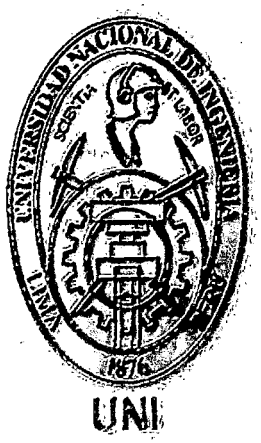
UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Industrial y de Sistemas

Sección de Posgrado



SOFTWARE PARA EL PROCESAMIENTO PARALELO DE CONSULTAS EN UN CLUSTER DE COMPUTADORAS

TESIS DE MAESTRIA

Para optar el grado académico de:

MAESTRO EN CIENCIAS CON MENCION EN INGENIERIA DE SISTEMAS.

ING. GABRIELA TINTAYA GALICIA

LIMA - PERU
2010

DEDICATORIA

A Dios que me ha dado la vida y por haberme permitido alcanzar una más de mis metas.

A mis padres, por estar conmigo en todo momento y por el apoyo incondicional brindado en cada etapa de mi vida.

A mí adorada hija Alexandra, que con su sonrisa alegra mi corazón; por ser mi razón, fuerza y templanza de mi vida.

A Nilton, por ser la persona que ha compartido momentos importantes en mi vida; y en su compañía todo es posible.

A mí querida hermana Carolina por el apoyo y estimulo constante brindado en todo momento.

AGRADECIMIENTO

Primero y antes que nada doy gracias a Dios, por estar conmigo en cada paso que doy y por fortalecer mi corazón e iluminar mi mente.

Esta tesis de Maestría, si bien ha requerido de esfuerzo y mucha dedicación por parte de la autora, no hubiese sido posible su culminación sin la cooperación desinteresada de un grupo especial de personas que estuvieron conmigo apoyándome en todo momento.

Mí más sincero agradecimiento al MSc. Zalatiel Carranza Ávalos por asesorar y orientar mi tesis, de igual manera a los señores miembros de jurado por las sugerencias realizadas en la etapa de desarrollo de la tesis.

Quedo también agradecida con los docentes de la Sección de Posgrado de la facultad de Ingeniería Industrial y de Sistemas, quienes contribuyeron con sus enseñanzas en esta nueva etapa de mi formación académica.

INDICE GENERAL

CAPITULO I

ASPECTOS GENERALES

CAPITULO II

MARCO DE REFERENCIA

CAPITULO III

ALGORITMOS DE SUBDIVISION DE CONSULTAS PARA SU PROCESAMIENTO PARALELO

CAPITULO IV:

DEMOSTRACION TEORICA DE LA HIPOTESIS

INDICE DE TABLAS

INDICE DE FIGURAS

DESCRIPTORES TEMATICOS

Arquitectura de Software
Algebra Relacional
División horizontal de consultas
División vertical de consultas
Interconsulta
Intraconsulta
Paralelismo
Cluster de computadores
Procesamiento paralelo de consultas
Modelo de vistas 4+1 de Kruchten

RESUMEN

Se realizó un análisis teórico del algoritmo planteado de división de consultas entre los nodos de un cluster de computadoras, que permitió encontrar la relación existente entre las diferentes variables que intervienen en este proceso, en base a la definición formal del algoritmo, se llega a demostrar teóricamente, que al comparar las modalidades de división de consulta vertical y horizontal, la primera produce tiempo de respuesta menor. El análisis ha permitido a su vez, interpretar el comportamiento del algoritmo con relación a las diferentes variables intervinientes.

Adicionalmente, con el objeto de comprobar los alcances prácticos del funcionamiento de los dos tipos de división de consultas, referidos, la tesis afronta el desarrollo de un software, situado entre la aplicación cliente y su gestor de Base de Datos, con capacidad de subdividir las consultas, para ser procesadas de manera paralela en un cluster de computadoras. Este software ha permitido establecer el grado de conveniencia que cada modalidad de división presenta.

ABSTRACT

Was performed a theoretical analysis of the algorithm proposed of division of queries between the nodes of a cluster of computers, which allowed to find the relationship between the different variables involved in this process, based on the formal definition of algorithm, was shown theoretically, that when comparing the arrangements for query division, vertical and horizontal, the first one produces less response time. The analysis has led in turn to interpret the behavior of the algorithm with respect to the different variables involved in.

Additionally, in order to verify the practical scope of operation of the two types of query division, referrals, the thesis facing the development of a software, between the client application and database manager with ability to subdivide queries to be processed in parallel on a cluster of computers. This software has allowed us to establish the degree of convenience that each type of division presents.

INTRODUCCION

En la actualidad, con el auge del comercio electrónico, el crecimiento de las empresas, la competencia entre ellas por liderar el mercado, la necesidad de desarrollar practicas comerciales más ágiles y una adecuada toma de decisiones. Ha llevado a las empresas e instituciones a almacenar información histórica e importante en gigantescas Data WareHouses y aplicar nuevas técnicas de extracción de la información como el Data Mining y utilizar sistemas OLAP.

Sin embargo, cuando la cantidad de información a manejar es muy grande, los tiempos de ejecución requeridos para procesar las consultas utilizadas por este tipo de técnicas pueden llegar a ser intolerablemente altos en computadores con un solo procesador.

Las computadoras multiprocesador combinadas con un SGBD empresarial se presentan como una solución a esta deficiencia, pero su elevado costo de compra y mantenimiento lo hace inalcanzable para la mayoría de las empresas. Esto conduce a buscar nuevas alternativas de solución frente a estos problemas.

Una solución de bajo costo es poner a trabajar de manera coordinada y cooperativa a un conjunto de computadores secuenciales interconectados entre si, para hacerlos actuar como un computador con capacidades de procesamiento paralelo; para lo cual se deberá contar con un sistema de software que se encargue de la gestión de dicho conjunto.

El objetivo de esta tesis es realizar todas las investigaciones que sean necesarias para desarrollar un software que combine el poder de varios microcomputadores de bajo costo, distribuya la carga de trabajo entre todos

los nodos, haciendo que las consultas se procesen más rápido y planteé una solución a los problemas antes mencionados. Además este software permitirá realizar a quien lo desee, pruebas de análisis del procesamiento de consultas, lo cual será útil para determinar la conveniencia de cambiarse a un modelo de procesamiento paralelo.

La tesis comenzará describiendo los aspectos generales, dentro del cual se planteará el problema y en base a ello se propondrán los objetivos generales y específicos; del objetivo general se obtendrá la hipótesis de la presente investigación. Como punto de partida, para la demostración de dicha hipótesis se elaborará un marco teórico donde se muestren los conceptos que estén directamente relacionados con el tema de la investigación, también se describirá temas relevantes que servirán de base para la construcción del software. Seguidamente, se propondrán los algoritmos de subdivisión de consultas explicando en detalle su funcionamiento, pseudocódigo, ventajas y desventajas de dichos algoritmos. En base a todo lo anterior, se realizará la demostración teórica de la hipótesis. El cual se apoyará fuertemente en la teoría del algebra relacional, planteada por el Dr. Edgar Frank Codd. Para poder contrastar los resultados obtenidos en la demostración teórica de la hipótesis se desarrollará un software; cuya arquitectura será elaborada utilizando el modelo de vistas 4+1 de Kruchten, y siguiendo los pasos definidos por la metodología RUP, mediante el cual se mostrará el modelo de casos de uso, el modelo de análisis, el modelo de diseño, y el modelo de implementación con sus respectivos diagramas. Finalmente con el prototipo del software desarrollado se realizará la demostración experimental de hipótesis.

CAPITULO I

ASPECTOS GENERALES

En este capitulo se realiza el planteamiento del problema sobre el cual se proponen los objetivos generales y específicos; y basándose en el objetivo general se plantea la hipótesis de la investigación.

1.1. DIAGNOSTICO Y ENUNCIADO DEL PROBLEMA

En la última década, los sistemas paralelos de bases de datos han estado casi descartados incluso para algunos de sus más firmes defensores. Actualmente están siendo comercializados con éxito por prácticamente todos los fabricantes de bases de datos (Teradata, Tamdem, Oracle, Microsoft). Este cambio lo han impulsado las siguientes tendencias:
Los requisitos transaccionales de las empresas han aumentado con el uso creciente de los ordenadores. Además, el crecimiento de la World Wide Web ha creado muchos sitios con millones de visitantes, y las cantidades crecientes de los datos recogidos por los visitantes han producido bases de datos extremadamente grandes en muchas empresas. Las empresas utilizan volúmenes crecientes de datos para planificar sus actividades y sus tareas, del orden de los terabytes. Las consultas utilizadas para estos fines se denominan consultas de ayuda a la toma de decisiones. Los sistemas con un único procesador no son capaces de tratar volúmenes de datos tan

grandes a la velocidad necesaria. Al abaratarse los microprocesadores, las máquinas paralelas se han vuelto comunes y relativamente baratas. El paralelismo se utiliza para proporcionar aceleración, las consultas se ejecutan más rápido debido a que se proporcionan más recursos, como procesadores y discos.

El paralelismo también se utiliza para proporcionar escalabilidad, y las cargas de trabajo crecientes se tratan sin aumentar el tiempo de respuesta mediante un aumento en el grado de paralelismo.

En la actualidad, con el auge del comercio electrónico, el crecimiento de las empresas, la competencia entre ellas por liderar el mercado, la necesidad de desarrollar practicas comerciales más ágiles y una adecuada toma de decisiones. Ha llevado a las empresas e instituciones a almacenar información histórica e importante en gigantescas Data WareHouses y aplicar nuevas técnicas de extracción de la información como el Data Mining. Sin embargo, cuando la cantidad de información a manejar es muy grande, los tiempos de ejecución requeridos para procesar las consultas utilizadas por este tipo de técnicas pueden llegar a ser intolerablemente altos en computadores con un solo procesador.

Las grandes compañías de Software y Hardware (como por ejemplo: Oracle, Microsoft, IBM, Informix, Teradata, Hewlett-Packard) ofrecen soluciones que explotan el procesamiento paralelo en sistemas multiprocesadores (fuertemente acoplado[1]) y sistemas multicomputadores (débilmente acoplado) para mejorar el rendimiento de los SGBD[2]. Aunque son efectivas, estas soluciones son bastantes costosas en términos de adquisición, mantenimiento y escalabilidad.

En nuestro país la adquisición de software pasa por diferentes procesos, especialmente en las instituciones publicas, ya que necesitan ser justificadas y previamente evaluadas. En este sentido se torna complicado realizar pruebas sobre entornos computacionales paralelos propietarios, ya que las

[1] Conjunto de procesadores que comparten la misma memoria principal.

[2] Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan.

versiones de prueba que ofrecen, exigen contar con hardware de elevado costo y en algunos casos también de software específico. Las instituciones y empresas no cuentan con un software de prueba el cual apoye a los responsables en la verificación de la optimización de su procesamiento de consultas, que les permita decidir adecuadamente si migrar o no al nuevo modelo previo a una costosa inversión de tiempo y dinero.

1.1.1. DEFINICIÓN DEL PROBLEMA

Se busca desarrollar un sistema de software económico que puede ser utilizado por sistemas que manejen gran cantidad de datos y que por su naturaleza requieran acceder a tecnologías de procesamiento paralelo, logrando minimizar el tiempo de respuesta de sus consultas.

El desarrollo del software permitirá responder las siguientes preguntas:

¿Existirá un sistema de bajo costo que procese las consultas paralelamente sobre un cluster de computadores?,

¿Seria compatible con las actuales aplicaciones y los SGBD existentes en las instituciones?,

¿Se podría afirmar que el procesar una consulta dividida en un cluster de computadores es mas rápido que procesar la consulta en un solo nodo del cluster?,

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollar un software de código abierto; que subdivida las consultas dirigidas a un SGBD, para su procesamiento paralelo sobre un cluster de computadores, que servirá como mecanismo de prueba para determinar la diferencia entre la división vertical y horizontal de consultas.

1.2.2. OBJETIVOS ESPECÍFICOS

- Diseñar un algoritmo para subdividir las consultas entre los nodos del cluster de computadoras.
- Demostrar teóricamente la hipótesis planteada.
- Elaborar la arquitectura del software.
- Elaborar las fases del software aplicando la metodología RUP.
- Desarrollar un prototipo del software.
- Demostrar que el software mejora el tiempo de respuesta de la consulta mediante el procesamiento paralelo.

1.3. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

1.3.1. IMPORTANCIA Y JUSTIFICACIÓN

La importancia y justificación del presente trabajo de investigación radica en que:

- Este proyecto aporta una herramienta que ilustra el funcionamiento de los algoritmos de división vertical y horizontal de consultas, además es de código abierto y puede ser modificado sin ninguna limitación.
- Es una herramienta económica, ya que los requerimientos del software no son altos en cuanto a hardware, además el cluster puede

ser construido con computadores de bajo costo y sin la necesidad de ningún hardware especial.

- Este trabajo es una contribución a los desarrolladores y analistas de bases de datos y en general a todas las personas que trabajan en la manipulación de base de datos, ya que ayuda a decidir la conveniencia de usar el esquema de la división vertical o división horizontal.

1.3.2. DELIMITACION DE LA INVESTIGACIÓN

- El prototipo del software será desarrollado y probado en una intranet con computadores con sistema operativo Windows XP.
- El prototipo del software se probará con el SGBD MySQL Server por ser un software libre (GNU) y Open Source[3].
- El prototipo solo aceptara instrucciones que se encuentren dentro del estándar SQL-99.
- Para el diseño físico de la red se utilizará topología estrella y el protocolo de comunicaciones TCP/IP.

1.4. HIPÓTESIS DE LA INVESTIGACIÓN

1.4.1. DEFINICIÓN DE LA HIPÓTESIS

El tiempo de respuesta de una consulta con complejidad alta usando división vertical es menor que el tiempo de respuesta de la misma consulta usando división horizontal

[3] Open Source (Código abierto) es el término por el que se conoce al software distribuido y desarrollado en una determinada forma. Este término empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original, en inglés, del software libre (free software).

1.4.2. DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

Variables independientes:

Las consultas de complejidad alta dirigidas a un SGBD: Las consultas son la forma principal de hacer solicitudes de información a la base de datos. Las consultas están compuestas de comandos que se envían a la base de datos en un formato predefinido, generalmente utilizando el formato SQL.

Una consulta básica SQL consta de 3 cláusulas SELECT, FROM, WHERE.

SELECT se usa para listar los atributos que se desean en el resultado de una consulta.
FROM lista las relaciones que van a intervenir en la consulta.
WHERE consta de 0 o mas predicados (Pi) que implica los atributos de las relaciones que aparecen en la cláusula FROM, separados por los operadores OR ó AND.

