UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

MODELO DE REPLICACIÓN DE DATOS EN SISTEMAS DISTRIBUIDOS DE BASES DE DATOS RELACIONALES

TESIS DE GRADO

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: JOSE LUIS PÉREZ RAMOS
TUTORA: M.Sc. ROSA FLORES MORALES
REVISOR: Lic. ABDÍAS PATZI CHOQUE

LA PAZ - BOLIVIA
2006

Dedicatoria

A mis padres y hermanos por el amor y confianza que nunca me negaron.

Agradecimientos

Doy gracias a Dios por la vida, por haberme permitido culminar la tesis de grado, sin cuya ayuda nada de esto seria posible.

Quiero agradecer a mi tutora M.Sc. Rosa Flores M., cuyos consejos fueron de gran valor para el desarrollo de este trabajo, por la paciencia que tuvo y la ayuda desinteresada que me brindó.

A mi revisor Lic. Abdías Patzi, por el tiempo dedicado en las revisiones de mi trabajo, por la ayuda y confianza que depositó en mí.

A mis buenos amigos Daniel Pérez, Fernando Arzabe y a todos aquellos que tuve la dicha de conocer en la carrera universitaria.

A mi padre, que en paz descanse, a mi madre, la mujer que más quiero en este mundo, y a mis hermanos por estar siempre a mi lado apoyándome incondicionalmente en todo momento.

A todos ellos muchas gracias.

Resumen

La globalización ha definido un nuevo escenario en los negocios, obligando en muchos casos, al establecimiento de vínculos entre distintas organizaciones, donde el intercambio de información resulta de vital importancia para las mismas y suele ser un problema mucho más crítico de lo que aparenta. Una primera aproximación de solución son las telecomunicaciones que, con la gran mejora en rapidez y seguridad que han logrado, pueden resolver parcialmente el problema antes planteado. Se dice parcialmente, pues la cantidad de usuarios de estas redes de comunicación, el tipo y la cantidad de información transferida entre ellos ha crecido de manera drástica y diversificada; generando un alto tráfico en la red. Este nuevo escenario obliga a que los sistemas informáticos estén orientados a no sobre cargar las redes.

La replicación de datos, consiste pues, en el transporte de datos entre dos o más servidores de bases de datos, permitiendo que ciertos datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales. La replicación de datos se ha convertido en una alternativa al problema antes mencionado pues mantiene varias copias de la información en más de un servidor de bases de datos que conforman el sistema distribuido. Este concepto se ha venido desarrollando por los proveedores de Sistemas de Gestión de Bases de Datos y por fabricantes especializados.

El Modelo de Replicación planteado está formado por un conjunto de agentes responsables de distribuir las réplicas entre el origen y sus destinos, y a los aspectos básicos de replicación, se incorporan opciones adicionales para ajustarse de mejor manera a los requerimientos actuales de las organizaciones. Es muy importante mencionar que este modelo está orientado, en cuanto a la oportunidad y economía, al tipo de replicación de datos asíncrono.

Finalmente el modelo es plasmado en un prototipo de Sistema de Replicación de Datos en Sistemas Distribuidos de Bases de Datos Relacionales.

Contenido

Lista de figuras

Lista de tablas

CAPÍTULO 1

Introducción

1.1. INTRODUCCIÓN

La globalización ha definido un nuevo escenario en los negocios, obligando, en muchos casos, al establecimiento de vínculos entre distintas organizaciones, donde el intercambio de información resulta de vital importancia para las mismas y suele ser un problema mucho más crítico de lo que aparenta. Una primera aproximación de solución son las telecomunicaciones que, con la gran mejora en rapidez y seguridad que han logrado, resuelven parcialmente el problema antes planteado. Se dice parcialmente, pues la cantidad de usuarios en estas redes informáticas, el tipo y la cantidad de información transferida entre ellos, ha crecido de manera drástica y diversificada, generando un elevado tráfico de datos y haciéndose necesario un servicio de alta disponibilidad de las redes de comunicaciones. Este nuevo escenario obliga a que los sistemas informáticos estén orientados a no sobre cargar las redes.

Por lo descrito arriba surge como alternativa interesante la replicación de datos. La replicación de datos permite que parte de la información de una base de datos sea almacenada en más de un lugar, y su principal utilidad es que permite aumentar la disponibilidad de los datos y mejora el desempeño de las consultas globales a la base de datos. [Elmasri, 2000]

Como consecuencia de lo anterior, los fabricantes de Sistemas de Gestión de Bases de Datos Relacionales (SGBDR) han comenzado a desarrollar herramientas de replicación de datos incorporados en sus productos. La introducción de técnicas de replicación de datos en la industria de los SGBDR, requiere de métodos que garanticen un alto grado de consistencia y oportunidad en los datos replicados. Sin técnicas adecuadas para soportar este proceso, tales herramientas probablemente no serán lo suficientemente confiables y adaptables, como para ser implementadas.

En la búsqueda de técnicas y metodologías de Ingeniería del Software que puedan solventar los problemas mencionados, nos encontramos con una, que por las características que posee resulta de interés a los requerimientos de replicación de datos, se trata del paradigma Agente.

Un agente según Marvin Minsky es "un programa computacional con cierta inteligencia" [Minsky,1994], o como lo propone [Janca,1995] "una entidad software a la que se pueden delegar tareas".

En la actualidad las herramientas disponibles de replicación de datos, garantizan una total consistencia y oportunidad de los datos replicados, sólo si se tienen las condiciones óptimas en las redes de datos. Es por ello que la presente tesis se centra en el estudio, desarrollo y evaluación de una propuesta de replicación de datos entre SGBDRs basada en una arquitectura de agentes, que garantice un aceptable grado de consistencia y oportunidad en los datos replicados, en redes de comunicación de datos de bajo costo económico.

Para facilitar la lectura de la tesis, realizamos a continuación un resumen de cada capítulo. Inicialmente el capítulo 1, describe la problemática que originó el presente trabajo de investigación, establece la hipótesis y los objetivos de la tesis. Luego el capítulo 2, presenta el marco teórico sobre los conceptos de SGBDRs, replicación de datos y la tecnología de agentes software. Prosiguiendo con el capítulo 3, donde se desarrolla del modelo de replicación de datos en sistemas distribuidos de bases de datos relacionales. En el capítulo 4, se tratan las métricas de evaluación del modelo propuesto y el análisis de los resultados obtenidos. Y finalmente el capítulo 4, describe las conclusiones obtenidas al final de la investigación, y las recomendaciones para posteriores trabajos.

1.2. ANTECEDENTES

Hoy en día, los sistemas informáticos juegan un papel clave en las organizaciones. La rapidez y simplicidad de los sistemas de información permiten que los usuarios realicen sus tareas de forma más eficiente, dedicando más tiempo a otras actividades en beneficio de la

organización. Años atrás, las organizaciones tenían mercados delimitados geográficamente, sin embargo, la necesidad de crecimiento de los negocios obligó al establecimiento de nuevas sedes en diversas ciudades las cuales tenían que manejar y coordinar las operaciones con sus sedes centrales, siendo un requisito indispensable que la información entre ellas fluya de manera rápida, segura y oportuna. Por lo anterior, el intercambio de información es de vital importancia para las organizaciones y suele ser un problema mucho más crítico de lo que aparenta [Zapata, 2001].

La replicación de datos se constituye en un medio para mejorar el rendimiento de las consultas globales, mediante el tratamiento local de la información, proporciona mayor disponibilidad de los datos distribuidos y aumenta la escalabilidad[1] en el número de usuarios, distribuyendo la carga de solicitudes en servidores de bases de datos más pequeños. Por lo general los fabricantes de SGBDRs incorporan herramientas para la replicación de datos, sin embargo estas soluciones sólo satisfacen los requerimientos básicos de replicación de datos, presentando muchas restricciones y limitaciones en su configuración e implementación.

Si bien, el propósito principal es la replicación de datos, se ha podido observar, que en ambientes de Desarrollo de Sistemas, existe una creciente necesidad de contar con herramientas de replicación de especificaciones (metadatos) y código fuente (procedimientos almacenados), en las fases de desarrollo, pruebas y puesta en marcha; motivo por el cual el modelo de replicación de datos propuesto, contemplará estos requerimientos.

En el ámbito nacional, no se tiene registrado algún fabricante de soluciones para la replicación de datos entre Sistemas de Gestión de Bases de Datos Relacionales (SGBDR), tampoco se tienen registrados trabajos de investigación al respecto, al menos no publicado hasta la fecha en bibliotecas especializadas del sistema universitario local.

En el ámbito internacional, existen varios fabricantes de SGBDR que proporcionan herramientas para la replicación de datos, entre ellos están: IBM, Microsoft, Oracle, Progress

[1] Capacidad del sistema para mantener su rendimiento medio conforme aumenta el número de clientes.

Software, Informix y Sybase, entre otros. También existen herramientas de terceros para la replicación de datos, entre ellos están los siguientes:

- *Symbiator* ®. Una herramienta de propagación de datos, que enlaza y fusiona información de múltiples bases de datos relacionales que pueden residir en entornos heterogéneos, siendo además capaz de sincronizar múltiples aplicaciones en distintas plataformas.
- *Bea Systems* ®. El cual implementa una robusta solución de replicación de datos, entre una gama muy variada de plataformas de sistema operativo y SGBDR, mediante el uso de su monitor transaccional TP Tuxedo.

Sin embargo ninguno de los dos fabricantes anteriormente mencionados, dispone de la opción de replicar cambios producidos por el lenguaje de definición de datos (en inglés *Data Definition Languaje* – DDL), limitándose a replicar cambios producidos por el lenguaje de manipulación de datos (en inglés *Data Manipulation Languaje* – DML).

1.3. PLANTEAMIENTO DEL PROBLEMA

No se percibe, por parte de los fabricantes de herramientas de replicación de datos, la creciente necesidad por automatizar el mayor número de procesos de replicación, lo cual beneficiaría al administrador del entorno de replicación, destinando más tiempo a aspectos estratégicos que operativos. Pareciera que no se está tomando en cuenta la alternativa de la técnica de replicación de datos, como una solución a tres problemas computacionales: rendimiento, disponibilidad y escalabilidad. Esta situación puede frustrar, a organizaciones limitadas por el hardware, la posibilidad de expandir y diversificar sus servicios a una sociedad con múltiples y dinámicos requerimientos. Todo ello puede deberse al no contar con técnicas y métodos que garanticen una aceptable consistencia y oportunidad en los datos replicados, trabajando en un entorno tolerante a fallos[2], donde el costo económico de la red de comunicación es relativamente bajo.

[2] Sistemas distribuidos que eventualmente pueden trabajar sin conexión.

1.3.1. Problema general

Los actuales sistemas de replicación de datos que trabajan en entornos tolerantes a fallos, no proporcionan un grado aceptable de consistencia y oportunidad de los datos replicados. Y en cuanto a los sistemas distribuidos de bases de datos relacionales (SDBDR), éstos muestran bajo rendimiento general del sistema y baja disponibilidad de datos globales.

1.3.2. Problemas específicos

Existen múltiples deficiencias en los tradicionales SDBDR y en las actuales herramientas de replicación de datos, entre los más importantes se mencionan los siguientes:

a) En los SDBDRs las consultas globales[3], por lo general tienen un bajo desempeño, y no existe disponibilidad de los datos globales cuando la red de comunicaciones presenta fallas.

b) En SGBDRs centralizados, el número de usuarios concurrentes[4] esta limitado por el hardware del servidor de base de datos.

c) En ambientes de replicación de datos, la distribución de réplicas en línea (*on line*) acarrea dos problemas importantes: *Económico*, demanda un gran ancho de banda[5] y una alta disponibilidad de la red de comunicaciones; *Operativo*, un corte no programado de la red de comunicaciones, impedirá la operabilidad de todo el sistema.

d) En ambientes de replicación de datos tolerantes a fallos, la no propagación de réplicas en línea puede causar inconsistencias permanentes entre las bases de datos involucradas en el proceso de replicación.

e) En muchos entornos de replicación, es preciso seleccionar los datos a replicar, en función al origen de los cambios, como ser: sucesión de eventos, cumplimiento de condiciones, roles autorizados, perfiles de usuarios, etc.

[3] Consultas sobre tablas de bases de datos remotas (distantes geográficamente).
[4] Conexiones simultáneas de usuarios a un sistema de base de datos.
[5] Capacidad o velocidad de transmisión de datos en redes de comunicación.

f) Si bien las herramientas actuales de replicación de datos permiten programar horarios de replicación, éstas no tienen la capacidad de recomendar horarios adecuados de replicación, en función a la carga de tráfico en la red o al historial de replicaciones.

g) Para ambientes de desarrollo de sistemas, es preciso que además de datos, se puedan replicar definiciones de objetos y procedimientos almacenados de la base de datos.

Sin dejar de lado los problemas específicos arriba descritos, creemos que es preciso hacer hincapié en los problemas de los incisos d) y f), por tratarse de la consistencia y oportunidad de los datos replicados, que serán encarados con mayor rigurosidad.

1.4. HIPÓTESIS

"*El Modelo de Replicación de Datos (MRD) en Sistemas Distribuidos de Bases de Datos Relacionales, basado en una arquitectura de agentes inteligentes, permitirá mantener un alto grado de consistencia y oportunidad en los datos replicados entre Bases de Datos Relacionales bajo condiciones tolerantes a fallos*".

1.5. OBJETIVOS

1.5.1. Objetivo general

Desarrollar un Modelo de Replicación de Datos en Sistemas Distribuidos de Bases de Datos Relacionales (MRD) que contribuya a una solución integral y económica en problemas de inconsistencia y falta de oportunidad de los datos replicados en entornos tolerantes a fallos, además de mejorar en el rendimiento de consultas globales, la disponibilidad de los datos y la escalabilidad del número de usuarios en los sistemas de bases de datos.

1.5.2. Objetivos específicos

a) Diseñar un agente de replicación, que en base a determinadas reglas y al conocimiento que adquiera en el transcurso del tiempo, tenga las siguientes capacidades:
   - Replicar datos, metadatos, y código fuente (procedimientos almacenados).
   - Replicar en modalidad fuera de línea (offline), también conocido como replicación asíncrona.
   - Permita los tipos de replicación en simple y doble sentido, es decir replicación asimétrica y simétrica.
   - Recomendar al usuario final los mejores horarios y/o intervalos de replicación.

b) Diseñar un agente de conflictos que se encargue de minimizar las posibles inconsistencias de replicación, asegurando que tales inconsistencias sean solamente temporales; esto se logrará en base a reglas pre-establecidas de resolución de conflictos de replicación.

c) Diseñar un agente asistente del usuario final, que colabore en tareas de:
   - Creación del entorno de replicación.
   - Mantenimiento de sitios de replicación y elementos a replicar.
   - Programación de horarios de replicación.
   - Clasificación de datos a replicar.
   - Encolamiento de transacciones y definiciones.

1.6. JUSTIFICACIÓN

El tipo de replicación asíncrono[6] hace posible disminuir el tráfico en la red y permite la tolerancia a fallos. Un ejemplo de ello es el siguiente: dado un sistema de bases de datos replicadas asincrónicamente, la falla de una de ellas provocará la no operabilidad de las aplicaciones conectadas a ésta, sin embargo las demás aplicaciones conectadas a las restantes bases de datos podrán seguir operando, durante cierto tiempo, sin problema alguno.

[6] Replicación a intervalos de tiempo, donde se presenta bastante inconsistencia en los datos replicados.

Definitivamente la selección de técnicas para el manejo de datos distribuidos, depende de las necesidades de la organización. Sin embargo, la replicación asincrónica de datos introduce una serie de ventajas, como ser: distribuir la carga de trabajo en múltiples servidores de bases de datos, elevar la disponibilidad de los datos, optimizar el tiempo de las transacciones al separar los sistemas transaccionales de los sistemas de decisión, reducir el tráfico de red y dar soporte optimizado para cambios de los modelos organizacionales [Osborne, 1999].

Por lo anterior se identifican tres importantes factores computacionales a los cuales beneficia la replicación asíncrona, que son: rendimiento de las transacciones globales, disponibilidad de los datos y escalabilidad de usuarios, que de manera general están presentes en las tres justificaciones siguientes.

1.6.1. Económica

En la actualidad, los sistemas informáticos juegan un papel clave en las organizaciones. La rapidez y simplicidad de los sistemas de información permiten que los usuarios realicen sus tareas de forma más eficiente, dedicando más tiempo a otras actividades en beneficio de la organización. Años atrás, las organizaciones tenían mercados delimitados geográficamente, sin embargo, la necesidad de crecimiento de los negocios obligó al establecimiento de nuevas sedes en diversas ciudades, las cuales tienen que manejar y coordinar las operaciones con sus sedes centrales, siendo un requisito indispensable que la información entre ellas fluya de manera rápida, segura y oportuna.

1.6.2. Social

Mediante la globalización, se ha definido un nuevo escenario en los negocios, obligando, en muchos casos, al establecimiento de vínculos entre organizaciones de distintos rubros y la apertura de sedes en nuevas ciudades. En estos casos, el intercambio de información resulta de vital importancia para las organizaciones y suele ser un problema mucho más crítico de lo que se piensa.

1.6.3. Técnica

Las redes de telecomunicaciones han logrado mejorar en rapidez y seguridad los problemas de vinculación entre las organizaciones. Pero esta solución es parcial, pues la cantidad de usuarios en estas redes, el tipo y la cantidad de información transferida entre ellos a crecido de manera drástica y diversificada, generando un alto tráfico en la red. Este nuevo escenario obliga a que los sistemas informáticos estén orientados a no sobre cargar las redes.

Por lo tanto un modelo de replicación asíncrono, desde ya, mejora los problemas de vinculación planteados, puesto que además de mejorar en rendimiento y disponibilidad, requiere un menor ancho de banda en la red de comunicaciones, que con un modelo de replicación en línea (síncrono).

1.7. METODOLOGÍA

El desarrollo del presente trabajo de investigación se lo llevó adelante en orden de las siguientes actividades:

- La etapa inicial de la investigación, comenzó con un intenso relevamiento de información referente a las herramientas de replicación de datos disponibles en el mercado actual.
- Una vez revisada la información básica y avanzada de los conceptos de replicación de datos, se procedió a programar y realizar visitas a direcciones de sistemas de algunas instituciones y empresas locales, que tienen planificada la expansión de sus centros de cómputo en otras ciudades de nuestro país. Estas fueron: el Servicio de Impuestos Nacionales (SIN), la Empresa Nacional de Telecomunicaciones (ENTEL S.A.) y el Registro Único para la Administración Tributaria Municipal (RUAT).
- Con la información obtenida y el propósito de medir la verdadera magnitud del problema, se hizo un previo análisis de los datos, en donde se pudo establecer los reales requerimientos de la replicación de datos.

- Teniendo claro el problema, se plantea un modelo de replicación de datos, donde se hace una introducción a la problemática, hipótesis y objetivos que persigue el trabajo de investigación.
- Se establece el correspondiente marco teórico, revisando el estado del arte de la Ingeniería del Software, sobre conceptos de bases de datos, replicación de datos, agentes inteligentes y metodologías orientadas a agentes.
- Una vez obtenidos los requerimientos y revisados los conceptos, se procede al desarrollo del marco aplicativo, identificándose la estructura del modelo y diseñándolo con la metodología MAS-CommonKADS [Iglesias,1998].
- Por último, el modelo resultante es sometido a una evaluación, tomándose en cuenta factores inherentes a la replicación de datos. Realizándolo en tres escenarios: el óptimo, el esperado y el pésimo.

1.8. ALCANCE Y LÍMITES

- El MRD dará soporte única y exclusivamente al tipo de replicación asíncrona, no pudiendo implementarse el tipo de replicación en línea.
- El MRD, además de replicar datos, permitirá la replicación de definiciones de objetos de la base de datos, es decir que, se podrán crear y/o modificar estructuras de objetos remotos[7], como ser tablas, índices, vistas, procedimientos almacenados, etc.
- El MRD sólo podrá ser implementado en SGBDRs dinámicos, es decir que tengan soporte para el manejo de disparadores[8] y proporcionen enlaces[9] hacia otros SGBDRs, formando así un sistema distribuido de bases de datos relacionales.
- Si el requerimiento de replicación es, que el grado de inconsistencia de los datos replicados sea nula, es decir que la consistencia sea total, por su naturaleza asíncrona no se recomienda el uso del MRD.

[7] Objetos de una base de datos geográficamente distante a la original.
[8] También llamados triggers, son procedimientos almacenados ejecutados por eventos sobre tablas.
[9] También llamados database links, son enlaces entre sistemas de bases de datos remotos.

1.9. APORTES

- Mostrar e incentivar el uso de la técnica de la replicación de datos entre SGBDRs, como una alternativa de solución a tres problemas computacionales: rendimiento de los sistemas distribuidos, disponibilidad de los datos y escalabilidad de usuarios.
- Para entornos tolerantes a fallos, brindar un modelo de replicación asíncrono que reduzca al mínimo la inconsistencia y mejore la oportunidad de los datos replicados.
- Mostrar las ventajas del uso de agentes inteligentes en aplicaciones de computación distribuida.

CAPÍTULO 2

Marco Teórico

2.1. INTRODUCCIÓN

La conceptualización de todo fenómeno real o abstracto, debe tomar en cuenta distintas teorías que lo respalden. Por ello, el presente capítulo describe los lineamientos teóricos bajo los cuales se desarrollará el trabajo de investigación. Se inicia describiendo los distintos tipos de modelos que pueden ser diseñados bajo un determinado sistema de clasificación. También se describen a los sistemas distribuidos y a los sistemas de gestión de bases de datos relacionales. Posteriormente se entra en detalle sobre la replicación de datos mostrando sus características, funcionalidades, ventajas y desventajas de su uso. Y para finalizar se muestran las características de los agentes inteligentes y las metodologías disponibles para el análisis y diseño de sistemas basados en agentes.

2.2. MODELO

2.2.1. Concepto

Un modelo fundamentalmente es algo que obtenemos después de un proceso de abstracción, es decir tomamos un sistema real y hacemos una imagen de él, más simple y más clara que el original. Al construir un modelo tratamos de captar lo que es esencial en el sistema, lo que a nosotros nos interesa estudiar y lo que pensamos que nos servirá para ese estudio, todo lo demás lo desechamos.

Un modelo facilita la comprensión de un sistema complejo, representando lo que es significativo para nuestro estudio, es una imitación de la realidad. Así, tenemos el objeto real, el sujeto que lo estudia y el modelo, que tiene relaciones de analogía o similitud con el objeto real y permite al sujeto obtener conclusiones relativas al sistema.

2.2.2. Clasificación

Haciendo una revisión de los distintos tipos de modelos, es importante mencionar a los siguientes:

- *Modelos de afirmación.* Estos modelos describen al sistema usando palabras, se usan en los sistemas más complejos donde no es factible determinar relaciones matemáticas. Estos modelos son muy débilmente predictivos y se limitan a hacer una descripción verbal y cualitativa del sistema. Este tipo de modelos son muy usados en sistemas administrativos.
- *Modelos físicos.* Son objetos materiales usados para demostración y, en menor medida, para experimentación cuantitativa. Un ejemplo típico son las maquetas.
- *Modelos gráficos.* Son modelos ideales que usan medios de expresión gráfica, un ejemplo muy característico son los mapas y planos.
- *Modelos formales.* Son los modelos abstractos y matemáticos ampliamente usados en la investigación científica. Consideran los parámetros y variables esenciales de un fenómeno y sus relaciones descritas en forma de ecuaciones matemáticas.
- *Otros.* La anterior es solo una de las muchas posibles clasificaciones de los modelos, existen otras que, por ejemplo consideran las relaciones de analogía (modelos funcionales, estructurales, de comportamiento, etc.), o bien la finalidad de uso del modelo (de demostración, experimentales, para toma de decisiones, etc.).

La siguiente lista muestra las características principales que debe tener todo modelo:

- ✓ Confiabilidad
- ✓ Sencillez
- ✓ Bajo costo de desarrollo y operación
- ✓ Manejabilidad
- ✓ De fácil entendimiento, tanto el modelo como los resultados
- ✓ La relación costo-beneficio debe ser positiva.

Una vez descrita la clasificación de modelos y observando el ambiente de desarrollo de la investigación, el modelo a aplicar en el presente trabajo será de tipo gráfico.

2.2.3. Construcción de modelos

Ahora veremos específicamente como se construye un modelo, de antemano aclaremos que por lo general no se trata de un proceso sencillo y que lo que viene a continuación son sólo algunas orientaciones sobre las técnicas básicas de modelamiento:

- *Ordenar las opiniones.* Para modelar, primero, se debe observar el sistema o fenómeno y recoger información relevante para nuestro propósito, luego se determina sobre qué base será construido el modelo según las relaciones de analogía que se observen. También en esta etapa se determinará el objetivo con el que será construido el modelo.
- *Elaborar los elementos esenciales y sus acoplamientos.* El modelo se va conformando de acuerdo a las relaciones de analogía encontradas.
- *Experimentar con modelos.* Se trata de buscar modelos alternativos o variantes del configurado originalmente para ver si se puede perfeccionar la similitud con el comportamiento relevante del modelo real.
- *Decidir la solución óptima.* De todos los modelos experimentados se escoge al que represente al sistema de la mejor manera para nuestros propósitos.

2.2.4. Prueba del modelo

Se deben diseñar y ejecutar pruebas que confronten la capacidad predictiva del modelo con respuestas conocidas del sistema, de manera de detectar si hay omisiones o errores relevantes. Para ello existen, al menos, tres técnicas muy usadas para modelar y que no se las puede dejar de mencionar:

- *Método de conclusiones por analogías.* Para obtener conclusiones por analogías, se debe buscar fenómenos semejantes cuya solución sea conocida, comparar sistemas distintos buscando semejanzas o analogías en su comportamiento, su estructura o su materialidad.
- *Método de la caja negra.* Para sistemas muy complejos un buen método es el de la caja negra, que consiste en un sistema (en principio definible) al que solo podemos

influenciar alimentándolo y observando sus reacciones. Así podemos definir un comportamiento "macro" sin entrar a los detalles internos del sistema.

- *Método de las aproximaciones sucesivas.* Este método consiste en definir un resultado óptimo y tratar de obtenerlo ingresando magnitudes al azar al sistema, por medio de la prueba y error nos acercaremos al óptimo esperado, lo que permitirá encontrar la relación buscada sobre el comportamiento del sistema.

2.3. BASES DE DATOS

2.3.1. Definición

Una Base de Datos (BD) es una colección de archivos interrelacionados que son creados con un Sistema de Gestión de Base de Datos (SGBD). Los tres componentes principales de un sistema de base de datos son el hardware, el software SGBD y los datos a manejar, así como el personal encargado del manejo del sistema.

2.3.2. Bases de datos relacionales

Una base de datos relacional es un modelo de administración de datos para modelar problemas reales. En éste la idea fundamental es el uso de *tablas*, compuestas de *registros* (filas de una tabla) y *campos* (columnas de una tabla); en donde el lugar y la forma en que se almacenen los datos no tienen relevancia. La información puede ser recuperada o almacenada por medio de *consultas* que ofrecen una amplia flexibilidad y poder para administrar la información. El lenguaje más común para construir consultas sobre bases de datos relacionales es el Lenguaje de Consultas Estructurado (en inglés *Structured Query Language* – SQL) un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

El SQL es un lenguaje estándar utilizado para crear, modificar, mantener y consultar una base de datos relacional. Las sentencias SQL pueden ser clasificadas en dos grupos:

- *DDL (Data Definition Language - Lenguaje de Definición de Datos).* Las sentencias DDL son aquellas utilizadas para la creación de una base de datos y todos sus objetos: tablas, índices, relaciones, disparadores (triggers), procedimientos almacenados, etc.
- *DML (Data Manipulation Language - Lenguaje de Manipulación de Datos).* Las sentencias DML son aquellas utilizadas para insertar, borrar, modificar y consultar los datos de una base de datos.

2.3.3. Sistema de gestión de base de datos

Un SGBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de una tarea específica. El objetivo primordial de un sistema manejador base de datos es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la base de datos. Un Sistema de Gestión de Base de Datos Relacional (SGBDR) es un tipo particular de un SGBD, en donde el lenguaje utilizado para recuperar los datos es el SQL.

2.3.4. Sistemas distribuidos de bases de datos

Las transacciones en un Sistema Distribuido de Bases de Datos deben conservar cuatro características muy importantes: ACID [Attaluri,1995].

- *Atomicidad*: para que una transacción se consuma debe haberse realizado por completo ("todo o nada").
- *Consistencia*: la base de datos se mantiene consistente antes y después de la transacción.
- *Aislamiento*: referente al nivel de exposición que tiene las modificaciones de los datos en una transacción para las demás transacciones.
- *Durabilidad*: permanencia de las modificaciones hechas por una transacción en la base de datos.

En [Couloris, 2001] se señalan algunos de los problemas que se pueden tener con las transacciones distribuidas, los cuales son: actualizaciones perdidas, recuperaciones inconsistentes, operaciones conflictivas, lecturas sucias, recuperación de transacciones, abortos en cascada, escrituras prematuras, etc.

2.4. REPLICACIÓN DE DATOS

2.4.1. Concepto

La replicación es el proceso de copiar y mantener objetos de una base de datos en diferentes bases de datos, que constituyen un sistema de bases de datos distribuido. Se capturan cambios aplicados a una base de datos y luego se aplican a cada uno de los demás de acuerdo a ciertas reglas definidas de replicación [Oracle,1999]. El objetivo de la replicación de datos es que las operaciones de las aplicaciones clientes sobre los datos replicados se realicen de forma consistente y con un tiempo de respuesta satisfactorio. La consistencia no se logra algunas veces, debido a que las réplicas de un mismo dato no son necesariamente idénticas, al menos no en cada instante de tiempo particular. Algunas réplicas pueden haber recibido actualizaciones que otras no hayan recibido.

Estrictamente hablando, existe una diferencia elemental entre la replicación y la distribución de datos, en la primera las transacciones globales[1] son realizadas localmente, sin la intervención de las restantes bases de datos, pero en la segunda se usa el protocolo *Two Phase Commit*[2] para cerrar una transacción que involucra a más de una base de datos.

2.4.2. Funcionalidades

Según [Buretta,1997], un servicio de replicación de datos debe proveer las siguientes funcionalidades:

[1] Transacciones que involucran a un conjunto de bases de datos.
[2] Mecanismo que consuma una transacción en línea en más de una bases de datos.

- Debe ser escalable: poder replicar volúmenes pequeños o grandes de datos a través de recursos heterogéneos.
- Deben soportar replicación no solamente de datos, sino también de otros tipos de objetos, por Ej. Procedimientos almacenados de BD.
- Deben soportar replicación sincrónica (tiempo real) y asincrónica (tiempo no real).
- Debe prever un mecanismo de seguridad. Los datos pueden cambiar la forma, pero no el contenido.
- Deben proveer un mecanismo de log (respaldo) que permita recuperar el sistema ante fallas.
- Debe soportar un mecanismo de recuperación automática.
- Debe soportar una Interfase de GUI o alguna herramienta de ambiente amigable de monitoreo.

Aparte de datos necesarios para la administración es interesante que las replicas sepan la medida de latencia[3] con respecto a la fuente primaria.

2.4.3. Tipos de replicación

Hay diversas formas de clasificar los tipos de replicación, en el presente documento se han recopilado dos de ellas, una teniendo en cuenta el tiempo de latencia y otra teniendo en cuenta el sentido del viaje de las réplicas.

2.4.3.1. Tiempo de latencia

- **Replicación sincrónica**

La replicación sincrónica es llamada de "consistencia fuerte" entre los datos almacenados, donde el grado de desactualización de los datos replicados respecto a los datos originales es nula, es decir la latencia es cero. Este tipo de transacciones son globales, también llamadas

[3] La medida de latencia es la cantidad de tiempo que una réplica puede estar inconsistente hasta llegar a estar consistente con la fuente primaria designada [Buretta,1997].

"todo o nada", en éstas una transacción no debería ser aceptada si alguno de los sitios intervinientes no están de acuerdo.

Estas transacciones respetan las propiedades ACID, son atómicas, consistentes, aisladas y los cambios son durables. Un ejemplo del tipo de replicación sincrónica se muestra en la figura 2.1.

