente adquiere conocimientos a través de una ontología la cual estará nutriendo en todo momento al agente para realizar las consultas en forma automática.

2.1 BASES DE DATOS Y AGENTES

Hasta la fecha se han realizado diversos estudios y aplicaciones en el área de Base de Datos. Desde el inicio de los trabajos en las ciencias de la computación en los años 50s [14], se buscó la forma de tener una herramienta para el manejo generalizado de los datos y la información. La aparición de los primeros Sistemas de Manejo de Base de Datos ocurre a finales de 60s, desde entonces se le han incorporado diversas mejoras, entre ellas se encuentran la solución de algunos problemas de concurrencia [24], Modelos Semánticos de Base de Datos [26], Bases de Datos Inteligentes [18].

Como comenta Stuart Middleton [22], los mayores trabajos de investigación y desarrollo se han dado en aplicaciones comerciales, en bancos y sistemas de reservaciones de aerolíneas, entre otros. Sin embargo existen muchas aplicaciones que cuentan con características propias y donde se ha dejado de lado el estudio de las bases de datos y sus problemas de acceso y recuperación de información, ejemplo de este tipo de aplicaciones son los sistemas de diseño y manufactura asistida por computadora (CAD/CAM), las bibliotecas digitales y sistemas federales de administración escolar.

A principios de los 90s, se inició el desarrollo de la tecnología de Agentes, fundamentada principalmente en los trabajos de la Inteligencia Artificial, los procesos asíncronos, y la computación distribuida.

A los agentes se les ha encontrado útiles en el procesamiento de información de Base de Datos, al permitirle a un usuario la personalización de sus consultas. Los agentes son importantes para reducir la sobrecarga de información. Los "*usuarios necesitan información útil, no un montón de datos*" [34]. Estos agentes necesitan comprender los diferentes formatos de los datos para registrar las bases de datos internas y externas en busca de información. Los agentes avanzados de administración de información filtrarán, condensarán y presentarán la información siguiendo una serie de reglas que les especifique el usuario.

Agentes inteligentes Notificadores [39], avisan a los usuarios de Bases de Datos sobre inclusiones / alteraciones de datos acerca de sus temas de interés. Estos temas deben ser previamente seleccionados por el usuario. En el Banco de Datos del CNCT (Centro Nacional de Competencia en Ciencia y Tecnología), existen o tienen usuarios registrados para recibir informaciones sobre las actualizaciones del banco antes citado. Cualquier interesado puede recibir informaciones sobre este banco.

Existen sistemas, los cuales hoy en día llevan a cabo diferentes tipos de tareas, estos manejan una relación de agentes con diferentes aplicaciones de aprendizajes. A continuación se mencionan:

APE (Sistema de Aprendizaje Automático) [19], es un sistema de aprendizaje, que utiliza técnicas de aprendizaje automático para aprender de los hábitos del usuario de

forma que, posteriormente, pueda llevar a cabo tareas repetitivas en lugar de hacerlo el propio usuario por sí mismo. El objetivo que persigue APE es, por un lado, el diseñar un agente de asistencia capaz de automatizar tareas repetitivas con un mínimo número de intervenciones por parte del usuario, utilizando técnicas de Programación por Demostración para poder llevar a cabo acciones repetitivas complejas y que interfieran lo menos posible al usuario, es decir, ofreciéndole solamente opciones adecuadas en el momento oportuno. Para lograr su cometido APE utiliza técnicas de aprendizaje automático, permitiéndole aprender rápida y eficientemente a realizar sugerencias al usuario así como saber en qué momento debe de realizar dichas sugerencias.

(CSCW) Cooperativos Soportados por Ordenador, Baecker [7], la define informalmente como: "la actividad de coordinación asistida por ordenador análoga a la resolución de problemas y de comunicación llevada a cabo por un grupo de colaboradores individuales". El énfasis principal de CSCW es el desarrollo de herramientas software (y hardware) como soporte de trabajos humanos colaborativos (para describir tales herramientas se ha acuñado el término *groupware*). Diversos autores han propuesto el uso de la tecnología de agente en groupware. Por ejemplo, Chang [8], sugiere sistemas en que los humanos colaboran no solamente con otros humanos, sino también con agentes artificiales. Para más detalles sobre CSCW se remite al conjunto de artículos editados por Baecker [7] y al artículo de Greif [13].

Como señala Gerhard [12], las plataformas modernas de cómputo y los ambientes de información son distribuidos, bastos, abiertos y heterogéneos. En particular las aplicaciones de bases de datos se vuelven complejas. Las computadoras donde residan las bases de datos objetivo para satisfacer una consulta deberán comportarse más como "individuos" o sea, agentes, que como partes pasivas de un macrosistema de información que las explote. Las propuestas del área de multiagentes pueden proporcionar una nueva manera de abordar la interacción para bases de datos.

Sociedad de agentes comunicantes [15]*:* Es un sistema de múltiples agentes éstos se desenvuelven con funciones bien definidas. Dentro de un marco establecido de protocolos de comunicación, son capaces de interactuar para lograr un fin en común. La interacción

de agentes descansa en la *negociación*, resolviéndose los problemas de manera descentralizada. Los agentes [43], que constituyan esta sociedad deben estar soportados bajo el estándar FIPA para asegurar la interoperabilidad con otros agentes.

Hay algunos proyectos orientados a hacer la recuperación de información mediante agentes, tales como las Bibliotecas Digitales.

Un interesante ámbito de aplicación de los agentes de recuperación de información son las Bibliotecas. Las colecciones de información publicados almacenados en formato electrónico ha aumentado considerablemente, por lo cual las bibliotecas deben incorporar este formato como una alternativa a las publicaciones tradicionales.

Dentro de este ámbito [32], los agentes de recuperación de información pueden resultar de gran utilidad como asistentes de los bibliotecarios, manejando los grandes volúmenes de material electrónico almacenado. En base a una solicitud, el agente puede recurrir a estrategias para seleccionar las bases de datos donde buscar y como categorizar los documentos encontrados, además pueden colaborar con el ínter bibliotecario estas son:

- ➲ Realizar consultas con distintas estrategias.
- ➲ Creando perfiles de usuarios, con base en sus áreas temáticas o preferencias de material.
- ➲ Manteniendo estadísticas de solicitudes/búsqueda.

Los agentes consumidores son responsables de representar los intereses del usuario, mantienen los modelos del usuario y los utilizan para asistirlo proporcionándole la información que le interesa.

A continuación se enumeran los sistemas que proporcionan este tipo de servicios a los usuarios [34]:

- ✪ *El sistema ZDL* (The Zuno Digital Library), actúa como un filtro de información y recolector de información, es decir que esta es una librería que maneja una enorme colección de datos. Por ejemplo permite que el usuario tenga una vista organizada de una fuente de colección de datos pero desorganizados tal como se maneja en la WWW, entonces los agentes para esto cubren tres roles importantes:

Consumidor; el cual se encarga de representar al usuario final que utiliza el sistema.

Productor; este representa un proveedor de contenido, el cual se apropia de la información que el consumidor adquiere.

Facilitador; funciona como puente entre el consumidor y proveedor para llevar a cabo su tarea encomendada.

✪ *Maxims:* Describe un agente de filtrado para correo electrónico, el programa aprende a priorizar, borrar, devolver, cortar, archivar mensajes de correo.

✪ *Newt:* Este es un programa que filtra noticias de Internet, toma una serie de artículos y de acuerdo a las preferencias del usuario selecciona los que considera recomendables para que el usuario lea. El agente Newt, se programa por medio de ejemplos de entrenamiento, el usuario le da artículos que debe y no debe leer para que aprenda o puede también programar al agente con reglas explícitas (por ejemplo "dame todos los artículos que contengan la palabra agente") guarda las palabras y luego realiza un análisis textual de los documentos para encontrar las palabras clave.

La tecnología de los buscadores al menos como funciona hoy en día no soluciona el problema de que de tanta información se pueda tener la que realmente le interese al usuario, es claro entonces que una posible solución sea la utilización de agentes ya que son herramientas ideales para buscar, filtrar u organizar información. A los usuarios les ayuda en la navegación y búsqueda de información en Internet a través de la personalización de la información y optimizando el acceso.

Existen muchas empresas que utilizan la tecnología de agentes para ofrecer información personalizada a sus usuarios. Por ejemplo la empresa Time Inc. *New Media* ofrece su popular servicio [34]:

✧ *Pathfinder,* que es una versión electrónica de las revistas Money, People, Sport Illustred y Fortune, entre otras, lo interesante es que ofrece un servicio denominado Personal Edition que consiste en un sistema personalizado de noticias, el suscriptor puede construir una serie de consultas que se usarán para crear una edición personalizada en Web de las noticias.

✧ Otro servicio parecido es el que ofrece el periódico de *San José Mercury News*, que es uno de los pioneros en este tema. Con NewsHound, los usuarios pueden crear hasta cinco perfiles especificando los términos obligados, opcionales y excluyentes que describen el tema de interés. Las noticias interesantes llegan al servicio estas se envían por correo electrónico cada hora a lo largo de todo el día.

Otro de los sistemas de filtrado de información que se menciona es la que se da en la programación de las Bases de Datos orientadas a la Web, donde la monitorización del usuario se convierte en algo indispensable para analizar sus acciones y le pueda servir de guía en el proceso de interacción.

En Scrapbook [28], los usuarios están interesados mediante la creación de una página personal seleccionando datos en el navegador Web y copiándolos a dicha página. La interacción se lleva a cabo mediante la selección por parte de los usuarios, para la posterior detección de datos mediante técnicas de (matching) que encajen con estos patrones. Otro de los trabajos relacionados es el de SmallBrowse [19], orientado a la navegación en dispositivos con una pantalla o display de características reducidas. Predice cuales son las tres ligas más relevantes para el usuario. La predicción de estas ligas se realiza mediante la inspección de un histórico durante la navegación del usuario, seleccionando los tres usados más frecuentemente.

La cantidad y variedad de información disponible hace necesario contar con herramientas que permitan manejar la información y tratar de evitar diferentes tipos de problemas que pudieran presentar, por ejemplo; tráfico en la red, tiempo de respuesta largos, fallas de conexión, entre otros.

➢ *Concurrencia de la información* [34]*:* El volumen de información disponible impide que se pueda encontrar información a una respuesta específica.

➢ *Los sistemas multiagentes* [12], donde interactúan varios agentes inteligentes persiguen algún conjunto común de objetivos o bien realizan un conjunto de tareas, constituyen en la actualidad un novedoso enfoque promisorio para problemas complejos.

2.2 ONTOLOGIAS EN LOS AGENTES

Ontología [23]*:* Las ontologías proveen una conceptualización del dominio del trabajo de una organización que representa conceptos principales y relacionados a las actividades del trabajo. Estas relaciones pueden relacionar información insólita tal como un número telefónico de la casa de un empleado, o ellos podrían representar una actividad tales como documentos actualizados, o la atención a conferencias. Se usa el término de ontología, para referirse a una estructura clasificada y la instancia dentro de una base de conocimientos.

El uso de ontologías para la recuperación de la información tiene ciertas ventajas sobre los métodos de acceso simples basados en palabras clave. Una ontología suministra un vocabulario compartido común para expresar información sobre el contenido de los documentos. Además, las ontologías incluyen axiomas para especificar relaciones entre conceptos. Por tanto, se pueden utilizar para formular consultas más complejas y recuperar exactamente la información en la que los usuarios están interesados. Con los axiomas es posible derivar información intencional. Para aportar significado a las páginas Web, éstas se pueden anotar con información ontológica. Esta información no es visible cuando la página se muestra con un navegador, pero si lo es para agentes o aplicaciones que buscan información en la Web. El conocimiento contenido en las páginas se anota usando las ontologías como un esquema para expresar metadatos. La forma de añadir esta meta-información depende del lenguaje que se este utilizando para ello. Se utiliza el lenguaje SHOE [16] (Simple HTML Ontology Extensions). SHOE, combina las características de los lenguajes de marcado, en la representación del conocimiento.

Ontobroker [9] [27], y el sistema SHOE [16], son dos sistemas anotadores de páginas web que funcionan en base a una ontología. Ambos proponen recoger la meta-información de las páginas y almacenarla en algún tipo de base de conocimiento. Para la construcción y manipulación de ontologías que representan dinámicamente una estructura semántica de Bases de Índices del mecanismo de búsqueda asociado, un usuario selecciona un tema en el que se inserta su búsqueda (o contexto de búsqueda) a partir de una ontología ofrecida por la herramienta. Este trabajo está relacionado con el proyecto Bright

(Brazilian Internet Guide in Hypertext), que evoluciona hacia un mecanismo de búsqueda llamado RADIX (algoritmo de ordenamiento).

Los dos sistemas experimentales Quickstep y Foxtrot, crean perfiles de usuarios desde un ambiente de monitoreo discreto y una retroalimentación relevante, la representación de perfiles en términos de una ontología de documentación. Una visualización novedosa de un perfil acerca la retroalimentación de tomar los perfiles. Los documentos de investigación son clasificados utilizando clases ontológicas en la recomendación de colaboración usando algoritmos para recomendar documentos vistos por personas similares sobre sus intereses comunes. Existen dos experimentos de tamaños diferentes, uno con 24 temas y otro con 200 temas, las evaluaciones son conducidas sobre diferentes aspectos en dónde la inferencia ontológica muestra y prevé un perfil de usuario, donde los conocimientos externos ontológicos usan susceptiblemente con éxito un sistema recomendado empleando una visualización del perfil para prever con exactitud el perfil del usuario. El desempeñó global de este sistema de ontología también es recomendado, presentado y comparado favorablemente con otros sistemas en la literatura.

En este capítulo, se presentaron trabajos relacionados con el acceso concurrente a una base de datos, se mencionan algunos trabajos relacionados con las Bases de Datos, así como también las diferentes tecnologías que se están aplicando sobre ellas, tales como las Bases de Datos Semántica y las Bases de Datos Inteligentes.

En el capítulo tres, se abordarán los temas relacionados a la metodología propuesta para desarrollar el sistema y comenzará la fase de análisis, diseño y construcción.

CAPÍTULO 3

DESARROLLO DEL SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES.

En este capítulo se desarrolla un sistema de control de concurrencia a una Base de Datos Mediante Agentes en el que se mejora el tiempo de respuestas a consultas frecuentes en una Base de Datos con acceso concurrente.

3.1 ANÁLISIS DE CASOS PARA DESARROLLAR EL SISTEMA

En esta sección se describe el análisis los tres casos que se tomaron en cuenta en este trabajo de tesis. El primero consiste de un modelo Cliente / servidor con conexión simultanea, el segundo es un modelo cliente/servidor que utiliza servlets y el tercero es un modelo que hace uso de una memoria caché, con un conjunto de agentes y una ontología, donde se genera un perfil de usuario y el agente hace propuestas automáticas de consultas que pueden resultar de interés para el usuario.

Caso 1: consiste de una aplicación basada en el modelo Cliente/Servidor (Figura 5) donde un cliente se conecta a través de un socket siempre y cuando el servidor se encuentre disponible. Una vez conectado, el servidor envía la respuesta a cada petición del usuario de la información. Cabe aclarar que el servidor maneja una cola de espera y en base a ella ejecuta los procesos que los clientes solicitan.

El beneficio que se obtiene al implementar el sistema Cliente/servidor, es que los clientes al momento de ejecutar un proceso obtienen una respuesta a su consulta sin importar el tiempo de espera.

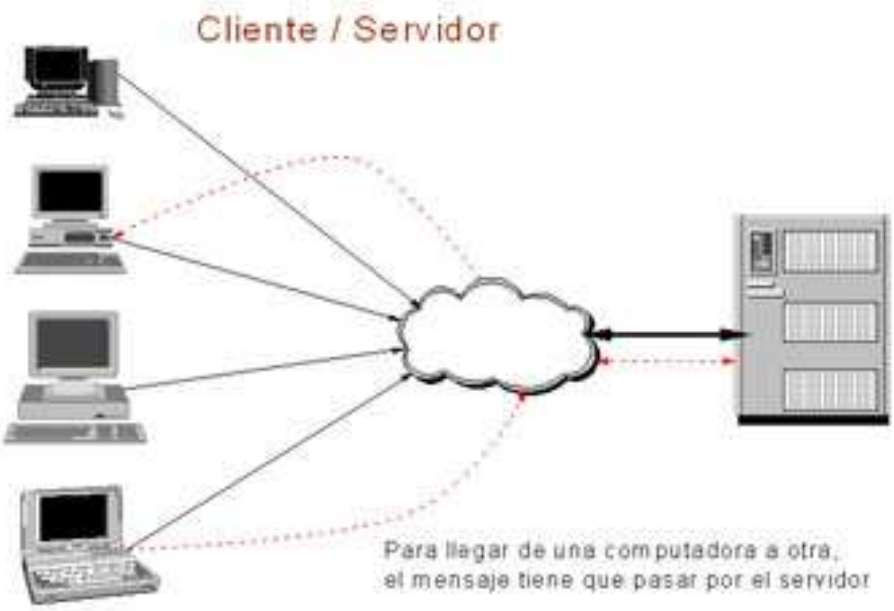


Figura 5. Caso 1: Modelo Cliente/Servidor con Conexión Constante.

Caso 2: Este caso utiliza el modelo cliente/servidor que se apoya en los Servlets (figura 6), mismos que permiten la atención a solicitudes pero sin mantener una conexión constante sino únicamente cada vez que se hace una solicitud. Como se sabe los Servlets se ejecutan en un servidor Web y presentan sus respuestas como páginas Web dinámicas.

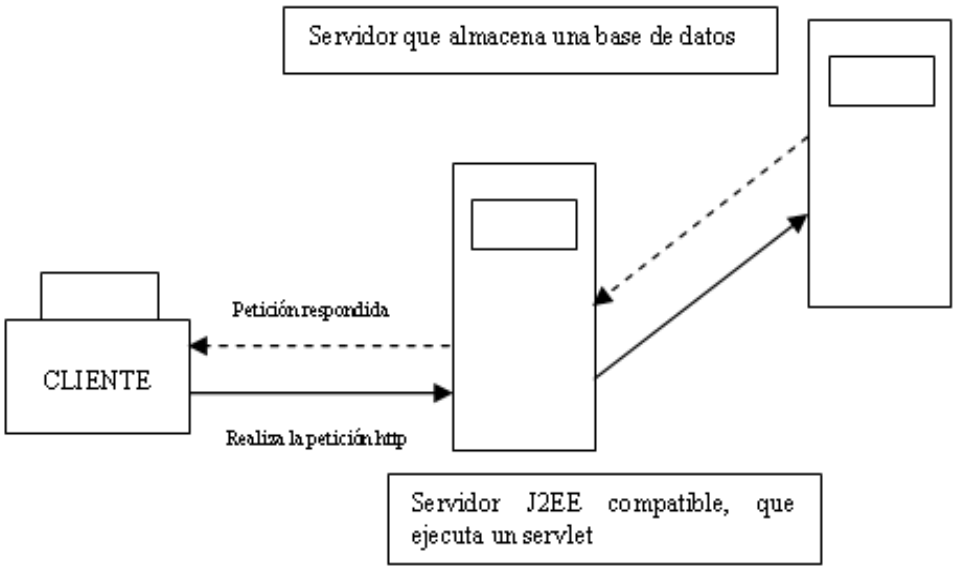


Figura 6. Caso 2: Modelo Cliente/Servidor con Servlets

Caso 3: consiste de una arquitectura ACO ver figura 7 de este capítulo, en la cual se tienen agentes que utilizan un perfil del usuario para interactuar con él y hacen búsquedas en su nombre, una memoria caché que almacena las solicitudes más frecuentes y contribuye a mejorar el tiempo de acceso a la base de datos. También se tiene una ontología con los conceptos de la biblioteca digital y que es útil cuando un agente requiere de hacer consultas sin la intervención del usuario. Se tienen dos tipos diferentes de agentes. El primero *agente-usuario* toma del perfil del usuario los temas de su preferencia para realizar búsquedas en la caché. El segundo *Agente-gestor de la base de datos* se encarga de realizar la búsqueda de aquellas consultas que no fueron localizadas por el agente-usuario en la caché. El agente gestor realiza la búsqueda en la base de datos y una vez que se localiza la información la almacena en la caché para que el agente-usuario lo tome y lo haga llegar al usuario quien realizo la petición de la consulta.

3.2 DISEÑO CONCEPTUAL DE LA ARQUITECTURA ACO

En este apartado se explica el diseño empleado en la construcción del caso tres, el cual ofrece las mayores ventajas para el acceso concurrente a una base de datos por muchos usuarios.

La arquitectura ACO, permite visualizar de forma general la estructura y funcionamiento del mismo, a través de la figura 7, se representa gráficamente el diseño conceptual del sistema, se puede tener una apreciación de como interactúan los objetos, al detectar las entradas, los procesos y las salidas generadas por el sistema. A continuación, se da una explicación del funcionamiento de cada uno de los objetos que conforman esta arquitectura, los cuales son: Usuario, Interfaz-hombre Agente, Perfil de usuario, Ontología, Caché de Consultas, Agentes Gestor de la Base de Datos y la base de Datos.

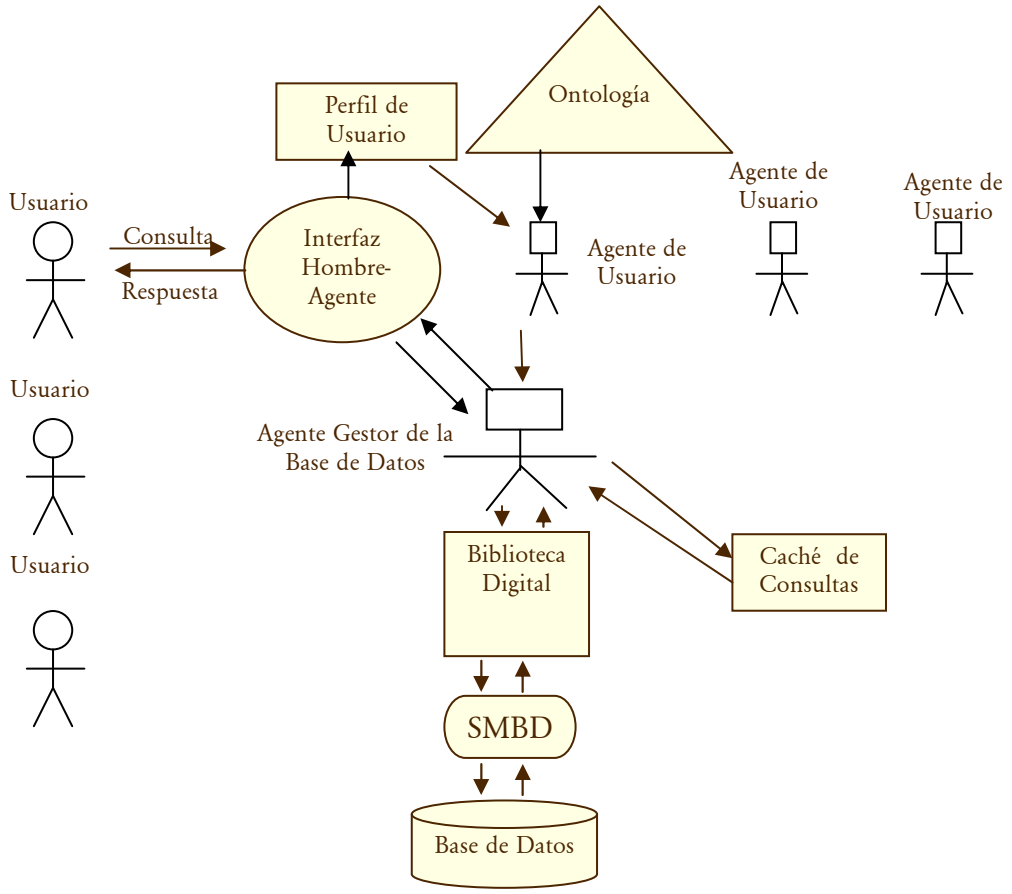


Figura 7. Caso 3: Aplicación del Diseño Conceptual de la Arquitectura ACO.

Usuario: Es aquella persona que desea ingresar al sistema, y que para lograrlo debe estar registrado en la base de datos. El usuario ingresa sus datos personales, su "login y password", con esto queda registrado para que, posteriormente, pueda ingresar al sistema sin complicaciones. Una vez que el usuario ya este dado de alta en el sistema, podrá consultar los temas que desee a través de un interfaz con que cuenta el sistema para llevar a cabo esta tarea.

Interfaz Hombre-Agente: Una vez que el usuario cumpla con las restricciones de seguridad del sistema y éste haya ingresado, procede a realizar sus consultas a través de la interfaz del sistema; *Interfaz Hombre-Agente* está un intermediario que existe entre el usuario y el agente usuario el cual se encarga de llevar la consulta hecha por el usuario, el agente toma la consulta y se inicia la tarea de búsqueda.