Numero de nodos del cluster de computadoras: Es la cantidad de computadoras que forman el cluster

Variables dependientes:

El tiempo de respuesta de cada consulta: El es tiempo total que demora en ejecutarse una consulta.

1.4.3. DEFINICION OPERACIONAL DE LAS VARIABLES

La definición operacional de las variables independientes y dependientes tienen los siguientes indicadores:

VARIABLE INDEPENDIENTE	
Variable	**Indicador**
Las consultas de complejidad alta dirigidas a un SGBD	Número de comparaciones que realiza la consulta
Numero de nodos del cluster de computadoras	Número de nodos

VARIABLE DEPENDIENTE	
Variable	**Indicador**
El tiempo de respuesta de cada consulta	Tiempo

1.5. METODOLOGÍA DE LA INVESTIGACIÓN

1.5.1. TIPO DE INVESTIGACION

La metodología de investigación que se utiliza, para la realización del presente "Software para el procesamiento paralelo de consultas en un cluster de computadoras" son de tipo exploratoria, descriptivo y experimental.

En la primera fase se hará uso del tipo de investigación exploratoria ya que se viene recopilando y analizando la información referente al tema, se hará uso de la investigación descriptiva porque se elaborará y evaluará algoritmos de paralelización vertical y horizontal para el procesamiento de consultas y

en la ultima fase se hará uso de la investigación experimental porque se realizará las pruebas con el software para poder medir su rendimiento en el procesamiento de consultas.

Para el diseño del software se utilizará la metodología RUP4 (Rational Unified Process) porque se adecua mejor al proyecto.

1.5.2. POBLACIÓN Y MUESTRA

POBLACION: La investigación tiene como población a todas las consultas que se pueden realizar a una base de datos.

MUESTRA: La muestra esta constituida un conjunto de consultas proporcionadas por Transaction Processing Performance Council (TPC) [TPC-H], la cuales han sido seleccionadas por: su alta complejidad, por la alta cantidad de registros que compara y porque dan respuesta a cuestiones criticas de negocio.

1.5.3. TÉCNICAS E INSTRUMENTOS

TECNICA

La técnica que se hace uso para el desarrollo del trabajo de investigación es la observación, por ser fundamental en todos los campos de la ciencia. La observación consiste en el uso sistemático de nuestros sentidos orientados a la captación de la realidad que se quiere estudiar. A través de los sentidos se logrará captar la realidad que nos rodea, para luego sistematizar la información obtenida.

INSTRUMENTOS

Los instrumentos que se van a usar son los siguientes:

[4] El RUP (Proceso Racional Unificado) es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso

- Como Software de prueba se usará el "Software para el Procesamiento Paralelo de consultas en un Cluster de Computadoras".
- Para enviar consultas y visualizar los resultados y otra información se utilizará una aplicación desarrollada en java "Interfaz de administración del software".
- Como base de datos la proporcionada por el TPC-H [TPC-H] con un factor de escala 1 (1 GB aprox).
- Como SGBD se usará MySQL Server 5.1
- 10 Computadoras
- 1 Switch

1.5.4. ANÁLISIS Y TRATAMIENTO DE LOS DATOS

Los datos se presentan de manera estructurada y sistemática.

- Las variables dependientes y variables independientes

  La variable independiente será la variable que se manipulará en las pruebas que se realicé; la variable dependiente será la variable en la que se mida el efecto del cambio de las otras variables.

  Estos resultados obtenidos se presentarán en una tabla en la que se observe el comportamiento de la variable dependiente en función de las consideradas independientes.
- Tablas de datos

  Presentarán los valores encontrados para cada variable. Las tablas de datos proporcionarán una primera idea de las tendencias de los resultados.

  La tabulación de los datos constituirá el primer paso para su presentación gráfica y para el tratamiento estadístico posterior.

  Su organización dependerá del número y de las características de las variables que intervengan en las pruebas que se van a realizar.
- La presentación de los datos

a) Histograma

Eje horizontal: Valores hallados para una determinada variable independiente.

Eje vertical: Número de casos encontrados para cada valor de la variable dependiente. El histograma permitirá obtener una primera impresión visual sobre la distribución de los datos.

b) Diagrama de barras

Presentará la misma información que el histograma, rotando la posición de los ejes.

Tanto el histograma como el diagrama de barras nos permitirán visualizar los resultados obtenidos en las pruebas realizadas y así permitirá llegar a determinadas conclusiones.

1.6. APORTES DE LA INVESTIGACIÓN

El aporte del presente trabajo radica en que:

- Este proyecto aporta una solución para las empresas que presentan bajo rendimiento en el procesamiento de consultas, y que desean evaluar alternativas de procesamiento paralelo para estas, previo a la realización de costosas inversiones.
- Es una solución económica, ya que los requerimientos del software no son altos en cuanto a hardware, además el cluster puede ser construido con computadores de bajo costo y sin la necesidad de ningún hardware especial.
- Por ser un proyecto de software libre, es entregado a la comunidad de desarrolladores y usuarios bajo licencia GNU, lo que implica una gran fuente de aporte educativo para los interesados. Y además se deja un modelo de arquitectura de software que será de mucha utilidad a la hora de diseñar e implementar proyectos de este tipo.

CAPITULO II

MARCO DE REFERENCIA

En este capitulo se describe los conceptos que están directamente relacionados con el tema de la investigación, también se especifica temas relevantes que sirve de base para la construcción del software y tecnologías informáticas utilizadas en el desarrollo de este.

2.1. ANTECEDENTES DE LA INVESTIGACION

Los antecedentes de la investigación encontrados, son descritos a continuación:

- Megaservidores SQL Server [GW04]: escalabilidad, disponibilidad y capacidad de administración. Microsoft SQL Server™ ha evolucionado para adaptarse a las bases de datos y aplicaciones de grandes dimensiones, incluidas las bases de datos de varios terabytes de capacidad compartidas por millones de personas. SQL Server logra proporcionar esta escalabilidad porque admite tanto el escalado vertical en los sistemas de varios procesadores simétricos (SMP), lo que permite a los usuarios agregar más procesadores, memoria, discos y redes para generar un único nodo de grandes dimensiones, como el escalado horizontal en los clústeres de varios nodos, lo que permite dividir una base de datos grande en un clúster

de servidores, en el que cada servidor almacena una parte de la base de datos y asume una parte del trabajo, mientras la base de datos sigue accesible como una única entidad.

- Tecnología ORACLE RAC [DCA06]: Sistemas escalables y de alta disponibilidad. Son las características de OPS (Oracle Parallel Server). No pudo tener la difusión que esperaban por razones tanto de coste como por la necesidad de que las aplicaciones se desarrollasen de forma específica teniendo en cuenta su entorno, estuvo disponible a partir de la versión nueve de Oracle. La nueva versión de esta tecnología se bautizo como RAC (Real Application Cluster). Permite que cualquier aplicación se pueda beneficiar de características de alta disponibilidad y escalabilidad horizontal sin que tuviera que desarrollarse de forma específica para ese entorno. En la versión actual de Oracle conocida como 11g, la escalabilidad horizontal se ha convertido en la característica central de la base de datos. La tecnología RAC crea un cluster de servidores donde todas las máquinas están ejecutando la misma base de datos de forma simultánea distribuyendo el trabajo entre todos los nodos que componen el cluster. La arquitectura física de un cluster RAC es muy similar a la de un cluster tradicional: Son necesarios al menos dos servidores, estos servidores necesitan tener acceso a una red pública de datos por donde se van a conectar los clientes y otra privada, que se recomienda que sea de alta velocidad (1Gb). Por ésta última, va a circular la información que se produce entre los nodos.

- MySQL Cluster 7.0: Architecture and New Features [MySQL09]
  MySQL Cluster es una base de datos de alta disponibilidad construido con una arquitectura única de nada compartido y un interfaz estándar SQL.
  El sistema consta de una serie de procesos de comunicación, o nodos que puede ser distribuido a través de los hosts para garantizar la

disponibilidad continua en el caso de la falla del servidor o la red. MySQL Cluster utiliza un motor de almacenamiento, que consiste en un conjunto de nodos de datos para almacenar los datos que se puede acceder utilizando el estándar SQL con MySQL Server o a través de la NDB (Network Database) API en tiempo real de acceso.
El NDB API es un interfaz de programación de aplicaciones orientada a objetos para el Cluster MySQL que implementa índices, escanea, procesa y maneja eventos. Las transacciones NDB, son ACID (Atomicity, Consistency, Isolation and Durability) compatibles con el cluster de MySQL por la descripción de la Arquitectura.
MySQL Cluster tolera fallos de nodos de datos y vuelve a reconfigurarse sobre la marcha para afrontar estas fallas.

- Procesamiento paralelo de consultas SQL generadas desde la Web [BCLMO06]: Presenta una solución para el procesamiento paralelo de un gran número de consultas SQL de solo lectura (select-from-where) provenientes de un servidor Web. La base de datos se distribuye uniformemente sin duplicados en cada PC, y los datos son administrados por un software tradicional de bases de datos relacionales (SQL).

2.2. MARCO TEORICO

2.2.1. MIDDLEWARE

2.2.1.1. Definición

Middleware es un término, ampliamente usado y posee muchas definiciones. Entre las más importantes se tiene:

Middleware se define, como la capa de software que se coloca entre el usuario y el entorno distribuido, abstrayendo al usuario de la complejidad y la heterogeneidad de las arquitecturas, protocolos, sistemas operativos, lenguajes de programación; es decir, otorga la posibilidad de intercomunicar aplicaciones desarrolladas en distintos lenguajes de programación, sistemas operativos y plataformas [PMC06]. En otras palabras, engloba los elementos que permiten la comunicación entre los sistemas.
El middleware es un módulo intermedio que no pertenece a los dominios del servidor, ni a la interfaz de usuario, ni a la lógica de la aplicación en los dominios del cliente, el middleware es una interfaz lógica estándar de los servicios de red [DCCAI01].

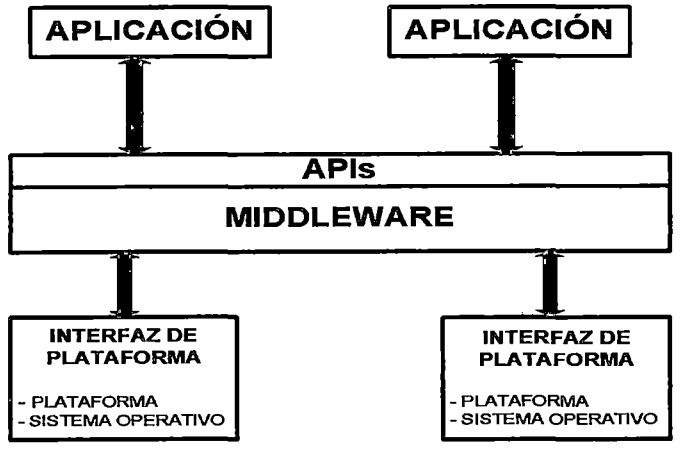


Figura 2 -1: Ejemplo de Middleware

2.2.1.2. Características

- Independiza el servicio de su implantación, del sistema operativo y de los protocolos de comunicación.
- Permite la convivencia de distintos servicios en una misma maquina.

2.2.1.3. Tipos de Middleware

Diversos autores han escrito acerca de los tipos o categorías de Middleware, lo que hace muy amplia la gama de clasificaciones de los middleware; sin embargo, [Ari02] hace una clasificación bastante completa. Esta categorización realizada por [Ari02], presenta nueve tipos de middleware, que son:

Distributed Tuples (DT)

Es el Middleware para acceso a base de datos que permite desarrollar sistemas independientes del manejador de base de datos que lo soporta.

Llamada a Procedimiento Remoto (Remote Procedure Call RPC)

Es el Middleware diseñado como servicio síncrono para permitir la gestión remota de redes. Esconde las operaciones de envío y recepción bajo el aspecto de una llamada convencional a una rutina o procedimiento. Los RPC tienen la misma semántica que las llamadas a procedimientos ordinarios; es decir, se realiza la llamada y se pasa el control al procedimiento servidor; cuando éste devuelve el resultado, el cliente recupera el control. El software que soporta RPC debe ocuparse de tres tareas importantes: la interfaz del servicio, la búsqueda del servidor, y la gestión de comunicación.

Middleware Orientado a Mensajes (Messaging Oriented Middleware MOM

Está diseñado para el servicio de mensajes con tecnología asíncrona. Permite el envío de mensajes entre aplicaciones, las aplicaciones sólo ponen y sacan mensajes de las colas, no se conectan. El cliente y el servidor pueden ejecutarse en diferentes tiempos (mensajes asíncronos), por lo que no necesariamente se requiere respuesta.

Middleware para tecnologías orientadas a objetos (Distributed Object Middleware DOM)

Los objetos piden servicio a otros objetos que se encuentran en la red. Se encarga de establecer comunicación entre los clientes y los objetos de forma transparente respecto a la distribución. Permite localizar a un objeto remoto dada una referencia a ese objeto. El núcleo de estos Middleware es Object Request Broker (ORB[5]). Ejemplo de este Middleware son: Common Object Request Broker Architecture (CORBA) de OMG, Remote Method Invocation (RMI) de SUN Microsystems y Distributed Component Object Model (DCOM) de Microsoft®.

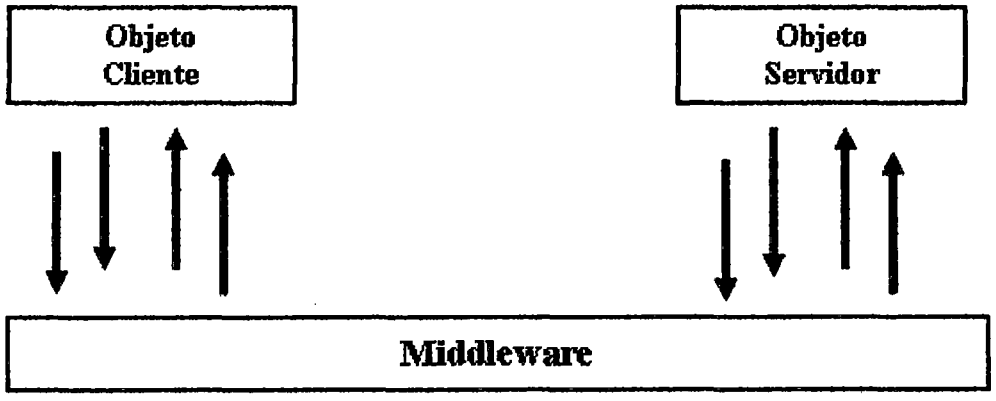


Figura 2 -2 : Middleware orientado a objetos

[5] **ORB:** Agente de Petición de Objeto (Object Request Broker); en una aplicación cliente/servidor, los servicios a los que accedemos pueden no estar disponibles siempre en un mismo servidor, de modo que estos servicios se solicitan a un agente, que es el que realmente conecta a la aplicación cliente con la aplicación servidor, sin que la aplicación cliente necesite saber dónde están ubicados físicamente estos servicios.