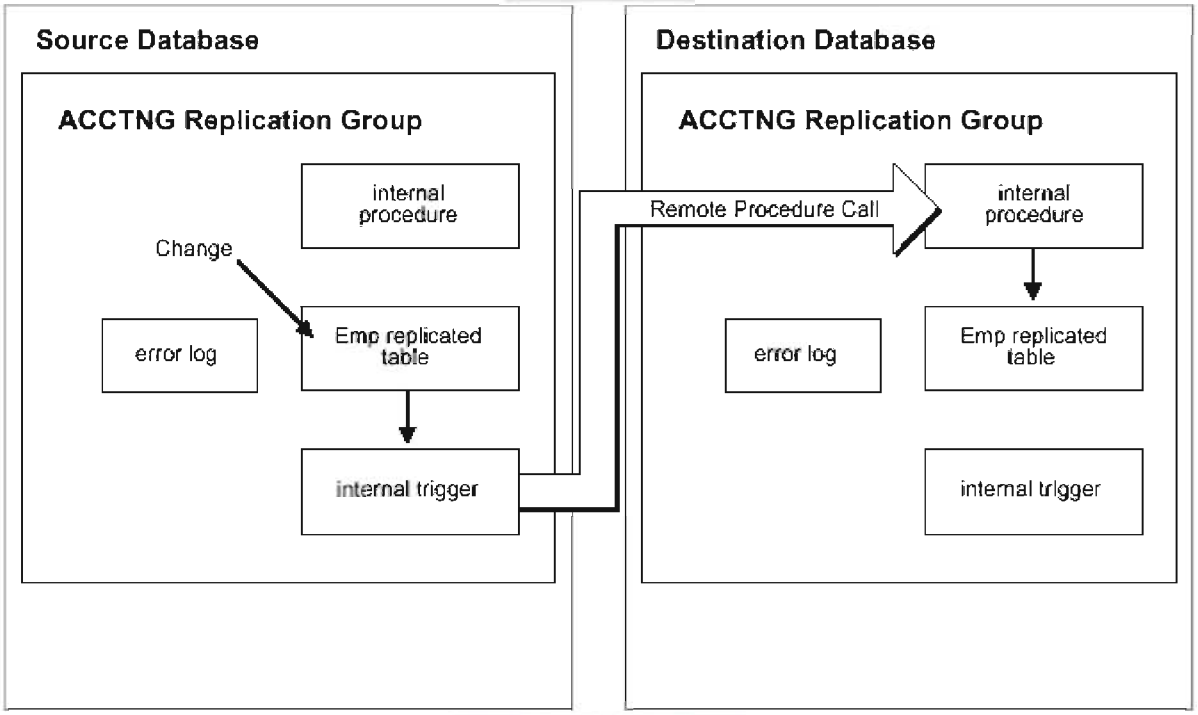


Figura 2.1. Mecanismo de replicación sincrónica de datos [Oracle,1999].

Se debe tener en cuenta en este tipo de replicación temas críticos de diseño y de infraestructura:

- Asegurar un buen sistema de recuperación que garantice la consistencia.
- Asegurar una infraestructura robusta y eficiente: un ambiente de red con un apropiado ancho de banda y con servicio de alta disponibilidad.
- El número de recursos manejado debe quedar en un número razonable dentro de los límites técnicos.

- **Replicación asincrónica**

La replicación de datos asincrónica es llamada "consistencia débil", entre los datos almacenados, en esta existe una desactualización de la copia replicada respecto de la original, es decir existe una cierta latencia, generalmente la latencia se mide en segundos si la infraestructura apropiada de recursos es suficiente. Un ejemplo del tipo de replicación asincrónica se muestra en la figura 2.2.

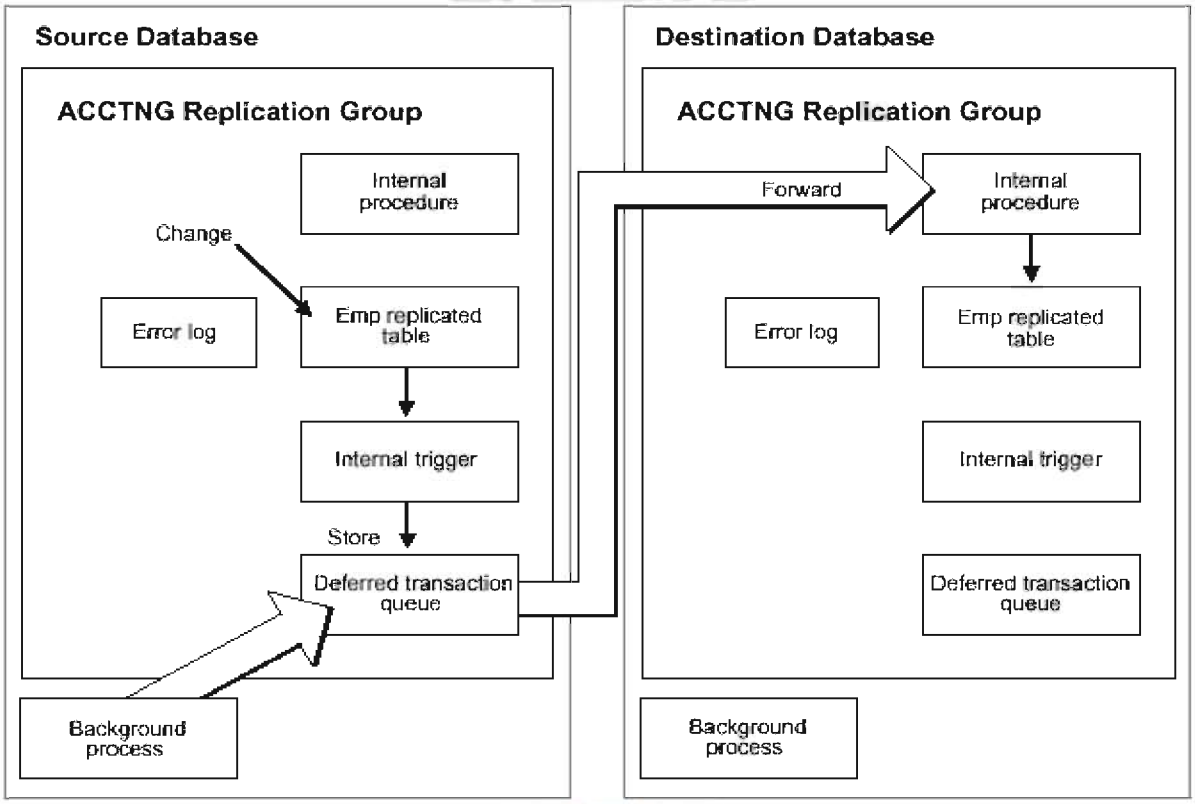


Figura 2.2. Mecanismo de replicación asincrónica de datos [Oracle,1999].

Consideraciones a tener en cuenta cuando se usa replicación asincrónica. Los temas críticos están más cerca de la arquitectura que de la infraestructura física, y las consideraciones para simplificar el diseño y la implementación son:

- Definir requerimientos para el soporte de semántica transaccional y/o integridad transaccional.
- Asegurar el apropiado nivel de integridad secuencial.
- Usar mecanismos de resolución y detección de conflictos.

¿En qué momento realizar la replicación asincrónica?. El sistema de replicación podría proveer varias alternativas para elegir el momento de replicar pudiendo ser las siguientes:

- Inmediatamente como sea posible.
- Programado.
- Cuando se disparen los triggers[4] por operaciones de inserción (*insert*), actualización (*update*) y eliminación (*delete*).
- Bajo un control manual.

2.4.3.2. Sentido de la réplica

- **Replicación asimétrica**

En un solo sentido (unidireccional), se tienen un sitio[5] actualizable y otro que contiene una copia del principal, pero este último será solo de lectura (ver figura 2.3).

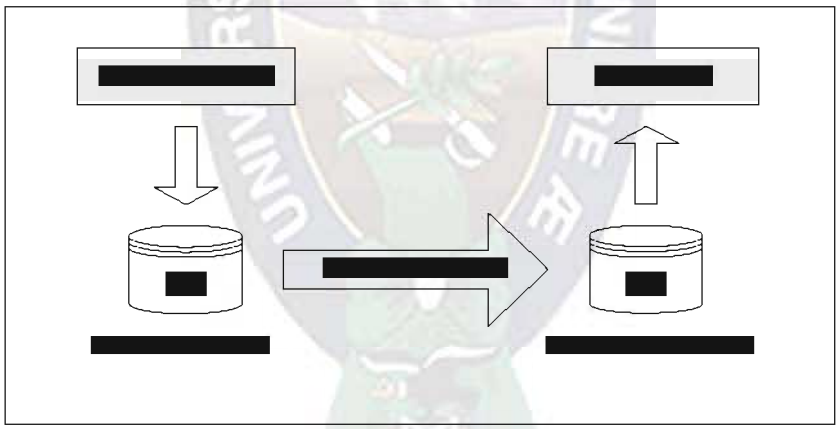


Figura 2.3. Tipo de replicación asimétrica (unidireccional) [Oracle,1999].

- **Replicación simétrica**

En ambos sentidos (bidireccional), se tienen varios sitios actualizando simultáneamente los datos, de tal forma que la replicación se produce en ambas direcciones (ver figura 2.4).

[4] Trigger: Procedimiento almacenado que es ejecutado automáticamente ante cierto evento de una tabla.
[5] SGBDR integrante del sistema o entorno de replicación.

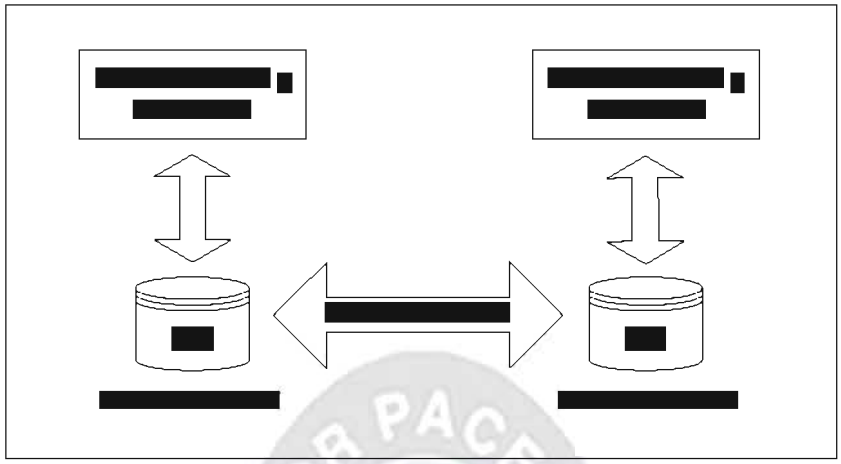


Figura 2.4. Tipo de replicación bidireccional [Oracle,1999].

2.4.4. Conflictos de Replicación

Al permitir la actualización de la información desde cualquier sitio de replicación en el sistema de bases de datos replicadas se debe buscar la convergencia de la información, existe la posibilidad de generación de conflictos. Estos suceden cuando una misma información es modificada en varios sitios dentro de un intervalo de tiempo que contiene al intervalo de replicación, y cuando la replicación es asincrónica [Cullas,2001].

Según [Oracle,1999] los tipos de conflictos que pueden ocurrir en un sistema de replicación asíncrono son:

- *Conflicto de pérdida de unicidad*: ocurre cuando la replicación de un registro viola una integridad (de llave primaria o unicidad).
- *Conflicto de actualización:* ocurre cuando una fila está siendo actualizada por dos transacciones diferentes simultáneamente.
- *Conflicto de eliminación:* ocurre cuando con una transacción elimina un registro mientras que otra transacción lo está actualizando o eliminando.

Para resolver los conflictos aparecen lo que se ha denominado métodos de resolución de conflictos, como [Oracle,1999]:

- *Ignorar*: en el momento del conflicto, ignora el error aunque no se haya actualizado.

- *Aplicar:* en el momento del conflicto, aplica los cambios basándose en la llave de la tabla involucrada.
- *Suma y promedio:* en el momento de conflicto utiliza una fórmula para obtener el valor a actualizar. Sólo es aplicable para un grupo de columnas formadas por campos numéricos.
- *Mayor:* en el momento del conflicto, inserta al final el valor que sea mayor.
- *Menor:* en el momento del conflicto, inserta al final el valor que sea menor.
- *Prioridad:* en el momento del conflicto, inserta al final los valores de las columnas de mayor prioridad.
- *Función:* en el momento del conflicto, define la lógica de resolución de conflictos con un programa complejo.

2.4.5. Ventajas y desventajas de la replicación

Las ventajas que se obtiene con la replicación son las siguientes [Buretta,1997]:

- Disponibilidad, si uno de los sitios que intervienen en una transacción falla, entonces la transacción puede ser resuelta por otro sitio que contenga la información.
- Incremento de paralelismo, en el caso de operaciones de solo lectura, se pueden procesar varias consultas simultáneamente, minimizando el acceso de datos entre los sitios.
- Se reduce considerablemente el tráfico y contención[6] en aplicaciones para el soporte de decisiones en un servidor[7] de base de datos local.

Las desventajas que se tienen con la replicación son las siguientes [Buretta,1997]:

- Costo incrementado sobre actualizaciones, el sistema debe asegurar que todas las réplicas sean consistentes.
- La implementación de un sistema con replicación trae aparejado un aumento del costo de administración.
- El costo del diseño se incrementa en los ambientes distribuidos con o sin replicación.

[6] Cuando varias solicitudes intentan utilizar simultáneamente el mismo recurso de la BD (locks).
[7] Host donde se encuentra la base de datos.

2.5. AGENTES INTELIGENTES

2.5.1. Agentes

Según lo proponen Rusell y Norving "un agente es cualquier cosa que puede ser vista como algo que percibe su ambiente a través de sensores y actúa sobre el mismo a través de efectores" [Rusell,1995].

Al considerar a los *agentes humanos*, se observa que éstos poseen sentidos que les sirven de sensores y que responden o actúan en tal ambiente mediante efectores. Por otro lado se tienen a los *agentes robóticos*, donde los sensores podrían ser cámaras de video, y los efectores, elementos mecánicos. Y un caso particular es el *agente software*, el cual es un programa de computación que se ejecuta en un ambiente, y que es capaz de realizar acciones dentro de este, con la finalidad de alcanzar objetivos particulares para el cual fue diseñado.

2.5.2. Conocimiento

La base sobre la cual se sustenta la inteligencia es el conocimiento. Definir el vocablo conocimiento no es tarea fácil dada la complejidad del mismo. A continuación se detallan algunas definiciones consideradas importantes:

1. Conocimiento es información que ha sido categorizada, interpretada, aplicada y revisada.
2. El conocimiento es una simple codificación de hechos, que incluye además la habilidad de utilizarlos en interacción con el mundo.
3. El conocimiento de algo, es la habilidad para construir un modelo mental que represente en forma precisa el objeto y las acciones que con él mismo y sobre él de pueden efectuar.

Una definición de conocimiento más completa [McGraw,1989] que pretende englobar las anteriores, considera al conocimiento como un conjunto de descripciones, relaciones y procedimientos, tales como:

- *Descripciones simbólicas de conceptos.* Un concepto es un símbolo que representa características de objetos o eventos de un dominio de aplicaciones. Estas características se definen tomando como base las relaciones compartidas por objetos, eventos o ideas diferentes
- *Descripciones simbólicas de relaciones.* A partir del establecimiento de la librería de conceptos, es posible organizar ésta jerárquicamente y definir relaciones entre los conceptos teniendo en cuenta las características, atributos o funciones que determinen asociaciones entre ellos.
- *Procedimientos para manipular ambos tipos de descripciones.* El adecuado uso y manipulación de los conceptos y relaciones establecidas se llevará a cabo por un mecanismo o procedimiento apropiado.

2.5.3. Definición de agente inteligente

Los agentes software son probablemente, el área de la tecnología de la información que crece de manera más rápida, a pesar de esto, aún no existe un acuerdo sobre qué es exactamente un agente inteligente. Sin embargo, luego de la revisión de varias de las definiciones existentes, a nuestro criterio, la que puede ser considerada como la más clara y completa, es la propuesta por Wooldridge y Jennings [Woold,1995]. Y es esta la definición que se adoptará como base para todo el trabajo posterior:

"Existen dos nociones de agente: una débil y otra fuerte.

- **Definición débil:** ...un sistema computacional hardware o software que goza de las siguientes propiedades:
  - *autonomía:* los agentes operan sin una directa intervención de humanos u otros, y tienen cierto grado de control sobre sus acciones y su estado interno.
  - *habilidad social:* los agentes interactúan con otros agentes (y posiblemente con humanos) vía algún tipo de lenguaje de comunicación entre agentes.
  - *reactividad:* los agentes perciben su ambiente, (que puede ser el mundo fisico, un usuario vía una interfaz gráfica, una colección de otros agentes, la red Internet, o tal

vez todos estos combinados), y responden de una manera rápida a cambios que ocurren en él.

- *pro-actividad:* los agentes no actúan simplemente en respuesta a su ambiente, son capaces de exhibir comportamiento oportunista, dirigido por objetivos, tomando iniciativas cuando sea apropiado.

- **Definición fuerte:** un agente, además de las características anteriores tiene una o más de las siguientes características:
  - *nociones mentales:* un agente tiene creencias, deseos e intenciones.
  - *racionalidad:* un agente realiza acciones a fin de lograr objetivos.
  - *veracidad:* un agente no es capaz de comunicar información falsa de propósito.
  - *adaptabilidad o aprendizaje*" [Woold,1995].

2.5.4. Arquitectura de agentes

Antes de definir lo que es una arquitectura de agentes, es conveniente recordar la definición de arquitectura, donde según la Ingeniería del Software:
" ... definimos un diseño del software como una descomposición del sistema en módulos – descripción de qué hace cada módulo y la relación entre estos módulos. Tal descripción es a menudo llamada la arquitectura del software o la estructura del software" [Ghezzi,1991].
Por tanto la arquitectura de un sistema, comprende:

1. División del sistema en módulos y descripción de cada uno de ellos, y
2. Relación de los distintos módulos componentes entre sí.

¿Qué es una arquitectura de agentes?
"Una metodología particular para construir agentes. Especifica cómo... el agente puede ser descompuesto en un conjunto de módulos componentes y cómo estos módulos pueden interactuar. El conjunto total de módulos y sus interacciones deben proveer una respuesta a la pregunta de cómo el dato monitoreado y el estado interno del agente determinan las acciones... y estados internos futuros." [Maes,1991].

2.6. METODOLOGÍAS ORIENTADAS A AGENTES

2.6.1. Definición

Una metodología puede definirse, en un sentido amplio, como un conjunto de métodos y técnicas que ayudan en el desarrollo de un producto software, tal como lo señala Rumbaugh: "Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convenciones en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso... Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo" [Rumbau,1991].

2.6.2. Metodologías orientadas a agentes

Luego de una revisión del estado del arte de las metodologías orientadas a agentes, podemos concluir que existen dos enfoques principales que siguen las mismas:

1. Extensión de metodologías orientadas a objetos, a fin de incluir aspectos relevantes de los objetos.
2. Extensión de metodologías de ingeniería del conocimiento (*Knowledge Engineering* - KE), tratando de aprovechar las técnicas de la KE para modelar características cognitivas[8] de los agentes.

2.6.2.1. Extensión de metodologías orientadas a objetos

Se pueden citar varias razones que justifican la extensión de las metodologías orientadas a objetos. Entre estas razones tenemos:

- Existen similitudes entre el paradigma[9] orientado a objetos y el paradigma orientado a agentes. Como establece [Shoham,1993], los agentes pueden ser considerados objetos activos, objetos con estados mentales.

[8] Relativo al conocimiento, o la capacidad de conocer.
[9] Un modelo que sirve de ejemplo o es ejemplar.

- El uso común de los lenguajes orientados a objetos para la implementación de los agentes.
- La popularidad de las metodologías orientadas a objetos.

- **Análisis y diseño orientado a agentes de Burmeister**

[Burmeis,1996] define tres modelos para analizar un sistema agente: el modelo de agente, que se limita al agente y a su estructura interna (creencias, planes, objetivos); el modelo organizacional, que describe el relacionamiento entre agentes (herencia y roles en la organización); y el modelo de cooperación, que describe la interacción entre los agentes.

Los pasos del proceso de desarrollo de cada modelo son:

- *Modelo de Agentes:* se identifican los agentes y sus ambientes utilizando una extensión de las tarjetas CRC (Clases-Responsabilidades-Colaboraciones) a fin de incluir creencias, motivaciones, planes y atributos de cooperación.
- *Modelo Organizacional:* propone la identificación de los roles de cada agente y la elaboración de diagramas utilizando la notación OMT [Rumbau,1991] para la jerarquía de herencia y las relaciones entre los agentes.
- *Modelo de Cooperación:* se identifican las cooperaciones y los modelos de cooperación, y los tipos de mensajes que se intercambian entre agentes y además se analizan los protocolos utilizados.

- **Técnicas de modelado de agentes para sistemas de agentes BDI**

Este método define dos niveles principales (externo e interno) para el modelado de agentes BDI[10] [George,1995].

El nivel externo consiste de la descomposición del sistema en agentes y la definición de sus interacciones. Esto se lleva a cabo a través de dos modelos:

[10] BDI, creencias (Beliefs), deseos (Desires) e intenciones (Intention).

- *Modelo de Agentes*: para describir la jerarquía de relaciones entre agentes y las relaciones entre agentes concretos.
- *Modelo de Interacción*: para describir las responsabilidades, servicios e interacciones entre agentes y sistemas externos.

El nivel interno lleva a cabo el modelado de cada clase de agente BDI a través de tres modelos:

- *Modelo de Creencias*: que describe las creencias a cerca del ambiente.
- *Modelo de Objetivos*: que describe los objetivos y eventos que un agente puede adoptar o responder.
- *Modelo de Planes*: que describe los planes que una gente puede utilizar para lograr sus objetivos.

- **Ingeniería de Software Orientada a Agentes (AOSE)**

El proceso de AOSE [Grosse,1995] sigue los métodos de diseño Orientado a Objetos y consiste de tres partes, que son:

- *Análisis del Sistema Orientado a Agentes*: en esta etapa, se identifican a los agentes que exhiben el comportamiento deseado del sistema. Cada agente tiene su conjunto propio único de capacidades que lo caracterizan.
- *Diseño Orientado a Agentes*: se define la integración de los agentes identificados en la etapa anterior y los roles de los mismos. Un aspecto importante de los agentes en un sistema multiagente es que usualmente éstos están organizados en jerarquías. Este modelo de organización también se define en la etapa de diseño.
- *Programación Orientada a Objetos*: Se elige una arquitectura apropiada de agentes, y los agentes se implementan de acuerdo a la especificación de las dos etapas anteriores.

2.6.2.2. Extensión de metodologías de ingeniería del conocimiento

Según [Glaser,1996] las metodologías de ingeniería del conocimiento pueden proveer una buena base para el modelado de sistemas multiagentes, dado que ellos se ocupan del desarrollo de sistemas basados en el conocimiento. Y como los agentes poseen características cognitivas, estas metodologías pueden proveer las técnicas para el modelado de estos agentes.

- **Metodología CoMoMAS**

[Glaser,1996] propone una extensión de la metodología CommonKADS [Schreiber,1994] para el modelado de sistemas multiagentes. Se definen los siguientes modelos:

- *Modelo de Agentes*: este es el modelo principal de la metodología y define la arquitectura y el conocimiento del agente, que es clasificado como un conocimiento social, cooperativo, de control, cognitivo y reactivo.
- *Modelo de Capacidad*: describe la competencia cognitiva y reactiva del agente. Distingue entre tareas, resolución de problemas y conocimiento reactivo.
- *Modelo de Tareas*: describe la descomposición de las tareas, y detalla si la tarea se resuelve por el usuario o el agente.
- *Modelo de Cooperación*: describe la cooperación entre los agentes.
- *Modelo del Sistema*: define los aspectos organizacionales de la sociedad de agentes junto con los aspectos arquitecturales de los agentes.
- *Modelo de Diseño*: reúne los modelos previos a fin de operacionalizarlos, junto con los requerimientos no funcionales.

- **Metodología MAS-CommonKADS**

Esta metodología [Iglesias,1998] extiende el modelo definido en CommonKADS, agregando técnicas de metodologías orientadas a objetos (OOSE [Jacobson,1997], OMT [Rumbau,1991]).

La metodología inicia con una fase de conceptualización que es una fase informal destinada a colectar los requerimientos del usuario y obtener una primera descripción del sistema desde el punto de vista del usuario. Para este propósito se utilizan técnicas de ingeniería del software orientada a objetos [Jacobson,1997].

La metodología incluye los siguientes modelos:

- *Modelo de Agentes*: describe las características principales de los agentes, incluyendo las capacidades de razonamiento, habilidades (sensores/efectores), servicios, objetivos, etc.
- *Modelo de Tareas*: describe las tareas llevadas a cabo por el agente, y la descomposición de tareas.
- *Modelo de la Experiencia*: describe el conocimiento que necesita el agente para llevar a cabo las tareas.
- *Modelo de Coordinación*: describe las conversaciones entre los agentes, esto es, sus interacciones, protocolos y capacidades requeridas.
- *Modelo de Organización*: describe la organización en la cual el sistema multiagente será introducido y la organización de la sociedad de agentes.
- *Modelo de Comunicación*: detalla las interacciones humano-agente, y los factores humanos para desarrollar estas interfases.
- *Modelo de Diseño*: reúne los modelos previos y se subdivide en tres submodelos:
  1. *Diseño de la aplicación*: composición o descomposición de los agentes del análisis.
  2. *Diseño de la arquitectura*: diseño de los aspectos relevantes de la red de agentes.
  3. *Diseño de la plataforma*: selección de la plataforma de desarrollo de los agentes para cada arquitectura de agente.

2.6.3. Metodología seleccionada: MAS-CommonKADS

Al igual que algunas metodologías orientadas a objetos, como ser el Proceso Unificado Racional [Rumbau,1991], el modelo de ciclo de vida propuesto por esta metodología para el desarrollo de sistemas es el modelo en espiral dirigido por riesgos [Boehm,1988].

MAS-CommonKADS propone los siguientes modelos para el desarrollo de sistemas multiagente:

- *Modelo de Agente (AM):* específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes.
- *Modelo de Organización (OM):* es una herramienta para analizar la organización humana en que el sistema multiagente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno.
- *Modelo de Tareas (TM):* describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo.
- *Modelo de la Experiencia (EM):* describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de CommonKADS y reutiliza las bibliotecas de tareas genéricas.
- *Modelo de Comunicación (CM):* describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción. Aún no se ha sido desarrollado en la metodología.
- *Modelo de Coordinación (CoM):* describe las interacciones entre agentes software.
- *Modelo de Diseño (DM):* mientras que los otros cinco modelos tratan del análisis del sistema multiagente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multiagente como paso previo a su implementación.

2.6.3.1. Conceptualización

Antes de iniciar la especificación de los modelos descritos más arriba, la metodología MAS-CommonKADS incluye una etapa preliminar: la fase de conceptuación, empleando análisis centrado en el usuario mediante la técnica de casos de uso. Esta técnica permite definir los usos que dan los distintos usuarios al sistema. La notación propuesta para la fase conceptualización es tanto textual como gráfica. La notación textual consiste en plantillas

textuales que especifican los casos de uso y los actores, mientras que la notación gráfica sigue una extensión de la notación original de Jacobson [Jacobson,1997] (ver figura 2.5).

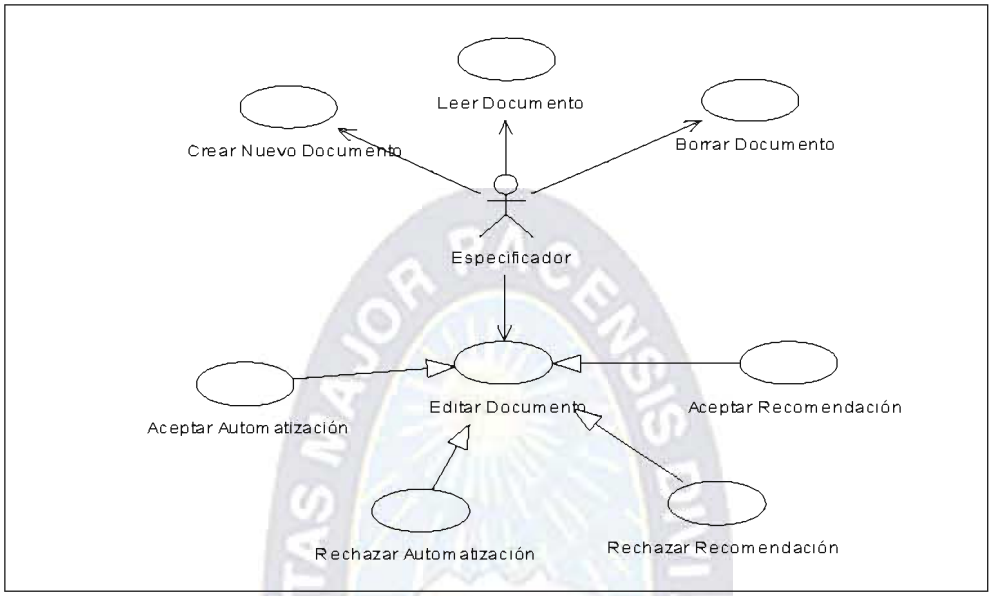


Figura 2.5. Ejemplo de Diagrama de Casos de Uso [BRJ,1997].

2.6.3.2. Modelo de agentes

El objetivo de este modelo es la descripción de los agentes software y humanos que operan en el sistema y sirve como punto de partida del resto de modelos de análisis de MAS-CommonKADS. El modelo de agente es una herramienta para analizar el sistema desde una perspectiva orientada a agentes. El desarrollo del modelo de agente consta de una fase de identificación de agentes, soportada por diferentes técnicas, que permite modelar el problema con agentes, y una fase de descripción de los mismos, en la que se describe con más detalle cuáles son las tareas encomendadas a los agentes y sus características. El resultado de esta etapa es un conjunto de plantillas de agentes similares al de la figura 2.6, en la que se describe las cualidades del agente. Además un conjunto de objetivos identificados por cada agente, los objetivos se describen por medio de plantillas de objetivos como se muestra en la figura 2.7.

Figura 2.6. Modelo de Agente: plantilla de agente [Iglesias,1998].

2.6.3.3. Modelo de organización

Los fines del modelo de organización son:

1. Comprender la organización en que se va a introducir un sistema multiagente y analizar la viabilidad del mismo.
2. Describir las relaciones de la sociedad multiagente.

En la figura 2.8 tenemos un ejemplo de un diagrama de organización, en el que se describe las relaciones de la sociedad multiagente, además de la relación de los agentes (cajas con los bordes superiores redondeados) con las entidades estáticas del entorno (cajas convencionales para representar clases en una notación Orientada a Objetos) como ser Usuario, Interfaz, etc. También se pueden ver relaciones de asociación entre las entidades del sistema.

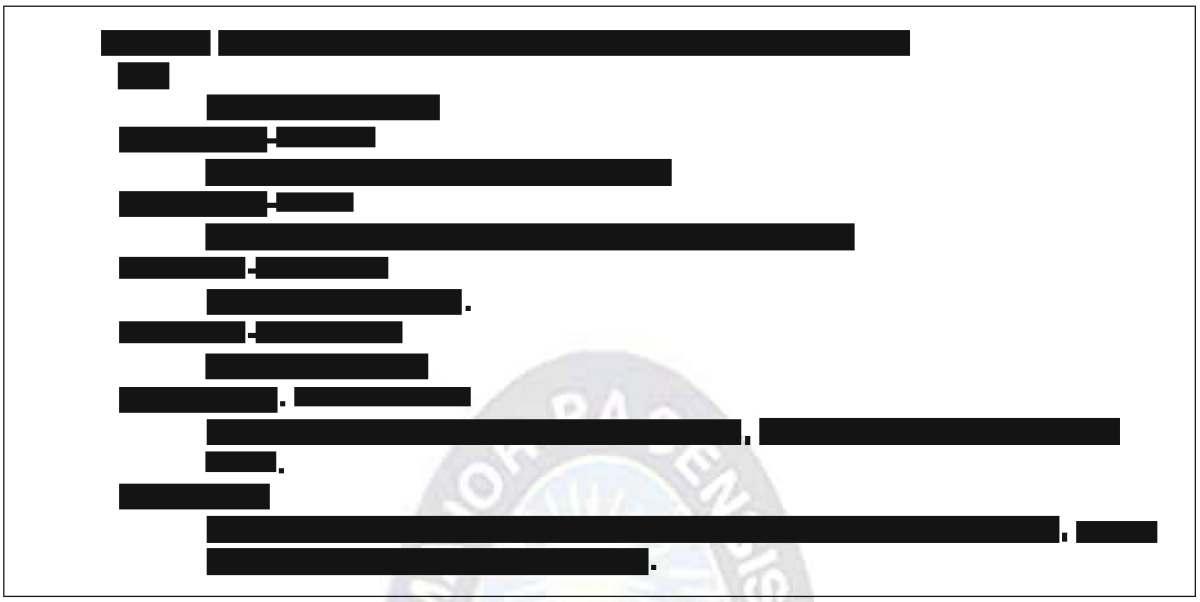


Figura 2.7. Modelo de Agentes: plantilla de objetivos [Iglesias,1998].

2.6.3.4. Modelo de tareas

El modelo de tareas de MAS-CommonKADS, da una visión de alto nivel de las tareas de la organización que deseamos modificar y de sus objetivos. Cada ejemplar del modelo de tareas está relacionado con un objetivo asignado a uno o varios agentes. Las tareas que requieren "experiencia" son desarrolladas en el modelo de la experiencia, mientras que las que requieran transferencia de información son desarrolladas en los modelos de comunicación y coordinación.

En la figura 2.9 se tiene un ejemplar de una tarea especificado utilizando la plantilla para tareas, como se puede ver, primero se especifica el objetivo asociado, luego se provee una pequeña descripción de la tarea, posteriormente se detallan la entrada, la salida, las condiciones de ejecución y la frecuencia. En la figura se observa también un apartado correspondiente a super- y sub- tarea, esto se debe a que la identificación de las distintas tareas resulta de una descomposición top-down de tareas más generales en tareas más específicas.