Agente-Usuario: Se encarga de realizar búsquedas de información de la siguiente manera: Primeramente este agente se encarga de buscar la información referente al perfil del usuario ya que los temas preferentes de los usuarios son almacenados en esa parte del sistema. Una vez que el agente-usuario conoce los temas de interés de los usuarios le sede la búsqueda al Agente Gestor, para que esté realice la búsqueda en la Caché.

Perfil de Usuario: En éste, se guardan los datos de los usuarios, así como también los temas de su interés y éste se nutre conforme los usuarios vayan ingresando los temas de su interés

Ontología: Se encarga de la organización jerárquica de los conceptos que se encuentran en la biblioteca digital. Utilizando la ontología y un conjunto de patrones de solicitudes a la base de datos es posible generar consultas que pueden resultar de interés para un usuario, con lo cual, se intenta brindarle ayuda al usuario en la formulación de sus consultas además se encarga de nutrir tanto el perfil del usuario así como también al Agente que se encarga de realizar la búsqueda de los temas de interés de los usuarios.

Caché de Consultas: Se almacenan los temas de interés más frecuentados por parte de los usuarios, a ella accesa el *agente-gestor de la base de datos* para realizar esta búsqueda, para que posteriormente el agente usuario realicé la entrega al usuario, esta Caché se nutre toda vez que los usuarios realizan otras consultas. Cuando la Caché ya no tenga espacio disponible para seguir almacenando otras consultas, se marcarán las consultas que hayan sido las menos frecuentadas por los usuarios y serán estas las que se eliminen, para que otros datos puedan almacenarse.

Esta caché cuenta con capacidad de 500 entradas e implementa un algoritmo para renovar las consultas menos populares con el que se garantiza que se elimina la consulta con menos accesos sin eliminar las que llegan al último.

Agente-Gestor de la Base de Datos: este agente se encarga de realizar las búsquedas de las consultas de los usuarios en la *Caché* si encuentra la información deseada, el *Agente-Gestor* entrega la información al *Usuario* interesado en la consulta, pero, si la consulta solicitada el *Agente-Gestor* no la encuentra en la *Caché*, entonces realiza la búsqueda de la

consulta solicitada en la Base de Datos y una vez que la encuentra, la toma para almacenarla a la Caché de Consultas, y entrega esa información al usuario que realizó esa petición. Este agente almacena las respuestas de las consultas en la tabla llamada respuesta de la base de datos como se observa en la tabla 2, en el campo Usuario se guarda el nombre del usuario quien realizo la petición de las consultas, en el campo número se almacena el número de secuencia que se le da a las respuestas con forme el *Agente-Gestor de la Base de Datos* las va encontrando, posteriormente estas respuestas se muestran al usuario en la liga llamada Consultas Solicitadas de la Pantalla Principal del Sistema.

Respuesta

Usuario	Numero
Juan	90
Juan	91

Tabla 2: Tabla donde se guarda el número que ya tiene la respuesta para el usuario

La estructura del archivo ver figura 7.1, donde se guardan las respuestas del *Agente-gestor de la base de datos* se llama número.dat.

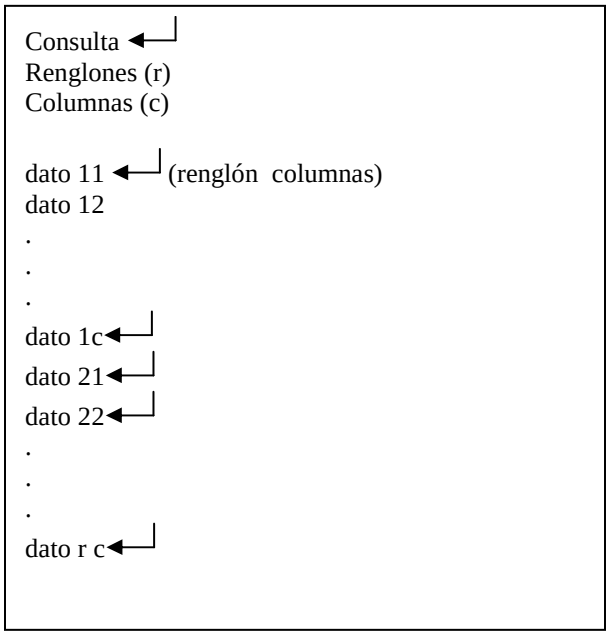


Figura 7.1: Estructura de archivo donde guardan las respuestas el *Agente-Gestor de la base de datos*

Donde consulta se refiere a las consultas que el usuario realizó, renglones y columnas es donde se almacenaran las respuestas de las consultas tomando en cuenta la estructura de la base de datos, dato es la respuesta esperada de la consulta de acuerdo con la base de datos.

Biblioteca Digital: Es la aplicación en donde se utiliza la arquitectura ACO y a través de ella se realizan las consultas de los usuarios

Sistema Manejador de la Base de Datos (SMBD): Es el sistema empleado para llevar a cabo la aplicación de la base de datos, en este caso la Biblioteca Digital.

Base de Datos: Es la base de datos del sistema en donde están almacenados toda la información y de la cual se tomarán las respuestas de las consultas hechas por los usuarios.

El funcionamiento de la arquitectura ACO se lleva a cabo de la siguiente manera, el *Agente-Gestor* de la base de datos de acuerdo a la petición del usuario selecciona las características mediante las preguntas o solicitudes que se realizaron y dispara un evento o conjunto de eventos (sensores), donde el *Agente-Gestor* interviene para la realización de la identificación de lo que requiere el usuario y así poder enviar una respuesta.

A continuación se describe el funcionamiento de esta arquitectura, como se muestra en la figura 8:

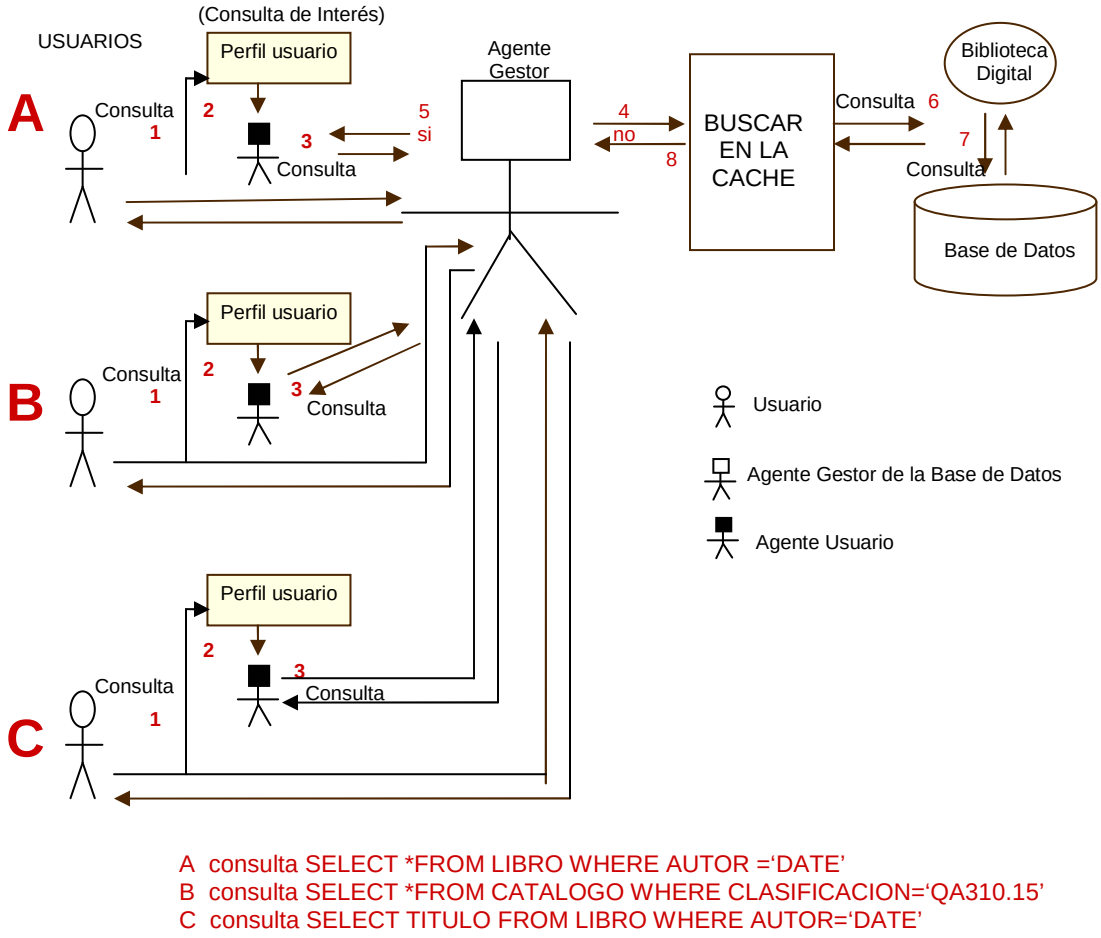


Figura 8. Caso 3: Funcionamiento de la Arquitectura ACO.

1) El usuario escribe una consulta en SQL.

2) La consulta se almacena en el perfil de usuario, y se manda al *agente-gestor de la base de datos* para llevar a cabo la búsqueda en la caché .

3) El Agente-usuario recibe consulta y la turna al Agente gestor de la base de datos.

4) El agente-gestor realiza la búsqueda en la Caché.

5) Si el *Agente-Gestor de la base de datos* localiza la información en la *Caché*, este regresa con la respuesta y hace la entrega al usuario solicitante,, pero si la consulta solicitada no fue localizada en la caché.

6) El *Agente-gestor* realiza la búsqueda de la solicitud en la Base de Datos.

7) Una vez que el A*gente gestor* localiza la información en la base de datos regresa con ella para que posteriormente la almacene en la *Caché.*

8) Una vez que la información se guarde en la *Caché*, el Agente gestor, hace la entrega al usuario de la información para que sea mostrada, al usuario quien realizó la petición de esa consulta.

Por ejemplo un usuario puede realizar cualquiera de estas consultas que a continuación se muestra:

A) consulta SELECT *FROM LIBRO WHERE AUTOR =‘DATE’

B) consulta SELECT *FROM CATALOGO WHERE CLASIFICACION=‘QA310.15’

C) consulta SELECT TITULO FROM LIBRO WHERE TITULO=‘BASE DE DATOS”

3.2.1 DIAGRAMAS DE CASOS DE USO DE LA ARQUITECTURA ACO

A continuación, se muestra en la figura 9, el diagrama del caso de uso del sistema empleado para el desarrollo del sistema:

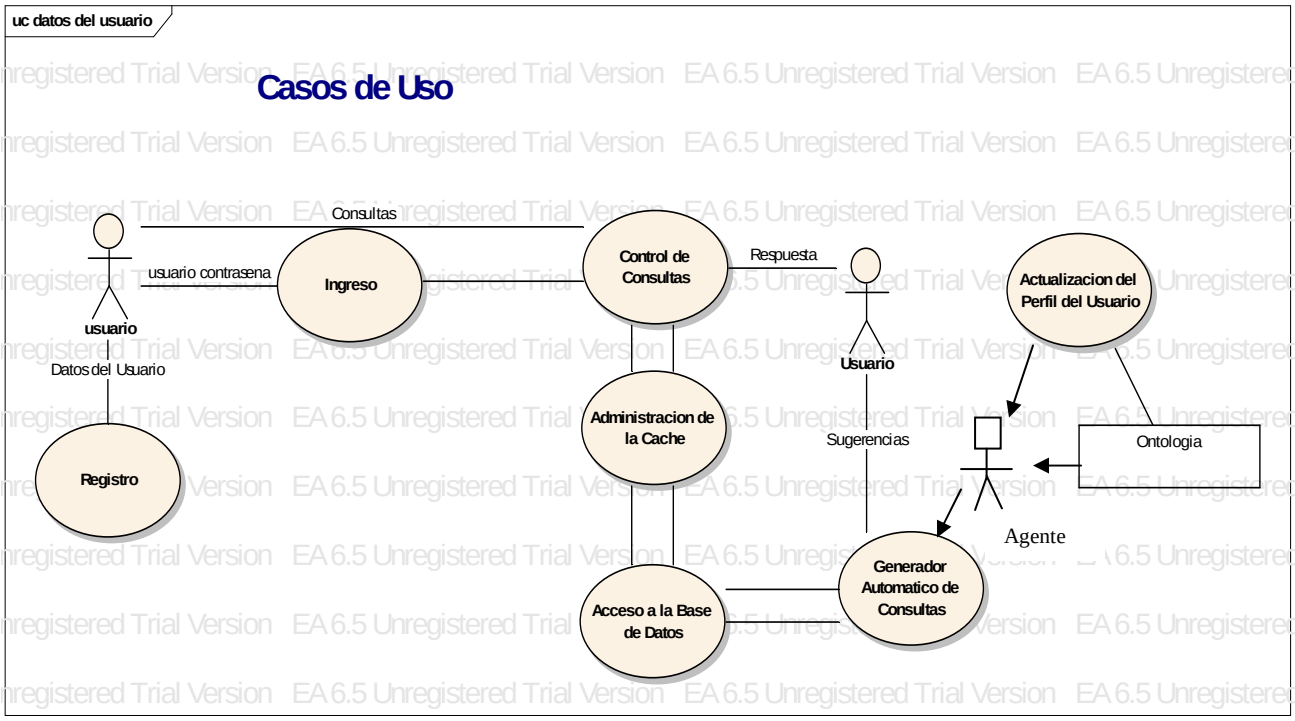
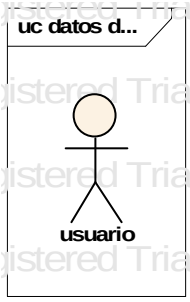


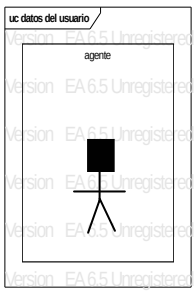
Figura 9. Diagrama de Caso de Uso del caso 3.

Como se aprecia en la figura 9, se ve a detalle en que consiste cada uno de los objetos del caso de uso del sistema.

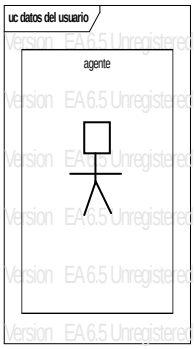


El *Usuario*; representa una persona que realiza consultas al sistema. Para ello el usuario debe iniciar una sesión, en donde se le pide el nombre de usuario y una contraseña, si es correcto este dato entonces se hace la consulta y se presenta la interfaz.

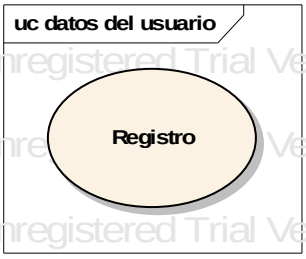
Una vez establecida la conexión al Sistema, el usuario puede emitir todas las consultas que desee y visualizar los resultados obtenidos de la consulta.



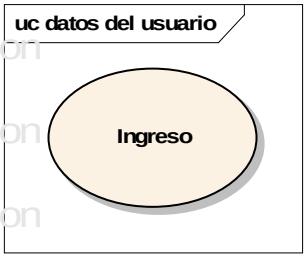
El *Agente-Usuario:* se encarga de realizar las búsquedas de las consultas hechas por los usuarios.



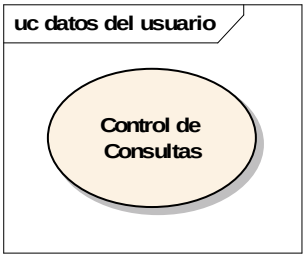
El *Agente-Gestor*: se encarga de buscar la información en la caché y si no la encuentra la busca en la base de datos.



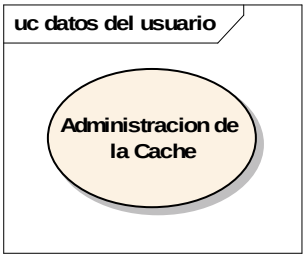
Registro: Es cuando el usuario ingresa todos sus datos al sistema para darse de alta, en el momento que el usuario se de alta este objeto permite que los datos del usuario nuevo se guarden en la base de datos.



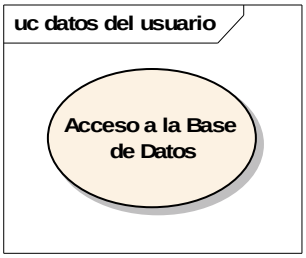
Ingreso: Se encarga de realizar la conexión del usuario con el sistema, el usuario debe ingresar su "login y password" de manera correcta para poder accesar al sistema.



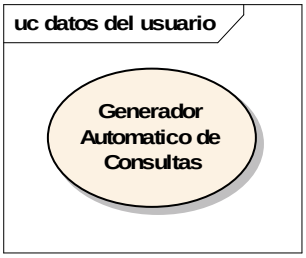
Control de Consultas: Es el registro de consultas que los usuarios realizan a la base de datos.



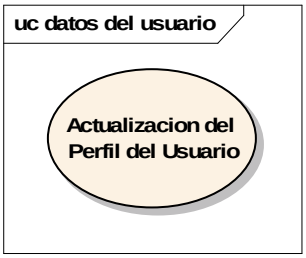
Administración Caché: Esta se encarga de almacenar aquellas consultas más frecuentadas por los usuarios. Además, aquellas consultas poco frecuentadas por los usuarios se irán marcando para que, posteriormente, sean eliminados para que existan registros vacíos para almacenar otros datos.



El *Acceso a la Base de Datos:* Se encarga de almacenar toda la información del sistema de la base de datos, el cual es accesada por el agente para realizar las consultas de los usuarios.



El *Generador Automático de Consultas:* se encarga de realizar la búsqueda de forma automática de las preferencias hechas por los usuarios. Estas consultas se ejecutan a través de los agentes.



El *Perfil de Usuario:* en ella se guardan los datos del usuario y sus temas preferentes, también se actualizan los temas de interés cada vez que así lo deseen.

La *Ontología:* es la organización jerárquica de los conceptos que se encuentran en el sistema, el cual es posible navegar para generar posibles solicitudes de información que efectúa el usuario, para brindarle ayuda al usuario en la formulación de sus consultas.

3.2.2 DIAGRAMAS DE CLASES DE LA ARQUITECTURA ACO

En los siguientes Diagramas de Clases, se muestran los objetos que se emplearon para la Conexión a la Base de Datos.

Las clases utilizadas para las conexiones de la Base de Datos se muestran en la figura 10. El diagrama de clases de la base de datos de la biblioteca digital que se tomó

como ejemplo para la aplicación del sistema son: Perfil de Usuario, Ontología, Revista, Video, Mapas, libros.

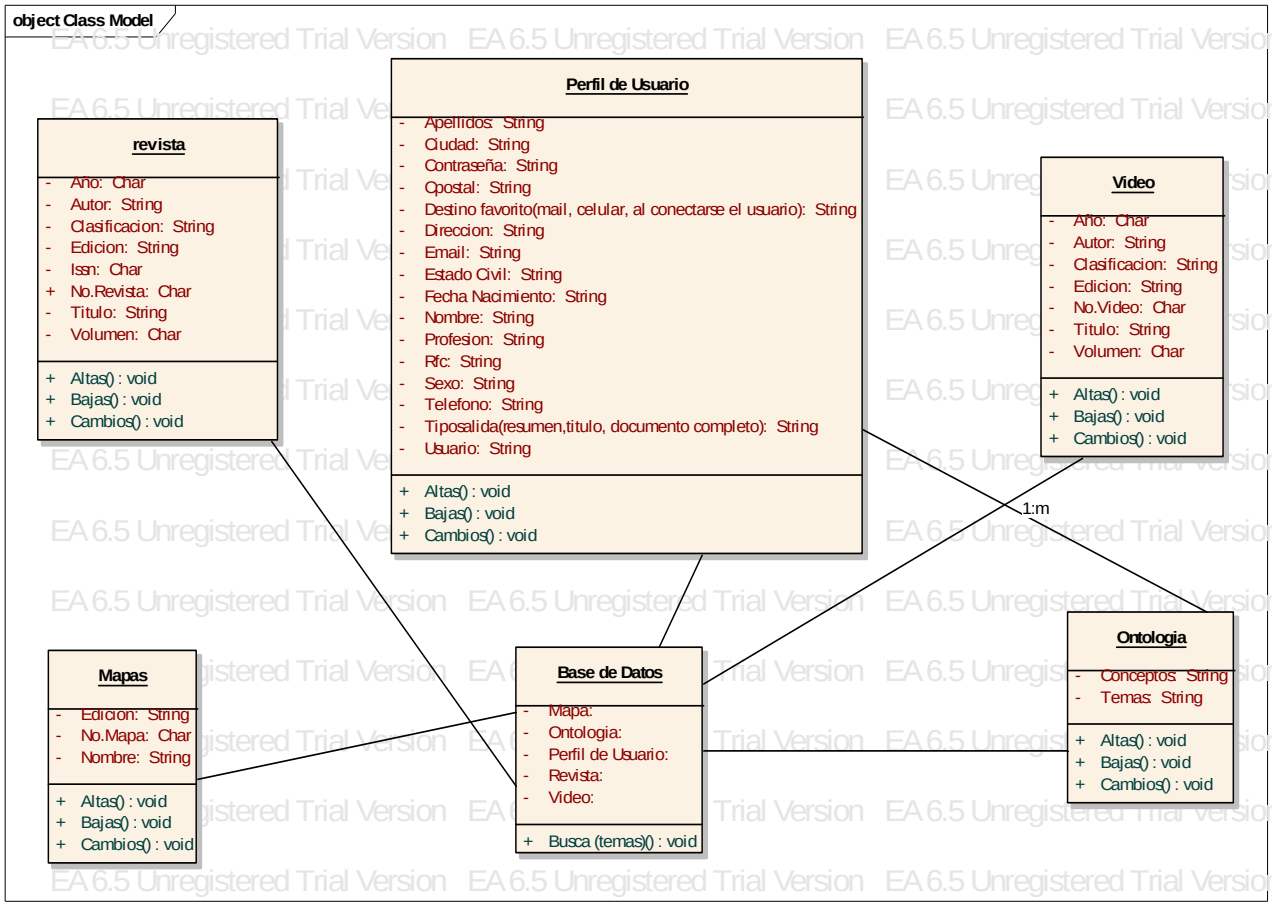


Figura 10. Diagrama de Clases del Sistema de la Base de Datos.

A continuación, se da una explicación acerca del diagrama de clase que se muestra en la figura 10:

a) *Perfil de Usuario:* Se encarga de almacenar aquellas consultas preferentes hechas por los usuarios.

b) *Ontología:* Se encarga de estructurar aquellos temas de interés de los usuarios, los cuales podrá visualizarlos posteriormente el usuario en forma de un árbol. Existe una relación con el perfil de usuario, ya que el perfil de usuario se estará

actualizando constantemente de acuerdo a los diversos temas de interés que tengas lo usuarios.

c) *Video:* en ella se almacena todo el material de la "mediateca" que está dados de alta en la base de datos. La cual se puede actualizar (altas, bajas, cambios).

d) *Revistas:* guarda el material de revistas dadas de: alta en la base de datos, también se puede actualizar (altas, bajas, cambios).

e) *Mapas:* almacena los mapas que se encuentran en la base de datos, se puede actualizar (altas, bajas, cambios).

En el diagrama (figura 11), se muestra de forma detallada cual es la funcionalidad al momento en que los usuarios se registran al sistema, parten desde un registro para poder obtener una cuenta y un password de inicio de sesión, en donde esta información es guardada en el perfil del usuario. Una vez que el usuario accesa con los datos que el sistema requiere para su ingreso accesa a la pantalla principal del sistema y continuar con la tarea que requiere en este caso la de consultar en la base de datos. En la clase de PantallaObtenerRegUsuario, se crea una pantalla para el ingreso de los datos del usuario, posteriormente, le envía una pantalla de registro de usuario enseguida enviarle una pantalla de interfaz del usuario. La clase PantallaCrearRegUsuario es para los usuarios nuevos que no han sido registrados, una vez dados de alta se le muestra una interfaz de acceso al usuario. El Perfil de Usuario guarda sus datos al manejador de la base de datos en donde se almacena la información. RegistroUsuario cada dato nuevo de perfil del usuario es registrado y actualizado.

