SEGURIDAD EN BASES DE DATOS DISTRIBUIDAS UTILIZANDO AGENTES MÓVILES

DAVID ANTONIO FRANCO BORRE

YASMIN MOYA VILLA

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
MAESTRIA EN CIENCIAS COMPUTACIONALES
CAMPUS UNITECNOLOGICA
CARTAGENA DE INDIAS
2005

SEGURIDAD EN BASES DE DATOS DISTRIBUIDAS UTILIZANDO AGENTES MÓVILES

DAVID ANTONIO FRANCO BORRE

YASMIN MOYA VILLA

PRIMER AVANCE DE LA TESIS PARA OPTAR AL TITULO DE MAGISTER EN CIENCIAS COMPUTACIONALES

Director:
MSC. MOISES RAMÓN QUINTANA ÁLVAREZ

Asesor:
MSC. WILSON PELAEZ HERNANDEZ

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
MAESTRIA EN CIENCIAS COMPUTACIONALES
CAMPUS UNITECNOLOGICA
CARTAGENA DE INDIAS
2005

CONTENIDO

pág.

INTRODUCCION

1. PROPUESTA DE INVESTIGACION 9
1.1 ANTECEDENTES 9
1.2 MARCO TEORICO 10
1.2.1 Seguridad de los sistemas de información 10
1.2.1.1 Políticas de seguridad 11
1.2.1.2 Control de acceso 12
1.2.1.3 Criptografía 12
1.2.2 Introducción a bases de datos distribuidas 15
1.2.3 Seguridad en bases de datos distribuidas 17
1.2.4 Conceptos, características y usos de agentes móviles 19
1.2.5 Seguridad en agentes móviles 21
1.2.6 Acceso a bases de datos distribuidas mediante agentes móviles. 22
1.3 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN 25
1.4 OBJETIVOS DE LA INVESTIGACIÓN 27
1.4.1 OBJETIVO GENERAL 27
1.4.2 OBJETIVOS ESPECÍFICOS 27
1.5 RESULTADOS ESPERADOS 28
1.6 ACTIVIDADES 28

2 TECNOLOGIA DE AGENTES MOVILES 29
2.1 CONCEPTOS BASICOS 29
2.1.1 Agente 29
2.1.2 Agente móvil 30
2.2 HISTORIA Y EVOLUCION DE LOS AGENTES MOVILES 32
2.3 SISTEMAS PARA EL DESARROLLO DE AGENTES MOVILES 35
2.3.1 Aglets 35
2.3.2 Concordia 37
2.3.3 JATLite 38
2.3.4 Odyssey 40
2.3.5 JADE 40
2.3.5 Voyager 42
2.3.6 ARA 42
2.3.7 Tacoma 43
2.4 ELECCION DE LA PLATAFORMA: AGLETS 43

3 BASES DE DATOS DISTRIBUIDA 44
3.1 Procesamiento Distribuido Vs. SGBDD 46
3.2 Ventajas e Inconvenientes de SGBDD 48
3.3 Diseño de Bases de Datos Distribuidas 50
3.4 Diseño de la Base de Datos de Prueba 54
3.4.1 Modelo Entidad Relación 55
3.4.2 Modelo Relacional 56
3.4.3 Fragmentación 56
3.4.4 Réplicas 56
3.5 Sistema Gestor de Base Datos Seleccionado: MSSQL Server 2000 57
3.5.1 Seguridad de SQL Server 57
3.5.1.1 Autentificación del login 58
3.5.1.2 Autentificación de SQL Server 58
3.5.1.3 Autentificación de Windows 2000 Server 58

3.5.1.4 Modo de Autentificación 59
3.5.1.5 Cuentas de Usuario y Roles de una Base de Datos 59
3.5.1.6 Validación de Permisos 60
3.5.2 Servicios de SQL Server 61
3.5.2.1 Servicio MSSQL Server 61
3.5.2.2 Servicio SQL Server Agent 62
3.5.2.3 Servicio Microsoft Distribuye Transaction Coordinate 63
3.5.3 Arquitectura de Duplicación o Replica 63
3.5.3.1 Dimensión de la Duplicación 64
3.5.3.2 Tipos de Duplicación 65
3.5.3.3 Componentes de la Duplicación 66
3.5.3.4 Agentes de Duplicación 68
3.5.3.5 Administrar de la Seguridad de la duplicación 70
3.5.3.6 Configurar de la Duplicación 73
3.5.3.7 Publicar de Datos y Procedimientos Almacenados 73
3.5.3.8 Suscribirse a Publicaciones 74
3.5.3.9 Supervisar la Duplicación 75
3.6 Puesta en Marcha de la Base de Datos 76
3.6.1 Ventajas del Modelo y Arquitectura de Base de Datos Planteado 79

4. AGENTES MOVILES: SEGURIDAD 80
4.1 Amenazas de Seguridad Inherentes al Sistema de Agentes Móviles 81
4.1.1 El Problema del Agente Malicioso 81
4.1.2 El Problema del Servidor Malicioso 83
4.2 Problemas del acceso a bases de datos distribuidas con agentes móviles 85
4.2.1 Autenticación 86
4.2.2 Autorización 86
4.2.3 Confidencialidad 87
4.2.4 Integridad 87

5. MODELO PROPUESTO PARA EL ACCESO A BASES A DATOS DISTRIBUIDAS UTILIZANDO AGENTES MOVILES 88
5.1 Modelo Propuesto 88
5.1.1 Sistema de agentes móviles 89
5.1.2 Modelo de Seguridad 91
5.1.3 Base de Datos 94
5.2 Utilizando ABADAM 95
5.2.1 ABADAM Cliente 96
5.2.2 ABADAM Servidor 100
CONCLUSIONES 105

BIBLIOGRAFIA 107

ANEXOS

LISTA DE FIGURAS

pág.

Figura 1. Propiedades de los Agentes 31

Figura 2. Contexto de los Aglets, Interfaz de Usuario y Monitor de Red 36

Figura 3. Sistema de Base de Datos Distribuida 47

Figura 4. Procesamiento Distribuido 47

Figura 5. Diagrama Entidad – Relación 55

Figura 6. Modelo de la Base de Datos de Prueba 77

Figura 7. Problema del Agente Malicioso 82

Figura 8. Problema del Servidor Malicioso 84

Figura 9. Modelo Propuesto ABADAM 89

Figura 10. Modelo de Seguridad de ABADAM 92

Figura 11. Pantalla inicial de ABADAM Cliente 93

Figura 12. Opción Conectar de la interfaz ABADAM Cliente 94

Figura 13. Captura de dirección IP del servidor, login y password 95

Figura 14. Agente conectado y esperando a que se digite alguna instrucción 96

Figura 15. Ejecución de una instrucción SELECT 97

Figura 16. Pantalla inicial de ABADAM Server 98

Figura 17. Mensaje de error al no encontrar el ODBC 99

Figura 18. Tahiti – Entorno de ejecución de los agentes 100

Figura 19. Tahiti – Log Information 100

LISTA DE ANEXOS

pág.

ANEXO A. Instalando ABADAM

ANEXO B. Instalando SQL Server

ANEXO C. Instalando la plataforma de Aglets 2

ANEXO D. Contenido del CD

INTRODUCCION

Internet ha sido el gran responsable del éxito y acogida de los agentes móviles logrando que esta tecnología haya evolucionado rápidamente y que cada día pueda brindar soluciones no sólo a nivel de Internet, sino en ambientes de redes de computadoras y sistemas distribuidos. Por tanto la tecnología de agentes móviles no sólo se utiliza para implementar sistemas basados en ellos, sino que sirve de apoyo para integrarse a otras tecnologías o aplicativos, como es el caso de sistemas distribuidos, para aprovechar las bondades que brindan los agentes móviles.

Este trabajo representa la implementación de un modelo para el acceso a bases de datos distribuidas utilizando agentes móviles (en adelante ABADAM). El documento consta de cinco capítulos. El primer capitulo del trabajo consta de la propuesta de investigación. En el segundo capitulo se presenta un estudio de las diferentes tecnologías para el desarrollo de agentes móviles, iniciando desde conceptos básicos, historia y evolución hasta presentar diferentes herramientas que existen en el mercado para la creación de los agentes móviles, este estudio fue de gran valor para decidir la herramienta que se utilizó en el desarrollo de este Proyecto.

En el tercer capitulo se presenta una introducción sobre bases de datos distribuidas, su origen y diferencias con otros tipos de sistemas de bases de datos, y el diseño de la base de datos distribuida de prueba que se utilizó, así como el sistema gestor de base de datos seleccionado en el trabajo de investigación. El cuarto capitulo describe los problemas de seguridad que se pueden tener al implementar sistemas de agentes móviles. Por último, en el quinto capitulo se describe el modelo propuesto para el acceso a base de

datos distribuidas utilizando agentes móviles (ABADAM), en donde se tiene en cuenta la tecnología escogida en el capitulo 2 para la implementación de los agentes móviles y el sistema gestor de base de datos descrito en el capitulo 3 para el diseño e implementación de la base de datos distribuida de prueba.

1. PROPUESTA DE INVESTIGACION

1.1 ANTECEDENTES

En los últimos años se ha hablado de agentes móviles como una nueva herramienta de programación en ambientes distribuidos e Internet. El crecimiento acelerado de las redes ha conducido a un mayor desarrollo de sistemas de información distribuidos donde se puede acceder a bases de datos distribuidas.

La necesidad de tener los programas cerca de los datos para aumentar el rendimiento de las aplicaciones dio origen a tecnologías como CORBA y DCOM. Con la aparición de los agentes móviles y lenguajes multiplataformas como Java, los ambientes distribuidos han demostrado mejorar el desempeño de las aplicaciones permitiendo una interacción local en el procesamiento de la información reemplazando tener una conexión constante en la red.

A pesar del desempeño que se ha logrado con el uso de agentes móviles para acceder a bases de datos distribuidas, el crecimiento de esta tecnología se ha visto retardada por problemas de seguridad.

La originalidad de la tesis consiste en diseñar e implementar un sistema de agentes móviles que colaboren en el proceso de autenticación para acceder a una base de datos distribuida, esta tesis pretende ser un aporte más de la aplicación de los agentes móviles en sistemas distribuidos.

1.2 MARCO TEÓRICO

1.2.1 Seguridad de los sistemas de información

La seguridad es un concepto asociado a la certeza, falta de riesgo o contingencia, conviene aclarar que esa certeza no es absoluta, los riesgos siempre están presente, independiente de las medidas que se tomen, por lo que se debe hablar de niveles de seguridad. La seguridad absoluta es imposible y la seguridad informática es un conjunto de políticas encaminadas a obtener altos niveles de seguridad en los sistemas informáticos. La seguridad en los sistemas de información abarca tres propiedades fundamentales: Confidencialidad, Integridad y Disponibilidad.

La confidencialidad se refiere a que la información solo puede ser conocida por personas o usuarios autorizados. Hay una gran cantidad de ataques contra la privacidad, especialmente en los medios de comunicación de los datos. La transmisión a través de un medio presenta muchas oportunidades para ser interceptada y copiada: conexión fraudulenta a la red, recepción electromagnética no autorizada o una simple intrusión en las computadoras donde está almacenada la información.

La integridad se refiere a la seguridad de que una información no ha sido modificada, eliminada, copiada, alterada, entre otros, durante el proceso de transmisión o en su propio equipo de origen. Un ataque común es encontrar intrusos que al no poder descifrar un paquete de información, lo que hacen es eliminarlo.

La disponibilidad de la información se refiere a la seguridad de los usuarios al solicitar una información pueda ser recuperada en el momento que se necesite, es decir, para evitar una pérdida o bloqueo, sea por ataque doloso, mala operación accidental o situaciones fortuitas o de fuerza mayor.

1.2.1.1 Políticas de Seguridad

Una política de seguridad informática es una manera de comunicarse con el personal de la organización, en relación con los recursos y servicios informáticos de la organización. No se puede considerar que una política de seguridad informática es una descripción técnica de mecanismos, ni una expresión legal que involucre sanciones a conductas de los empleados, es más bien una descripción de lo que se desea proteger y él por qué de ello, pues cada política de seguridad es una invitación a cada uno de sus miembros a reconocer la información como uno de sus principales activos así como, un motor de intercambio y desarrollo en el ámbito de sus negocios. Por tal razón, las políticas de seguridad deben concluir en una posición consciente y vigilante del personal por el uso y limitaciones de los recursos y servicios informáticos.

1.2.1.2 Control de acceso

Históricamente las organizaciones y el hombre han tenido la necesidad de controlar el acceso a ciertas áreas y lugares. Esta necesidad es motivada inicialmente por temor a que personas inescrupulosas o delincuentes puedan robar y/o extraer material valioso de acuerdo a criterios personales, sociales,

comerciales, entre otros. Tecnológicamente han cambiado ciertas cosas, pero en el fondo persisten las razones y motivos para mantener mecanismos de control de acceso sobre áreas e información que se desea proteger. Los mecanismos de validación han sufrido modificaciones: hoy en día tenemos controles biométricos, magnéticos, visuales, etc

El control de acceso es el mecanismo por el que se asegura o se intenta asegurar que los recursos son sólo accedidos por los sujetos autorizados, y que solamente pueden realizar las operaciones autorizadas.

1.2.1.3 Criptografía

La utilización de la criptografía tiene como finalidad prevenir algunas fallas de seguridad en un sistema computacional. La seguridad informática debe de ser considerada como un aspecto de gran importancia en cualquier organización que trabaje con sistemas computacionales. El hecho que gran parte de las actividades humanas estén cada vez más relacionadas con los sistemas computacionales hace de la seguridad un factor muy importante en el diseño de los sistemas de información.

La criptografía actual se inicia a mediados de la década de los 70's. No es hasta la invención del sistema conocido como DES (Data Encryption Standard) en 1976 que se da a conocer ampliamente, principalmente en el sector industrial y comercial. Posteriormente se crea el sistema RSA (Rivest, Shamir, Adleman) en 1978, de ahí en adelante se da el comienzo de la criptografía en un gran número de aplicaciones: en transmisiones militares,

en transacciones financieras, en comunicación de satélite, en redes de computadoras, en líneas telefónicas, en transmisiones de televisión, entre otros.

La criptografía se divide en dos grandes grupos, la criptografía de clave privada o simétrica y la criptografía de clave pública o asimétrica, DES pertenece al primer grupo y RSA al segundo.

La criptografía simétrica se refiere al conjunto de métodos que permiten tener comunicación segura entre las partes siempre y cuando anteriormente se hayan intercambiado la clave, o comúnmente llamada clave simétrica. La simetría se refiere a que las partes tienen la misma llave tanto para cifrar como para descifrar. Este tipo de criptografía se conoce también como criptografía de clave privada o criptografía de llave privada.

La criptografía simétrica ha sido la más usada en toda la historia, ésta ha podido ser implementada en diferentes dispositivos, manuales, mecánicos, eléctricos, hasta los algoritmos actuales que son programables en cualquier computadora. La idea general es aplicar diferentes funciones al mensaje que se quiere cifrar de tal modo que solo conociendo una clave pueda aplicarse de forma inversa para poder así descifrar. Aunque no existe un tipo de diseño estándar, quizá el más popular es el de Fiestel, que consiste esencialmente en aplicar un número finito de interacciones de cierta forma, que finalmente da como resultado el mensaje cifrado. Este es el caso del sistema criptográfico simétrico más conocido: DES.

La criptografía asimétrica es por definición aquella que utiliza dos claves diferentes para cada usuario, una para cifrar que se le llama clave pública y otra para descifrar que es la clave privada. El nacimiento de la criptografía

asimétrica se dio al estar buscando un modo más práctico de intercambiar las llaves simétricas. Diffie y Hellman[1], proponen una forma para hacer esto, sin embargo no fue hasta que el popular método de Rivest, Shamir y Adleman RSA publicado en 1978[2], cuando toma forma la criptografía asimétrica, su funcionamiento esta basado en la imposibilidad computacional de factorizar números enteros grandes.

Actualmente la Criptografía asimétrica es muy usada, sus dos principales aplicaciones son el intercambio de claves privadas y la firma digital, una firma digital se puede definir como una cadena de caracteres que se agrega a un archivo digital que hace el mismo papel que la firma convencional que se escribe en un documento de papel ordinario. Los fundamentos de la criptografía asimétrica pertenecen a la teoría de números.

1.2.2 Introducción a bases de datos distribuidas

El desarrollo de los sistemas de información en los últimos años y el crecimiento gigantesco de la información en las empresas ha originado la dispersión de los datos en diferentes sitios locales o geográficamente separados. La necesidad de integrar y manejar esos datos ocasiona el

[1] DIFFIE, W. y HELLMAN W. New Directions in Cryptography. Transactions on Information Theory Vol IT22 No 6. 1976. p. 644-654.

[2] RIVEST, R.L.; SHAMIR A. y ADLEMAN L. A Method for Obtaining Digital Signature and Public-Key Cryptosystems. Communication of the ACM Vol 21 No 2. 1978. p. 120-126.

nacimiento de una tecnología capaz de administrar esa información. Una tecnología que se adapta a la situación expuesta son las Bases de Datos Distribuidas.

Una Base de Datos Distribuida[3] es una colección de datos relacionados lógicamente, pero dispersos sobre diferentes sitios de una red de computadoras, cada sitio en la red tiene capacidad de procesamiento autónomo y puede ejecutar aplicaciones locales, las computadoras no comparten memoria ni discos y pueden variar en tamaño y poder de procesamiento.

Recientemente por la naturaleza dispersa de la información, motivos económicos, interconexión de bases de datos existentes, rendimiento de la base de datos, consistencia y disponibilidad, se percibe una tendencia hacia la distribución de los sistemas en diferentes sitios interconectados a través de una red de computadoras. Esta distribución ocasiona un aumento en la complejidad del diseño e implementación de los sistemas de información.

Las bases de datos distribuidas se encuentran normalmente en sitios geográficos diferentes que se administran en forma independiente y tienen una conexión más lenta. En una base de datos distribuida se pueden dar dos tipos de transacciones: locales y globales[4]. Una transacción local es aquella que accede a los datos de la computadora donde se inicio la transacción; y una transacción global es aquella que accede a los datos ubicados en una

[3] CERI, S y PELAGATTI, G. Distributed Databases principles and systems. McGraw-Hill. 1985.

[4] SILBERSCHATZ, Abraham; KORTH, Henry y SUDARSHAN, S. Fundamentos de Bases de Datos. 3 ed. McGraw-Hill. 1999. p. 409.

computadora diferente de donde se inicio la transacción, o bien accede a los datos de varias computadoras.

Considerando una relación ***r*** que debe guardarse en la base de datos, existen varios enfoques en el almacenamiento distribuido de los datos: Réplica y Fragmentación.

La réplica de los datos genera varias copias idénticas de la relación, cada réplica se guarda en una computadora diferente, se deberá guardar una copia de la misma en dos o más computadoras. La réplica presenta un mayor rendimiento en la lectura de los datos, un aumento del paralelismo y un aumento de la sobrecarga en las actualizaciones.

La fragmentación divide la relación en varios fragmentos, cada fragmento se guarda en una computadora diferente. Existen dos tipos de fragmentación: Horizontal y Vertical. La fragmentación horizontal divide la relación asignando cada tupla de ***r*** a uno o varios fragmentos; y la fragmentación vertical divide la relación en grupos de columnas, donde cada grupo conforma un fragmento.

1.2.3 Seguridad en bases de datos distribuidas

La necesidad de controlar el acceso es uno de los puntos a los cuales se le dedica mucho esfuerzo en los sistemas compartidos, especialmente en los sistemas distribuidos, ya que se requiere de la implementación de procesos y procedimientos muy estrictos. La complejidad del manejo de información

compartida aumenta cuando los usuarios de la misma se encuentran en cualquier localidad diferente a donde se encuentran los datos, por lo cual hay la necesidad de proteger los datos e información de manera que personas o sistemas no autorizados no puedan leer o modificarlas y que el acceso sea denegado a personal no autorizado. Una forma de controlar el acceso es a través de la implementación de políticas de control de acceso, como: Control de Acceso Discrecional (Discretionary Access Control - DAC), Control de Acceso Obligatorio (Mandatory Access Control - MAC), Control de Acceso basado en Roles (Role-Based Access Control - RBAC), Control de Acceso basado en Tareas (Task-Based Access Control - TBAC) y Control de Acceso basado en Restricciones (RBAC). Las políticas de acceso mas populares y que se utilizan en la actualidad son[5]: MAC, DAC y RBAC;

MAC, consiste en la clasificación de tanto los sujetos como los objetos en el sistema. Una clase de acceso es asignada a cada objeto y a cada sujeto. Una clase de acceso es un elemento de un conjunto de clases parcialmente ordenadas. Mientras que la forma más general de conjunto de clases de acceso es un conjunto de etiquetas, a veces se define como un conjunto formado por dos componentes, un nivel de seguridad y un conjunto de categorías.

DAC, esta estrategia de control de acceso está basada en la idea de que los sujetos acceden a los objetos en base a su identidad y a unas reglas de autorización, que indican para cada sujeto, las acciones que se pueden realizar sobre cada objeto del sistema. Con esta estrategia, si un usuario desea realizar una operación sobre un objeto, se busca en el sistema una regla de autorización que le dé permiso para realizar esa operación sobre

[5] JEONG, Min-A; KIM, Jung-Ja y WON, Yonggwan. A Flexible Database Security System using Multiple Access Control Policies. IEEE. 2003. p. 236.

ese objeto, y si no se encuentra se le deniega el acceso. El motivo por el que a este mecanismo de control de acceso se le llama discrecional es porque el usuario puede otorgar la autorización de acceso a otros usuarios.

RBAC, en esta política los permisos se asocian con roles, y a los usuarios se les hace miembros de los roles. De este modo los usuarios consiguen permisos. Esto simplifica considerablemente la gestión de permisos. Los roles representan a cada grupo funcional de las organizaciones, agrupando en cada uno de ellos a aquellos usuarios que realizan funciones y tienen responsabilidades similares.

La política MAC carece de flexibilidad para satisfacer condiciones en un control de acceso complejo[6]. A pesar que la política DAC puede controlar mejor la flexibilidad en el acceso que MAC, no puede controlar un flujo ilegal de usuarios no autorizados. RBAC puede proveer métodos simples de manejo de seguridad, además prevenir el abuso de accesos correctos permitiendo solamente la menor cantidad de privilegios al usuario.

En el año 2003, fue publicado en la IEEE el artículo “A Flexible Database Security Siystem using Múltiple Access Control Policies” , en el cual los autores (Min-A Jeong, Jung-Ja Kim y Yonggwan Won) proponen un sistema de seguridad para bases de datos que puede controlar individualmente el acceso de usuarios a diferentes grupos de datos y es conveniente para el caso en el que los privilegios de acceso de usuarios a datos arbitrarios se cambian frecuentemente. El sistema propuesto trabaja en su primer fase con una modificación a los modelos MAC y RBAC; en la segunda fase con una modificación del modelo DAC.

[6] LINDGREEN, R. y HERSCHBERG, I. On the Validity of the Bell-LaPadula Model. Computer & Security. Vol. 13. 1994. p. 317-338.

Los trabajos realizados hasta el momento apuntan a la seguridad de los datos verificando los accesos de usuarios en el mismo servidor donde está la información. En este proyecto se propone un sistema para colaborar en el proceso de autenticación de usuarios, te tal manera que si alguien realiza una solicitud para acceder a la base de datos, el sistema verifica la información y, si no es correcta, la solicitud es rechazada sin que tenga que intervenir el servidor de base de datos.

1.2.4 Concepto, características y usos de agentes móviles

Un Agente Móvil[7] es un objeto especial que tiene un estado de datos (otros objetos no agentes, estructuras y bases de datos), un estado de código (las clases del agente y otras referencias a objetos) y un estado de ejecución (el control de procesos que se ejecutan en el agente).

La característica principal de un agente móvil es que accede a un conjunto de métodos que le permiten moverse de un servidor a otro. Un agente móvil tiene: un programa que define las actividades que ejecutará y un grado de inteligencia que, además de ayudarlo a interactuar con los usuarios, lo ayuda a la resolución de tareas adversas de su entorno.

[7] FRITZ, Hohl. An approach to solve the problem of malicious hosts in mobile agent systems. Institute of parallel and distributed systems, University of Stuttgart. Alemania. 1997.

Las características básicas de un agente móvil son: autonomía, movilidad, concurrencia, direccionabilidad, continuidad, reactividad, sociabilidad y adaptabilidad.

Una de las principales motivaciones para el uso de los agentes móviles es su aplicabilidad en sistemas distribuidos. Danny Lang presenta siete ventajas que se obtienen al utilizar agentes móviles:

1. Reducen el tráfico de red.
2. Superan el estado latente de las redes.
3. Encapsulan protocolos.
4. Se ejecutan de forma asíncrona y autónoma.
5. Se adaptan dinámicamente.
6. Son heterogéneos por naturaleza.
7. Son robustos y tolerables a fallas.[8]

Entre las aplicaciones de los agentes móviles, Venners Bill cita las siguientes:

- Recolección de datos de distintos sitios.
- Búsqueda de filtrado.
- Monitorización.
- Comercio electrónico.
- Procesamiento paralelo.
- Distribución masiva de información.
- Interacción y negociación.

[8] LANGE, Danny. Mobile Agents: Enviroments, Technologies, and applications. Proceedings of the Practical Applications of intelligent Agents and Multi-Agent Technology Conference, PAAM98. 1998.