Middleware para Procesamiento de Transacciones (Transaction Processing Monitors TP Monitors)

Ya que facilita la conectividad y el acceso a un gran número de usuarios con servicios de back-end limitados. Este tipo de Middleware requiere del soporte de un Monitor; es decir, un programa que supervise las transacciones entre procesos, con el propósito de asegurar el éxito de la transacción, o en caso de ocurrir un error, tomar acciones apropiadas. Su principal uso es coordinar el flujo de solicitudes entre los dispositivos y las aplicaciones que procesan esas solicitudes. Algunos productos que proporcionan Middleware para procesamiento de transacciones son: BEA Tuxedo, Transarc's Encina y CICS[6] de IBM.

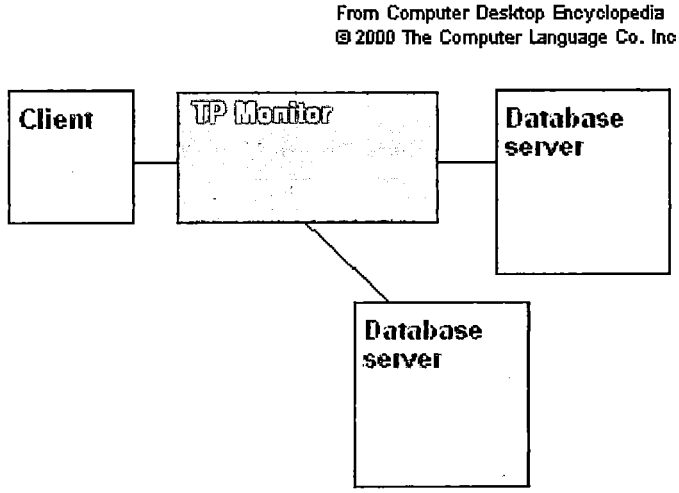


Figura 2 -3 : Middleware orientado a transacciones

Database Access Technology (DBAT)

Son las Aplication Programming Interface (API) creando una capa transparente para el acceso a base de datos, ocultando la complejidad dada por el manejador de base de datos. Ejemplo de estas API son Java

[6] **CICS O Monitor de Teleproceso:** Acrónimo en inglés de Customer Information Control System (Sistema de control de información de clientes), es un servidor de transacciones que se ejecuta principalmente en mainframes IBM con los sistemas operativos z/OS o VSE. También existen versiones de CICS para otros entornos, como OS/400, OS/2.

Database Connectivity (JDBC) desarrollado por SUN y Open Database Connectivity (ODBC) desarrollado por Microsoft®.

Component Oriented Framework (COF)

Este Middleware está soportado en el modelo de desarrollo de aplicaciones basado en componentes, permite la creación de una aplicación como conjunto de componentes reusables. Los componentes son una evolución del software orientado a objetos para dar respuesta a la reusabilidad. Los Middleware basados en componentes permiten a éstos "pegarse" con otros componentes para lograr la integración. Algunos ejemplos son: Enterprise Java Beans (EJB), especificación creada por SUN Microsystems; define un framework para los componentes Java del lado del servidor. COM+[7] es la siguiente generación de DCOM; simplifica la programación y es igualmente diseñado por Microsoft®. CORBA Component Model (CCM).

Middleware basado en directorios (Directory Services DS)

Permiten reducir costos administrativos, simplifican y/o distribuyen tareas administrativas, reducen el número de passwords para un usuario mediante una única combinación de login/password, aumentan la seguridad y proporcionan una única localización para la información de los usuarios. Son similares a la bases de datos, pero contienen información más descriptiva; la frecuencia de lectura sobre ellos es bastante más alta que la de escritura y en consecuencia su manejo es más simple que el de una base de datos ya que no hace actualizaciones complejas. LDAP Lightweight Directory Access Protocol (LDAP) es un protocolo para acceder a los DS. Como ejemplo, se tienen los Domain Name System (DNS) que conforman una fuente de datos distribuidas por múltiples máquinas en Internet, cuya tarea es convertir nombres en direcciones.

---

[7] **COM+:** Basado sobre Microsoft (component object model COM), para crear components básicos y aplicaciones distribuidas.

Application Servers (AS)

Es el Middleware que se enfoca en la parte de la de aplicación o lógica del negocio.

Conceptualmente un sistema puede tener muchas capas; sin embargo, la arquitectura más popular es de tres capas: interfaz, aplicación y base de datos. Aquí se encuentran, por ejemplo, Internet Information Servers (IIS), TOMCAT, Apache, Oracle9i Application Server, IBM WebSphere Application Server. Como se aprecia, las tecnologías Middleware fundamentalmente dependen de los mecanismos que permiten la comunicación entre los sistemas. Es decir, todas las tecnologías Middleware permiten comunicación. La diferencia entre ellas se encuentra en el énfasis que presentan como apoyo en aspectos como datos y procesos.

2.3. ARQUITECTURAS PARALELAS

Los sistemas de cómputo con procesamiento en paralelo surgen de la necesidad de resolver problemas complejos en un tiempo razonable, utilizando las ventajas de memoria, velocidad de los procesadores, formas de interconexión de estos y distribución de la tarea, a los que en su conjunto se les denomina arquitectura en paralelo. Se entiende por una arquitectura en paralelo a un conjunto de procesadores interconectados capaces de cooperar en la solución de un problema [Car06].

2.3.1. Clasificación de las Arquitecturas Paralelas

Clasificación moderna hace alusión única y exclusivamente a los sistemas que tienen más de un procesador (por ejemplo máquinas paralelas).
La clasificación moderna divide a los sistemas en dos tipos:

- Sistemas multiprocesador (fuertemente acoplados) y
- Sistemas multicomputadores (débilmente acoplados).

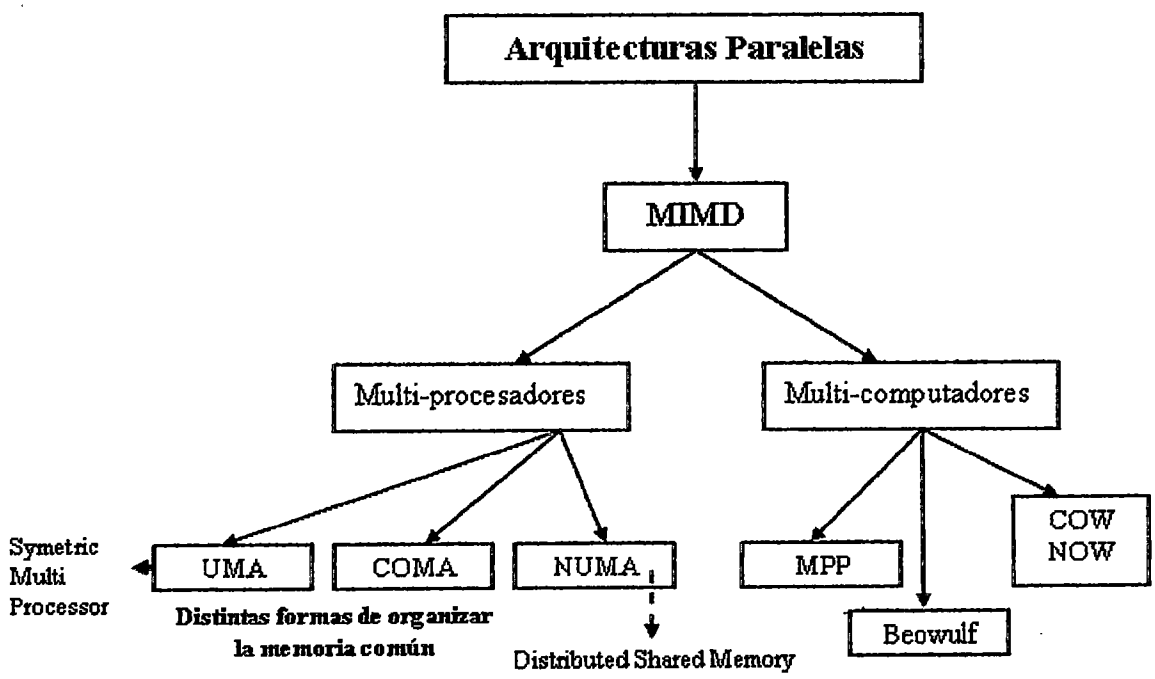


Figura 2 4 : Clasificación de arquitecturas paralelas [BEL06]

2.3.2. Multicomputadoras o Equipo Paralelo de Memoria Distribuida

Los sistemas multicomputadoras se pueden ver como una computadora paralela en el cual cada procesador tiene su propia memoria local. En estos sistemas la memoria se encuentra distribuida y no compartida como en los sistemas multiprocesador.

Los procesadores se comunican a través de paso de mensajes, ya que éstos sólo tienen acceso directo a su memoria local y no a las memorias del resto de los procesadores.

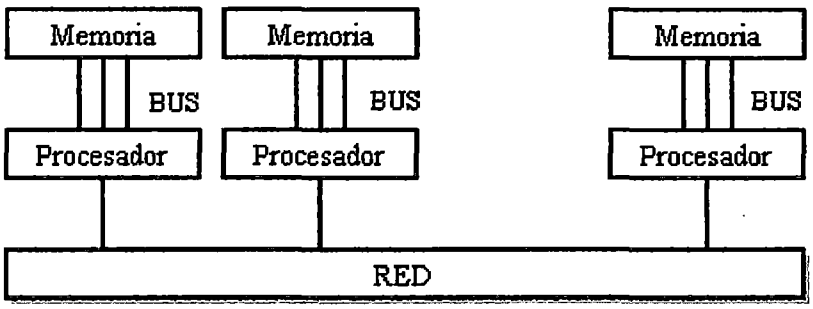


Figura 2 5 : Arquitectura de una computadora paralela con memoria distribuida

La transferencia de los datos se realiza a través de la red de interconexión que conecta un subconjunto de procesadores con otro subconjunto. La transferencia de unos procesadores a otros se realiza por múltiples transferencias entre procesadores conectados dependiendo del establecimiento de dicha red.

Dado que la memoria está distribuida entre los diferentes elementos de proceso, estos sistemas reciben el nombre de distribuidos. Por otra parte, estos sistemas son débilmente acoplados, ya que los módulos funcionan de forma casi independiente unos de otros. Este tipo de memoria distribuida es de acceso lento por ser peticiones a través de la red, pero es una forma muy efectiva de tener acceso a un gran volumen de memoria.

2.3.3. Cluster de Computadoras

Un clúster es un sistema de equipos independientes acoplados dispersamente que actúa como un solo sistema. Los nodos del clúster pueden estar formados por sistemas de un solo procesador o SMPs. Los nodos pueden estar conectados mediante una red o un bus de comunicaciones de elevada velocidad de una marca determinada. Los equipos de un clúster trabajan en conjunto para que los clientes vean el clúster como un único servidor altamente confiable y de elevado rendimiento. Al contar con un diseño modular, el clúster puede escalarse horizontalmente a incrementos y a un costo asequible mediante la adición de servidores, discos y redes [GW04].

Los Clusters han evolucionado para apoyar actividades en aplicaciones que van desde súper cómputo y software de misiones críticas, servidores Web y comercio electrónico, bases de datos de alto rendimiento.
El cómputo en Clusters surge como resultado de la convergencia de varias tendencias que incluyen, la disponibilidad de microprocesadores de alto rendimiento más económicos y redes de alta velocidad, el desarrollo de herramientas de software para cómputo distribuido de alto rendimiento, y la creciente necesidad de potencia computacional para aplicaciones en las ciencias computacionales y comerciales.

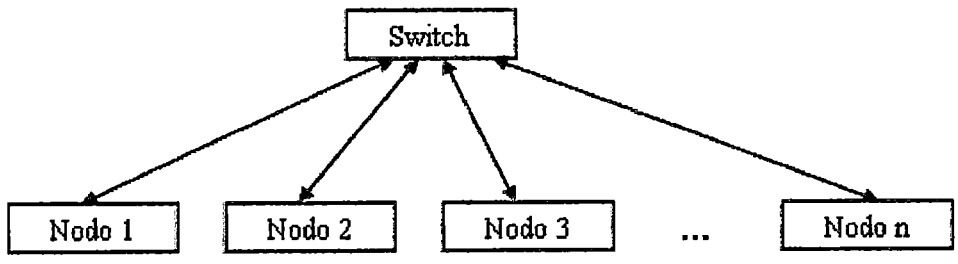


Figura 2 -6 : Arquitectura de un cluster

2.3.3.1. Tipos de Cluster

Existen 3 tipos de clusters que son:

Cluster de Alta Disponibilidad o Cluster de Redundancia (Fail-over o High-Availability)

Este tipo de cluster esta diseñado para mantener uno o varios servicios disponibles incluso a costa de rendimiento, ya que su función principal es que el servicio jamás tenga interrupciones como por ejemplo un servicio de bases de datos.

Cluster de Alto Rendimiento (HPC o High Performance Computing)

Es un número grande de máquinas individuales que actúan como una sola máquina muy potente. Este tipo de clusters se aplica mejor en problemas grandes y complejos que requieren una cantidad enorme de potencia computacional. Entre las aplicaciones más comunes de clusters de alto rendimiento se encuentra el pronóstico numérico del estado del tiempo, astronomía, investigación en criptografía, análisis de imágenes.

Cluster de Servidores Virtuales

Permite que un conjunto de servidores de red compartan la carga de trabajo de tráfico de sus clientes. Al balancear la carga de trabajo de tráfico en un arreglo de servidores, mejora el tiempo de acceso y confiabilidad. Además como es un conjunto de servidores el que atiende el trabajo, la falla de uno de ellos no ocasiona una falla catastrófica total. Este tipo de servicio es de gran valor para compañías que experimental grandes volúmenes de tráfico en sus sitios Web.