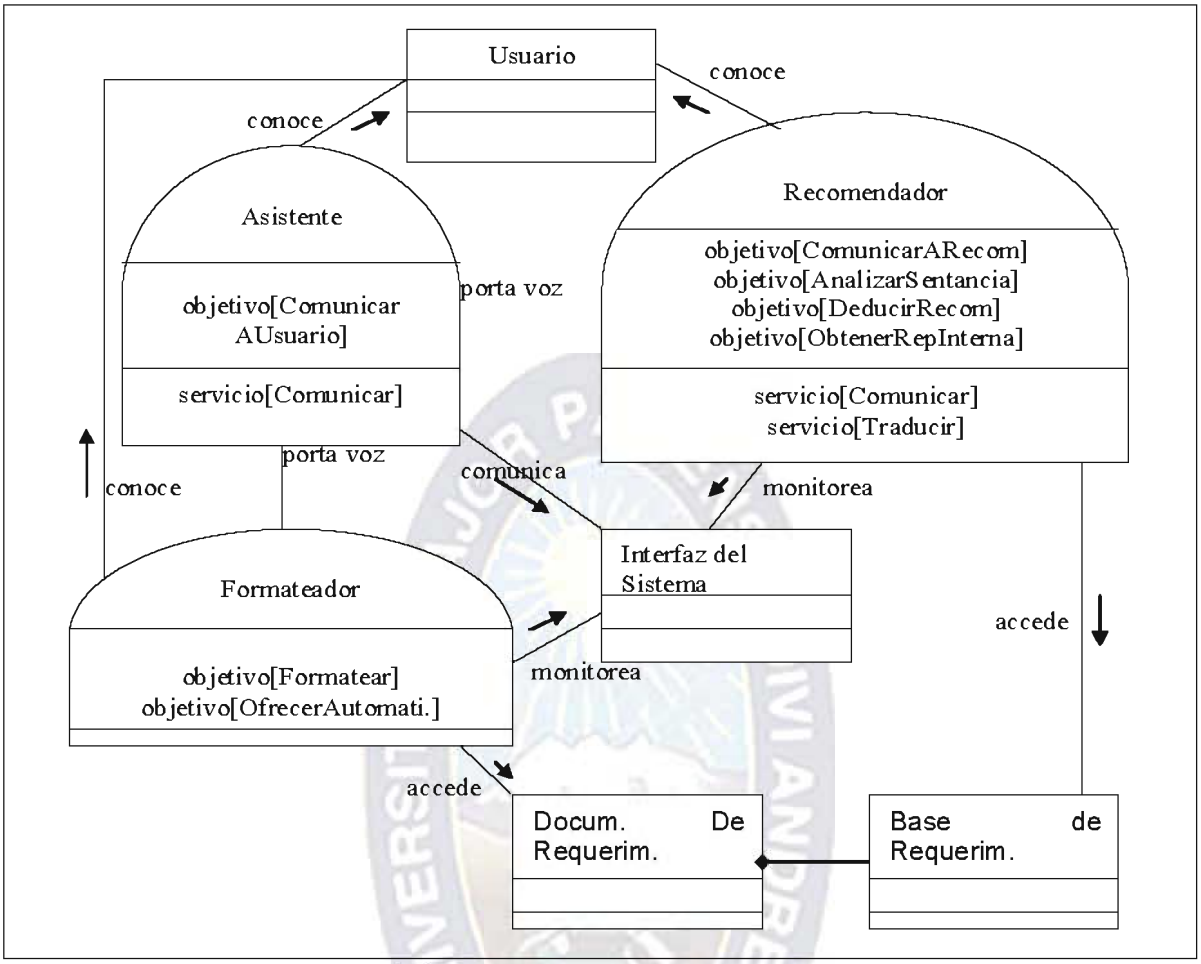


Figura 2.8. Diagrama de Organización: organización multiagente [Iglesias,1998].

2.6.3.5. Modelo de la experiencia

MAS-CommonKADS desarrolla en el modelo de la experiencia las tareas que requieren conocimiento para ser llevadas a cabo y que permitirán caracterizar al agente como un sistema basado en conocimiento. Para ello se describe la ontología o esquema del modelo, es decir, los principales conceptos del dominio y sus relaciones. Como se ve en la figura 2.10, se especifican los conceptos del dominio por medio de plantillas de concepto.

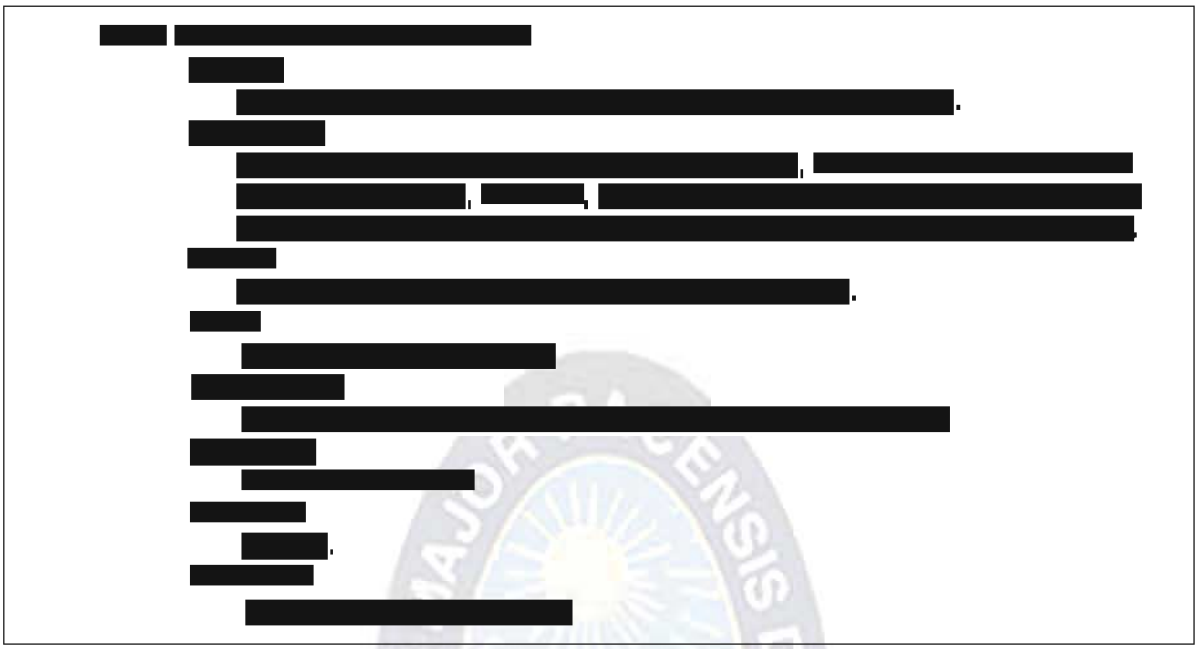


Figura 2.9. **Modelo de Tareas: plantilla de tarea [Iglesias,1998].**

2.6.3.6. Modelo de la coordinación

Presenta un modelo para describir las interacciones entre los agentes de una arquitectura multiagente. El modelo se estructura en torno a las conversaciones entre agentes. El modelo ofrece diversas técnicas y notaciones para la identificación y descripción de las conversaciones entre los agentes, las interacciones asociadas, y los objetivos que se pretenden con dichas conversaciones. El desarrollo del modelo combina técnicas de metodologías orientadas a objetos con lenguajes de descripción formal de ingeniería de protocolos.

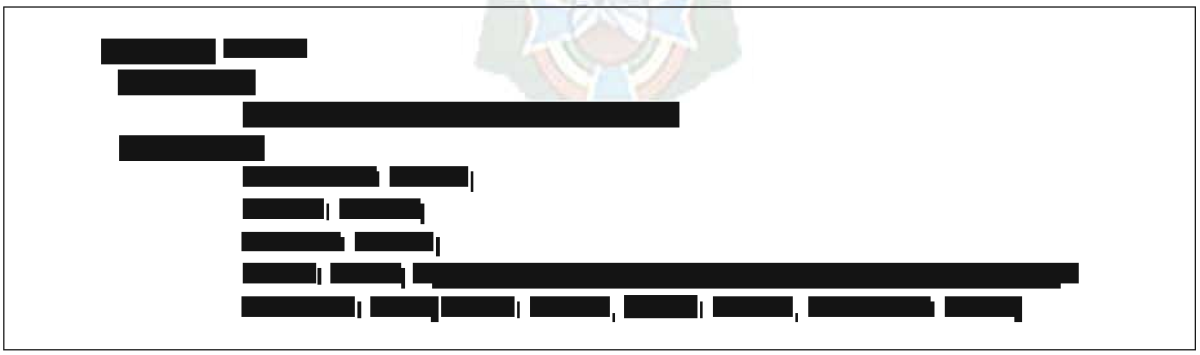


Figura 2.10. Modelo de Experiencia: plantilla de concepto de dominio [Iglesias,1998].

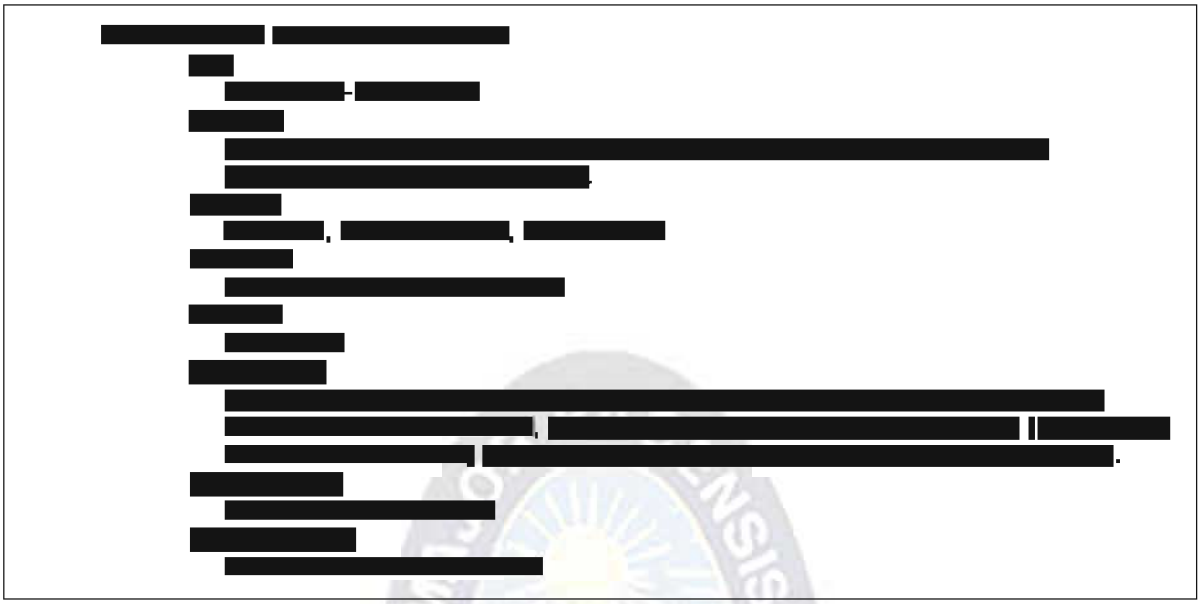


Figura 2.11. Modelo de la Coordinación: plantilla de conversación [Iglesias,1998].

En la figura 2.11 se presenta una plantilla de una comunicación entre tres agentes (Asistente, Recomendador, Formateador), por cada conversación se definen intervenciones que tiene por objetivo determinar los mensajes intercambiados entre los agentes, en la figura 2.12 se puede ver un ejemplo de una intervención, la notación es parecida a los diagramas de secuencia de la notación UML.

Figura 2.12. Modelo de Coordinación: intervención [Iglesias,1998].

2.6.3.7. Modelo de diseño

Este modelo se ha basado en concebir un sistema multiagente como una red que ofrece funcionalidades a los agentes (modelo de red) y una serie de agentes que operan en la red. En este modelo se recogen las especificaciones del resto de modelos y se decide cómo deben llevarse a cabo.

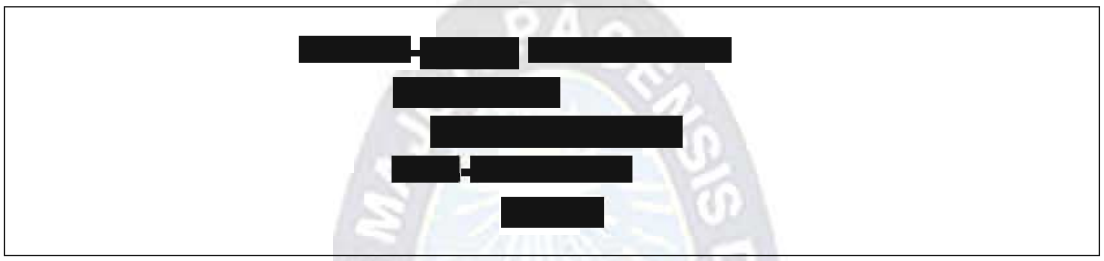


Figura 2.13. **Modelo de Diseño: plantilla de sistema agente [Iglesias,1998].**

En la figura 2.13, podemos ver una plantilla de sistema agente, en el que se puede especificar la arquitectura del agente a implementar y también en el caso de que tenga algún subsistema.

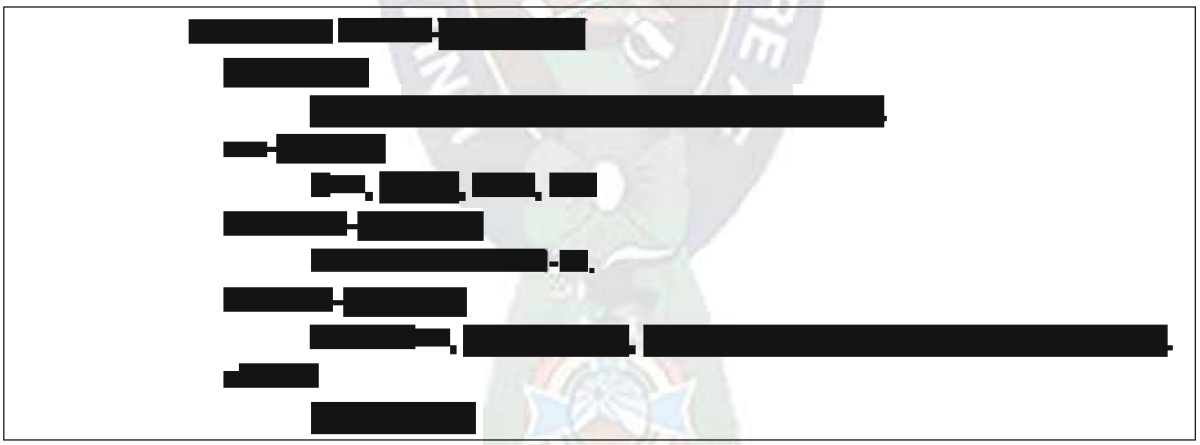


Figura 2.14. Modelo de Diseño: plantilla de plataforma [Iglesias,1998].

En al figura 2.14, podemos ver una plantilla de plataforma en la que se especifica todos los datos implementativos: lenguaje, plataforma de ejecución, usuario, etc.

CAPÍTULO 3

Marco Aplicativo

3.1. INTRODUCCIÓN

En este capítulo se describen los lineamientos de elaboración y desarrollo de la propuesta del Modelo de Replicación de Datos en Sistemas Distribuidos de Bases de Datos Relacionales (MRD), realizando una amplia descripción en el aspecto teórico y cubriendo una modesta parte en el aspecto práctico. Se aplicarán selectivamente los fundamentos teóricos descritos en el capítulo anterior, haciendo énfasis en lo que respecta a los conceptos de sistemas de bases de datos, replicación de datos y la metodología MAS-CommonKADS [Iglesias,1998] para el diseño de agentes.

3.2. REPRESENTACIÓN DEL MODELO PROPUESTO

Antes de empezar con la presentación del MRD es preciso describir el modelo básico de replicación de datos con el que todo profesional estándar de informática tiene por entendido. En esto, aparentemente la replicación de datos no tiene mayor utilidad que la copia sincrónica o asincrónica de los datos desde una base de datos activa (en producción) hacia otra pasiva (solo lectura), donde esta última funciona solamente como respaldo de la primera ante un posible siniestro o caída permanente del sistema. En la figura 3.1 se muestra un diagrama del tipo básico de replicación de datos, bajo el concepto de backup de una base de datos.

Al modelo básico de replicación de datos se le ha agregado mayores funcionalidades de replicación y que van de acuerdo a los requerimientos actuales de intercambio e integración de sistemas de bases de datos relacionales. También resulta sorprendente observar que los sistemas convencionales de replicación de datos, solamente dan soporte a modificaciones realizadas por el Lenguaje de Manipulación de Datos (DML), y no así a los cambios realizados por el Lenguaje de Definición de Datos (DDL).

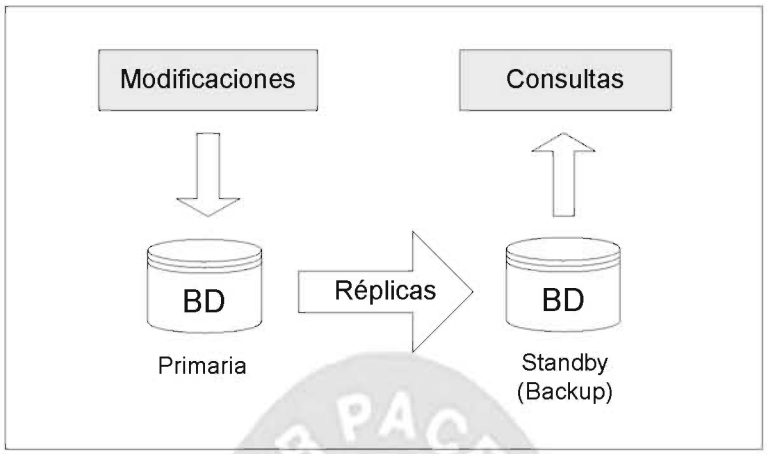


Figura 3.1. Modelo básico de replicación de datos [Oracle,1999].

Es importante mencionar que el modelo propuesto es un modelo de replicación asincrónico, aunque el ideal resultaría en uno sincrónico, sin embargo la experiencia nos dice que en sistemas distribuidos la independencia de gestión y operación obliga al uso de un modelo tolerante a fallos. El modelo general de replicación de datos planteado es el que se muestra en la figura 3.2.

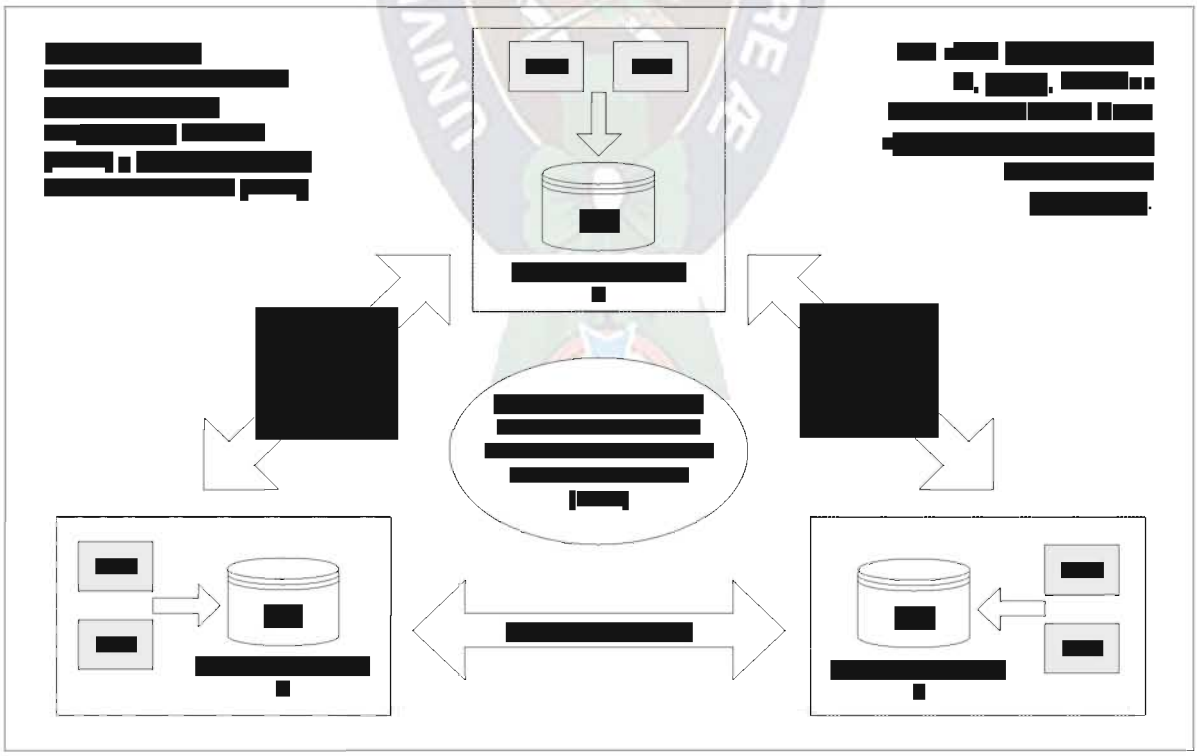


Figura 3.2. Propuesta general del MRD.

3.2.1. Modelo lógico

Desde un punto de vista lógico se intenta describir el MRD, el mismo contiene varios elementos que serán descritos posteriormente durante este capítulo. La figura 3.3 muestra la representación lógica del MRD.

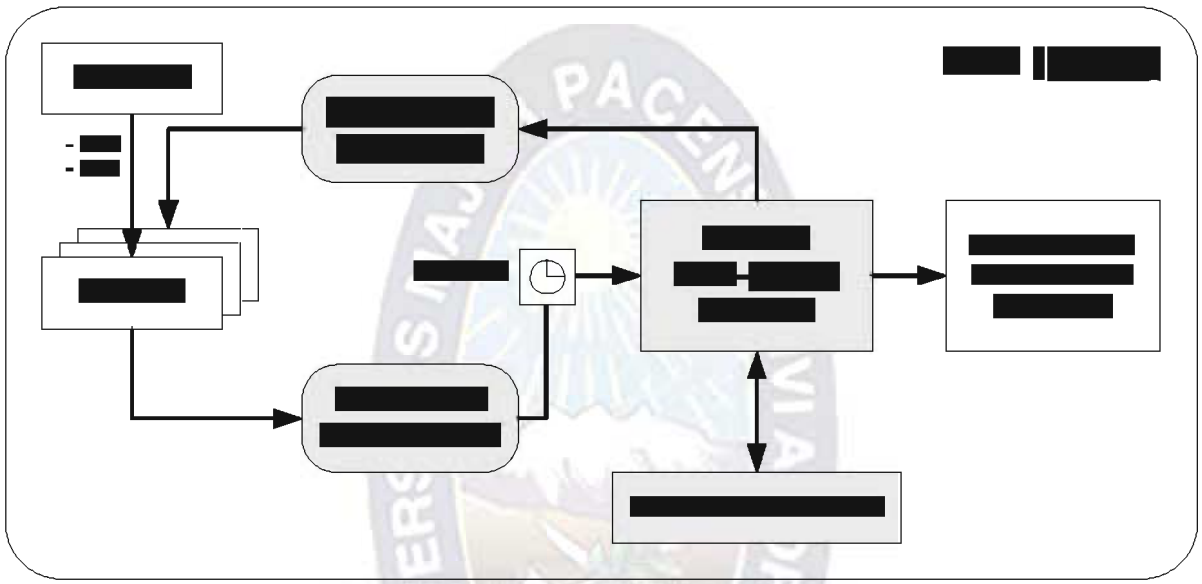


Figura 3.3. Representación lógica del MRD.

3.2.2. Modelo físico

Si bien una representación lógica describe la relación entre los elementos conceptuales del MRD, es preciso especificar una arquitectura general que el sistema adoptará con la aplicación del modelo propuesto, y esto lo hacemos mediante una representación física del MRD (ver figura 3.4).

3.3. IDENTIFICACIÓN DE LA ESTRUCTURA DEL MODELO

En este apartado se describe de manera clara y concreta la conceptualización y el análisis de la propuesta del MRD, describiéndose detalladamente los componentes y procesos principales que fundamentarán el diseño e implementación posterior del modelo.

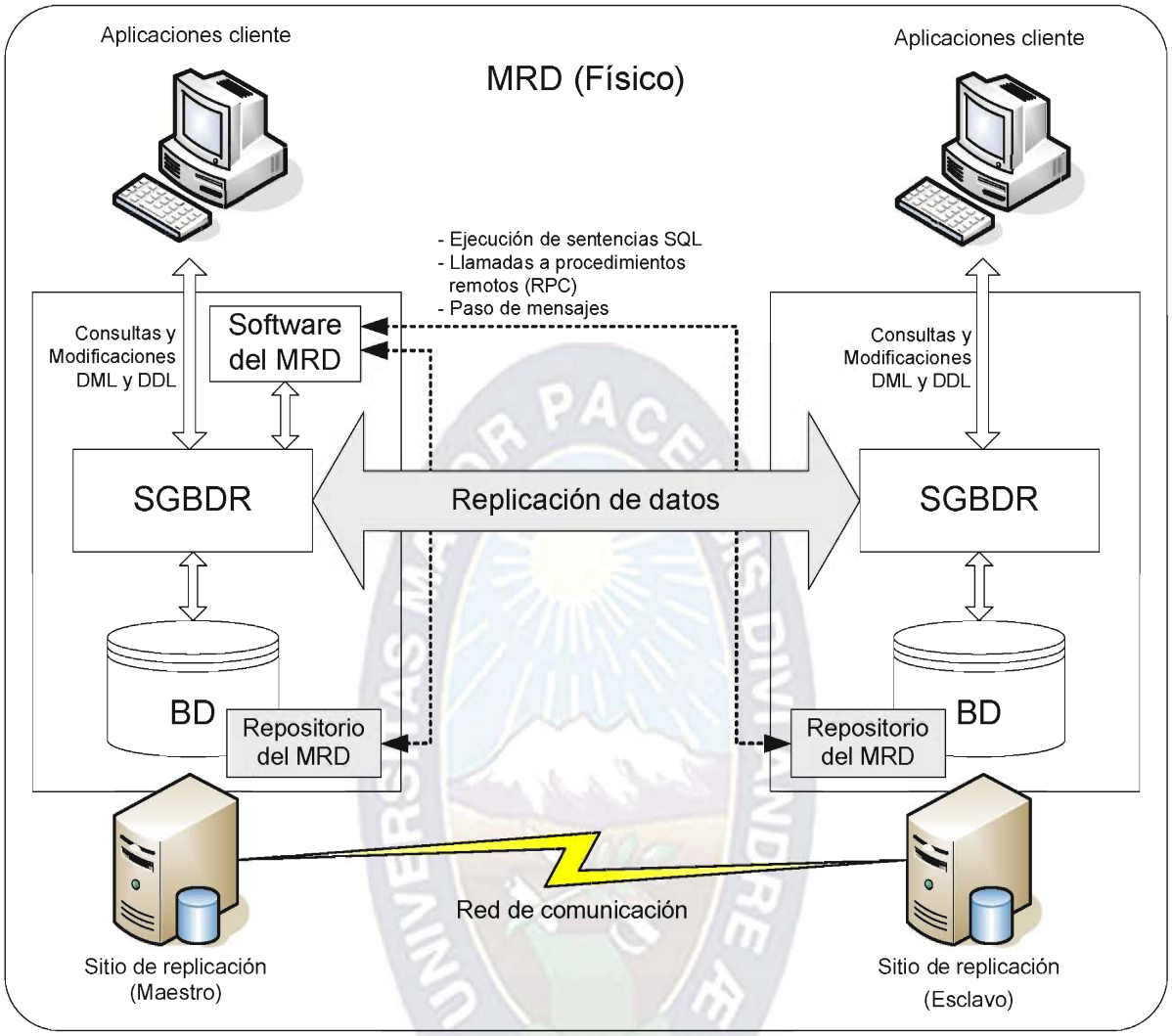


Figura 3.4. Representación física del MRD.

3.3.1. Entorno de replicación

Para tener un mejor entendimiento del concepto de entorno de replicación, será necesario describir los componentes de un Sistema de Gestión de Base de Datos Relacional (SGBDR). Típicamente un SGBDR esta compuesto por dos componentes principales: la Instancia de Servidor y la Base de Datos. La primera esta compuesta por todos los procesos (demonios) cargados en la memoria del host[11] donde reside el software del SGBDR (motor de base de datos), y la segunda es la propiamente dicha base de datos física, compuesta por un conjunto de archivos que son manipulados por la Instancia de Servidor mediante instrucciones al

[11] Equipo computacional que funciona como servidor de BD ó de aplicaciones cliente.

Sistema Operativo. Sin embargo para propósitos ilustrativos de la presente tesis, se tomará al SGBDR como sinónimo de Instancia de Servidor.

Entonces, el entorno de replicación propuesto, consistirá en la interrelación de varios SGBDRs enlazados por redes informáticas y administrado por el software que resulte de la implementación del MRD (ver figura 3.5).

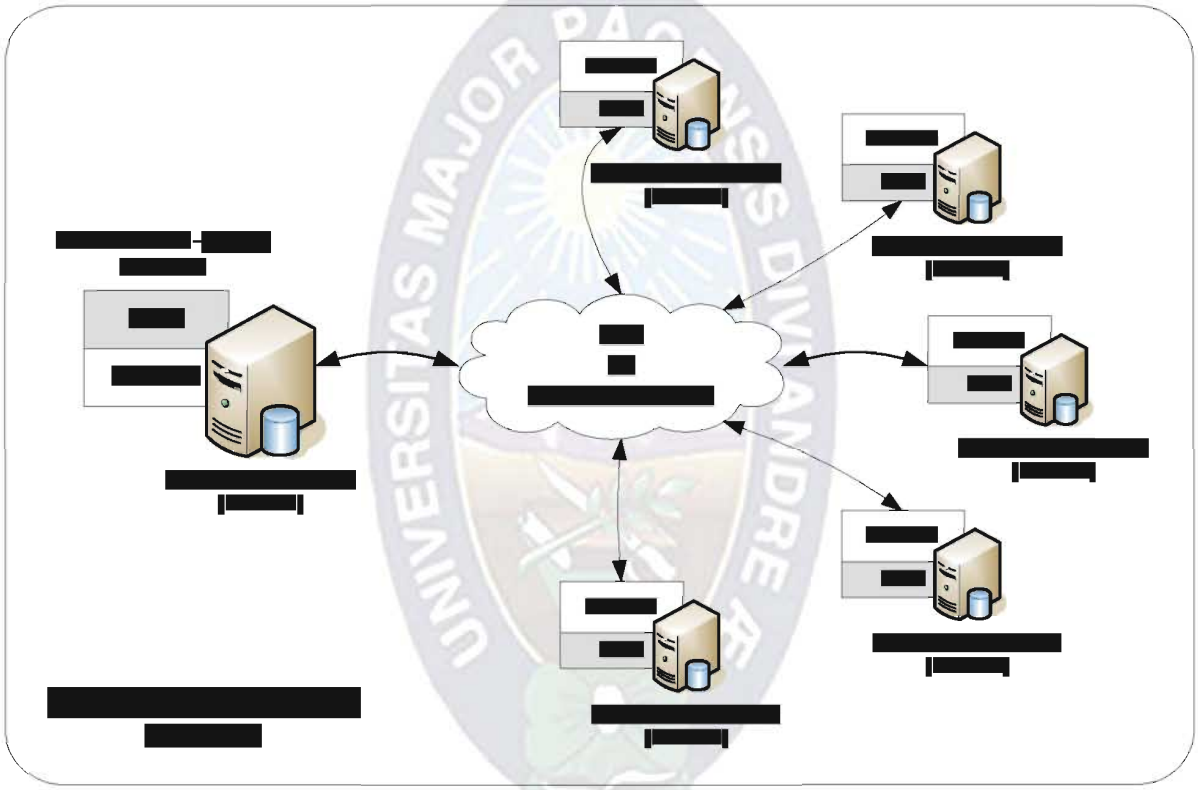


Figura 3.5. Entorno de replicación del MRD.

3.3.2. Sitios de replicación

Un sitio de replicación será aquel host que contenga al SGBDR, al repositorio[12] y/o software del MRD, y que forme parte del entorno de replicación del MRD. Como se puede observar en la figura 3.5, un entorno de replicación esta compuesto de dos tipos de sitios de replicación: un sitio maestro y uno o más sitios esclavos.

[12] Almacenamiento físico de objetos dentro de una base de datos.

3.3.2.1. Sitio maestro

Un sitio maestro de replicación será aquel donde se tenga instalado el SGBDR, el repositorio y el software del MRD (ver figura 3.4). Todos los agentes desarrollados residirán en la memoria física de este host, y para comunicarse con los sitios esclavos de replicación, se hará uso de los links[13] que proporcionan los SGBDRs. Una importante restricción es, que en un entorno de replicación sólo puede haber un sitio maestro, sin embargo si el requerimiento es disponer de varios sitios maestros, se deberán crear varios entornos de replicación.

3.3.2.2. Sitio esclavo

Un sitio esclavo de replicación será aquel componente del entorno de replicación en donde se tenga instalado el SGBDR y el repositorio del MRD (ver figura 3.4). En un sitio esclavo solo se podrá almacenar los cambios efectuados en la base de datos, pero no se tendrá la capacidad y autonomía de replicarlos a los demás sitios.