- Cálculos en multiprocesos.
- Redes parcialmente desconectadas.
- Entretenimiento y entrega de correo inteligentemente.[9]

Una de las principales plataformas para el desarrollo de agentes móviles es JAVA dado que maneja de manera eficiente la seguridad y portabilidad por su característica de lenguaje interpretado (su salida no es un ejecutable, sino un código binario ejecutado por una máquina virtual que emula el interprete de Java). Vale la pena mencionar algunos sistemas que permiten la implementación de agentes móviles: Aglets, Java-to-go, ARA, Voyager, Mole, Agent TCL, Odyssey, TACOMA, entre otros.

1.2.5 Seguridad en agentes móviles

Las aplicaciones basadas en agentes móviles no deben implementarse hasta que se tomen medidas mínimas de seguridad en la red, como tener un firewall y brindar protección a los servidores contra código malicioso que viene en un agente móvil, todo esto se puede prevenir con controles y protecciones adecuadas para el acceso en ambientes de ejecución.

El tema de la seguridad de agentes móviles ha sido un factor fundamental para que no sea aceptado en su totalidad en Internet, pero hay plataformas para la creación de agentes móviles que permiten alcanzar un nivel de

[9] BILL, Venners. Solve real problems with aglets, type of mobile agent. JavaWorld – Under the hood Magazine. Mayo 1997. Pág. 2-4.

seguridad aceptable. Los Aglets son capaces de restringir el acceso de los agentes generalizándolos en dos grupos: confiables y no confiables.

1.2.6 Acceso a bases datos distribuidas mediante agentes móviles

La tecnología de agentes móviles[10] soluciona diversos problemas en varias áreas. Por un lado, proporciona una solución al derroche de ancho de banda que se produce en la red. Este ancho de banda en una aplicación distribuida es un bien escaso y, por tanto, valioso. Una transacción o consulta realizada entre un cliente y el servidor puede requerir bastantes viajes por la red para completarse, cada uno de los cuales provoca un cierto tráfico de datos y consume ancho de banda.

En un sistema en el que se tenga muchos clientes o mucho volumen de transacciones (o ambas cosas a la vez) posiblemente se sobrepase el ancho de banda disponible, lo que se traducirá en una disminución del rendimiento de la aplicación completa. Aplicando a este problema la tecnología de agentes móviles se puede contar con un agente que, dada la consulta o transacción a realizar, se trasladase desde el cliente al servidor, complete en él la operación y regrese con los resultados de la misma, necesitando de esta manera sólo dos viajes por la red al eliminar todo el tráfico intermedio de datos y resultados.

[10] ACEBAL, César F. y CUEVA, Juan M. Acceso a bases de datos distribuidas mediante el uso de agentes móviles. NOVATICA Julio/Agosto 2000 No. 146. p. 48.

Los trabajos mas recientes con agentes móviles se han desarrollado en código Java, dos de esos trabajos tratan como acceder a base de datos distribuida basada en la tecnología de agentes móviles. El primero: “A Mobile Agent based for Distributed Database Access on the Internet”[11], donde se presenta un sistema basado en agentes móviles para acceder a bases de datos distribuidas en Internet, allí adoptan las funciones de trabajo de un agente master-worker para manejar las transacciones distribuidas del protocolo three-tier commit(3TC), el cual puede reducir significativamente el embotellamiento de la comunicación. El sistema trabaja con un agente móvil entre el cliente y el servidor. Cuando el agente móvil viaja a través de la red, el cliente original, el cual puede ser un servidor móvil, se puede desconectar de la red, ya que esta conexión es requerida sólo cuando se lanza y se rescata los resultados desde el agente móvil. Cuando el agente llega hasta el servidor con la petición, se realiza un tipo de acceso local. Para acceder a la base de datos local, el sistema usa el puente JDBC-ODBC, el cual puede transformar las llamadas API JDBC en llamadas ODBC y enviarlas a un driver ODBC previamente instalado en el sitio destino. Al adoptar este tipo de acceso a los datos, se tiene la ventaja de que el acceso a la base de datos es independiente de cualquier driver JDBC, luego las tareas distribuidas pueden ser conducidas sobre un gran grupo de DBMSs de diferentes marcas. Con el uso de la tecnología de agentes móviles, la conectividad requerida entre clientes y servidores es poca, ya que un agente puede migrar automáticamente a otro sitio para comunicarse con el agente master-worker mediante el envió de un nuevo agente o enviando mensajes ya sea para mandar resultados u obtener nuevas instrucciones. Además de lo anterior, las transacciones distribuidas pueden ser enviadas a múltiples agentes que

[11] YAN, Wang; KEN, Law y TAN, Kian-Lee. A mobile agent based system for distributed database access on the internet. Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference on , Vol. 2, 22-25 Aug. 2000 Pages:1587 - 1590.

pueden trabajar en paralelo para promover una ejecución eficiente de tareas, cosa que no se puede hacer con los tradicionales Applets, ya que estos deben trabajar en serie y acceder a los sitios de uno en uno para ejecutar una tarea distribuida.

El segundo: "Mobile Agents for WWW Distributed Database Access"[12], donde se presenta un sistema para el desarrollo de aplicaciones cliente/servidor distribuidas sobre la Web basadas en agentes móviles. El sistema lo llaman "DBMS-Aglet Framework". Al igual que el primer sistema, este también propone agentes móviles entre el programa cliente y el servidor pero utiliza las nuevas tecnologías de agentes móviles y demuestra ser mas flexible, escalable y robusto que la actual conectividad a base de datos basada en JDBC (especificación estándar de Java para acceso y manipulación de bases de datos relacionales). El sistema propuesto crea y arranca uno o varios agentes móviles que viajan directamente al servidor SQL remoto, allí el agente inicia un driver JDBC local, se conecta a la base de datos y realiza cualquier consulta especificada por el cliente que lo envió. Cuando el agente completa la tarea, el mismo se regresa directamente a la máquina cliente en el Applet-DBMS desde donde fue inicializado, creado y arrancado.

Los dos sistemas anteriormente descritos muestran claramente el uso que le dan a la tecnología de agentes móviles, en donde el cliente hace una petición y uno o varios agentes toman esa petición, viajan a través de la red hasta el servidor, luego realiza lo solicitado y devuelve una respuesta al cliente. Lo que se propone alcanzar en este proyecto está antes de lo que proponen estos sistemas, ya que el proyecto consistiría en un sistema de agentes

[12] PAPASTAVROU, Stavros; SAMARAS, George y PITOURA, Evaggelia. Mobile agents for WWW distributed database access. Data Engineering, 1999. Proceedings., 15th International Conference on , 23-26 March 1999 Pages: 228 – 237.

móviles que trabajarían como un proceso colaborativo en la autenticación de usuarios, es decir que cuando llegue una petición de un cliente para acceder a la base de datos, sea el sistema de agentes propuesto quien decide si esa petición llega o no hasta el servidor, esto dependiendo de los registros de su base de conocimientos de los usuarios.

1.3 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

El tema de seguridad de la información siempre es de gran interés mas aún cuando se habla de ambientes en donde dicha información no se encuentra en un mismo lugar, sino que está distribuida en diferentes sitios geográficos que manejan múltiples plataformas. En el manejo de bases de datos distribuidas es de vital importancia el control que se hace sobre los accesos a las mismas, es decir, los datos almacenados deben estar protegidos contra una pérdida accidental, contra el acceso malintencionado y no autorizado a los mismos.

La seguridad de las bases de datos suele hacer referencia a la protección contra los accesos malintencionados. Este es el aspecto que motiva a la realización de este proyecto, el cual pretende mostrar el uso de agentes móviles como parte colaborativa en el manejo de la seguridad de un sistema de base de datos distribuido.

Si alguien no autorizado realiza una solicitud para acceder a la base de datos, el sistema verifica la información y, si no es correcta, la solicitud es rechazada. Muchas solicitudes de acceso viajan por la red hasta llegar al servidor para realizar el proceso de verificación de datos, por lo que se

aumenta el tráfico en el canal y se desgasta el sistema atendiendo solicitudes que podrían ser evitadas dados los antecedentes de las peticiones.

Por lo anterior, en este proyecto se propone diseñar e implementar mediante un sistema de agentes móviles una herramienta que, además de disminuir en gran medida el tráfico en el canal, garantice al servidor de bases de datos que no lleguen peticiones con origen anteriormente rechazadas por su no-autorización.

Dado que el uso de agentes móviles en el acceso a bases de datos distribuidas ha mostrado ser flexible, escalable y robusto, surge la inquietud sobre la implementación de un sistema de agentes móviles que capture todas las peticiones de acceso que se hagan a una base de datos distribuida y, mediante una base de conocimiento pueda tomar decisiones de rechazar o permitir la autenticación a los usuarios, de esta manera, sólo lograrán llegar al servidor de base de datos las peticiones que el agente considere según su base de conocimientos.

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 OBJETIVO GENERAL

Diseñar e implementar un software para autenticar el acceso a una base de datos distribuida usando agentes móviles.

1.4.2 OBJETIVOS ESPECÍFICOS

- Diseñar y crear una base de datos distribuidas para realizar pruebas.

- Diseñar e implementar un sistema de agentes móviles que capture peticiones de acceso a una base de datos distribuida y sea capaz de tomar decisiones de rechazo o aceptación según su base de conocimiento.

- Realizar pruebas con el sistema de agentes móviles en la base de datos de prueba.

1.5 RESULTADOS ESPERADOS

- Sistema de agentes móviles que colabora en la autenticación para acceder a una base de datos distribuida.

- Estadísticas de evaluación del rendimiento del proceso de autenticación de acceso a la bases de datos distribuida usando el sistema de agentes móviles frente al proceso normal de autenticación.

- Artículo científico en inglés para ser publicado posteriormente en una revista indexada reconocida.

1.6 ACTIVIDADES

Para el logro de los objetivos planteados, se llevarán a cabo las siguientes actividades:

- Estudio de herramientas para implementar agentes móviles.
- Capacitación en diseño y creación de bases de datos distribuidas.
- Creación de una base de datos distribuida para realizar pruebas.
- Diseño del sistema de agentes móviles.
- Implementación y pruebas del sistema de agentes móviles con la base de datos.
- Elaboración del documento de tesis.
- Elaboración de un artículo científico para mostrar resultados del proyecto.

2. TECNOLOGÍA DE AGENTES MÓVILES

Actualmente se están realizando a nivel mundial investigaciones que conllevan al desarrollo de nuevas aplicaciones basadas en agentes móviles, ampliando así las diversas áreas donde es aplicable la tecnología de agentes móviles (Aplicaciones Web, Sector Industrial, Ingeniería del Software, Automatización y Control, entre otras). La finalidad de este capítulo es presentar un estudio de la tecnología de agentes móviles, iniciando desde conceptos básicos, historia y evolución hasta presentar diferentes herramientas que existen en el mercado para la creación de los agentes. Este estudio nos ayudará a decidir cuál de las herramientas estudiadas se amoldará mejor para el desarrollo de esta tesis.

2.1 CONCEPTOS BASICOS

2.1.1 Agente.

Según Wooldridge y Jennings: “Un Agente es un sistema de hardware o (más usualmente) un sistema de computadora basado en software que tiene las siguientes propiedades:

- Autonomía: los agentes operan sin la directa intervención del usuario, y tiene cierto control sobre sus acciones y su estado interno.

- Habilidad social: los agentes interactúan con otros agentes (y posiblemente humanos) usando algún tipo de lenguaje de comunicación de agentes.
- Reactividad: los agentes perciben su entorno (que puede ser el mundo físico, el usuario a través de una interfaz gráfica, una colección de otros agentes, INTERNET, o una combinación de estos), y responden a los cambios que ocurren en él.
- Pro-actividad: los agentes no actúan solamente en respuesta a su entorno, sino que son capaces de exhibir un comportamiento dirigido por objetivos tomando la iniciativa."[13]

El término agente o agente inteligente para algunos investigadores, especialmente para los que trabajan en el área de Inteligencia Artificial, tiene un significado mucho más amplio. La mayoría de estos investigadores establecen que un agente debe ser como un sistema computacional que además de tener las propiedades mencionadas por Wooldridge y Jennings también tengan propiedades aplicadas al ser humano, propiedades como racionalidad y adaptación.

2.1.2 Agente móvil.

Un agente para que sea catalogado como agente móvil debe ser capaz de ejecutarse en diferentes computadoras, es decir, que pueda suspender su ejecución en un servidor y reanudarla en otro servidor una vez que se haya desplazado a este.

[13] WOOLDRIDGE, M.; JENNINGS, N. Intelligent Agents: Theory and Practice. Knowledge Engineering Review. Octubre de 1995.

Según Lange y Oshima: “Un agente móvil es una entidad que tiene cinco atributos: estado, implementación, interfaz, identificador y un ente principal (persona u organización)”[14] (Figura 1). Cuando el agente se mueve en la red toma estos atributos y viaja con ellos. El estado lo necesita el agente para reanudar el cómputo después de viajar. La implementación se necesita para la ejecución del agente independiente de su ubicación. La interfaz se necesita para comunicación del agente. El identificador se necesita para reconocer y ubicar al agente mientras viaja. Y el ente principal determina la responsabilidad moral y legal.

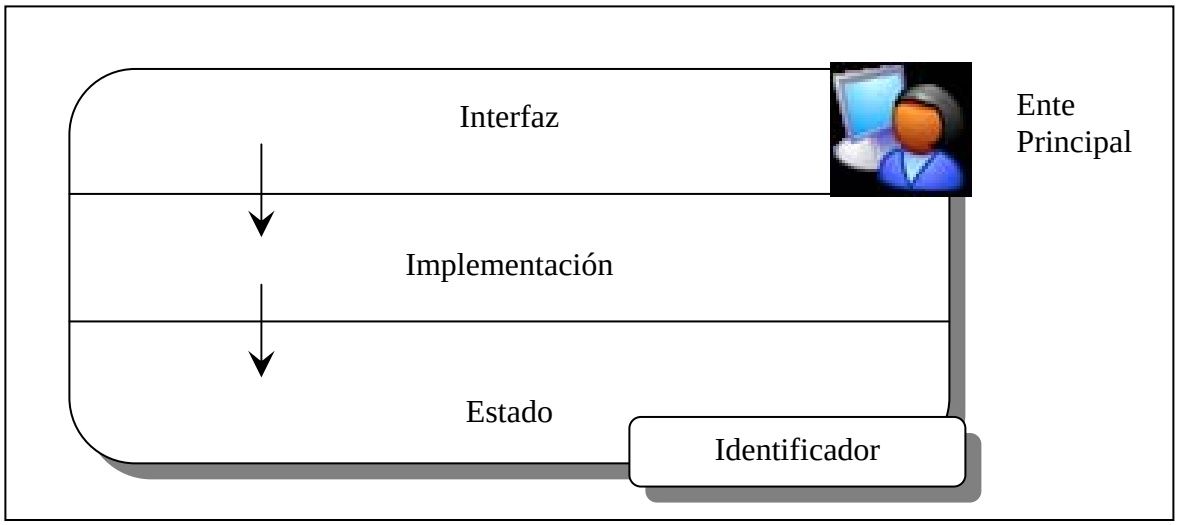


Figura 1.Propiedades de los Agentes

[14] LANGE, Danny y OSHIMA, Mitsuru.Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley. 1998. p.17.

2.2 HISTORIA Y EVOLUCIÓN DE LOS AGENTES MÓVILES

La evolución de los agentes ha sido muy variada, ya que depende del ámbito en el que se desarrollan. El estudio de agentes comenzó en aplicaciones de inteligencia artificial, sin embargo, con el advenimiento de Internet han nacido nuevas tendencias, por lo cual, los agentes guardan una gran relación con Internet.

Una de las primeras compañías que invirtió en agentes fue General Magic[15], que fue fundada en 1990 para crear una tecnología basada en agentes para dispositivos de computación portables; esta tecnología llamada Telescrip fue desarrollada por AT&T, Motorola y algunos otros.

Posteriormente crearon Tabriz AgentWare, el cual ejecutaba y gestionaba aplicaciones basadas en agentes que estaban en servidores y Telescrip fue adaptado para trabajar con servidores en Internet.

Mientras Telescrip era utilizado en el desarrollo de agentes móviles, se creaba el Lenguaje de Manipulación y consultas de Conocimiento (KQML, el cual no es un sistema de agentes móviles, sino para compartir conocimiento). De igual manera se desarrollaba el lenguaje de Agent TCL en Darmouth Collage, Agent for Remote Action (ARA) en la universidad de Kaiserslautern, Alemania y Penguin. El poco éxito de algunos de estos sistemas de agentes radica en que son basados en lenguajes propietarios o poco difundidos, en donde para poder realizar una aplicación basada en agentes móviles había que aprender el propio lenguaje de codificación, además, una aplicación hecha en Telescrip, por ejemplo, no podía interaccionar con otra hecha en ARA, por el hecho de ser propietarios.

[15] Lawton George. Agent to roam the Internet. Sunworld Online Magazine. Oct. de 1996.

Entre 1995 y 1996, el número de servidores en Internet casi se duplicó, así que los agentes de Internet que ya existían tomaron más fuerza, además, se presenta el gran éxito del lenguaje Java cuya mayor ventaja era la independencia de la plataforma y ofrecía una base segura para el desarrollo de agentes móviles. Java brindaba la invocación de métodos remotos (RMI) y la serialización para implementar sistemas de agentes móviles.

Debido a la necesidad que una tecnología ayudara a resolver el tráfico y la independencia de tareas rutinarias en Internet, en los últimos años se ha dado una gran proliferación en los sistemas de agentes móviles, entre los que se encuentra:

- Agent TCL (Darmouth College)
- Aglets (IBM)
- ARA - Agent for Remote Action (University of Kaiserslautern)
- JATLite (Stanford University)
- Java Agent Template (H. Robert Frost)
- Concordia (Mitsubishi Electric)
- JADE (Universidad de Parma)
- Odyssey (General Magic)
- Voyage (ObjectSpace)
- Tacoma

La mayor parte de estos sistemas de agentes móviles están implementados en Java, aunque todavía prevalecen algunos en lenguaje propietario.

El paradigma de agentes móviles incrementa la sofisticación de los tipos posibles de comunicación sin restringir el ancho de banda disponible de los componentes de Internet[16].

La tecnología de agentes móviles es una infraestructura de software que puede montarse encima de una gran variedad de computadoras y hardware de comunicaciones. Tiene tres componentes principales: el lenguaje en el cual los agentes son programados, una máquina o interprete para ese lenguaje y los protocolos de comunicación que permite a esa máquina residir en diferentes computadoras para lograr el envió e intercambio de agentes.

El lenguaje Java se ha convertido en la mejor tecnología para el desarrollo de agentes móviles, esto debido a que Java dispone de casi todo lo necesario para crear agentes que puedan navegar por Internet, ya que ofrece portabilidad, seguridad, serialización de objetos, RMI (Remote Method Invocation), ejecución de múltiples hilos concurrentemente y ubicuidad.

Los lenguajes orientados a objetos poseen características que permiten satisfacer parte de los requerimientos de la programación orientada a agentes[17] y en particular de agentes móviles[18]. Un agente puede modelarse mediante un objeto cuyos métodos representan sus habilidades, y las variables de instancia su estado mental. De esta forma, un objeto encapsulará, en sus métodos capacidades de

[16] Jim. Mobile Agent White Paper. General Magic, 1996.

[17] Shohan, Y. Agent-Oriented Programming. Artificial Intelligence, 60(1), p.51-92, Marzo de 1993.

[18] D. Wong, N. Paciorek and D. Moore. Java-based Mobile Agents . Communications of the ACM. Vol. 42. No. 3, p. 92–102, Marzo de 1999.

comportamiento de un agente y su conocimiento estará dado por el estado de sus variables de instancia.

En la siguiente sesión se hace una descripción de los sistemas mas usados para el desarrollo de Agentes Móviles.

2.3 SISTEMAS PARA EL DESARROLLO DE AGENTES MÓVILES

2.3.1 Aglets

Uno de los principales marcos de trabajo para agentes Móviles, desarrolladas en Java, son los Aglets, creados por el equipo del laboratorio de desarrollo de IBM en Tokio[19]. Un Aglet tiene una serie de estados que definen su funcionamiento. Los principales eventos en la vida de un Aglet son: Creación o Clonación; Liberación de recursos; Movilidad; y Persistencia (permite la desactivación y activación del Aglet).

El servidor crea un contexto para los Aglets. Allí serán insertados todos los Aglets que llegan al servidor, además tiene un monitor de red que permite monitorear la red buscando por otros Aglets, y un administrador de seguridad que protege al sitio (Figura 2).

[19] LANGE, Danny y OSHIMA, Mitsuru. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley. 1998.

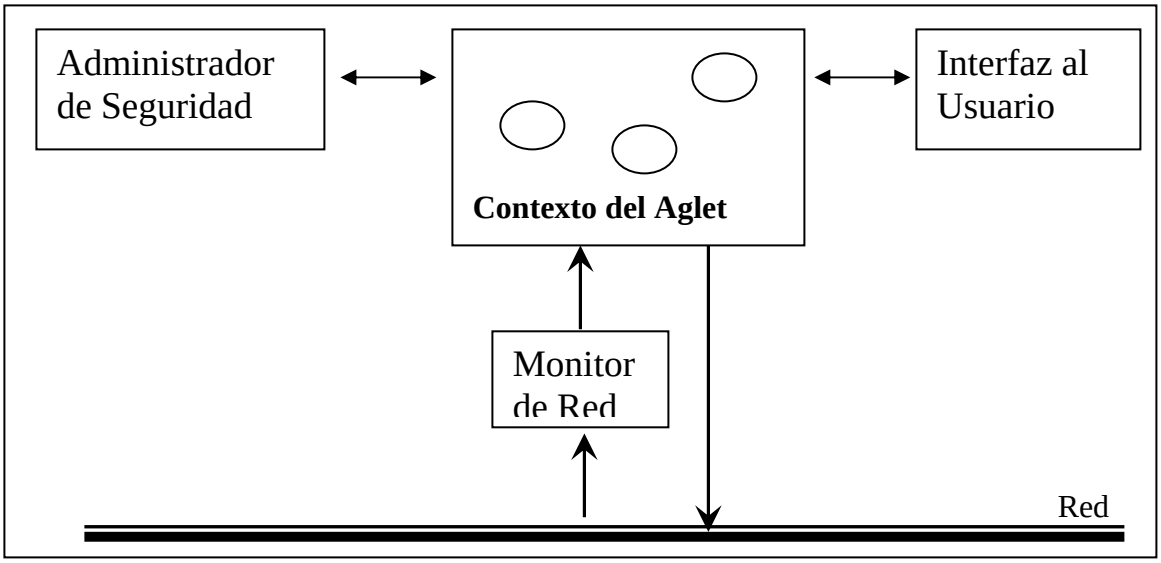


Figura 2. Contexto de los Aglets, Interfaz al usuario y Monitor de red.[7]

Los Aglets pueden enviarse mensajes, aún sin “conocerse” entre ellos, a través del empleo de un framework incorporado para tal fin que permite independencia de la ubicación, soporta mensajes sincrónicos y asincrónicos, y puede ser extendido. Para lograr la independencia de ubicación emplea un mecanismo de Proxy, de esta manera cada agente envía el mensaje al Proxy en lugar de hacerlo al destinatario.

La estructura definida en los Aglets permite que estos actúen autónomamente colaborando con otros Aglets. La colaboración es una necesidad en los agentes porque en general ninguno está aislado ni puede realizar tareas complejas solo. La movilidad y el paso de mensajes son características importantes para la colaboración.

Los Aglets no pueden migrar con su estado: memoria, pila de ejecución y registros, ya que están implementados en Java donde un programa no tiene acceso directo para manipular su estado (definido en el modelo de seguridad de la

máquina virtual Java - JVM). Por lo tanto, cuando el Aglet va a ser transferido, debe almacenar todo lo necesario para luego retomar su estado cuando es reactivado, esto es llamado movilidad débil.

Al utilizar los Aglets, se tiene un modelo de fácil comprensión para la programación de agentes móviles sin tener que hacer modificaciones en la máquina virtual de Java o código nativo. También se tiene el soporte de una comunicación dinámica que permite a los agentes comunicarse con otros agentes tanto conocidos como desconocidos. Además, se proveen mecanismos de seguridad que son de fácil comprensión y lo suficientemente simples para permitir que los usuarios finales puedan especificar una serie de restricciones de acceso a los agentes móviles.

2.3.2 Concordia

Concordia es un framework implementado en Java creada por Mitsubishi Electric que provee un modelo para trabajar con aplicaciones móviles[20]. Está compuesto por un conjunto de clases que permite: la ejecución del servidor, el desarrollo de la aplicación del agente y la activación del agente en si.

Tiene mecanismos de protección para los servidores y agentes. La protección de agentes consiste en evitar que se viole la integridad de la información contenida en un agente móvil durante su paso por la red o cuando es almacenado en un disco local. Para lograr esta funcionalidad, antes de trasmitir un agente de un sitio

[20] WALSH, Tom; PACIOREK Noemi y WONG, David. Security and Reliability in Concordia. Mitsubishi Electric ITA, Horizon System Laboratory. USA, 1998.

a otro sobre una red, *Concordia* codifica los *bytecodes* correspondientes al agente, los datos del usuario y la información de estado usando el mecanismo de *clave pública*. Luego los servidores *Concordia* se autentican uno a otro intercambiando certificados digitales.