2.4. PARALELISMO

El paralelismo es un caso particular de la concurrencia [SQS06].
Se habla de paralelismo cuando ocurre la ejecución simultánea de instrucciones.
El procesamiento paralelo tiene como principal objetivo explotar el paralelismo inherente a las aplicaciones informáticas. Todas las aplicaciones no presentan el mismo grado de paralelismo: unas se pueden paralelizar mucho y en cambio otras muy poco. Al lado de este factor cuantitativo evidente, es necesario considerar también un factor cualitativo: la manera a través de la cual se explota el paralelismo. Cada técnica de explotación del paralelismo se denomina fuente [Par02]. Se distingue tres fuentes principales:

- El paralelismo de control
- El paralelismo de datos
- El paralelismo de flujo

2.4.1. El Paralelismo de Control

La explotación del paralelismo de control proviene de la constatación natural de que en una aplicación existen acciones que podemos "hacer al mismo tiempo". Las acciones, llamadas también tareas o procesos pueden ejecutarse de manera más o menos independiente sobre unos recursos de cálculo llamados también procesadores elementales (PE) [Par02].

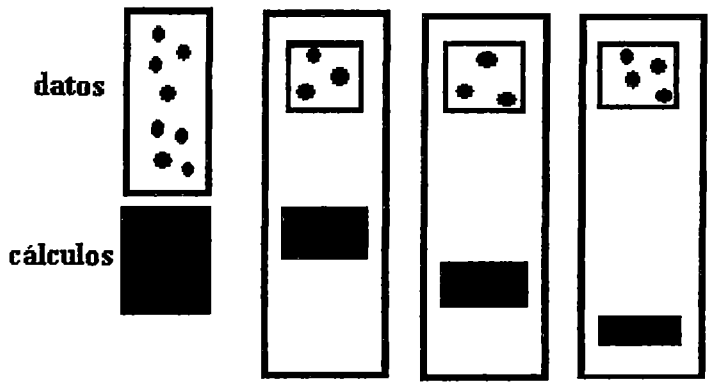


Figura 2 -7 : Paralelismo de Control

En el caso de que todas las acciones sean independientes es suficiente asociar un recurso de cálculo a cada una de ellas para obtener una ganancia en tiempo de ejecución que será lineal: N acciones independientes se ejecutarán N veces más rápido sobre N Elementos de Proceso (PE) que sobre uno solo. Este es el caso ideal, pero las acciones de un programa real suelen presentar dependencias entre ellas. Se distingue dos clases de dependencias que suponen una sobrecarga de trabajo:

- Dependencia de control de secuencia: corresponde a la secuenciación en un algoritmo clásico.
- Dependencia de control de comunicación: una acción envía informaciones a otra acción.

La explotación del paralelismo de control consiste en administrar las dependencias entre las acciones de un programa para obtener así una asignación de recursos de cálculo lo más eficaz posible, minimizando estas dependencias.

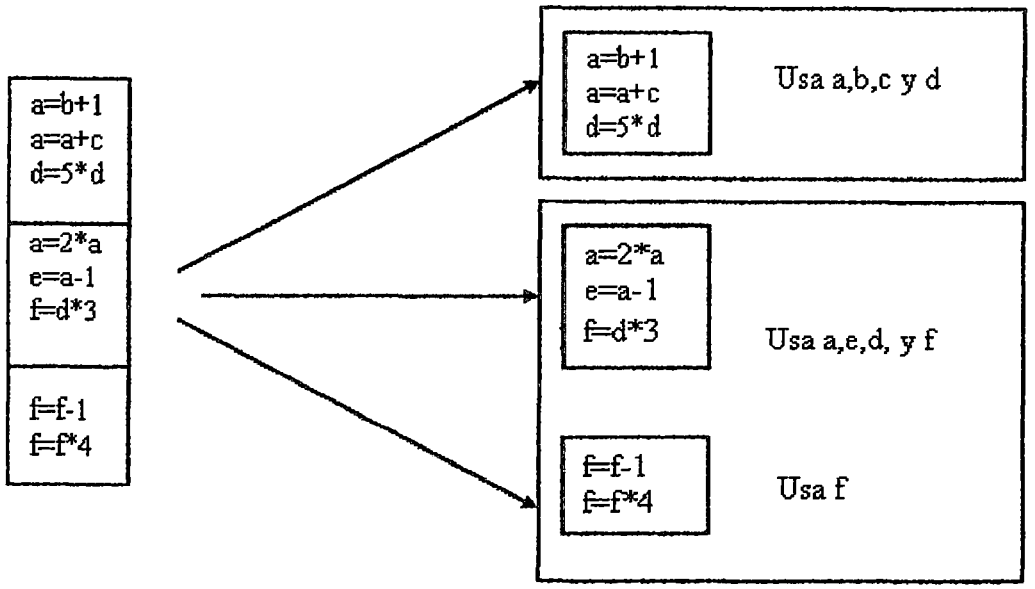


Figura 2 8 : Ejemplo de paralelismo de control aplicado a la ejecución simultánea de instrucciones

2.4.2. El Paralelismo de Datos

La explotación del paralelismo de datos proviene de la constatación natural de que ciertas aplicaciones trabajan con estructuras de datos muy regulares (vectores, matrices) repitiendo una misma acción sobre cada elemento de la estructura. Los recursos de cálculo se asocian entonces a los datos. A menudo existe un gran número (millares o incluso millones) de datos idénticos. Si el número de PE (Elementos de Proceso) es inferior al de datos, éstos se reparten en los PE disponibles.

Como las acciones efectuadas en paralelo sobre los PE son idénticas, es posible centralizar el control. Siendo los datos similares, la acción a repetir tomaría el mismo tiempo sobre todos los PE y el controlador podría enviar, de manera síncrona, la acción a ejecutar a todos los PE.

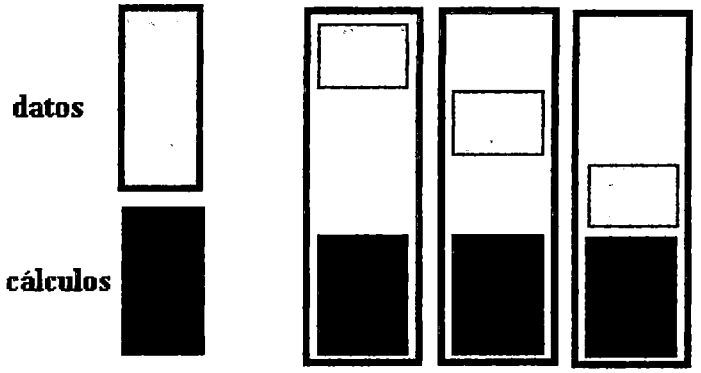


Figura 2 9 : Paralelismo de datos

Las limitaciones de este tipo de paralelismo vienen dadas por la necesidad de dividir los datos vectoriales para adecuarlos al tamaño soportado por la máquina, la existencia de datos escalares que limitan el rendimiento y la existencia de operaciones de difusión (un escalar se reproduce varias veces convirtiéndose en un vector) y Beta-reducciones que no son puramente paralelas.

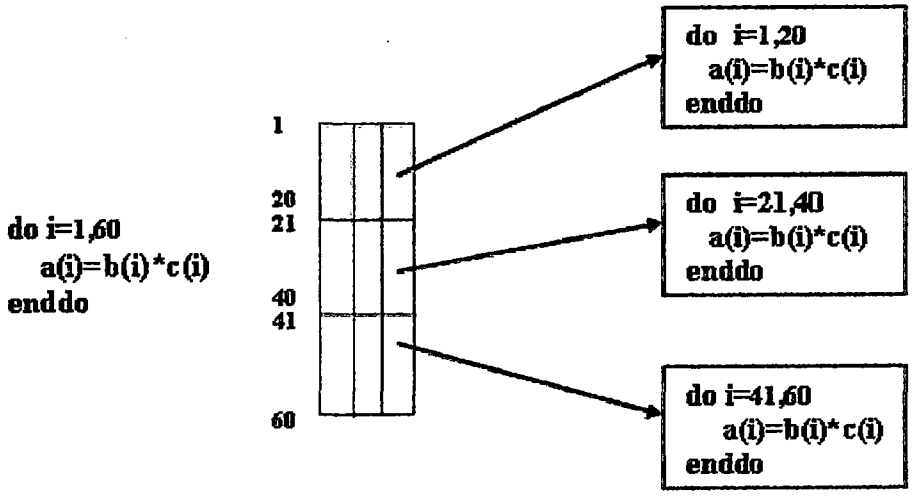


Figura 2 10 : Ejemplo de la aplicación del paralelismo de datos a un bucle

2.4.3. El Paralelismo de Flujo

La explotación del paralelismo de flujo proviene de la constatación natural de que ciertas aplicaciones funcionan en modo cadena: disponemos de un flujo de datos, generalmente semejantes, sobre los que debemos efectuar una sucesión de operaciones en cascada.

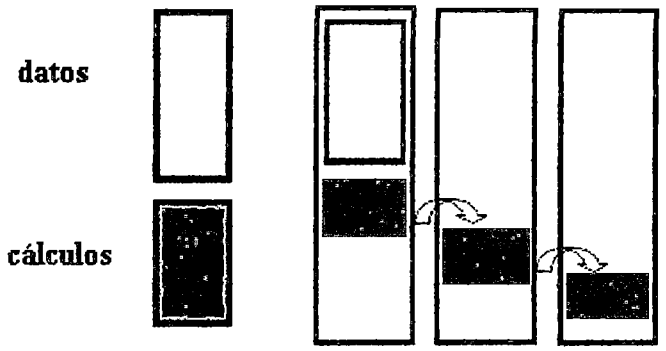


Figura 2 -11 : Paralelismo de flujo

Los recursos de cálculo se asocian a las acciones y en cadena, de manera que los resultados de las acciones efectuadas en el instante t pasen en el instante t + 1 al PE siguiente. Este modo de funcionamiento se llama también segmentación o pipeline.
El flujo de datos puede provenir de dos fuentes:

- Datos de tipo vectorial ubicados en memoria. Existe entonces una dualidad fuerte con el caso del paralelismo de datos.
- Datos de tipo escalar provenientes de un dispositivo de entrada. Este dispositivo se asocia a menudo a otro de captura de datos, colocado en un entorno de tiempo real.

En ambos casos, la ganancia obtenida está en relación con el número de etapas (número de PE). Todos los PEs no estarán ocupados mientras el primer dato no haya recorrido todo el cauce, lo mismo ocurrirá al final del

flujo. Si el flujo presenta frecuentes discontinuidades, las fases transitorias del principio y del fin pueden degradar seriamente la ganancia. La existencia de bifurcaciones también limita la ganancia obtenida.

2.5. BASE DE DATOS

2.5.1. Definición

Se define una base de datos como una serie de datos organizados y relacionados entre sí, y un conjunto de programas que permitan a los usuarios acceder y modificar esos datos.

Las bases de datos proporcionan la infraestructura requerida para los sistemas de apoyo a la toma de decisiones y para los sistemas de información estratégicos, ya que estos sistemas explotan la información contenida en las bases de datos de la organización para apoyar el proceso de toma de decisiones o para lograr ventajas competitivas. Por este motivo es importante conocer la forma en que están estructuradas las bases de datos y su manejo [Bur06].

Uno de los propósitos principales de un sistema de base de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.

2.5.2. Bases de Datos Relacionales

Este es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José California, no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado

de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

El modelo relacional es un ejemplo de un modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos, o atributos.

2.5.3. Bases de Datos Distribuidas

Una base de datos distribuida (BDD) es la unión de las bases de datos con Redes distribuidas geográficamente.

La base de datos está almacenada en varias computadoras conectadas en red, (ya sea en el mismo lugar físicamente o distribuidas a lo largo de la red) lo que permite al acceso de datos desde diferentes máquinas. Está manejada por el Sistema de Administración de Datos Distribuida SABDD o Sistema de Gestión de Base de datos distribuida. Son la evolución de los Cliente-Sevidor.

La razón principal detrás de las BDD son los organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a la información, sin tener todo centralizado en un solo punto. Ejemplo: bancos, cadenas de hoteles, campus de distintas universidades, sucursales de tiendas departamentales.

Los principales problemas que se generan por el uso de la tecnología de bases de datos distribuidas son en lo referente a duplicidad de datos y a su integridad al momento de realizar actualizaciones a los mismos. Además, el control de la información puede constituir una desventaja, debido a que se encuentra diseminada en diferentes localidades geográficas.

2.5.4. Bases de Datos Paralelas

2.5.4.1. SGBD Paralelo

SGBD paralelo: Un SGBD que se ejecuta sobre múltiples procesadores y discos que han sido diseñados para ejecutar operaciones en paralelo (cuando sea posible) con el propósito de mejorar el rendimiento.
La fuerza que ha impulsado los sistemas paralelos de bases de datos ha sido la demanda de aplicaciones que han de manejar bases de datos extremadamente grandes (del orden de terabytes), o que tienen que procesar un número enorme de transacciones por segundo (del orden de miles). Para medir el rendimiento existen dos medidas:

- Productividad (Throughput): número de tareas en un intervalo de tiempo
- Tiempo de respuesta: cantidad de tiempo para completar una tarea desde que se envió.

El objetivo del paralelismo en los sistemas de bases de datos suele ser asegurar que la ejecución del sistema continuará realizándose a una velocidad aceptable, incluso en el caso de que aumente el tamaño de la base de datos o el número de transacciones (ampliabilidad).
Existen algunos factores que trabajan en contra de la eficiencia del paralelismo y pueden atenuar tanto la ganancia de velocidad como la ampliabilidad:

- *Costes de inicio.* El inicio de un único proceso lleva asociado un coste de inicio. En una operación paralela compuesta por miles de procesos, el tiempo de inicio puede llegar a ser mucho mayor que el tiempo real de procesamiento, lo que influye negativamente en la ganancia de velocidad.

- *Interferencia.* Como los procesos que se ejecutan en un sistema paralelo acceden con frecuencia a recursos compartidos, pueden sufrir un cierto retardo como consecuencia de la interferencia de cada nuevo proceso en la

competencia con los procesos existentes por el acceso a los recursos más comunes, como el bus del sistema, los discos compartidos o incluso los bloqueos. Este fenómeno afecta tanto a la ganancia de velocidad como a la ampliabilidad.

- *Sesgo.* Al dividir cada tarea en un cierto número de pasos paralelos se reduce el tamaño del paso medio. Es más, el tiempo de servicio de la tarea completa vendrá determinado por el tiempo de servicio del paso más lento. Normalmente es difícil dividir una tarea en partes exactamente iguales, entonces se dice que la forma de distribución de los tamaños es sesgada [CHMP05].

2.5.5. Replicación de Base de Datos

La réplica de la base de datos es la creación y el mantenimiento de copias múltiples de la misma base de datos.

En la mayoría de aplicaciones de réplica de base de datos, un servidor mantiene la copia principal de la base de datos y los servidores adicionales mantienen las copias auxiliares de la base de datos principal.

Las actualizaciones a la base de datos se envían al servidor principal de la base de datos y después es propagada a los servidores auxiliares de la base de datos.