3.3.3. Componentes de la estructura

Si bien los conceptos de entorno y sitios de replicación, nos permiten tener una apreciación física de la estructura del MRD, deberá ser necesario el uso de alguna herramienta conceptual que nos permita observar el panorama desde un punto de vista lógico. Después de haber revisado el estado del arte de la Ingeniería del Software, encontramos interesante el paradigma Orientado a Objetos [Jacobson,1997] y el Lenguaje Unificado de Modelamiento (*Unified Modeling Language* - UML) [BRJ,1997], en donde el Diagrama de Clases permite la representación de la estructura estática del MRD, identifica a las entidades estructurales y las relaciones entre los mismos (ver figura 3.6).

[13] También llamados database links, son enlaces entre sistemas de bases de datos remotos.

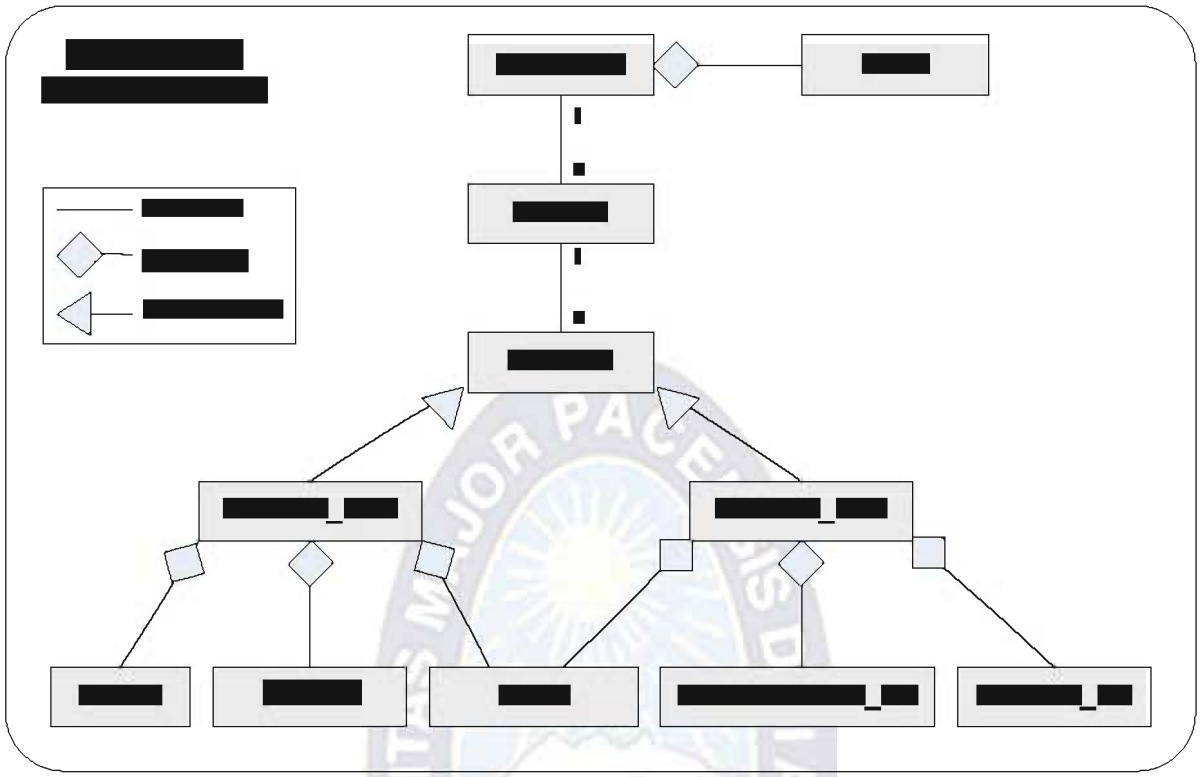


Figura 3.6. Diagrama de clases de la estructura estática del MRD.

3.3.3.1. Esquema de replicación

Dentro de los SGBDRs existen diferentes tipos de objetos, los cuales deben ser organizados de manera lógica, para ello se conforman los esquemas, que son conjuntos de objetos heterogéneos que pertenecen al mismo propietario, es decir, al propietario del esquema. Los objetos pueden ser: tablas de datos, procedimientos almacenados, disparadores (triggers), vistas, índices, constraints, secuencias, etc. Para más información sobre estos conceptos refiérase a Fundamentos de Sistemas de Base de Datos [Elmasri,2000].

Un entorno de replicación se asocia a un esquema de replicación, por tanto el nombre que se asigne al entorno, será el mismo para el esquema, y éste será el mismo en todos los sitios que formen parte del entorno de replicación. Por lo general los esquemas son creados en la base de datos como repositorios, en archivos de datos específicamente dedicados para su almacenamiento.

3.3.3.2. Grupos y objetos de replicación

- **Objetos de replicación.** Un objeto de replicación consiste en un conjunto de elementos que permitirán replicar datos y metadatos de una base de datos relacional. Los datos serán las filas de una tabla y los metadatos (definiciones) serán las estructuras de los objetos de la base de datos (tablas, índices, disparadores, procedimientos almacenados, etc.). Desde este punto de vista se tienen dos tipos de objetos de replicación:
  - *Objetos de replicación de datos.* Se utilizará para la replicación de las filas modificadas de una tabla, y estará compuesto por la tabla, una cola de transacciones y una lógica de replicación (ver figura 3.11).
  - *Objetos de replicación de definiciones.* Se utilizará para la replicación de las definiciones de los objetos, como ser, por ejemplo, la creación y/o alteración de la estructura de una tabla ó procedimiento almacenado. Estará compuesto por el objeto de la BD, una cola de definiciones y un componente de respaldo/recuperación (ver figura 3.12). Este último, como medida de seguridad en caso de falla en la ejecución de una sentencia DDL, debiéndose restaurar la definición inicial de la estructura del objeto de base de datos.

- **Grupos de replicación.** Un grupo de replicación será la agrupación lógica de varios objetos homogéneos de replicación, y varios grupos de replicación pertenecerán a un esquema de replicación. Existirán dos tipos de grupos de replicación:
  - *Grupos de replicación de datos.* Este tipo de grupo estará formado por uno o más objetos de replicación de datos.
  - *Grupos de replicación de definiciones.* Este tipo de grupo estará formado por uno o más objetos de replicación de definiciones.

El Agente de Replicación será el único encargado de ejecutar los grupos de replicación establecidos en el entorno de replicación, y registrar estadísticas de éxito y error en los procesos consecutivos de replicación.

3.3.3.3. Colas

Una cola se la representa fisicamente como una tabla de datos, en donde se almacenarán las referencias a las transacciones y definiciones. Existirán dos tipos de colas:

- *Colas de transacciones*. En este tipo de cola se almacenarán las direcciones fisicas de las filas modificadas, de una o más tablas de datos.
- *Colas de definiciones*. En este tipo de cola se almacenarán las referencias a los objetos de la base de datos, de los cuales se desea replicar su estructura (definición) o cuerpo (código fuente).

3.3.4. Encolamiento

El encolamiento de transacciones y definiciones consiste en insertar las direcciones y referencias de las filas y objetos, respectivamente, en colas diseñadas para ello.

3.3.4.1. Tipos de encolado

Se tienen dos tipos de encolamiento, dependiendo del tipo de cola utilizado. Se describe a continuación cada uno de ellos:

- *Encolamiento de transacciones*. Una vez seleccionada y definida la tabla de datos a replicar, ésta entra en un proceso de encolamiento automático y continuo, vía un trigger de encolado. Este trigger responde ante cualquier cambio sobre las filas de la tabla, como ser operaciones de inserción, actualización y eliminación de datos (ver figura 3.7).
- *Encolamiento de definiciones*. Este proceso, a diferencia del anterior, no encola automáticamente los cambios a la estructura de un objeto de la base de datos, debiendo realizarlo manualmente (ver figura 3.8). Es recomendable hacerlo así, por factores de desempeño, seguridad, confiabilidad y compatibilidad de estándares exigidos por los SGBDRs.

3.3.4.2. Criterios de encolado

Inicialmente describiremos los criterios de encolamiento para transacciones:

- *Encolamiento total*: Este criterio de encolamiento es el más utilizado por los fabricantes de soluciones de replicación, y consiste en encolar todas las modificaciones generadas en las filas a replicar, sin tomar en cuenta aspectos cualitativos y condiciones particulares.
- *Encolamiento por rol*: Los SGBDRs usan roles para organizar a los usuarios y permitirles el acceso a ciertas áreas de negocio; por ello resulta interesante seleccionar las transacciones a replicar, en función al rol que posean determinados usuarios de aplicaciones.
- *Encolamiento por perfil*: Los SGBDRs definen tipos de usuarios, en función al perfil que posean; estos pueden ser: usuarios de aplicaciones, usuarios de desarrollo, usuarios de soporte, usuarios administradores, etc. Se puede usar esta cualidad de los usuarios para seleccionar las transacciones a replicar.
- *Encolamiento por lista de usuarios*: Otra alternativa, algo similar a las anteriores, es crear una lista de usuarios, donde todas las transacciones originadas por dichos usuarios deberán ser replicadas.
- *Encolamiento por condición*: Este criterio de encolamiento, resulta el de mayor flexibilidad, puesto que deja al usuario administrador del sistema, la introducción de condiciones de cumplimiento en los datos para un eventual encolado de transacciones.

Para el encolamiento de definiciones, el único criterio de encolado es el siguiente:

- *Encolamiento manual*. Este encolado se lo debe hacer desde la interfase del MRD, para adicionar y/o seleccionar los objetos (definiciones) de la base de datos replicar, también se deben seleccionar el sitio origen y los sitios destinos a los cuales se deben propagar las réplicas. Para ello es necesario la intervención del Administrador de la Base de Datos (DBA), o el encargado en la administración del sistema de replicación.

3.3.4.3. Proceso de encolado

Como se explicó anteriormente, existen dos tipos de encolamiento, por lo tanto es natural suponer que existan dos procesos distintos de encolado.

- **Proceso de encolado de transacciones.** Este proceso consiste en el almacenamiento automático de las direcciones físicas de los registros modificados, en una cola de transacciones. La política de almacenamiento y lectura en la estructura de la cola, será la FIFO (*First Input First Output*). En la figura 3.7 se describe el proceso de encolado de transacciones originadas por el DML.

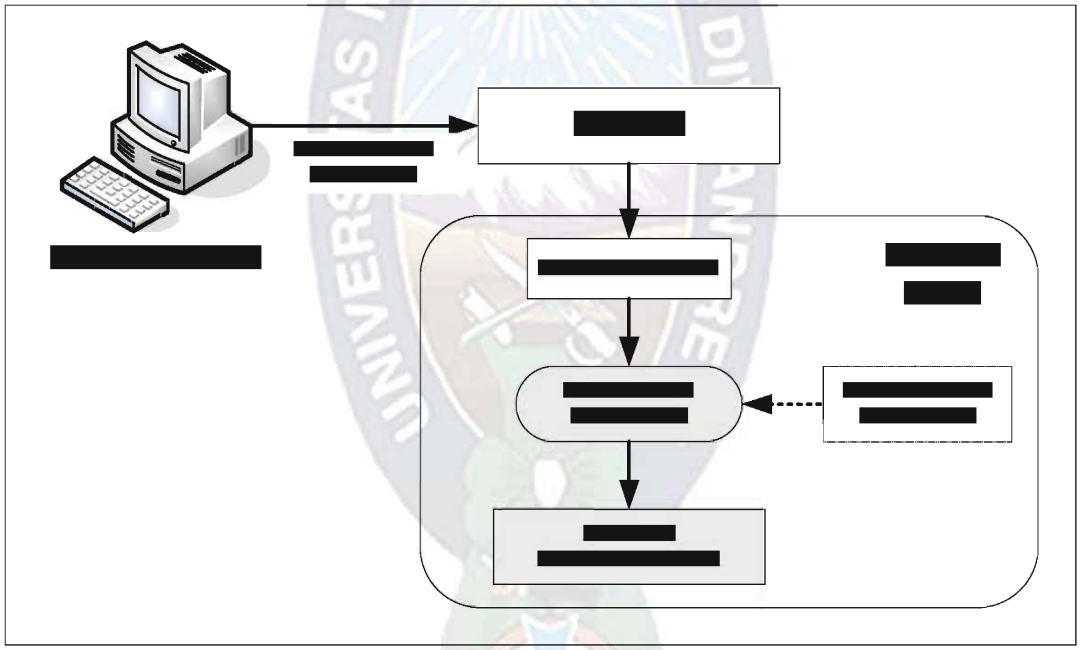


Figura 3.7. Encolamiento de transacciones.

- **Proceso de encolado de definiciones.** Este proceso consistirá en la selección manual (vía interfase del MRD) de los objetos cuyas definiciones deseen ser replicadas, y opcionalmente podrán ser editadas por el DBA para conseguir una réplica a medida (estático) al sitio de replicación de destino. La figura 3.8 describe el proceso de encolado de definiciones.

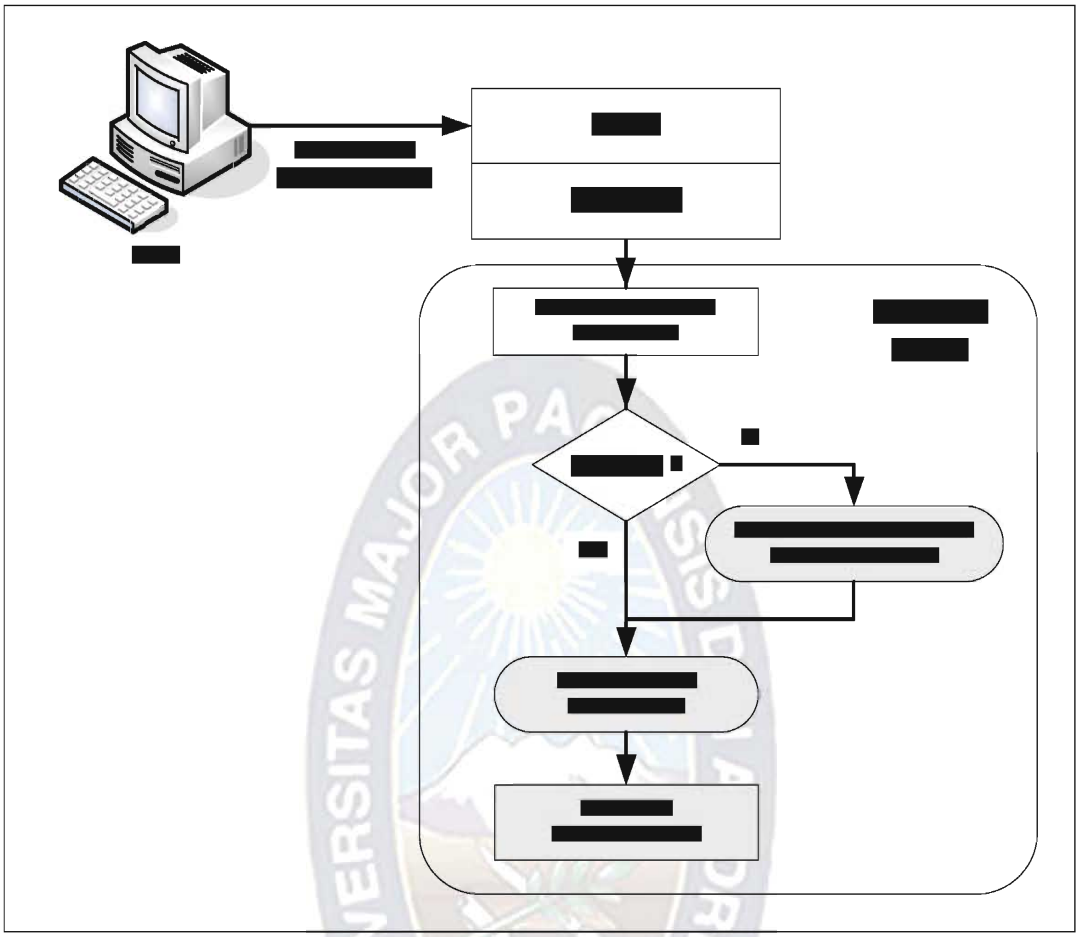


Figura 3.8. Encolamiento de definiciones.

3.3.5. Replicación

3.3.5.1. Tipos de replicación

Los tipos de replicación soportados por el MRD serán los siguientes:

- En cuanto al tiempo de latencia, únicamente se soportará el tipo asincrónico.
- En cuanto al sentido del viaje de las réplicas, se soportará ambos tipos, el unidireccional y el bidireccional.

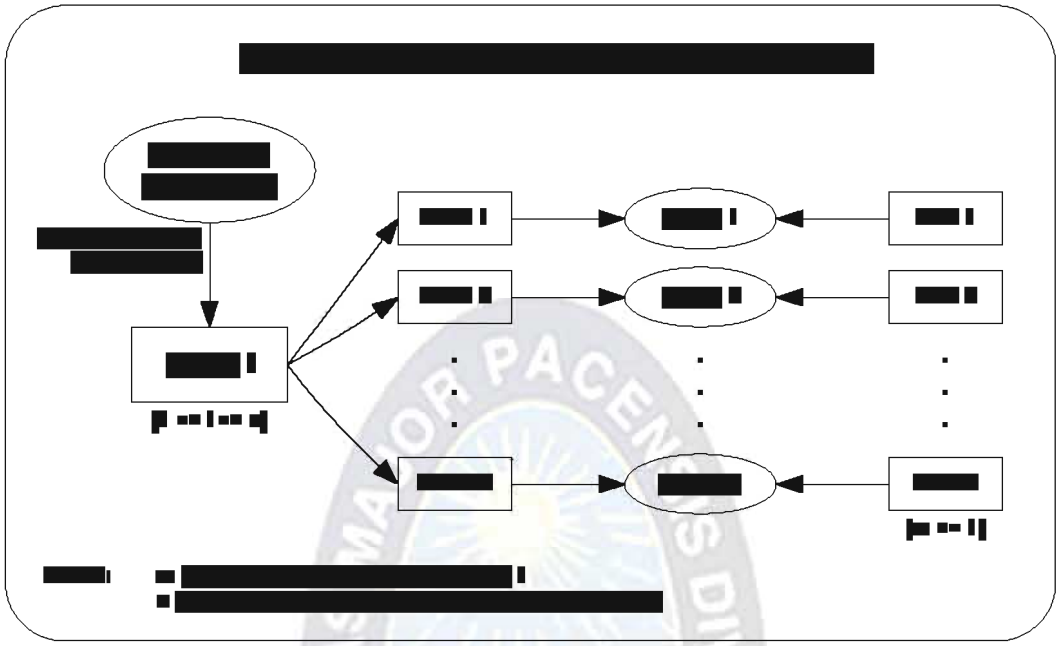


Figura 3.9. Grupo de objetos con colas individuales.

3.3.5.2. Lógica de replicación

La lógica de replicación será el módulo de código fuente asociado a un objeto de replicación de datos, cuyo propósito será enviar y registrar la réplica del sitio de replicación origen hacia el sitio de replicación destino. El código fuente inicialmente será creado por la interfase del MRD, sin embargo el DBA podrá editarlo para un ajuste particular.

Todo objeto de replicación de datos, deberá contener tres elementos: una lógica, una tabla y una cola de transacciones. Una cola de transacciones podrá estar asociado a un único objeto de replicación (ver figura 3.9), ó a un grupo de objetos de replicación (ver figura 3.10).

3.3.5.3. Componente de respaldo/recuperación

Las funciones que debe cumplir el componente de respaldo/recuperación son las siguientes:

- *Extracción de respaldos:* Antes de la modificación de un objeto remoto (destino), se deberá extraer una copia de la definición de su estructura.

- *Generación de sentencias DDL:* En base a la información del objeto, generar las sentencias necesarias para crear y/o modificar la definición de su estructura, sin perder los datos almacenados en él.
- *Ejecución remota de sentencias:* Mediante los links (enlaces entre bases de datos) que proporcionan los SGBDR distribuidos, se deberá ejecutar las sentencias, en el sitio de destino.
- *Recuperación ante fallas:* En caso de fallar la ejecución de las sentencias, restaurar la definición inicial del objeto modificado, mediante la ejecución de los respaldos guardados.

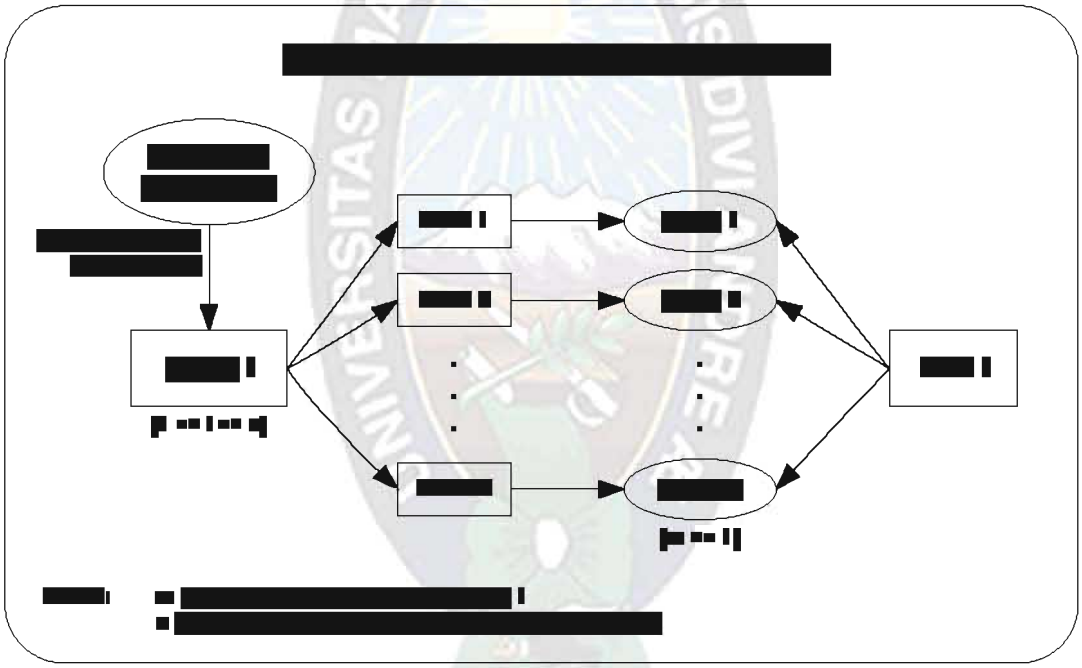


Figura 3.10. Grupo de objetos apuntando a una única cola.

3.3.5.4. Proceso de replicación

El proceso de replicación de transacciones (ver figura 3.11) realiza las siguientes tareas:

- El Agente de Replicación iniciará el proceso, ejecutando los grupos de replicación habilitados para ello.
- Los grupos de replicación contienen uno o más objetos, los cuales serán ejecutados por orden cronológico de creación.

- Con el objeto de replicación en memoria, leer la cola de transacciones (FIFO), extrayendo las direcciones físicas de las filas modificadas, el tipo de operación efectuado (inserción, actualización o eliminación), fecha de la operación, etc.
- Dirigirse a la tabla de datos correspondiente y extraer los datos actualizados.
- Ejecutar la lógica de replicación, pasando como parámetros: los datos actualizados, el tipo de operación, sitio origen, sitio destino, etc.
- Si la lógica de replicación reportase un error, se deberá enviar el error (mensaje) al Agente de Conflictos de Replicación, para que el mismo sea corregido.
- Actualizar la cola de transacciones, fijando como procesado el registro replicado.

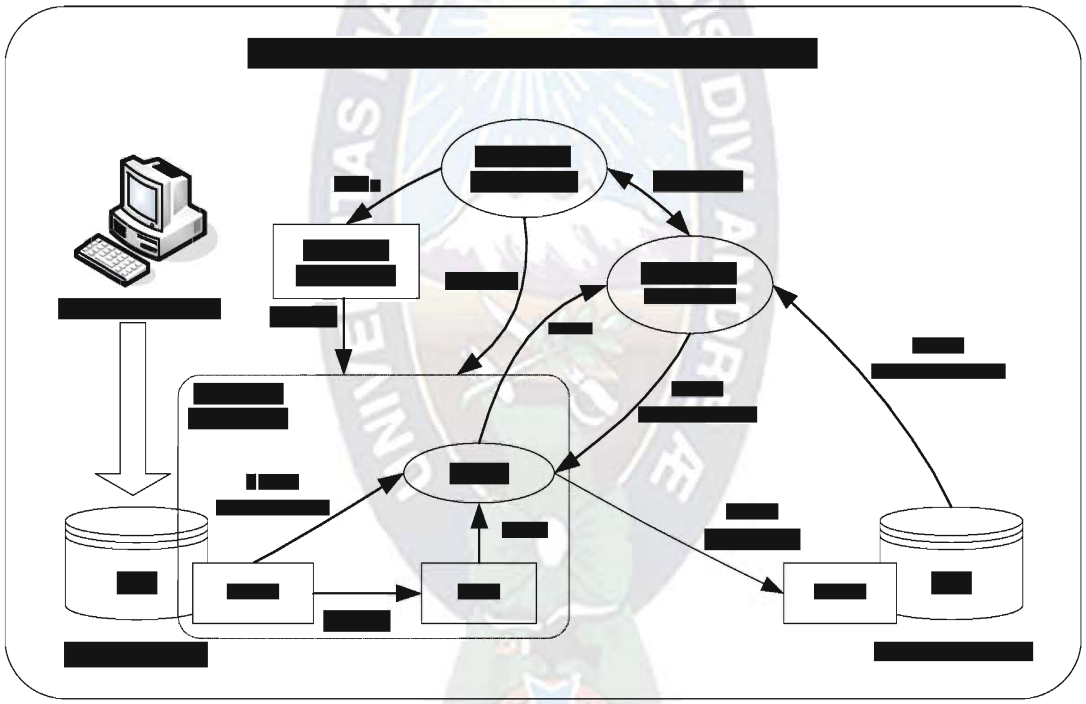


Figura 3.11. Proceso de replicación de transacciones.

Por otro lado, el proceso de replicación de definiciones de objetos de la BD (ver figura 3.12) realiza las siguientes tareas:

- El Agente de Replicación iniciará el proceso, ejecutando los grupos de replicación habilitados para ello.
- Los grupos de replicación contienen uno o más objetos, los cuales serán ejecutados por orden cronológico de creación.

- Con el objeto de replicación en memoria, leer la cola de definiciones, extrayendo los Identificadores de los objetos de la BD.
- El componente de respaldo/recuperación deberá extraer, del sitio destino, un respaldo (copia) de la definición del objeto original.
- Si la replicación es dinámica: dirigirse al diccionario de objetos de la BD, generar la sentencia DDL y ejecutarlo remotamente en el sitio destino.
- Si la replicación es estática: ejecutar la sentencia DDL de la cola, en el sito destino.
- Si la ejecución de las sentencias DDL reportase algún error, se deberá enviar el error (mensaje) al Agente de Conflictos de Replicación, para que envíe de retorno un mensaje de reintento, si el problema se puede corregir, ó un mensaje de restauración, para que el objeto sea restaurado a su estado inicial.
- Actualizar la cola de definiciones, fijando como procesado el objeto replicado.

3.3.6. Actualización de réplicas

3.3.6.1. Latencia

La medida de latencia es la cantidad de tiempo que una replica puede estar inconsistente hasta llegar a estar consistente con la fuente primaria designada [Buretta,1997]. Por ello, es muy importante mencionar que el hecho de encolar y posteriormente replicar las copias, puede generar inconsistencia permanente, puesto que durante el tiempo de latencia, existe inconsistencia temporal de las réplicas. Si la inconsistencia temporal tiende a convierte en inconsistencia permanente, se deberá poseer de un mecanismo muy riguroso de resolución de conflictos de replicación.

3.3.6.2. Tipos de actualizaciones

Los tipos de actualización de réplicas que el MRD soporta son dos: actualización automática y actualización manual.

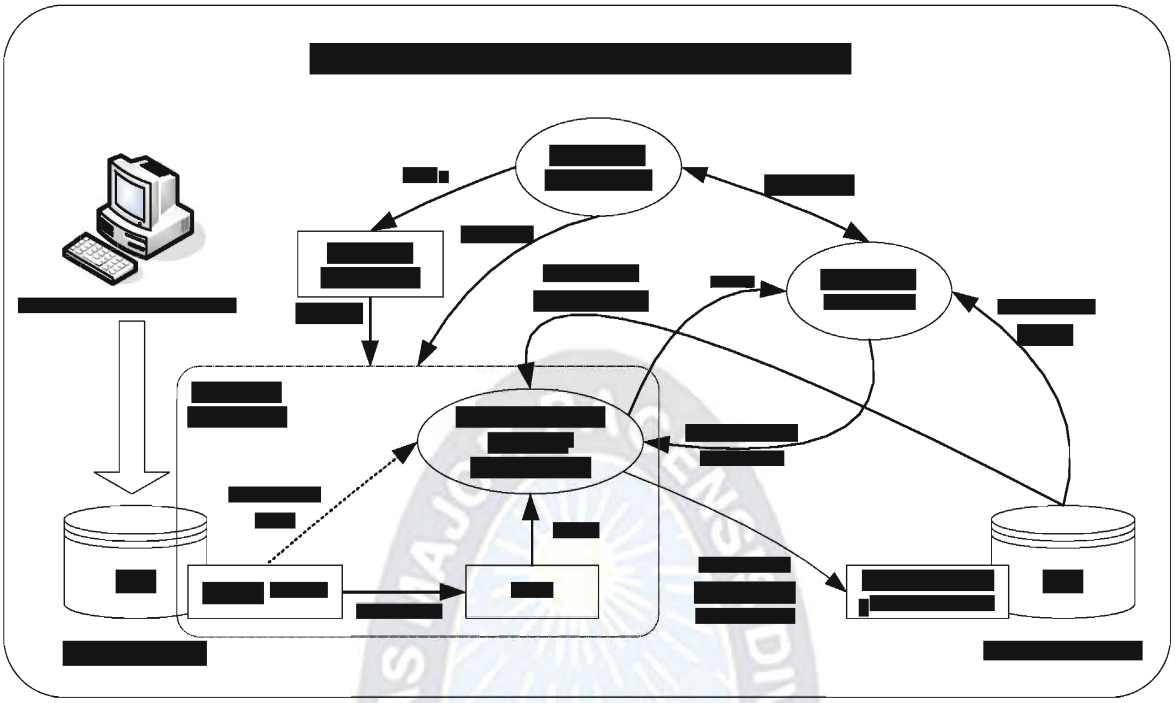


Figura 3.12. Proceso de replicación de definiciones.

- **Actualización automática**

  - *Relativo:* Este tipo de actualización consiste en la programación de intervalos de tiempo, relativos a la última ejecución. Por ejemplo, se podrá programar la ejecución de un grupo de replicación cada 10 minutos durante todo el año.
  - *Absoluto:* En este caso la programación es en fecha y hora calendario. Por ejemplo, se podrá programar la ejecución de un grupo de replicación cada lunes de la semana a las 08:00 hrs. a.m.
  - *Persistente:* Este tipo de actualización se ha convertido en un factor determinante de éxito o fracaso de un sistema de replicación, y sin embargo las herramientas actuales no lo implementan. Consiste en el intento continuo y duradero de ejecutar un grupo de replicación, dependiendo de las condiciones del tráfico en la red, frecuencia de ejecución, carga y tiempo que demanda su ejecución, estadísticas de ejecuciones con éxito y fracaso, etc. Como se puede notar, son condiciones exigentes, donde la aplicación de agentes de software podrían mostrar interesantes resultados.

- *Prioritario:* Este es otro caso, que no es tratado por los sistemas convencionales de replicación, y consiste en la ejecución de un grupo de replicación con alta prioridad. Es decir que si se tiene que elegir la ejecución de dos grupos para no saturar el tráfico en la red, se deberá elegir al grupo con tipo de actualización de mayor ponderación prioritaria.

- **Actualización manual**

Este tipo de actualización, como su nombre lo indica, consiste en la ejecución manual de un grupo de replicación, por parte del administrador de la BD, sin embargo el MRD recomendará la factibilidad o no, de la ejecución del grupo de replicación en ese momento.

3.3.6.3. Agente de replicación

Básicamente, el agente de replicación será el encargado de iniciar los procesos de replicación, en base a horarios establecidos y consultando su base de conocimiento, del cual obtendrá los estados de disponibilidad de la red de comunicación, tráfico en la red, carga y tiempo estimado de ejecución, estadísticas, etc., También registrará todos los eventos sucedidos en las etapas de replicación, esto para mantener actualizada su base de conocimiento. Es importante notar que todo objeto que requiera ser replicado, deberá formar parte de algún grupo de replicación.

3.3.7. Conflictos de replicación

Al replicar los datos de manera asíncrona y permitir la actualización de la información desde varios sitios origen hacia los sitios destino, debe buscarse la convergencia de la información, puesto que existe la posibilidad de generación de conflictos. Estos suceden cuando una misma información es modificada en varios sitios dentro del mismo intervalo de tiempo.