Principalmente el framework permite que las aplicaciones accedan a información distribuida en cualquier momento. Las aplicaciones pueden procesar información aún si el usuario está desconectado de la red, usando diversos tipos de redes (LAN, Intranet, Internet) y dispositivos de clientes múltiples (PC, PDA, teléfonos).

Las características más importantes de los agentes definidos en *Concordia* son: movilidad y acceso a datos eficiente. Los agentes tienen reflejado en su estado un itinerario a seguir y la consulta que realizarán en la base de datos. Sin embargo, es el usuario el que crea y envía el agente, indicando eventualmente su itinerario (también puede ser programado en la aplicación). Con lo cual estos “agentes” no resultan tan autónomos como para ser llamados agentes, aunque es posible extender la funcionalidad definida en el framework para lograr la autonomía deseada.

2.3.3 JATLite

Este framework fue creado por la Universidad de Stanford se utiliza para la construcción de sistemas multi-agente, está implementado en Java y provee un conjunto de paquetes Java con:

1. Herramientas para la comunicación, a través del intercambio de mensajes KQML usando TCP/IP.

2. Servidor de nombres de agentes, con la información de todos los nombres y direcciones de los agentes existentes.

3. Funcionalidad para que los Applets de Java intercambien mensajes con cualquier agente registrado en Internet.

En general, provee la funcionalidad básica para construir aplicaciones de sistemas multiagente. Tanto el servidor de nombres como la funcionalidad para el intercambio de mensajes son centralizados lo que puede producir un cuello de botella en el sistema cuando se trata de una aplicación real con varios agentes ejecutándose simultáneamente[21].

La arquitectura por niveles de JATLite está formada por los niveles (de mayor a menor nivel de abstracción): *Protocolo* (soporta SMTP, FTP), *Ruteo* (registro de nombres), *KQML* (almacenamiento y reconocimiento de mensajes en KQML), *Base* (comunicación simple basada en TCP/IP) y *Abstracto* (clases abstractas con los paquetes JATLite).

En general el término “agente” en JATLite es usado para significar una comunidad de “entidades” que intercambian mensajes a fin de desarrollar una tarea (agentes reactivos). Esto significa que cada entidad coopera con las otras a fin de solucionar un problema, aún así JATLite no cuenta con estructuras definidas que le permitan representar el problema en su estado interno, aprender, o decidir que acciones llevar a cabo a fin de lograr autonomía.

[21] O´HARE, G. y JENNINGS, N. Foundations of Distributed Artificial Intelligence. John Wiley & Sons. 1996.

2.3.4 Odyssey

General Magic Inc. inventó el agente móvil y creó Telescrip, el primer sistema de Agente Móvil comercial. Basado en un lenguaje propietario y arquitectura de red, Telescrip tuvo una vida corta. Debido a la expansión de Internet y al éxito del lenguaje Java, General Magic decidió reimplementar el paradigma de Agente Móvil en Odyssey basado en java. Este sistema implementa efectivamente los conceptos de Telescrip en clases de Java. Como resultado se obtuvo una librería de clases de Java que habilita a los desarrolladores crear sus propias aplicaciones de Agentes Móviles.

2.3.5 JADE

JADE *Java Agent Development framework*) es una herramienta para la creación de agentes desarrollado por la Universidad de Parma desde 1997 y que sigue en constante revisión y evolución, su última versión es la JADE 3.2 del 26 de Julio de 2004. Es una herramienta implementada completamente en *Java*, cuya finalidad es simplificar el desarrollo de sistemas multiagentes a través de un conjunto de sistemas, servicios y agentes.

JADE fue desarrollado con las especificaciones de FIPA, Foundation for Intelligent Physical Agents. Con JADE los agentes se pueden distribuir a través de las máquinas (que incluso no necesitan compartir el mismo sistema operativo) y la configuración puede ser controlada a través de una GUI remota. La configuración se puede cambiar en tiempo ejecución mientras agentes se mueven de una máquina a otra. JADE necesita de Java para ejecutarse y el requisito mínimo del sistema es la versión 1.4 de JAVA.

Entre las principales características que ofrece JADE se tienen:

- Su distribución es freeware.
- No presenta una interfaz para desarrollo pero sí para el control de la ejecución del sistema de agentes.
- La comunicación entre los agentes se realiza mediante ACL (Agents Comunication Lenguaje), estándar de FIPA.
- La movilidad es posible dentro de la plataforma JADE.
- JADE permite el registro y eliminación automática de agentes con el *AMS (Agent Management System)* en cualquier momento de la ejecución ya sea local o remotamente.

JADE proporciona un conjunto de agentes que simplifican la administración de la plataforma de agentes. Actualmente los que están disponibles son las siguientes:

- RMA, *Remote Management Agent,* actúa como consola gráfica para la gestión y control de la plataforma. Es necesaria para iniciar el resto de herramientas.
- *Dummy Agent* es una herramienta de depuración y visualización, en la que es posible componer mensajes ACL y enviarlos, visualizar la lista de todos los mensajes ACL enviados o recibidos, así como la información contenida en dichos mensajes.
- *Sniffer* es un agente capaz de interceptar mensajes ACL y mostrarlos gráficamente utilizando una notación similar a los diagramas de secuencia de UML. Es bastante útil para depurar las sociedades de agentes mediante la observación de cómo intercambian mensajes ACL.

- *SocketProxyAgent* es un agente simple que actúa como enlace bidireccional entre una plataforma JADE y una conexión TCP/IP, y se utiliza para canalizar los mensajes entrantes y salientes de una plataforma JADE.
- *DF GUI* es interfaz gráfico utilizado junto con el *Directory Facilitator* de JADE para crear complejas redes de dominios y subdominios. Esta interfaz permite controlar el conocimiento de los DFs y registrar, modificar o buscar dentro de ellos.

2.3.6 Voyager (ObjectSpace)

Plataforma para computación distribuida con agentes en Java. Voyager provee un gran conjunto de objetos que se mueven como un agente en la red. Voyager permite a los programadores crear aplicaciones de redes usando tanto la técnica de programación tradicional como la programación distribuida basada en agentes.

2.3.7 ARA - Agent for Remote Action (University of Kaiserslautern)

Basado en el lenguaje Tcl. Es una plataforma para la portabilidad y la ejecución segura de agentes móviles en redes heterogéneas. El proyecto de investigación soporta básicamente sistemas de agentes móviles generales ofreciendo ejecuciones seguras y portables, y no tanto modelos de cooperación, comportamientos inteligentes o modelamiento de usuarios.

2.3.8 Tacoma

Proyecto de la Universidad de Tromso y Cornell enfatiza en sistemas operativos para agentes y la manera en que los agentes pueden ser usados para resolver problemas tradicionales direccionados por sistemas operativos. El sistema Tacoma esta basado en Unix y el protocolo de control de transmisión (TCP). El sistema soporta agentes escritos en C, Tcl/Tk, Perl, Pithon, y Escheme (Elk). El propio sistema está implementado en C.

2.4 ELECCION DE LA PLATAFORMA: AGLETS

De las diferentes plataformas estudiadas para el desarrollo de agentes móviles, Aglets tiene una gran ventaja: la comunicación entre los agentes es directa, es decir, no necesita de ninguna herramienta adicional para que los agentes se comuniquen entre sí, como sucede con sistemas como JATLite de la Universidad de Stanford y otros sistemas para el desarrollo de agentes descritos en este documento, los cuales utilizan KQML, TCP/UDP, sockets, entre otros para la comunicación.

Otra alternativa, en caso de no adquirir la licencia comercial de Aglets, es trabajar con la herramienta JADE, ya que ofrece también la movilidad de los agentes, además, JADE es de libre distribución y está regida por los estándares de la FIPA, Foundation for Intelligent Physical Agents.

3. BASES DE DATOS DISTRIBUIDAS

Las organizaciones son cada vez mas competitivas y exigen productos que resuelvan eficaz y rápidamente sus necesidades. Al mismo tiempo también son cooperativas en proyectos comunes con otras, siendo necesario la integración de datos y procesos. Las fusiones de grandes organizaciones exigen una total integración. Alta disponibilidad, servicio ininterrumpido y arquitecturas abiertas son algunas de las características mas importantes demandadas por las grandes organizaciones, esto se puede palpar, por ejemplo, en el pago con tarjeta o con teléfono móvil, servicio de cajeros automáticos, la comunicación entre sistemas diferentes, entre otros.

Con la llegada de las minicomputadoras, las tareas de procesamiento de datos, tales como el control de inventarios y el procesamiento de pedidos pasaron de mainframes corporativas a sistemas departamentales más pequeños. El explosivo aumento de popularidad de las computadoras personales en los años ochenta llevó la potencia de las computadoras directamente a millones de personas. Conforme las computadoras y las redes de computadoras se extendían a través de las organizaciones, los datos ya no residían en un único sistema bajo el control de un único SGBD (Sistema Gestor de Base de Datos). En vez de ello, los datos se fueron extendiendo a través de muchos sistemas diferentes, cada uno con su propio gestor de base de datos.

Con frecuencia los diferentes sistemas informáticos y los diferentes sistemas de gestión de base de datos procedían de diferentes fabricantes. Estas tendencias

han hecho que la industria informática y la comunidad de gestión de datos enfocaran su atención en los problemas de gestión de bases de datos distribuidas.

Un sistema de computación distribuida consiste en un conjunto de computadores que no necesariamente tienen que ser homogéneos, que están interconectados entre sí formando una red, y que cooperan para realizar una determinada tarea. Un sistema de computación distribuida parte un problema grande en pequeñas piezas, y soluciona cada una de ellas eficientemente de una manera coordinada.

Uno de los primeros sistemas distribuidos fue R*, creado por IBM[22] a principios de los ochenta. El avance de las comunicaciones, la difusión cada día mayor de Internet y la aparición de la informática móvil, el área de las bases de datos distribuidas está basada cada vez más en relación con la tecnología web. Las bases de datos distribuidas son necesarias porque por lo regular las empresas ya están distribuidas, por lo menos desde el punto de vista lógico (en divisiones, departamentos, proyectos, etc.) y muy probablemente en el sentido físico también (en plantas, talleres, laboratorios, y demás), de lo cual se desprende que en general la información ya está también distribuida, porque cada unidad de organización dentro de la empresa mantendrá por fuerza los datos pertinentes a su propio funcionamiento. Así pues, un sistema distribuido permite que la estructura de la base de datos refleje la estructura de la empresa, los datos locales se pueden mantener en forma local, donde por lógica deben estar, pero al mismo tiempo es posible obtener acceso a datos remotos en caso necesario.

La distribución de datos a través de las distintas sedes o departamentos de una organización permite que estos datos residan donde han sido generados o donde son más necesarios, pero continuar siendo accesibles desde otros lugares o

[22] Williams, R. R*: An Overview of the Architecture. IBM Research Report RJ3325. 1998.

departamentos diferentes. El hecho de guardar varias copias de la base de datos en diferentes sitios permite que puedan continuar las operaciones sobre la base de datos aunque algún sitio se vea afectado por algún desastre natural, como una inundación, un incendio o un terremoto. Se han desarrollado los sistemas gestores de bases de datos distribuidos, SGBDD, para manejar datos distribuidos geográfica o administrativamente a lo largo de múltiples sistemas de bases de datos.

3.1 Procesamiento distribuido Vs. SGBDD

Es importante hacer una distinción entre SGBD distribuido (SGBDD) y procesamiento distribuido. El procesamiento distribuido consiste de una base de datos centralizada que puede ser accedida a través de una red. El punto clave de la definición de una base de datos distribuida es que el sistema está formado por datos que están físicamente distribuidos a través de una red. Si los datos están centralizados, incluso aunque los usuarios puedan acceder a los datos a través de la red, no se puede considerar como un sistema distribuido. Las figuras 3 y 4 esquematizan la diferencia planteada:

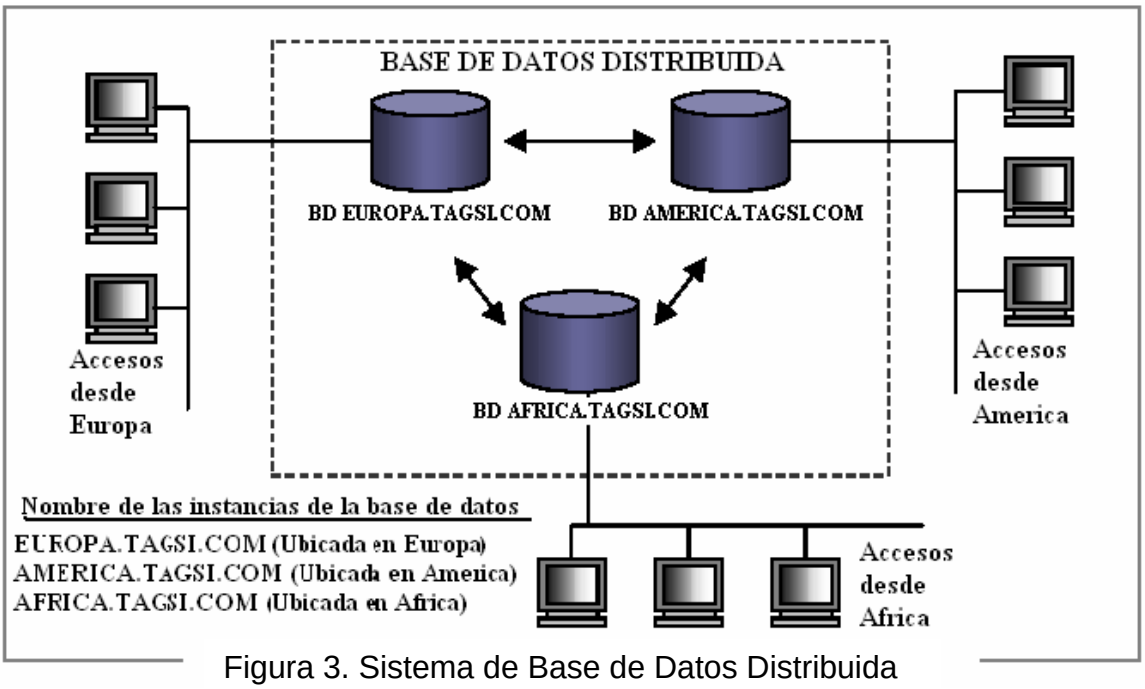


Figura 3. Sistema de Base de Datos Distribuida

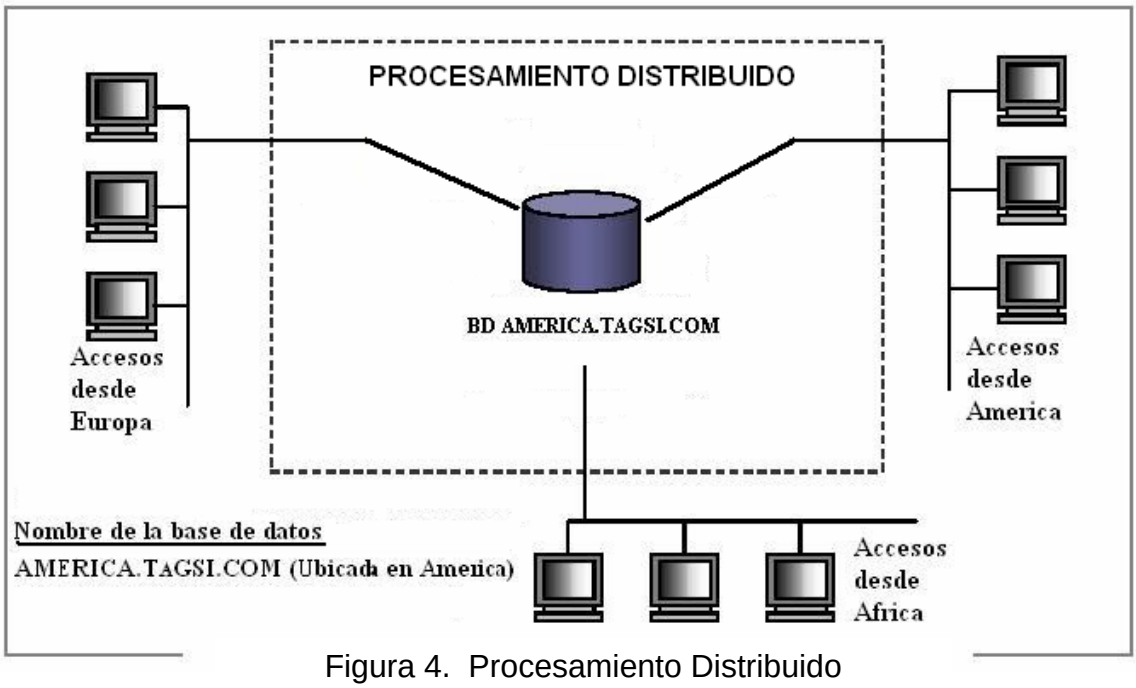


Figura 4. Procesamiento Distribuido

3.2 Ventajas e Inconvenientes de SGBDD

La distribución de los datos y aplicaciones tiene ventajas potenciales con respecto a los sistemas de bases de datos centralizados. Existen varias razones que justifican la construcción de sistemas distribuidos de bases de datos.

La mayor ventaja de los sistemas distribuidos de bases de datos es que proporcionan un entorno en el que los usuarios de un nodo pueden ser capaces de acceder a los datos que residen en otros nodos. Sin esta capacidad, un usuario que deseara realizar una transferencia entre dos nodos distintos tendría que recurrir a algún mecanismo externo para poner en contacto los sistemas existentes.

La ventaja principal del compartimiento de datos por medio de la distribución de los mismos es que cada nodo puede conservar un cierto grado de control sobre los datos que tiene almacenados localmente. En un sistema centralizado hay un administrador local de la base de datos que es responsable del sistema completo. Cada administrador local de las bases de datos recibe una parte de las responsabilidades del administrador central. Cada administrador puede tener un grado de autonomía local diferente, dependiendo del diseño del sistema distribuido de bases de datos. La posibilidad de autonomía local es, con frecuencia, una ventaja fundamental de las bases de datos distribuidas.

Si en un sistema distribuido falla un nodo, los nodos restantes pueden continuar funcionando. En particular si se duplican los elementos de datos en varios nodos, una transacción que necesite un determinado elemento de datos puede encontrarlo en cualquiera de dichos nodos. De esta manera, el fallo de un nodo no

implica necesariamente el cierre del sistema. Deben existir mecanismos para reintegrar fácilmente en el sistema al nodo que falló cuando se haya recuperado o se haya reparado. Aunque la recuperación de fallos es más compleja en los sistemas distribuidos que en los centralizados, la capacidad de la mayoría del sistema para continuar funcionando a pesar del fallo de un nodo revierte en una disponibilidad creciente.

En un entorno distribuido, es mucho más fácil manejar la expansión. Se pueden añadir nuevas localizaciones a la red sin afectar las operaciones de las otras localizaciones. Esta flexibilidad permite que una organización se pueda expandir de manera relativamente fácil.

El principal inconveniente de los sistemas distribuidos de bases de datos es la complejidad añadida que es necesaria para garantizar la coordinación apropiada entre los nodos. La implementación de un sistema distribuido de bases de datos es más difícil y por tanto más costosa.

Las transacciones distribuidas requieren la cooperación activa de dos o más copias independientes del software SGBD que se está ejecutando en sistemas informáticos diferentes. Deben utilizarse protocolos especiales de transacción.

Cuando un SGBD realiza sus tareas, accede frecuentemente a sus catálogos de sistema. ¿Dónde debería mantenerse el catálogo en una base de datos distribuida? Si está centralizado en un sistema, el acceso remoto a él será lento, afectando al SGBD. Si está distribuido a través de muchos sistemas diferentes, los cambios deben propagarse y sincronizarse por toda la red.

Cuando las transacciones de dos sistemas diferentes tratan de acceder a datos bloqueados en el otro sistema, puede ocurrir un interbloqueo en la base de datos

distribuida, incluso aunque el interbloqueo no sea visible a ninguno de los dos sistemas por separado. El SGBD debe detectar interbloqueos globales en una base de datos distribuida.

Si uno de los sistemas donde se ejecuta un SGBD distribuido falla, el operador de ese sistema debe ser capaz de ejecutar sus procedimientos de recuperación con independencia del resto de los sistemas en la red, y el estado recuperado de la base de datos debe ser consistente con el de los otros sistemas.

3.3 Diseño de Bases de Datos Distribuidas

Considerando una relación *r* que debe guardarse en la base de datos. Hay varios enfoques del almacenamiento de esa relación en la base de datos distribuida:

- Réplica. El sistema conserva varias réplicas (copias) idénticas de la relación. Cada réplica se guarda en un emplazamiento o nodo diferente, lo que da lugar a la réplica de los datos. Si falla uno de los emplazamientos que contienen la relación r, ésta aún se podrá encontrar en otro emplazamiento. Por tanto, el sistema puede seguir procesando las consultas que impliquen a r, a pesar del fallo en un emplazamiento. El sistema debe asegurarse de que todas las réplicas de la relación r sea consistentes; en caso contrario, puede dar lugar a resultados erróneos. Por tanto, siempre que se actualice r, la actualización debe extenderse a todos los emplazamientos que contengan réplicas.

- Fragmentación. La relación se divide en varios fragmentos. Cada fragmento se guarda en un emplazamiento o nodo diferente. Si la relación r está fragmentada,

se dividirá en cierto número de fragmentos *r1,r2,...,rn*. Estos fragmentos contendrán suficiente información como para permitir la reconstrucción de la relación original *r*. Dado que una relación se corresponde esencialmente con una tabla y la cuestión consiste en dividirla en fragmentos menores, surgen dos alternativas lógicas para llevar a cabo el proceso: la división horizontal y la división vertical.

La división o fragmentación horizontal trabaja sobre las tuplas, dividiendo la relación en subrelaciones que contienen un subconjunto de las tuplas que alberga la primera. La fragmentación vertical, en cambio, se basa en los atributos de la relación para efectuar la división. Estos dos tipos de partición se consideran los fundamentales y básicas.

- Réplica y fragmentación. La relación se divide en varios fragmentos. El sistema conserva réplicas de cada fragmento. Si se replica la relación *r*, se guardará una copia de la misma en dos o mas emplazamientos. En el caso más extremo, se tendrá una réplica completa, en la que se guarda una copia en cada emplazamiento del sistema.

Desde el punto de vista del usuario, un sistema distribuido deberá ser idéntico a un sistema no distribuido. En otras palabras, los usuarios de un sistema distribuido deberán comportarse exactamente como si el sistema no estuviera distribuido. Todos los problemas de los sistemas distribuidos deberían ser internos o a nivel de realización, no externos o a nivel del usuario. Algunas de las características que se deben tener en cuenta para el diseño de sistemas distribuidos son:

- **Autonomía local**: Los sitios de un sistema distribuido deben ser autónomos. La autonomía local significa que todas las operaciones en un

sitio dado se controlan en ese sitio; ningún sitio X deberá depender de algún otro sitio Y para su buen funcionamiento (pues de otra manera el sitio X podría ser incapaz de trabajar, aunque no tenga en sí problema alguno, si cae el sitio Y). La autonomía local implica también un propietario y una administración local de los datos, con responsabilidad local : todos los datos pertenecen en realidad a una base de datos local, aunque sean accesibles desde algún sitio remoto. Por tanto, las cuestiones de seguridad, integridad y representación en almacenamiento de los datos locales permanecen bajo el control de la instalación local.

- **No dependencia de un sitio central**. La autonomía local implica que todos los sitios deben tratarse igual; no debe haber dependencia de un sitio central maestro para obtener un servicio central, como por ejemplo un procesamiento centralizado de las consultas o una administración centralizada de las transacciones, de modo que todo el sistema dependa de ese sitio central.

  La dependencia de un sitio central sería indeseable al menos por las siguientes razones : en primer lugar, ese sitio central podría ser un cuello de botella; en segundo lugar, el sistema sería vulnerable; si el sitio central sufriera una falla, todo el sistema dejaría de funcionar.

- **Operación continua**. En un sistema distribuido, lo mismo que en uno no distribuido, idealmente nunca debería haber necesidad de apagar a propósito el sistema. Es decir, el sistema nunca debería necesitar apagarse para que se pueda realizar alguna función, como añadirse un nuevo sitio o instalar una versión mejorada del DBMS en un sitio ya existente.

- **Independencia con respecto a la localización**. La idea básica de la independencia con respecto a la localización (también conocida como transparencia de localización) es simple, no debe ser necesario que los usuarios sepan dónde están almacenados físicamente los datos, sino que más bien deben poder comportarse (al menos desde un punto de vista lógico) como si todos los datos estuvieran almacenados en su propio sitio local. La independencia con respecto a la localización es deseable porque simplifica los programas de los usuarios y sus actividades en la terminal.

- **Independencia con respecto a la fragmentación**. Un sistema maneja fragmentación de los datos si es posible dividir una relación en partes o fragmentos para propósitos de almacenamiento físico. La fragmentación es deseable por razones de desempeño: los datos pueden almacenarse en la localidad donde se utilizan con mayor frecuencia, de manera que la mayor parte de las operaciones sean sólo locales y se reduzca al tráfico en la red.

  Existen en esencia dos clases de fragmentación: horizontal y vertical, correspondientes a las operaciones relacionales de restricción y proyección respectivamente. En términos más generales, un fragmento puede ser cualquier subrelación arbitraria que pueda derivarse de la relación original mediante operaciones de restricción y proyección (en el caso de la proyección, las proyecciones deben conservar la clave primaria de la relación original). La reconstrucción de la relación original a partir de los fragmentos se hace mediante operaciones de reunión y unión apropiadas (reunión en el caso de fragmentación vertical, y la unión en casos de fragmentación horizontal). Actualmente existe un número considerable de algoritmos para fragmentación vertical en BDR. Los trabajos presentados

por Hoffer y Severance[23], Hammer y Niamir[24], Navathe[25], entre otros, son muy conocidos por la sociedad de investigadores.