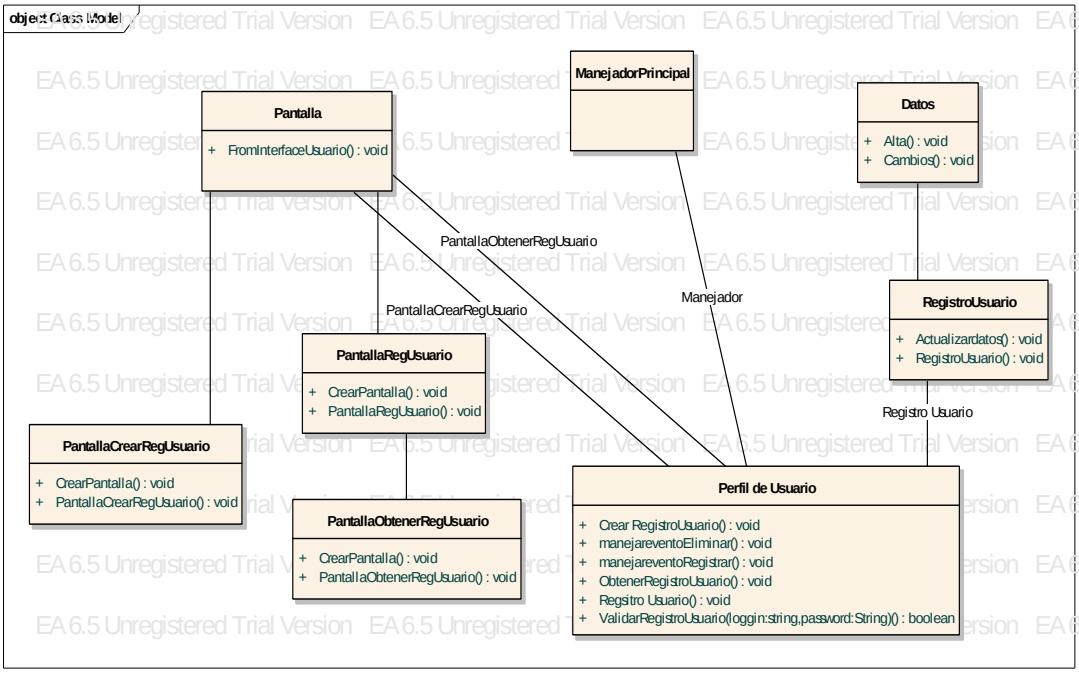


Figura 11. Diagrama de Clases para el modulo de Usuario.

3.2.3 DIAGRAMAS DE INTERACCIÓN DE LA ARQUITECTURA ACO

En los siguientes esquemas se muestran los diagramas de interacción para llevar a cabo el ingreso de los usuarios y los mensajes que intercambian de forma ordenada, según la secuencia del tiempo.

En la figura 12, se puede ver, que el eje vertical representa el usuario que ingresa al sistema para realizar las diferentes tareas que requiera y el tiempo que esta tarea es ejecutada por cada usuario y en el eje horizontal se tiene las principales tareas o actividades que realiza el usuario para ingresar al sistema, estas son: login, seguridad y usuario.

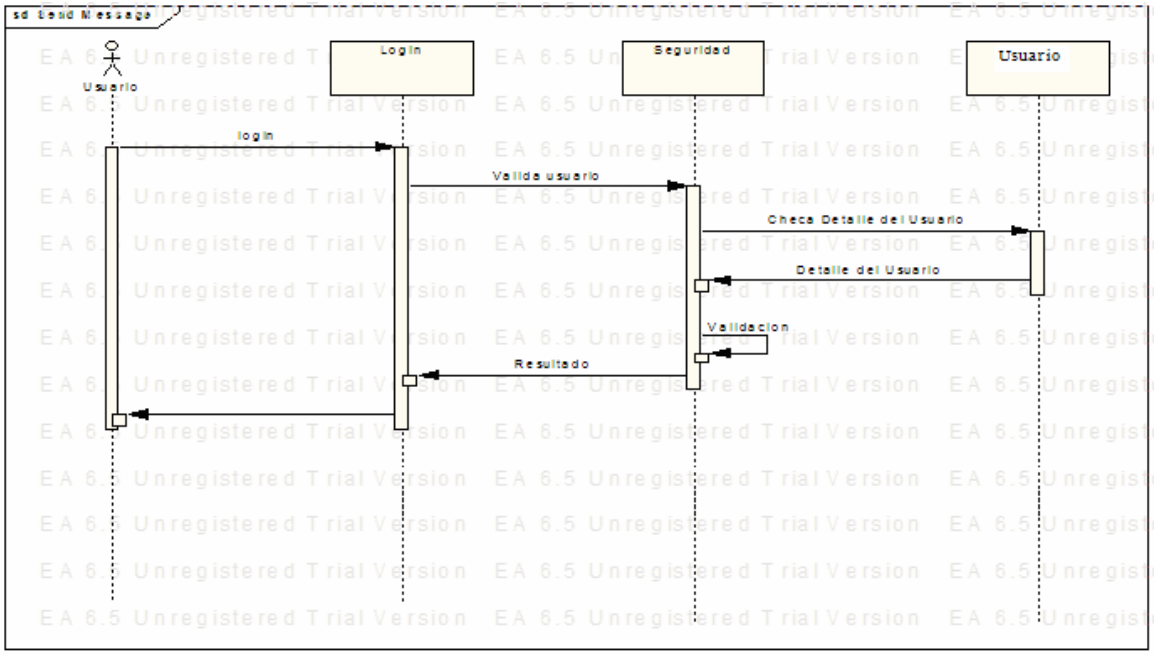


Figura 12. Diagrama de Interacción de Usuarios.

En este diagrama el *usuario* ingresa sus datos de registro en este caso: "login, password"; una vez que el usuario termine de ingresar sus datos, se validan y se verifica que sean los correctos, estos datos se corroboran a detalle para determinar que son los datos correctos para así poder ingresar al sistemas.

En la figura 13, se describen con detalle la forma en que los usuarios realizan las consultas dentro del sistema y esto es muestra el proceso que se lleva hasta la obtención del resultado esperado por parte del usuario.

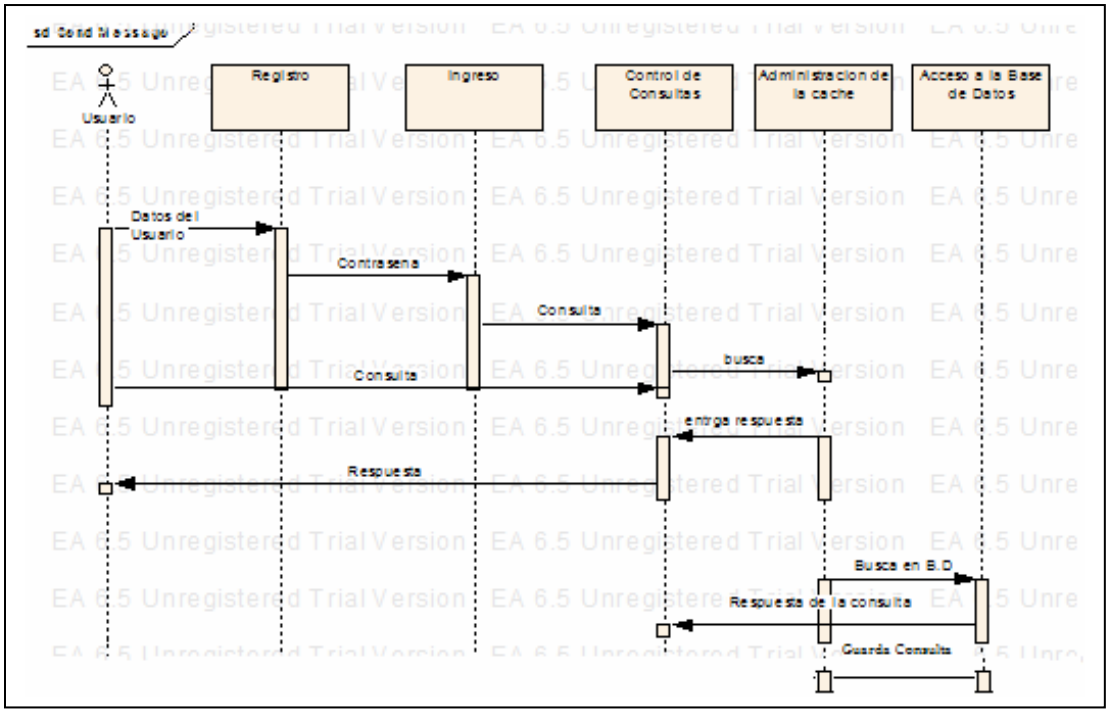


Figura 13. Diagrama de Interacción de la Consulta de Usuarios.

Datos del Usuario, si el usuario es nuevo tiene que registrar sus dato para ser dado de alta y proponer una cuenta de acceso (login) y una contraseña (password), pero si el usuario ya cumplió esta etapa sólo tiene que solicitar el ingreso para realizar las consultas que desee.

Las consultas del usuario se manejan a través de un *control de consultas*, una vez que el usuario ingresa su consulta, se procede a buscar la información en la *administración de la caché*, de ser encontrado la información buscada, se realiza la entrega de la misma al usuario quien realizó la petición; en caso contrario, es decir, que la información no fuera hallada en la *administración de la caché*, se procede una búsqueda en el *Acceso a la base de datos* para que, posteriormente, esta información localizada llegue al solicitante; y también, se guarde en la caché.

En la figura 14, se describe el diagrama de secuencia con la intervención del agente, en donde el agente toma la función de realizar las búsquedas automáticas de las consultas de los usuarios.

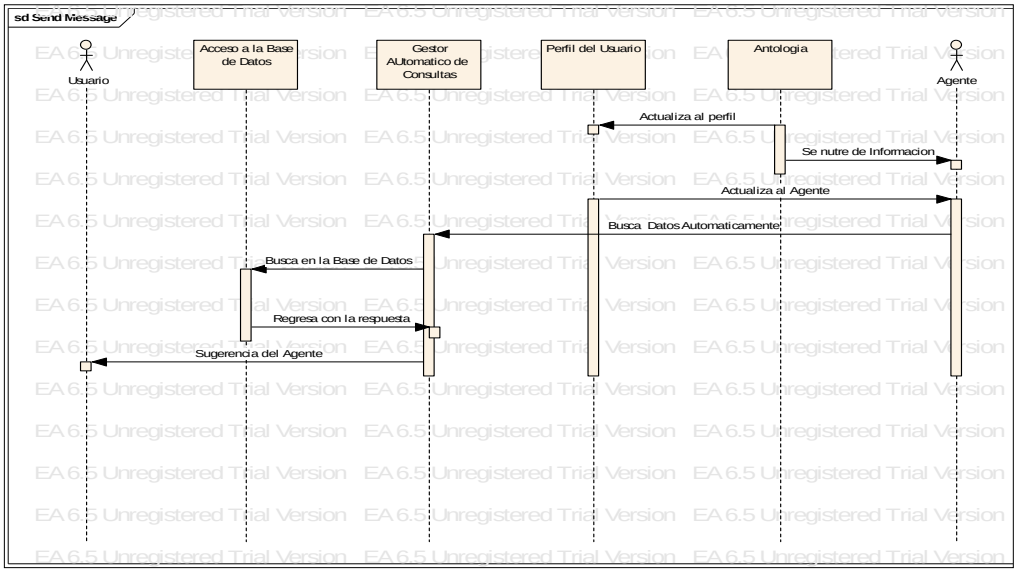


Figura 14. Diagrama de Interacción de las Consultas Automáticas del Agente.

En este diagrama se observa que las consultas hechas por los usuarios se van guardando en el perfil de usuario, a su vez en el *perfil de usuario*, se actualizan los conceptos existentes en la *ontología* de acuerdo al interés de cada uno de los usuarios.

El *agente*, sabe de los temas preferentes de los usuarios a través del perfil de usuario ya que en ella residen los temas más importantes y de esta manera los agentes podrán realizar las consultas de manera automática en la bases de datos y darle sugerencias de la información solicitada.

3.2.4 DIAGRAMA DE LA BASE DE DATOS

Para la aplicación del Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes se diseñó el siguiente diagrama de Entidad / Relación generalizado de la base de datos, como se muestra en la figura 15:

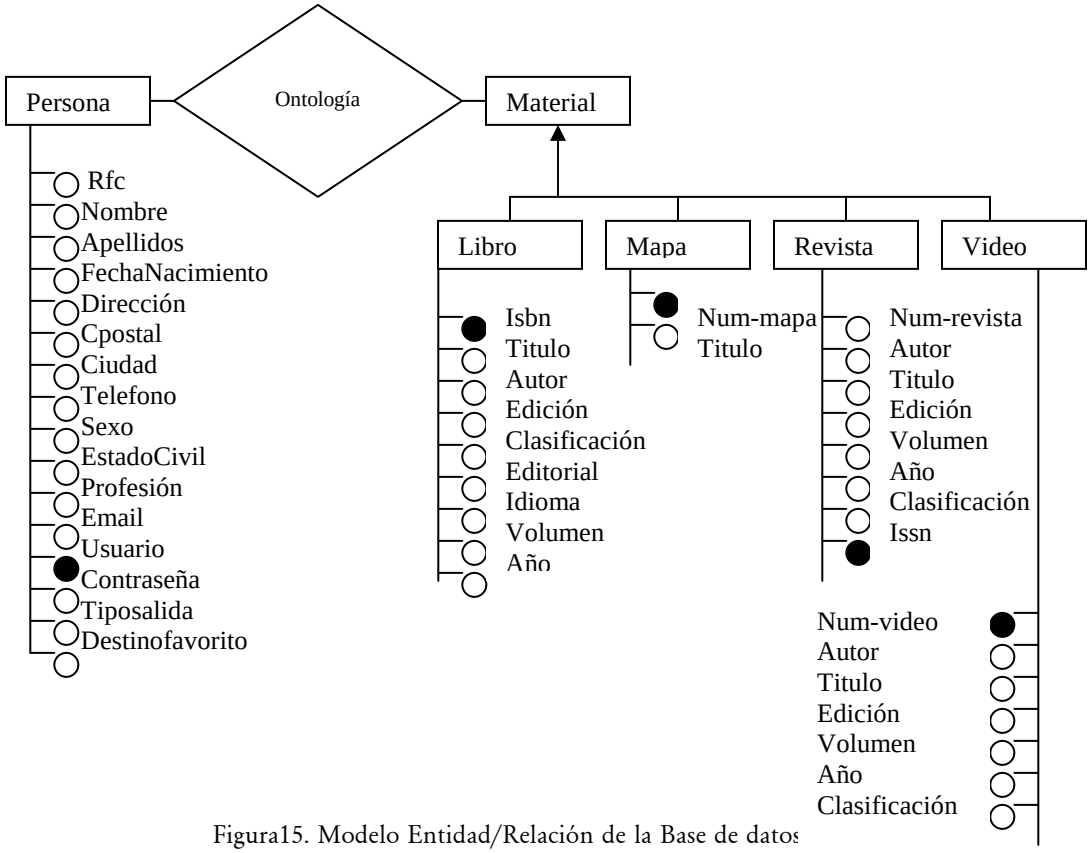


Figura15. Modelo Entidad/Relación de la Base de datos

Persona: Se refiere a los datos de las personas que están dadas de alta en el sistema, y cuenta con cada uno de sus atributos.

Ontología: Se almacenan los conceptos de forma jerárquica de la Biblioteca Digital

Material: Se refiere a libro, mapa, revista, video que están almacenados en la base de datos, a continuación se da una breve explicación de cada uno de ellos:

Libro: Se refiere al material bibliográfico que está almacenado en la base de datos.

Mapa: Se refiere a los mapas que están almacenados en la base de datos.

Revista: Se refiere a las revistas que están almacenados en la base de datos.

Video: Son todos los videos que están almacenados en la base de datos.

Tab_Libro		
Campo	**Tipo**	**Longitud**
Isbn	Texto	10 Caracteres
Titulo	Texto	50 Caracteres
Autor	Texto	50 Caracteres
Edición	Texto	10 Caracteres
Clasificación	Texto	10 Caracteres
Editorial	Texto	20 Caracteres
Idioma	Texto	10 Caracteres
Volumen	Texto	5 Caracteres
Año	Texto	20 Caracteres
Tab_Mapa		
Campo	**Tipo**	
Num-mapa	Texto	10 Caracteres
Titulo	Texto	50 Caracteres
	Texto	50 Caracteres
Tab_Persona		
Campo	**Tipo**	
Rfc	Texto	10 Caracteres
Nombre	Texto	50 Caracteres
Apellidos	Texto	50 Caracteres
FechaNacimiento	Texto	10 Caracteres
Dirección	Texto	30 Caracteres
Cpostal	Texto	8 Caracteres
Ciudad	Texto	20 Caracteres
Teléfono	Texto	15 Caracteres
Sexo	Texto	15 Caracteres
EstadoCivil	Texto	10 Caracteres
Profesión	Texto	30 Caracteres
Email	Texto	15 Caracteres
Usuario	Texto	10 Caracteres
Contraseña	Texto	10 Caracteres
Tiposalida	Texto	10 Caracteres
Ontología	Texto	10 Caracteres
Destinofavorito	Texto	10 Caracteres
Tab_Revista		
Campo	**Tipo**	
Num-revista	Texto	10 Caracteres

Tabla 3. Descripción del diagrama E/R.

Autor	Texto	50 Caracteres
Titulo	Texto	50 Caracteres
Edición	Texto	10 Caracteres
Volumen	Texto	10 Caracteres
Año	Texto	20 Caracteres
Clasificación	Texto	10 Caracteres
Issn	Texto	10 Caracteres
Tab_Video		
Campo	**Tipo**	
Num_video	Texto	10 Caracteres
Autor	Texto	50 Caracteres
Titulo	Texto	50 Caracteres
Edición	Texto	10 Caracteres
Volumen	Texto	10 Caracteres
Año	Texto	20 Caracteres
Clasificación	Texto	10 Caracteres

Tabla 3a. Continuación de la Descripción del diagrama E/R.

En base a la descripción del diagrama se tienen las siguientes tablas relacionales: Libro, Mapa, Persona, Revista Video, en donde cada una ellas cuenta con diferentes elementos, tales como: campo, el tipo de datos y longitud.

Más adelante se describe el Diccionario de Datos, de la biblioteca digital que se utilizó para las pruebas del sistema, en donde se muestra una lista y una descripción detallada de todos los elementos de almacenamiento de la información empleada.

En un diccionario de datos básico, cada entrada o elemento consiste en un conjunto de detalles que describen los datos utilizados o transformados por el sistema.

Entonces cada dato o atributo, se identifica por los siguientes elementos:

Nombre de los datos: para distinguir un dato de otro, se deben asignar nombres significativos que se utilizan para tener una referencia de cada elemento a través del proceso.

Descripción de los datos: Establece brevemente lo que representa el dato en el sistema.

Alias: Con frecuencia el mismo dato puede conocerse con diferentes nombres dependiendo de quién lo utilice, y estos son conocidos como alias o nombres.

Longitud del campo o tipo: Identifica el número de espacio para guardar los posibles valores del campo (letras, números o símbolos).

Dominio: Se refiere a los posibles valores que puede tomar un dato de acuerdo con su identificación (longitud, tipo, unicidad, etc.).

Rango: Se refiere a los valores permitidos con el dominio y que corresponden con la realidad del dato.

A continuación se describe el diccionario de datos, del proyecto.

Tabla. Libro

Campo	Tipo	Longitud	Descripción	Rango	Llave Primaria	Llave alterna	Valor Nulo
Isbn	Texto	10	Número del libro	A-Z, 0-9	Si	No	No
Título	Texto	50	Título del libro				No
Autor	Texto	50	Nombre de quien escribió el libro				No
Edición	Texto	10	Número de la edición del libro				No
Clasificación	Texto	10	Número de clasificación del Libro				No
Editorial:	Texto	20	Quién edita el libro				No
Idioma:	Texto	10	Lengua en la que esta escrito el libro				No
Volumen:	Texto	5	Es el número del tomo del libro				No
Año	Texto	20	Año en que fue editado el libro				No

Tabla 4. Diccionario de Datos de la tabla Libro.

Tabla. Mapa

Campo	Tipo	Longitud	Descripción	Rango	Llave Primaria	Llave alterna	Valor Nulo
Num_mapa	Texto	10	Número del libro	A-Z, 0-9	Si	No	No
Título	Texto	50	Título del libro				No

Tabla 5. Diccionario de Datos de la tabla Mapa.

Tabla. Persona

Campo	Tipo	Longitud	Descripción	Rango	Llave Primaria	Llave alterna	Valor Nulo
Rfc	Texto	10	Registro Federal de contribuyentes	A-Z, 0-9	No	No	No
Nombre	Texto	50	Nombre del usuario				No
Apellidos	Texto	50	Apellidos materno y paterno				No
Fecha Nacimiento	Texto	10	Fecha en la que el usuario nació				No
Dirección	Texto	30	Ubicación del usuario				No
Cpostal	Texto	8	Número postal usuario				No
Ciudad	Texto	20	Ciudad de donde es el usuario				No
Teléfono	Texto	15	Teléfono del usuario				No
Sexo:	Texto	15	Sexo del usuario				No
EstadoCivil	Texto	10	Estado civil de usuario				No
Profesión	Texto	30	A que se dedica el usuario				No
Email	Texto	15	Correo electrónico del usuario				No
Usuario	Texto	10	"Login" de identificación del usuario para ingresar al sistema		Si		No
Contraseña	Texto	10	Candado de acceso del usuario				No
Tiposalida	Texto	10	Forma en la que el usuario desea ver el la información solicitada				No
Ontología	Texto	10	Temas de interés del usuario				No

Tabla 6. Diccionario de Datos de la tabla Persona.

Tabla. Revista

Campo	Tipo	Longitud	Descripción	Rango	Llave Primaria	Llave alterna	Valor Nulo
Num-revista	Texto	10	El número de revista	A-Z, 0-9	No	No	No
Autor	Texto	50	Nombre de quién escribió la revista				No
Título	Texto	50	Título de la revista				No
Edición	Texto	10	Número de la edición del revista				No
Volumen	Texto	10	Número del tomo de la revista				No
Año	Texto	20	Año en que fue editado el revista				No
Clasificación	Texto	10	Número que tiene la revista de clasificación				No
Issn	Texto	10	Identificador único de la revista		Si	No	No

Tabla 7. Diccionario de Datos de la tabla Revista.

Tabla. Video

Campo	Tipo	Longitud	Descripción	Rango	Llave Primaria	Llave alterna	Valor Nulo
Num_video	Texto	10	Identificador del video	A-Z, 0-9	Si	No	No
Autor	Texto	50	Nombre de quien creo el video				No
Título	Texto	50	Título que lleva el video				No
Edición	Texto	10	Número de la edición del video				No
Volumen	Texto	10	Número del tomo del video				No
Año	Texto	20	Año en que fue editado el video				No
Clasificación	Texto	10	Número que tiene el video para la clasificación				No

Tabla 8. Diccionario de Datos de la tabla Video.

En las tablas anteriores, se pudo observar el Diccionario de Datos empleados en la base de datos que fue empleado para la aplicación del Sistema de Control de Concurrencia a una Base de Datos a través de Agentes.

3.3 CONSTRUCCIÓN DEL SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES.

En esta sección se describe la interfaz gráfica del usuario (GUI), para su construcción se utilizó el lenguaje orientado a objetos Java. A continuación, se muestra el diseño de las pantallas del sistema, al mismo tiempo que se describe el contenido de cada una de ellas:

En primer lugar, se muestra la pantalla de `NUEVO USUARIO` en la figura 16:

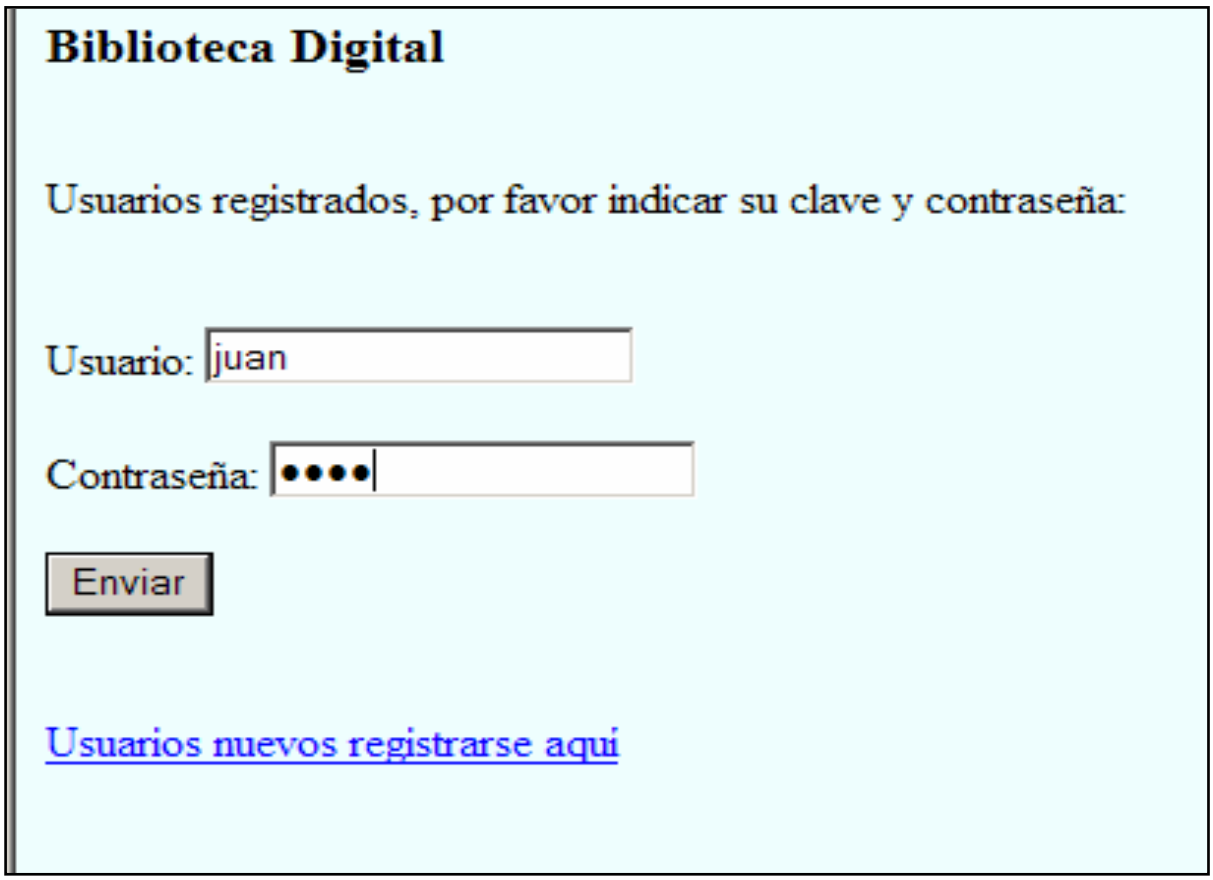


Figura 16. Interfaz del Ingreso del Usuario.