3.3.7.1. Tipos de conflictos

Los conflictos de replicación que se abordarán en el presente trabajo, solo abarcarán al tipo de actualizaciones efectuadas por el DML, puesto que resultan de mayor interés para los objetivos propuestos. Los tipos de conflictos que pueden ocurrir en el sitio de replicación destino son:

- *Conflicto de pérdida de unicidad:* ocurre cuando la replicación de un registro viola una integridad de llave primaria o unicidad (*primary key* o *unique index*).
- *Conflicto de actualización:* ocurre cuando una fila está siendo actualizada por dos transacciones y/o replicaciones diferentes simultáneamente.
- *Conflicto de eliminación:* ocurre cuando con una transacción y/o replicación se elimina un registro mientras que otra lo está actualizando o eliminando.
- *Conflicto de valores nulos:* ocurre cuando el registro a replicar, contiene campos con valores nulos, debiendo no ser así en el sitio destino.
- *Conflicto de duplicidad de filas:* ocurre cuando la lógica de replicación debe actualizar un único registro, sin embargo en el sitio destino existen registros duplicados.

3.3.7.2. Agente de conflictos

Para resolver los conflictos de replicación se hará uso de métodos de resolución de conflictos. Estos métodos aunados con nociones mentales, e implementados en un Agente de Conflictos de Replicación, ayudarán al administrador de la BD a tomar decisiones sobre los errores generados por conflictos de replicación.

Existen métodos de resolución de conflictos como [Oracle, 1999]:

- *Ignorar:* en el momento del conflicto, ignora el error aunque no se haya actualizado.
- *Aplicar:* en el momento del conflicto, aplica los cambios basándose en la llave de la tabla involucrada.
- *Suma y Promedio:* en el momento de conflicto utiliza una fórmula para obtener el valor a actualizar. Sólo es aplicable para un grupo de columnas formadas por campos numéricos.

- *Mayor:* en el momento del conflicto, inserta al final el valor que sea mayor.
- *Menor:* en el momento del conflicto, inserta al final el valor que sea menor.
- *Prioridad:* en el momento del conflicto, inserta al final los valores de las columnas de mayor prioridad.
- *Función:* en el momento del conflicto, define la lógica de resolución de conflictos con un programa, tal vez complejo.

Estos son solo algunos métodos de resolución de conflictos ya que como se podrá apreciar pueden existir innumerables criterios para decidir cual debe ser el valor a actualizarse en caso de un conflicto.

3.4. DESARROLLO DEL MODELO DE REPLICACIÓN

3.4.1. Metodología seleccionada

La metodología *MAS-CommonKADS* [Iglesias,1998], abarca de manera más exhaustiva las características del presente trabajo de investigación.

3.4.2. Conceptualización

El objetivo de la fase de conceptualización es obtener una primera aproximación al sistema que queremos desarrollar.

3.4.2.1. Identificación de los actores

En este problema, podemos identificar un único actor que interactúa con el sistema: el administrador del sitio de replicación maestro (*Database Administrator* - DBA), que es el usuario que realizará la creación, configuración, monitoreo y emisión de reportes del entorno de replicación del MRD.

3.4.2.2. Descripción de los actores

A continuación se describe al actor DBA empleando una plantilla textual [Iglesias,1998].

Actor *DBA*

descripción

persona que interactúa con el entorno de replicación del MRD.

3.4.2.3. Identificación de los casos de uso

Se identifican los siguientes casos de uso para el actor DBA:
Configurar el esquema de replicación, Crear el esquema, Modificar el esquema, Eliminar el esquema, Configurar sitios de replicación, Adicionar sitios, Modificar sitios, Eliminar sitios, Configurar grupos de replicación DML, Crear grupos DML, Modificar grupos DML, Eliminar grupos DML, Establecer criterios de encolamiento, Configurar grupos de replicación DDL, Crear grupos DDL, Modificar grupos DDL, Eliminar grupos DDL, Establecer modalidad de encolamiento, Configurar objetos de replicación DML, Crear objetos DML, Modificar objetos DML, Editar proceso de encolado DML, Editar lógica de replicación, Eliminar objetos DML, Configurar objetos de replicación DDL, Crear objetos DDL, Modificar objetos DDL, Editar proceso de encolado DDL, Editar componente de respaldo/recuperación, Eliminar objetos DDL, Programar horarios de replicación, Establecer métodos de resolución de conflictos, Monitorear el entorno de replicación, Leer logs de colas, Leer actividad de sitios de replicación, Leer actividad del entorno de replicación, Purgar transacciones inconsistentes, Activar/desactivar el entorno de replicación, Forzar inicio de procesos de replicación.

3.4.2.4. Descripción gráfica de los casos de uso

Los casos de uso [Jacobson,1997] describen las posibles interacciones o usos de un usuario con el sistema. Los usuarios del sistema se denominan actores, y representan a las entidades externas al sistema. La descripción gráfica de casos de uso del actor DBA, se lo presenta en la figura 3.13.

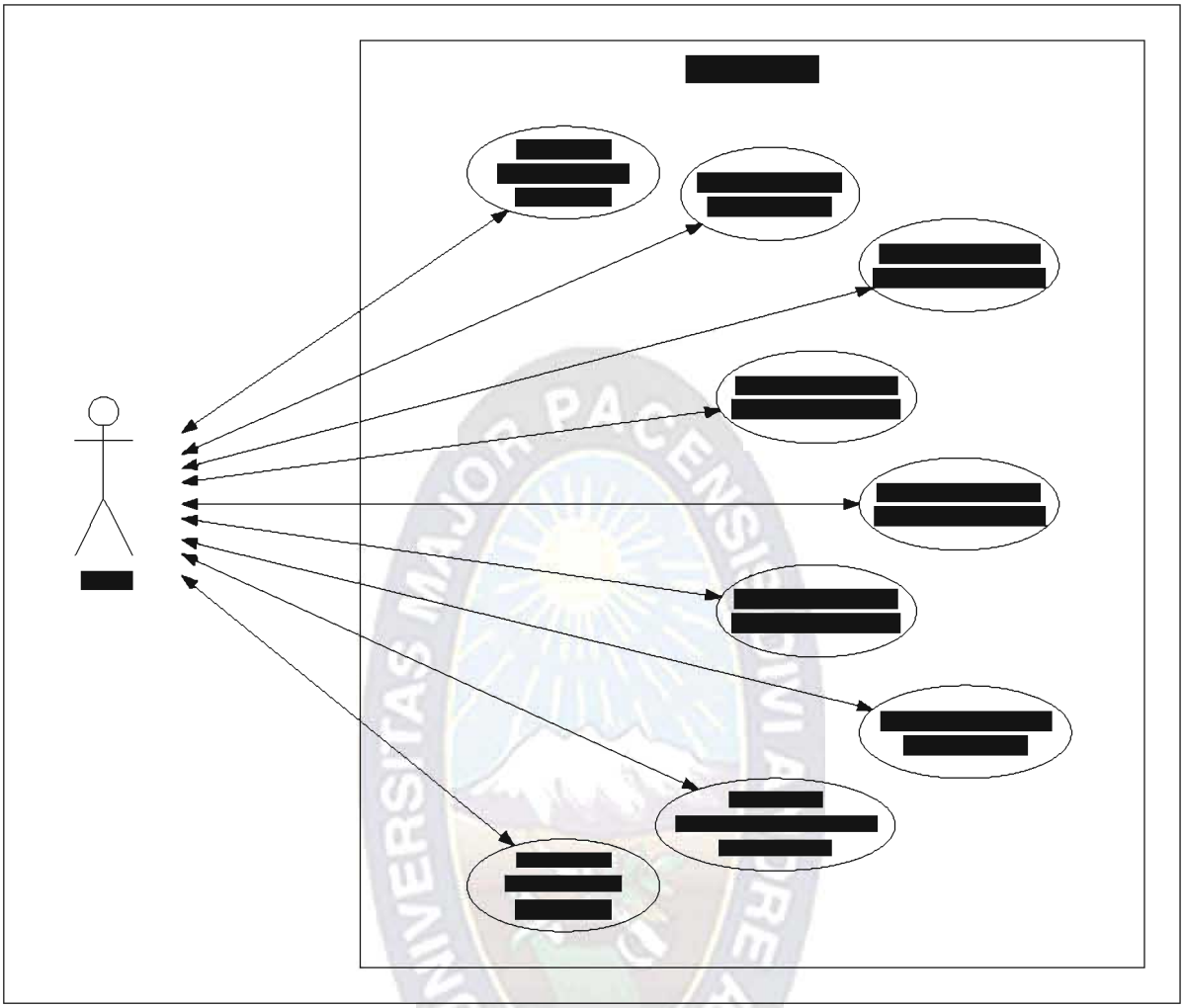


Figura 3.13. Casos de uso del actor DBA.

El detalle de los casos de uso extendidos se encuentran en el Anexo A.

3.4.2.5. Descripción textual de los casos de uso

A continuación se describe textualmente un caso de uso identificado para el actor DBA.

Caso de uso *Configurar grupos de replicación DML*

resumen

el DBA configura grupos de objetos de transacciones generados con el lenguaje de manipulación de datos (DML).

actores

DBA

precondiciones

ser usuario administrador, existir uno ó mas sitios de replicación esclavos.

descripción

el usuario selecciona la opción Configurar grupos DML, en la vista de configuración el usuario puede crear, modificar y eliminar grupos de objetos de transacciones. Durante la ejecución de este caso de uso, el usuario puede ejecutar los siguientes casos de uso: Establecer criterios de encolamiento, Crear grupos DML, Modificar grupos DML, Eliminar grupos DML.

excepciones

ninguna

postcondición

una vez configurados los grupos de replicación, se podrán configurar los objetos asociados a dichos grupos.

Las restantes descripciones textuales de casos de uso, identificados para el único actor, se encuentran en el Anexo A.

3.4.3. Modelo de agentes

El propósito del modelo de agente es describir los agentes que participan en la resolución del problema y la repercusión del sistema en los agentes humanos.

3.4.3.1. Identificación de los agentes (primera-iteración)

Partiendo del actor encontrado en la fase conceptual, podemos identificar al único agente externo al sistema: *DBA*. Vamos a mantener este actor como agente. La búsqueda de distribución en el problema puede ser útil en este caso. Si atendemos a la distribución del conocimiento, podemos determinar dos dominios diferentes: ofrecer recomendaciones de especificación y ofrecer automatización de tareas. No encontramos distribución de la información, ya que todo se encuentra en una base de datos centralizada.

3.4.4. Modelo de tareas

El modelo de tareas permite mostrar la descomposición funcional del sistema, y parte de las áreas funcionales (constituyente función) de la organización en que se va a introducir el sistema inteligente. En nuestro caso el sistema inteligente realiza la recomendación y toma de decisiones en la replicación de transacciones y especificaciones (*ReplicaciónDeDatos*), además de la recomendación y toma de decisiones en la resolución de conflictos de replicación (*ResoluciónDeConflictos*). Adicionalmente es necesaria la descripción de tareas

de apoyo a la configuración y monitoreo por parte del usuario DBA (*AsistirAlUsuario*). La descomposición de las tareas *ReplicaciónDeDatos*, *ResoluciónDeConflictos* y *AsistirAlUsuario* se presentan en los siguientes diagramas:

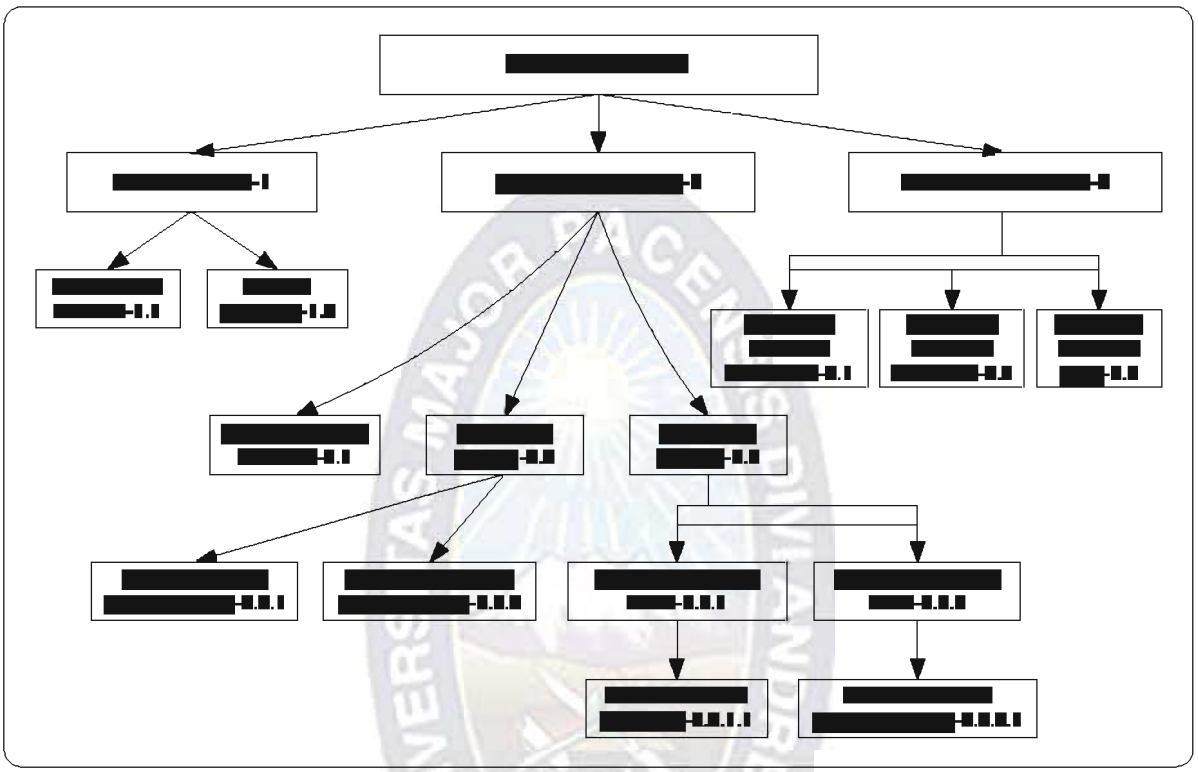


Figura 3.14. Descomposición de la tarea *AsistirAlUsuario*.

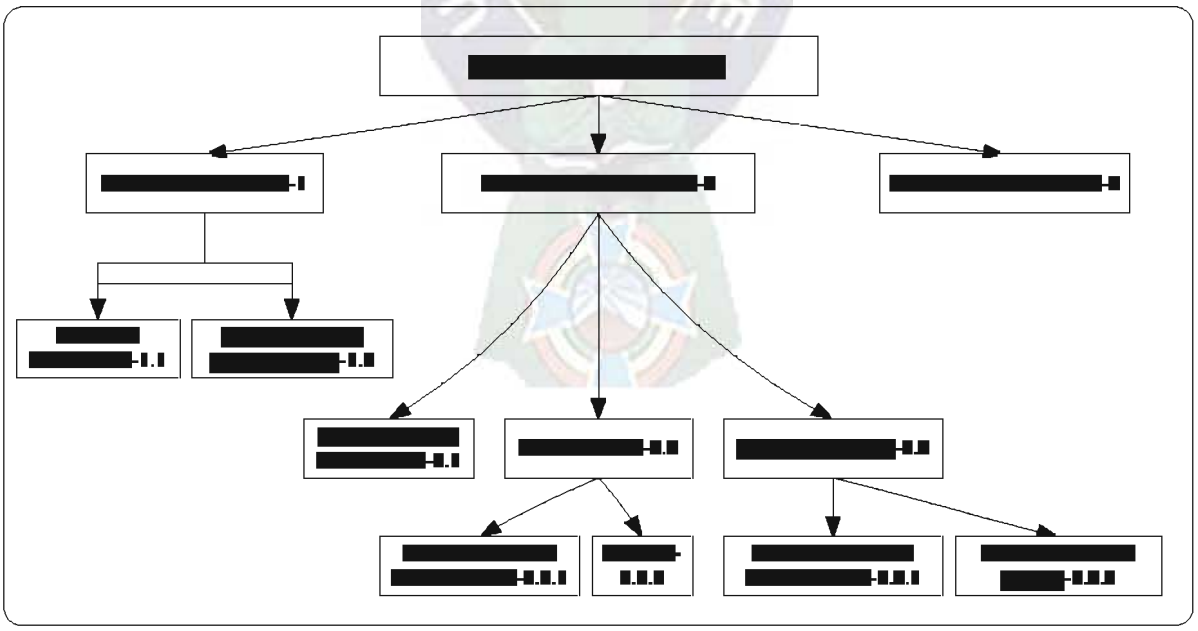


Figura 3.15. Descomposición de la tarea *ResoluciónDeConflictos*.

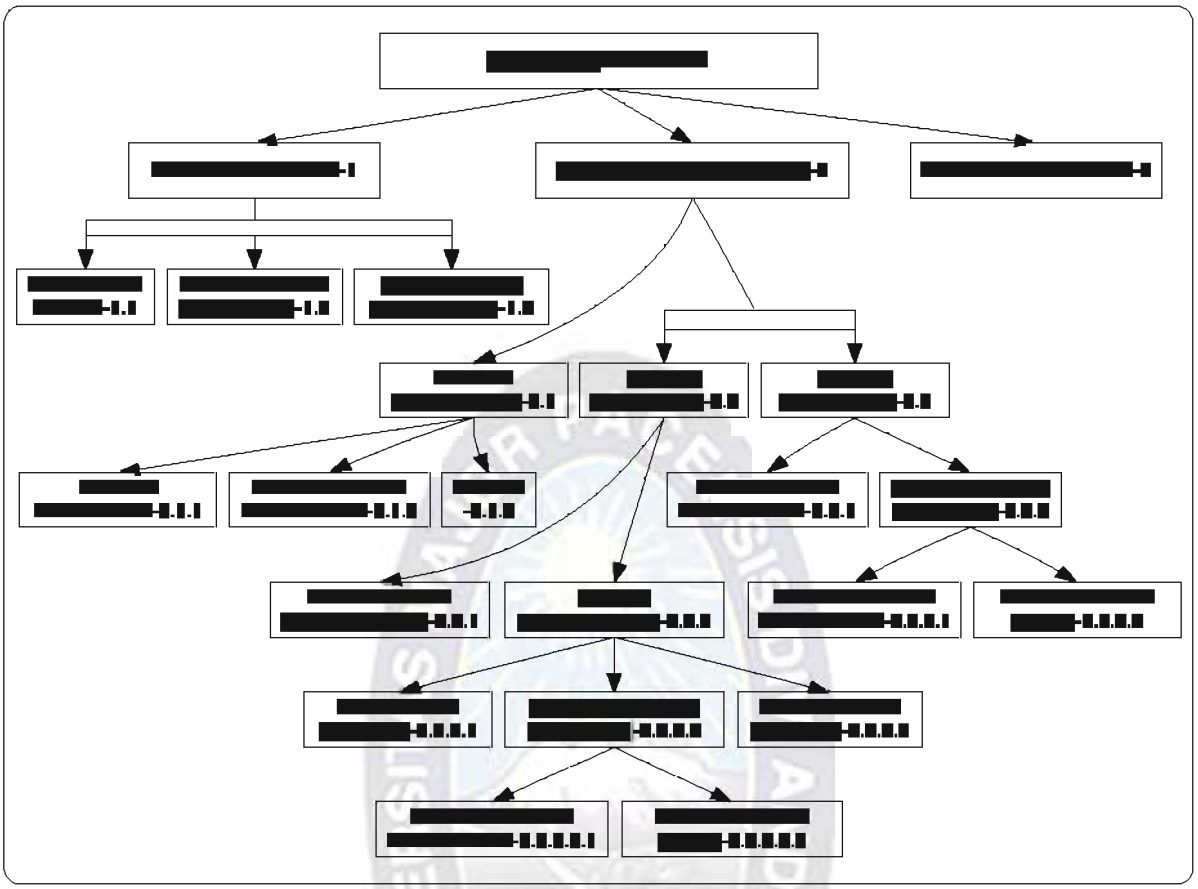


Figura 3.16. Descomposición de la tarea *ReplicaciónDeDatos*.

3.4.4.1. Plantillas de tareas y capacidades

A continuación presentamos dos tareas representativas, identificadas en la descomposición anterior, y una capacidad identificada (ver Anexo A para las demás plantillas de tareas).

Tarea *ConsultarBaseDeConocimiento*

objetivo

encontrar en la Base de Conocimiento posibles recomendaciones.

descripción

en esta tarea se buscan posibles recomendaciones, en la Base de Conocimientos, sobre tiempos esperados y frecuencias de replicación, jerarquías y prioridades de grupos y objetos de replicación, etc.

entrada

mensajes retornados por la tarea *VerificarConectividad* y por la supertarea.

salida

mensajes de recomendación.

precondición

invocación previa de la tarea *VerificarConectividad*.

supertarea

VerificarDisponibilidad

subtareas
ninguna
frecuencia
por cada parámetro de la entrada.

Tarea *Deducir*
objetivo
clasificar el orden de los mensajes de recomendación.
descripción
en esta tarea se clasifican las recomendaciones, tanto las ideas del propio agente, como las recibidas de otros agentes. La clasificación se realiza de acuerdo a un criterio propio del agente.
entrada
mensajes de recomendación.
salida
mensajes de recomendación ordenados según criterio.
precondición
mensajes de recomendación recibidos.
supertarea
VerificarDisponibilidad
frecuencia
por cada elemento de entrada.
requiere
ClasificaciónSegunCriterio

Capacidad *ClasificaciónSegúnCriterio*
descripción
es la capacidad que permite que el agente decida si es conveniente o no proveer la recomendación ideada al usuario u otro agente. En caso de toma de decisiones, permite que el agente pueda dar como respuesta una lista de posibles recomendaciones, en un orden tal que, el primero en la lista es lo más conveniente según el agente.

3.4.5. Modelo de agente

3.4.5.1. Identificación y descripción de los agentes (segunda-iteración)

Tras el análisis de las tareas realizado, podemos identificar algunos agentes del sistema multiagente. Siguiendo el criterio de asignar agentes de interfaz a los agentes humanos del sistema, definimos un agente *Asistente* (para la tarea *AsistirAlUsuario*) para el usuario DBA.

Si atendemos a la distribución en el problema, definimos un agente *Replicador* (para la tarea *ReplicaciónDeDatos*) y un agente *Corrector* (para la tarea *ResoluciónDeConflictos*). A continuación se presentan las plantillas de los agentes identificados en el dominio de aplicación.

Agente *Asistente*

tipo

agente software inteligente estacionario.

descripción

este agente se encarga de asistir al usuario DBA en su tarea de creación, configuración y monitoreo del entorno de replicación. Para ello debe responder a las acciones del usuario y en forma autónoma asistirlo. También debe rastrear continuamente las vistas de los elementos del entorno de replicación, la actividad general y los registros de salida (logs).

Capacidad-razonamiento

experiencia

conocimiento de prácticas de creación y configuración de elementos del entorno, conocimiento del repositorio del MRD (repositorio que contiene el diccionario de datos, plantillas de creación de objetos y especificaciones generales del Modelo de Replicación de Datos actualizadas con la herramienta).

comunicación

interacción con el usuario DBA

coordinación

interacción con el *Replicador* para iniciar procesos de replicación, e interacción eventual con el *Corrector* para algunos tipos de conflictos.

Capacidad-general

habilidades

monitorear las acciones del usuario DBA.

lenguaje-com

especificaciones internas para los sensores de entrada, y lenguaje natural para los efectores(sensores de salida).

Restricción

normas

el agente *Asistente* es el único encargado de comunicarse con el usuario DBA, los demás agentes lo realizan a través de éste agente.

preferencias

on demand (a demanda de solicitudes).

permisos

el agente *Asistente* puede tener acceso al repositorio del MRD para crear y configurar el entorno de replicación.

Agente *Replicador*

tipo

agente software inteligente estacionario.

descripción

este agente se encarga de rastrear solicitudes del usuario, horarios de replicación pendientes y mensajes del agente *Corrector*. Para luego dar inicio al proceso de replicación y en base al resultado obtenido, mostrar las recomendaciones correspondientes.

Capacidad-razonamiento

experiencia

conocimiento de prácticas de replicación, conocimiento del repositorio del MRD (repositorio que contiene información histórica y actual sobre grupos y objetos de replicación, tiempos promedios de ejecución, latencias, horarios con alto y bajo tráfico en la red, etc.).

comunicación

interacción con el agente *Asistente* a fin de comunicarse con el usuario DBA.

coordinación

interacción con el agente *Asistente* y el agente *Corrector*.

Capacidad-general

habilidades

monitorear las acciones del usuario DBA y del agente *Corrector*.

lenguaje-com

especificaciones internas para los sensores de entrada, y lenguaje natural para los efectores(sensores de salida).

Restricción

normas

el agente *Replicador* responde ante acciones del usuario (vía el agente *Asistente*) y mensajes del agente *Corrector*.

preferencias

on demand (a demanda de solicitudes de entrada).

permisos

el agente *Replicador* puede tener acceso al repositorio del MRD para consultar y actualizar la base de conocimiento.

Agente *Corrector*

tipo

agente software inteligente estacionario.

descripción

este agente se encarga de rastrear solicitudes del usuario y mensajes del agente *Replicador*, e intenta dar soluciones a los problemas de replicación recibidos, y en base al resultado obtenido, mostrar las recomendaciones correspondientes.

Capacidad-razonamiento

experiencia

conocimiento de prácticas de resolución de conflictos, conocimiento del repositorio del MRD (repositorio que contiene información histórica y actual sobre conflictos de replicación de datos, conflictos resueltos y no resueltos, frecuencias de aparición, etc.).

comunicación

interacción con el agente *Asistente* a fin de comunicarse con el usuario DBA.

coordinación

interacción con el agente *Asistente* y el agente *Replicador*.

Capacidad-general

habilidades

monitorear las acciones del usuario DBA y del agente *Replicador*.

lenguaje-com

especificaciones internas para los sensores de entrada, y lenguaje natural para los efectores(sensores de salida).

Restricción

normas

el agente *Corrector* responde ante acciones del usuario (vía el agente *Asistente*) y mensajes del agente *Replicador*.

preferencias

on demand (a demanda de solicitudes de entrada).

permisos

el agente *Corrector* puede tener acceso al repositorio del MRD para consultar y actualizar la base de conocimiento.

3.4.5.2. Distribución tareas-agentes

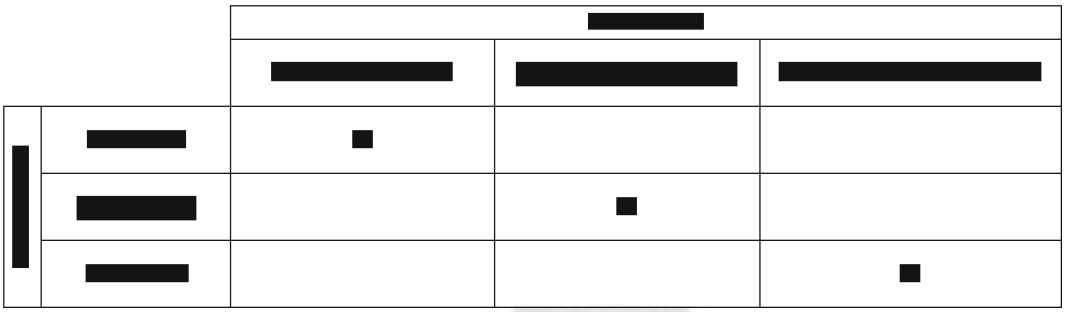


Tabla 3.1. Asignación de tareas a agentes.

3.4.5.3. Identificación y descripción de objetivos

Identificamos los siguientes Objetivos a partir de la asignación de tareas a agentes.

- **Objetivos del Agente *Asistente*:** Monitorear el ambiente para detectar acciones del usuario, Comunicar recomendaciones al usuario, Monitorear el entorno de replicación.
- **Objetivos del Agente *Replicador*:** Rastrear horarios programados de replicación, Verificar condiciones físicas mínimas para la replicación, Recomendar al *Asistente* horarios de replicación basado en experiencias pasadas, Replicar objetos de especificaciones y de transacciones, Solicitar solución de conflictos al *Corrector*.
- **Objetivos del Agente *Corrector*:** Identificar el tipo de conflicto, Idear solución para el conflicto, Intentar corregir el conflicto presentado.

Definición de objetivos a través de plantillas de objetivos (ver Anexo A):

Objetivo Idear solución para el conflicto
tipo
objetivo no persistente de aprendizaje
parámetros-entrada
objeto con conflictos de replicación, lista de posibles acciones
parámetros-salida
plan con las instrucciones necesarias de solucionar el conflicto.
condición-activación
cada vez que se lo invoque
lenguaje rep. conocimiento
representación interna
descripción
este objetivo permite que el *Corrector*, consultando la base de conocimiento, planifique la mejor solución a los conflictos de replicación del objeto en cuestión.

3.4.6. Modelo de la experiencia

El desarrollo del modelo de la experiencia en un sistema multiagente puede subdividirse en el conocimiento de la aplicación, el propio agente, el resto de agentes y el entorno.

3.4.6.1. Identificación de las tareas genéricas

En el dominio de aplicación identificamos las siguientes tareas como genéricas (tareas que requieren conocimiento para su ejecución): ConsultarBaseDeConcocimiento, Deducir, EsperarMensajeDeOtroAgente, ComunicarAOtroAgente.

3.4.6.2. Identificación y descripción del esquema del modelo

La ontología[14] del dominio describe las entidades relevantes del dominio y las relaciones mantenidas entre los ejemplares de dichas entidades. *CommonKADS* distingue la categoría ontológica *Concepto*, como la clase de objetos del dominio de estudio, se corresponde con el término "entidad" del modelado entidad-relación y "clase" en la orientación a objetos. Cada *Concepto* se identifica por un nombre, los posibles superconceptos, y un conjunto de propiedades (ver Anexo A). El siguiente es el concepto *Replicador*.

```
Concepto Replicador
        descripción
                representa al agente Replicador
        propiedades
                idReplicador: String;
                objetivos: collection(idObjetivo: String,
                                      regla: String);
                plan: collection(idPlan: String,
                                 traza: String);
                umbral: Integer;  /*representa el límite de aceptación de una recomendación*/
```

[14] Una ontología constituye: "una formal, especificación explícita de una conceptualización compartida" [Gruber,1993].

3.4.6.3. Correspondencia entre esquema del modelo y tareas genéricas

Esta actividad se subdivide en identificar correspondencias entre papeles de diferentes tareas genéricas una vez adquirido el esquema del modelo y, a continuación, relacionar los papeles de las tareas genéricas con términos del esquema.

Identificamos las siguientes relaciones:

- ConsultarBaseDeConcocimiento **usa**: BaseDeConocimiento, Replicador, Corrector.
- EsperarMensajeDeOtroAgente **usa**: Replicador, Corrector.
- ComunicarAOtroAgente **usa**: Replicador, Corrector.
- Deducir **usa**: BaseDeConocimiento, Replicador, Corrector.

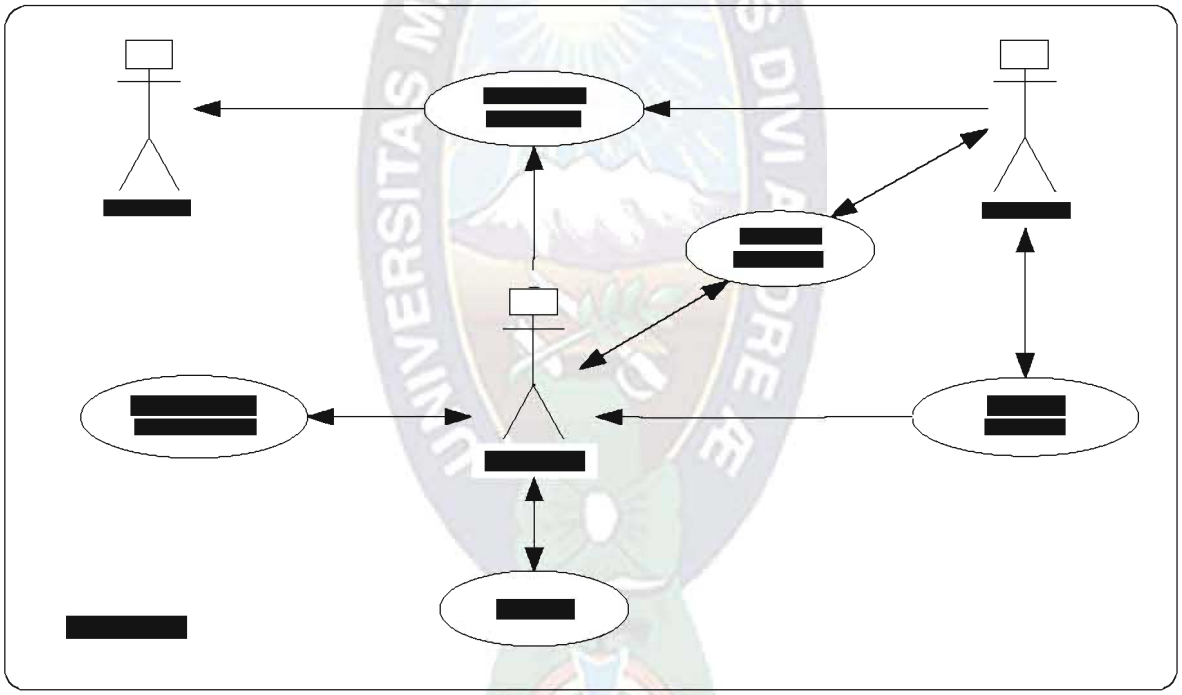


Figura 3.17. Conversaciones basadas en casos de uso internos.