3.4 Diseño de la Base de Datos de Prueba

Para realizar las pruebas del sistema de agentes móviles implementado en este proyecto, se modela parte de la información que se maneja en una universidad, esto es la información sobre estudiantes, facultades y programas, se asume que un estudiante no puede estar matriculado simultáneamente en dos programas, se manejan dos facultades con los siguientes programas adscritos:

- **Facultad de Ciencias e Ingenierías**: Ingeniería Civil, Ingeniería de Sistemas, Ingeniería Mecatrónica, Ingeniería Industrial, Ingeniería Eléctrica, Ingeniería Electrónica e Ingeniería Mecánica.

- **Facultad de Ciencias Económicas**: Administración de Empresas, Contaduría Pública y Economía.

[23] Hoffer, J. y Severeance, D. The Use of Cluster Analysis in Physical Database Design. Proc. First International Conference on Very Large Data Bases. Fragingham, MA, Septiembre 1975.

[24] Hammer, M. y Niammir, B. A Heuristic Approach to Attribute Partitioning. Proc. ACM SIGMOD International Conference on Management of Data. Boston, MA. 1979.

[25] Navathe, S., Ceri, G., Wiederhold y Dou, J. Vertical Partitioning Algorithm for Database Design. ACM Transaction on Database Systems, Vol. 9, No.4, Diciembre 1984.

3.4.1 Modelo Entidad Relación

El diagrama entidad relación es el siguiente:

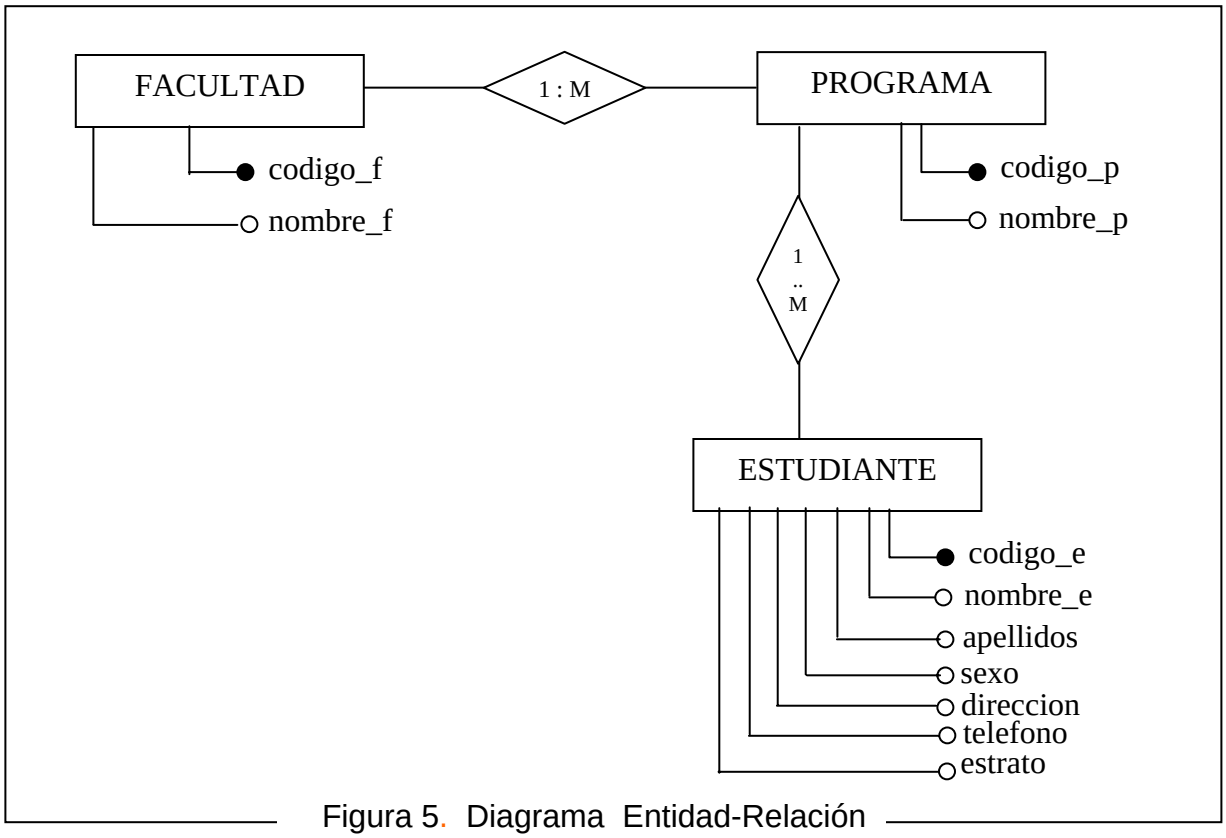


Figura 5. Diagrama Entidad-Relación

3.4.2 Modelo Relacional

FACULTAD {c**odigo**_**f**, nombre_f}
PROGRAMA {**codigo**_**p**, nombre_p, codigo_f}
ESTUDIANTE {**codigo**_**e**, nombre_e, apellidos, sexo, direccion, telefono, estrato, codigo_p}

3.4.3 Fragmentación

El tipo de fragmentación aplicada es la horizontal. La fragmentación se realiza teniendo en cuenta la facultad a la que pertenece cada estudiante, por lo que los datos quedan divididos en dos fragmentos, donde cada fragmento queda almacenado en máquinas diferentes, ya que las facultades están físicamente distantes.

3.4.4 Réplicas

Para lograr un nivel aceptable de tolerancias a fallos se realizan réplicas por cada fragmento, es decir, que se necesitan dos máquinas más para llevarlas a cabo.

3. 5 Sistema Gestor de Base de Datos Seleccionado: MSSQL Server 2000

Se escogió MSSQL Server 2000 para gestionar la base de datos, montado sobre el sistema operativo Windows 2000 Server. SQL Server es un sistema administrador para Bases de Datos relacionales (RDBMS) que usa Transact-SQL[26] para mandar peticiones entre un cliente y el SQL Server. Con Transact-SQL se puede tener acceso a la información, realizar búsquedas, actualizar y administrar sistemas de Bases de Datos Relacionales.

SQL Server 2000 brinda herramientas que permiten manejar de manera precisa la arquitectura de base de datos aquí propuesta, como es la replicación horizontal de datos distribuidos en maquinas diferentes.

3.5.1 Seguridad de SQL Server

SQL Server está integrado con el sistema de seguridad del sistema operativo si así se requiere. Esta integración permite acceder tanto al sistema operativo como a SQL Server con el mismo *user name* y *password*. Los permisos de SQL Server 2000 pueden asignarse directamente a usuarios individuales y a grupos de usuarios del sistema operativo.

En Windows 2000, SQL Server utiliza Kerberos y la delegación de seguridad para admitir tanto la autenticación integrada (con la seguridad del sistema operativo) como los inicios de sesión de SQL Server.

[26] Este es una versión de SQL (Structured Query Languaje) usado como lenguaje de programación para SQL Server

SQL Server valida a los usuarios con 2 niveles de seguridad; autentificación del login y validación de permisos en la Base de Datos de cuentas de usuarios y de roles. La autentificación identifica al usuario que está usando una cuenta y verifica sólo la habilidad de conectarse con SQL Server. El usuario debe tener permiso para acceder a las Bases de Datos en el Servidor. Esto se cumple para asignar permisos específicos para la Base de Datos, para las cuentas de usuario y los roles. Los permisos controlan las actividades que el usuario tiene permitido realizar en la Base de Datos del SQL Server.

3.5.1.1 Autentificación del login

Un usuario debe tener una cuenta para conectarse al SQL Server. Este reconoce 2 mecanismos de autentificación: Autentificación de SQL Server y de Windows 2000 Server. Cada uno tiene un diferente tipo de cuenta.

3.5.1.2 Autentificación de SQL Server

Cuando se usa, un administrador del Sistema de SQL Server, define una cuenta y un password SQL Server. Los usuarios deben suministrar tanto el login como el password cuando se conectan al SQL Server.

3.5.1.3 Autentificación de Windows 2000 Server

Cuando se usa, el usuario no necesita de una cuenta de SQL Server, para conectarse. Un administrador del sistema debe definir, ya sea cuentas de

Windows 2000 Server o grupos de Windows 2000 Server como cuentas válidas de SQL Server.

3.5.1.4 Modo de autentificación

Cuando SQL Server está corriendo en Windows 2000 Server, un sistema administrador puede especificar que está corriendo en uno de 2 modos de autentificación:

- Modo de autentificación de Windows 2000 Server: Sólo está autorizada la autentificación de Windows 2000 Server. Los usuarios no pueden usar cuentas de SQL Server.
- Modo mixto: Cuando se usa este modo de autentificación, los usuarios se pueden conectar a SQL Server con la autentificación de Windows 2000 Server y con la de SQL Server.

3.5.1.5 Cuentas de Usuario y Roles en una Base de Datos

Después de que los usuarios han sido autentificados, y se les ha permitido conectarse al SQL Server, deben tener cuentas en la Base de Datos. Las cuentas de usuario y los roles, identifican permisos para ejecutar tareas.

Las cuentas de usuario utilizadas para aplicar permisos de seguridad son las de usuarios, o grupos de Windows 2000 Server o las de SQL Server. Las cuentas de usuario son específicas para cada Base de Datos.

Los roles permiten reunir a los usuarios en una sola unidad a la cual se le pueden aplicar permisos. SQL Server contiene roles de servidor y de Base de Datos predefinidos, para tareas administrativas comunes, de manera que pueden asignársele determinados permisos administrativos a un usuario en particular. También se pueden crear roles de Base de Datos definidos por el usuario. En SQL Server, los usuarios pueden pertenecer a varios roles:

- Roles fijos del Servidor: Proveen agrupamientos con privilegios administrativos a nivel del Servidor. Son administrados independientemente de las Bases de Datos de usuarios a nivel servidor.
- Roles fijos de la Base de Datos: Proveen agrupamientos con privilegios administrativos a nivel de Base de Datos.
- Roles de usuarios definidos en la Base de Datos: También se pueden crear roles para Base de Datos, para representar un trabajo desarrollado por un grupo de empleados dentro de una organización. No es necesario asignar y quitar permisos a cada persona. En función de que cambia un rol, se pueden cambiar fácilmente los permisos del rol y hacer que los cambios se apliquen automáticamente a todos los miembros del rol.

3.5.1.6 Validación de Permisos:

Dentro de cada Base de Datos, se asignan permisos a las cuentas de usuarios y a los roles para permitir o limitar ciertas acciones. SQL Server acepta comandos después de que un usuario ha accesado a la Base de datos.

SQL Server realiza los siguientes pasos cuando valida permisos:

1. Cuando el usuario realiza una acción, tal como ejecutar un comando de Transact-SQL o elegir una opción de un menú, los comandos de Transact SQL son enviadas al SQL Server.
2. Cuando SQL Server recibe un comando de Transact –SQL, checa que el usuario tenga permiso de ejecutar dicha instrucción.
3. Después, SQL realiza cualquiera de las siguientes acciones:
    a) Si el usuario no tiene los permisos adecuados, SQL Server devuelve un error.
    b) Si el usuario tiene los permisos adecuados, SQL Server realiza la acción.

3.5.2 Servicios de SQL Server

Los servicios de SQL Server incluyen MSSQLServer, SQLServerAgent, Microsoft Distributed Transaction Coordinator (MSDTC), y Microsft Search. Aunque estos servicios de SQL generalmente corren en Windows NT, también pueden correr como aplicaciones.

3.5.2.1 Servicio MSSQLServer

Este servicio es el motor de la Base de Datos. Este es el componente que procesa todas las declaraciones de Transact-SQL y administra todos los archivos que definen a la Base de Datos dentro del Servidor. Sus características son:

- Asignar los recursos de la computadora a múltiples usuarios simultáneos.

- Previene problemas lógicos, tales como sincronización de peticiones de usuarios que desean actualizar la misma información al mismo tiempo.
- Garantiza la integridad y consistencia de datos.

3.5.2.2 Servicio SQLServerAgent

Es un servicio que trabaja en conjunto con SQL Server para desempeñar las siguientes tareas administrativas:

- Administración de Alertas: Las alertas brindan información acerca del estado de un proceso, tal como cuando un trabajo está completo o cuando ocurre un error. El agente de SQL Server monitorea la aplicación de Windows 2000 Server y genera alertas.

- Notificación: El agente de SQL Server puede enviar e-mail, o iniciar otra aplicación cuando ocurre una alerta, por ejemplo, se puede programar una alerta para que ocurra cuando una Base de Datos o cuando una transacción está casi completa o cuando un respaldo de la Base de Datos ha terminado exitosamente.

- Ejecución de Tareas: El agente de SQL Server incluye un motor de creación y planeación de tareas. Las tareas pueden ser simples operaciones de un solo paso, o pueden ser tareas complejas de varios pasos que requieren planeación. También se pueden crear pasos de las tareas con Transact-SQL, leguajes script, o comandos del Sistema Operativo.

- Administración de Réplicas o duplicación: La replicación es el proceso de copiar datos o transacciones de un SQL Server a otro. El agente de SQL Server es responsable de sincronizar los datos entre los servidores, monitorear los datos para buscar cambios y replicar la información en otros servidores.

3.5.2.3 Servicio Microsoft Distributed Transaction Coordinator

MSDTC permite a los clientes incluir muchos tipos de datos en una transacción. Coordina la correcta realización de las transacciones distribuidas para asegurar que todas las actualizaciones en todos los servidores sean permanentes; o en caso de errores, que las modificaciones sean canceladas.

3.5.3 Arquitectura de duplicación o Replica

La duplicación es una tecnología muy importante para la distribución de datos y procedimientos almacenados en una organización. La tecnología de duplicación de Microsoft SQL Server permite hacer copias duplicadas de los datos, mover dichas copias a ubicaciones diferentes y sincronizar los datos automáticamente de forma que todas las copias tengan los mismos valores en los datos. La duplicación entre bases de datos se puede implementar en el mismo servidor o en servidores diferentes conectados a través de LAN, WAN o Internet.

El modelo de duplicación utilizado por SQL Server está basado en el concepto de publicar y suscribir, SQL Server acepta:

- Duplicación en orígenes de datos heterogéneos.
- Interfaces de programación para la duplicación de datos desde orígenes de datos heterogéneos.
- Interfaces de programación para invocar los agentes de duplicación.
- Interfaces SQL-DMO para instalar y supervisar la duplicación.

3.5.3.1 Dimensiones de la duplicación

La duplicación de datos es una tecnología compleja. SQL Server ofrece una variedad de tecnologías de duplicación que se pueden personalizar para adaptarlas a los requisitos específicos de las aplicaciones. Cada tecnología proporciona beneficios y restricciones diferentes, según los requisitos, en tres áreas importantes:

- Coherencia transaccional
- Autonomía de los sitios
- Partición de datos para evitar conflictos

La coherencia transaccional respecto a las operaciones distribuidas, como la duplicación, agrega matices adicionales a las cuestiones normales de las bases de datos en cuanto a las propiedades ACID (atomicidad, coherencia, aislamiento y durabilidad) y los niveles de aislamiento de las transacciones. En el contexto de la duplicación, la coherencia transaccional significa que, tras la duplicación, los datos almacenados en los sitios participantes serán los mismos que se habrían obtenido si se hubieran realizado todas las transacciones en un solo sitio.

La autonomía de los sitios hace referencia al efecto que las operaciones realizadas en un sitio tienen sobre otro sitio. Existe una autonomía completa si la capacidad de un sitio para realizar su trabajo habitual es independiente de su

conexión con otro sitio y del estado de las operaciones de dicho sitio. Por ejemplo, el protocolo de confirmación en dos fases (2PC, *Two-Phase Commit*) hace que todos los cambios que dependen de todos los demás sitios participantes puedan aceptar inmediata y correctamente la transacción. Si un sitio no está disponible, no continúa la operación. En el otro extremo se encuentran los sitios de duplicación de mezcla, que trabajan independientemente y pueden estar totalmente desconectados de todos los demás sitios.

Pueden dividirse los datos entre varios sitios para proporcionar su propia garantía de la coherencia transaccional. La falta de coherencia transaccional garantizada no implica necesariamente la incoherencia transaccional. Si se diseña la aplicación de manera que cada sitio participante trabaje con datos divididos de manera estricta, o divididos a partir de otros sitios, puede mantenerse la coherencia transaccional. Por ejemplo, En el diseño de la base de datos de este proyecto cada facultad tiene sus datos independientes de la otra y un estudiante no puede pertenecer simultáneamente a mas de una facultad, de manera que no hay conflicto entre las diferentes facultades.

3.5.3.2 Tipos de duplicación

Microsoft SQL Server proporciona tres tipos principales de duplicación y dos opciones que se pueden utilizar al diseñar aplicaciones distribuidas:

- Instantánea
- Transaccional
    - Opción Suscriptores de actualización inmediata
    - Opción Duplicación de la ejecución de procedimientos almacenados
- Combinación

Cada uno de estos tipos proporciona funciones y atributos diferentes en cuanto a la coherencia transaccional y se pueden organizar para conseguir garantizar una coherencia inmediata continuada a partir del distinto origen de los datos convergentes.

La duplicación de instantáneas toma una instantánea de los datos actuales de una publicación en el publicador y reemplaza la copia completa en el suscriptor de forma periódica, en lugar de publicar los cambios cuando éstos se producen.

La duplicación transaccional marca para su duplicación las transacciones seleccionadas en el registro de transacciones de la base de datos del publicador y, después, las distribuye de forma asíncrona a los Suscriptores como cambios incrementales, al mismo tiempo que mantiene la coherencia transaccional.

La duplicación de mezcla permite que los sitios realicen cambios autónomos en los datos duplicados y, posteriormente, mezclen los cambios realizados en todos los sitios. La duplicación de mezcla no garantiza la coherencia de transacciones.

3.5.3.3 Componentes de la duplicación

El modelo de duplicación de Microsoft SQL Server 2000 está basado en el concepto de publicar y suscribir, este modelo se compone de:

- Publicadores
- Distribuidores
- Suscriptores
- Publicaciones
- Artículos
- Suscripciones de inserción

- Suscripciones de extracción

El publicador es un servidor que hace que los datos estén disponibles para su duplicación en otros servidores. Además de identificar los datos que se van a duplicar, el publicador detecta qué datos han cambiado y mantiene información acerca de todas las publicaciones del sitio. Cualquier elemento de datos que se duplique tiene un solo publicador, incluso si puede ser actualizado por un número cualquiera de suscriptores o publicado de nuevo por un suscriptor.

El distribuidor es el servidor que contiene la base de datos de distribución. La función concreta del distribuidor es diferente en cada tipo de duplicación de SQL Server .

Los suscriptores son servidores que almacenan duplicados y reciben actualizaciones. SQL Server permite que los suscriptores actualicen los datos (pero un suscriptor que hace actualizaciones no es lo mismo que un publicador). Un suscriptor puede, a su vez, ser un publicador para otros suscriptores.

Una publicación es una colección de artículos y un artículo es un grupo de datos que se duplican. Un artículo puede ser una tabla entera, sólo algunas columnas (con un filtrado vertical), sólo algunas filas (con un filtrado horizontal) o incluso un procedimiento almacenado (en algunos tipos de duplicación). Una publicación suele tener varios artículos. Esta agrupación de artículos facilita la suscripción a una unidad (la publicación), que contiene todos los datos relevantes y necesarios. Los suscriptores se suscriben sólo a la publicación, no a los artículos individuales de la misma.

Con una suscripción de inserción, el publicador propaga los cambios al suscriptor sin que éste lo solicite. Normalmente, las suscripciones de inserción se utilizan en aplicaciones que se emplean para enviar cambios a los suscriptores en cuanto se

producen. Las suscripciones de inserción son adecuadas para las publicaciones que requieran un movimiento de datos casi en tiempo real sin tener que sondear y donde la mayor carga de trabajo del procesador del Publicador no afecte a su rendimiento. También se pueden insertar los cambios en los suscriptores de forma programada.

En una suscripción de extracción, el suscriptor solicita actualizaciones periódicas de todos los cambios del publicador. Las suscripciones de extracción son adecuadas para las publicaciones que tengan un gran número de suscriptores (por ejemplo, suscriptores que utilicen Internet). Las suscripciones de extracción también son adecuadas para los usuarios móviles independientes, ya que permiten que el usuario determine cuándo va a sincronizar los cambios de los datos. Una misma publicación puede admitir una combinación de suscripciones de inserción y de extracción.

Los componentes de duplicación anteriores se implementan con un diseño modular. Se puede instalar estos componentes en equipos diferentes para repartir las cargas de trabajo y minimizar el efecto de la duplicación de SQL Server en el rendimiento de los servidores.

3.5.3.4 Agentes de duplicación

Además de los componentes básicos, el diseño de una duplicación puede tener dos o más agentes de duplicación:

- Agente de instantáneas
- Agente de lector del registro
- Agente de distribución

- Agente de mezcla

El agente de instantáneas prepara el esquema y los archivos de datos iniciales de las tablas y los procedimientos almacenados publicados, almacena la instantánea en el distribuidor y registra la información acerca del estado de la sincronización en la base de datos de distribución. Cada publicación tiene su propio agente de instantáneas, que se ejecuta en el Distribuidor y se conecta con el Publicador. El agente de instantáneas se ejecuta normalmente en el Agente SQL Server y se puede administrar directamente mediante el Administrador corporativo de SQL Server.

El agente de lector del registro pasa las transacciones marcadas para duplicación desde el registro de transacciones del publicador a la base de datos de distribución. Cada base de datos publicada que utilice una duplicación transaccional tiene su propio agente de lector del registro, que se ejecuta en el Distribuidor y se conecta con el Publicador.

El agente de distribución pasa las transacciones y los trabajos de instantáneas almacenados en tablas de la base de datos de distribución a los suscriptores. Las publicaciones transaccionales y de instantáneas que estén configuradas para la sincronización inmediata cuando se cree una nueva suscripción de inserción tienen su propio agente de distribución que se ejecuta en el Distribuidor y se conecta con el Suscriptor. Las publicaciones transaccionales y de instantáneas que no estén configuradas para la sincronización inmediata compartirán un agente de distribución con el Publicador y el Suscriptor que se ejecute en el Distribuidor y se conecte con el Suscriptor. Las suscripciones de extracción a publicaciones de instantáneas o transaccionales tienen agentes de distribución que se ejecutan en el Suscriptor en lugar de en el Distribuidor. Las publicaciones de mezcla no tienen agente de distribución. El agente de distribución se ejecuta normalmente en el

Agente SQL Server y se puede administrar directamente mediante el Administrador corporativo de SQL Server.

El agente de mezcla mueve y concilia los cambios incrementales de los datos que hayan ocurrido después de la creación de la instantánea inicial. Cada publicación de mezcla tiene su propio agente de mezcla, que se conecta con el Publicador y con el Suscriptor, y actualiza a los dos. En una mezcla completa, el agente envía primero desde el suscriptor todos los cambios cuya generación sea 0 ó mayor que la última generación enviada al publicador. El agente recopila las filas y aquéllas que no entren en conflicto se aplican a la base de datos de publicación. Las filas que estén en conflicto se controlan mediante la resolución de conflictos asociada con el artículo en la definición de la publicación. Todos los cambios se aplican con procedimientos almacenados derivados desde las tablas del Publicador cuando se generó la instantánea o se aplicó por primera vez. Por último, el agente invierte el proceso al descargar los cambios del publicador en el suscriptor y al aplicar dichos cambios a la base de datos del suscriptor. Las suscripciones de inserción a publicaciones de mezcla tienen agentes de mezcla que se ejecutan en el Publicador, mientras que las suscripciones de extracción a publicaciones de mezcla tienen agentes de mezcla que se ejecutan en el Suscriptor. Las publicaciones de instantáneas y las transaccionales no tienen agentes de mezcla.

3.5.3.5 Administrar la seguridad de la duplicación

La seguridad de la duplicación es una parte importante del diseño y la implementación de las aplicaciones distribuidas. La duplicación se aplica a los cambios de datos realizados en cualquier parte de la red a la base de datos del servidor, y viceversa.

La disponibilidad descentralizada de los datos duplicados aumenta la complejidad de la administración y de la restricción del acceso a esos datos. La duplicación de Microsoft SQL Server utiliza una combinación de mecanismos de seguridad para proteger los datos y la lógica de negocio de las aplicaciones:

- Requisitos de las funciones

  Al asignar inicios de sesión de usuario a funciones específicas de SQL Server, éste permite a los usuarios realizar sólo aquellas actividades de bases de datos y de duplicación que están autorizadas para esa función. La duplicación concede ciertos permisos a la función fija de servidor **sysadmin**, a la función fija de base de datos **db_owner**, al inicio de sesión actual y a la función **public**. Por ejemplo, sólo los miembros de la función de servidor **sysadmin** pueden configurar la duplicación.

- Seguridad de vínculo administrativo del distribuidor

  SQL Server proporciona un vínculo administrativo seguro entre el distribuidor y un publicador remoto. Los publicadores pueden tratarse como publicadores en los que se confía o en los que no se confía.