Aquí se despliega una pantalla solicitando la clave del usuario y su contraseña. Si el usuario es nuevo entonces hace "click" en la liga "Usuarios Nuevos Registrase aquí", y

ahí le aparecerá una pantalla como se puede ver en la (figura 17), en donde el usuario escribirá sus datos personales, así como también, propondrá una clave de acceso y su contraseña para poder ingresar al sistema.

En la interfaz de ingreso del usuario ya registrado, el proceso a seguir es el siguiente, el usuario escribe su clave y contraseña posteriormente presiona el botón enviar, se le mostrara la PANTALLA PRINCIPAL del sistema como se muestra en la figura 18, en la cual el usuario podrá realizar sus consultas.

En la pantalla de la figura 17, interfaz del Nuevo Usuario, el usuario ingresa todos sus datos personales así como también propone una clave y contraseña para ingresar al sistema, a continuación se muestra dicha figura:

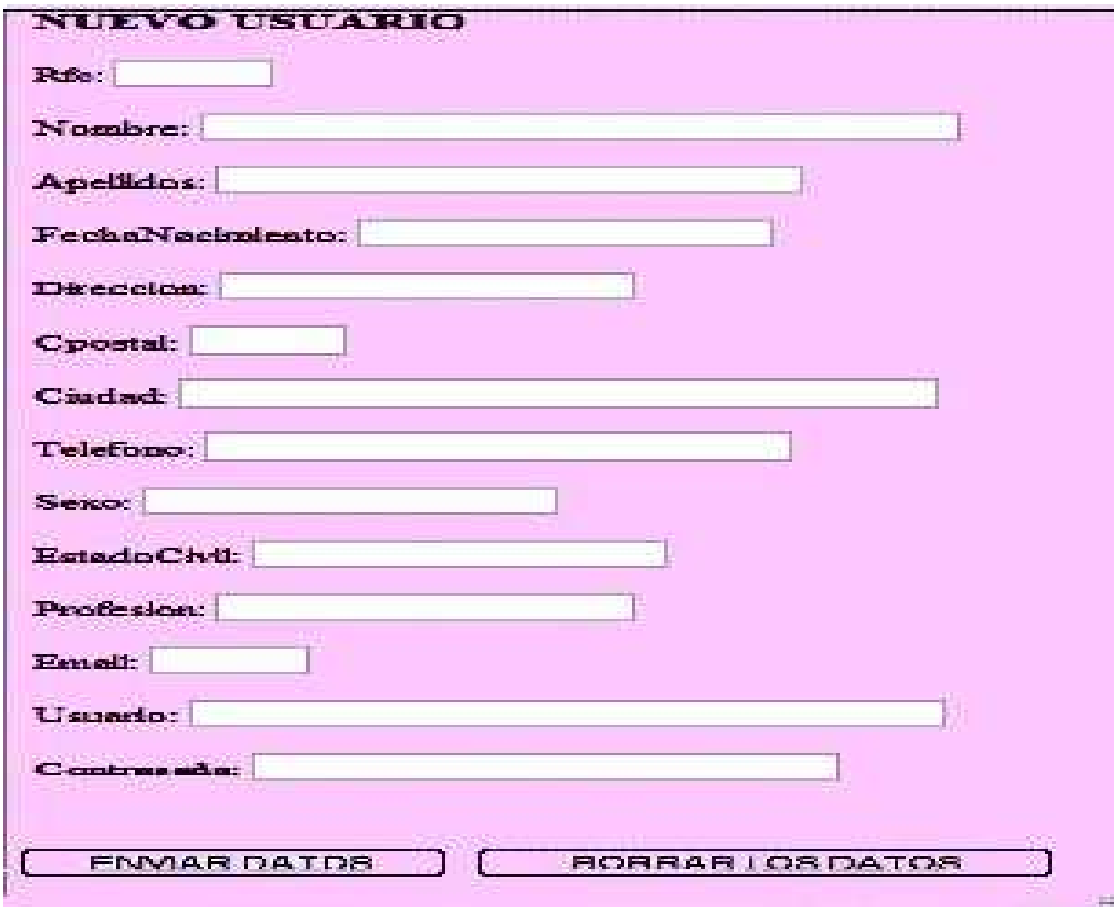


Figura 17. Interfaz del Nuevo Usuario.

Las casillas corresponden a los datos personales del usuario, se muestran en la figura 17, tales como nombre, apellidos, clave de acceso y su contraseña, etc., en donde el usuario ingresa sus datos, cuando el usuario halla terminado de llenar todas las casillas

para el registro de sus datos debe presionar el botón ENVIAR DATOS donde se da la orden al Sistema Manejador de la Base de Datos de almacenar estos, al momento de que son almacenados en la base de datos, enseguida se le muestra la pantalla principal del sistema al usuario para realizar las consultas que desee como se puede ver en la figura 18, o bien presiona el botón BORRAR LOS DATOS si el usuario decide en un momento dado ya no darse de alta en el sistema.

En la figura 18, cuando el usuario ingrese correctamente su clave y su password le aparecerá la pantalla principal del sistema, en esta se muestra lo siguiente: se da la bienvenida al usuario mostrándole su nombre y apellidos, al mismo tiempo se le visualiza un botón llamado desconectar que al presionarlo el usuario, le permitirá desconectarse de la base de datos y de este modo abandonar la sesión del sistema. Se muestra también un cuadro de texto en donde se escriben las consultas que se desean, también se muestra el botón limpiar que al presionarlo el usuario, borrará aquellas consultas y respuestas para iniciar nuevas consultas.

De lado derecho de esta pantalla se observan dos ligas una llamada consultas solicitadas y la otra lleva por nombre sugerencias de los agentes:

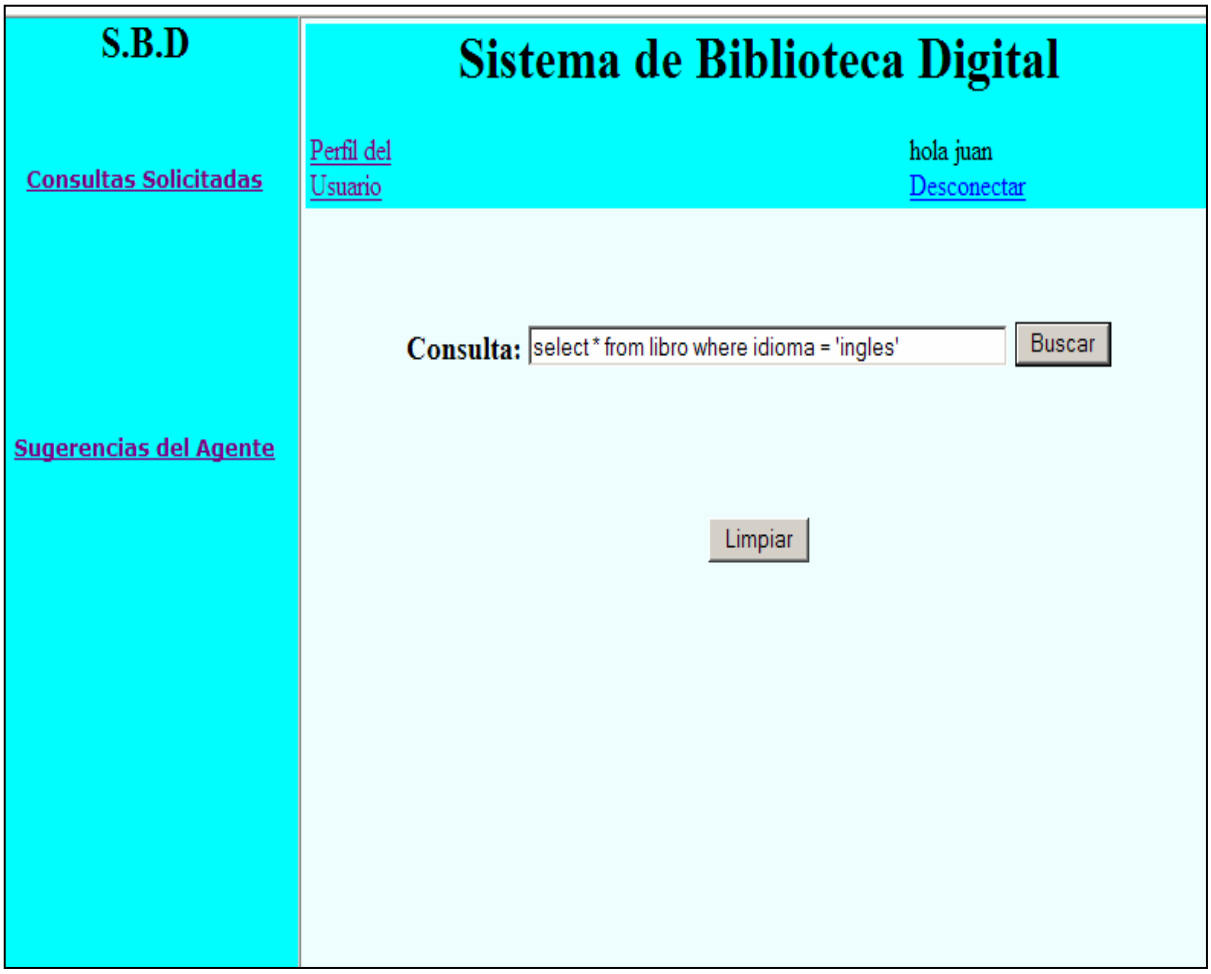


Figura 18. Pantalla Principal del Sistema.

En esta pantalla se tiene un cuadro texto como se mencionó antes, en la cual el usuario introducirá su consulta y al hacer "Click" en el botón BUSCAR el sistema realizará la búsqueda de aquellas consultas que el usuario requiere de la base de datos, mostrando la respuesta en la CAJA DE TEXTO que se encuentra en la parte de abajo de este botón.

Enseguida se dan algunos ejemplos de las consultas que los usuarios pueden realizar:

a) consulta SELECT *FROM MAPA WHERE TITULO ='OAXACA'

b) consulta SELECT *FROM CATALOGO WHERE CLASIFICACION='QA310.15'

c) consulta SELECT TITULO FROM REVISTA WHERE TITULO='REDES'

d) consulta SELECT *FROM VIDEO WHERE TITULO =‘SISTEMA SOLAR’

e) consulta SELECT *FROM VIDEO WHERE CLASIFICACION=‘V333.45’

f) consulta SELECT TITULO FROM REVISTA WHERE TITULO=’COMPUTACION’

Al hacer Click en la liga de `CONSULTAS SOLICITADAS`, le mostrará la interfaz de la figura 19, en donde se encuentran en forma de lista todas aquellas respuestas de las consultas previas que el usuario realizó al sistema:

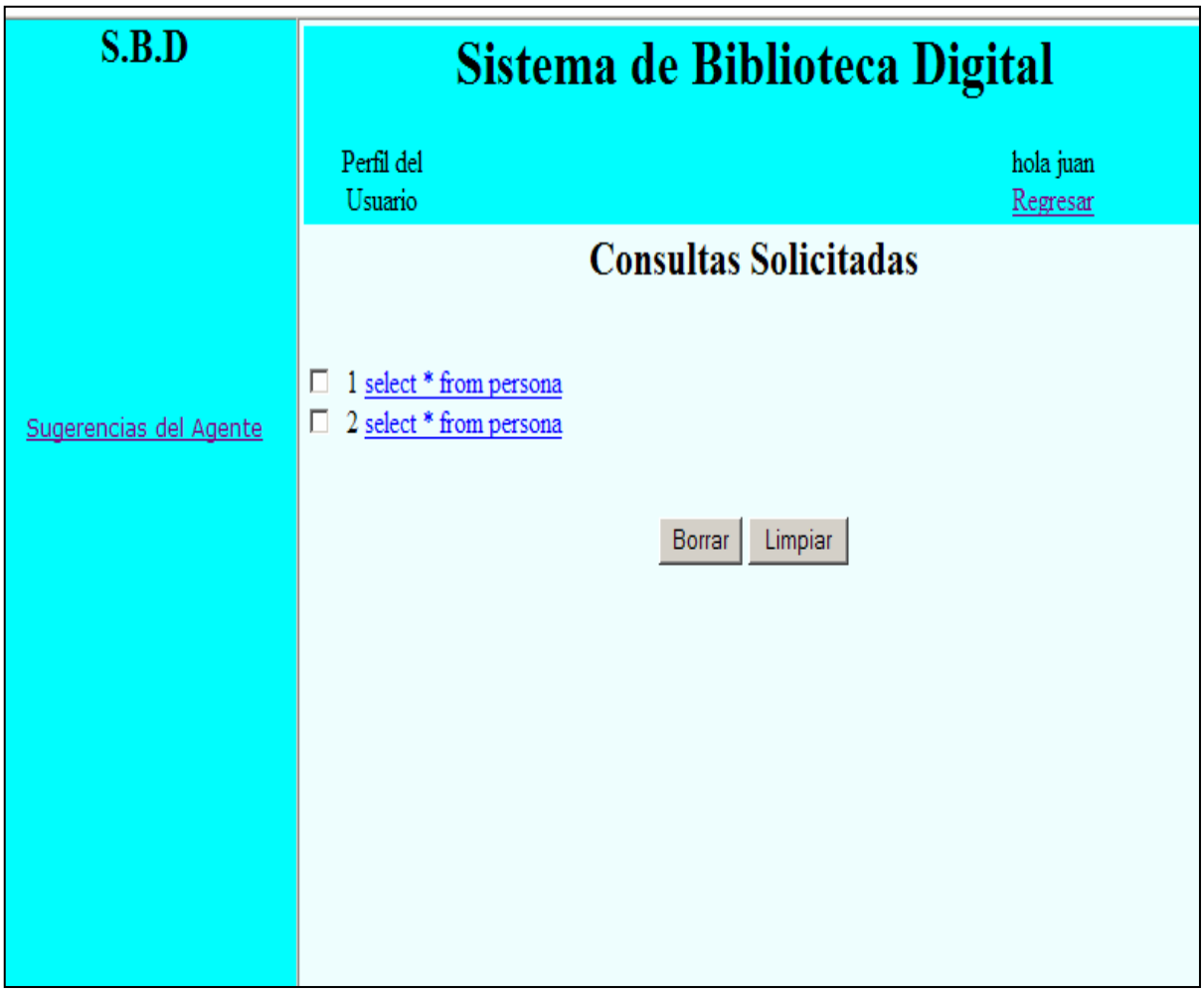


Figura 19. Interfaz de las ligas de la Respuesta de Consultas.

En esta pantalla se mantiene el nombre del usuario con la opción de desconectar, el perfil del usuario, pero aparece un nuevo cuadro de texto identificado como respuestas

de consultas, en donde se ve la consulta y en la parte de abajo la respuesta correspondiente.

Si el usuario oprime el botón borrar, se le permitirá eliminar las respuestas que ya no deseé, esto será posible cuando el usuario marque la caja con el tema que desea borrar.

El botón regresar, al presionarlo se permitirá al usuario regresar a la pantalla principal del sistema.

En la figura 20, se muestran los resultados de la liga que se genera entre las respuestas y las consultas hechas por el usuario:

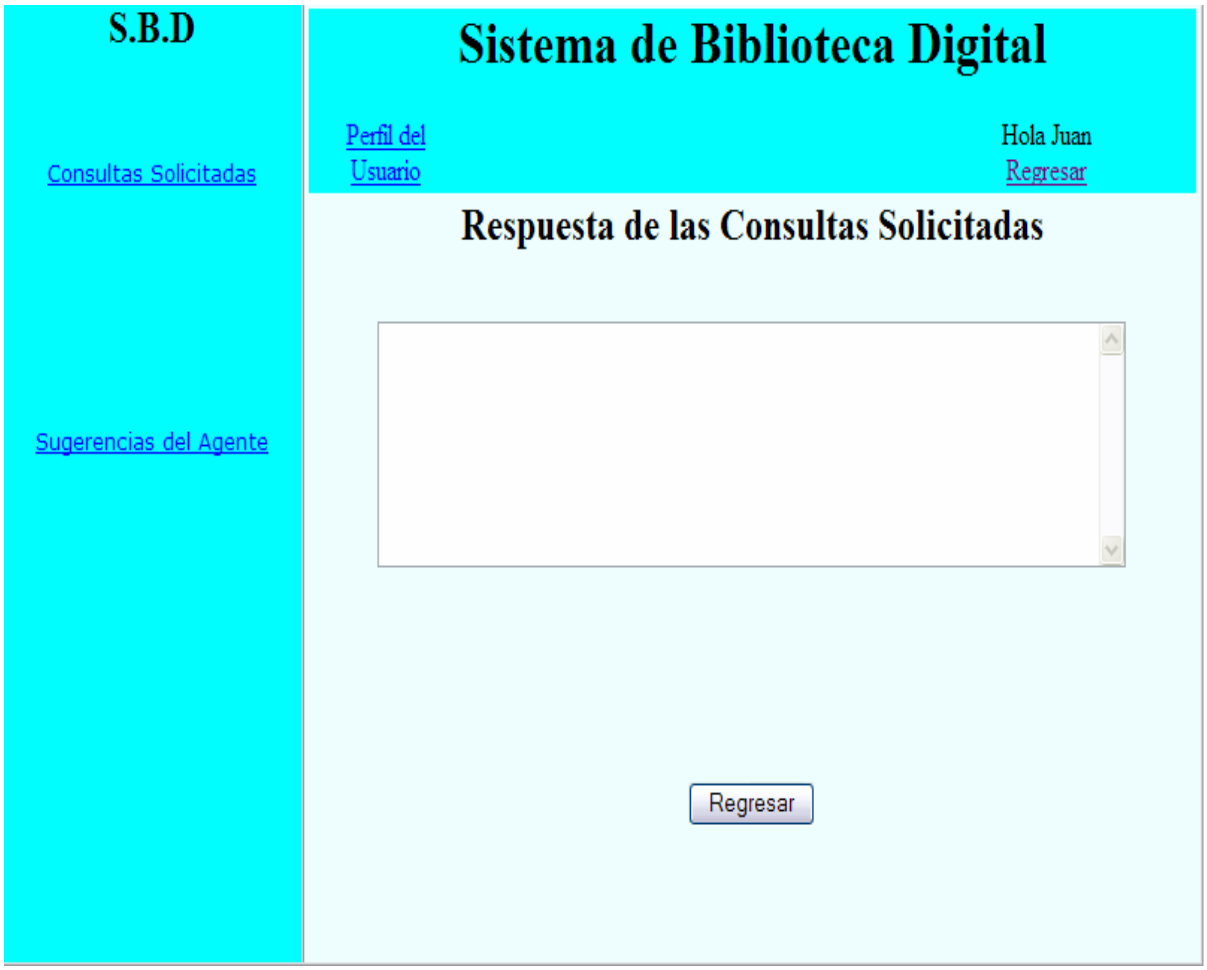


Figura 20. Interfaz de la Respuesta de Consultas.

En la pantalla de la figura 20, el usuario visualiza las respuestas que se tienen de la lista de las ligas de acuerdo a las consultas emitidas por los usuarios, es decir el usuario cuando selecciona una liga y la oprime se envía una respuesta y es la que se muestra en la figura 20, además el usuario si lo desea podrá seguir realizando otras consultas, y cuando oprima el botón REGRESAR regresara a la pantalla principal.

La liga SUGERENCIAS DEL AGENTE, al presionarla el usuario verá las respuestas que el agente le proporciona como sugerencia de acuerdo a los temas de su preferencia, como se puede ver en la figura 21, el usuario al tener esta lista de sugerencias realizará la selección de la respuesta que más se apegue a su interés:

Figura 21. Interfaz de las ligas de las Sugerencias de los Agentes.

Además el usuario tiene la opción de seguir realizando otras consultas si así lo requiere. El usuario al oprimir el botón BORRAR, podrá eliminar aquellas consultas que ya no sean de su interés, esto es posible seleccionando la caja con el tema que desee eliminar. El botón REGRESAR le permitirá al usuario volver a la interfaz principal del sistema.

En la figura 22, el usuario visualizará las respuestas de los temas de su interés, en el momento que seleccione una liga para ver el contenido de la misma, el usuario puede realizar nuevas búsquedas si así lo desea.

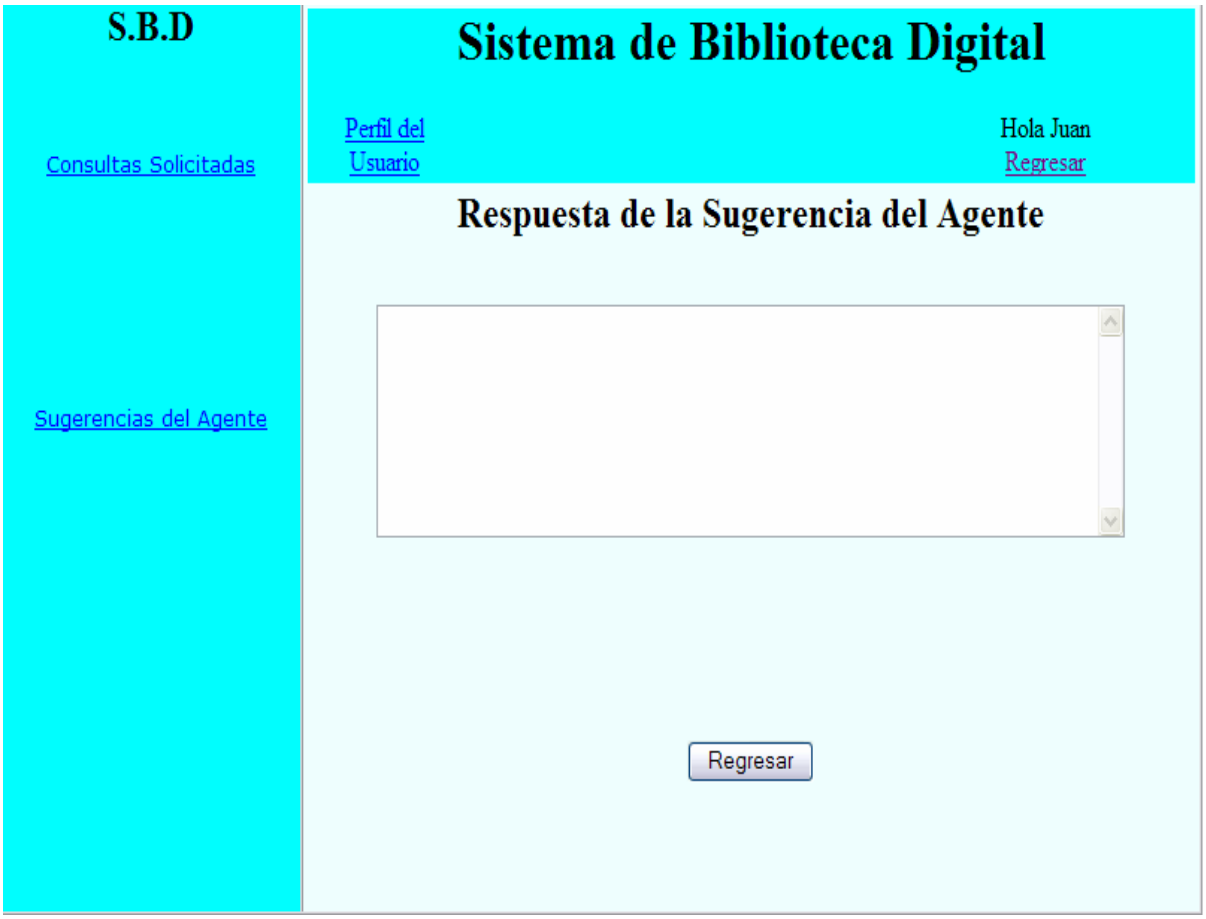


Figura 22. Interfaz de la Respuesta de Sugerencias del Agente.