Las lecturas de la base de datos se dividen entre todos los servidores de la base de datos, que da lugar a una ventaja grande del funcionamiento debido a la distribución de carga.

Además, la réplica de la base de datos puede también mejorar la disponibilidad porque los servidores auxiliares de la base de datos se pueden configurar para que asuman el control que desempeña el servidor principal de base de datos si este llega a ser inasequible.

La replicación nos facilitará la reconstrucción de las estructuras sobre la base de datos de respaldo a partir de la información obtenida durante el análisis de la base de datos de producción. Además, una vez llevada a cabo la replicación, podremos realizar las actualizaciones sobre la base de datos de respaldo de manera que esta se mantenga siempre en un estado adecuado para suplantar a la base de datos de producción en caso de ser necesario [Per02].

2.6. ARQUITECTURA DE SOFTWARE (AS)

2.6.1. Definición

Según Clements [Cle96a]: La Arquitectura de Software (AS) es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.
Frente a la abundancia de definiciones del campo de la AS, existe en general un acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos [BCK98].
La definición "oficial" de Arquitectura de Software es la que brinda el documento de la IEEE Std 1471-2000, adoptada también por Microsoft:
"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el contexto en que se implantaran, y los principios que orientan su diseño y evolución".
Obsérvese entonces que la noción clave de la arquitectura es la organización (un concepto cualitativo o estructural).

2.6.2. Importancia

El desarrollo de la arquitectura de software es una de las etapas fundamentales y, en muchos casos, la más importante en el desarrollo de software, pues es aquí donde los profesionales aportan todos sus conocimientos, creatividad y experiencia para crear la mejor propuesta de solución que se dará al cliente que cumpla con los requerimientos funcionales y no funcionales establecidos para el sistema en desarrollo, así como sus preocupaciones principales de lo que esperan del sistema [Her06].

Para ello, también es importante que los diferentes interesados en el sistema se involucren en el diseño de la arquitectura pues, con ello, se podrá acordar y consensuar de una mejor manera la solución a la que se llegue después de conocer los requerimientos.

Desarrollar una arquitectura de software es como llevar a cabo el diseño arquitectónico de un edificio que será construido. Para construir, los ingenieros, albañiles, plomeros, electricistas, etc. requieren estudiar y comprender los planos de los cimientos, la estructura, y de toda la infraestructura necesaria para que cuente con servicios tales como: luz, agua, teléfono, red de datos, etc. Además de aportar sus propios puntos de vista y experiencia, Cuando no existen estos planos que nos guíen, estaremos construyendo algo que se nos viene a la imaginación justo en el momento de realizarlo.

2.6.3. Vistas de la Arquitectura del Sistema

Como se mencionó en el tópico anterior la AS presenta al Sistema desde diferentes perspectivas, estas son conocidas como vistas (views); la cantidad y tipos de vistas difieren en función de cada tendencia arquitectónica

Uno de los más aceptados es el framework arquitectónico 4+1 presentado en [Kru95] que además va vinculado al RUP, este framework define cuatro vistas para la arquitectura (4) en conjunto con los escenarios (1), como se puede ver en la siguiente figura:

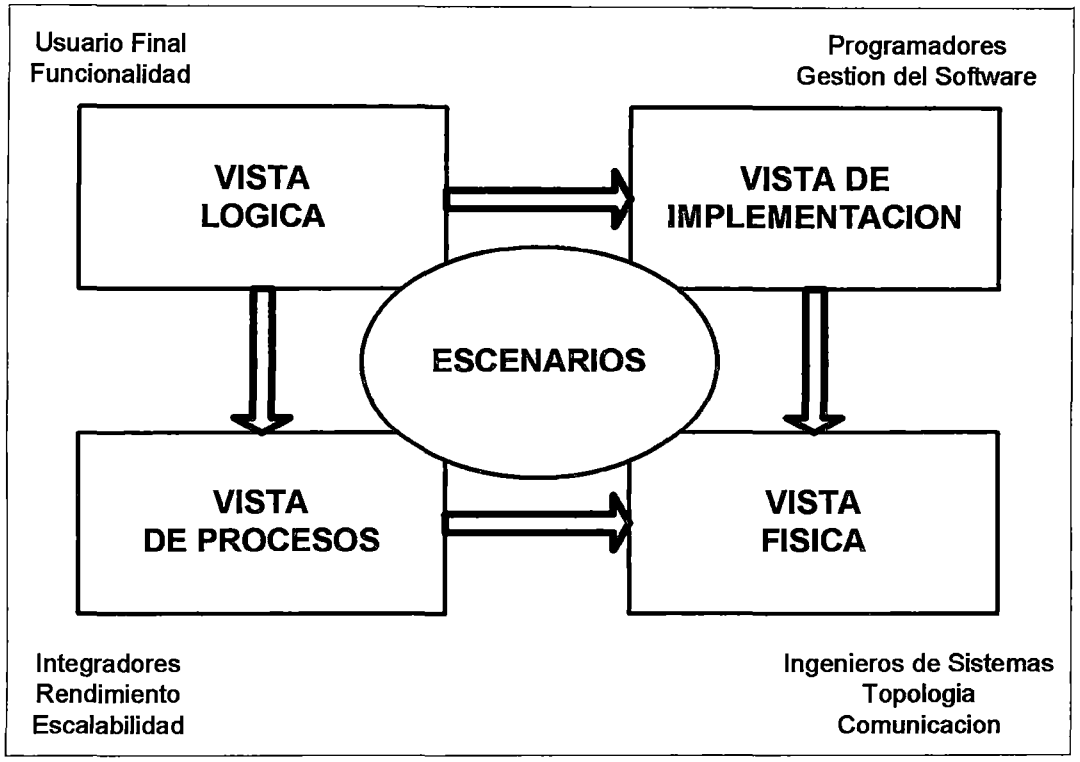


Figura 2 12 : Framework arquitectónico 4+1

Además de las 4 vistas incluiremos las siguientes vistas que forman parte del escenario, que van orientadas hacia los requerimientos funcionales y no funcionales del sistema:

- Vista de Casos de Uso
- Vista de Restricciones
- Vista QoS

A continuación una descripción de las vistas:

- Vista de Casos de Uso: Describe el proceso de negocio más significativo y el modelo del dominio. Presenta los actores y los casos de uso para el sistema (requerimientos funcionales).

- Vista de Restricciones: Describe restricciones tecnológicas, normativas, uso de estándares, entre otros, las cuales deben ser

respetadas tanto por el proceso de desarrollo como por el producto desarrollado.

- Vista QoS: Incluye aspectos de calidad, y describe los requerimientos no-funcionales del sistema.
- Vista Lógica: Describe la arquitectura del sistema presentando varios niveles de refinamiento. Indica los módulos lógicos principales, sus responsabilidades y dependencias.
- Vista de Procesos: Describe los procesos concurrentes del sistema.
- Vista de Implementación: Describe los componentes de deployment construidos y sus dependencias.
- Vista Física o de Deployment: También conocida como Vista de Despliegue presenta aspectos físicos como topología, infraestructura informática, e instalación de ejecutables. Incluye además plataformas y software de base.

2.6.4. Diferencia entre Arquitectura y Diseño

Aun existen discrepancias en si la Arquitectura y el Diseño son la misma cosa, hay quienes indican claramente que la arquitectura se encuentra en un nivel de abstracción por encima del diseño como un artefacto mas en el proceso de desarrollo de software [RK04].

En alguna medida, la arquitectura y el diseño sirven al mismo propósito. Sin embargo, el foco de la AS en la estructura del sistema y en las interconexiones la distingue del diseño de software tradicional, tales como el diseño orientado a objetos, que se concentra más en el modelado de abstracciones de más bajo nivel, tales como algoritmos y tipos de datos. A medida que la arquitectura de alto nivel se refina, sus conectores pueden

perder prominencia, distribuyéndose a través de los elementos arquitectónicos de más bajo nivel, resultando en la transformación de la arquitectura en diseño.

Queda claro que la arquitectura va antes del diseño del software como una visión general del Sistema, presentado desde diferentes puntos de vista (views), cada punto de vista dirigido a un sector específico del equipo de desarrollo.

2.6.5. Documento de Arquitectura de Software (SAD)

Este documento se elabora en conjunto entre los interesados en el desarrollo y los desarrolladores, en el se documentan las vistas de la arquitectura del sistema y se explica el por que de cada vista.
Aunque no existe un esquema en particular acerca de lo que debe contener el SAD presentamos el esquema sugerido en [Kru95]

Titulo
Historia de Cambios
Tabla de Contenidos
Lista de Figuras
1. Alcance
2. Referencias
3. Representación de la Arquitectura del Software
4. Objetivos y Restricciones
5. Arquitectura Lógica
6. Arquitectura de Procesos
7. Arquitectura de Desarrollo
8. Arquitectura Física
9. Escenarios
10. Tamaño y Rendimiento
11. Calidad

Apéndices

A. Acrónimos y Abreviaciones

B. Definiciones

C. Principios del Diseño

2.7. TRANSACTION PROCESSING PERFORMANCE COUNCIL – TPC (Concilio de Rendimiento para Procesamiento de Transacciones)

2.7.1. Definición

El TPC es una corporación sin fines de lucro fundada para definir los benchmarks[8] de bases de datos y procesamiento de transacciones, así como para difundir a los clientes, información objetiva y verificable sobre el rendimiento de los equipos.

En 1989, TPC publicó su primer benchmark, el TPC Benchmark A (TPC-A). Según esta prueba, el 90 por ciento de las transacciones debían completarse en menos de dos segundos; el número de terminales interactuando con el sistema que estaba siendo sometido a prueba (SUT - system under test), fue reducida a un requerimiento de 10 terminales por TPS y su costo estaba incluido en el sistema de precios; TPC-A podía correr en la configuración de una red de área local. Los requerimientos del benchmark, orientados a la producción, fueron fortalecidos para prevenir el reporte de picos e insostenibles índices de rendimientos. Específicamente, los requerimientos ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) fueron apoyados con la adición de pruebas específicas para asegurar la viabilidad de estos requisitos. Finalmente, TPC-A definió que toda la información de las pruebas de benchmark, debía ser revelada en un reporte completamente abierto.

Los primeros resultados del TPC-A fueron anunciados en julio de 1990. Cuatro años después, en la cima de su popularidad, 33 compañías fueron publicadas en los benchmarks TPC y 115 sistemas diferentes tuvieron

[8] Prueba para la evaluación del desempeño de hardware o software. Prueba realizada para evaluar la funcionalidad de sistemas de cómputo, programas o dispositivos.

resultados publicados por TPC-A. En total, aproximadamente 300 resultados fueron difundidos.

Con el transcurrir del tiempo, los benchmark han ido evolucionando. Para julio de 1992 fue aprobado un nuevo benchmark, llamado TPC-C, cuyo primer resultado se publicó en septiembre del mismo año. Para los momentos actuales, el consejo cuenta con las nuevas pruebas:
Benchmarks Activas
– TPC-C: Online transaction processing (OLTP)
– TPC-H: Decisión support for ad hoc queries
– TPC-App (application server benchmark)
Benchmarks en desarrollo
– TPC-E (OLTP)
– TPC-DS (decisión support) [TPC-H].

2.7.2. TPC Benchmark H (TPC-H)

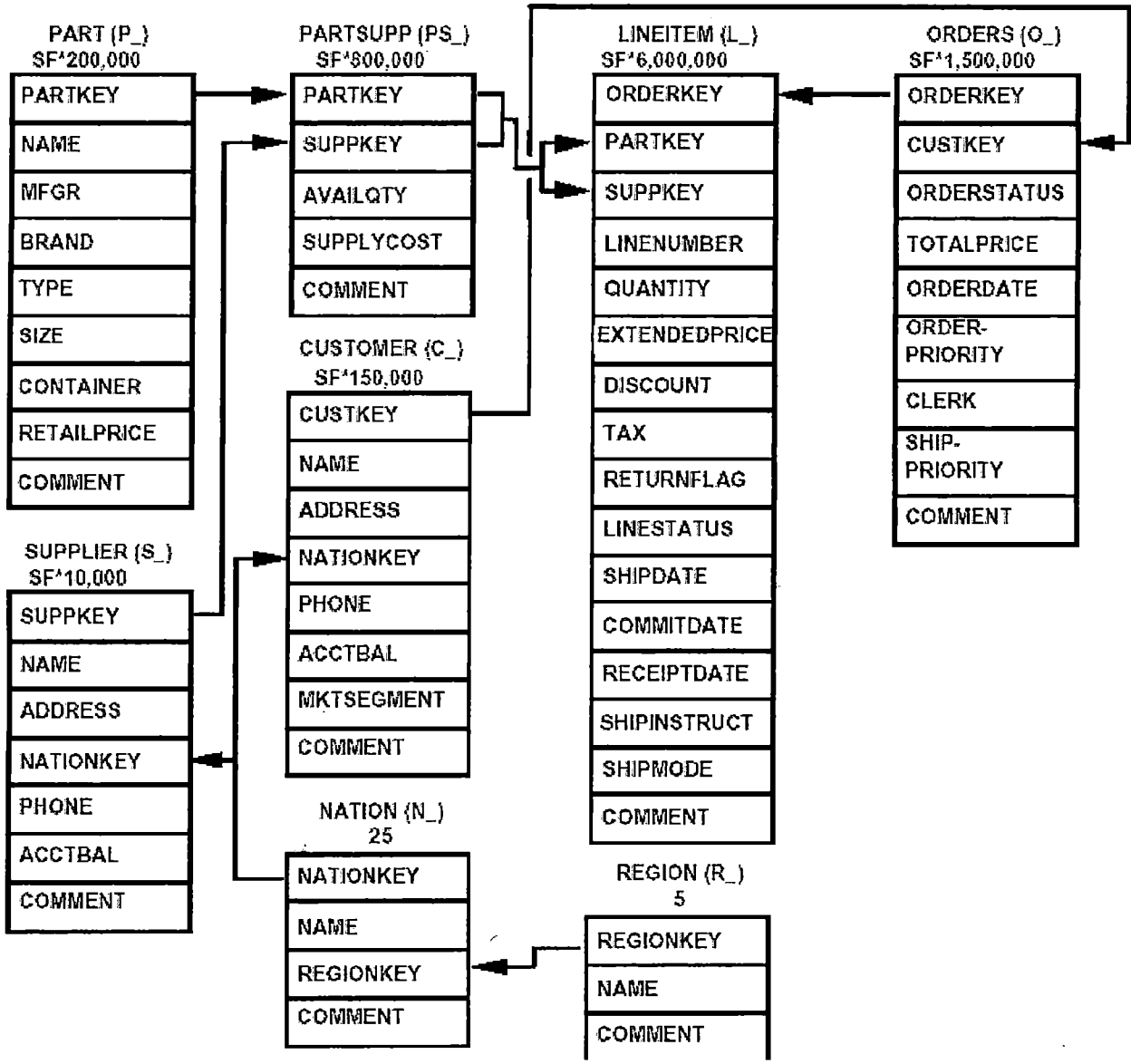
El Benchmark TPC-H es una prueba de rendimiento a sistemas de soporte de decisiones. Consiste en una "suite" de negocios orientados a búsquedas convenientes y modificaciones simultáneas de datos. Las búsquedas y la población de datos han sido elegidas para tener una amplia relevancia en la industria. Este benchmark ilustra las decisiones de los sistemas de respaldo que examinan grandes volúmenes de datos, ejecutan búsquedas con un elevado grado de complejidad y responde a situaciones críticas de negocios.