- Seguridad de la carpeta de instantáneas

  El sistema operativo o servicio FTP impide que los usuarios tengan acceso a determinados archivos del servidor. El usuario debe tener un inicio de sesión válido para leer o escribir los archivos utilizados en el proceso de duplicación.

- Suscriptores registrados

SQL Server permite limitar el acceso a las publicaciones a los suscriptores registrados que son bien conocidos para el publicador, a los Anónimos o a los suscriptores que tengan inicio de sesión en las listas de acceso de la publicación. Para asegurar la duplicación de los datos con los orígenes de datos heterogéneos, SQL Server utiliza definiciones de servidor vinculado para los suscriptores heterogéneos.

- Listas de acceso a la publicación

  Al admitir las listas de acceso a la publicación (PAL, *Publication Access List*) en cada servidor, SQL Server permite determinar qué inicios de sesión tienen acceso a las publicaciones. SQL Server crea la lista PAL con los inicios de sesión predeterminados, aunque se pueden agregar o eliminar inicios de sesión de la lista.

- Seguridad de inicio de sesión del agente

  Con la seguridad de inicio de sesión de los agentes, SQL Server requiere que cada usuario suministre una cuenta válida de inicio de sesión para conectarse al servidor. Los agentes de duplicación tienen que utilizar inicios de sesión válidos cuando se conectan a los publicadores, distribuidores y suscriptores. Pero los agentes pueden utilizar también distintos inicios de sesión y modos de seguridad cuando se conectan a distintos servidores simultáneamente.

- Seguridad de los suscriptores de actualización inmediata

  Para los suscriptores de actualización inmediata, la duplicación de SQL Server aplica mecanismos de seguridad a los procedimientos almacenados del publicador y al vínculo RPC del publicador.

3.5.3.6 Configurar la duplicación

La configuración de un sistema de duplicación resulta más fácil si se utilizan asistentes para la duplicación, disponibles en el menú **Herramientas** del Administrador corporativo de SQL Server. Se pueden usar estos asistentes para habilitar, modificar y deshabilitar servidores configurados como distribuidores, publicadores y suscriptores.

Se puede usar el Asistente para creación de publicaciones o el Asistente para configurar Publicar y Distribuir. A menos que se tengan necesidades específicas (por ejemplo, crear un distribuidor remoto para poder trabajar con otros publicadores), es más fácil utilizar el Asistente para creación de publicaciones.

Asimismo, se puede implementar la duplicación utilizando procedimientos almacenados del sistema o SQL-DMO.

3.5.3.7 Publicar datos y procedimientos almacenados

Los datos se publican cuando se crea una publicación, se seleccionan las tablas o los procedimientos almacenados que van a estar incluidos en la publicación, y se hace que la publicación esté disponible para los suscriptores. A una tabla incluida en una publicación se le denomina artículo. Un artículo puede ser la tabla entera o un subconjunto de filas o de columnas.

Importante Una tabla utilizada en una publicación de instantáneas o transaccional puede tener un máximo de 255 columnas y un tamaño máximo de fila de 8000

bytes. Una tabla utilizada en una publicación de mezcla puede tener un máximo de 246 columnas y un tamaño máximo de fila de 6000 bytes.

También se puede seleccionar que la ejecución de un procedimiento almacenado se publique como artículo en una publicación transaccional. No se puede publicar un procedimiento almacenado como artículo en una publicación de mezcla

3.5.3.8 Suscribirse a publicaciones

Suscribirse significa acordar la recepción de una copia de una publicación o de un artículo. Una base de datos destino de un suscriptor se suscribe a los datos duplicados de una base de datos de publicación de un publicador.

Al configurar una *suscripción*, ésta puede ser una suscripción de inserción o una suscripción de extracción. El tipo de suscripción que se elije quedará determinado por la configuración de la publicación y por quién realiza el proceso de suscripción.

Una *suscripción de inserción* se efectúa mientras un publicador está siendo administrado (administración centralizada). Como parte del proceso de creación o administración de una publicación, las suscripciones se crean al insertar una copia de una publicación en uno o varios suscriptores.

Una *suscripción de extracción* se efectúa mientras un suscriptor está siendo administrado (administración descentralizada). La suscripción se crea al extraer una copia de una publicación de un publicador. También puede crear un tipo especial de suscripción de extracción: suscripciones anónimas. Las suscripciones anónimas no quedan registradas en el publicador y se pueden utilizar en aplicaciones para Internet. Las suscripciones de extracción o las anónimas tienen que estar habilitadas en el publicador, pero las crea el suscriptor.

Después de configurar una suscripción, las tablas de publicación y de destino tienen que ser *sincronizadas* inicialmente. La sincronización hace que el esquema y los datos de la tabla de la base de datos destino sean idénticos al esquema y los datos de la tabla de la base de datos de publicación. En la base de datos de destino también puede haber otras tablas. El esquema de toda la base de datos de destino no tiene que ser idéntico al de la base de datos de publicación.

Cuando se configura una suscripción, Microsoft SQL Server efectúa automáticamente y de forma predeterminada la sincronización inicial al crear la suscripción. Sin embargo, también se puede especificar que la sincronización tenga lugar a horas programadas o que no haya sincronización

3.5.3.9 Supervisar la duplicación

El Monitor de duplicación es un componente del Administrador corporativo de SQL Server diseñado para ver el estado de los agentes de duplicación y solucionar los posibles problemas de un distribuidor. El Monitor de duplicación sólo se activa como componente cuando el servidor está habilitado como distribuidor y el usuario actual es miembro de la función del servidor **sysadmin**. Todos los agentes de duplicación tienen que estar programados mediante el Agente de SQL Server.

El Monitor de duplicación del Administrador corporativo de SQL Server puede ser utilizado para:

- Ver la lista de los publicadores, las publicaciones y las suscripciones a las publicaciones admitidas por el distribuidor.
- Ver los agentes de duplicación programados, y supervisar en tiempo real el estado más actual y el historial de cada agente.

- Configurar alertas en el monitor, relacionadas con sucesos de la duplicación.

  Si el suceso tiene lugar, Microsoft SQL Server responde automáticamente; para ello, ejecuta la tarea que se haya definido o envía un mensaje de correo electrónico o un mensaje a un localizador (*pager* o "busca"*)* al operador especificado.

Después de haber configurado la duplicación, también se puede utilizar el Visor de sucesos de Microsoft Windows 2000 Server para ver los mensajes de SQL Server.

3.6 Puesta en Marcha de la Base de Datos

La Universidad modelada en este proyecto consta de tres sedes; una sede administrativa, en donde se encuentra un servidor llamado AURICA desde donde se realiza el proceso de matricula de estudiantes, una vez hecho el proceso de matricula, los datos del estudiante son distribuidos a una de las otras dos sedes, dependiendo del programa al cual se haya matriculado así: si quedó adscrito a uno de los programas que ofrece la Facultad de Ciencias e Ingeniería (identificada con el código 01), los datos son enviados a un servidor llamado NERO y, si el estudiante quedó adscrito a uno de los programas que ofrece la Facultad Ciencias Económicas, sus datos son enviados a un servidor llamado ZEUS (Ver siguiente figura).

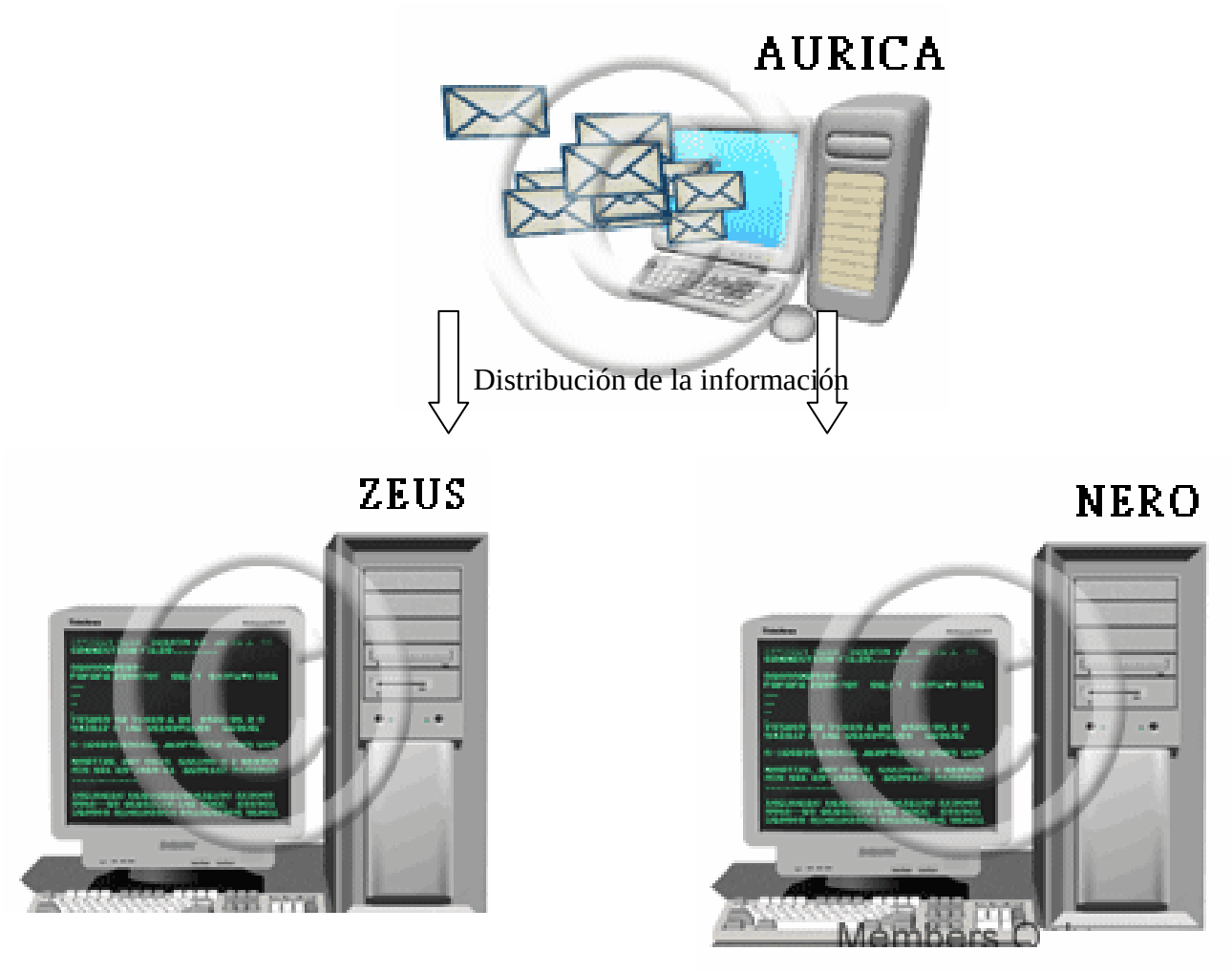


Figura 6. Modelo de la Base de Datos de Prueba

Los tres servidores fueron configurados con el sistema operativo MS Windows 2000 Server y se les instaló la versión Estándar del MSSQL 2000 (ver Anexo A para cualquier información sobre la instalación de este sistema). Las tres sedes están conectadas por radioenlace y en la Intranet de la universidad, los tres servidores son reconocidos tanto por su dirección IP como por su nombre.

En el proceso de la configuración de los servidores para la distribución y duplicación de datos, se deben realizar algunas tareas, como son:

1. Hacer que los equipos se vean, es decir, que haya comunicación entre AURICA, ZEUS y NERO, esto se logró colgando los servidores a la red de la Universidad. Para la resolución de nombres actualizamos en cada equipo el archivo *host* que por lo general se encuentra en la ruta c:\winnt\system32\drivers\etc, en donde asignamos a cada dirección IP el nombre correspondiente de servidor.
2. Linkear los servidores (LinkedServer), con el fin de poder conectarme desde el Enterprise de SQL de un servidor a otro. Este proceso se realiza en la pestaña security del explorador del Enterprise en cada servidor.

   Para probar el paso 2 se realizó una consulta desde NERO a la tabla Estudiante y efectivamente se obtuvo buena respuesta. Para conectarse con otro servidor, luego de linkearlo, desde el Query Analyzer y con el respectivo password se puede realizar dicha conexión.
3. Crear la base de datos en cada servidor. La base de datos Unitecno se creó en AURICA. En ZEUS y NERO también se crea dicha base de datos (no es necesario crear la estructura), sin importar que se asigne o no el mismo nombre

Luego de los pasos anteriores, se tienen los servidores listos para la distribución y duplicación de los datos.

Se realizaron dos publicaciones de tipo transaccional (transactional publication) de Unitecno, ya que de los tres tipos de publicaciones disponibles en SQL 2000, ésta

es la que garantiza una consistencia transaccional, lo cual es muy necesario a la hora en que los directivos de la Universidad tomen las mejores decisiones con información veraz y consistente en las tres sucursales.

Las publicaciones se realizan en AURICA, el cual es el Distribuido., los Suscriptores son NERO y ZEUS, ellos se suscriben a la publicación respectiva según la información publicada. Los artículos para ambas publicaciones son: ESTUDIANTE, FACULTAD Y PROGRAMA.

Para realizar la replica horizontal, tal y como se mencionó anteriormente, se tuvo en cuenta el código de la Facultad para poder distribuir y replicar la información a cada servidor respectivo.

3.6.1 Ventajas del Modelo y Arquitectura de Base de datos planteado

Con el modelo planteado en este proyecto, en caso de que la Universidad decida expandirse o crear nuevas sucursales en donde ofrezca los mismos programas, es muy sencillo que dichas sedes cuenten con distribución duplicada de los datos de los estudiantes, basta con montar un servidor por cada nueva sede y suscribir dichos servidores a la publicación respectiva según programas ofrecido.

También al momento de crear nuevas facultades, el modelo ofrece la portabilidad y escalabilidad para ello, ya que solo sería necesario crear una nueva publicación desde AURICA con la respectiva información de Facultad.

Mediante el Monitor de Replicas que se adiciona en el Distribuidor AURICA, también se garantiza una buena administración de las transacciones a ser distribuidas y duplicadas, esto ayudado de los diferentes Agentes que ofrece MSSQL para garantizar la consistencia en cada transacción.

4. AGENTES MOVILES: SEGURIDAD

Los agentes móviles son programas que viajan con su código y datos desde un computador a otro a través de Internet o cualquier red, lo que conlleva a que surjan varios problemas de seguridad porque tanto los agentes móviles como los servidores con que interactúan son vulnerables a ataques y brechas de seguridad. Algunos problemas de seguridad que se pueden presentar son: Autenticación, Autorización y Confidencialidad.

La movilidad de los agentes y la interacción con otras máquinas de la red, puede hacer que surjan algunos requerimientos de seguridad en el acceso a base de datos distribuidas que podrían ser clasificados de dos tipos[27]. El primero de ellos se refiere a los problemas de seguridad inherentes a los sistemas de agentes móviles y en el segundo se refiere a los aspectos que se requieren para garantizar la seguridad en el ámbito de la Base de Datos Distribuida. A continuación se expone con mayor detalle cada uno de estos problemas.

[27] MAN, Mo y WEI V. A Taxonomy for Attacks on Mobile Agent. IEEEXplore. EUROCON'2001 Trends in Communications, International Conference. Volumen 2. p. 385-388. 2001.

4.1 Amenazas de Seguridad Inherentes al Sistema de Agentes Móviles

La seguridad ha sido tradicionalmente un inconveniente en los sistemas informáticos, pero el sistema de agentes móviles por sus características presenta un nuevo tipo de inconvenientes de seguridad. Los dos problemas principales son: El problema del agente malicioso y el problema del servidor malicioso.

4.1.1 El Problema del Agente Malicioso.

Cuando los agentes móviles llegan al servidor, necesitan algunos permisos de usuario para poder ejecutar su código. Normalmente el control de acceso a los equipos de una red se hace por medio de contraseñas, en donde cada usuario se autentica al inicio con su *login* y *password*; pero este esquema funciona para sistemas estáticos y no para agentes móviles, puesto que el agente debe llevar una contraseña para cada sitio de su itinerario, lo que aumenta su tamaño y así mismo incrementa los requerimientos de seguridad para poder proteger esas contraseñas.

Entre los agentes pueden existir agentes maliciosos que realizan acciones no deseadas o destructivas, como el acceso no autorizado, alteración de la información, sobrecargar o causar incluso daños a la integridad del servidor. En la figura 7 se presenta un agente malicioso que llega al servidor para realizar labores que atentan contra la integridad del servidor, como es la utilización no autorizada de los recursos para obtener información privada.

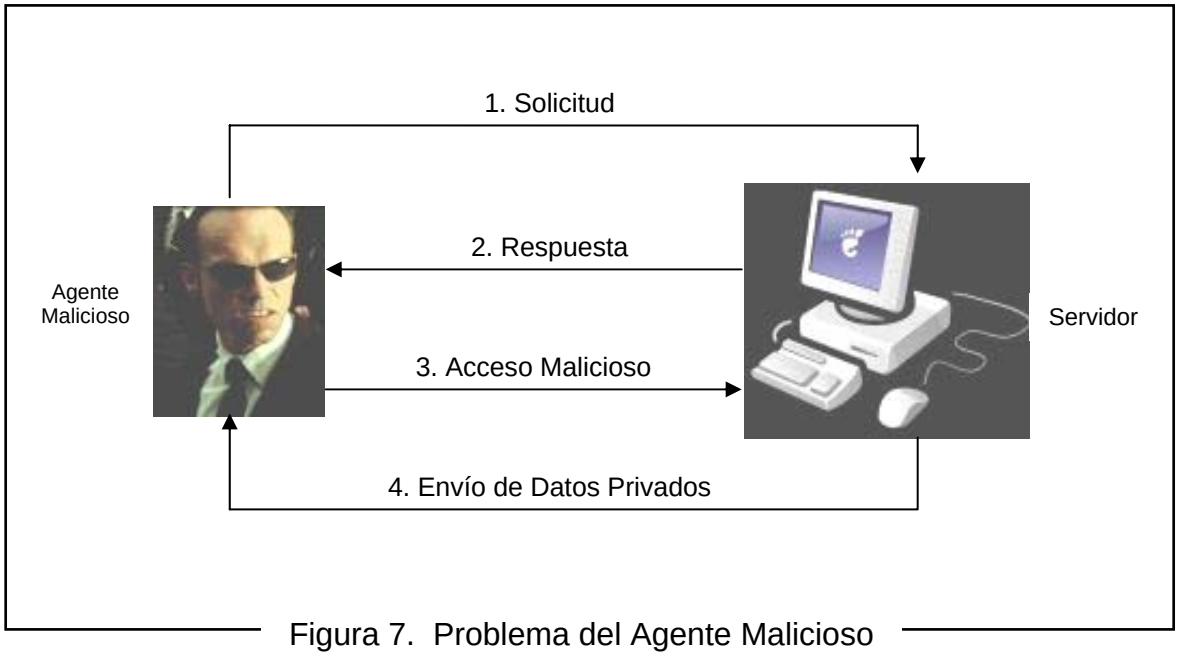


Figura 7. Problema del Agente Malicioso

El problema del agente malicioso puede generar tres tipos de amenazas a los servidores:

- **El acceso no autorizado a los recursos.** Cuando un agente móvil llega a un servidor, ejecuta su código usando los recursos de este. Por esta razón, los recursos del servidor están expuestos a ataques de agentes maliciosos, por lo tanto si no existen mecanismos de control adecuados, agentes no deseados podrían acceder los datos privados del sistema. En el caso de las Bases de Datos Distribuidas, si un agente malicioso accede a los datos puede modificarlos y causar inconsistencias en la base de datos.

- **Sobrecarga de los recursos del servidor.** Se deben plantear mecanismos de protección adecuados para prevenir el consumo excesivo de los recursos del servidor por parte de agentes no deseados, debido que al sobrecargarlo el

tiempo de respuesta se incrementa o inclusive hacer que no haya respuesta a los agentes que buscan acceder a la base de datos.

- **Creación de agentes residentes en el Servidor.** Un ataque del cual puede ser víctima un servidor por parte de un agente malicioso, es la creación de un agente que resida en el servidor. Con un agente de este tipo podría suceder cualquiera de los dos ataques mencionados anteriormente, ó que se clonen tantos agentes que podrían también sobrecargar al servidor hasta bloquearlo.

La solución que se le da normalmente al problema del agente malicioso se lleva a cabo por medio de restricción de privilegios a los agentes, pero esto podría ocasionar que los usuarios que traten de acceder a la base de datos mediante un agente no pueda hacerlo, por tal razón es necesario buscar mecanismos alternos de protección que brinden seguridad y permitan la realización de las tareas de los agentes.

4.1.2 El problema del Servidor Malicioso

Un servidor malicioso es aquel que podría espiar ó alterar el código y/o los datos de un agente, proporcionar llamadas falsas al sistema, retornar al agente datos no deseados, entre otros. En la figura 8 se ilustra un ataque de un servidor malicioso a un agente móvil.

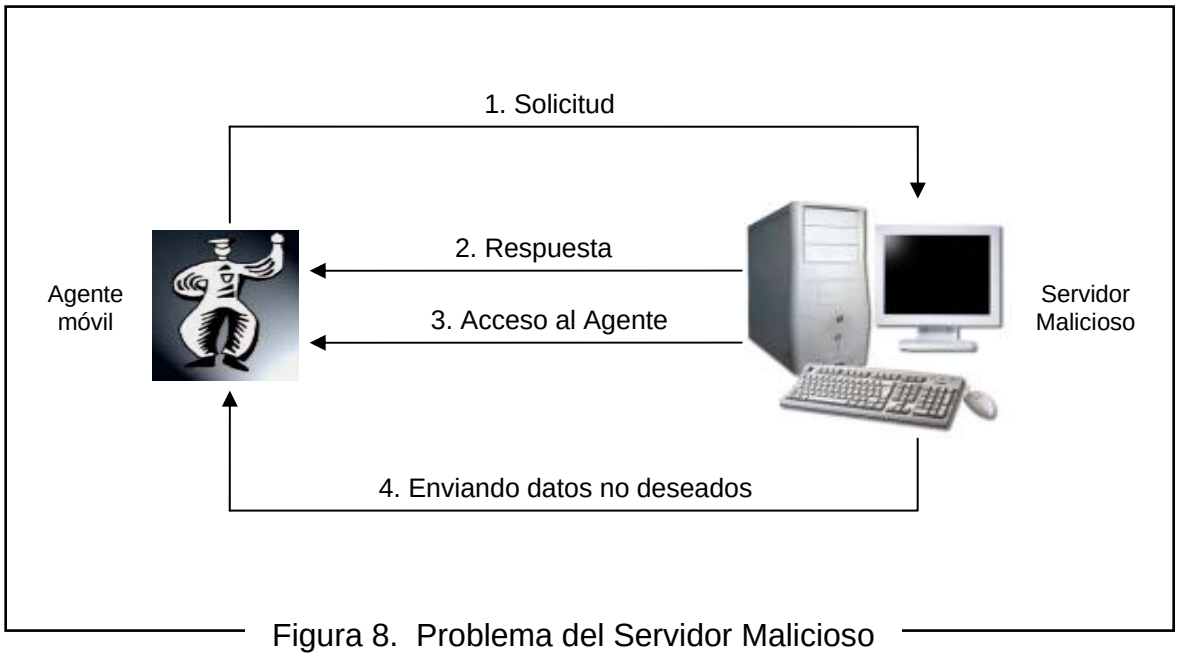


Figura 8. Problema del Servidor Malicioso

Cuando el agente móvil llega al servidor, ya no depende de la máquina que lo envió, por lo tanto, el servidor receptor debe instanciar la clase que dio origen al agente, leer el estado de datos que se encuentra serializado y ejecutarlo, por lo que el código y datos del agente móvil quedan totalmente expuestos.

Los ataques de servidores contra agentes móviles se podrían clasificar de diversos tipos:

- **Clonación del Agente.** Uno de los ataques al agente móvil consiste en que un servidor lo captura y construye una réplica de él, de esa manera, se hace pasar por el agente original para los propósitos que ese programa residente en el servidor. En el caso de acceso a bases de datos se podría clonar un agente para robar información o alterar los datos.

- **Negación de servicio.** Los ataques de negación del servicio se pueden dar cuando el servidor impide que el agente continúe su itinerario o permitir al agente continuar, pero no al sitio que tiene determinado en el itinerario, sino a aquel que el servidor elija. En el caso de acceso a bases de datos un servidor podría hacer esperar al agente indefinidamente o desviarlo a una máquina no deseada.

- **El ataque del hombre en el medio.** Un agente móvil puede ser alterado e interceptado mientras se desplaza de un máquina a otra. Una manera de hacerlo es mediante el uso de un programa capaz de monitorear los agentes móviles pudiendo observar los mensajes intercambiados por ellos para extraer o modificar los datos. Con un ataque de este tipo se podrían capturar datos como passwords o información confidencial.

- **Modificación de los datos que lleva el agente.** Cuando un agente móvil se ejecuta en un servidor, este puede observar el contenido del agente, por tal motivo, si no se establecen los mecanismos de protección adecuados, un servidor malicioso puede alterar los datos que trae consigo el agente.