En la figura 23, se le permite al usuario realizar los cambios en sus datos personales o bien modificar sus preferencias de salida de la información, la forma en la que desea que se le envié la información y los temas de la ontología. Al presionar el botón

EVIAR DATOS, se guardan los datos que fueron modificado en la base de datos y al momento de ejecutar esa tarea, el usuario regresa a la interfaz principal del sistema, pero si el usuario oprime el botón BORRAR LOS DATOS, el proceso de modificación ya no se ejecuta por lo que el usuario regresa a la interfaz principal.

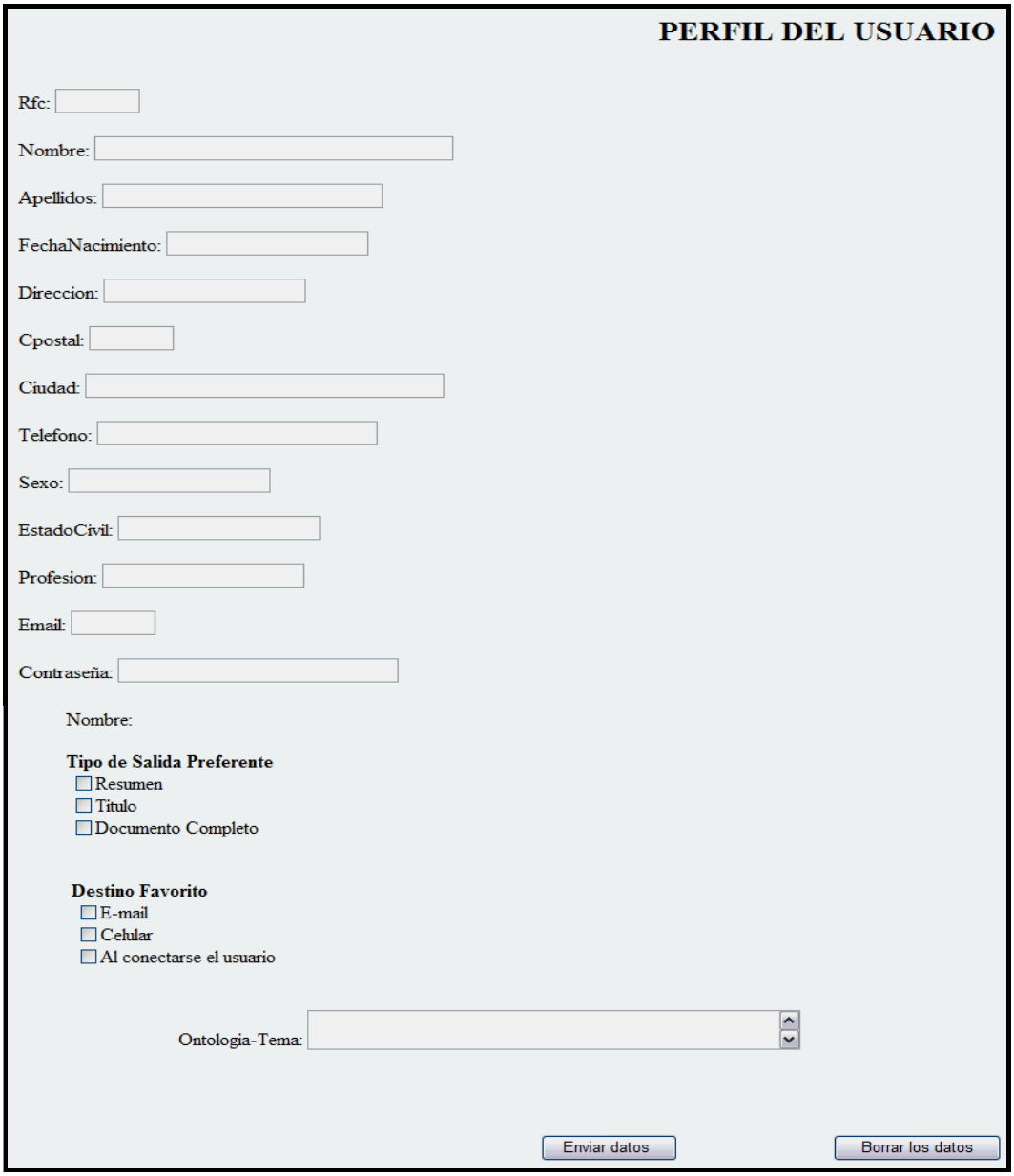


Figura 23. Interfaz del Perfil del Usuario.

En la *Ontología-Tema* el usuario puede realizar una selección de los temas que sean de su interés, para que posteriormente se lleve a cabo una consulta. La ontología esta representada en forma de árbol, como se muestra en la figura 24:

Tema1
- Subtema 1-1
- - Subtema 1-1-1
- - Subtema 1-1-2
- Subtema 1-2

Tema 2
- Subtema 2-1
- Subtema 2-2
- - Subtema 2-2-1
- - Subtema 2-2-2
- Subtema 2-3
- - Subtema 2-3-1
- - Subtema 2-3-2

Figura 24. Ontología de Temas.

Se muestra un árbol de conceptos donde los temas y subtemas se especifican mediante guiones, que indican las relaciones de subordinación y coordinación, como se muestra en la figura 24. Los temas del primer nivel forman los temas de la raíz, y cada uno de ellos tendrá sus subtemas, sub-subtemas y así sucesivamente.

Por otra parte el sistema permite registrar en la base de datos las referencias que comprenden el acervo bibliográfico.

En este capítulo se realizó la fase de Análisis, Diseño y Construcción del sistema, el análisis es una de las partes más importantes, debido a que se recopila toda la información para identificar y diagnosticar el problema. En la parte de Diseño, se esquematizó la

arquitectura del Sistema, dónde, básicamente se muestra el funcionamiento de esté, a través de las entradas, procesos y salidas. En la construcción, se llevo a cabo la creacion del Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes, desarrollando cada parte de la programación en este caso el agente, la caché, entre otros programas que fueron creados para su funcionamiento.

En el siguiente capítulo, se presentan las pruebas y resultados del sistema.

CAPÍTULO 4
PRUEBAS Y RESULTADOS

En este capítulo, se describen los requerimientos del sistema para instalarlo y se presentan los resultados obtenidos de las pruebas realizadas usando los tres casos propuestos: el modelo cliente/servidor a través de una conexión constante, cliente/servidor con el uso de la tecnología de servlets y el uso de la arquitectura ACO.

4.1 REQUERIMIENTOS PARA INSTALAR Y EJECUTAR EL SISTEMA

El sistema de Control de Concurrencia a una Base de Datos Mediante Agentes implantado mediante la arquitectura ACO tiene los requerimientos siguientes:

a) Requerimientos de Hardware:

1) Procesador Pentium IV a 1.8 GHZ para el servidor.
2) 256 MB en RAM mínimo, por el uso de un manejador de base de datos y el servidor Tomcat.
3) Espacio en disco duro mínimo 20 GB, dado que se almacena el sistema y la base de datos. Este tamaño debe ser congruente con el de la base de datos.
4) Resolución de 600 * 800 píxeles, para la visualización de las paginas Web.
5) Una tarjeta de red en cada computadora.

b) Software para instalar:

1) Sistema Operativo Windows XP o Superior.
2) Tomcat 4.1
3) Java 1.5

4.1.1 INSTALACIÓN DEL SOFTWARE EN EL SERVIDOR

En el servidor se debe instalar primeramente JDK (Java Development Kit), en este caso se uso la versión jdk1.5.0_06. A continuación se debe instalar el servidor Tomcat de servlets. En las pruebas se utilizó la versión 4.1 de tomcat.

▶ Para instalar el JDK de Java se obtiene el archivo con el nombre jdk1.5.0_06, y se hace doble "Click" con lo que se inicia la instalación de Java.

▶ En uno de los pasos se solicita el plug-in para visualizar los "applets" en los navegadores de Internet, se recomienda utilizar el que se despliega.

▶ Una vez terminada la instalación de Java, se verifica que la variable de entorno JAVA-HOME, identifique el directorio que se utiliza para ejecutar Java.

➔La instalación de tomcat 4.1, se hace con doble "Click", en el archivo tomcat 4.1.exe, con lo cual se descomprimen los archivos necesarios y se instalan en el directorio apropiado.

➔Hecho lo anterior, se puede acceder a tomcat de la siguiente manera: Inicio/All Programs/Apache Tomcat 4.1/Start tomcat, lo que queda activo es el servidor de servlets.

⇨La instalación del Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes (Arquitectura ACO) se hace copiando los archivos del sistema en los directorios apropiados del servidor tomcat.

⇨En la dirección tomcat C:\Program Files\Apache Tomcat 4.0\webapps\examples\WEB-INF\classes, se almacenan los archivos .class

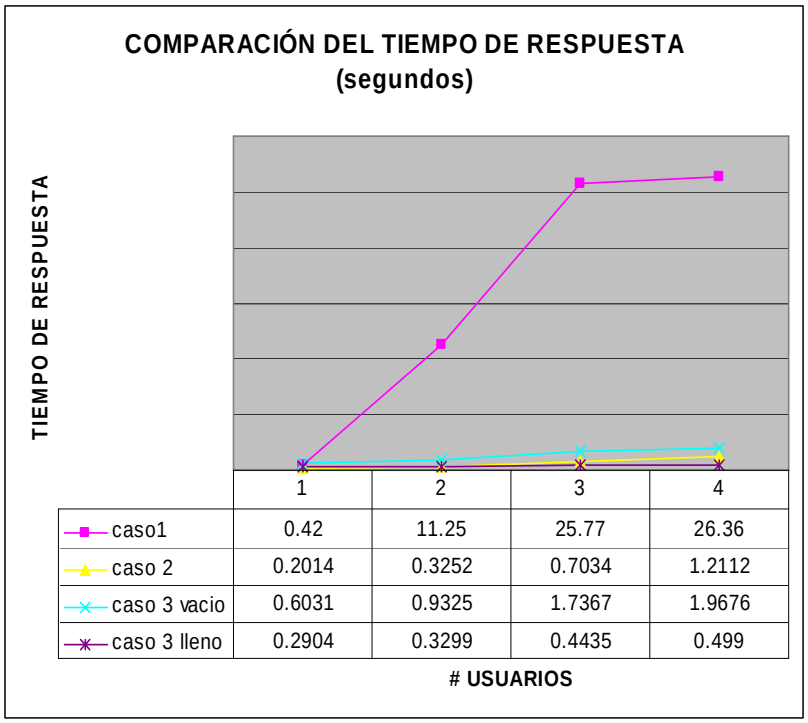
⇨En la dirección C:\Program Files\ApacheTomcat 4.1\webapps\ROOT, alojar todos los archivos del sistema que fueron creados en ambiente web, por mencionar algunos, estos serian, Usuario.html, Nuevo Usuario.html, Sugerencias del Agente.html, etc.

⇨Una vez que se tenga en tomcat los archivos colocados en las carpetas que les corresponde, se ejecuta el servidor tomcat.

⇨Cuando ya se tenga el servidor tomcat corriendo, se procede a realizar lo siguiente. Se abre el explorador y se teclea lo siguiente: http://localhost:8080/usuario.html, con esta instrucción se ejecuta el Sistema de Control de Concurrencia a una Base de Datos Mediante Agentes.

4.2 CASOS DE PRUEBA

En esta sección se presentan los resultados de las pruebas en las cuales se comparan los dos casos contra la arquitectura ACO desarrollada en esta tesis (caso tres). Los resultados de la figura 25, muestran que al utilizar la arquitectura ACO se logra una mejora significativa respecto al modelo cliente/servidor (caso uno), compárese la línea rosa contra la azul y morada.



	1	2	3	4
caso1	0.42	11.25	25.77	26.36
caso 2	0.2014	0.3252	0.7034	1.2112
caso 3 vacio	0.6031	0.9325	1.7367	1.9676
caso 3 lleno	0.2904	0.3299	0.4435	0.499

Figura 25. Comparación del Tiempo de Respuesta para los tres casos

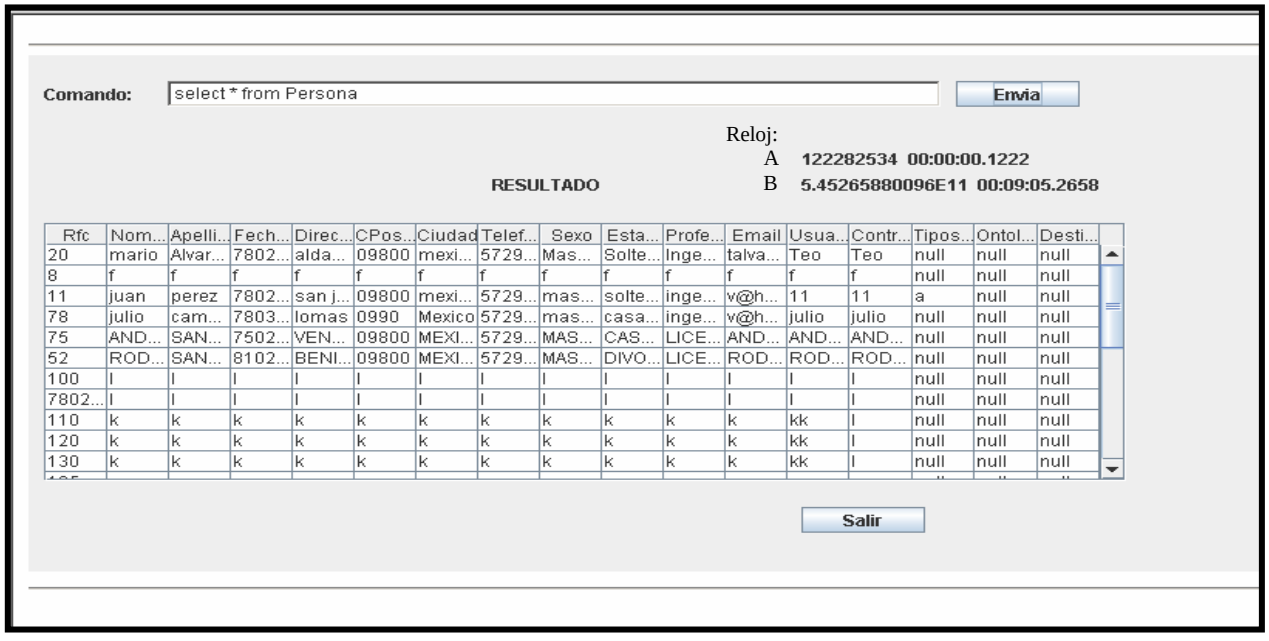
En la figura 25, se observa también que el caso 2 utilizando Servlets es mejor cuando la arquitectura ACO atiende una consulta por primera vez y la almacena en la Caché, pero para las posteriores consultas se observa que se logra un mejor tiempo de respuesta para los usuarios que hacen la misma consulta.

En las pruebas realizadas se emplearon los mismos datos en los tres casos para que la comparación se hiciera con las mismas entradas y condiciones de ejecución.

En la primera condición se emplearon conjuntos de datos variando el número de usuarios manteniendo constante el tamaño de las consultas. En la figura 1 los grupos de usuarios 1, 2, 3 y 4 corresponden a 1, 25, 50 y 100 usuarios concurrentes simulados mediante un proceso que genera clientes. La decisión de utilizar hasta 100 usuarios fue en base que el sistema Unicorn que se utilizó como motivación de este trabajo, presenta problemas de conexión y tiempo de respuesta a partir de 60 usuarios concurrentes.

Las pruebas se realizaron en el servidor Pentium IV HT. con 1GB en RAM.

El caso 1 ver figura 26, es la aplicación cliente/servidor con conexión constante que hasta ahora continúa en el mercado mundial de la informática.



Rfc	Nom...	Apelli...	Fech...	Direc...	CPos...	Ciudad	Telef...	Sexo	Esta...	Profe...	Email	Usua...	Contr...	Tipos...	Ontol...	Desti...
20	mario	Alvar...	7802...	alda...	09800	mexi...	5729...	Mas...	Solte...	Inge...	talva...	Teo	Teo	null	null	null
8	f	f	f	f	f	f	f	f	f	f	f	f	f	null	null	null
11	juan	perez	7802...	san j...	09800	mexi...	5729...	mas...	solte...	inge...	v@h...	11	11	a	null	null
78	julio	cam...	7803...	lomas	0990	Mexico	5729...	mas...	casa...	inge...	v@h...	julio	julio	null	null	null
75	AND...	SAN...	7502...	VEN...	09800	MEXI...	5729...	MAS...	CAS...	LICE...	AND...	AND...	AND...	null	null	null
52	ROD...	SAN...	8102...	BENI...	09800	MEXI...	5729...	MAS...	DIVO...	LICE...	ROD...	ROD...	ROD...	null	null	null
100	l	l	l	l	l	l	l	l	l	l	l	l	l	null	null	null
7802...	l	l	l	l	l	l	l	l	l	l	l	l	l	null	null	null
110	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null
120	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null
130	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null

Figura 26. Pantalla donde se muestra el tiempo de respuesta para el caso 1.

Los elementos de la pantalla de la figura 26 son:

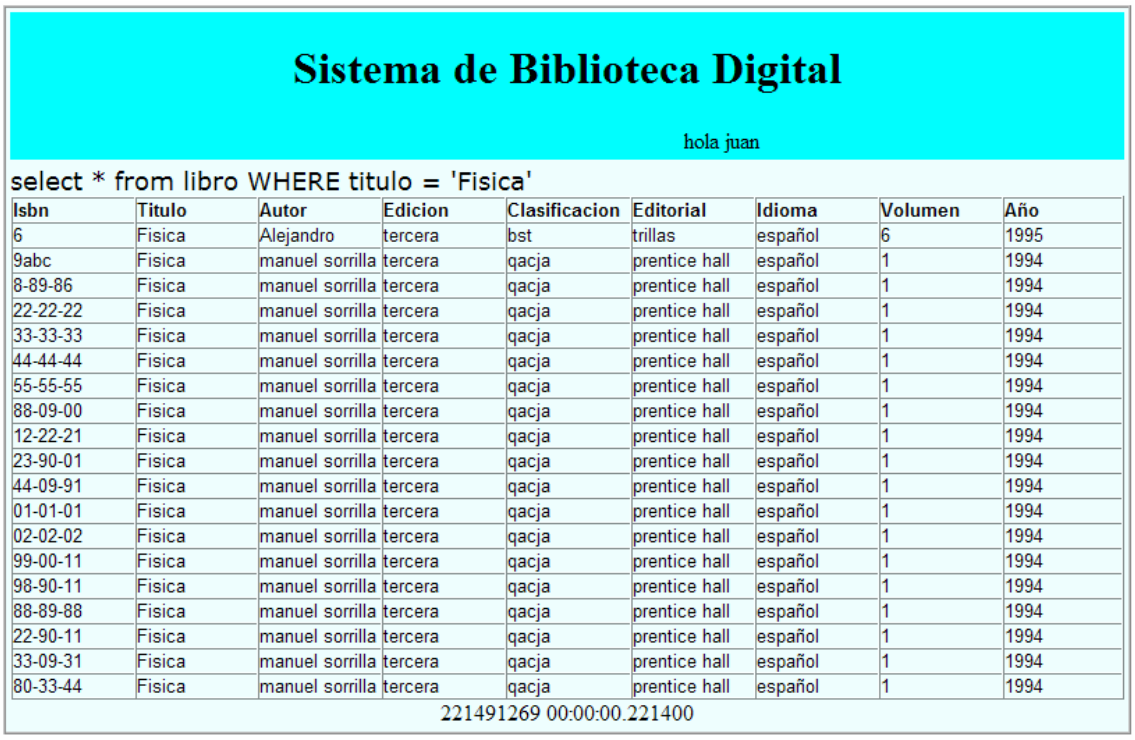
a. Comando: en ella el usuario escribe la consulta que desea buscar en la base de datos del sistema.
b. Botón Envía: Este botón al momento en que el usuario le da "Click" lanza consulta que fue introducida en la caja de texto.
c. Reloj: Aquí se muestran dos relojes con diferentes tipos de tiempos estos se identifican como A y B.

   Reloj A, muestra el tiempo que tarda un usuario en el sistema para dar una respuesta de la consulta que se realizo.

   Reloj B, muestra el tiempo que un usuario permanece en el sistema, es decir desde el momento que se conecto para realizar su consulta y obtuvo su resultado, hasta que abandona la sesión.
d. Resultado: En la cajita de texto lo que el usuario visualiza son los resultados que se obtienen de acuerdo con la consulta que escribió.
e. Botón Salir: Cuando el usuario oprime este botón le permitirá abandonar la sesión en el sistema.

El segundo caso ver figura 27, es una arquitectura cliente servidor a través de Servlets que se explicó en el capítulo tres. Esta tecnología está disponible para el desarrollo de aplicaciones, en esta su principal ventaja es que la conexión no es constante. El servidor que soportó la ejecución de los Servlets es Apache Tomcat 4.1. Como se observó brinda ventajas también en el tiempo de respuesta respecto al modelo cliente/servidor tradicional.

En el tercer caso corresponde a la arquitectura ACO, en la cual, cada consulta se atiende mediante un Agente Gestor de la Base de Datos. Este agente cuenta con el acceso a una memoria Caché con capacidad de 500 entradas e implementa un algoritmo para renovar las consultas menos populares con el que se garantiza que se elimina la consulta con menos accesos sin eliminar las que llegan al último. Las pruebas como se comentó arriba muestran la ventaja de utilizar esta arquitectura cuando las consultas se encuentran en la caché.



Sistema de Biblioteca Digital

hola juan

select * from libro WHERE titulo = 'Fisica'

Isbn	Titulo	Autor	Edicion	Clasificacion	Editorial	Idioma	Volumen	Año
6	Fisica	Alejandro	tercera	bst	trillas	español	6	1995
9abc	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
8-89-86	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
22-22-22	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
33-33-33	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
44-44-44	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
55-55-55	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
88-09-00	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
12-22-21	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
23-90-01	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
44-09-91	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
01-01-01	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
02-02-02	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
99-00-11	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
98-90-11	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
88-89-88	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
22-90-11	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
33-09-31	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
80-33-44	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994

221491269 00:00:00.221400

Figura 27. Pantalla donde se muestra el tiempo de respuesta en la parte inferior en las pruebas del caso 2.

Las consultas en la Arquitectura ACO tienen un costo adicional respecto al caso 2 de los Servlets debido a la búsqueda que se debe de hacer en la memoria Caché antes de determinar que se tiene que acceder a la base de datos y entonces se suma el tiempo de respuesta del acceso a la base de datos al de la búsqueda en la Caché, pero una vez que la consulta ya se encuentra en la Caché, entonces el tiempo de acceso se mantiene casi constante, solamente afectado por la carga de trabajo sobre el procesador. En la figura 28, se muestra una pantalla donde se despliegan los resultados de una consulta en el caso 3

Isbn	Titulo	Autor	Edicion	Clasificacion	Editorial	Idioma	Volumen	Año
22-09-01	matematicas	luis	segunda	qast	prentice hall	ingles	2	1993
6	Fisica	Alejandro	tercera	bst	trillas	español	6	1995
9	Literatura	Manuel	primera	q2345	trillas	ingles	9	2001
9abc	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
8-89-86	Fisica	manuel sorrilla	tercera	qacja	prentice hall	español	1	1994
98-92-100	matematicas	luis	segunda	qast	prentice hall	ingles	2	1993
90-90-1-2	matematicas	luis	segunda	qast	prentice hall	ingles	2	1993
87-78-90	Teoria General de Sistemas	John P. van Gigch	2a. Edicion	q295g54	trillas	español	1	1997
78-89-02	Teoria General de Sistemas	John P. van Gigch	2a. Edicion	q295g54	trillas	español	1	1997
23-89-00	Teoria General	John P.	2a.					

Figura 28. Pantalla donde se muestra el tiempo de respuesta para el caso 3 con 100 usuarios.

4.2.1 VENTAJAS Y DESVENTAJAS DE LA ARQUTECTURA ACO

La Tabla 9, compara las ventajas y desventajas de las arquitecturas presentadas en los tres casos de prueba 1 y 2 respecto a la Arquitectura ACO.