La medida de rendimiento reportada por TPC-H es llamado "rendimiento métrico compuesto por consultas por hora TPC-H" (QphH@Size), y refleja múltiples aspectos de las capacidades de los sistemas para procesar búsquedas. Estos, incluyen la selección del tamaño de bases de datos contra las cuales son ejecutadas las búsquedas; el poder de procesamiento de éstas cuando son presentadas por un flujo sencillo; y cuando son

realizadas por múltiples usuarios, de manera simultánea. La relación métrica TPC-H precio/rendimiento es expresada $/QphH@Size [TPC-H].

2.7.2.1. Base de Datos TPC - H

La base de datos que utiliza para realizar las pruebas de rendimiento es un data warehouse donde esta almacenada información historia.



Leyenda

- Los paréntesis con una letra en su interior, indica el prefijo del nombre de la tabla.
- Las fechas indicas las relaciones entre las tablas.
- El número que esta encima de cada tabla, indica el número de filas que contiene cada tabla. Estos son definidos por un factor de escala.

CAPITULO III

ALGORITMOS DE SUBDIVISION DE CONSULTAS PARA SU PROCESAMIENTO PARALELO

En este capitulo se muestra el funcionamiento, pseudocódigo y las diferencias entre el Algoritmo de división horizontal y el algoritmo de división vertical de consultas, además de algunas ventajas y desventajas de ambos enfoques.

Además se explica el algoritmo de división vertical planteado en este trabajo de tesis, el cual será implementado en el prototipo que será desarrollado en los subsecuentes capítulos.

3.1. INTRODUCCION

Se tiene un cluster de computadores, en donde cada nodo cuenta con un replica total de la base de datos a manejar, además cada nodo cuenta con un SGBD el cual se encarga de administrar su replica correspondiente.

Para que el software use de manera eficientemente el cluster, se debe diseñar un algoritmo que permita dividir la carga de procesamiento entre todos sus nodos, y así conseguir un procesamiento paralelo. Se conoce que las consultas llegaran al software de manera secuencial y serán recibidas

en una cola para su procesamiento; se presentan dos maneras de dividir esta cola entre los nodos:

- División horizontal de las consultas
- División vertical de las consultas

En la siguiente sección se muestra en que consisten, ventajas y desventajas, los problemas que presentan y se plantea algoritmos para cada caso.

3.2. DIVISION HORIZONTAL DE LAS CONSULTAS

La división horizontal también conocida como división inter-consulta [BUD08] consiste en dividir las consultas de la cola de entrada entre los nodos del cluster, de tal manera que cada nodo se encargue de procesar una consulta a la vez, con esto se consigue el procesamiento paralelo de las consultas.
A continuación se presenta un grafico de su funcionamiento.

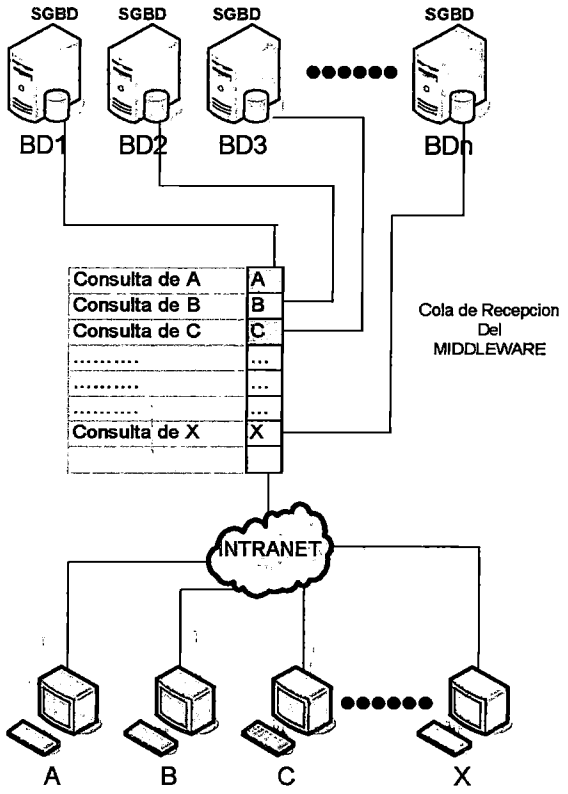


Figura 3-1 : División Horizontal de Consultas

3.2.1. Ventajas de la División Horizontal

Este tipo de división es la mas sencilla de implementar y se caracteriza por que eleva el numero de consultas procesadas por hora (QpH), este es un factor importante que se debe tener en cuenta.

Es útil en Sistemas que reciben un gran número de consultas simultáneas, como por ejemplo un Web Server. Por lo tanto este tipo de división es óptimo, cuando lo que se requiere es procesar el mayor número de consultas en el menor tiempo posible.

3.2.2. Desventajas de la División Horizontal

Como se puede ver cada consulta por si sola no mejora su tiempo de respuesta, el rendimiento es solo apreciable a nivel global al procesar un gran numero de consultas.

3.3. DIVISION VERTICAL DE LAS CONSULTAS

La división vertical también conocida como división intra-consulta [BUD08] consiste en subdividir cada consulta de la cola de entrada, y luego procesar cada parte en un nodo diferente del cluster.

A continuación se presenta una grafica de su funcionamiento.

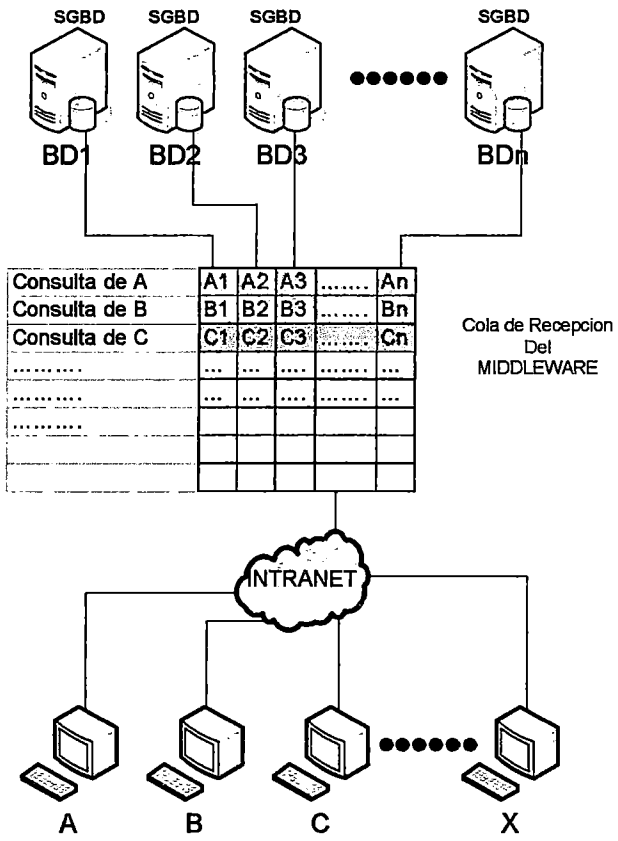


Figura 3-2: División Vertical de Consultas

3.3.1. Ventajas de la División Vertical

Esta división se caracteriza por que aparte de elevar el número de consultas procesadas por hora (QpH), mejora también el tiempo de respuesta individual de cada consulta.
Es útil en Sistemas en el que el tiempo de respuesta de las consultas es un factor crucial, como por ejemplo en Data WareHousing y Data Mining. Por lo tanto este tipo de división es óptimo cuando lo que se requiere es procesar una consulta dada en el menor tiempo posible.

3.3.2. Desventajas de la División Vertical

En este tipo de división se presentan otro tipo de problemas que no se presentaban o que eran más fáciles de tratar en la división horizontal: como por ejemplo el balance de carga, la reunión de los resultados y la manera de dividir la consulta.

3.4. ALGORITMOS DE DIVISION VERTICAL

3.4.1. Algoritmo de Partición Virtual Simple (PVS)

Este algoritmo debe su nombre a que se crean particiones virtuales de la tabla a procesar, considerando que cada nodo tiene una replica de la base de datos y un SGBD, este algoritmo limita la cantidad de datos a procesar por cada SGBD añadiendo predicados a las consultas. Haciendo esto fuerza a cada SGBD a procesar un diferente subconjunto de los datos. Por ejemplo consideremos la siguiente consulta, extraída del TPC-H Benchmark [TPC-H].

```
SELECT sum(precio * descuento) as redito
FROM Articulo
WHERE fechaVenta >= date '1994-01-01'
```

AND fechaventa < date '1994-01-01' + interval '1 year'
AND descuento BETWEEN 0.06-0.01 AND 0.06 + 0.01
AND cantidad < 24

Por simplicidad se usa un consulta sin joins, usando PVS esta consulta seria rescrita agregando los predicados "AND IdArticulo >=:p1 AND IdArticulo <:p2 a la cláusula WHERE. La consulta rescrita puede entonces ser enviada a los nodos que participaran en su procesamiento. Cada nodo recibirá la misma consulta pero con diferentes valores para p1 y p2, de tal manera que se escanee todo el rango de IdArticulo. Como se ve en el siguiente ejemplo para 2 nodos.

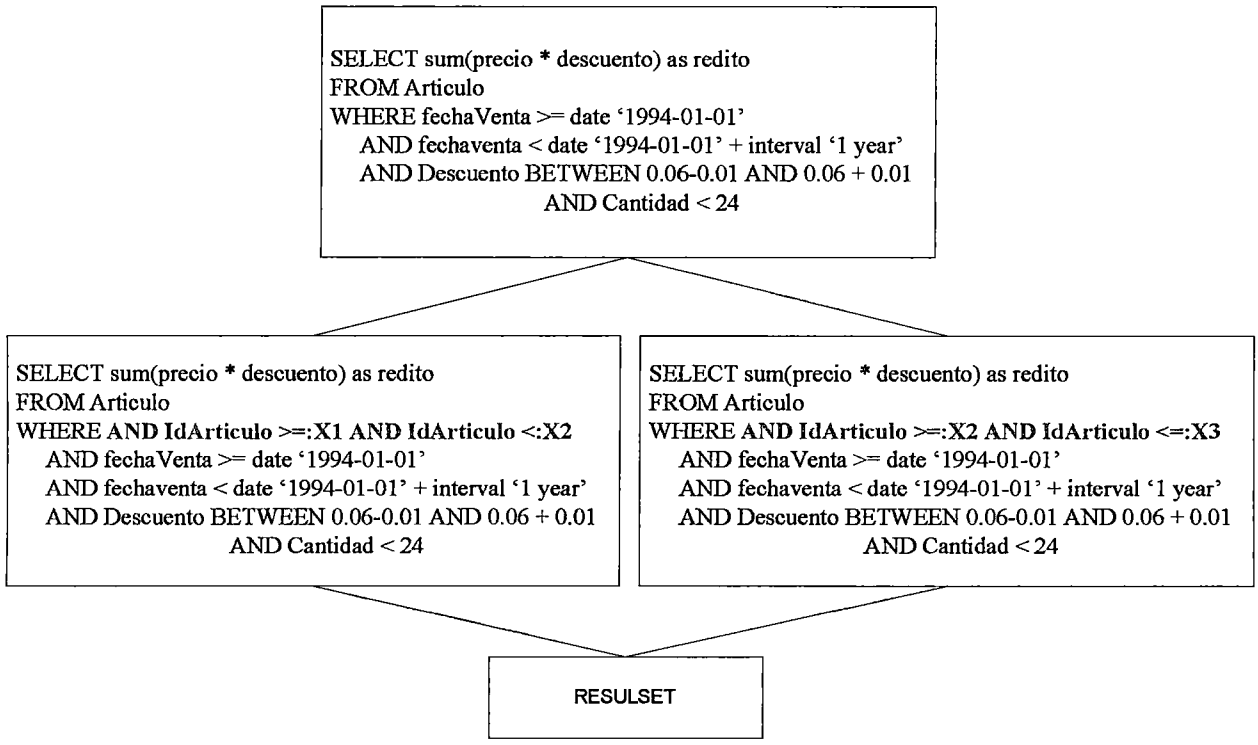


Figura 3-3: Algoritmo de Partición Virtual Simple (PVS)

Donde los intervalos : X1,:X2 y :X3 pertenecen al rango de IdArticulo, además se cumple que:

a) X1 < X2 < X3

b) [X1,X2> U [X2,X3] = rango de IdArticulo

Como tal, se puede crear tantos intervalos como nodos existen en el cluster.

3.4.2. Pseudocodigo

```
funcion ejecutarConsultaConPVS ( Consulta: Cadena ) : ResultSet

variables     nroNodos: entero
              tamañoParticion: entero
              i : entero
              consultaParametrizada: Cadena
              campoParticion : TCampo {Clase que contendra metadatos de un
                                       campo de una tabla}
              colectorResultados : TColectorResultados {Clase que sirve para
                                       adicionar Resultados}

inicio

1 {recuperar el numero de nodos disponibles}

    nroNodos ← ObtenerNroNodos()

2 {Obtener el campo de la consulta a ser particionado}

    campoParticion ← ObtenerCampoParticion (Consulta)

3 {Obtener el tamaño de la partición en base al nro de nodos y
  el campo a particionar}

    tamañoParticion ← ObtenerTamañoParticion( NroNodos,
                                campoParticion)

4 {Crear el objeto que servira para que cada nodo adicione sus
  resultados}

    colectorResultados ← Nuevo TColectorResultados.create ()

5 {ejecutar las subconsultas en los nodos}

    para i=1 hasta NroNodos hacer
```

```
6 {Parametrizar la consulta para el intervalo i en el campo
   campoParticion}

     consultaParametrizada ← parametrizarConsulta (i,
                  tamañoParticion, campoParticion)

7 {ejecutar la consulta parametrizada remotamente en el nodo
  "i" en un nuevo hilo, cada nodo adicionara su resultado en el
  colector cuando termine}

     ejecutarConsultaNodo (i, ConsultaParametrizada,
                  colectorResultados)

   fin para

8 {extraer el resultado acumulado del colector para ser devuelto}

   retornar ← colectorResultados.obtenerResultado ()

fin.
```

3.4.3. Problemas del PVS

La eficiencia de este algoritmo depende de algunos factores. El Campo elegido para realizar la partición debe estar indexado para evitar que los nodos realicen una búsqueda total en la tabla. PVS es una técnica de descomposición estática, que requiere que los límites de la partición sean determinados antes de la ejecución de la consulta. Una vez enviado, las sub-consultas no pueden ser dinámicamente cambiadas. Esta característica no permite reasignar las tareas a los nodos.