4.2 Problemas del acceso a bases de datos distribuidas con agentes móviles

Con el desarrollo que ha tenido Internet y las comunicaciones se ha incrementado exponencialmente el uso de bases datos distribuidas apareciendo los agentes móviles como una alternativa para mejorar el desempeño en este tipo de aplicaciones distribuidas. El modelo de agentes móviles va teniendo un mayor

auge cada vez, tanto que existen en el mercado herramientas para crear agentes móviles, se cuenta con estándares, plataformas y productos que facilitan su desarrollo (como se expuso en el capitulo dos), pero debido a problemas de seguridad el acceso a bases de datos distribuidas mediante agentes móviles no ha tenido el auge esperado. Por tanto se hace necesario buscar soluciones para lograr un nivel de seguridad adecuado para proteger el acceso a la bases de datos y la forma de alcanzarlo es logrando mejorar aspectos de seguridad como son[28]: la autenticación, autorización, confidencialidad e integridad.

4.2.1 Autenticación.

Mediante la autenticación un usuario asegura decir que es quién dice ser. Para lograr esta autenticación en el acceso a la base de datos distribuida el propio motor de la base de datos lo autentica por lo general con un login y password, que en algunos casos podrían ser utilizados métodos como: la Identificación Biométrica, Documentos de Identificación y Firmas Digitales. El problema radica que login y passwords podrían viajar con los agentes y ser interceptados por agentes o programas maliciosos para violar el acceso a la base de datos.

4.2.2 Autorización.

Con la autorización se asegura que el usuario que va a acceder a la información posea los permisos de seguridad o privilegios necesarios, para que la información en la red permaneciera privada. En una base de datos después que se autentica el usuario, este usuario posee unos permisos asignados por el administrador de la

[28] BORSELIUS, Niklas. Mobile agent security. Electronics & Communication Engineering Journal. Volumen 14. p. 211-218. 2002.

base de datos de acuerdo a los derechos o necesidades del usuario. Por lo tanto este problema ya no dependería de los agentes móviles sino del administrador de la base de datos que debe asignar los privilegios adecuados a los usuarios, tal como se especifica en este documento en el capitulo 3, aparte 3.5.1.

4.2.3 Confidencialidad

La confidencialidad asegura que ningún usuario sin los privilegios adecuados tenga acceso a la información. De igual manera que en la autorización se depende mucho del administrador de la base de datos, pero cuando los datos viajen con el agente se podría considerar un mecanismo de encriptación para proteger la confidencialidad de la información que viaja en la red.

4.2.4 Integridad

La integridad asegura que los datos sean consistentes y no hayan sido alterados. Los motores de bases de datos tienen sus propios mecanismos para mantener la integridad de los datos, labor que depende del administrador de la base de datos. Por lo tanto el sistema de agentes móviles también debe proporcionar mecanismos criptográficos para que los datos que transportan en la red no sean alterados, dichos mecanismos no se manipulan en este proyecto, ya que el enfoque de seguridad aquí planteado no alberga esa área, este proyecto está en un nivel superior, es decir, el sistema de agentes trabaja con los mecanismos de encriptación que traen por defecto.

5. MODELO PROPUESTO PARA EL ACCESO A BASES DE DATOS DISTRIBUIDAS UTILIZANDO AGENTES MOVILES

Después de haberse analizado los aspectos concernientes a los agentes móviles y los problemas de seguridad que estos conllevan se presentará en este capitulo un modelo de seguridad para el sistema de agentes móviles que acceden a una base de datos distribuida, además de brindar un proceso que colabore en la autenticación de los usuarios que quieren acceder al servidor de base de datos mediante este sistema.

5.1 Modelo propuesto

El modelo propuesto se denomina **ABADAM** (Acceso a Bases de Datos utilizando Agentes Móviles), ABADAM es un modelo implementado utilizando tres grandes componentes: el lenguaje de programación JAVA, el API AGLETS y SQL. El sistema propuesto está conformado por un sistema de agentes móviles, un modelo de seguridad y la base de datos (Figura 9).

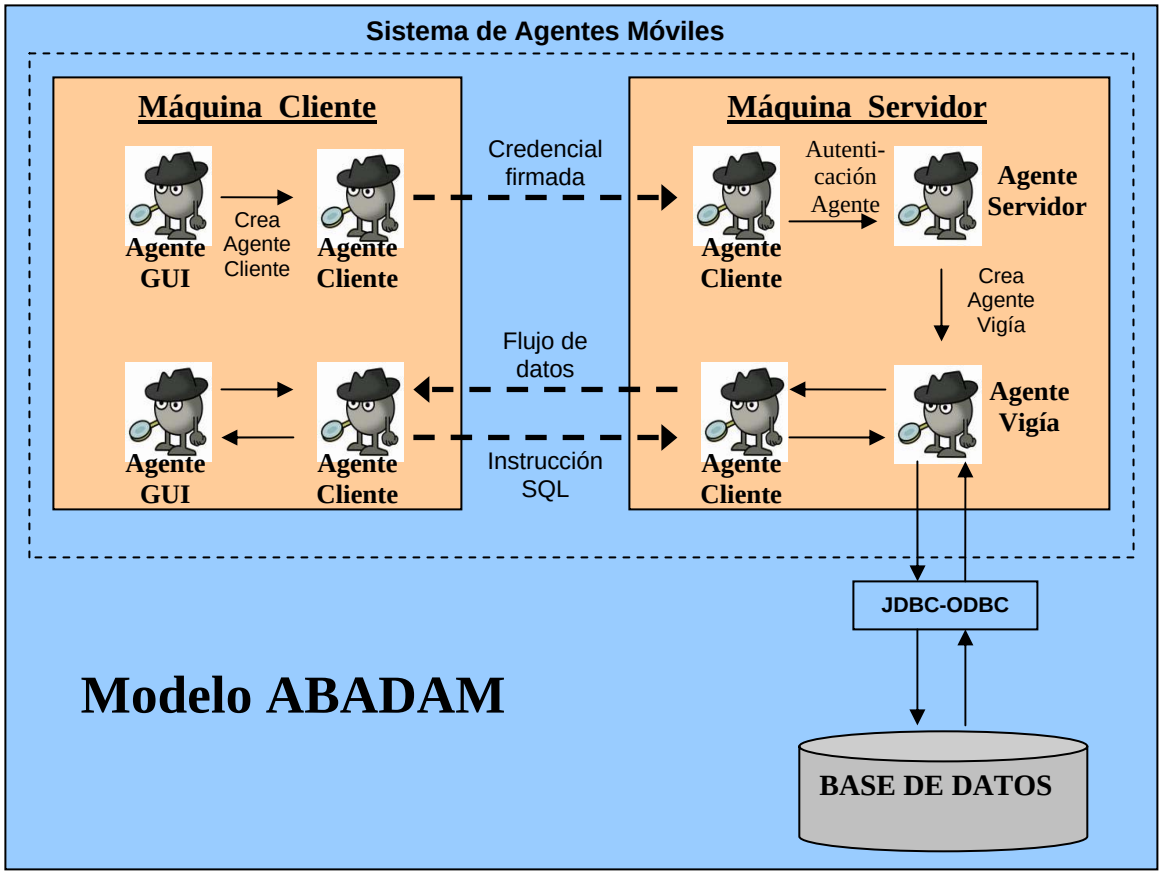


Figura 9. Modelo Propuesto ABADAM

5.1.1 Sistema de agentes móviles

El sistema de agentes móviles propuesto en este proyecto está clasificado de la siguiente forma:

- **Agente GUI**. La interfaz gráfica del usuario de ABADAM es invocada por el agente GUI que consiste en un agente estacionario y permite al usuario digitar su usuario, contraseña y dirección IP del servidor. Estos datos son entregados al agente móvil cliente para que se conecte a la base de datos,

si el agente cliente regresa con éxito de la conexión se le solicitará que proceda a la captura de la instrucción SQL para realizar la operación deseada en la base de datos y desplegar los resultados de la transacción.

- **Agente Servidor**. El agente servidor es un agente estacionario ubicado en la máquina donde se encuentra la base de datos, este agente tiene la función de crear un agente vigía por cada agente cliente válido que llegue al servidor a establecer una sesión con la base de datos.

- **Agentes clientes**. El agente cliente es un agente móvil que creado por la interfaz del usuario y se comunica con el agente servidor, el cual le asigna un agente vigía con quien tratará el cliente mientras dure la sesión.

- **Agentes vigías**. El agente vigía es un agente estacionario creado por el agente servidor por cada agente cliente valido que llegue al servidor. Este agente vigía tiene la función de comunicarse con el agente cliente mientras dure la sesión del usuario para que se de el intercambio de mensajes entre el cliente y el servidor.

Por cada usuario conectado a la base de datos del servidor, existirá un agente móvil cliente y por consiguiente existirá un agente vigía por cada agente cliente. De la misma manera si un usuario cierra su sesión o interfaz se destruirá el agente cliente y vigía correspondiente a ese usuario.

5.1.2 Modelo de Seguridad

El modelo de seguridad propuesto en ABADAM brinda un alto nivel de seguridad para proteger al servidor de base de datos de agentes maliciosos dando solución al problema del agente malicioso expuesto en la sección 4.1.1 del capitulo anterior debido a que solo tendrán acceso aquellos agentes válidos o confiables. El modelo de seguridad de ABADAM está basado fundamentalmente en el proceso de autenticación o identificación de un agente válido que llega al servidor, este proceso será capaz de detectar agentes no válidos y los considerará como agentes maliciosos, los cuales eliminará en el mismo instante que lleguen al servidor. Además ABADAM brinda un componente adicional de seguridad convirtiéndose también en un proceso colaborador en la autenticación de los usuarios, debido a que si algún usuario mal intencionado o hacker, creara un agente móvil cliente portando un login y password valido no podrá realizar la autenticación al no portar la credencial que lo acredite como un agente cliente confiable y será destruido inmediatamente.

La plataforma que se utilizó para crear el sistema de agentes móviles de ABADAM fue Aglets 2.0.2, entre los servicios de seguridad que esta plataforma cuenta[29] no presenta la criptografía, por lo que el modelo de seguridad se diseño basado en el esquema de firmas digitales y listas de control de acceso, para evitar que agentes maliciosos no pudieran acceder la base de datos localizada en el servidor.

[29] LANGE, Danny y OSHIMA Mitsuru. Programming and Deploying Java Mobile Agentes with Aglets. Addison Wesley. 1998. p. 176-177.

ABADAM implementa un sistema de firma digital utilizando un algoritmo de la familia SHA[30] (Secure Hash Algorithm) , para establecer una comunicación segura entre el cliente y el servidor se genera una credencial firmada digitalmente. Esta credencial debe contener el nombre del host de donde proviene el agente móvil, la dirección IP, nombre de la clase u objeto del agente, y la identificación interna del agente. ABADAM cuenta con varios niveles o momentos de seguridad para detectar si un agente es malicioso (ver figura 10).

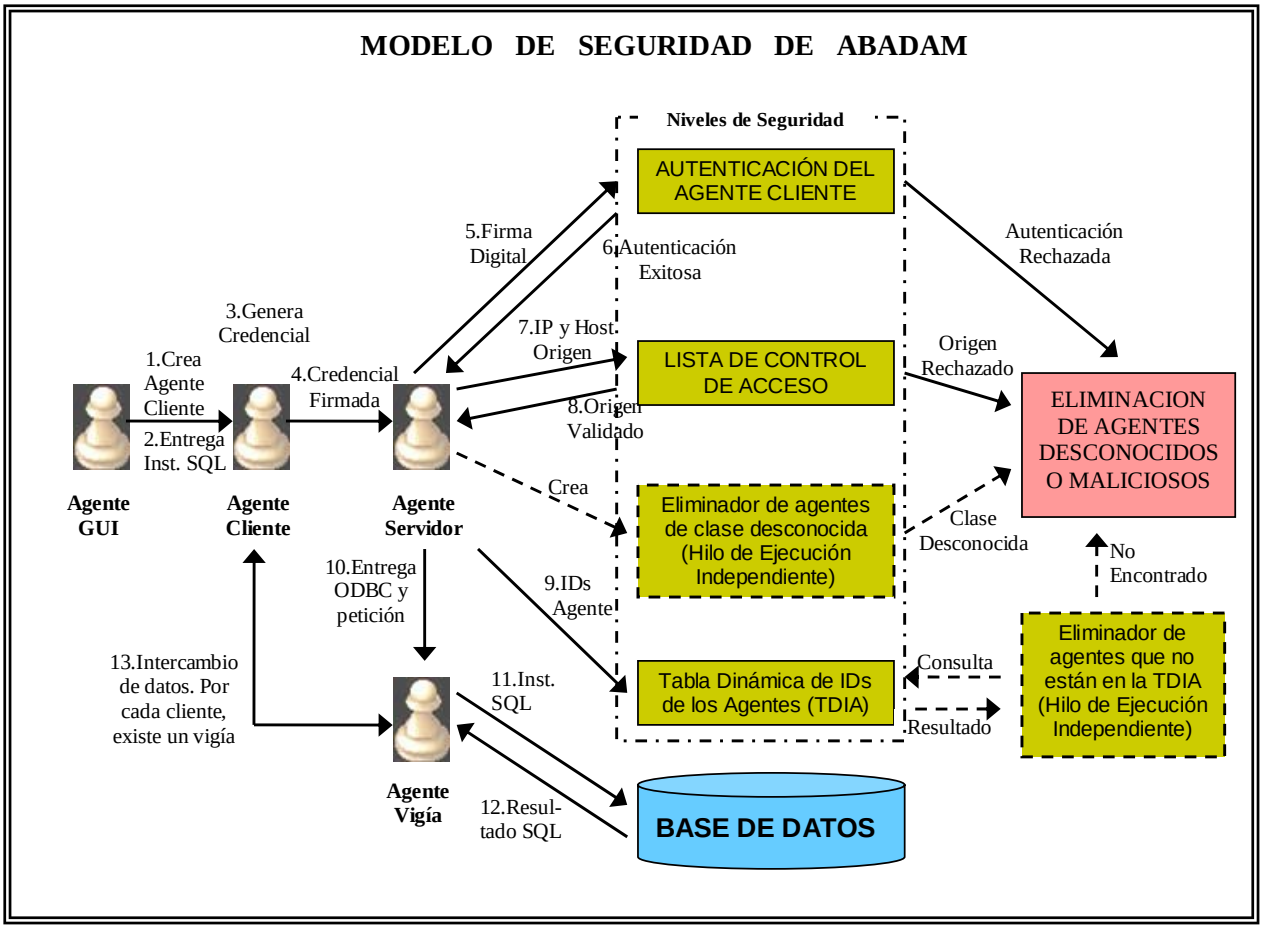


Figura 10. Modelo de Seguridad de ABADAM

[30] OAKS, Scott. Java Security. Second Edition. O'reilly & Associates, Inc. 2001. 618 p.

En la figura 10 están enumerados 13 pasos que se dan, en su orden, al momento en que un cliente desea conectarse a la base de datos, si el cliente no resulta confiable, estos pasos no se darán en su totalidad y el agente cliente será eliminado dependiendo del caso.

El primer nivel de seguridad de ABADAM se da en el momento que el agente móvil, despachado por el cliente, llega al servidor y consiste en validar la credencial por su firma digital, si el agente servidor no puede validar la firma se asumirá que está tratando con un agente malicioso y lo eliminará instantáneamente, de lo contrario el proceso seguirá su curso.

El segundo nivel de seguridad ocurre una vez se ha comprobado la fidelidad de la firma digital, en ese momento se realiza una validación adicional de seguridad que consiste en verificar mediante el nombre del host y la dirección IP del cliente coincida con algunas de las entradas de la lista de control de acceso ubicada en el servidor, si el agente cliente no cumple con esta validación será destruido por considerarse un agente malicioso. En este nivel controla que agentes maliciosos o no validos se apoderen de un agente valido y traten de acceder al servidor.

El tercer nivel de seguridad consiste en crear una lista dinámica con la identificación interna de cada agente después de haber superado los niveles explicados en los dos párrafos anteriores, la identificación interna de cada agente corresponde a un único número de identificación asignado al agente, de tal manera que no pueden existir agentes válidos que no estén en esa lista, de lo contrario serían considerados agentes maliciosos y serían eliminados del servidor.

Un cuarto nivel de seguridad independiente a los mencionados en los tres párrafos anteriores consiste en un hilo de ejecución del agente servidor que se encarga de

destruir cualquier agente que no pertenezca a un nombre de objeto valido (entiéndase como objeto válido los tipos de agente manejados por ABADAM), este control de seguridad surge para eliminar cualquier clase de agente malicioso que quiera atacar el servidor.

El Sistema de agentes propuesto en este proyecto evita que agentes no deseados o malintencionados lleguen al servidor, con lo cual elimina cualquier posibilidad que el servidor colapse por sobrecarga de agentes no válidos.

Los aspectos descritos resaltan claramente el nivel de seguridad brindado por el Sistema de Agentes de este proyecto, además de mejorar el rendimiento de la red al no mantener canales constantes entre cliente y servidor, aspecto que no es fácilmente manejado en la forma tradicional de comunicación cliente servidor.

5.1.3 Base de Datos

La base de datos con la que se probó el sistema de agentes de este proyecto está claramente descrita en el capitulo 3. El agente vigía es el que se conecta a la base de datos mediante el ODBC creado en el servidor de base de datos.

A pesar que en este proyecto se trabajó con el motor de MS-SQL Server, ABADAM está diseñado para adaptarse a cualquier motor de base de datos, basta con configurar el ODBC apropiado.

El sistema en general ofrece un alto grado de flexibilidad y escalabilidad en cuanto al crecimiento en el número de sedes de la universidad simulada para trabajar la distribución de los datos, pues no solo se aplica a un modelo de universidad, sino

que en general, ABADAM puede ser utilizado en cualquier proceso que requiera la autenticación de usuarios para acceder a los datos.

Los servicios de ABADAM no actúan en el marco del proceso de autenticación de usuarios del servidor de base de datos, sino que implementa un sistema de seguridad previo que autentica los agentes clientes que portan el login y password para acceder a la base de datos, mejorando los niveles de seguridad en ambientes tan hostiles como son los canales de comunicación compartidos que se manejan en sistemas distribuidos.

Con su sistema de agentes, ABADAM evita que el servidor de base de datos esté expuesto a ataques de otros agentes o de usuarios no válidos, ya que a dicho servidor solo llegan peticiones de usuarios validos y no directamente, sino mediante el mismo sistema de seguridad de agentes implementado, luego entonces, el sistema de seguridad de ABADAM no afecta ni riñe con la seguridad que maneja el servidor de base de datos, sino que le colabora para que permanezca descargado de la tarea de verificar la validez de usuarios , evitándole así el consumo innecesario de recursos (no permanencia del canal de comunicación) y la exposición a ataques .

5.2 Utilizando ABADAM

El sistema ABADAM tiene dos componentes, el que se ejecuta en el cliente y el que se ejecuta en el servidor. En el anexo A de esta tesis encontrará las instrucciones de instalación. El entorno en que se ejecutan los agentes de ABADAM es Tahiti, por lo tanto debe estar ejecutándose tanto en el cliente como en el servidor para que pueda funcionar la aplicación, por defecto se ejecuta el

Tahiti cuando se arranca ABADAM pero si se cierra Tahiti ejecutándose el aplicativo este también se cerrará.

5.2.1 ABADAM Cliente

La figura 11 muestra la interfaz que carga el Agente GUI cuando está ejecutándose.

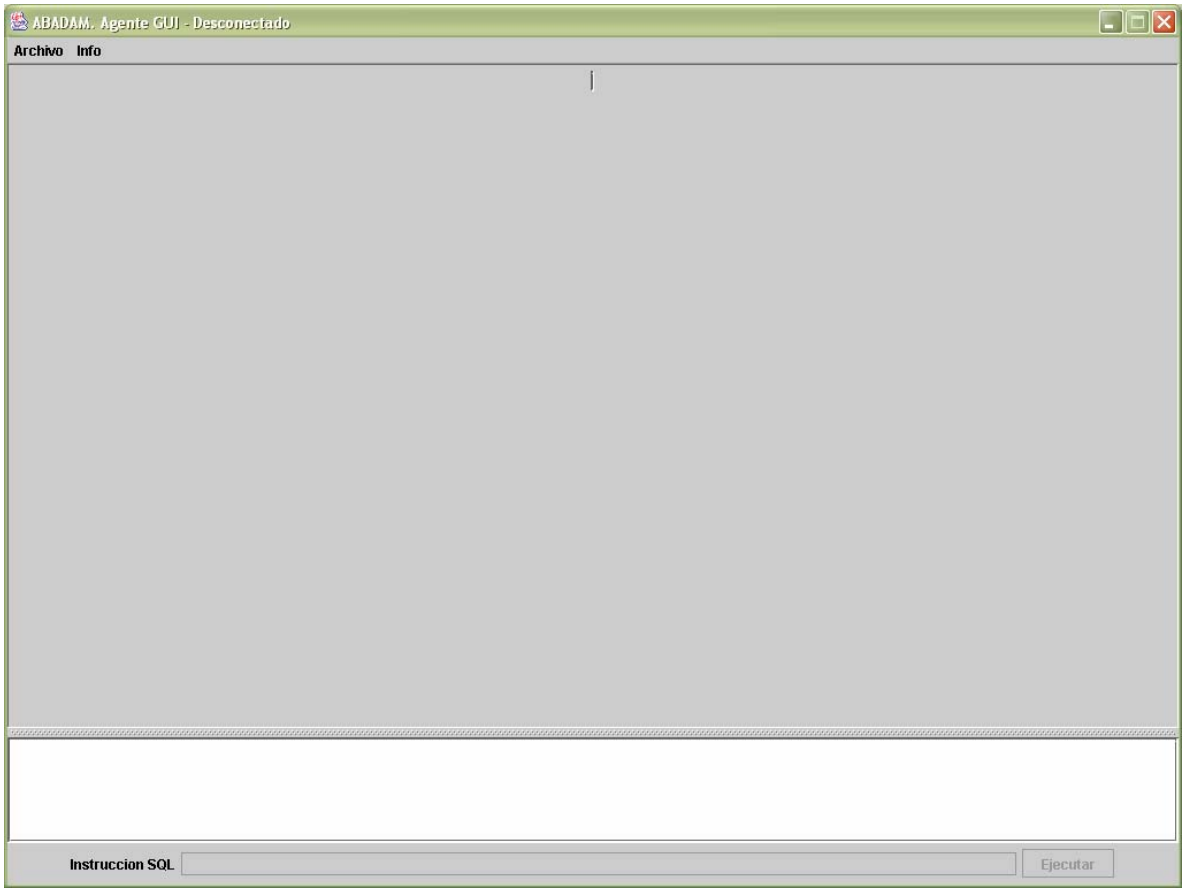


Figura 11. Pantalla inicial de ABADAM Cliente

Para conectarse a la base de datos se debe seleccionar la opción Conectarse del menú Archivo (Figura 12)

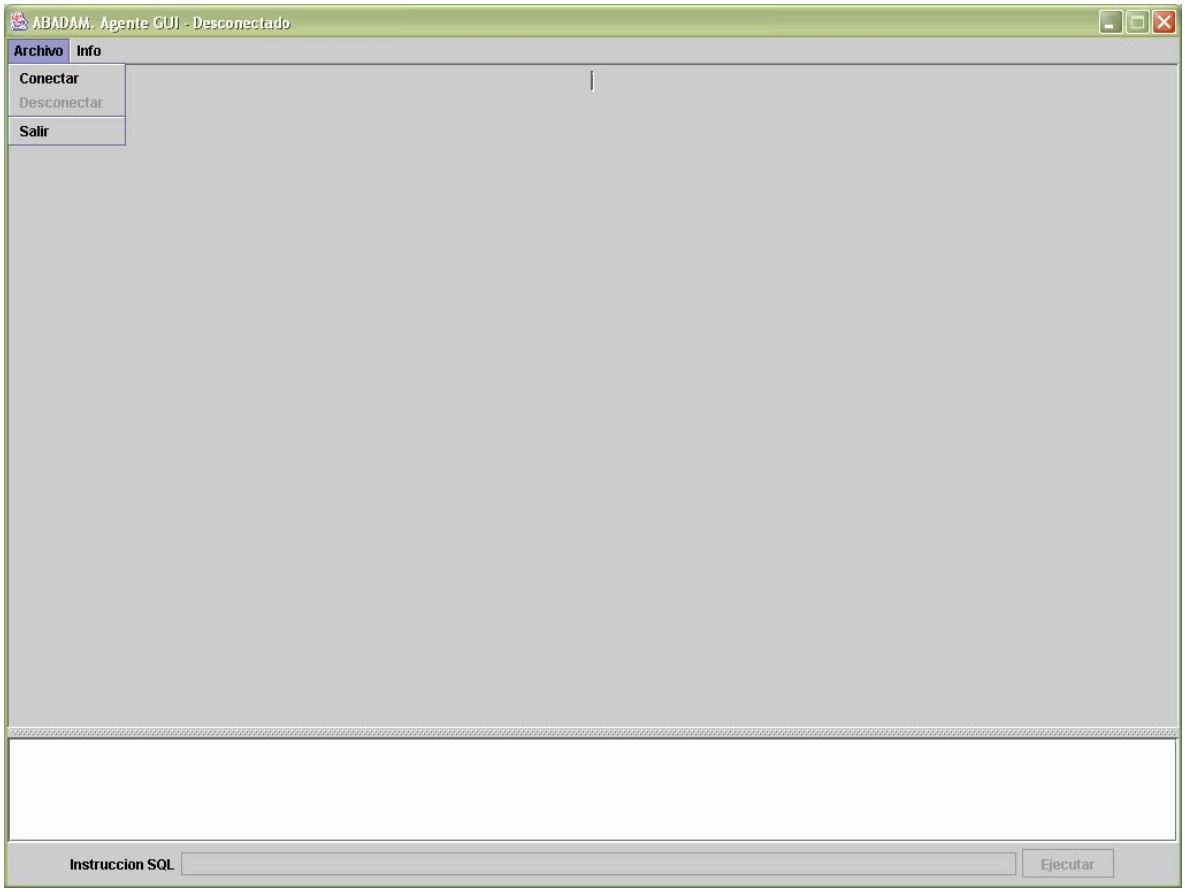


Figura 12. Opción Conectar de la interfaz ABADAM Cliente

Al seleccionar la opción Conectar se le solicitará al usuario la dirección IP del servidor, login y password (Figura 13). La credencial de seguridad que acredita al

agente móvil cliente como agente valido ante el servidor viaja encapsulada en el código del aglets.