Arquitectura	Ventajas	Desventajas
Cliente/Servidor, con conexión Constante	Es una arquitectura común.	Atiende un número limitado de usuarios concurrentemente por su conexión constante.
Cliente/Servidor, con la tecnología de Servlets	Las conexiones en los clientes no están limitadas porque hay desconexión en cada consulta.	Se requiere un servidor para su funcionamiento. El tiempo de respuesta crece conforme aumenta el número de usuarios.
Arquitectura ACO	Mejora el tiempo de respuesta cuando la consulta se encuentra en la Caché. Permite que un usuario especifique varias consultas al mismo tiempo, mismas que se procesan concurrentemente.	El tiempo de respuesta para una consulta que no está en la Caché tarda algunas centésimas de segundo más por la búsqueda en la Caché antes del acceso a la Base de Datos.

Tabla 9. Ventajas y Desventajas de la arquitectura ACO

CONCLUSIONES

En esta tesis se desarrollo un trabajo motivado en la solución de un problema real con el tiempo de respuesta en un sistema de Biblioteca Digital. El sistema propuesto de control de concurrencia mediante agentes está implementado mediante una arquitectura que utiliza Agentes, una memoria Caché y una Ontología para mejorar el tiempo de respuesta a un grupo de usuarios conectados concurrentemente a una Base de Datos de una Biblioteca Digital. En esta Arquitectura lo importante es que las transacciones a la base de datos sean primordialmente de Consulta y sean mínimas las de Actualización. El motivo de esto es que las operaciones de actualización provocan que las entradas en la Caché se invaliden. En este trabajo no se consideraron las actualizaciones para centrarnos en la mejora a los tiempos de respuesta.

La Arquitectura ACO utiliza un Agente Gestor de la Base de Datos para almacenar en la memoria Caché las consultas más frecuentes, decidiendo la eliminación de la menos popular cuando se llena y es llega una consulta nueva.

Las pruebas realizadas permiten observar que la Arquitectura ACO mejora el tiempo de respuesta respecto a la arquitectura Cliente/Servidor tradicional y solamente mejora a la arquitectura de Servlets cuando las consultas se encuentran en la Caché. A pesar de esto si se recomienda el uso de la Arquitectura ACO porque hay un mayor número de usuarios que encuentran su respuesta en la Caché y con esta se mejora el tiempo de respuesta respecto a los casos anteriores.

TRABAJOS A FUTURO

Existen varios aspectos que se han encontrado durante el desarrollo de este trabajo y que requieren de esfuerzo para llevarlos a un producto final. Entre estos se tiene:

a) Probar la interfase con otras bases de datos existentes, por ejemplo, Datawarehouse.
b) Ampliar la arquitectura del sistema de control de concurrencia para aplicarla a una base de datos distribuida.
c) Modificar al Agente-usuario para una búsqueda en una base de datos en la Web, por ejemplo conectarlo con google.

REFERENCIAS Y BIBLIOGRAFÍA

[1] Abraham Silberschaz, Henry F. Korth S. Sudarshan, Fundamentos de Bases de Datos, 3a Edición, Mc Graw Hill 1998.

[2] Andrew S. Tanenbaum, Sistemas Operativos Modernos, Editorial Prentice Hall Hispanoamericana, S.A. 1a Edición 1993

[3] C.J. Date, 2001, Introducción a los sistemas de bases de datos. Séptima Edición. Pearson Educación, México D.F.

[4] John P. Van Gigch, Teoría General de Sistemas, Editorial Trillas, Mexico, 1997.

[5] Thomas M. Connolly, Carolyn E. Begg, Addison Wesley, Sistemas de Bases de Datos.

[6] Agentes Inteligentes en Educación, Gonzalo Villarreal Farah, Centro Comenius Universidad de Santiago de Chile, gvillarr@comenius.usach.cl

[7] Baecker, R. M., editor (1993). *Readings in Groupware and Computer-Supported Cooperative Work.* Morgan Kaufmann Publishers: San Mateo, CA.

[8] Baecker, R. M., editor (1993). *Readings in Groupware and Computer-Supported Cooperative Work.* Morgan Kaufmann Publishers: San Mateo, CA.

[9] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. *Ontobroker: The Very High Idea.* Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, May 1998.

[10] Cacheand Query for Wide Area Sensor Databases Amol Deshpande*y;* Suman Nath*z;* Phillip B. Gibbons Srinivasan Seshan*z;* amol@cs.berkeley.edu sknath@cmu.edu, phillip.b.gibbons@intel.com srini@cmu.edu, Intel Research Pittsburgh *y* U.C. Berkeley *z* Carnegie Mellon University.

[11] Esther Sánchez Lucas y Rosa Martínez Rubio, Ontologías http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18 (consultado 5-12-2000)

[12] Galindo, L., "Una Metodología para el Desarrollo de Sistemas de Información Basados

en Computadoras", Memorias del 6ª Congreso Nacional de Ingeniería Electromecánica y de Sistemas. SEPI, ESIME – Z, IPN. Págs. 698-708, Mexico, D.F., Mexico, Noviembre 2001.

[13] Gerhard Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. 2nd printing, MIT Press, 2000.

[14] Greif, I. (1994). Desktop agents in group-enabled products. *Communications of the ACM,* 37(7).

[15] *Implicit*: An Agent Based Recommendation System for Web Search, Alexander Birukov Department of Information and Communication Technology University of Trento -Italy birukou@dit.unitn.it Enrico Blanzieri Department of Information and Communication Technology University of Trento -Italy enrico.blanzieri@unitn.it Paolo Giorgini Department of Information and Communication Technology University of Trento -Italy paolo.giorgini@unitn.it

[16] James P. Fry, Edgar H. Sibley, Evolution of Data-Base Management Systems. *Departme.nt of Information Systems Mana˜eme˜lt, University of Maryland, College Park, Maryland 207, and National Bureau of Standards, Washington, D.C. 20285.*

[17] James R. Chen, Nathalie Mathé & Shawn Wolfe. Collaborative Information Agents on the World Wide Web. ACM, 1998.

[18] Jeff Heflin. *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment.* Ph.D. Thesis, University of Maryland, College Park, 2001.

[19] Jesús-Manuel Olivares-Ceja, A Purposeful Agents Interaction Model, Mixed Ontologies and Unexpected Events (PhD Theses in Spanish), Research Center for Computing at The National Polytechnique Institute, Mexico City, 2002

[20] Jonathan J. King Special Issue on Al and Database Research

[21] Lieberman, H. (ed): Your Wish is my Command. Programming By Example. Morgan Kaufmann Publishers. Academic Press, USA. 2001.

[22] Lilia Esther González Rodríguez, Sistema de Información para el Apoyo al Seguimiento de los Egresados de un Programa de Postgrado.

[23] Maes, P. (1994). Social interface agents: Acquiring competence by leaming from users and other agents. In Etzioni, O., editor, *Sofware Agents - Papers from the 1994 Spring Symposium (Technical Report SS-94-03)*. AAAI Press.

[24] Naser S. Barghouti And Gail E. Kaiser Concurrency Control in Advanced Database Applications

[25] Ontological User Profiling in Recommender Systems, Stuart E. Middleton, Nigel R. Shadbolt, And David C. De Roure, University of Southampton

[26] Philip A. Bernstein and Nathan Goodman, Concurerncy Control in Dristributed Database Systems, Computer Coorporation of America, Cambrudge, Massachusetts 02139.

[27] Recommender Systems Stuart E. Middleton, Nigel R. Shadbolt, and DAVID C. DE ROURE, University Of Southampton.

[28] Richard Hull, Roger King, Semantic Database Modeling: Survey, Applications, and Research Issues. *Computer Science Department, University of Southern California, Los Angeles, California 90089-0782.*

[29] Stefan Decker, Michael Erdmann, Dieter Fensel and Ridi Studer. *Ontobroker: Ontology based Access to Distributed and Semi-Structured Information.* In R. Meersman et al. (eds.), Semantic Issues in Multimedia Systems, Kluwer Academic Publisher, Boston, 1999.

[30] Sugiura, A.; Koseki, Y.: Internet Scrapbook: Automating Web browsing tasks by programming-by-demonstration. Proceedings of the 7th International WWW Conference, Brisbane, Australia, April 1998. Elsevier Science.

[31] Thomas R. Gruber, Toward Principles for the Design of Ontologies Used for Knowledge Sharing *en Formal Ontology in Conceptual Analysis and Knowledge Representation*, edited by Nicola Guarino and Roberto Poli, Kluwer Academic Publishers, Italy 1993.

[32] Verena Kantere, Aris Tsois, Using ECA Rules to Implement Mobile Query Agents for Fast-Evolving Pure P2P Database Systems, Knowledge and Database Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens.

[33] Wilhelm Hasselbring, *Tilburg University*
Programming Languages and Systems for Prototyping Concurrent Applications

[34] Zick, Laura. The Work of Information Mediators: A Comparison of Librarians and Intelligent Software Agents. For CSCI 590: Intelligent Systems, IUPUI. 1999.

REFERENCIAS DE INTERNET

[35] Agentes inteligentes: definicion y tipologia. Los agentes de información, Pedro Hípola y Benjamín Vargas-Quesada.
http://www.elprofesionaldelainformacion.com/contenidos/1999/abril/agentes_inteligentes_definicion_y_tipologia_los_agentes_de_informacion.html

[36] Agentes Taxonomía y Aplicaciones
http://72.14.203.104/search?q=cache:WTlsAOQhqDYJ:www.it.uc3m.es/liliana/paginas/masinfo/agentes.doc+definicion+de+agente+con+gestion+de+estimulo&hl=es&gl=mx&ct=clnk&cd=37

[37] http://www.cienciasmisticas.com.ar/informatica/hardware/memorias/index.php

[38] http://demo.imh.es/Electroneumatica/Ud03/modulos/m_en001/glosario.htm
[39] http://enciclopedia.us.es/index.php/Latencia

[40] http://es.wikipedia.org/wiki/Tomcat

[41] http://es.wikipedia.org/wiki/Radix_sort

[42] http://es.wikipedia.org/wiki/Ley_de_Amdahl

[43] http://es.wikipedia.org/wiki/Redes_P2P

[44] El lío del proxy-caché *Antonio Caravantes*

[45] FIPA http://www.fipa.org/. 2001

[46] Harjinder, S. Gill. Prakash, C. Rao. Data Warehousing. La Integracion de Informacion para la Mejor Toma de Decisiones. Prentice Hall. Mexico, 1996. http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap8-1.html

[47] http://www.informatica.us.es/~ramon/tesis/CORBA/Seminario-MASIF/

[48] InterSystems Spain, 28108 Alcobendas. Madrid, www.InterSystems.es

[49] José Camilo Daccach T. contacto@deltaasesores.com, http://www.gestiopolis.com/delta/prof/PRO260.html

[50] http://www.mastermagazine.info/definicion/3868.php

[51] http://www.microsoft.com/spain/technet/recursos/articulos/Glossary.mspx

[52] On the Development of Data Models", en M. L. Brodie, J. Mylopoulos & J. W. Schmidt (eds.), 19-47. http://elies.rediris.es/elies9/4-2.htm

[53] http://www.osmosislatina.com/java/componentes.htm

[54] http://64.233.187.104/search?q=cache:37JXIlyAmasJ:imaginatica.eii.us.es/2005/agentes/info/material-botia/jade.pdf+definicion+FIPa&hl=es&gl=mx&ct=clnk&cd=2

[55] http://www.spinec.org/?m=200605

[56] Wikipedia, Memoria Cache

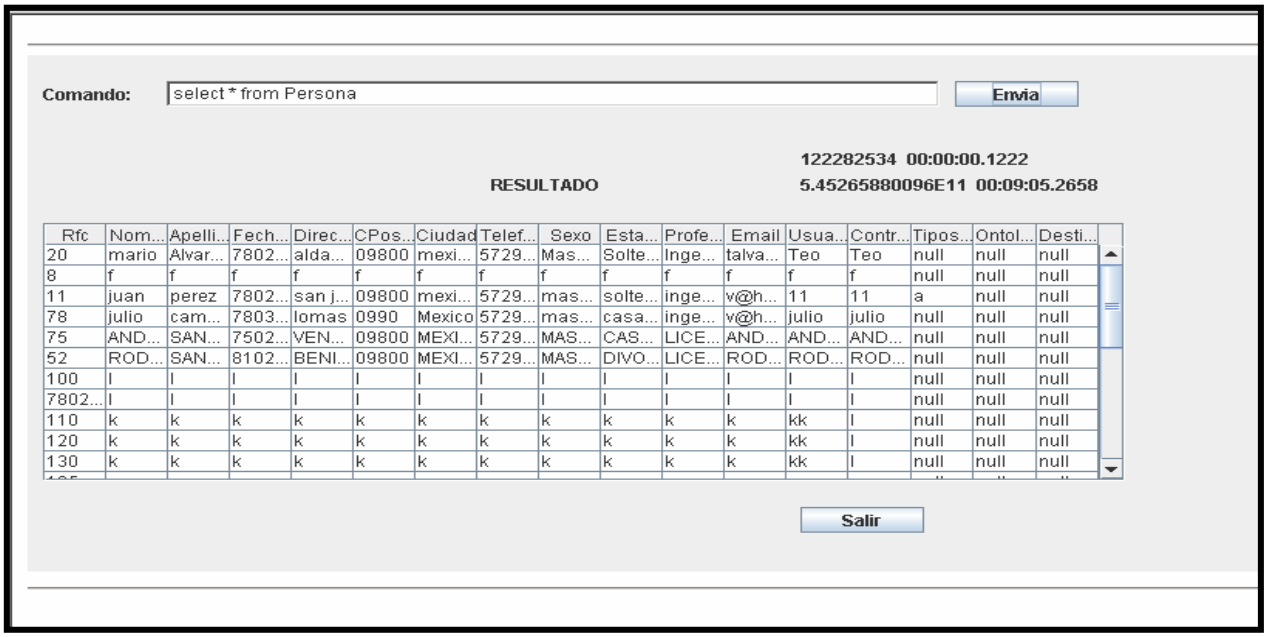
[57] Wooldridge, M. y Jennings, N. R., 1995 *Intelligent agents: theory and practice.* *http://pattie.wwww.media.mit.edu/people/pattie/CHI97/sld001.htm*

[58] http://www.monografias.com/trabajos28/teoria-sistemas/teoria-sistemas.shtml

[59] Wooldridge, M. y Jennings, N. R., 1995 Intelligent agents: theory and practice. http://pattie.www.media.mit.edu/people/pattie/CHI97/sld001.htm

ANEXO A

CASO UNO, MODELO CLIENTE/SERVIDOR CON CONEXIÓN CONSTANTE



Rfc	Nom...	Apelli...	Fech...	Direc...	CPos...	Ciudad	Telef...	Sexo	Esta...	Profe...	Email	Usua...	Contr...	Tipos...	Ontol...	Desti...
20	mario	Alvar...	7802...	alda...	09800	mexi...	5729...	Mas...	Solte...	Inge...	talva...	Teo	Teo	null	null	null
8	f	f	f	f	f	f	f	f	f	f	f	f	f	null	null	null
11	juan	perez	7802...	san j...	09800	mexi...	5729...	mas...	solte...	inge...	v@h...	11	11	a	null	null
78	julio	cam...	7803...	lomas	0990	Mexico	5729...	mas...	casa...	inge...	v@h...	julio	julio	null	null	null
75	AND...	SAN...	7502...	VEN...	09800	MEXI...	5729...	MAS...	CAS...	LICE...	AND...	AND...	AND...	null	null	null
52	ROD...	SAN...	8102...	BENI...	09800	MEXI...	5729...	MAS...	DIVO...	LICE...	ROD...	ROD...	ROD...	null	null	null
100	l	l	l	l	l	l	l	l	l	l	l	l	l	null	null	null
7802...	l	l	l	l	l	l	l	l	l	l	l	l	l	null	null	null
110	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null
120	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null
130	k	k	k	k	k	k	k	k	k	k	k	kk	l	null	null	null

```
import java.lang.Math;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
import java.net.*;
import java.text.*;
import javax.swing.table.*;
import java.util.*;


public class Comandoenvia extends JApplet
//implements Observer
{
TextField comando  = new TextField(80);
JTextArea resultado = new JTextArea("" ,20 ,80);
JLabel eComando = new JLabel("Comando:");
JLabel eResultado = new JLabel("RESULTADO");
JLabel eAviso = new JLabel("AVISO");
JButton enviar = new JButton("Envia");
JLabel eConexion = new JLabel("TX 00:00:00");
```

```
JLabel eDuracion = new JLabel("TC 00:00:00");
double tConexion = 0;
long inicioConexion = 0, finConexion = 0;
JButton salir = new JButton("Salir");
Socket conexion = null;
DataInputStream is ;
PrintStream os ;
String entra ,sale ;


DefaultTableModel modelo = new DefaultTableModel(0 ,1);
JTable tabla ; // = new JTable(modelo);
JScrollPane panelScroll = new JScrollPane(tabla);

ActionListener accion = new ActionListener()
{
 public void actionPerformed(ActionEvent e)
 {
   try
   {
   long inicio ,fin ,duracion ;
   double decimas ,decimasc ;

       inicio = System.nanoTime();
       envia();
       fin = System.nanoTime();
       duracion = fin - inicio;
       decimas = ( (long)(((double)duracion / 1000000000D) * 10000) ) / 10000D;

 NumberFormat precision4 = NumberFormat.getInstance();
       precision4.setMaximumFractionDigits(4);
       precision4.setMinimumFractionDigits(4);
   String dato = precision4.format(decimas);
       eDuracion.setText(""+duracion+ "  00:00:" +
        ( decimas < 10 ? "0" : "") + dato);



       finConexion = System.nanoTime();
       tConexion = finConexion - inicioConexion;
       decimasc = ( (long)(((double)tConexion / 1000000000D) * 10000) ) / 10000D;
       int cxseg = (int)( (decimasc % 3600) % 60);
       int cxmin = (int)((decimasc % 3600) / 60);
       int cxhora = (int)(decimasc / 3600);
       double cxdecim = decimasc - (int)decimasc;

   String dato2 = precision4.format(cxseg+cxdecim);
       eConexion.setText(""+tConexion+ "  " +
       ( cxhora < 10 ? "0" : "") + cxhora + ":" +
       ( cxmin < 10 ? "0" : "") + cxmin + ":" +
       ( cxseg < 10 ? "0" : "") + dato2);

   }
   catch(IOException ex)
   {
```

```
        resultado.insert("FALLO LA CONEXION CON EL SERVIDOR" ,0);
    }
  }
};

ActionListener accionSalir = new ActionListener()
{
  public void actionPerformed(ActionEvent e)
  {
    long inicio ,fin ,duracion ;
    double decimas ,decimasc ;

 NumberFormat precision4 = NumberFormat.getInstance();
        precision4.setMaximumFractionDigits(4);
        precision4.setMinimumFractionDigits(4);


        finConexion = System.nanoTime();
        tConexion = finConexion - inicioConexion;
        decimasc = ( (long)(((double)tConexion / 1000000000D) * 10000) ) / 10000D;
        int cxseg = (int)( (decimasc % 3600) % 60);
        int cxmin = (int)((decimasc % 3600) / 60);
        int cxhora = (int)(decimasc / 3600);
        double cxdecim = decimasc - (int)decimasc;

    String dato2 = precision4.format(cxseg+cxdecim);
        eConexion.setText(""+tConexion+ "  " +
        ( cxhora < 10 ? "0" : "") + cxhora + ":" +
        ( cxmin < 10 ? "0" : "") + cxmin + ":" +
        ( cxseg < 10 ? "0" : "") + dato2);


  }
};

    public void init()
    {
    Container c = getContentPane();

      c.setLayout(null);

      c.add(eComando);
      eComando.setBounds(10 ,20 ,70 ,20);
      eComando.show();

      c.add(comando);
      comando.setBounds(90 ,20 ,500 ,20);
      comando.show();

      c.add(enviar);
      enviar.setBounds(600 ,20 ,80 ,20);
      enviar.show();

      c.add(eDuracion);
      eDuracion.setBounds(500 ,70 ,200 ,20);
```

```
    eDuracion.show();

    c.add(eConexion);
    eConexion.setBounds(500 ,90 ,200 ,20);
    eConexion.show();


    c.add(eResultado);
    eResultado.setBounds(300 ,90 ,100 ,20);
    eResultado.show();

    c.add(eAviso);
    eAviso.setBounds(100 ,110 ,300 ,20);
    eAviso.show();

    c.add(panelScroll);
    panelScroll.setBounds(10 ,130 ,700 ,200);
    panelScroll.show();

    c.add(salir);
    salir.setBounds(500 ,350 ,80 ,20);
    salir.show();

    // ASIGNA LA ACCION AL BOTON ENVIA
    enviar.addActionListener(accion);
    comando.addActionListener(accion);
    salir.addActionListener(accionSalir);
    inicioConexion = System.nanoTime();
  }

  public void envia() throws IOException
  {
   try
   {
      conexion = new Socket("localhost" ,7777);
      is = new DataInputStream(conexion.getInputStream());
      os = new PrintStream(conexion.getOutputStream());

      // ENVIA EL COMANDO
      os.println(comando.getText());
      os.flush();

   String sColumnas = is.readLine();
   String sRenglones = is.readLine();

   int columnas = Integer.parseInt(sColumnas);
   int renglones = Integer.parseInt(sRenglones);

   if( renglones == 0 )
   {
      eAviso.setText("NO HAY DATOS PARA ESTA CONSULTA");

  Container c = getContentPane();
     c.add(eAviso);
     eAviso.setBounds(100 ,110 ,300 ,20);
```

```
      eAviso.show();

      modelo = new DefaultTableModel(0 ,0);
      tabla = new JTable(modelo);
      c.add(panelScroll);
      panelScroll.setBounds(10 ,130 ,600 ,200);
      panelScroll.show();
      panelScroll.hide();
      return;
   }

   modelo = new DefaultTableModel(0 ,columnas);
   tabla = new JTable(modelo);
 tabla.setSelectionMode((int)ListSelectionModel.SINGLE_SELECTION);
   panelScroll = new JScrollPane(tabla);
   panelScroll.createHorizontalScrollBar();
   panelScroll.createVerticalScrollBar();
   panelScroll.setWheelScrollingEnabled(true);

   // COLOCA LOS ENCABEZADOS DE LAS COLUMNAS EN LA TABLA
   for( int c = 0; c < columnas; c++ )
   {
   String sDato = is.readLine();
     tabla.getColumnModel().getColumn(c).setHeaderValue(sDato);
   }

   String info [] = new String[columnas];
   for( int r = 0; r < renglones; r++ )
   {
      for( int c = 0; c < columnas; c++ )
      {
      String sInfo = is.readLine();

         info[c] = sInfo;
      }
      modelo.addRow(info);
   }

   Container c = getContentPane();
      c.add(panelScroll);
      panelScroll.setBounds(10 ,130 ,700 ,200);
      panelScroll.show();

      eAviso.setText("");
      c.add(eAviso);
      eAviso.setBounds(100 ,110 ,300 ,20);
      eAviso.show();

      conexion.close();
    }
    catch(IOException e)
    {
      resultado.insert("FALLO EL SERVIDOR" ,0);
    }
   }
```

```
}



import java.net.*;
import java.io.*;
import java.sql.*;

class ServidorComandoenvia
{
public static void main(String args[])
{
ServerSocket sServ = null;
boolean escuchando = true;
// DETERMINA LOS COMPONENTES QUE SERVIRAN PARA REFERENCIAR LA BD
  String driver1 = "sun.jdbc.odbc.JdbcOdbcDriver";
  String driver = "com.ms.jdbc.odbc.JdbcOdbcDriver";
  //String url = "jdbc:odbc:datos";
  String nombreBD = "e:/Investigador Actual/Beni/basedatos/datos.mdb";
  String ubicacion = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+nombreBD;
  String usuario = null;
  String contrasena = null;

   try
   {
      sServ = new ServerSocket(7777);
      while( escuchando )
      {
      Socket sCliente = null;

         System.out.print("Escucho...");
         try
         {
            sCliente = sServ.accept();
         }
         catch(IOException e)
         {
            continue;
         }
         if( sCliente != null )
         {
         DataInputStream is ;
         PrintStream os ;
         String entra ,sale ;

         try
         {
           is = new DataInputStream(sCliente.getInputStream());
           os = new PrintStream(sCliente.getOutputStream());

           System.out.println("conectado");
           String comando = is.readLine();
           System.out.println("comando:" + comando);
           try
```