CAPITULO IV

DEMOSTRACION TEORICA DE LA HIPOTESIS

En este capitulo se realiza una demostración teórica de la hipótesis de esta tesis, tomando como punto de partida la teoría del "Algebra Relacional", el cual es el fundamento teórico del "Modelo Relacional" y la definición del algoritmo de subdivisión de consultas desarrollado en el capitulo anterior.

La demostración teórica de la hipótesis consta de dos partes. Primero se demuestra que una consulta procesada con el algoritmo de subdivisión de consultas devuelve el mismo resultado que si se procesara sin el algoritmo y partiendo de la validez de la primera parte se analiza teóricamente el tiempo de ejecución de la consulta.

4.1. CONSIDERACIONES PREVIAS

4.1.1. Modelo Relacional - Conceptos Generales

4.1.1.1. Datos Atómicos

Todos los valores de datos en el modelo relacional son atómicos, esto implica que cada posición de fila columna en cada tabla siempre tiene solo un dato y nunca un conjunto de valores.

4.1.1.2. Tuplas (t)

Una tupla de una relación o de una tabla corresponde a una fila de aquella tabla. Las tuplas están comúnmente desordenadas puesto que matemáticamente una relación se define como un conjunto y no como una lista.

4.1.1.3. Atributos (A)

Un atributo de una relación o tabla corresponde a una columna de la tabla. Los atributos están desordenados y se referencia por nombres y no por la posición que ocupan.

4.1.1.4. Dominios (Dom(A))

Un dominio se define como un conjunto de valores del mismo tipo. Todo atributo extrae sus valores de un dominio, de aquí que dos atributos pueden ser comparados solo si pertenecen al mismo dominio.

4.1.1.5. Esquema de una Relación

El esquema de una relación es el conjunto que identifica todos los Atributos de un objeto. Se representa como:

$$E = \{A_1, A_2, A_3, \ldots\ldots A_n\}$$

Donde Ai ,(i=1..n) corresponde a un atributo y "n" el numero de atributos de interés del objeto.

4.1.1.6. Relación (R)

Formalmente, una relación R es un conjunto de n-tuplas tal que una n-tupla cualquiera "x" es:

$$\{ (Ai , a) / Ai \in E, a \in Dom(Ai), \forall i, i = 1, .. n \}$$

Donde E es el esquema de la relación.

4.1.1.7. Grado de una Relación

Es el número de atributos en la relación.

4.1.1.8. Cardinalidad de una Relación (Card)

La cardinalidad en un determinado momento esta definida como el número de tuplas en la relación. Esta puede cambiar en cualquier momento.

4.1.1.9. Compatibilidad de dos Relaciones

Sean A y B dos relaciones con esquemas

$E_1 = \{A_{11},A_{12},.....A_{1n}\}$ y $E_2 = \{A_{21},A_{22},.....A_{2n}\}$ se dice que son compatibles si y solo

si:

1. $\forall P, P \in E_1 \exists P', P' \in E_2$ tal que $Dom(P) = Dom(P')$
2. $\forall P, P \in E_2 \exists P', P' \in E_1$ tal que $Dom(P) = Dom(P')$

O sea que ambas relaciones deben ser del mismo grado "n" y que el i-esimo atributo de cada relación se debe basar en el mismo dominio.

4.1.1.10. Unión de Relaciones

Sea A y B dos relaciones compatibles entonces se define:

AUB = C / C es una relación compatible con A y B, que contiene todas las tuplas de A y B

4.1.1.11. Igualdad de Relaciones

Sea A y B dos relaciones, entonces:

$A = B \Leftrightarrow$ A es compatible con B y **NO EXISTE** una tupla en A que no este en B y viceversa

4.1.1.12. Clave Primaria

Sea "A" una relación con esquema $E_1 = \{A_1, A_2, A_3, \ldots\ldots An\}$. Una llave de la relación "A" es un atributo o un conjunto de atributos $K = \{Ci, Cj, \ldots\ldots Cx\}$ que cumple

- **Unicidad**: No existen dos tuplas de "A" tales que para ellas el conjunto de atributos que componen "K" tienen los mismos valores.
- **Minimalidad**: Ninguno de los atributos que componen K puede ser eliminado sin afectar la unicidad.

4.1.1.13. Llaves Externas

Se define como un atributo (o atributos) en una relación cuyos valores se requieren para emparejar a los valores de la llave primaria de otra relación. La llave externa y su correspondiente llave primaria deben ser definidas en el mismo dominio.
Una llave externa no tiene que ser componente de la llave primaria que la contiene. El emparejamiento entre una llave externa y una llave primaria representa una referencia entre dos relaciones.

4.1.1.14. Reglas de Integridad

Existen básicamente dos reglas:

- **Regla de Integridad de Entidad**: Norma sobre la imposibilidad de que un atributo que componga la llave primaria de una relación base acepte valores nulos

- **Regla de Integridad Referencial**: Formula que si una relación R2 incluye una llave externa FK que empareja con una llave primaria PK de alguna relación R1, entonces cada valor de FK en R2 debe:
  a) Ser igual al valor de PK en alguna tupla de R1
     ó
  b) Ser totalmente Nula, esto es, cada atributo en FK debe ser nulo

4.1.1.15. Estructura Básica de una Consulta SQL

Una consulta básica SQL consta de 3 cláusulas: select, from, where.

select: se usa para listar los atributos que se desean ver en el resultado de una consulta. Corresponde a la operación de proyección del algebra relacional.
from: lista las relaciones que van a intervenir en la consulta. Corresponde a la operación de producto cartesiano del algebra relacional.
where: consta de 0 ó mas predicados (Pi)que implica los atributos de las relaciones que aparecen en la cláusula from, separados por los operadores OR | AND. Corresponde al predicado de selección del algebra relacional.
Se muestra a continuación. Una consulta "x" :

select $A_1, A_2, \ldots A_n$
from $R_1, R_2, \ldots R_n$
where P_1 OR|AND P_2 OR|AND ..... P_n

Donde un predicado Pi ,con i=1..n, es igual a (val op val) donde

val = constante | Atributo
op = <> | < | > | <= | >= | IS NULL | IS NOT NULL | LIKE | BETWEEN | IN

De lo anterior podemos representar una consulta básica como la terna:

$$x = \{ (A_1, A_2, \ldots A_n), (R_1, R_2, \ldots R_n), (P_1 \text{ <OR | AND> } P_2 \ldots \text{<OR | AND> } P_n) \} \quad \ldots (*)$$

4.2. Definición de la Función de Ejecución de Consultas (h)

Sea "x" una consulta de la forma definida en (4.1.1.15.*), entonces existe una función "h" tal es que:

$$\mathbf{h(x)} = R(A_1, A_2, \ldots A_n)$$

donde toda tupla de la relación $R \in (R_1 \times R_2 \times \ldots \times R_n)$ y

$(P_1 \text{ <OR | AND> } P_2 \ldots \text{ <OR | AND> } P_n) = \text{TRUE}$ para dicha tupla

4.3. Definición Formal del Algoritmo de División de Consultas - PVS (f)

Sea "x" una consulta de la forma definida en (4.1.1.15.*), "n" el número de nodos del cluster, y "f" una función de división de consultas que utiliza el algoritmo PVS para dividir, entonces se tiene que:

$$\mathbf{f(x, n)} = (x_1, x_2, x_3, x_4, \ldots, x_n), \text{ donde } x_i,\ i=1..n \text{ es una subconsulta de } x$$

Se sabe que "f" divide la consulta agregando un predicado P' que divide el campo clave de la relación (Tabla) con mayor cantidad de tuplas.
Y sea R' la relación con mayor cantidad de tuplas y un atributo A' ϵ R' el cual se va a dividir, entonces:

P' es igual a (A' >= Linf)AND (A' < Lsup)

Considerando que:

1. A' no necesariamente pertenece al conjunto de atributos de "x".
2. A' debe ser un campo clave o un campo único (no debe tener valores repetidos).
3. Linf y Lsup son el límite inferior y el límite superior respectivamente, y variaran para cada "xi".
4. Linf y Lsup para cada intervalo son calculados de acuerdo al numero de tuplas de R' y de acuerdo al numero de nodos del cluster (n). de acuerdo a la formula siguiente.
   i. dif = (Cardinalidad(R') div n)
   ii. Linf = 1 // para el primer intervalo
   iii. Lsup = Linf + dif //para el primer intervalo
5. Para determinar los siguientes intervalos excepto el ultimo usara
   i. Linf = Lsup //Lsup del anterior es Linf del nuevo intervalo
   ii. Lsup = Linf + dif //Lsup del nuevo intervalo
6. En el ultimo intervalo se cumple que
   i. Linf = Lsup
   ii. Lsup = (Cardinalidad(R')) + 1
7. Los intervalos son cerrados en la izquierda y abiertos en la derecha.

Entonces de lo anteriormente expuesto se tiene que:

Si x = { $(A_1, A_2, \ldots A_k)$,($R_1, R_2, \ldots R_m$),(P_1 <OR | AND> $P_2 \ldots$ <OR | AND> P_j) }

Cada subconsulta de X seria:

$$x_i = \{ (A_1, A_2, \ldots A_k),(R_1, R_2, \ldots R_m),(P' \textbf{ AND } P_1 <OR \mid AND> P_2 \ldots <OR \mid AND> P_j)\}$$

Donde para todo x_i, i=1..n tal es que x_i es una subconsulta de x en "n" nodos.

4.4. Definición Formal del Algoritmo de Ejecución de Consultas Subdivididas - PVS) (g)

Como consecuencia de lo anterior, si se aplica la función "h" a cada subconsulta entonces se puede afirmar que:

$h(x_i)$ es compatible con h(x) ; para todo $h(x_i)$, i=1..n es una subconsulta de X en "n" nodos.

Entonces se define la función "g" que utiliza la división vertical para procesar un conjunto de "n subconsultas" en "n" nodos, y obtener un solo resultado.

$g (y , n) = h(x_1) \cup h(x_2) \cup \ldots\ldots \cup h(x_n) = z$ / z es una relación compatible con h(x)

donde: $y = f (x , n) = (x_1, x_2, x_3, \ldots\ldots, x_n)$, donde x_i, i=1..n es una subconsulta de x

Lo anterior es cierto por la definición de la Unión de Relaciones (cap 4.1.1.10)

Las "n" subconsultas al ser procesadas paralelamente retornan cada uno un resultado parcial que son compatibles entre si. Los cuales pueden ser reunidos en un solo resultado sin inconsistencias. Solo quedaría

demostrar que el resultado de la ejecución del algoritmo es igual al resultado de ejecutar la consulta sin aplicar el algoritmo.

DEMOSTRACION: EL RESULTADO DE LA EJECUCION DE UNA CONSULTA DIVIDIDA (g(y,n)) ES IGUAL AL RESULTADO DE LA EJECUCION DE LA MISMA CONSULTA NO DIVIDA (h(x))

Ahora se prueba que:

$$h(x) = g(y, n) \quad \text{tal es que} \quad y = f(x,n) = (x_1, x_2, x_3, x_4, ..., x_n)$$

Se sabe por la definición de igualdad de relaciones (cap 4.1.1.11) que:

h(x) = g(y , n) **SI Y SOLO SI** h(x) es compatible con g(y , n)

Y

NO EXISTE una tupla en h(x) que no este en g(y , n) y Viceversa

Se conoce que h(x) es compatible con g(y, n) por los resultados obtenidos en 4.3.

Por lo tanto solo se necesita probar la segunda parte

NO EXISTE una tupla en h(x) que no este en g(y , n) y viceversa

Demostrar que NO EXISTE una tupla en h(x) que no este en g(y , n)

Se va a demostrar por contradicción, es decir demostrar que EXISTE una tupla en h(x) que no este en g(y , n).

Se sabe que:

$$x = \{ (A_1, A_2, ...A_j),(R_1, R_2, ... R_m),(P_1 <OR \mid AND> P_2 ...<OR \mid AND> P_k) \}$$

Además

$h(x) = R(A_1, A_2, ...A_j)$

donde toda tupla de la relación $R \in (R_1 \times R_2 \times ... \times R_m)$ y

$(P_1 <OR \mid AND> P_2 .... <OR \mid AND> P_k) = TRUE$ para dicha tupla

Sea "t" la tupla que pertenece a h(x) y no pertenece a g(y,n), entonces por lo anterior:

$t \in (R_1 \times R_2 \times ... \times R_m)$ y $(P_1 <OR \mid AND> P_2 .... <OR \mid AND> P_k) = TRUE$ para "t" **.....(i)**

Por otra parte $f(x,n) = (x_1, x_2, x_3, x_4, ..., x_n)$ donde se sabe que

$xi = \{ (A_1, A_2, ...A_j), (R_1, R_2, ... R_m), (P' \text{ AND } P_1 <OR \mid AND> P_2 ... <OR \mid AND> P_k)\}$

Además por definición:

$g(y, n) = h(x_1) \cup h(x_2) \cup ..... \cup h(x_n)$ donde por definición se tiene que:

para todo $h(xi) = Ri(A_1, A_2, ...A_j)$
donde toda tupla de la relación $Ri \in (R_1 \times R_2 \times ... \times R_m)$ y
$(P' \text{ AND } P_1 <OR \mid AND> P_2 ... <OR \mid AND> P_k) = TRUE$ para dicha tupla

Ahora la tupla "t" no pertenece a f (y , n) por lo tanto $(P' \text{ AND } P_1 <OR \mid AND> P_2 ... <OR \mid AND> P_m) = FALSE$ para dicha tupla, pero por (i) sabemos que $(P_1 <OR \mid AND> P_2 .... <OR \mid AND> P_m) = TRUE$ para "t" por lo tanto P' debe ser falso, pero P' se obtiene dividiendo la relación con mayor cardinalidad de "x" (cap 4.3), luego para que P' sea falso la tupla "t" debe tener un valor "a" en el campo A' / (a < primer elemento de A') o (a > ultimo elemento de A'), lo cual es absurdo por lo tanto se rechaza el que EXISTE una tupla en h(x) que no este en g (y , n) .