3.4.7. Modelo de coordinación

El modelo de coordinación nos permite describir y desarrollar las interacciones entre los agentes de un sistema multiagente. En este apartado se presentan las actividades, necesarias para el presente trabajo, que se suceden en el desarrollo del modelo, éstas nos permiten profundizar en las interacciones entre los agentes, que se agrupan en conversaciones.

3.4.7.1. Identificación de las conversaciones

Se identifican las conversaciones necesarias para satisfacer un objetivo de un agente (o del sistema). Las distintas conversaciones las identificamos utilizando diagramas de caso de uso internos (ver figura 3.17).

3.4.7.2. Descripción de las conversaciones

Para cada conversación utilizaremos la plantilla textual del constituyente *Conversación* [Iglesias,1998] y emplearemos la notación de casos de uso propuesta por [Jacobson,1997] con extensiones para distinguir los agentes internos del sistema de los usuarios del mismo. A continuación presentamos una descripción textual de conversación por medio de las plantillas de conversaciones (ver Anexo A).

Conversación *Comunicar al Usuario*

tipo
comunicar información

objetivo
comunicar al usuario de las recomendaciones provenientes de los agentes Replicador y Corrector.

agentes
Asistente, Replicador, Corrector

iniciador
Replicador o Corrector

servicio
comunicar

descripción
los agentes Replicador y Corrector tienen por objetivo comunicar sus recomendaciones al usuario, para ello se contactan con el agente Asistente (ya que es la interfaz con el usuario) para solicitarle el servicio de comunicar la recomendación.

precondición
recomendación recibida

postcondición
recomendación comunicada

3.4.7.3. Descripción de las intervenciones

Para la descripción de las intervenciones emplearemos la notación de MSC[15] básicos para expresar la notación de casos de uso. Las intervenciones de una conversación tienen por objetivo determinar los mensajes intercambiados entre los agentes.

[15] MSC (*Message Sequence Charts*; Diagramas de Secuencias de Mensajes), verlo en [Iglesias,1998].

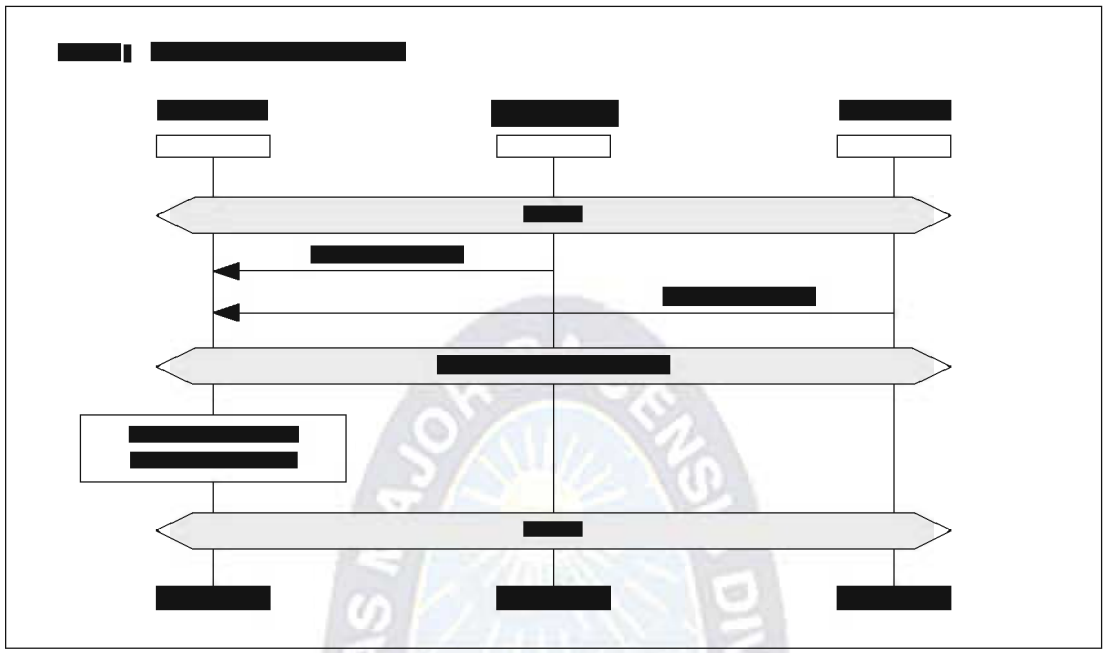


Figura 3.18. Intervención: comunicar al usuario.

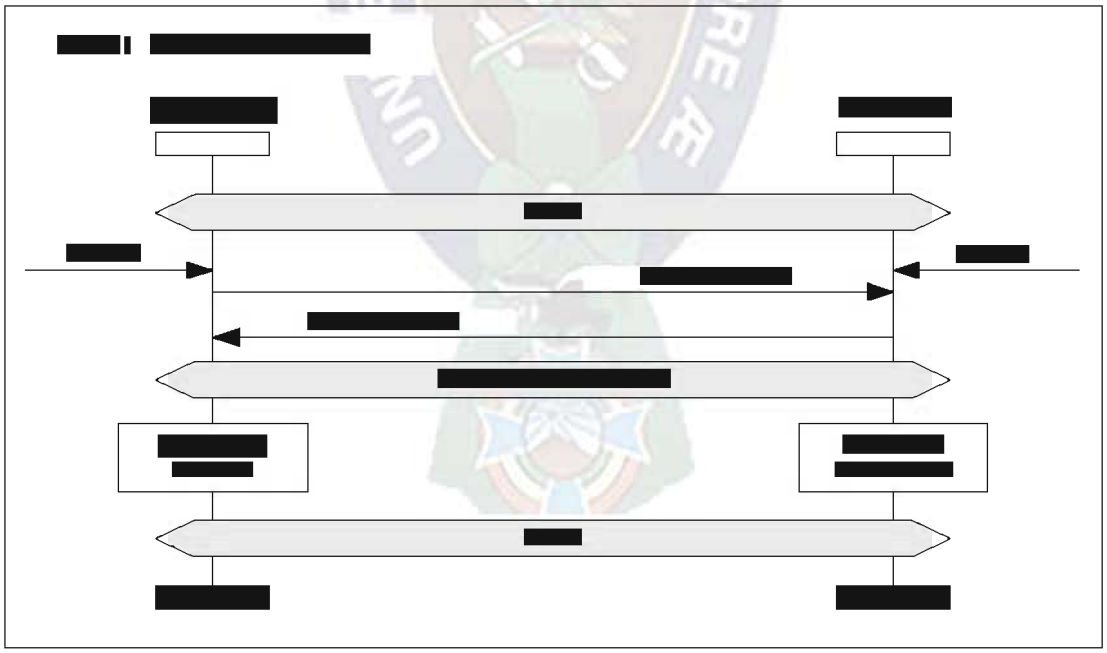


Figura 3.19. Intervención: rastrear ambiente.

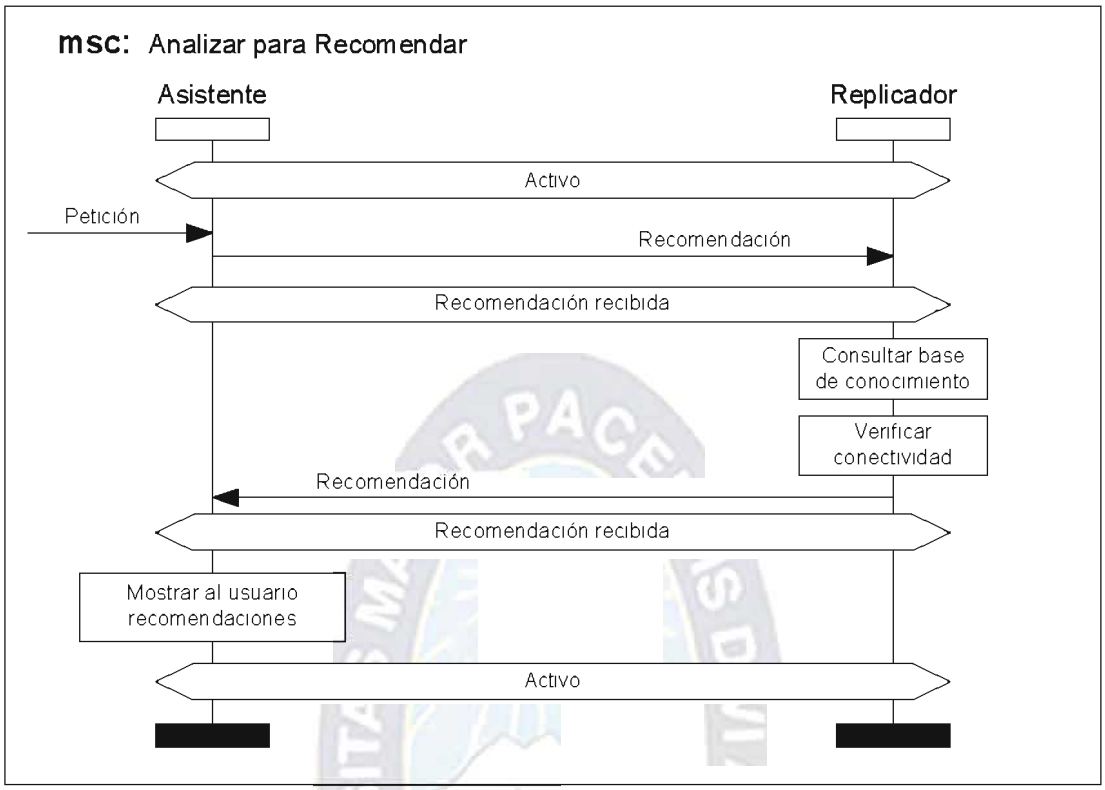


Figura 3.20. Intervención: analizar para recomendar.

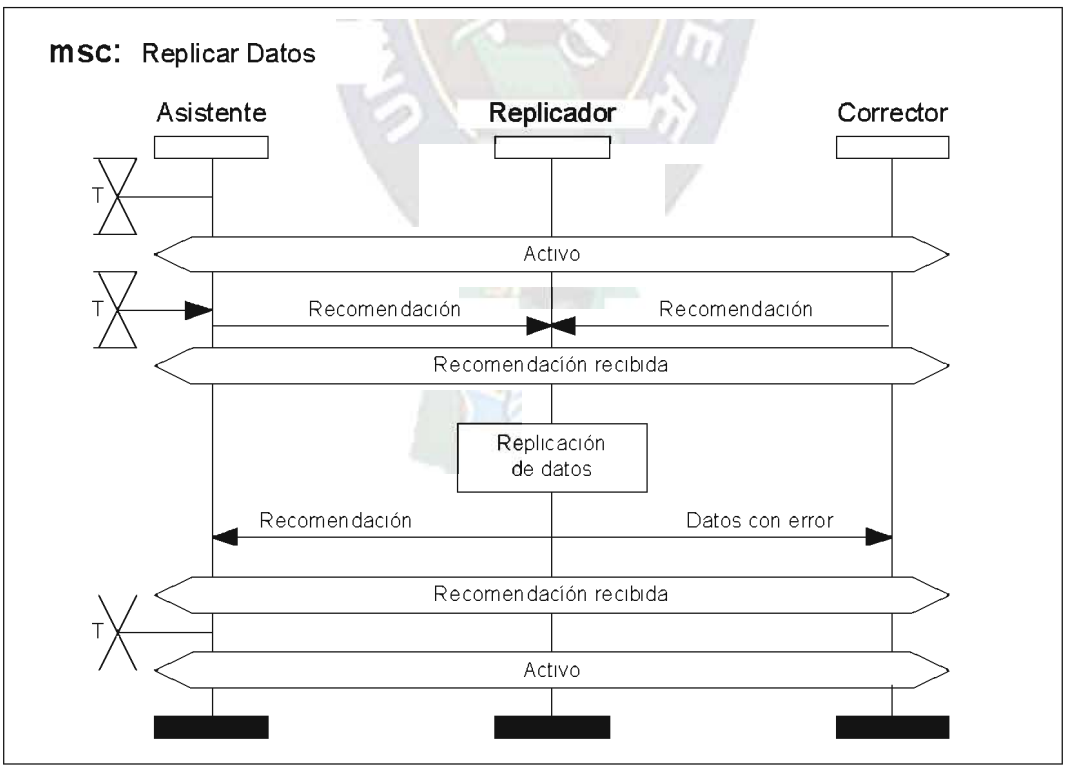


Figura 3.21. Intervención: replicar datos.

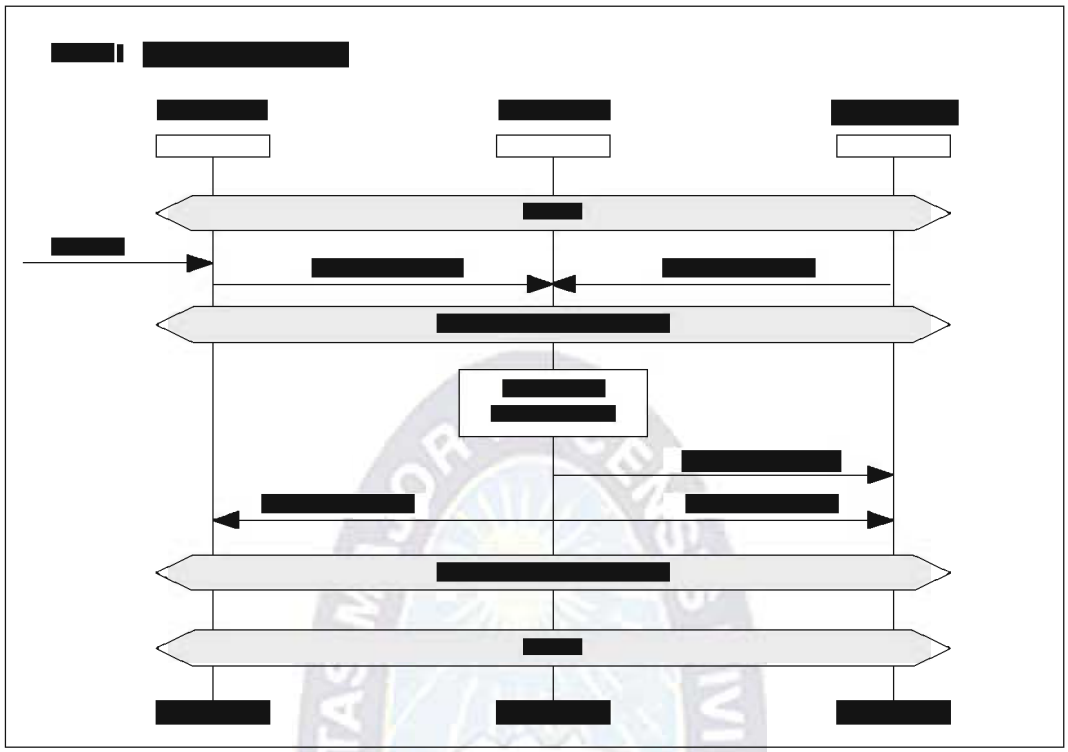


Figura 3.22. Intervención: corregir conflicto.

3.4.7.4. Identificación de los servicios

Los servicios se derivan de las conversaciones. Identificamos los siguientes servicios: Comunicar, Traducir y Procesar. Se presenta la descripción utilizando las plantillas de servicios [Iglesias,1998].

Servicio *Comunicar*
tipo
comunicación
objetivo
comunicar un mensaje a otro agente o al usuario
parámetros-entrada
mensaje
ontología
comunicación

Servicio *Traducir*
tipo
traducción
objetivo
traducir una acción a su representación interna
parámetros-entrada
peticiones del usuario

parámetros-salida
representación interna de la petición
ontología
traducción

Servicio *Procesar*
tipo
distribución
objetivo
distribuir réplicas de transacciones y definiciones en varios sitios destino.
parámetros-entrada
registros modificados
parámetros-salida
registros modificados replicados
ontología
distribución

3.4.8. Modelo de organización multiagente

El modelo de organización de *MAS-CommonKADS* tiene como objetivo analizar desde una perspectiva de grupo las relaciones entre los agentes (tanto software como humanos) que interactúan con el sistema. Con este modelo se especifican las relaciones "estáticas" entre los agentes del sistema (ver figura 3.23).

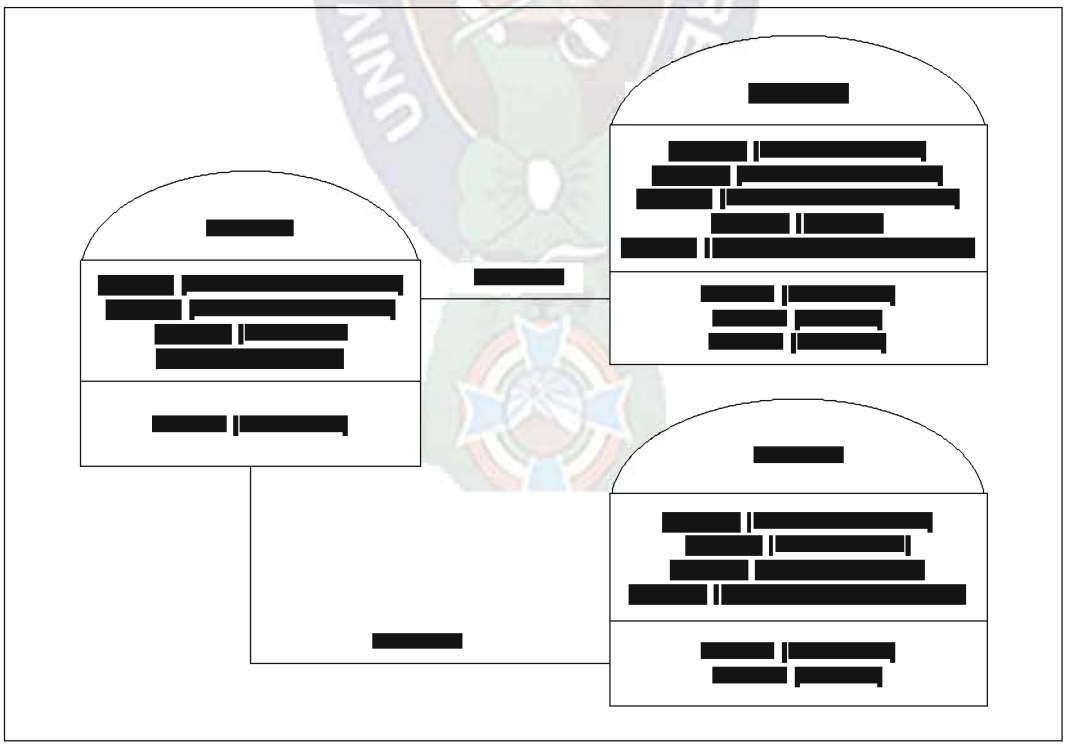


Figura 3.23. Estructura organizativa de agentes.

3.4.8.1. Identificación de los objetos del entorno

Los objetos del entorno son: Sitios de Replicación (bases de datos remotos), Red de Datos, Interfaz de Usuario y Usuario (ver figura 3.24).

3.4.9. Modelo de diseño

El modelo de diseño tiene como objetivo documentar todas las decisiones de diseño y determinar por una parte la arquitectura de la red de agentes, y por otra la arquitectura de agente más adecuada para cada agente. En el modelo de diseño deben considerarse los requisitos no funcionales del sistema y establecerse una relación entre los modelos del análisis y la arquitectura del agente. Los requisitos no funcionales así como la arquitectura concreta que definamos pueden obligar a modificar partes de los modelos del análisis.

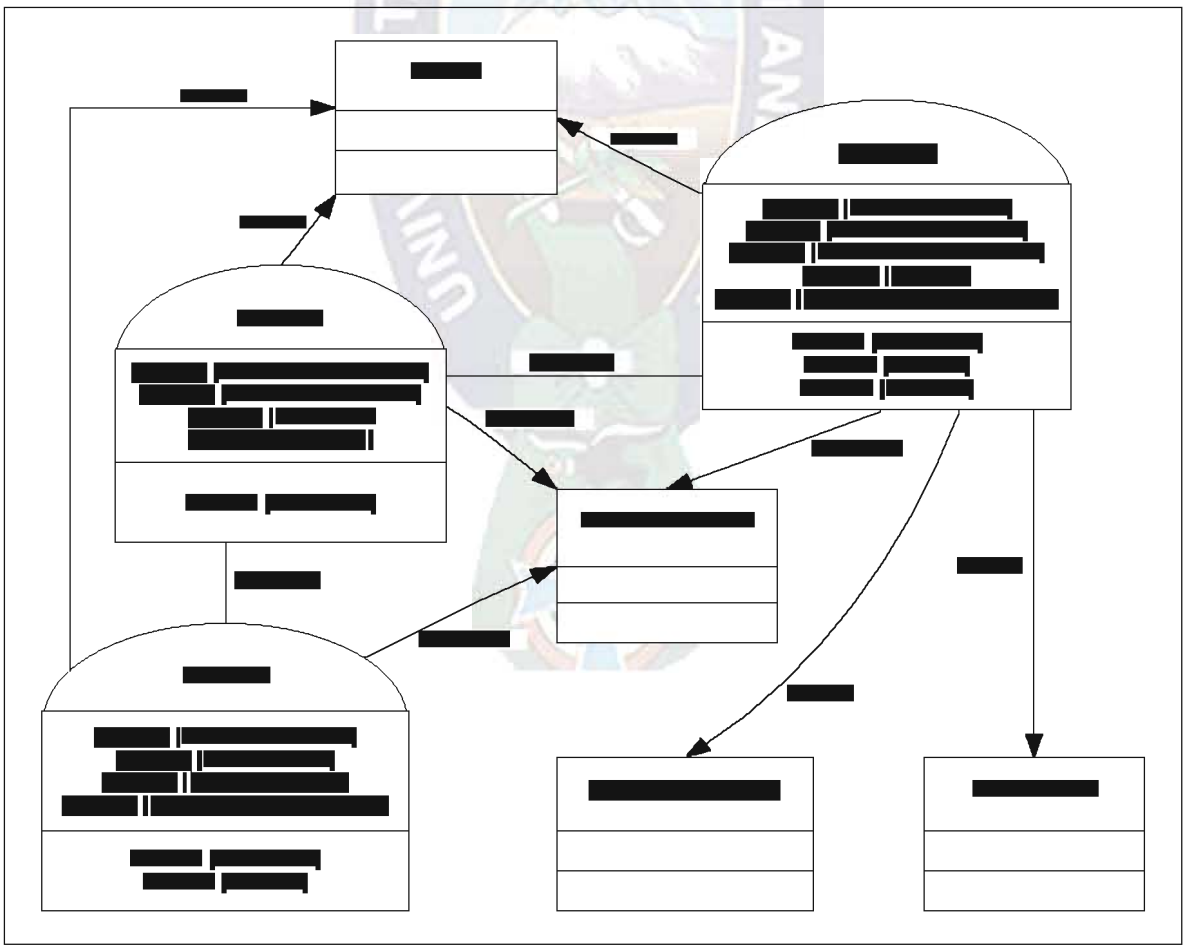


Figura 3.24. Relación de los agentes con los objetos del entorno.

3.4.9.1. Diseño de los agentes

En el diseño de los agentes seleccionamos una arquitectura de agente para cada agente y descomponemos el agente en módulos que se corresponden con módulos de la arquitectura. A continuación se describe el diseño de los agentes del sistema multiagente.

Sistema-Agente *Asistente*
arquitectura
arquitectura reactiva
tiene-subsistema
ninguno

Sistema-Agente *Replicador*
arquitectura
arquitectura híbrida
tiene-subsistema
ninguno

Sistema-Agente *Corrector*
arquitectura
arquitectura híbrida
tiene-subsistema
ninguno

3.4.9.2. Diseño de la plataforma

En este apartado documentamos las decisiones del software y hardware empleado. En la plantilla de plataforma se muestra los detalles del diseño de implementación.

Plataforma *Sistema-Multiagente*
descripción
no se utiliza plataforma multiagente predefinida
usa-lenguaje
Java , HTML, XML, JSP
hardware-requerido
PC con Windows 2000, XP ó 2003
software-requerido
Microsoft Internet Explorer 6, Sun Application Server, Oracle 8i Database
usuario
Administrador de Base de Datos (DBA)

3.5. IMPLEMENTACIÓN DEL MODELO

En esta sección presentamos la descripción del sistema basado en agentes, desarrollado en el trabajo de investigación.

3.5.1. Descripción del prototipo

El Modelo de Replicación de Datos (MRD) fue implementado sobre tres agentes software: *Replicador, Corrector* y *Asistente*. El ambiente de estos agentes es el host del Sitio de Replicación Maestro, y el Administrador de la Base de Datos (DBA), es quien, interactúa con el sistema, realizando las operaciones de:

- *Configuración de Sitios*
- *Configuración de Grupos*
- *Configuración de Objetos*
- *Monitoreo del Entorno de Replicación*

Las tres primeras opciones permiten Listar, Crear, Modificar y Eliminar Sitios, Grupos y Objetos de Replicación, respectivamente. En tanto que la cuarta opción permite visualizar los Elementos del entorno de replicación, Actividad de los sitios de replicación y las Pistas de errores.

El sistema conforma una sociedad multiagente, donde el único agente que puede comunicarse con el usuario final es el *Asistente*, mediante la interfaz de usuario. El agente *Replicador* está encargado de replicar los datos del servidor origen hacia los demás sitios destino, y el agente *Corrector* es el encargado de solventar posibles problemas de inconsistencia en los datos replicados (ver Anexo B).

3.5.2. Plataforma de implementación

A continuación se detallan las herramientas de implementación utilizadas para la construcción del sistema multiagente.

Interfaz del Sistema:
HTML, Microsoft Internet Explorer 6.0

Agentes:

- *Módulo Replicador y Módulo Corrector:*
  Servlets, EJBs Session Bean.
- *Módulo Asistente:*
  MVC (Model/View/Controller) – Struts (EJB, JSP, Servlets)
- *Componente pro-activo:*
  Java Timer (EJB) of Sun Microsystems, Inc.
- *Base de Conocimiento:*
  Java Class, Tables and Views

Servidor de Aplicaciones J2EE:
Sun Java System Application Server 8.1.

Bases de Datos Relacional:
Oracle 8i Enterprise Edition, release 8.1.7.

Para la no dependencia del proveedor, las herramientas utilizadas como plataforma de desarrollo de la aplicación cumplen la especificación J2EE.

CAPÍTULO 4

Evaluación del MRD

4.1. INTRODUCCIÓN

Este capítulo trata la medición de los aspectos centrales del presente trabajo de investigación, la consistencia y oportunidad de los datos replicados. Para ello se parte identificando la técnica de prueba a aplicar y los casos de prueba correspondientes. Posteriormente se arma un escenario próximo a la realidad, referido a la arquitectura de un sistema, el diseño de un modelo de datos, el uso de un aplicativo, un canal de datos limitado y con intermitencia, etc. Finalmente en base a los resultados obtenidos por el prototipo del MRD, se evaluará la eficacia del Modelo de Replicación de Datos.

4.2. CUALIDADES CENTRALES

Son muchas las cualidades que debe poseer un sistema de replicación de datos, sin embargo para fines del presente trabajo, los aspectos a evaluar se centrarán en la consistencia y oportunidad de los datos replicados.

4.2.1. Consistencia

Los procesos de replicación deben garantizar que los datos replicados, entre las bases de datos involucradas, no queden en un estado intermedio de una transacción con parte de los comandos ejecutados y parte no, además que en general no es aceptable que distintos clientes obtengan diferentes resultados al acceder a los mismos datos. Como mínimo no es aceptable si el resultado lleva a una inconsistencia detectable y significativa entre diferentes aplicaciones o incluso dentro de una misma aplicación.

4.2.2. Oportunidad

La oportunidad es una medida intangible que hace referencia a la disposición de la información hacia los usuarios en el plazo más breve posible, a fin de evitar que pierda valor debido a su desfase con la dinámica de su entorno. En un sistema de replicación simétrica y asíncrona, la falta de oportunidad de los datos replicados puede incidir negativamente en la consistencia de los datos.

4.3. ESQUEMA DE PRUEBAS

4.3.1. Técnicas de prueba

Las técnicas de prueba para la evaluación de software proporcionan distintos criterios para generar casos de prueba, estas técnicas se agrupan en:

- *Técnicas de caja blanca o estructurales*. Se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- *Técnicas de caja negra o funcionales*. Realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

Debido a la naturaleza técnica/operativa del presente trabajo, y de acuerdo a los métodos de prueba de modelos mencionados en el Capítulo 2, se vio conveniente aplicar la técnica de caja negra.

4.3.2. Identificación de los casos de prueba

El número de casos de prueba a utilizar será únicamente el necesario para la demostración de la hipótesis planteada en el Capítulo 1, sin embargo cada caso de prueba cuenta con las condiciones necesarias y suficientes para dicha demostración.

4.3.2.1. Caso de prueba: Consistencia

Para este caso de prueba se tomará como referencia un "*Sistema de cobro de impuestos a la propiedad de vehículos automotores*", en este intervienen dos bases de datos relacionales (BD1 y BD2) y un aplicativo que efectúa (simula) el pago de impuestos. El procedimiento consiste en que simultáneamente el aplicativo es ejecutado en ambas BD y con el mismo conjunto de datos. El propósito es medir el porcentaje de transacciones (pago de impuestos) que llegan a replicarse con éxito en la BD destino, y los que no llegan a replicarse o entran en conflicto con otras transacciones, deberán ser marcadas y/o registradas en alguna tabla auxiliar, por tratarse de pagos duplicados; todo este proceso deberá ser transparente para el aplicativo, puesto que será el prototipo del MRD quien tome decisiones para ello. La configuración del entorno de replicación será asíncrono, tipo de replicación simétrico (ambos sentidos), tipo de actualización persistente y un grupo de replicación que incluya a todas las tablas involucradas en cada transacción.

4.3.2.2. Caso de prueba: Oportunidad

Para este caso de prueba, de manera similar, se tomará como referencia el sistema arriba mencionado, en este intervienen también dos bases de datos relacionales (BD1 y BD2) y el mismo aplicativo que simula el pago de impuestos. El procedimiento consiste en ejecutar el aplicativo en una sola base de datos (BD1) registrando los tiempos de las transacciones y de sus correspondientes replicaciones. El propósito es medir el tiempo de latencia[16] en la base de datos destino (BD2), puesto que a esta prueba se adiciona cortes en la red de datos cada 30 segundos de períodos similares. La configuración del entorno de replicación será similar al caso anterior cambiando solamente el tipo de replicación a asimétrico (un solo sentido).

4.3.3. Escenario para los casos de prueba

Ejemplo único.- Se describe una arquitectura de sistema distribuido (WAN), con un servicio frame relay de 16kb/seg., que interconecta 2 BDs Oracle (Oracle® Database), uno de los

[16] Tiempo de desactualización del dato replicado respecto al dato modificado.

cuales es el sitio maestro de replicación y el otro un sitio esclavo. Se elaboró un pequeño modelo de datos para ver la relación entre las tablas (transacción) que forman un grupo de replicación. El aplicativo carga 50 datos ficticios de vehículos automotores con deuda y efectúa el pago de impuestos cada 3 segundos, cubriendo a todos en un período de 2,5 minutos.

4.3.3.1. Arquitectura del sistema

Se propone una arquitectura de sistema distribuido para realizar los correspondientes casos de prueba, la misma se ilustra en la figura 4.1.

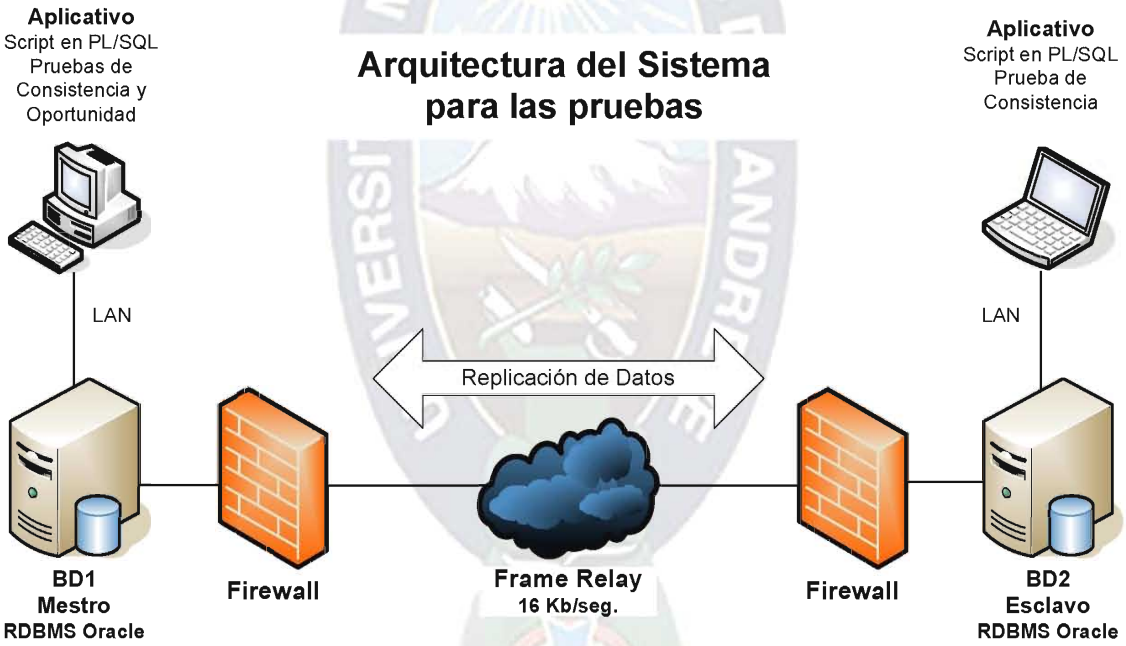


Figura 4.1. Ejemplo de una Arquitectura del Sistema para los casos de prueba.