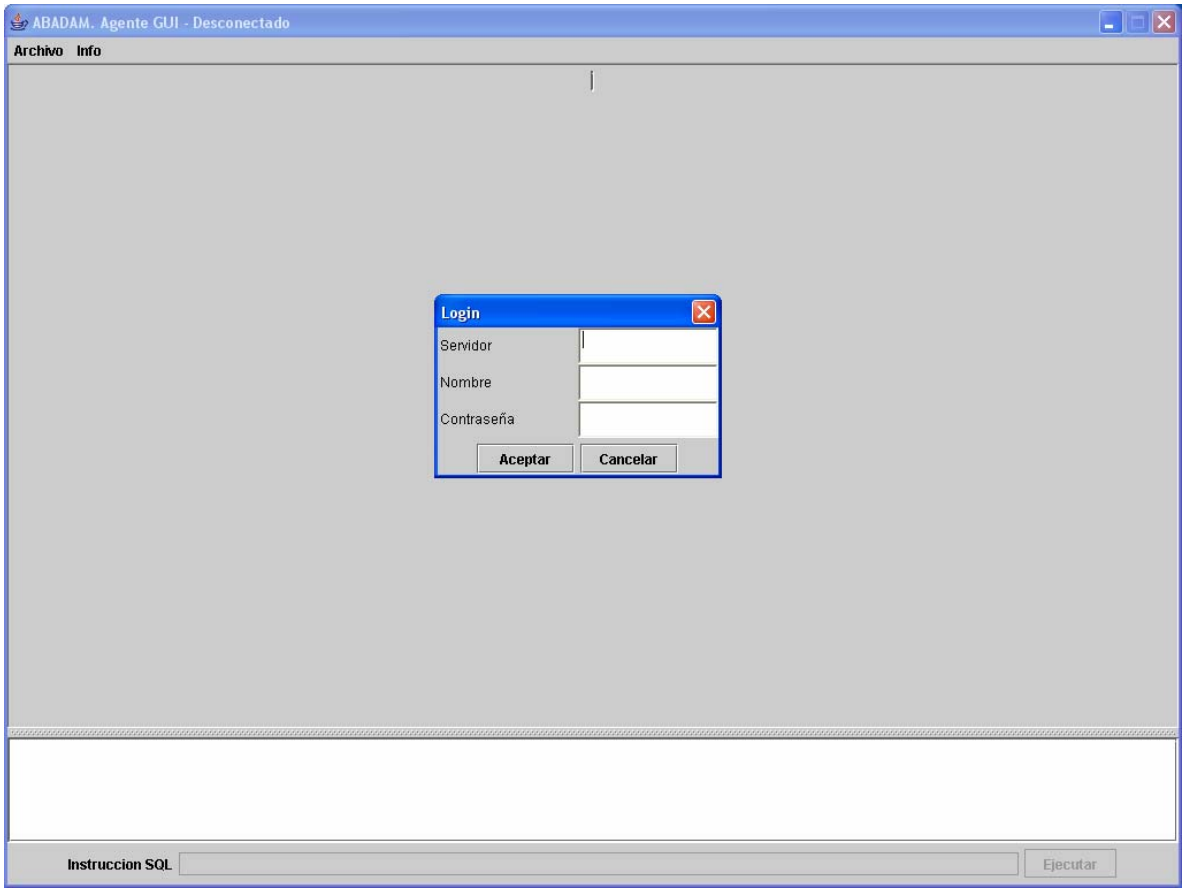


Figura 13. Captura de dirección IP del servidor, login y password.

Si la conexión tiene éxito aparecerá en la barra de titulo de la ventana un mensaje que indica a que dirección IP se encuentra conectado y se habilitará el campo para digitar la instrucción SQL (Figura 14).

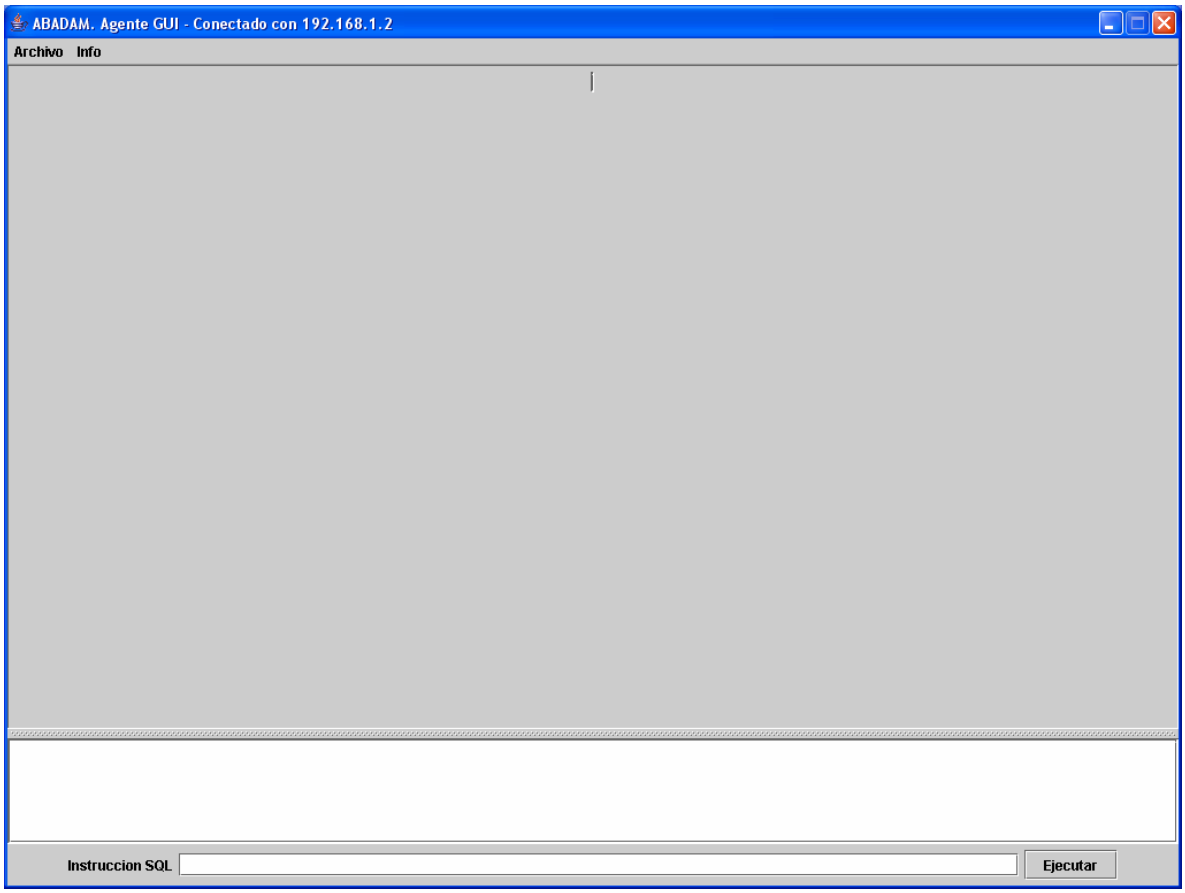


Figura 14. Agente conectado y esperando a que se digite alguna instrucción

Una vez se haya logrado la conexión se puede proceder a digitar una instrucción SQL, si la instrucción corresponde a un SELECT el resultado de la consulta se mostrará en una tabla dinámica acorde al número de filas y columnas que se generen (Figura 15).

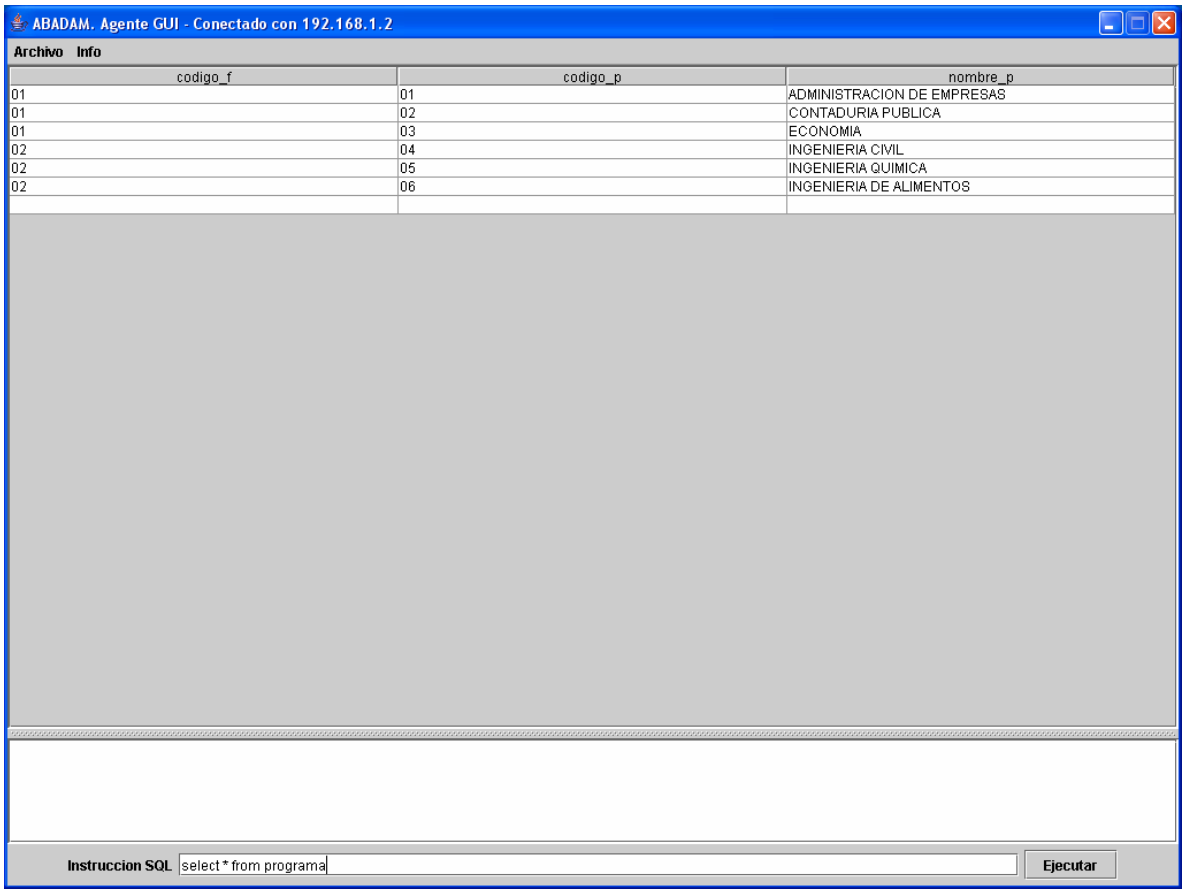


Figura 15. Ejecución de una instrucción SELECT

5.2.2 ABADAM Servidor

Al ejecutar el agente servidor se solicita al administrador del sistema que digite el nombre del ODBC de la base de datos que se desee conectar (Figura 16).

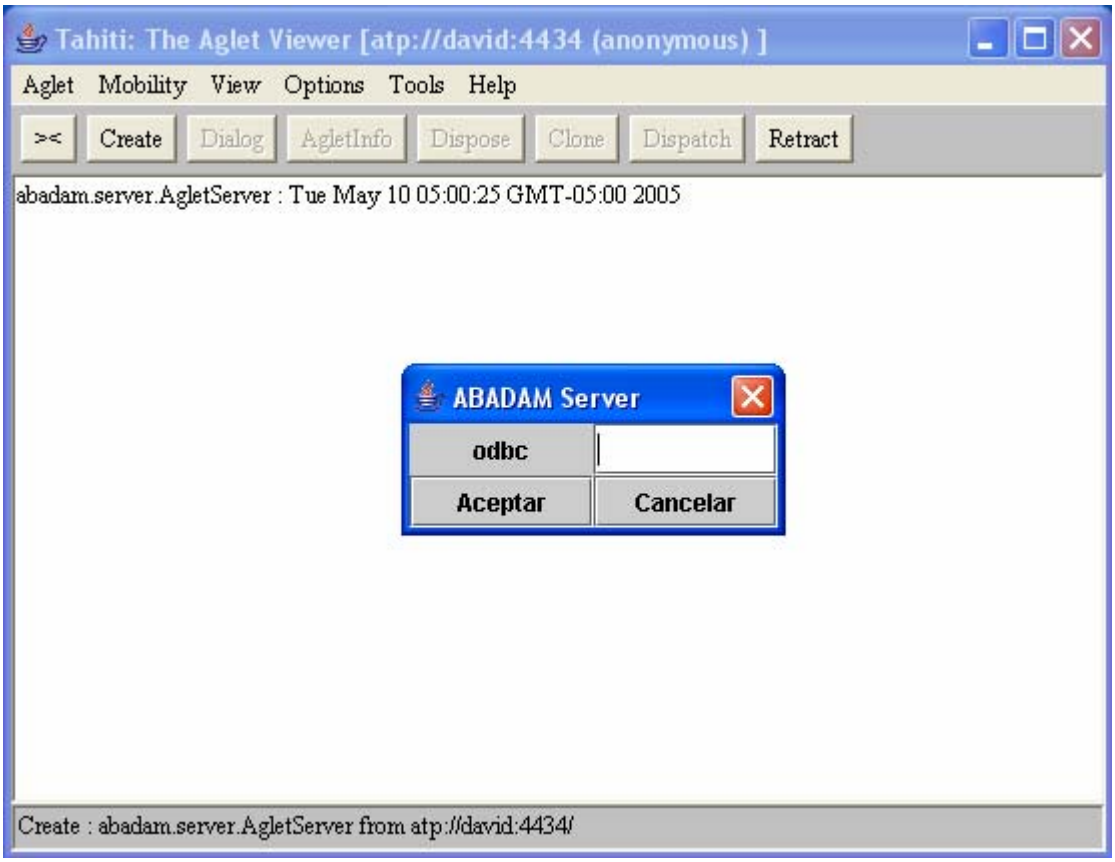


Figura 16. Pantalla inicial de ABADAM Server.

En caso de que se digitara mal el nombre del ODBC o no existiera se indicaría mediante un mensaje en el no se encuentra el origen de los datos (Figura 17).

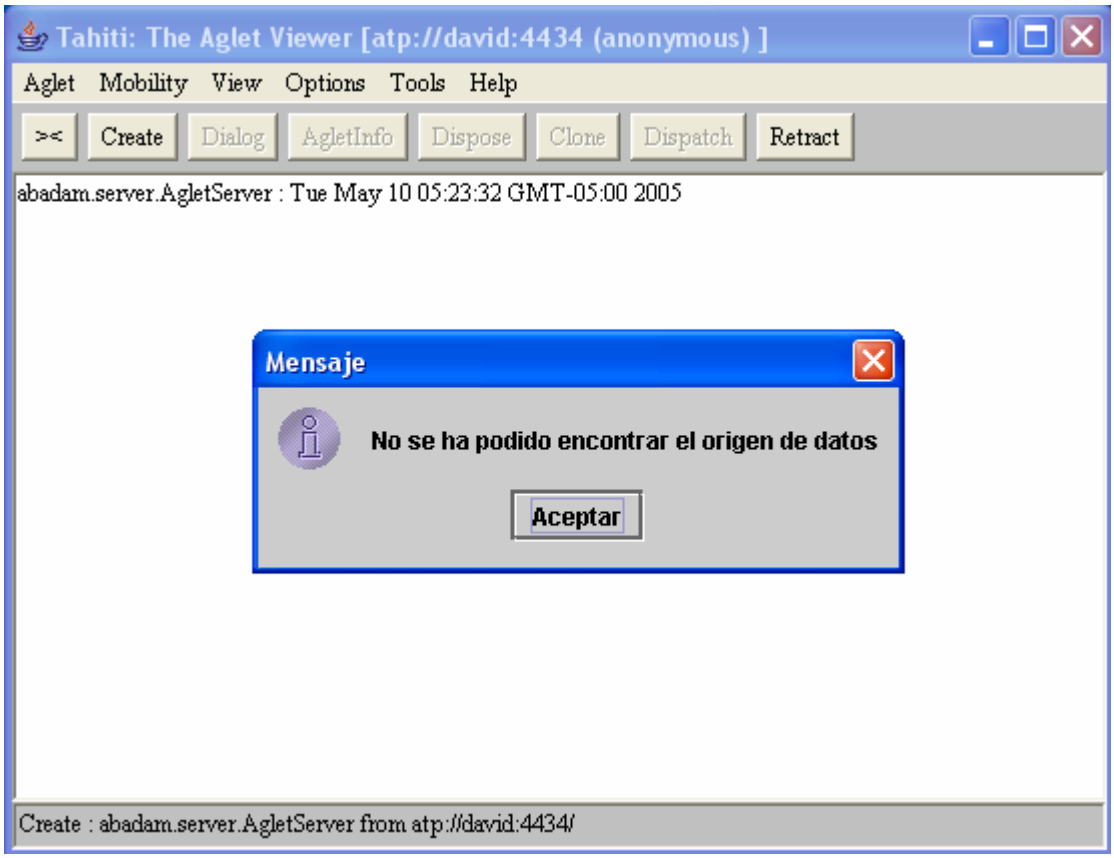


Figura 17. Mensaje de error al no encontrar el ODBC

Complementario al modelo de seguridad expuesto en la sección 5.1 de este capitulo, tahiti cuenta con una interfaz gráfica útil para monitorear el tráfico de agentes en el servidor, en la vista principal (Figura 18) aparecen todos los agentes que se estén ejecutándo en el servidor, en la figura 17 solo aparece ejecutándose el agente servidor *abadam.server.AgletServer,* pero por cada usuario o agente móvil cliente valido que se conecte a la base de datos debe aparecer un *abadam.server.Watcher* y cuando un agente móvil cliente llegue al servidor debe aparecer un agente *abadam.cliente.Cliente* durante el instante que se ejecuta en el servidor.

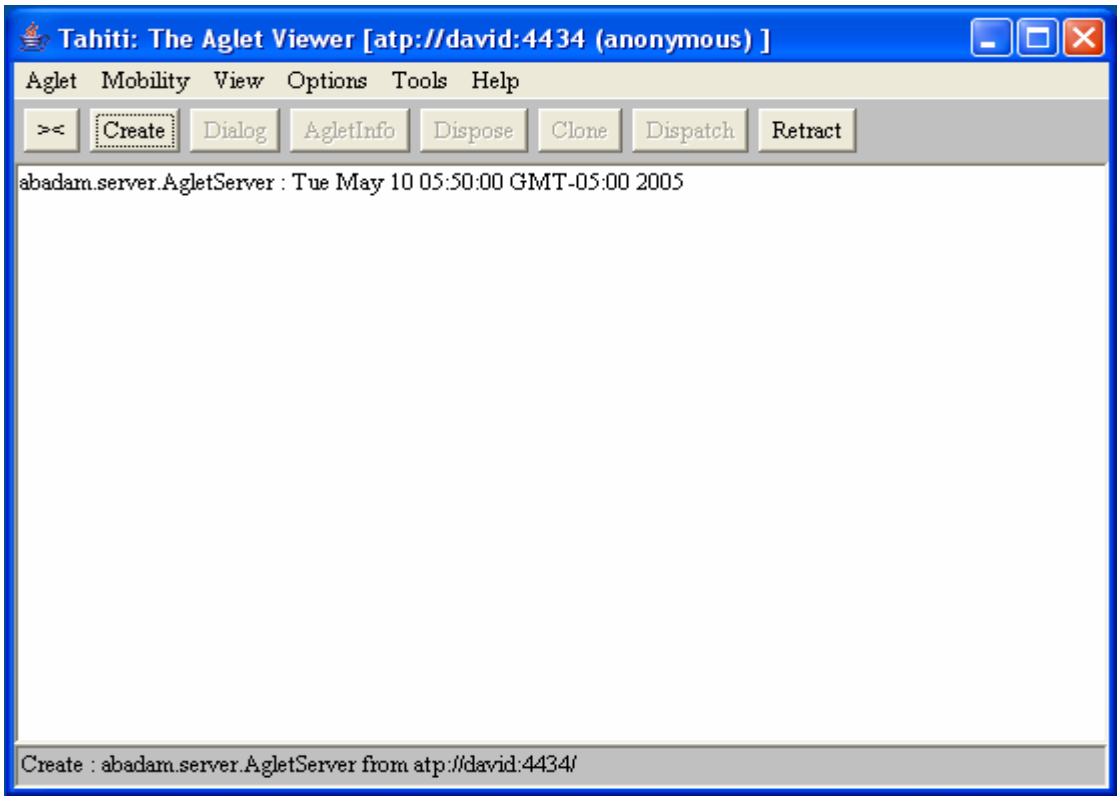


Figura 18. Tahiti – Entorno de ejecución de los agentes

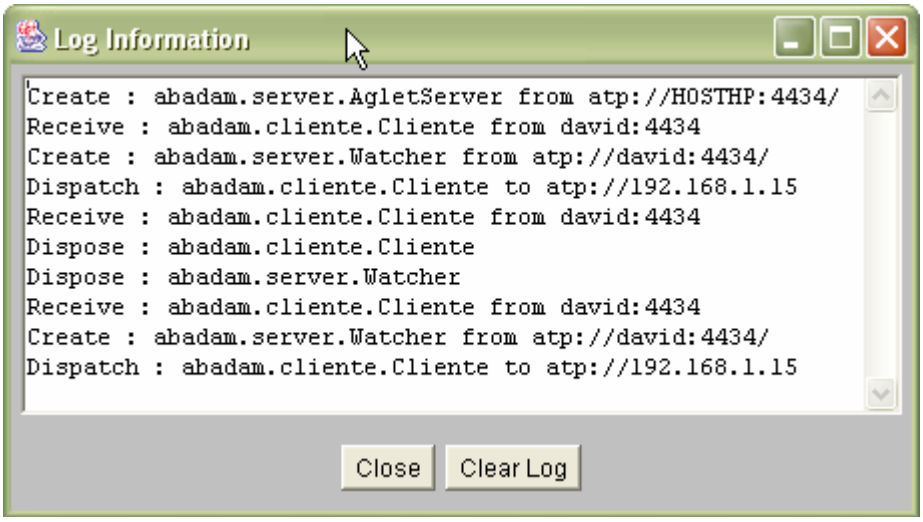


Figura 19. Tahiti – Log Information

Tahiti también cuenta con un log de registros para revisar que agentes han llegado al servidor (Figura 19).

CONCLUSIONES

En ABADAM los usuarios que desean conectarse a la Base de Datos lo hacen mediante el sistema de agentes móviles que brinda un alto nivel de seguridad contra ataques de otros agentes porque aunque un agente malicioso obtuviese un login y password validos para conectarse a la base datos, no lo lograría al no tener la credencial que lo amerite como agente de ABADAM.

El alto nivel de seguridad se extiende también al no permitir el alojamiento de agentes no deseados o maliciosos en el servidor porque serán destruidos en el mismo instante que lleguen al servidor, esto también evita que el sevidor sea colapsado por la llegada de agentes no deseados. Este aspecto de ABADAM soluciona uno de los problemas por los cuales los agentes móviles no han tenido mucha acogida, como es el problema de seguridad en cuanto al manejo de ataques de agentes maliciosos expuesto en la sección 4.1.1 de este documento.

Uno de los aportes mas importantes de ABADAM al campo de acción de los agentes móviles, además del alto nivel de seguridad que maneja, es que brinda un mecanismo colaborativo en el proceso de autenticación de usuarios, ya que autentica previamente el agente cliente que porta el login y password como agente valido, aspecto que se refleja de manera positiva en el rendimiento del servidor de base de datos, pues, a dicho servidor solo llegaran peticiones válidas de usuarios. Esto no sucede en algunos sistemas tradicionales en los cuales muchos usuarios no autorizados establecen contacto con el servidor de base de datos produciendole a éste consumo de recursos y riesgos innecesarios.

ABADAM es una herramienta multiplataforma porque se puede ejecutar sobre cualquier sistema operativo que tenga instalada la máquina virtual de Java y, ademas, permite al usuario conectarse no solo a bases de datos distribuidas, como se planteó inicialmente en este proyecto, sino que, ademas está diseñado para conectarse a cualquier base de datos mediante un driver ODBC, sea distribuida o no, y ejecutar cualquier instrucción SQL sobre la Base de Datos a la que se logre conectar.

Las aplicaciones donde se puede implementar este modelo no solo se limitan al aspecto de seguridad, sino que además, ABADAM al no tener ocupado el canal de cominicaciones permanentemente mientras se está conectado a la base de datos ofrece un mayor rendimiento en la red institucional, sobre todo en redes donde hay demasiado tráfico, como es el caso de instituciones que además de soportar el tráfico de datos de las aplicaciones empresariales también tienen que soportar el tráfico exagerado que producen los usuarios conectados a la red navegando y descargando archivos.