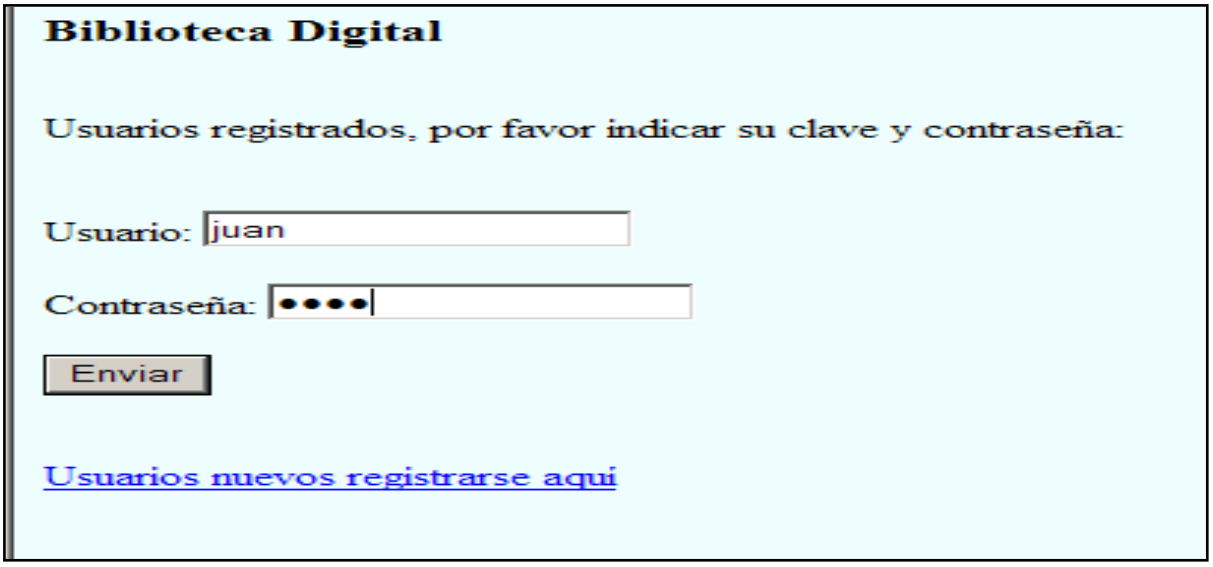
```
            {
              Class.forName(driver);
            }
            catch(ClassNotFoundException e)
            {
               try
               {
                  Class.forName(driver1);
               }
               catch(ClassNotFoundException e1)
               {
                  System.out.println("No se encuentra el driver: " + e.getMessage());
               }
            }

            try
            {
             // EFECTUA LA CONEXION CON LA BASE DE DATOS
             Connection con = DriverManager.getConnection(ubicacion ,usuario ,contrasena);
           //  DatabaseMetaData dbmd = con.getMetaData();

             // CREA UNA CONSULTA TEMPORAL
             Statement stmnt = con.createStatement(  ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

            // EJECUTA EL COMANDO
             ResultSet respuesta = stmnt.executeQuery(comando);
             // ENCABEZADOS DE LA RESPUESTA
             ResultSetMetaData encabezado = respuesta.getMetaData();

             int columnas = encabezado.getColumnCount();
             respuesta.last();
             int renglones = respuesta.getRow();
             respuesta.first();

             os.println(""+columnas); // COLUMNAS
             os.flush();

             os.println(""+renglones); // RENGLONES
             os.flush();
System.out.println("R="+renglones+ " C="+columnas);
             for( int c = 0; c < columnas; c++ )
             {
             String sEnc = encabezado.getColumnName(c + 1);

                os.println(sEnc); // ENCABEZADOS
                os.flush();
             }

            for( int r = 0; r < renglones; r++ )
            {

                for( int c = 0; c < columnas; c++ )
                {
                String dato = respuesta.getString(c + 1);
```

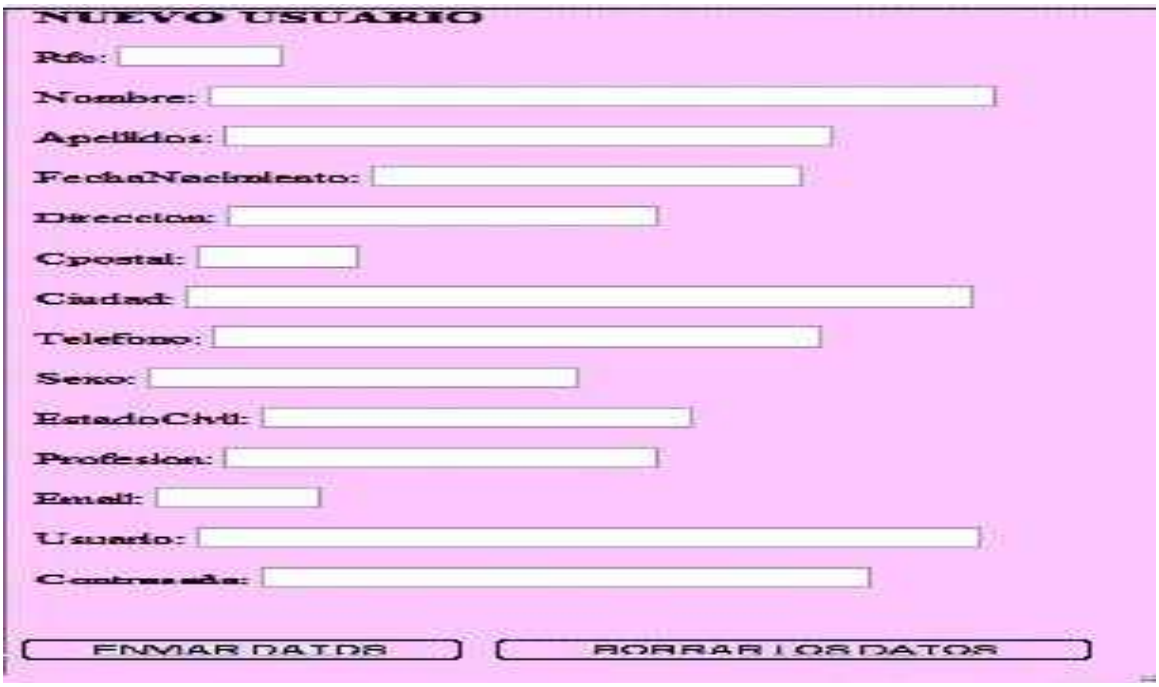
```
                    os.println(dato); // DATOS
                    os.flush();
                    System.out.print(dato+" ");
                  }
                  respuesta.next();
System.out.println();
               }

               //CIERRA LA CONEXION CON LA BASE DE DATOS
               stmnt.close();
               con.close();
              } // try efectua conexion
              catch(SQLException se) // try de conexion
              {
              System.out.println("ERROR 1");
              }
           }
           catch(IOException  e)
           {
            System.out.println("ERROR 2");
            }
        }
        try
        {
           sCliente.close();
        }
        catch(IOException  e)
        {
        System.out.println("ERROR 3");
        }
        }
    }
    catch(IOException  e)
    {  }
  }
}


<html>
<head>
<title>Envia comandos SQL a una Base de Datos</title>
</head>
<body>
<hr>
<applet code=Comandoenvia width=800 height=400>
</applet>
<hr>
</body>
</html>
```

ANEXO B

CASO DOS, MODELO CLIENTE/SERVIDOR CON SERVLETS

```
<html>
 <head>
  <title>Biblioteca Digital</title>
 </head>
 <body bgcolor="#FFFFFF">
  <h3>Biblioteca Digital</h3>
  <BR>
  Usuarios registrados, por favor indicar su clave y contraseña:
 <BR>
     <form action ="http://localhost:8080/examples/servlet/Usuario" method=GET >
         Usuario: <input type=text name=usuario>
     <BR><BR>
     Contraseña: <input type=password name=pwd >
     <BR><BR>
     <input type=submit value="Enviar">
     <BR><BR>
     <form action ="http://localhost:8080/examples/servlet/Alta" method=GET >
     <A href="http://localhost:8080/Alta.html">
      <font size=2>Usuarios nuevos registrarse aquí</font>
    </A>
     <BR><BR>
     </form>
         </form>
 </body>
</html>
```



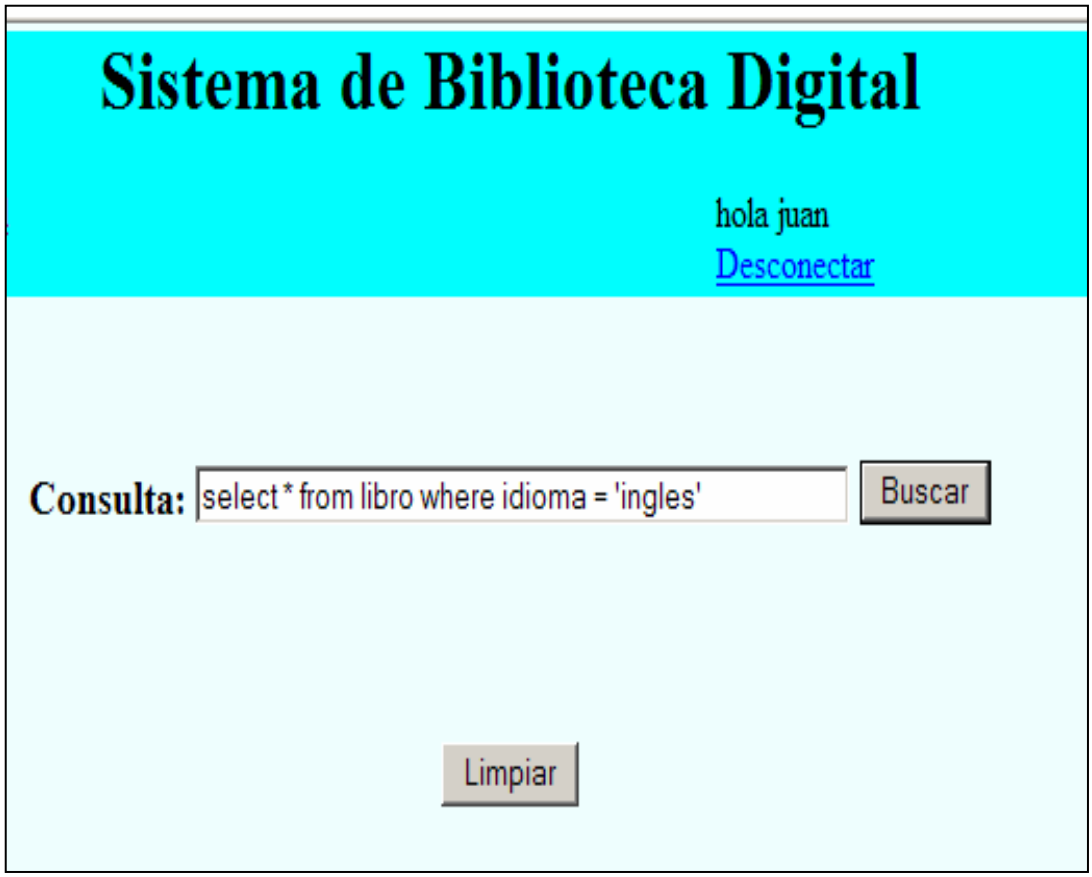



```
<html>
<head>
  <title>NUEVO USUARIO</title>
</head>
<body bgcolor="#FFCCFF">
<h3>NUEVO USUARIO</h3>
<form action="http://localhost:8080/examples/servlet/Alta">
   <p>Rfc: <input type="text" size="5" name="Rfc"> </p>
   <p>Nombre: <input type="text" size="40" name="Nombre"> </p>
   <p>Apellidos: <input type="text" size="30" name="Apellidos"> </p>
   <p>FechaNacimiento: <input type="text" size="20" name="FechaNacimiento"> </p>
   <p>Direccion: <input type="text" size="20" name="Direccion"> </p>
   <p>Cpostal: <input type="text" size="5" name="Cpostal"> </p>
   <p>Ciudad: <input type="text" size="40" name="Ciudad"> </p>
   <p>Telefono: <input type="text" size="30" name="Telefono"> </p>
   <p>Sexo: <input type="text" size="20" name="Sexo"> </p>
   <p>EstadoCivil: <input type="text" size="20" name="EstadoCivil"> </p>
   <p>Profesion: <input type="text" size="20" name="Profesion"> </p>
   <p>Email: <input type="text" size="5" name="Email"> </p>
   <p>Usuario: <input type="text" size="40" name="Usuario"> </p>
   <p>Contraseña: <input type="text" size="30" name="Contraseña"> </p>
   <br><br><br>
   <center><INPUT TYPE=SUBMIT VALUE="ENVIAR
DATOS">             
        
   <INPUT TYPE=RESET VALUE="BORRAR LOS DATOS"></center>
</form>
</body>
</html>
```

Alta

```
<HTML>
   <HEAD>
      <TITLE>NUEVO USUARIO</TITLE>

   </HEAD>

   <BODY>
<body bgcolor="#FFCCFF">

<form>

<form action ="http://localhost:8080/examples/servlet/Alta" method=GET>
Rfc: <INPUT NAME="Rfc"><P>
Nombre: <INPUT NAME="nombre"><P>
Apellidos: <INPUT NAME="apellido"><P>
Fecha-Nacimiento: <INPUT NAME="fecha"><P>

C.Postal: <INPUT NAME="postal"><P>
Ciudad: <INPUT NAME="ciudad"><P>
Telefono: <INPUT NAME="telefono"><P>


Sexo
<input type="radio" name="media" value="media" checked>Hombre
<input type="radio" name="media" value="dk">Mujer

Estado Civil:
<SELECT NAME="OPCION">
  <OPTION>soltero
  <OPTION>Casado
  <OPTION>Divorciado
  <OPTION>Otro
</SELECT><br><br>


Profesión: <INPUT NAME="profesion"><P>
E-mail: <INPUT NAME="email"><P>

<br>
<br>


Usuario: <INPUT TYPE=TEXT NAME=variable>
Contraseña: <INPUT TYPE=PASSWORD NAME=variable VALUE="password">
<br><br>

<center>
<form action ="http://localhost:8080/examples/servlet/Alta" method=GET>
<INPUT TYPE=SUBMIT VALUE="ENVIAR
DATOS">             
        
```

```
<INPUT TYPE=RESET VALUE="BORRAR LOS DATOS">
</form>
</center>
   </BODY>
</HTML>
```




```
<HTML><HEAD>
<TD Height=30>
<center><h1>Sistema de Biblioteca Digital</h1></center>
</TD>
</TR>
<TR>
<TD>
<table>
<TR>
<TD width=100>
<TD width=300>
</TD>
<TD width=200>
hola juan Perez<br>
<a href="http://localhost:8080//examples/servlet/Desconectar" method=GET>Desconectar</a>
</TD>
```

```
</TR>
</table>
</TD>
</TR>
</table>
</TD>
</TR>
<TD>
<br><br><br>
<form>
<center>
<h3>Consulta: <INPUT TYPE="text" NAME="consulta" SIZE=48 MAXLENGTH=48>
<a href="http://localhost:8080//examples/servlet/Buscar" method=GET><INPUT TYPE="submit"
VALUE="BUSCAR"></a>
<br><br><br>
<form>
<textarea name="comment" rows=10 cols=60>
</textarea>
</form>
<br><br>
<INPUT TYPE="submit" VALUE="LIMPIAR">
</form>
</center>
</TD>
</TR>
</table>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
</BODY>
</html>
```

Usuario

```
<html>
  <head>
    <title>Biblioteca Digital</title>
  </head>
  <body bgcolor="#FFFFFF">
    <h3>Biblioteca Digital</h3>
    <BR>
    Usuarios registrados, por favor indicar su clave y contraseña:
   <BR>
        <form action ="http://localhost:8080/examples/servlet/Usuario" method=GET >
            Usuario: <input type=text name=usuario>
        <BR><BR>
        Contraseña: <input type=password name=pwd >
        <BR><BR>
        <input type=submit value="Enviar">
        <BR><BR>
        <form action ="http://localhost:8080/examples/servlet/Alta" method=GET >
        <A href="http://localhost:8080/Alta.html">
```

```
      <font size=2>Usuarios nuevos registrarse aquí</font>
    </A>
     <BR><BR>
     </form>
         </form>
 </body>
</html>


import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import java.text.*;
import java.lang.Math;
import javax.swing.*;


public class Busca2 extends HttpServlet
{
String driver1 = "sun.jdbc.odbc.JdbcOdbcDriver";
String driver = "com.ms.jdbc.odbc.JdbcOdbcDriver";
String nombreBD = "e:/Investigador Actual/Beni/basedatos/datos.mdb";
String ubicacion = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+nombreBD;
String usuario = null;
String contrasena = null;
Connection conexion;
String user;
HttpSession sesion ;
String consulta;
String respuesta;

 // INICIALIZA EL SERVLET
 public void init(ServletConfig config) throws ServletException
 {
    super.init(config); // ASEGURA LA CARGA DEL SERVLET
    try
    {
       Class.forName(driver);
    }
    catch(ClassNotFoundException e)
    {
       try
       {
          Class.forName(driver1);
       }
       catch(ClassNotFoundException e1)
       {
          System.out.println("No hay driver: " + e.getMessage());
       }
    }
 }
```

```
 public void service(HttpServletRequest  peticion, HttpServletResponse respuesta)
   throws ServletException, IOException
  {
     respuesta.setContentType("text/html");
  ServletOutputStream out = respuesta.getOutputStream();

  HttpSession session = peticion.getSession();
  String user = (String)session.getAttribute("usuario");



String contexto = peticion.getContextPath();

      if( user == null )
      {
        out.println("<html>");
        out.println("<BODY bgcolor=\"#FFFFFF\">");
        out.println("<P>Debe darse de alta en el Sistema por favor" + user);
        out.println("</BODY></html>");
        return;
      }

        consulta = peticion.getParameter("consulta");
try
{

long inicio = System.nanoTime();
        // BUSCAR LA CONSULTA EN LA BASE DE DATOS PORQUE NO ESTUVO EN LA CACHE
        Connection con = DriverManager.getConnection(ubicacion ,usuario ,contrasena);
        Statement stmnt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
        ResultSet resp = stmnt.executeQuery(consulta);

        // ENCABEZADOS DE LA RESPUESTA
        ResultSetMetaData encabezado = resp.getMetaData();
        int columnas = encabezado.getColumnCount();
           resp.last();
        int renglones = resp.getRow();
           resp.first();

    out.println("<HTML>");
    out.println("  <HEAD>");
    out.println("    <TITLE>Sistema de Biblioteca Digital</TITLE>");
    out.println("  </HEAD>");
    out.println("  <BODY LINK=#0000ff LINK=#800080 bgcolor=#FFFFFF>");
    out.println("    <TABLE BORDER=1 CELLSPACING=0 CELLPADDING=0>");
    out.println("      <TR>");
    out.println("         <!--");
    out.println("        <TD WIDTH=200 VALIGN=top BGCOLOR=#00ffff>");
    out.println("          <center>");
    out.println("            <h2>S.B.D</h2>");
    out.println("          </center><br><br>");
    out.println("          <center>");
    out.println("            <FONT face=verdana size=2>");
```

```
   out.println("            <a href=\""+ contexto +"/servlet/Respuesta\" method=GET>Consultas
Solicitadas</a><br><br><br><br><br><br><br><br><br>");
   out.println("            <a href=\""+ contexto +"/servlet/Sugerencia\" method=GET>Sugerencias del
Agente</a>");
   out.println("
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>");
   out.println("          </FONT>");
   out.println("        </center>");
   out.println("      </TD>");
   out.println("       -->");
   out.println("      <TD WIDTH=600 VALIGN=top BGCOLOR=#eeFFFF>");
   out.println("      <table>");
   out.println("        <TR>");
   out.println("          <TD>");
   out.println("            <table BGCOLOR=#00ffff CELLSPACING=0 CELLPADDING=0 width=800>");
   out.println("              <TR>");
   out.println("                <TD Height=30>");
   out.println("                  <center>");
   out.println("                    <h1>Sistema de Biblioteca Digital</h1>");
   out.println("                  </center>");
   out.println("                </TD>");
   out.println("              </TR>");
   out.println("              <TR>");
   out.println("                <TD>");
   out.println("                  <table>");
   out.println("                    <TR>");
   out.println("                      <TD width=100>");
   out.println("        <!--");
   out.println("                        <center><a href=\""+ contexto +"/examples/servlet/Perfil\"
method=GET>Perfil del Usuario </a></center>");
   out.println("        -->");
   out.println("                      </TD>");
   out.println("                      <TD width=300>");

   out.println("                      </TD>");
   out.println("                      <TD width=200>");
   out.println("                        <center>hola "+user+"<BR>");
out.println("<!--");
   out.println("            <a href=\""+ contexto
+"/servlet/Usuario?usuario=juan&pwd=juan\">Regresar</a>");
out.println("-->");
   out.println("                        </center>");
   out.println("                      </TD>");
   out.println("                    </TR>");
   out.println("                  </table>");
   out.println("                </TD>");
   out.println("              </TR>");
   out.println("            </table>");
   out.println("          </TD>");
   out.println("        </TR>");
   out.println("        <TR>");
   out.println("          <TD>");
   out.println("<!-- INICIO LA PARTE VARIABLE -->");
```

```
out.println("<font face=verdana size=4>"+consulta+"</font>");

   out.println("<table width=800 valign=top border=1 cellpadding=0 cellspacing=0>");
   out.println("<TR>");

      for( int co = 0; co < columnas; co++ )
      {
      String sEnc = encabezado.getColumnName(co + 1);

         out.println("  <TD WIDTH="+ (int)(800/columnas) + ">");
         out.println("<font face=arial size=2> <B>");
         out.println("  "+sEnc);
         out.println("</B></font>");
         out.println("  </TD>");
      }

      out.println("</TR>");

      for( int r = 0; r < renglones; r++ )
      {
         out.println("<TR>");
         for( int c = 0; c < columnas; c++ )
         {
         String dato = resp.getString(c + 1);

            out.println("  <TD WIDTH="+ (int)(800/columnas) + ">");
         out.println("<font face=arial size=2>");
            out.println("  "+dato);
         out.println("</font>");
            out.println("  </TD>");
         }
         out.println("</TR>");
         resp.next();
      }
   out.println("</table>");
   out.println("</font>");
   out.println("<!-- FIN DE LA PARTE VARIABLE -->");

long termino = System.nanoTime();

long duracion = termino - inicio;
double decimas = ( (long)(((double)duracion / 1000000000D) * 10000) ) / 10000D;

NumberFormat precision6 = NumberFormat.getInstance();

      precision6.setMaximumFractionDigits(6);
      precision6.setMinimumFractionDigits(6);
String dato = precision6.format(decimas);

   out.println("                        <center>"+ duracion + " " +
    ("00:00:" + (decimas < 10 ? "0" : "") + dato) + "</center>");

   out.println("          </TD>");
   out.println("        </TR>");
   out.println("      </table>");
```

```
   out.println("        </TD>");
   out.println("      </TR>");
   out.println("    </TABLE>");
   out.println("</BODY>");
   out.println("</HTML>");
  }
catch(SQLException exc)
{

   out.println("<HTML>");
   out.println("<HEAD> </HEAD>");
   out.println("<BODY>Falla en la Base de Datos"+exc.toString() );
   out.println("</BODY>");
   out.println("</HTML>");

}
}
}
```

ANEXO C

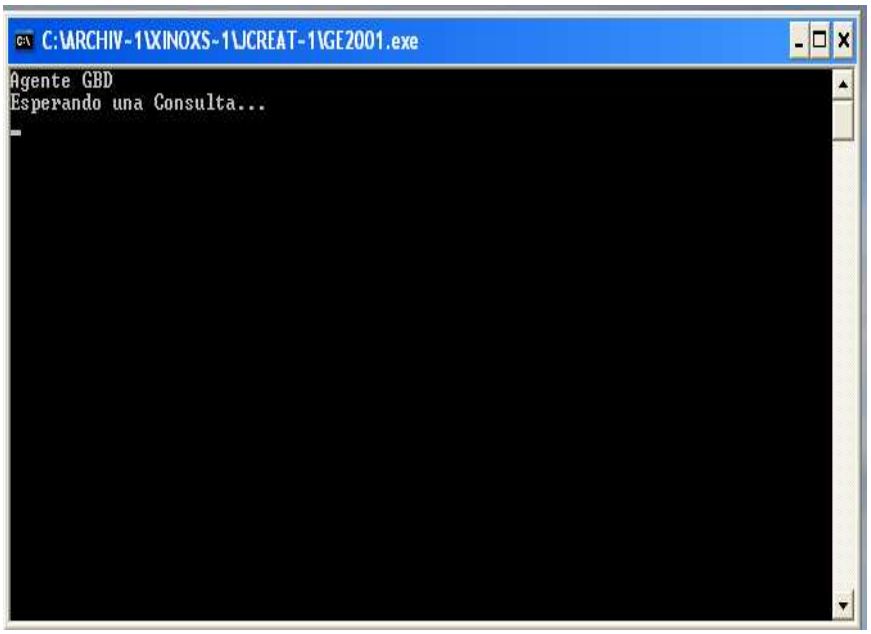
CASO TRES, SISTEMA DE CONTROL DE CONCURRENCIA A UNA BASE DE DATOS MEDIANTE AGENTES