4.3.3.2. Modelo de datos

Se plantea un modelo de datos (ver figura 4.2), donde las entidades de color gris oscuro forman parte del único grupo de replicación, y el correspondiente proceso de replicación incluirá a todos los objetos de dicho grupo. Por otro lado la entidad de color gris claro

representa la tabla auxiliar en caso de conflictos. Y finalmente la entidad de color blanco representa la tabla del universo de vehículos.

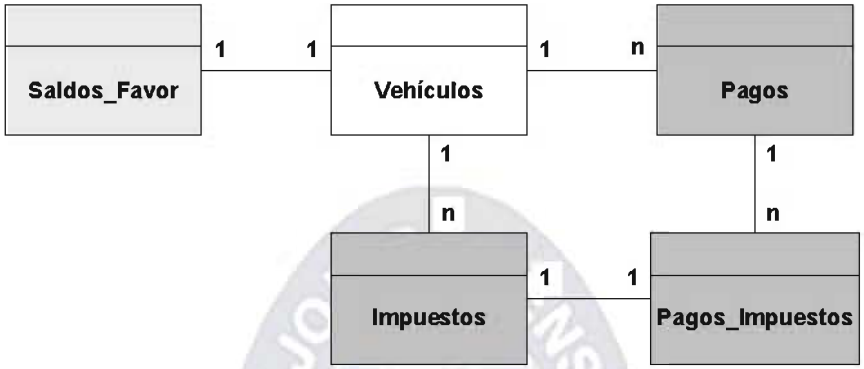


Figura 4.2. Ejemplo de un Modelo de Datos para los casos de prueba.

4.3.3.3. Aplicativo

El aplicativo que nos permitirá simular los pagos de impuestos consiste en un script escrito en PL/SQL de Oracle® Database (ver figura 4.3), éste recupera los datos de 50 vehículos y entra en un bucle para realizar las inserciones en las tablas involucradas en la transacción (grupo de replicación). Si la transacción tiene éxito el prototipo replica toda la transacción, si no tiene éxito (conflicto) encamina la transacción a la tabla auxiliar, esto fue previamente definido al momento de crear el grupo de replicación.

El script fue ejecutado sobre la herramienta Oracle® SQL*Plus versión 9.2.0.4, y algunos de los comandos utilizados para registrar los tiempos son descritos a continuación:

- **Spool Test_MRD.log.-** Crea y abre un archivo tipo texto llamado `Test_MRD.log`, donde se registran todas las salidas en pantalla que produce el script.
- **Set timing on.-** Establece un cronómetro asociado a la presente sesión de cliente, donde cada vez que es ejecutada una sentencia SQL escribe en pantalla el tiempo, en centésimas de segundo, que duró dicha ejecución.
- **Select to_char(sysdate,'HH24:MI:SSxFF') from dual.-** Extrae la hora, minuto, segundo y centésimas de segundo del servidor de BD y lo escribe en pantalla. Con la misma instrucción se obtiene el tiempo de latencia del registro replicado, pero esta vez ejecutado en el servidor de BD destino.

- **Delay (3).-** Espera tres segundos para continuar con la siguiente instrucción.
- **Set timing off.-** Elimina el cronómetro de la sesión de cliente.
- **Spool off.-** Cierra el archivo tipo texto abierto previamente.

```
spool Test_MRD.log
set timing on
declare
  /* recuperación de la deuda de los 50 vehículos para las correspondientes pruebas */
  dummy  number(1);
  cursor cur_vehi is
   select v.num_sec vnumsec, i.num_sec inumsec, i.cod_alcaldia, i.monto_determinado
     from vehiculos v, impuestos i
    where v.num_sec=i.nsec_vehiculo and i.gestion=2005
      and i.estado='PE' and rownum<=50;
begin
  /* Bucle de 50 iteraciones */
  select to_char(sysdate,'HH24:MI:SSxFF') as 'Inicio de la replicación' from dual;
  for cur in cur_vehi loop
   delay (3);  /* cada 3 segundos se realiza un pago */
   begin
    select 1 into dummy
      from impuestos
     where nsec_vehiculo=cur.vnumsec and gestion=2005 and estado='AC';
   exception
     when no_data_found then
       dummy := 0;
   end;
   /* simplificación de un pago al contado y total */
   if dummy=0 then
    insert into pagos values (pago_sec.nextval, cur.vnumsec, cur.cod_alcaldia,
      sysdate, 'PE');
    insert into pagos_impuestos values (pagoimpuesto_sec.nextval,
      inum_sec, pago_sec.currval, 'AC');
    insert into impuestos values (impuesto_sec.nextval, cur.vnumsec,
      cur.cod_alcaldia, 2005, cur.monto_determinado,'AC');
    select to_char(sysdate,'HH24:MI:SSxFF') as 'Fin de la n-ésima iteracion'
      from dual;
   end if;
/*
** Las anteriores inserciones en la primera BD, serán replicadas a la segunda BD
** por el prototipo del Modelo de Replicación de Datos (MRD)
*/
  end loop;
end;
/
set timing off
spool off
```

Figura 4.3. Script en PL/SQL que efectúa el pago de impuestos de automotores.

4.4. RESULTADO DE LAS PRUEBAS

Una vez generados los archivos de tipo texto, en ambos servidores de BDs, los datos obtenidos fueron tabulados y graficados para una correcta interpretación de los resultados.

En lo que respecta a la consistencia de los datos replicados, simplemente se realizó una consulta de los registros replicados en el repositorio del prototipo del MRD, en la que se

halló que de los 50 vehículos, solo uno tuvo conflictos de replicación que no fue resuelto (ver tabla 4.1), por lo que el 98% de los datos fueron replicados con éxito, y el 2% falló en el proceso (ver figura 4.4).

Nro. dato	Código del vehículo	Tablas replicadas		
		Pagos	Pagos_Impuestos	Impuestos
1	1019	ok	Ok	ok
2	1023	ok	ok	ok
3	1053	ok	ok	ok
4	1058	ok	ok	ok
5	1127	ok	ok	ok
6	1128	ok	ok	ok
7	1138	ok	ok	ok
8	1167	ok	ok	Ok
9	1187	ok	ok	Ok
10	1196	ok	ok	Ok
11	1198	ok	ok	Ok
12	1202	ok	ok	Ok
13	1224	ok	ok	Ok
14	1322	ok	ok	Ok
15	1344	ok	ok	Ok
16	1375	ok	ok	ok
17	1384	ok	ok	ok
18	1391	ok	ok	ok
19	1394	ok	ok	ok
20	1484	ok	ok	ok
21	1520	ok	ok	ok
22	1529	ok	ok	ok
23	1549	ok	ok	ok
24	1561	ok	ok	ok
25	1567	ok	ok	ok
26	1600	ok	ok	ok
27	1613	ok	ok	ok
28	1617	ok	ok	ok
29	1659	ok	ok	ok
30	1688	ok	ok	ok
31	1714	ok	ok	ok
32	1731	ok	ok	ok
33	1762	ok	ok	ok
34	1794	ok	ok	ok
35	1850	ok	ok	ok
36	1851	ok	ok	ok
37	1852	ok	ok	ok
38	1916	ok	ok	ok
39	1942	ok	ok	ok
40	1991	ok	ok	ok
41	2002	ok	ok	ok
42	2012	ok	ok	ok
43	2095	ok	ok	ok
44	2114	ok	ok	ok
45	2138	ok	ok	ok
46	2146	ok	ok	ok
47	2156	no	no	no
48	2184	ok	ok	ok
49	2191	ok	ok	ok
50	2195	ok	ok	ok

Tabla 4.1. Resultados de la prueba de Consistencia.

Donde **ok** representa el registro replicado
no representa el registro no replicado

		Tiempos de replicación (segs.)									
Nro.	Código del	Pagos			Pagos_Impuestos			Impuestos			Promedio de
dato	vehículo	BD1	BD2	DP	BD1	BD2	DPI	BD1	BD2	DI	diferencia
1	1019	0,21	2,51	2,30	0,19	2,50	2,31	0,20	2,53	2,33	2,31
2	1023	0,19	2,56	2,37	0,18	2,56	2,38	0,25	2,49	2,24	2,33
3	1053	0,11	2,01	1,90	0,11	2,01	1,90	0,17	2,01	1,84	1,88
4	1058	0,16	2,57	2,41	0,19	2,57	2,38	0,11	2,57	2,46	2,42
5	1127	0,15	3,20	3,05	0,16	3,20	3,04	0,15	3,20	3,05	3,05
6	1128	0,07	3,10	3,03	0,07	3,05	2,98	0,15	3,15	3,00	3,00
7	1138	0,12	2,90	2,78	0,10	2,90	2,80	0,12	2,87	2,75	2,78
8	1167	0,11	2,41	2,30	0,10	2,41	2,31	0,11	2,41	2,30	2,30
9	1187	0,13	2,43	2,30	0,13	2,42	2,29	0,16	2,78	2,62	2,40
10	1196	0,09	2,39	2,30	0,09	2,39	2,30	0,17	2,39	2,22	2,27
11	1198	0,10	32,45	32,35	0,09	32,45	32,36	0,15	32,45	32,30	32,34
12	1202	0,05	32,35	32,30	0,04	32,36	32,32	0,06	32,36	32,30	32,31
13	1224	0,06	32,45	32,39	0,06	32,45	32,39	0,27	32,45	32,18	32,32
14	1322	0,06	32,89	32,83	0,06	32,89	32,83	0,06	32,89	32,83	32,83
15	1344	0,04	32,99	32,95	0,04	32,99	32,95	0,04	32,99	32,95	32,95
16	1375	0,07	32,04	31,97	0,07	33,10	33,03	0,03	33,10	33,07	32,69
17	1384	0,08	32,31	32,23	0,07	32,31	32,24	0,30	32,31	32,01	32,16
18	1391	0,04	32,11	32,07	0,03	32,11	32,08	0,04	32,67	32,63	32,26
19	1394	0,03	32,33	32,30	0,03	32,33	32,30	0,03	32,33	32,30	32,30
20	1484	0,12	32,56	32,44	0,12	32,54	32,42	0,12	32,56	32,44	32,43
21	1520	0,17	2,70	2,53	0,17	2,69	2,52	0,16	2,80	2,64	2,56
22	1529	0,16	1,90	1,74	0,17	1,90	1,73	0,16	1,90	1,74	1,74
23	1549	0,15	1,80	1,65	0,15	1,80	1,65	0,15	1,80	1,65	1,65
24	1561	0,13	1,90	1,77	0,13	1,90	1,77	0,13	1,90	1,77	1,77
25	1567	0,13	2,00	1,87	0,14	2,00	1,86	0,15	2,00	1,85	1,86
26	1600	0,12	2,10	1,98	0,12	2,10	1,98	0,19	2,16	1,97	1,98
27	1613	0,11	2,67	2,56	0,11	2,67	2,56	0,11	2,99	2,88	2,67
28	1617	0,09	2,15	2,06	0,09	2,15	2,06	0,09	2,15	2,06	2,06
29	1659	0,09	2,03	1,94	0,08	2,03	1,95	0,09	2,03	1,94	1,94
30	1688	0,10	1,90	1,80	0,10	1,90	1,80	0,11	1,90	1,79	1,80
31	1714	0,06	32,17	32,11	0,06	32,15	32,09	0,06	32,17	32,11	32,10
32	1731	0,03	32,56	32,53	0,03	32,56	32,53	0,03	32,87	32,84	32,63
33	1762	0,06	32,36	32,30	0,06	32,36	32,30	0,06	32,80	32,74	32,45
34	1794	0,07	32,37	32,30	0,07	32,37	32,30	0,07	32,37	32,30	32,30
35	1850	0,21	32,56	32,35	0,19	32,56	32,37	0,21	32,56	32,35	32,36
36	1851	0,11	32,67	32,56	0,11	32,67	32,56	0,13	32,87	32,74	32,62
37	1852	0,11	32,95	32,84	0,11	32,95	32,84	0,12	32,56	32,44	32,71
38	1916	0,08	32,78	32,70	0,09	32,71	32,62	0,08	32,78	32,70	32,67
39	1942	0,06	32,45	32,39	0,06	32,45	32,39	0,06	32,45	32,39	32,39
40	1991	0,05	32,40	32,35	0,05	32,40	32,35	0,07	32,56	32,49	32,40
41	2002	0,04	2,56	2,52	0,04	2,56	2,52	0,05	2,56	2,51	2,52
42	2012	0,12	2,56	2,44	0,12	2,56	2,44	0,17	2,45	2,28	2,39
43	2095	0,14	2,45	2,31	0,14	2,45	2,31	0,19	2,60	2,41	2,34
44	2114	0,17	2,48	2,31	0,17	2,48	2,31	0,15	2,75	2,60	2,41
45	2138	0,09	2,46	2,37	0,08	2,46	2,38	0,09	2,17	2,08	2,28
46	2146	0,05	2,43	2,38	0,05	2,43	2,38	0,06	2,13	2,07	2,28
47	2156	0,08	2,11	2,03	0,08	2,09	2,01	0,08	2,11	2,03	2,02
48	2184	0,11	2,56	2,45	0,11	2,56	2,45	0,11	2,59	2,48	2,46
49	2191	0,07	2,57	2,50	0,08	2,57	2,49	0,07	2,49	2,42	2,47
50	2195	0,09	2,89	2,80	0,09	2,87	2,78	0,10	2,89	2,79	2,79

Tabla 4.2. Resultados de la prueba de Oportunidad.

Donde **BD1** tiempo de escritura en la base de datos origen de la transacción
BD2 tiempo de escritura en la base de datos destino de la replicaron
DP diferencia de tiempo de escritura entre el pago replicado menos el pago original
DPI diferencia de tiempo de escritura entre el pago_impuesto replicado menos el pago_impuesto original
DI diferencia de tiempo de escritura entre el impuesto replicado menos el impuesto original

En cuanto a la oportunidad de los datos replicados, los resultados fueron obtenidos del archivo tipo texto que generó el script (ver figura 4.3) ejecutado en la BD origen, y con un script similar, pero solo de lectura, se obtuvieron los tiempos de replicación en la BD destino (ver tabla 4.2), y sabiendo que la latencia ideal es de 0 seg. con la red de datos operable y de 30 seg. con el corte programado en la red, se observa que los tiempos de replicación son muy próximos a dichos valores (ver figura 4.5).

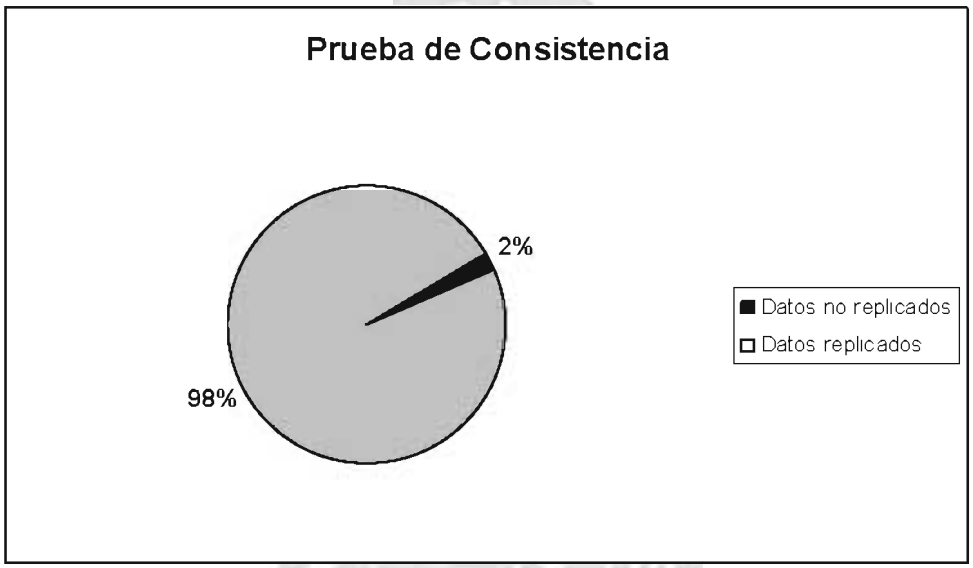


Figura 4.4. Gráfico de resultados de la prueba de Consistencia.

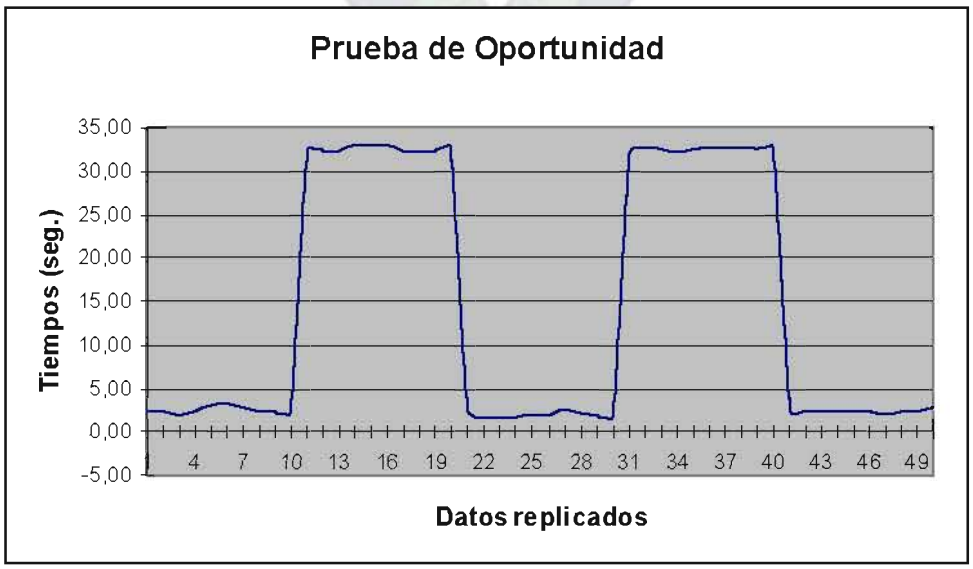


Figura 4.5. Gráfico de resultados de la prueba de Oportunidad.

4.5. EVALUACIÓN DEL MODELO

Una vez obtenidos los resultados de las pruebas, se deben buscar indicadores que muestren el grado de error en los datos replicados, en cuanto a la consistencia y oportunidad.

En cuanto a la consistencia, se puede observar fácilmente la relación entre ellos, como se lo muestra a continuación:

Conjunto de datos: 50 registros

Datos replicados: 49 registros

Datos no replicados: 1 registro

Por lo tanto, será la eficacia de la consistencia $C = 98\%$, contra un $e = 2\%$ de error.

En un sistema de replicación síncrono la latencia (L) es cero, sin embargo la transacción finaliza cuando el dato está replicado en todas las BDs involucradas, por ello el tiempo de la transacción inicial aumenta considerablemente. Esto dependerá del ancho de banda de la red, saturación del canal, protocolo de comunicación, cortes imprevistos en la red, lectura/escritura previa en la BD de origen, volumen de la información a enviar, etc.

La oportunidad esta relacionada directamente al tiempo de latencia, y sabiendo que se tiene un ancho de banda de 16 kb/seg. con cortes de comunicación de 30 seg. cada 30 seg., se registró los siguientes tiempos de latencia máximos y mínimos:

Diferencia mínima: *sin corte* 1,65 seg., *con corte* 32,10 – 30 = 2,10 seg.

Diferencia máxima: *sin corte* 3,05 seg., *con corte* 32,95 – 30 = 2,95 seg.

La latencia varía entre 1,65 y 3,05 seg., entonces la latencia promedio $L = 2{,}35$ seg.

CAPÍTULO 5

Conclusiones y Recomendaciones

5.1. ESTADO DE LA HIPÓTESIS

Según la hipótesis inicialmente planteada en el Capítulo 1, sección 1.4, menciona que el MRD permitirá mantener un alto grado de Consistencia y Oportunidad en los datos replicados en entornos tolerantes a fallos, ésta no está absolutamente demostrada, puesto que si bien, las pruebas preliminares del prototipo arrojaron resultados interesantes, éstos no son definitivos.

Según normas internacionales, un alto grado de una métrica en general, se mide en el porcentaje del 99,98%, y habiendo el MRD obtenido resultados relativamente menores en las pruebas de consistencia y oportunidad, se puede concluir que la hipótesis será cumplida cuando la latencia de replicación sea la mínima posible. Sin embargo, reconocemos que el uso de metodologías orientadas a agentes, es una interesante alternativa para solucionar problemas inherentes a la replicación de datos.

5.2. CONCLUSIONES

Las conclusiones estarán enmarcadas sobre los objetivos inicialmente trazados:

- Respecto al objetivo general, se ha comprobado que el MRD mantiene un aceptable grado de Consistencia y Oportunidad de los datos replicados en sistemas conversacionales tolerantes a fallos, siendo además una herramienta integrada a soluciones de escalabilidad, y a un bajo costo económico y operativo.

En cuanto a los objetivos específicos:

- Se diseñó un agente de replicación (*Replicador*) de acuerdo a las cualidades mínimas requeridas, como ser: replicación de distintos tipos de objetos (datos y

procedimientos almacenados), replicación exclusivamente asíncrona, replicación simétrica y asimétrica, y la de generar ciertas recomendaciones al usuario final.

- Se diseñó un agente para la resolución de conflictos de replicación (*Corrector*), que sin embargo, no garantiza en su totalidad la resolución de conflictos, puesto que los resultados de las pruebas arrojaron ciertas inconsistencias a mayor número de transacciones.
- Se diseñó un agente asistente al usuario final (*Asistente*), el cual evidentemente colabora en las tareas requeridas para la administración y monitoreo del entorno de replicación.

5.3. RECOMENDACIONES

Esperamos que este estudio y sus resultados hayan contribuido a una línea de investigación y puedan servir como base para trabajos futuros en el área. Algunas posibles áreas de interés podrían ser las siguientes.

Siguiendo la línea de Ingeniería de Software para agentes se podría ampliar el prototipo del MRD para la replicación de Instancias de Objetos dentro del paradigma Orientado a Objetos. También podría ser de interés la replicación entre Bases de Datos Orientados a Objetos, donde el acceso y modificación a los datos no es, necesariamente, mediante el SQL.

Se podrían desarrollar aplicaciones de agentes en varias áreas, entre ellas algunas significativas podrían ser: el área de trabajo Colaborativo y el área de Informática Educativa sobre la cual existen varios trabajos de tesis en la Biblioteca Especializada de la Carrera de Informática – U.M.S.A.

Bibliografía

[Attaluri,1995] Attaluri, J.; "*The CORDS multidatabase*", project. IBM Systems Journal 34, pp. 39-62, 1995.

[Boehm,1988] Boehm, B.; "*A spiral model of software development and enhancement*", In R. Donald, editor, Software Management, pages 120–131. IEEE Computer Society Press, 1993, Reprinted from em Computer, Vol. 21, No. 5, pp. 61-72, 1988.

[Booch,1994] Booch, G.; "*Object Oriented Analysis and Design with Applications*", The Benjamin/Cummings Publishing Company, 1994.

[Brazie,1997] Brazier, F.M.T, Dunin-Keplicz,B.M, Treur,J and Verbrugge, R.; "*Modelling Internal Dynamic Behaviour of BDI Agents*". Proceedings of the Third International Workshop on Formal models of Agents, Lectures Notes Springer Verlag, 1997.

[BRJ,1997] Booch, G., Rumbaugh, J., Jacobson, I.; "*Unified Modeling Language*". Addison Wesley Longman, Inc. Massachusetts, USA, 1997.

[Brooks,1986] Brooks, R. A.; "*A robust layered control system for a mobile robot*", IEEE Journal of Robotics and Automation, Vol. 2, N° 1, pp. 14-23, 1986.

[Buretta,1997] Buretta, M.; "*Data Replication – Tools and Techniques for managing distributed information*", Editorial Wiley, 1997.

[Burmeis,1996] Burmeister, B.; "*Models and Methodology for agent-oriented analysis and design*", In K. Fischer, editor, Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems, DFKI Document D-96-06, 1996.

[Couloris,2001] Coulouris, G., Dollimore, J. and Kindberg, T.; "*Distributed Systems: Concepts and Design*", Tercera Edición.Addison-Wesley, Pearson Education, 2001.

[Cullas,2001] Cullas, V. J.; "*Sistema de apoyo a la gestión de un ambiente de replicación de datos en un sistema distribuido de bases de datos*", Tesis para optar por el Título de Ingeniero Informático, PUCP-PERU, 2001.

[Elmasri,2000] Elmasri, R and Navathe, S. B.; "*Fundamentals of database systems*", Addison-Wesley, Third Edition, 2000.

[George,1995] Georgeff, M. and Rao, A.; "*BDI Agents: From Theory to Practice*", Proceedings of the First Internacional Conference on Multi-Agent Systems (ICMAS – 95), San Francisco, USA, Junio, 1995.

[Ghezzi,1991] **Ghezzi, C., Jazayeri, P. y Mandrioli, D.; "*Fundamentals of software Engineering*", Englewood Cliffs, N.J: Prentice Hall, 1991.**

[Giret,1999] Giret, A.; "*Uso de la Orientación a Objetos para el diseño e implementación de Agentes*", Tesis de Ingeniería Informática. Universidad Católica Nuestra Señora de la Asunción. Asunción Paraguay, 1999.

[Glaser,1996] Glaser, N.; "*Contribution to Knowledge Modelling in a Multi-Agent Framework (the CoMoMAS Approach)*", PhD thesis, L'Universitité Henri Poincaré, Nancy I, Francia, Noviembre, 1996.

[Grosse,1995] Grosse, A. ; "*Position Paper*", http://www.AgentLink.org DAI Lab., 1995.

[Harel,1987] Harel, D.; "*Statecharts: A visual formalism for complex systems*", Sci. Computer Program, Vol. 8, N° 1, pp. 231-247,1987.

[Hermans,1996] Hermans, B.; "*Intelligent Software Agents On The Internet: An Inventory Of Currently Offered Funcionality In The Information Society & A Prediction Of (Near) Future Developments*", Tilburg University, The Netherlands, 1996.

[Iglesias,1998] Iglesias, C. A.; "*Definición de una metodología para el desarrollo de sistemas multiagente*", PhD thesis, Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Enero 1998.

[Jacobson,1997] Jacobson, I., Christerson, M., Jonsson, P., and Övergaard, G.; "*Object-Oriented Software Engineering*", A Use Case Driven Approach, ACM Press, 1997.

[Kinny,1996] Kinny, D., Georgeff, M., and Rao, A.; "*A methodology and modelling technique for system of BDI agents*", In W. Van der Velde and J. Perram, editors, Agents Breaking Away: Proceedings f the Seventh European Workshop on Modelling Autonomous Agents in a Multy-Agent World MAAMAW'96, Springer-Verlag: Heidelberg, Germany, 1996.

[Maes,1991] Maes, P.; "*Agents that reduce work and information overload*", Communication of the ACM, Vol. 37, N°. 7, pp. 31-40, 1991.

[McGraw,1989] McGraw, K. and Harbinson, K.; "*Knowledge Adquisition. Principles and Guidelines*", Prentice Hall, 1989.

[Microsoft,2001] Microsoft Documentation; "*Libros en pantalla de SQL Server*", 2001.

[Minsky,1994] Minsky, M.; "*A Conversation with Marvin Minsky about Agents*", Communication of the ACM. Vol. 37, N°. 7, pp 81-91, 1994.

[Oracle,1999] Oracle Documentation; "*Oracle8i Replication*", Release 2 (8.1.6). A76959-01, 1999.

[Osborne,1999] Osborne McGraw-Hill; "*MCDBA SQL Server 7.0 Administration Study Guide*", Osborne McGraw-Hill, U.S.A., 1999.

[Ramos,1997] Ramos, E.; *"Conocimiento: Adquisición y Técnicas para grupos"*, Escuela de Computación, Facultad de Ciencias, Universidad Central de Venezuela, 1997.

[Romero,1999] Romero, S.; "*Procesamiento de consultas en sistemas de multibase de datos*", Tesis, Departamento de Sistema Computacionales, Fundación Universidad de las Américas, Cholula Puebla, 1999.

[Rumbau,1991] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, V.; "*Object-Oriented Modeling and Design*", Prentice-Hall, 1991.

[Russell,1995] Russell, S. J. and Norvig P.; "*Artificial Intelligence: A Modern Approach*", Englewood Cliffs, NJ: Prentice Hall, 1995.

[Shoham,1993] Shoham, Y.; "*Agent-oriented programming*", Artificiall Intelligence, Vol. 60, N°. 1, pp. 51-92, Marzo 1993.

[Schreiber,1994] Schreiber, A. Th., Weilinga, B. J., Akk.ermans, J. M., and Van de Velde, W.; "*CommonKads: A comprehensive methodology for KBS development*",University of Amsterdam, Netherlands Energy Research Foundation ECN and Free University of Brussels, 1994.

[Woold,1995] Wooldridge, M. and Jennings N. R.; "*Agent Theories, Architectures, and Languages: a Survey*", in Wooldridge and Jennings Eds., Intelligence Agents, Berlin: Springer-Verlag, Vol. 1, N° 22, 1995.

ANEXO A

Descripciones Gráficas y Textuales

Para el desarrollo de sistemas multiagentes basados en la metodología *MAS-CommonKADS* [Iglesias,1998] es necesario el uso de plantillas pre-definidas y casos de uso [Jacobson,1997], por lo cual, en este anexo se detallan las descripciones gráficas y textuales del desarrollo del Modelo de Replicación de Datos (MRD). Para evitar una extensión excesiva, no se desarrollará por completo, sino las descripciones que se consideren más representativas.

A.1. EXTENSIÓN DE CASOS DE USO DEL ACTOR *DBA*

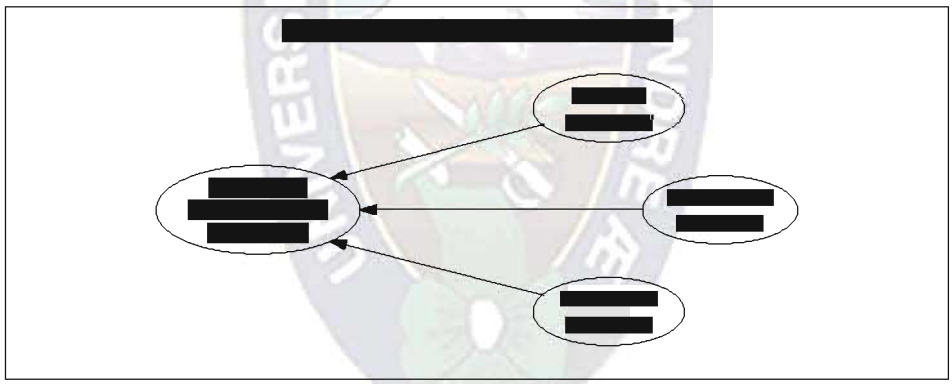


Figura A.1. Extensión del caso de uso: Configurar el esquema de replicación.

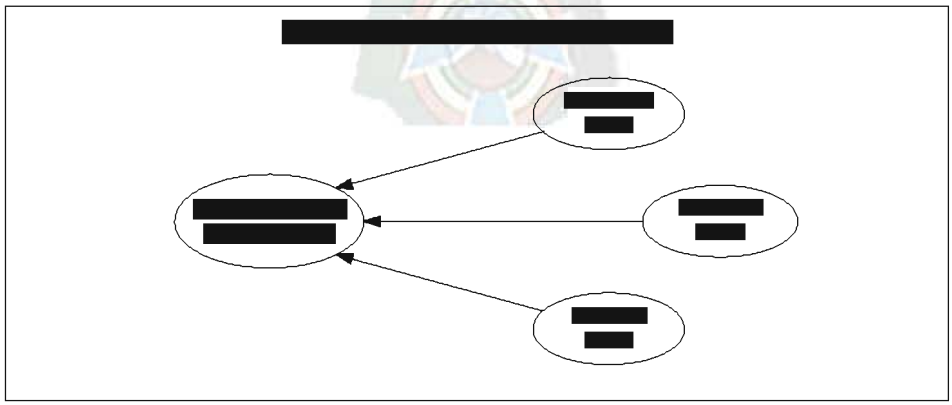


Figura A.2. Extensión del caso de uso: Configurar sitios de replicación.

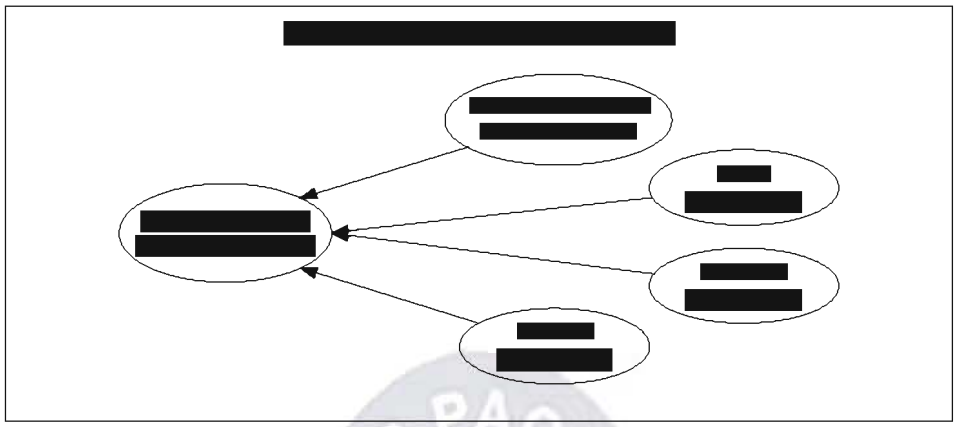


Figura A.3. Extensión del caso de uso: Configurar grupos de replicación DML.