BIBLIOGRAFÍA

ACEBAL, César y CUEVA, Juan. Acceso a bases de datos distribuidas mediante el uso de agentes móviles. NOVATICA Julio/Agosto 2000 No. 146. p. 48.

AVANCINI, Henri. FraMaS: Un Framework para Sistemas Multi-agente basado en Composición. Tesis de Maestría. Universidad Nacional del Centro de la Provincia de Buenos Aires. 2000. 154 p.

BELL, David y GRIMSON, Jane. Distributed Database Systems. Addison-Wesley Longman Publishing Co., Inc. 1992. 410 p.

BIGUS, Joseph y BIGUS, Jennifer. Constructing Intelligent Agents Using Java. 2 ed. John Wiley & Sons Inc., 2001. 432 p.

BILL, Venners. Solve real problems with aglets, type of mobile agent. JavaWorld – Under the hood Magazine. Mayo 1997. p. 2-4.

BORSELIUS, Niklas. Mobile agent security. Electronics & Communication Engineering Journal. Volumen 14. p. 211-218. 2002.

COMPUTER ASSOCIATES. Ingres r3, Distributed Option User Guide. Computer Associates Internacional, Inc. 2004. 161 p.

COMPUTER ASSOCIATES. Ingres r3, Replicator Option User Guide. Computer Associates Internacional, Inc. 2004. 229 p.

FRITZ, Hohl. An approach to solve the problem of malicious hosts in mobile agent systems. Institute of parallel and distributed systems, University of Stuttgart. Alemania. 1997.

GASSER, Morrie. Building a Secure Computing System. Val Nostrand Reinhold. 1988. 308 p.

JEONG, Min-A; KIM, Jung-Ja y WON, Yonggwan. A Flexible Database Security System using Multiple Access Control Policies. IEEE. 2003. p. 236.

LANG, Dany y OSHIMA, Mitsuru. Programming and Deploying Java Mobile Agents with Aglets. Addison Wesley Professional, 1998. 256 p.

MAN, Mo y WEI V. A Taxonomy for Attacks on Mobile Agent. IEEEXplore. EUROCON'2001 Trends in Communications, International Conference. Volumen 2. p. 385-388. 2001.

MOSQUERA, Celestino. Análisis y estudio experimental de herramientas basadas en agentes. Tesis de Grado. Universidad de Coruña – España. 2001. 143 p.

O´HARE, G. y JENNINGS, N. Foundations of Distributed Artificial Intelligence. John Wiley & Sons. 1996.

OAKS, Scott. Java Security. Second Edition. O'reilly & Associates, Inc. 2001. 618 p.

ORACLE CORPORATION. Oracle Database 10g: Heterogeneous Connectivity Administrator's Guide. Oracle Corporation. 2003. 178 p.

ORACLE CORPORATION. Oracle: Heterogenous Services. Oracle Corporation. 2001. 150 p.

ORACLE CORPORATION. Oracle's Solutions For The Distributed Environment. Oracle Corporation. 2001. 22 p.

OZSU, M. Tamer y VALDURIEZ, Patrick. Principles of Distributed Database Systems. 2 ed. Pearson Education, 1999. 666 p.

PAPASTAVROU, Stavros; SAMARAS, George y PITOURA, Evaggelia. Mobile agents for WWW distributed database access. Data Engineering, 1999. Proceedings., 15th International Conference on , 23-26 March 1999 Pages: 228 - 237

PEREZ, Jesús. Sahara: Arquitectura de seguridad integral para sistemas de agentes móviles. Tesis Doctoral. Universidad de Oviedo - España. 2000. 364 p.

SILBERCHATZ, Abraham; KORTH, Henry y SUDARSHAN S. Fundamentos de Bases de Datos. 3 ed. McGrawHill. 1999. 641 p.

SUMMERS, Rita. Secure Computing. McGrawHill. 1997. 692 p.

VIGNA, Giovanni. Mobile Agents and Security, Lecture notes in Computer Science. Springer-Verlag Telos. 1999. 256 p.

YAN, Wang; KEN, Law y TAN, Kian-Lee. A mobile agent based system for distributed database access on the internet. Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference on , Volume: 2, 21 25 Aug. 2000 Pages:1587 - 1590 vol.2.

WOOLDRIDGE, M.; JENNINGS, N. Intelligent Agents: Theory and Practice. Knowledge Engineering Review. Octubre de 1995.

ANEXO A. INSTALANDO ABADAM

En el CD que acompaña esta obra se encontrarán dos carpetas: Instalador_Cliente e Instalador_Servidor. Cada carpeta tiene un archivo ejecutable que descomprime en la raíz del disco duro las carpetas y archivos necesarios para ejecutar ABADAM.

EJECUCION DE ABADAM EN LA MAQUINA CLIENTE

Después de ejecutar el archivo de instalación del cliente crear un acceso directo que ejecute el comando: \Aglets\bin\cliente.bat, o ejecutarlo directamente si lo prefiere.

EJECUCION DE ABADAM EN LA MAQUINA SERVIDOR

Después de ejecutar el archivo de instalación del cliente crear un acceso directo que ejecute el comando: \Aglets\bin\servidor.bat, o ejecutarlo directamente si lo prefiere.

ANEXO B. INSTALANDO SQL SERVER

REQUERIMIENTOS MÍNIMOS DE HARDWARE:

SQL Server 2000 requiere el siguiente hardware como mínimo:

- Computadora: DEC Alpha AXP y sistemas compatibles, Intel o compatibles (Pentium 166 Mhz. o superior, Pentium PRO, o Pentium II).
- Memoria: 32 MB de RAM.
- Unidad de Disco: Un CD-ROM, más un disco duro con al menos 80 MB de espacio libre en disco para la instalación mínima.

La siguiente tabla muestra la cantidad mínima de espacio disponible en disco que requieren las diferentes instalaciones:

Opción de Instalación	Espacio en Disco
Completa	210 MB
Típica	185 MB
Herramientas de Administración	90 MB
Mínima	80 MB

La cantidad mínima de memoria para SQL Server en un procesador Intel es de 16 megabytes (MB). SQL Server para plataformas RISC requerirá de más memoria debido a la cantidad promedio de baja densidad de las instrucciones de la computadora.

Sin embargo, considerando en general al software, hardware, aplicaciones e inversión de personal en los sistemas cliente/servidor, agregar más memoria es generalmente una sabia decisión, y por comparación una inversión económica. Muchas instalaciones aseguran que 32 MB es un buen inicio, y no es poco común que se configuren los servidores con 128 MB o incluso más memoria, la cual asignan para usos en beneficio de los usuarios.

El punto en el que la memoria deja de proporcionar beneficios generales, depende completamente de cada situación, y es determinada principalmente por la ubicación o referencia de los accesos de la base de datos. El punto importante que se debe recordar es que los incrementos de memoria que son relativamente pequeños, tan solo un porcentaje del total de la memoria, rara vez aportan un beneficio significativo. Dos cosas controlan esta situación: SQL Server usa memoria principal extra como buffer de caché; y la mayoría de los estudios de estadísticas de caché indican que se presenta una curva ligeramente plana después de varios *megabytes*.

Es por esta razón, que en un equipo de 32 MB, si se otorga a SQL Server una memoria de 14 MB, 16 MB, o 18 MB, difícilmente habrá una diferencia significativa en su desempeño. Por el contrario, intentar "saturar" Windows NT con excesiva memoria para SQL Server podría resultar en un bajo desempeño debido al excesivo mapeo.

Para resumir, el parámetro óptimo de memoria depende de cuanta RAM esté instalada en el servidor SQL Server y de qué otras aplicaciones estén en ejecución

en dicho servidor. En un servidor dedicado para SQL Server, con 32 MB de memoria física RAM, se puede configurar 16 MB para uso por SQL Server.

Esto posibilitaría que Microsoft Windows 2000 Server tenga suficiente memoria para la ejecución de sus propios procesos y evitará la paginación a disco duro.

Es importante fijar la memoria para SQL Server de forma apropiada, es decir, fijar la cantidad de RAM dedicada a SQL Server. Este parámetro depende de la cantidad de RAM física que tenga el servidor y del uso y requerimientos de prestaciones de SQL Server.

La memoria está designada en bloques de 2 KB. Por ejemplo, para un servidor dedicado a SQL Server con 128 MB de RAM, se puede fijar la memoria para SQL Server a 64 MB (32.768 bloques de 2-KB).

El proceso de instalación

Cuando se inicia la instalación, se ejecuta un único programa, sin importar el sistema operativo ni la configuración final del servidor. La versión Estándar de SQL fue la utilizada para el proyecto, cuyos pasos de instalación se describen a continuación:

1. En la primera pantalla del proceso de instalación (ver figura 1) se escoge la primera opción: SQL Server 2000 Components.

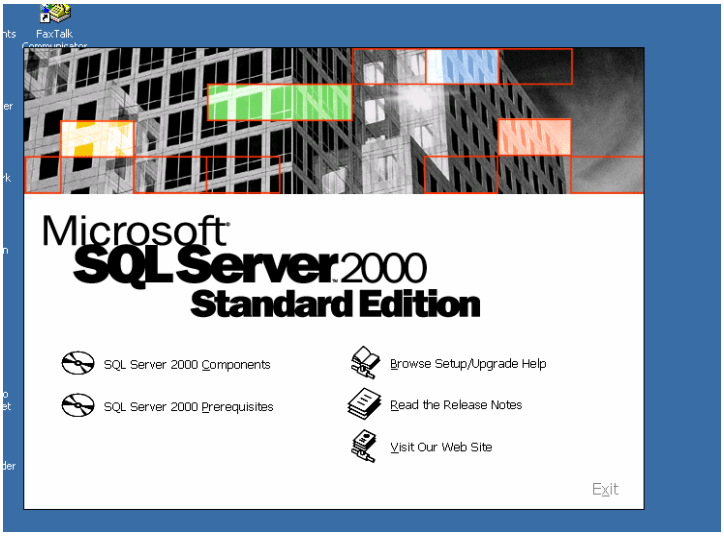


Figura 1. Pantalla inicial de la instalación

2. Luego en la ventana siguiente la opción Install Database Serve con lo que pasamos al paso siguiente.
3. Se presenta un Wizard que ayuda en la instalación de una nueva instancia de SQL Server o a modificar una instancia existente. Siguiente.
4. Se escoge el nombre del equipo en donde se realizará la instalación o modificación de la instancia. El equipo puede ser local o remoto, en este caso se trata de una instalación local. Siguiente.
5. Se escoge la opción Create a new instance of SQL Server, or install client tools. Siguiente.
6. Se registra el nombre del usuario y la compañía para la cual trabaja. Siguiente
7. Se lee la licencia y se acepta o se rechazan los términos estipulados. Para este caso se aceptan. Siguiente
8. Se selecciona el tipo de instalación Server and Client Tools. Siguiente
9. Se escribe el nombre de la instancia de SQL Server a instalar. El nombre debe tener máximo 16 caracteres y empezar con una letra o cualquier otro carácter aceptable. Siguiente.

10. Se escoge la opción de instalación de Setup Typical. Siguiente
11. Se escoge la opción Use a Domain User Acount y se proporciona el nombre de usuario y el password para la autentificación para entrar a SQL Server. Para este caso se utilizó la cuenta de usuario de dominio Administrador y se colocó un password válido. Siguiente.
12. Se escoge el modo mixto de autentificación, de SQL y de Windows, se proporciona el password para el usuario “sa” o System Administrador. Siguiente.
13. Se escoge el modo de licencia por dispositivo, ya que cada máquina que se utilizó para la practica tiene solo un procesador. Continuar.
14. Con la información proporcionada se empieza a realizar la instalación
15. Dependiendo de las características del equipo o de la congestión de la red (en caso de instalación remota), se debe esperar mientras se realiza la instalación. Luego finalizar y ya estará lista la instalación o reparación de la instancia.

Herramientas de Administración

Asistente para copiar bases de datos

El Asistente de administración permite de manera muy sencilla hacer copia de la base de datos y de todos sus objetos a otro servidor. Desde SQL Server Enterprise Manager, en el menú Tools se escoge la opción Wizard…, con lo cual aparece la ventana mostrada en la figura 2, en donde se hace la selección allí resaltada, es decir Copy Database Wizard.

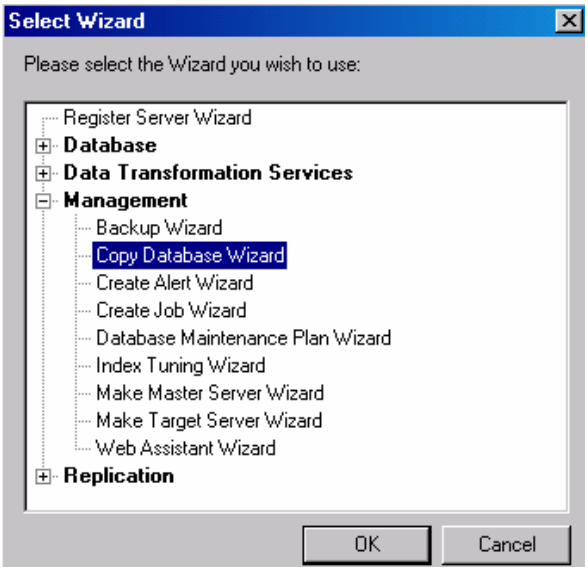


Figura 2. Selección del Wizard para copiar Bases de Datos

Luego de la selección anterior, aparece la ventana mostrada en la figura 3, en donde se informa al usuario todas los beneficios que obtiene al finalizar el wizard.

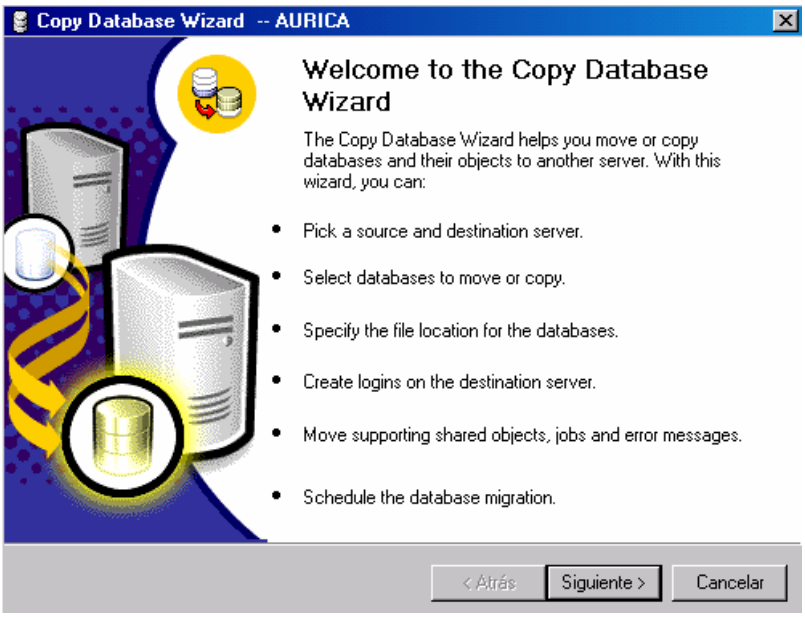


Figura 3. Inicio del Wizard para copiar bases de datos

Cabe anotar que el servidor al cual se va a realizar la copia, ya debe estar registrado como servidor SQL Server y debe estar activo en la red.

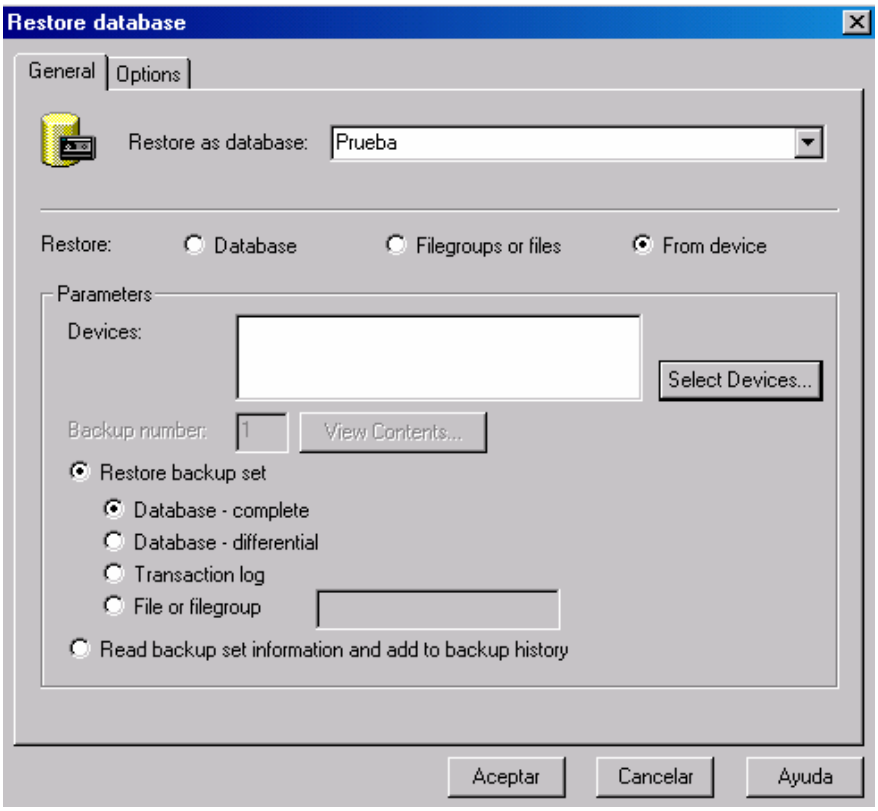
Luego se selecciona el nombre del servidor donde está la base de datos y se escoge el modo de autentificación, por Windows o por SQL Server. De manera análoga se hace en la ventana donde aparece el servidor de destino.

Se continua con el dialogo del wizard con las opciones recomendadas hasta completar toda la información necesaria para hacer la copia donde se pulsa finalizar.

Si hubo algún error en la operación, se mostrará una ventana en donde se puede hacer seguimiento a dicho error.

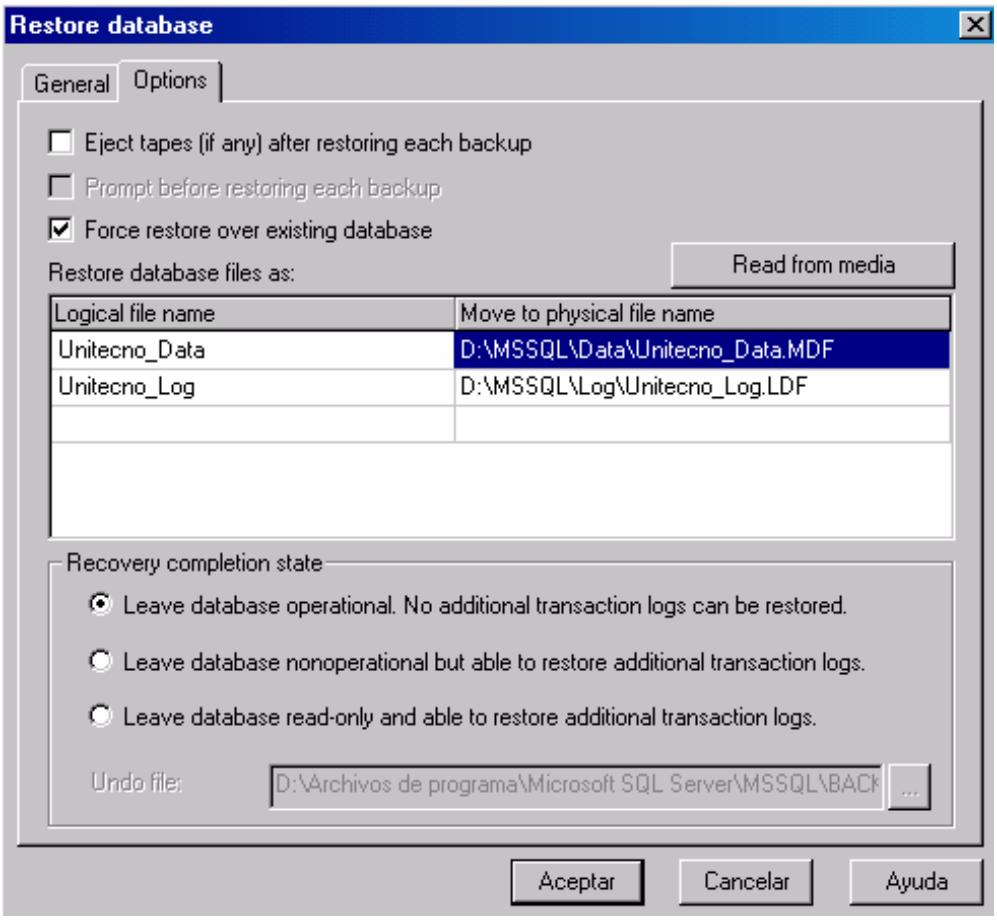
Como Restaurar la estructura de la base de datos de prueba junto con todos los datos

Desde SQL Server Enterprise, se escoje la opción Restore Database del menú Tool, con lo que se obtiene una ventana como la que se muestra a continuación:



- En la opción Restore as Database, se escoge el nombre de la base de datos en donde queremos hacer la restauración, para ello, previamente se ha debido crear una base de datos escogiendo la opción create database, haciendo clic derecho sobre la carpeta Database del SQL Server Enterprise.
- En Restore se escoge la opción From Devices, desde donde se selecciona el archivo “Unitecno backup.BAK” que va contenido en la carpeta “Copia Base Datos” en el CD desde donde se descargó este documento
- Desde el botón Select devices clic en el botón Add y allí se escoge la ruta donde se encuentra el archivo “Unitecno backup.BAK”. Aceptar

- Desde la pestaña Option de la ventana Restore Database se escoge la opción *Force restore over existing database* como se muestra en la siguiente figura:



- Aceptamos todos los términos y ya está lista la base de datos junto con toda su estructura para ser utilizada.
- Es muy importante anotar que las publicaciones hechas de esta base de datos no se copian, así que si se quiere tener la configuración para las replicas y la distribución de los datos, hay que remitirse al Capitulo 3, sesión 3.6 del documento principal.

ANEXO C. INSTALANDO LA PLATAFORMA AGLETS 2

El archivo de instalación es un archivo jar (Java Archive) que contiene la biblioteca para crear los aglets y la plataforma sobre la que se ejecutan, el nombre del jar refleja la versión de aglets que contiene: aglets-2.0.2.jar

Para iniciar la instalación se deben seguir los siguientes pasos:

1. **Descomprimir el archivo**. Para descomprimir el archivo se debe crear una carpeta donde se desee descomprimir el archivo jar, después ubicarse desde la línea de comandos en la carpeta y descomprimir el archivo ejecutando:

   jar xvf aglets-2.0.2.jar

   Durante la descompresión se verán algunas líneas desplazandose sobre la pantalla que indicando que se han extraído los archivos. Una vez terminado este proceso se habrán creado las siguientes carpetas:

   - bin. Esta carpeta contendrá los programas ejecutables de la plataforma Aglets 2 y otros archivos necesarios para completar la instalación de la plataforma.
   - cnf. Esta carpeta contiene los archivos de configuración de la plataforma.
   - public. Esta carpeta contiene algunos ejemplos de agentes y se podría utilizar como directorio base para crear los agentes.
   - lib. Esta carpeta contiene la librería de Aglets 2 (como un archivo jar) y otras librerías requeridas por la plataforma de agentes móviles.

2. **Instalación de la plataforma**. Para instalar la plataforma se necesita ejecutar Apache Ant, una herramienta creada expresamente para instalar y compilar aplicaciones Java. Aglets 2 es lanzado con una versión de Ant que es apropiada para instalar la plataforma, sin embargo es posible usar otra versión de Ant (recomendable una versión superior a la 1.5). Para obtener mayor información acerca de Ant se puede consulta el sitio web http://www.apache.org.

   Para instalar Aglets se debe pasar a la carpeta bin creada al descomprimir el archivo de instalación y ejecutar *ant* desde la línea de comandos.

3. **Configuración de políticas.** Como cualquier aplicación de Java, la plataforma Aglets entradas en el archivo de políticas (*policy*) para abrir sockets, ejecutar agentes, acceso local de archivos, entre otros. Se pueden copiar entradas del archivo *policy* de Aglets al archivo *policy* de Java o se puede dejar que Ant haga eso por usted mediante el siguiente comando:

   ant install-home

4. **Ejecución del servidor Aglets**. Una vez instalada la plataforma Aglets se puede ejecutar el servidor Aglets por defecto, que es llamado Tahiti. Tahiti se ejecuta a través del comando *agletsd*, que se encuentra en la carpeta *bin* e inicia el servidor Aglets. Por defecto se crea un usuario y contraseña para autenticarse en el servidor Tahiti que son *anonymous* y *aglets* respectivamente.

ANEXO D. CONTENIDO DEL CD

El CD contiene las siguientes carpetas:

- INSTALADOR_CLIENTE: Instalar en la máquina cliente.
- INSTALADOR_SERVIDOR: Instalar en la máquina servidor.
- COPIA BASE DATOS: Backup de la Base de Datos de Prueba.
- FUENTES_ABADAM: Programas fuentes de java (*.java).
- JAVADOCS_ABADAM: Documentación de las clases y métodos utilizados.
- ARTICULO_TESIS
- DOCUMENTO_TESIS