Cache

```
import java.net.*;
import java.io.*;
import java.util.*;

class Cache
{
	String consulta;
	Vector encabezado;
	int    columnas;
	int    renglones;
	Vector info;

	Cache(String c, int cols, int rengs)
	{
		consulta=c;
		columnas=cols;
		renglones=rengs;
		encabezado=new Vector();
		info=new Vector();
	}

	void addE(String e)
	{
		encabezado.add(e);
	}

	void addInfo(String dato)
	{
		info.add(dato);
	}

	String getInfo(int r, int c)
	{
		return (String) info.get(r * columnas + c);
	}

	String getConsulta()
	{
```

```
                return consulta;
        }

        boolean generaArchivo(long _secuencia)
        {
                try
                {
        BufferedWriter salida = new BufferedWriter(new FileWriter("c:\\"+ _secuencia +".doc",false));


      System.out.println("GENERANDO EL ARCHIVO SECUENCIA"+columnas+" r="+renglones+"
"+consulta);
        salida.write(consulta+"\n");
        salida.write(""+columnas+"\n");
         System.out.println("cols");
        for( int i = 0; i < columnas; i++ )
        {
        String enc = (String)encabezado.get(i);
            salida.write(enc+"\n");
        }

        salida.write(""+renglones+"\n");

        System.out.println("encs");
        for( int c = 0; c < columnas; c++ )
           for( int r = 0; r < columnas; r++ )
           {
           String dato = this.getInfo(r, c);
            System.out.println("dato "+c +" r "+dato);

            if(dato==null )
                salida.write("\n");
            else
                salida.write(dato+"\n");
           }
         salida.close();
                        return true;
                }
                catch(Exception e)
                {
                        System.out.println("err");
                }
                return false;
        }
}
```

Agente Gestor de la Base de Datos




```
import java.net.*;
import java.io.*;
import java.util.*;
import java.sql.*;

class AgenteGBD extends Thread
{
static Vector cache = new Vector();
String consulta ;
int destino ;
long secuencia ;

  AgenteGBD (Socket conexion)
  {
      consulta = null;
      destino = 0;
      secuencia = 0;
      try
      {
      DataInputStream is = new DataInputStream(conexion.getInputStream());

         consulta = is.readLine();
         System.out.println("llego"+consulta);
         String destinoS = is.readLine();
         String secuenciaS = is.readLine();
         destino = Integer.parseInt(destinoS);
         secuencia = Long.parseLong(secuenciaS);
      }
      catch(IOException e)
      {}
   }

  public void run()
  {
          // ACCESA LA CACHE PARA BUSCAR LA CONSULTA
      boolean esta = this.busca(consulta ,secuencia);

      if( esta )
                        return;

                    // ACCESA LA BASE DE DATOS PARA OBTENER LA RESPUESTA DE LA
CONSULTA
    String driver1 = "sun.jdbc.odbc.JdbcOdbcDriver";
  String driver = "com.ms.jdbc.odbc.JdbcOdbcDriver";
  String nombreBD = "e:/Investigador Actual/Beni/basedatos/datos.mdb";
  String ubicacion = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ="+nombreBD;

  String usuario = null;
  String contrasena = null;
```

```
try
 {
   Class.forName(driver);
 }
 catch(ClassNotFoundException e)
 {
    try
    {
       Class.forName(driver1);
    }
    catch(ClassNotFoundException e1)
    {
       System.out.println("No se encuentra el driver: " + e.getMessage());

    }
 }

 try
 {
  System.out.println("voy a conectarme");
  // EFECTUA LA CONEXION CON LA BASE DE DATOS
  Connection con = DriverManager.getConnection(ubicacion ,usuario ,contrasena);
//  DatabaseMetaData dbmd = con.getMetaData();

  System.out.println("ahy ya me conect");

  // CREA UNA CONSULTA TEMPORAL
  Statement stmnt = con.createStatement(  ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);
  System.out.println("YA se creo el stmnt"+consulta);

 // EJECUTA EL COMANDO
  ResultSet respuesta = stmnt.executeQuery(consulta);
  System.out.println("ya ejecute el comando");
  // ENCABEZADOS DE LA RESPUESTA
  ResultSetMetaData encabezado = respuesta.getMetaData();

  int columnas = encabezado.getColumnCount();
  respuesta.last();
  int renglones = respuesta.getRow();
  respuesta.first();

                        // SE CREA EL OBJETO QUE VA EN LA CACHE
                                Cache elemento = new Cache(consulta, columnas, renglones);
                                System.out.println("c r"+columnas+ " "+renglones);
                                // ADICIONA LOS ENCABEZADOS EN EL OBJETO CACHE
  for( int c = 0; c < columnas; c++ )
  {
  String sEnc = encabezado.getColumnName(c + 1);
                          System.out.println("encabezado"+sEnc);

                                                elemento.addE(sEnc);
```

```
            }

           for( int r = 0; r < renglones; r++ )
           {
               for( int c = 0; c < columnas; c++ )
               {
               String dato = respuesta.getString(c + 1);


                                                    elemento.addInfo(dato);
               }
               respuesta.next();
            }

            //CIERRA LA CONEXION CON LA BASE DE DATOS
            stmnt.close();
            con.close();
              System.out.println("sec generation"+secuencia);
            elemento.generaArchivo(secuencia);

            // DA DE ALTA EL ELEMENTO DE LA CACHE EN LA CACHE
            cache.add(elemento);

           } // try efectua conexion
           catch(SQLException se) // try de conexion
           {
           System.out.println("ERROR 1");
           }

   }

  synchronized static boolean busca(String _consulta ,long _secuencia)
  {
  int i ;

     for( i = 0; i < cache.size(); i++ )
     {
     Cache elemento = (Cache)cache.get(i);

        if( _consulta.equals(elemento.getConsulta()) == true )
        {
             // CON EL CONTENIDO DE elemento GENERAR EL ARCHIVO secuencia.DAT
           return elemento.generaArchivo(_secuencia);
        }
     }
            return false;
  }
}

class ServidorCache
{
  public static void main(String args[])
  {
  ServerSocket servicio = null;
```

```
 boolean escuchando = true;

            System.out.println("Agente GBD");
   try
   {
            servicio = new ServerSocket(7777);
       while( escuchando )
       {
       Socket conexion = null;

            System.out.println("Esperando una Consulta...");
          try
          {
            conexion = servicio.accept();
              new AgenteGBD(conexion).start();
          }
          catch(IOException e)
          {
            continue;
          }
       }
   }
   catch(IOException  excepcion)
   {
            System.out.println("ERROR EN EL AGENTE GBD");
   }
 }
}
```

Agente InvocadorAGBD

```
import java.net.*;
import java.io.*;

class InvocadorAGBD extends Thread
{

 String consulta;
 int destino;
 long secuencia;

   InvocadorAGBD (String c, int d, long s)
   {
    consulta = c;
            destino  = d;
            secuencia = s;
   }

 public  void run()
 {

 Socket conexion ;
```

```
 DataInputStream is ;
 PrintStream os ;

  try
  {
     conexion = new Socket("localhost" ,7777);
     is = new DataInputStream(conexion.getInputStream());
     os = new PrintStream(conexion.getOutputStream());
     System.out.println("servidor contactado");

     os.println(consulta);
     os.flush();
     os.println(""+ destino);
     os.flush();
     os.println(""+ secuencia);
     os.flush();
     conexion.close();

  }
  catch(IOException e)
  { }
 }


}
```

Acceso del Usuario

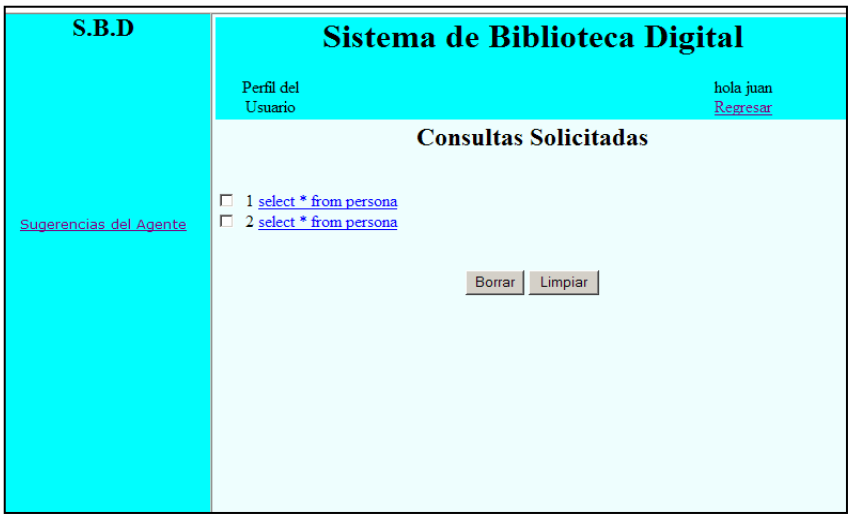
```
<html>
  <head>
    <title>Biblioteca Digital</title>
  </head>
  <body bgcolor="#FFFFFF">
    <h3>Biblioteca Digital</h3>
    <BR>
    Usuarios registrados, por favor indicar su clave y contraseña:
   <BR>
       <form action ="http://localhost:8080/examples/servlet/Usuario" method=GET >
            Usuario: <input type=text name=usuario>
       <BR><BR>
       Contraseña: <input type=password name=pwd >
       <BR><BR>
       <input type=submit value="Enviar">
       <BR><BR>
       <form action ="http://localhost:8080/examples/servlet/Alta" method=GET >
       <A href="http://localhost:8080/Alta.html">
        <font size=2>Usuarios nuevos registrarse aquí</font>
     </A>
       <BR><BR>
       </form>
            </form>
  </body>
</html>
```

Pagina Principal del Sistema


```
<HTML><HEAD>
<TITLE>Biblioteca Digital</TITLE></HEAD>
<BODY LINK="#0000ff" LINK="#800080" bgcolor="#FFFFFF">
<TABLE BORDER=1 CELLSPACING=0 CELLPADDING= >
<TR>
 <TD WIDTH="200" VALIGN="150" BGCOLOR="#00ff00">
<center><h2>S.B.D</center><br><br>
<center>
<FONT face="verdana" size="2">
<a href="/examples/servlet/Respuesta" method=GET>Consultas
Solicitadas</a><br><br><br><br><br><br><br><br><br>
<a href="/examples/servlet/Sugerencia" method=GET>Sugerencias del Agente</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
</FONT>
</center>
```

```
</TD>
<TD WIDTH="600" VALIGN="150" BGCOLOR="#FFFFCC">
<table>
<TR>
<TD>
<table>
<TR>
<TD Height=30>
<center><h1>Sistema de Biblioteca Digital</h1></center>
</TD>
</TR>
<TR>
<TD>
<table>
<TR>
<TD width=100>
<a href="/examples/servlet/Perfil" method=GET>Perfil del Usuario </a>
</TD>
<TD width=300>
</TD>
<TD width=200>
hola juan<br>
<a href="/examples/servlet/Desconectar" method=GET>Desconectar</a>
</TD>
</TR>
</table>
</TD>
</TR>
</table>
</TD>
</TR>
<TD>
<br><br><br>
<form action="/examples/servlet/Buscar" method=GET>
<center>
<h3>Consulta: <INPUT TYPE="text" NAME="consulta" SIZE=48 MAXLENGTH=48>
<input TYPE="submit" value="Buscar">
<br><br><br>
<br><br>
<INPUT TYPE="reset" VALUE="Limpiar">
</center>
</form>
</TD>
</TR>
</table>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
</BODY>
</html>
```

Pantalla de Consultas Solicitadas




```
<HTML><HEAD>
<TITLE>Sistema de Biblioteca Digital</TITLE></HEAD>
<BODY LINK="#0000ff" LINK="#800080" bgcolor="#FFFFFF">
<TABLE BORDER=1 CELLSPACING=0 CELLPADDING= >
<TR>
 <TD WIDTH="200" VALIGN="150" BGCOLOR="#00ff00">
<center><h2>S.B.D</center><br><br>
<center>
<FONT face="verdana" size="2">
<a href="http://localhost:8080/examples/servlet/Respuesta" method=GET>Consultas
Solicitadas</a><br><br><br><br><br><br><br><br><br>
<a href="http://localhost:8080/examples/servlet/Sugerencia" method=GET>Sugerencias del Agente</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
</FONT>
</center>
</TD>
<TD WIDTH="600" VALIGN="150" BGCOLOR="#FFFFCC">
<table>
<TR>
<TD>
<table>
<TR>
<TD Height=30>
<center><h1>Sistema de Biblioteca Digital</h1></center>
</TD>
</TR>
<TR>
<TD>
<table>
<TR>
<TD width=100>
```

```
<a href="http://localhost:8080//examples/servlet/Perfil" method=GET>Perfil del Usuario </a>
</TD>
<TD width=300>
</TD>
<TD width=200>
hola juan Perez<br>
<a href="http://localhost:8080//examples/servlet/Desconectar" method=GET>Desconectar</a>
</TD>
</TR>
</table>
</TD>
</TR>
</table>
</TD>
</TR>
<TD>
<br><br><br>
<form>
<center>
<h2>Respuesta de Consultas</h2>
<h3>Consulta: <INPUT TYPE="text" NAME="consulta" SIZE=48 MAXLENGTH=48>
<a href="http://localhost:8080//examples/servlet/Buscar" method=GET><INPUT TYPE="submit"
VALUE="BUSCAR"></a>
<br><br><br>

<FORM ACTION=\"http://localhost:8080/examples/servlet/Guarda\" METHOD=POST>

  <INPUT TYPE=TEXT NAME=f SIZE=3 VALUE=1>
  <INPUT TYPE=Checkbox NAME=c>
  <INPUT TYPE=submit VALUE="BORRAR">
</FORM>

</form>
<br><br>
<INPUT TYPE="submit" VALUE="REGRESAR">
</form>
</center>
</TD>
</TR>
</table>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
</BODY>
</html>
```

Sugerencias del Agente


```
<HTML><HEAD>
<TITLE>Sistema de Biblioteca Digital</TITLE></HEAD>
<BODY LINK="#0000ff" LINK="#800080" bgcolor="#FFFFFF">
<TABLE BORDER=1 CELLSPACING=0 CELLPADDING= >
<TR>
 <TD WIDTH="200" VALIGN="150" BGCOLOR="#00ff00">
<center><h2>S.B.D</center><br><br>
<center>
<FONT face="verdana" size="2">
<a href="http://localhost:8080/examples/servlet/Respuesta" method=GET>Consultas
Solicitadas</a><br><br><br><br><br><br><br><br><br>
<a href="http://localhost:8080/examples/servlet/Sugerencia" method=GET>Sugerencias del Agente</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
</FONT>
</center>
</TD>
<TD WIDTH="600" VALIGN="150" BGCOLOR="#FFFFCC">
<table>
<TR>
<TD>
<table>
<TR>
<TD Height=30>
<center><h1>Sistema de Biblioteca Digital</h1></center>
</TD>
</TR>
<TR>
<TD>
<table>
<TR>
<TD width=100>
<a href="http://localhost:8080//examples/servlet/Perfil" method=GET>Perfil del Usuario </a>
</TD>
```

```
<TD width=300>
</TD>
<TD width=200>
hola juan Perez<br>
<a href="http://localhost:8080//examples/servlet/Desconectar" method=GET>Desconectar</a>
</TD>
</TR>
</table>
</TD>
</TR>
</table>
</TD>
</TR>
<TD>
<br><br><br>
<form>
<center>
<h2>Sugerencia del Agente</h2>
<h3>Consulta: <INPUT TYPE="text" NAME="consulta" SIZE=48 MAXLENGTH=48>
<a href="http://localhost:8080//examples/servlet/Buscar" method=GET><INPUT TYPE="submit"
VALUE="BUSCAR"></a>
<br><br><br>

<FORM ACTION=\"http://localhost:8080/examples/servlet/Guarda\" METHOD=POST>

  <INPUT TYPE=TEXT NAME=f SIZE=3 VALUE=1>
  <INPUT TYPE=Checkbox NAME=c>
  <INPUT TYPE=submit VALUE="BORRAR">
</FORM>

</form>
<br><br>
<INPUT TYPE="submit" VALUE="REGRESAR">
</form>
</center>
</TD>
</TR>
</table>
</TD>
</TR>
</TABLE>
</BODY>
</HTML>
</BODY>
</html>
```

ANEXO D

SISTEMA UNICORN, (SISTEMA INTERBIBLIOTECARIO) DE LA BIBLIOTECA NACIONAL DE CIENCIA Y TECNOLOGÍA DEL INSTITUTO POLITÉCNICO NACIONAL

Este sistema fue adquirido por el Instituto Politécnico Nacional, para la Biblioteca Nacional de Ciencia y Tecnología, a principios de su creación de está última.

El sistema Unicorn, tiene la funcionalidad de almacenar el acervo bibliográfico, tesis, entre otros materiales, de todas las bibliotecas que forman parte del Instituto Politécnico Nacional.

El sistema Unicorn, además de almacenar material bibliográfico, cuenta con otros módulos tales como préstamo, adquisiciones, reservaciones, por mencionar algunos; esto, con la finalidad de darle un mejor funcionamiento a las bibliotecas.

A esté sistema Unicorn, a diario se conectan muchos usuarios, como se muestra en la figura 1, para operar con el sistema, para la realización de diversas actividades tales como, realizar préstamos de material bibliográfico, procesar material bibliográfico, dar de alta nuevos usuarios, entre otras actividades.

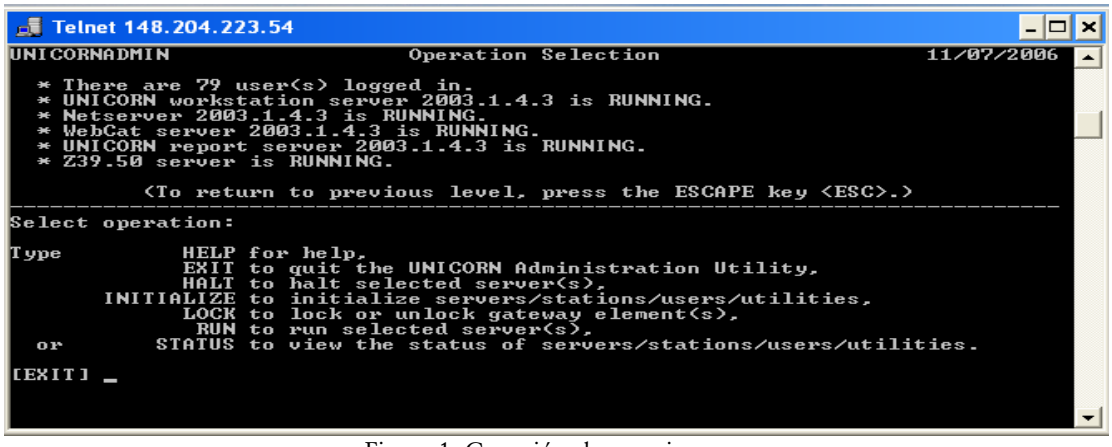


Figura 1. Conexión de usuarios.

A continuación se muestra una grafica de las estadísticas de acceso de los usuarios, esta grafica se puede ver en la figura 2.

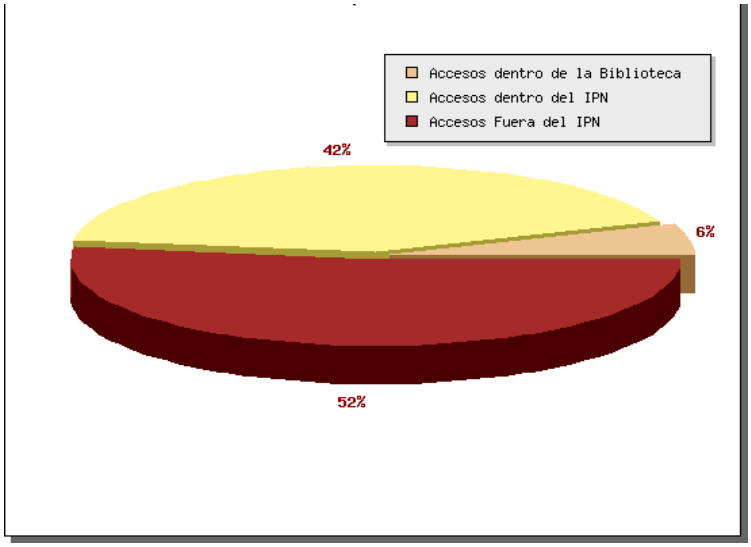


Figura 2

Este sistema ejecuta muchos procesos, al mismo tiempo, sólo que esto trae ciertos problemas, tales como, si un usuario quiere conectarse, en algunas ocasiones no le es posible conectarse de inmediato sino mas bien tiene que realizar varias intentos para poder lograr la conexión, y esto, para los usuarios es molesto ya que se pierde mucho tiempo en estar esperando un acceso al sistema.

Otro problema que se presenta es que al estar conectados muchos usuarios al sistema el tiempo de respuesta no es favorable y provoca que los usuarios se desesperen.

Unicorn permite que los usuario puedan realizar consultas via Internet para renovar algún material que se les fue prestado por parte de la Biblioteca, solo que en lagunas ocasiones presenta algunos problemas tales como la que se muestra en las siguientes figuras3, ya que no le permite al usuario realizar la tarea que requiere y emite mensajes raros, como “El webcat no esta disponible, reintente más tarde” y en la figura 4, emite respuesta “Error interno”.

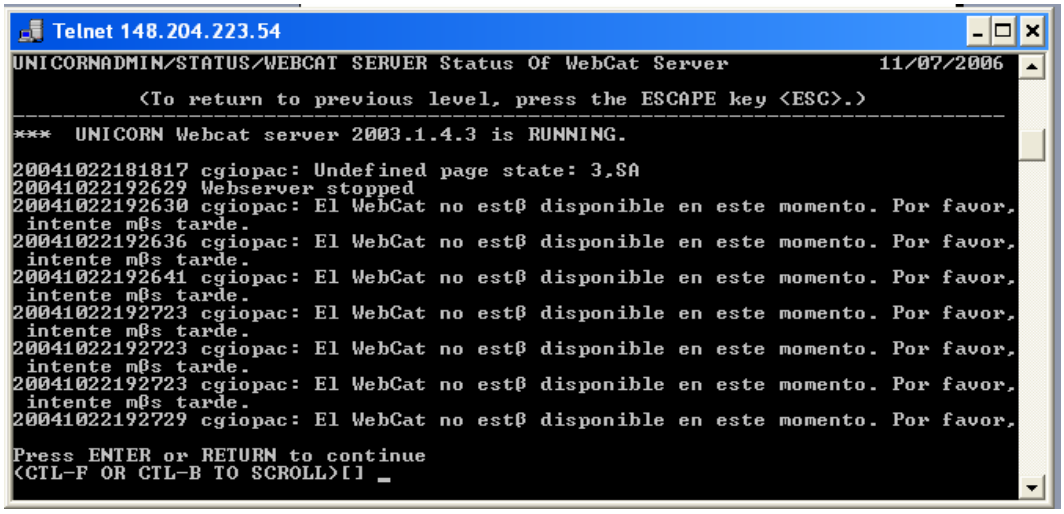


Figura 3.

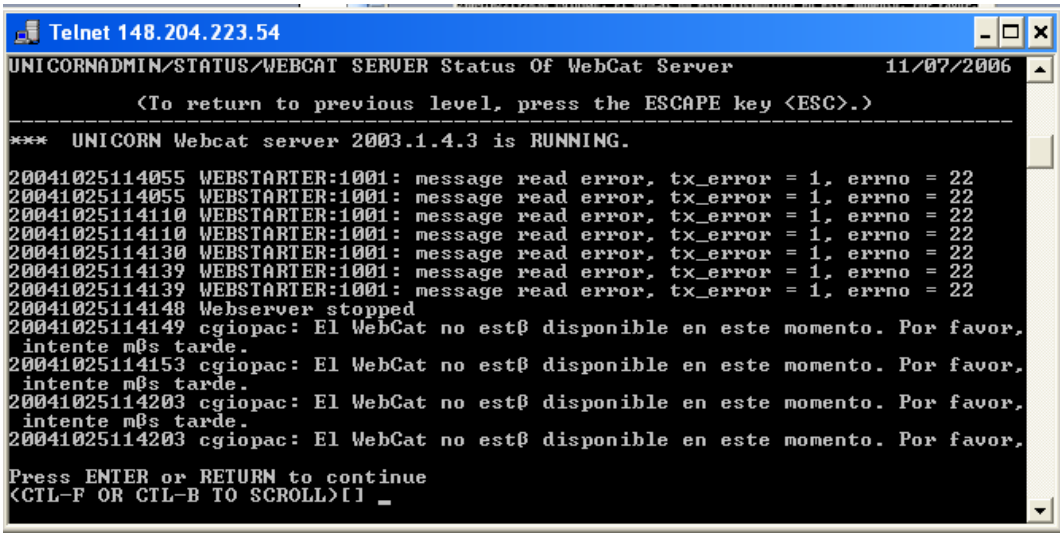


Figura 4.

Como se puede observar con las imágenes mostradas, se refleja que el Sistema Unicorn presenta algunas dificultades, pero la que más se reflejan, es cuando un usuario desea conectarse para realizar diversas consultas y el tiempo de respuesta no es la esperada.