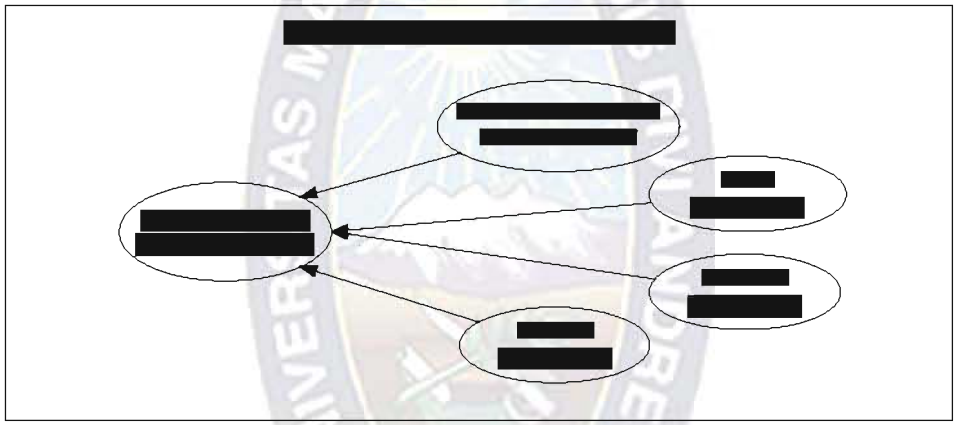


Figura A.4. Extensión del caso de uso: Configurar grupos de replicación DDL.

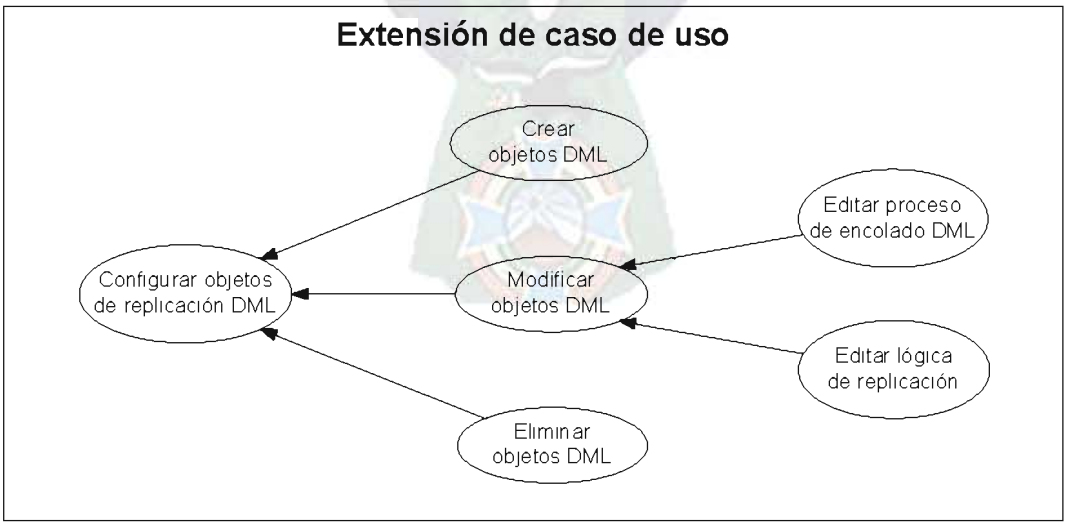


Figura A.5. Extensión del caso de uso: Configurar objetos de replicación DML.

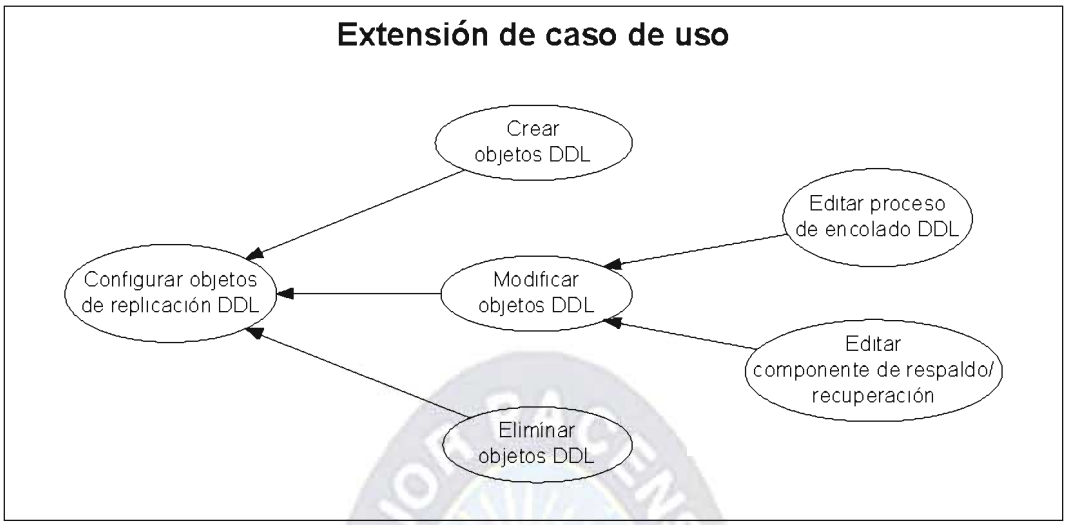


Figura A.6. Extensión del caso de uso: Configurar objetos de replicación DDL.

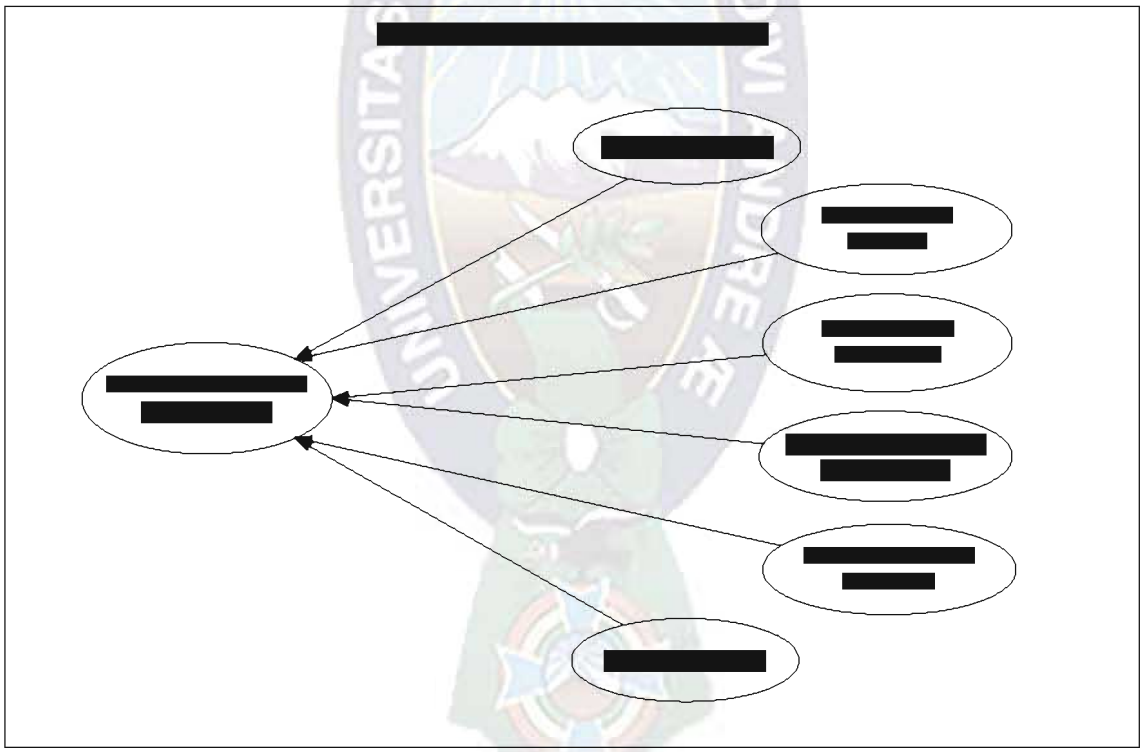


Figura A.7. Extensión del caso de uso: Monitorear el entorno de replicación.

A.2. DESCRIPCIÓN TEXTUAL DE CASOS DE USO

Caso de uso *Configurar el esquema de replicación*

resumen

el DBA configura el esquema de replicación, donde podrá crear, modificar y eliminar el esquema de replicación.

actores

DBA

precondiciones

ser usuario administrador, instalar el software del MRD.

descripción

el usuario selecciona en el menú la opción Configurar Esquema, en la vista de configuración el usuario puede crear , modificar y eliminar el esquema de replicación. Durante la ejecución de este caso de uso, el usuario puede ejecutar los casos de uso: Crear el esquema, Modificar el esquema, Eliminar el esquema.

excepciones

ninguna

postcondición

antes de comenzar con la creación de grupos y objetos de replicación, es preciso crear el esquema de replicación.

Caso de uso *Establecer criterios de encolamiento*

resumen

el DBA establece los criterios para el encolado de transacciones.

actores

DBA

precondiciones

ser usuario administrador, existir uno ó mas sitios de replicación esclavos.

descripción

el usuario selecciona la opción Establecer criterios, en la vista correspondiente el usuario debe elegir las opciones de encolamiento disponibles en el MRD:

- *total*: encola las modificaciones originadas por todos los usuarios y procesos de la base de datos.
- *por rol*: encola las modificaciones originadas por usuarios que poseen un rol o conjunto de roles determinado.
- *por perfil*: encola las modificaciones originadas por usuarios que poseen un perfil determinado.
- *por lista de usuarios*: encola las modificaciones originadas por usuarios que están presentes en una lista determinada. Esta lista es editada por el usuario administrador.
- *por condición*: el encolado esta sujeto a ciertas condiciones en los datos del registro original y/o modificado. Esta condición es editada por el usuario administrador.

excepciones

ninguna

postcondición

una vez establecido el criterio y conformados los objetos de replicación, las modificaciones en el criterio deberán realizarse por objeto, y no así por grupo.

Caso de uso *Establecer modalidad de encolamiento*

resumen

el DBA establece modalidad de encolado de definiciones.

actores

DBA

precondiciones

ser usuario administrador, existir uno ó mas sitios de replicación esclavos.

descripción

el usuario selecciona la opción Establecer modalidad, en la vista correspondiente el usuario debe elegir si el encolado de la sentencia DDL será estática ó dinámica, consistiendo cada uno en lo siguiente:

- *estática*: se generará la sentencia de creación, modificación ó eliminación, en el acto y se encolará para su posterior ejecución (replicación).
- *dinámica*: se generará la sentencia de creación, modificación ó eliminación, en el momento de su ejecución.

El único criterio de encolamiento para grupos DDL, será el:

- *manual*: donde el encolado será realizado por el usuario administrador.

excepciones

ninguna

postcondición

la modalidad establecida, podrá ser modificada en cualquier momento.

Caso de uso *Editar componente de respaldo/recuperación*

resumen

el DBA edita el componente de respaldo y recuperación del objeto DDL.

actores

DBA

precondiciones

ser usuario administrador, existir como mínimo un objeto de replicación DDL.

descripción

el usuario selecciona la opción Editar componente, en la vista de edición el usuario personaliza el procedimiento almacenado encargado de generar los scripts de respaldo ante posibles fallas de inconsistencia en la replicación. Y en tal caso, éste debe ejecutar dichos scripts para restaurar la definición original del objeto destino. Con ello el componente deberá garantizar la recuperación total de los datos y estructura del objeto destino no replicado. Existirá un único componente de respaldo/recuperación para todos los sitios destino.

excepciones

ninguna

Caso de uso *Programar horarios de replicación*

resumen

el DBA programa los horarios de replicación para los grupos DML y DDL.

actores

DBA

precondiciones

ser usuario administrador, existir uno ó mas grupos de transacciones y definiciones.

descripción

el usuario selecciona la opción Horarios de replicación, en la vista correspondiente el usuario debe seleccionar el grupo de replicación y re-establecer la actualización (manual ó automático) y el tipo de actualización (relativo, absoluto, persistente ó prioritario). Adicional a ello el usuario elegirá en activar o desactivar el uso del Agente de Replicación, leer los logs, actividad, historial, etc., del agente software.

excepciones

ninguna

postcondición

ninguna.

Caso de uso *Establecer métodos de resolución de conflictos*

resumen

el DBA establece los métodos de resolución de conflictos ante posibles inconsistencias de replicación de transacciones y definiciones.

actores

DBA

precondiciones

ser administrador, existir uno ó mas grupos de transacciones y definiciones.

descripción

el usuario selecciona la opción Resolución de conflictos, en la vista correspondiente el usuario debe seleccionar los métodos convencionales, y también tiene la posibilidad de introducir criterios complejos (procedimientos) para la resolución de conflictos de replicación. Estos son aplicables tanto a grupos como a objetos de ambos tipos (DML y DDL).

excepciones

ninguna

postcondición

ninguna.

A.3. PLANTILLAS DE TAREAS

Tarea *ReplicaciónDeDatos*

objetivo

rastrear el sitio maestro de replicación, replicar con éxito las transacciones y especificaciones establecidas y, en casos críticos, recomendar al usuario acciones oportunas a tomar.

descripción

en esta macro tarea se rastrean continuamente los eventos que puedan dar inicio a los procesos de replicación, y una vez identificados, iniciar la replicación de los correspondientes grupos de objetos. En caso que el agente no pueda tomar decisiones, deberá recomendar al usuario las posibles acciones a tomar.

precondición

entorno de replicación activado

supertarea

ninguna

subtareas

RastrearSitioMaestro, IniciarReplicaciónDeGrupos, MostarRecomendación

tipo_de_descomposición

funcional

Tarea *AccionesDelUsuario*

objetivo

rastrear acciones del usuario y ejecutarlas si corresponden.

descripción

en esta tarea se rastrea e identifica las acciones del usuario, obteniendo las solicitudes de inicio del proceso de replicación, para actualizar la base de conocimiento e iniciar el proceso de replicación solicitado.

entrada

acciones del usuario para el inicio de la replicación.

salida

actualización de la base de conocimiento y retorno de acciones del usuario a la supertarea.

supertarea

RastrearSitioMaestro

subtareas

ninguna

frecuencia
una vez , por cada acción del usuario.

Tarea *EsperarMensajeDeOtroAgente*
objetivo
captar recomendaciones provenientes de otro agente.
descripción
en esta tarea se esperan recomendaciones, sobre el inicio de procesos de replicación, provenientes de otros agentes.
entrada
mensajes de recomendación

salida
actualización de la base de conocimiento y retorno de recomendaciones recibidas de otros agentes hacia la supertarea.
supertarea
RastrearSitioMaestro
subtareas
ninguna
frecuencia
una vez, por cada recomendación recibida.

Tarea *ConsultarBaseDeConocimiento*
objetivo
encontrar en la Base de Conocimiento posibles recomendaciones.
descripción
en esta tarea se buscan posibles recomendaciones, en la Base de Conocimientos, sobre tiempos esperados y frecuencias de replicación, jerarquías y prioridades de grupos y objetos de replicación, etc.
entrada
mensajes retornados por la tarea *VerificarConectividad* y por la supertarea.
salida
mensajes de recomendación.
precondición
invocación previa de la tarea *VerificarConectividad*.
supertarea
VerificarDisponibilidad
subtareas
ninguna
frecuencia
por cada parámetro de la entrada.

Tarea *Deducir*
objetivo
clasificar el orden de los mensajes de recomendación.
descripción
en esta tarea se clasifican las recomendaciones, tanto las ideas del propio agente, como las recibidas de otros agentes. La clasificación se realiza de acuerdo a un criterio propio del agente.
entrada
mensajes de recomendación.
salida
mensajes de recomendación ordenados según criterio.
precondición
mensajes de recomendación recibidos.
supertarea

VerificarDisponibilidad

subtareas

ninguna

frecuencia

por cada elemento de entrada.

requiere

ClasificaciónSegunCriterio

Tarea *ComunicarAOtroAgente*

objetivo

comunicar mensajes ordenados a otros agentes.

descripción

en caso de presencia de errores de replicación, esta tarea comunica mensajes a otros agentes, y éstos solucionan dichos problemas, ó en caso de fracaso devuelven recomendaciones.

entrada

cortes en la red de comunicaciones, inconsistencias causadas por la replicación, incumplimiento de reglas establecidas, etc.

salida

mensajes ordenados según criterios establecidos.

supertarea

EjecutarProcesoDeReplicación

subtareas

ninguna

frecuencia

por cada elemento de entrada.

Tarea *ResoluciónDeConflictos*

objetivo

resolver parcial o totalmente, los distintos tipos de conflictos que se presentan en los procesos de replicación.

descripción

esta tarea atiende acciones del usuario y mensajes de los otros agentes ante eventuales existencias de conflictos en todos los procesos de replicación.

entrada

acciones del usuario y mensajes de los otros agentes.

salida

mensajes y recomendaciones

precondición

entorno de replicación activado.

supertarea

ninguna

subtareas

RastrearSitioMaestro, CorrecciónDeConflictos, MostrarRecomendación.

tipo_de_descomposición

funcional

frecuencia

por cada elemento de entrada.

Tarea *DeterminarTipoDeConflicto*

objetivo

encontrar una categoría para el conflicto recibido.

descripción

en esta tarea se clasifica el conflicto recibido, esta clasificación abarca distintos tipos, como ser problemas de: conectividad, disponibilidad, tráfico en la red,

presencia de objetos inválidos, inconsistencia de los datos replicados, violación de reglas de integridad, incongruencia de datos esperados, etc.

entrada

mensajes con conflictos de replicación.

salida

conflictos clasificados.

precondición

mensajes de conflictos recibidos.

supertarea

CorrecciónDeConflictos

subtareas

ninguna

frecuencia

por cada elemento de entrada.

Tarea *IdearSolución*

objetivo

identificar conflictos clasificados que pueden ser solucionados.

descripción

en esta tarea se identifican los conflictos que pueden ser solucionados (conflictos clasificados), se determina si pueden ser solucionados de acuerdo a criterios del propio agente.

entrada

conflictos clasificados.

salida

ponderación de soluciones según criterio.

precondición

conflicto clasificado recibido

supertarea

CorrecciónDeConflictos

subtareas

ConsultarBaseDeConocimiento, Deducir.

tipo_de_descomposición

funcional

frecuencia

por cada elemento de entrada.

A.4. DESCRIPCIÓN DE OBJETIVOS

Objetivo Comunicar recomendaciones al usuario

tipo

objetivo persistente

parámetros-entrada

recomendaciones codificadas

parámetros-salida

recomendaciones en lenguaje natural

condición-activación

recibir sentencia de recomendación

condición-finalización

comunicar al usuario

lenguaje rep. conocimiento

lenguaje natural

descripción

este objetivo permite que todas las recomendaciones, provenientes tanto del *Replicador* como del *Corrector,* sean comunicadas al usuario.

Objetivo Rastrear horarios programados de replicación

tipo

objetivo persistente de aprendizaje

parámetros-entrada

horarios de grupos de replicación pendientes de ejecución

parámetros-salida

invocación a los métodos de replicación

condición-activación

inicio de la aplicación

condición-finalización

fin de la aplicación

lenguaje rep. conocimiento

sentencias algebraicas para la entrada, paso de argumentos entre objetos para la salida.

descripción

este objetivo rastrea persistentemente la lista de horarios de replicación, con el propósito de verificar la existencia de procesos de replicación pendientes de ejecución.

Objetivo Recomendar al *Asistente* horarios de replicación basado en experiencias pasadas

tipo

objetivo no persistente

parámetros-entrada

sitios de replicación que intervienen en el proceso

parámetros-salida

sentencias de recomendación

condición-activación

solicitud de inicio de replicación

condición-finalización

comunicar al *Asistente*

lenguaje rep. conocimiento

lenguaje natural.

descripción

este objetivo permite que el *Replicador* comunique recomendaciones de los mejores horarios de replicación hacia el Usuario, vía el *Asistente*.

Objetivo Replicar objetos de especificaciones y de transacciones

tipo

objetivo no persistente de aprendizaje

parámetros-entrada

especificaciones y transacciones no replicadas

parámetros-salida

especificaciones y transacciones replicadas, o mensajes de error

condición-activación

recibir sentencia de inicio de replicación

condición-finalización

mensaje de error, o especificaciones y/o transacciones replicadas con éxito

lenguaje rep. conocimiento

lenguaje natural para la salida de mensajes de errores, representación interna para la salida correspondiente a replicación sin error, para la entrada algebra relacional.

descripción

este objetivo permite que el *Replicador* inicie y conduya el proceso de replicación de especificaciones y transacciones.

Objetivo Idear solución para el conflicto

tipo

objetivo no persistente de aprendizaje

parámetros-entrada

objeto con conflictos de replicación, lista de posibles acciones

parámetros-salida

plan con las instrucciones necesarias de solucionar el conflicto.

condición-activación

cada vez que se lo invoque

lenguaje rep. conocimiento

representación interna

descripción

este objetivo permite que el *Corrector*, consultando la base de conocimiento, planifique la mejor solución a los conflictos de replicación del objeto en cuestión.

A.5. DESCRIPCIÓN DEL ESQUEMA DEL MODELO

Concepto Esquema

descripción

representa el esquema de negocio para aplicar la replicación (entorno de replicación).

propiedades

idEsquema: String;
nombre: String;
descripcion: String;
contenido: String;
estado: String.

Concepto Sitio

descripción

representa a los sitios de replicación (bases de datos) que conforman el esquema de replicación (agregación).

propiedades

idSitio: String;
idEsquema: String;
nombre: String;
tipo: String;
descripcion: String;
cadenaConexion: String;
usuario: String;
password: String;
versionRDBMS: String;
estado: String.

Concepto Grupo

descripción

representa a los grupos de replicación asociados al esquema de replicación.

propiedades

idGrupo: String;
idEsquema: String;
nombre: String;
tipoSQL: String;
descripcion: String;
tipoCola: String;
actualizacion: String;

tipoActualizacion: String;
tipoEncolado: String;
inicio: Date;
intervalo: String;
idSitioOrigen: String;
estado: String.

Concepto Objeto
descripción
representa a los objetos de replicación asociados a los grupos de replicación.
propiedades
idObjeto: String;
idGrupo: String;
nombre: String;
tipoObjeto: String;
estado: String.

Concepto BaseDeConocimiento
descripción
representa al repositorio de datos en el que se almacenan resultados de procesos de replicación ejecutados y programados. Está compuesto por tablas, vistas, secuencias y procedimientos almacenados.
propiedades
idProceso: Integer;
logica: String;
agenteOrigen: String;
nroTransacciones: Integer;
fechaHoraEjecutada: Date;
duracion: BigDecimal;
recursosEstimados: class(usoProcesador: Integer,
usoMemoria: Integer,
anchoBanda: Integer,
nroIntentos: Integer,
duracion: BigDecimal);
recursosUtilizados: class(usoProcesador: Integer,
usoMemoria: Integer,
anchoBanda: Integer,
nroIntentos: Integer,
duracion: BigDecimal);
errores: collection(idError: String,
descripcion: String
cantidad: Integer);
fecha: Date.

A.6. DESCRIPCIÓN DE LAS CONVERSACIONES

Conversación *Comunicar al Usuario*
tipo
comunicar información
objetivo
comunicar al usuario de las recomendaciones provenientes de los agentes Replicador y Corrector.
agentes
Asistente, Replicador, Corrector

iniciador
Replicador o Corrector
servicio
comunicar
descripción
los agentes Replicador y Corrector tienen por objetivo comunicar sus recomendaciones al usuario, para ello se contactan con el agente Asistente (ya que es la interfaz con el usuario) para solicitarle el servicio de comunicar la recomendación.
precondición
recomendación recibida
postcondición
recomendación comunicada

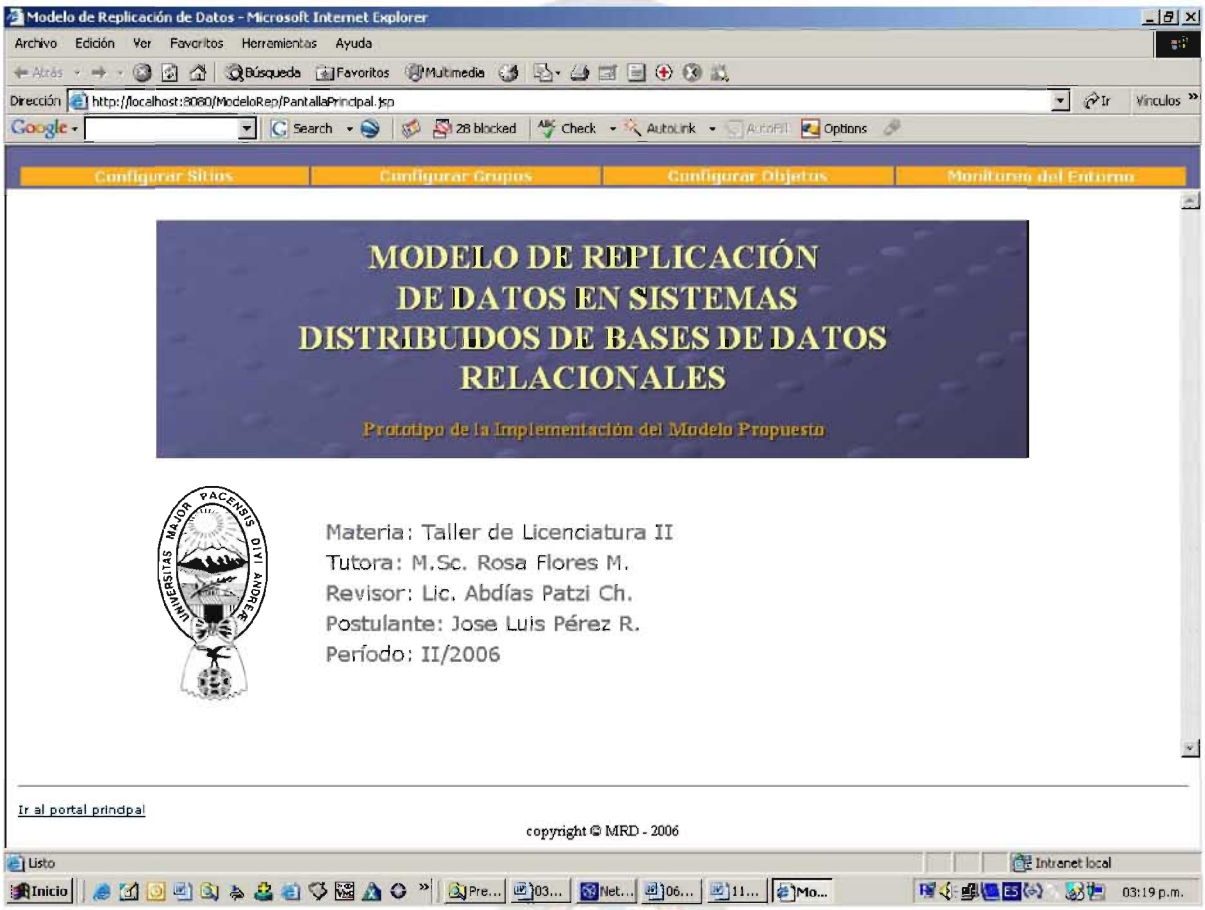
Conversación *Analizar para Recomendar*
tipo
obtener y comunicar información
objetivo
en base a la experiencia adquirida del agente Replicador y a las condiciones del entorno, verificar si corresponde iniciar el proceso de replicación, o en su lugar comunicar al agente Asistente posibles recomendaciones.
agentes
Asistente, Replicador
iniciador
Replicador
servicio
comunicar
descripción
el agente Replicador verifica si corresponde iniciar el proceso de replicación, en caso contrario comunica al Asistente (para que este comunique al usuario) las posibles recomendaciones de horarios, y éste analice otras posibilidades.
precondición
recomendación deducida
postcondición
recomendación comunicada

Conversación *Corregir Conflicto*
tipo
procesar y comunicar información
objetivo
corregir posibles conflictos de replicación encontrados en ciertos registros de transacciones y/o definiciones.
agentes
Corrector, Replicador, Asistente
iniciador
Corrector
servicio
procesar, comunicar
descripción
en base a las peticiones del usuario o del Replicador, el agente Corrector analiza las posibles causas de los conflictos encontrados, y busca posibles soluciones, para luego idear un plan que corrija los conflictos en los registros. Una vez aplicado el plan, envía las correspondientes recomendaciones hacia el usuario o al agente Replicador.
precondición
recomendación deducida
postcondición
recomendación comunicada

ANEXO B

Interfases del Prototipo

B.1. INTERFASE: PANTALLA PRINCIPAL



B.2. INTERFASE: CONFIGURACIÓN DE SITIOS

B.3. INTERFASE: CONFIGURACIÓN DE GRUPOS

Configuración de Grupos de Replicación

- Listado de Grupos de Replicación
- Adición de Grupos de Replicación
- Modificación de Grupos de Replicación
- Eliminación de Grupos de Replicación

B.4 INTERFASE: CONFIGURACIÓN DE OBJETOS

Configuración de Objetos de Replicación

- Listado de Objetos de Replicación
- Adición de Objetos de Replicación
- Modificación de Objetos de Replicación
- Eliminación de Objetos de Replicación

B.5 INTERFASE: LISTADO DE SITIOS DE REPLICACIÓN

Listado de Sitios

Id del Sitio	Nombre del Sitio	Descripción	Cadena de Conexión	Usuario	Versión
S1	RDIS	Sitio Maestro	jdbc:oracle:thin:@192.9.200.156:1533:rdis	repadmin	Oracle 8i
S2	ICBA	Sitio Esclavo	jdbc:oracle:thin:@192.168.150.81:1525:icba	repadmin	Oracle 8i

Ir atrás | Adición de Sitios de Replicación

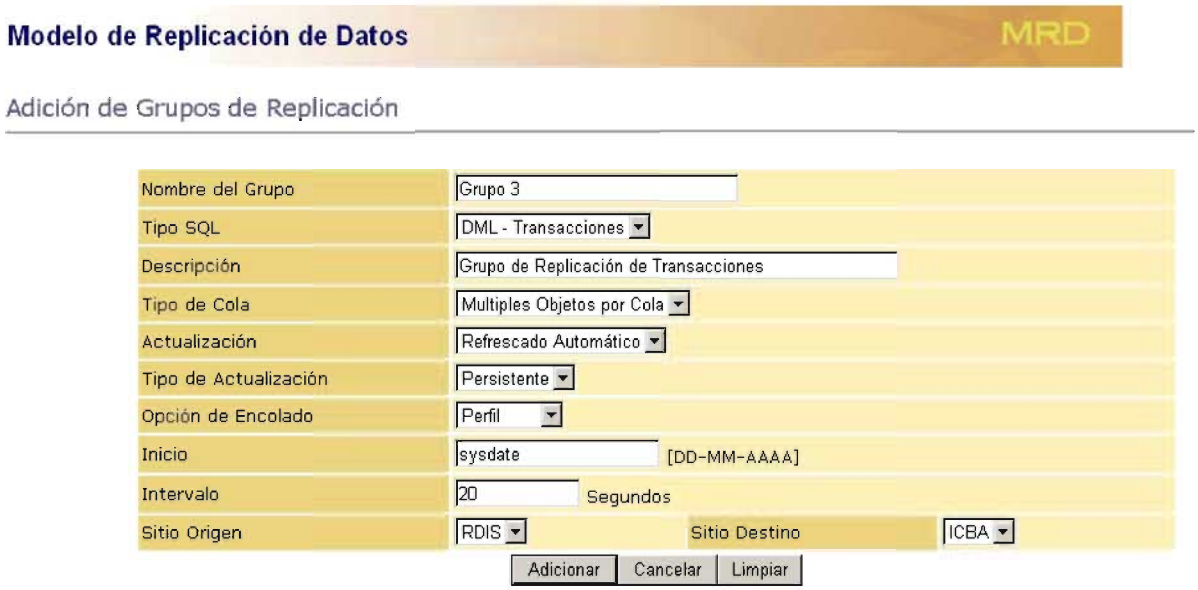
B.6. INTERFASE: LISTADO DE GRUPOS DE REPLICACIÓN

Listado de Grupos

Sitio Origen	Sitio Destino	Id del Grupo	Nombre del Grupo	Tipo SQL	Descripción
RDIS	ICBA	G1	Grupo 1	DML	Grupo de Transacciones 1
RDIS	ICBA	G2	Grupo 2	DML	Grupo de ReplicaciÃ³n de Transacciones 2

Ir atrás | Adición de Grupos de Replicación

B.7. INTERFASE: ADICIÓN DE GRUPOS



B.8. INTERFASE: SELECCIÓN DE GRUPOS

B.9. INTERFASE: ADICIÓN DE OBJETOS