Por consiguiente: NO EXISTE una tupla en h(x) que no este en g (y , n)

Demostrar que NO EXISTE una tupla en g (y , n) que no este en h(x)

De igual manera se demostrara por contradicción, es decir demostrar que si EXISTE una tupla en g (y, n) que no este en h(x).
Se sabe que:

$g(y, n) = h(x_1) \cup h(x_2) \cup \ldots\ldots \cup h(x_n)$ donde se sabe
que $h(xi) = Ri(A_1, A_2, \ldots A_j)$ para todo $i = 1 .. n$
donde toda tupla de la relación $Ri \in (R_1 \times R_2 \times \ldots \times R_m)$ y
(P' AND P_1 <OR | AND> P_2 ...<OR | AND> P_k) = TRUE para dicha tupla.

Sea "t" la tupla que pertenece a g(y,n) y no pertenece a h(x), entonces para dicha tupla se cumple que:

(P' AND P_1 <OR | AND> P_2 ...<OR | AND> P_m) = TRUE. **.....(ii)**

Por otro lado h(x) esta definido como:

$h(x) = R(A_1, A_2, \ldots A_j)$

donde toda tupla de la relación $Ri \in (R_1 \times R_2 \times \ldots \times R_m)$ y
(P_1 <OR | AND> P_2 ...<OR | AND> P_k) = TRUE para dicha tupla.

Como "t" no pertenece a h(x) entonces (P_1 <OR | AND> P_2 ...<OR | AND> P_k) debe ser FALSE para dicha tupla, lo cual es una contradicción a (ii), ya que por definición del AND lógico (P_1 <OR | AND> P_2 ...<OR | AND> P_k) = TRUE.

Por lo tanto se rechaza que EXISTE una tupla en g (y , n) que no este en h(x).

Por consiguiente NO EXISTE una tupla en g (y , n) que no este en h(x)

POR CONSIGUIENTE

Se logra demostrar que el resultado de la ejecución de una consulta dividida es igual al resultado de la ejecución de la misma consulta no dividida

4.5. Demostración Teórica de la hipótesis H

H: El tiempo de respuesta de una consulta con complejidad alta usando división vertical
es menor que el tiempo de respuesta de la misma consulta usando división horizontal

Sea:

x: una consulta

t1: tiempo de ejecución de "x" en "n" nodos, $(n > 1)$

t2: tiempo de ejecución de "x" en 1 nodo

Entonces se debe probar que:

$t1 < t2$

Teóricamente se debe probar que

$$T(h(x)) > T(g(y, n)) \;/\; y = f(x, n) \wedge n > 1$$

Donde T es la función que determina el tiempo de ejecución de una función.

DEMOSTRACION

Para probar que t1 < t2 se debe realizar una análisis del Tiempo de Ejecución de las diferentes funciones que intervienen en la ejecución de las consultas, particularmente de las funciones f, h, g.

Sea:

x: una consulta de la forma definida en 4.1.1.15.*

R': Relación de "x" a particionar virtualmente.

n: número de nodos del cluster.

f: función de división de consultas que utiliza el algoritmo PVS.
h: función de ejecución de consultas.
g: función de procesamiento de subconsultas en "n" nodos

Análisis del Tiempo de Ejecución de la función de subdivisión de consultas f(x,n) :

Sabemos que "f" utiliza el algoritmo PVS para dividir la consulta, por lo tanto de acuerdo al algoritmo se tiene:

T(f (x , n)) = T(Costo de Dividir la Cardinalidad de R' entre "n")+T(Calcular los Intervalos)

Como dividir es una operación simple y los intervalos se calculan en un bucle de "n" iteraciones, tenemos que:

$$\boxed{T(f(x,n)) = 1 + n} \qquad (i)$$

Análisis del Tiempo de Ejecución de la función de ejecución de consultas h(x)

Por ser "x" una consulta se sabe por definición que:
$x = \{ (A_1, A_2, \ldots A_k), (R_1, R_2, \ldots R_m), (P_1 <OR \mid AND> P_2 \ldots <OR \mid AND> P_j) \}$
y por definición de h(x)

$h(x) = R(A_1, A_2, \ldots A_k)$ donde toda tupla de la relación $R \in (R_1 \times R_2 \times \ldots \times R_m)$ y
$(P_1 <OR \mid AND> P_2 \ldots <OR \mid AND> P_j) = TRUE$ para dicha tupla

Por lo tanto h(x) efectúa un producto cartesiano y luego verifica que cada una de las tuplas cumpla con las condiciones dadas, es decir:

$$T(h(x)) = \begin{cases} Card(R) \times nP & \text{si numero de Relaciones de "x" es 1} \\ Card(R_1 \times R_2 \times \ldots \times R_m) \times nP & \text{si el numero de Relaciones de "x" es mayor a 1} \end{cases}$$

donde nP es el numero de Predicados de "x"

Como la cardinalidad de las relaciones es un número natural y el número de predicados es un número natural, entonces el resultado del producto será también un número natural, este resultado será el número total de comparaciones que realiza la consulta "x", y se representara como:

$$T(h(x)) = r \quad \forall r \in N \qquad \ldots\ldots\ldots\ldots \textbf{(ii)}$$

Análisis de Tiempo de Ejecución de g (y, n)

Si se sabe que $(x_1, x_2 \ldots, x_n)$ son subconsultas de "x" en "n" nodos, y se aplica la función "h" a cada subconsulta; por la definición de h(x) cada subconsulta recorrerá solo una partición de la relación R' que se encuentra tanto en "x" como en las subconsultas de "x", a esta partición virtual la denominaremos R" y luego se tiene:

$$T(h(x)) = r = \begin{cases} Card(R') \times nP & \text{si numero de Relaciones de x es 1} \\ Card(R_1 \times R_2 \times \ldots \mathbf{R'} \ldots\ldots \times R_m) \times nP & \text{si el numero de Relaciones de x es mayor a 1} \end{cases}$$

y

$$T(h(x_i)) = r' = \begin{cases} Card(R'') \times nP & \text{si numero de Relaciones de x es 1} \\ Card(R_1 \times R_2 \times \ldots R'' \ldots \times R_m) \times nP & \text{si el numero de Relaciones de x es mayor a 1.} \end{cases}$$

Para todo $i \in [1,n]$ tal es que "x_i" es una subconsulta de "x" en "n" nodos.

Luego como R'' es una partición de R' y por la definición del algoritmo PVS Podemos afirmar que:

$$Card(R'') = \frac{Card(R')}{n} \quad \forall n > 1$$

De lo anterior podemos deducir lo siguiente:

$Card(R_1 \times R_2 \times \ldots R'' \ldots \times R_m) = Card(R_1) \times Card(R_2) \times \ldots Card(R'') \ldots \times Card(R_m)$ esto por la definición de producto cartesiano y cardinalidad de relaciones.

$$Card(R_1 \times R_2 \times \ldots R'' \ldots \times R_m) = Card(R_1) \times Card(R_2) \times \ldots \frac{Card(R')}{n} \ldots \times Card(R_m)$$

esto reemplazando: Card(R'') = Card(R') / n

$$Card(R_1 \times R_2 \times \ldots R'' \ldots \times R_m) = \frac{Card(R1) \times Card(R2) \times \ldots Card(R') \ldots \times Card(Rm)}{n},$$ esto debido a que: $a \times \frac{b}{2} = \frac{a \times b}{2}$; $\forall a,b \in N$

$Card(R_1 \text{ X } R_2 \text{ X } \ldots \text{ R''}\ldots \text{ X } R_m) = \dfrac{Card(R1\,X\,R2\,X\ldots X\ R'\,X\ldots X\ Rm)}{n}$, por la definición de producto cartesiano y cardinalidad de relaciones.

Luego por las definiciones de T(h(x)) y T($h(x_i)$)

se puede afirmar que: $r' = \dfrac{r}{n} \quad \forall n > 1$

Entonces se concluye que:

$$\boxed{T(h(x_i)) = \frac{r}{n} \quad , \forall n > 1} \quad \ldots\ldots\ldots \textbf{(iii)}$$

Por otra parte se sabe que:

$g(y, n) = g(f(x, n)) = h(x_1) \cup h(x_2) \cup \ldots \cup h(x_n)$

de lo cual se afirma que:

$T(g(y, n)) = T(f(x, n)) + T(h(x_1) \cup h(x_2) \cup \ldots \cup h(x_n))$,

Pero como $h(x_1)$, $h(x_2)$.... $h(x_n)$ son procesadas paralelamente, se puede afirmar que:

$T(h(x_1) \cup h(x_2) \cup \ldots \cup h(x_n)) = T(h(x_i)) + T(u)$

donde T(u) es el tiempo que demora en unir los resultados parciales

Finalmente :

$$T(g(y, n)) = T(f(x, n)) + T(h(x_i)) + T(u)$$

Luego de (i) y (iii)

$$T(g(y, n)) = (1+n)+\frac{r}{n}+u \qquad \textbf{(iv)}$$

Demostración que T(h(x)) > T(g (y, n))

Se basará en los resultados obtenidos hasta el momento, se va a demostrar que:

T(h(x)) > T(g (y, n)) .....(A)

Pero se debe considerar que el número de nodos (n) debe ser mayor o igual a 2 ya que esta condición es necesaria para dividir la consulta (x) en "n" subconsultas.

De lo anterior y de las definiciones (ii) y (iv) y reemplazando ambas en (A) se tiene que demostrar lo siguiente:

$$r > (1+n)+\frac{r}{n}+u \quad , \quad n >= 2$$

despejando r

$$r-\frac{r}{n} > 1+n+u \Longrightarrow r(1-\frac{1}{n}) > 1+n+u \Longrightarrow r > \frac{1+n+u}{1-\frac{1}{n}}$$

$$r > \frac{1+n+u}{\frac{n-1}{n}} \Longrightarrow$$

$$r > \frac{n\times(1+n+u)}{n-1} \quad , n >= 2 \qquad \ldots\ldots\ldots\ldots\ldots\ldots (I)$$

Se observa que se tiene 3 variables:

r : numero de comparaciones que realiza la consulta "x"

n : numero de nodos del cluster

u : cantidad de registros que devuelve la consulta

Ahora se tiene que ver para que valores se cumple la Inecuación (I), para lo cual se grafica la inecuación teniendo en cuenta la siguiente consideración:

"u" es un valor entero que de antemano no se conoce su valor , pero por la definición de la función de ejecución de consultas, se puede afirmar que:

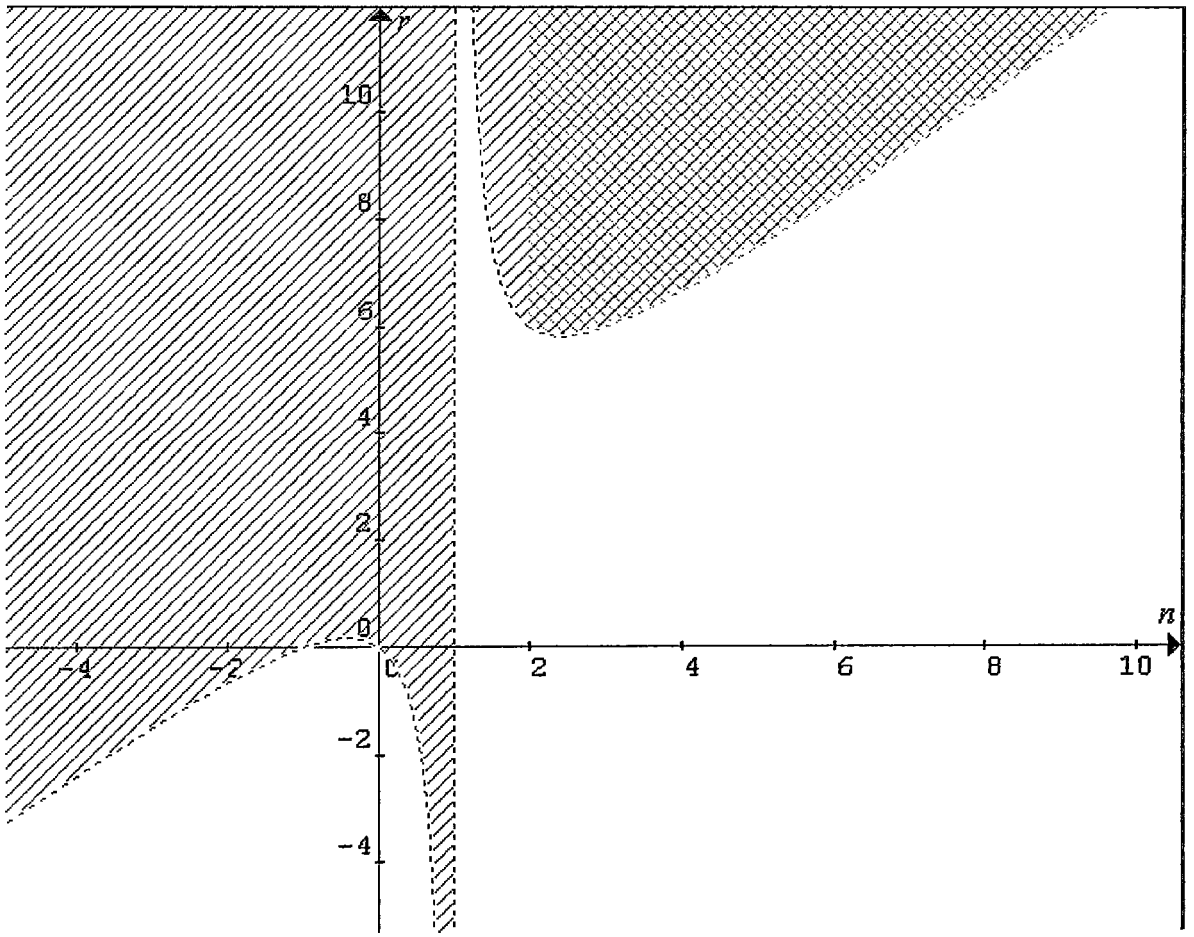
$$\boxed{0 \le u \le r \ , \ \forall u,r \in \mathrm{N}} \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{(A)}$$

Por lo que se graficará la ecuación reemplazando el valor de u empezando por su mínimo valor para poder ver el comportamiento de la curva y analizar los resultados.

Si "u=0" (La consulta no devuelve resultados) entonces la Inecuación (I) se convertiría en:

$$r > \frac{n \times (1+n)}{n-1}$$

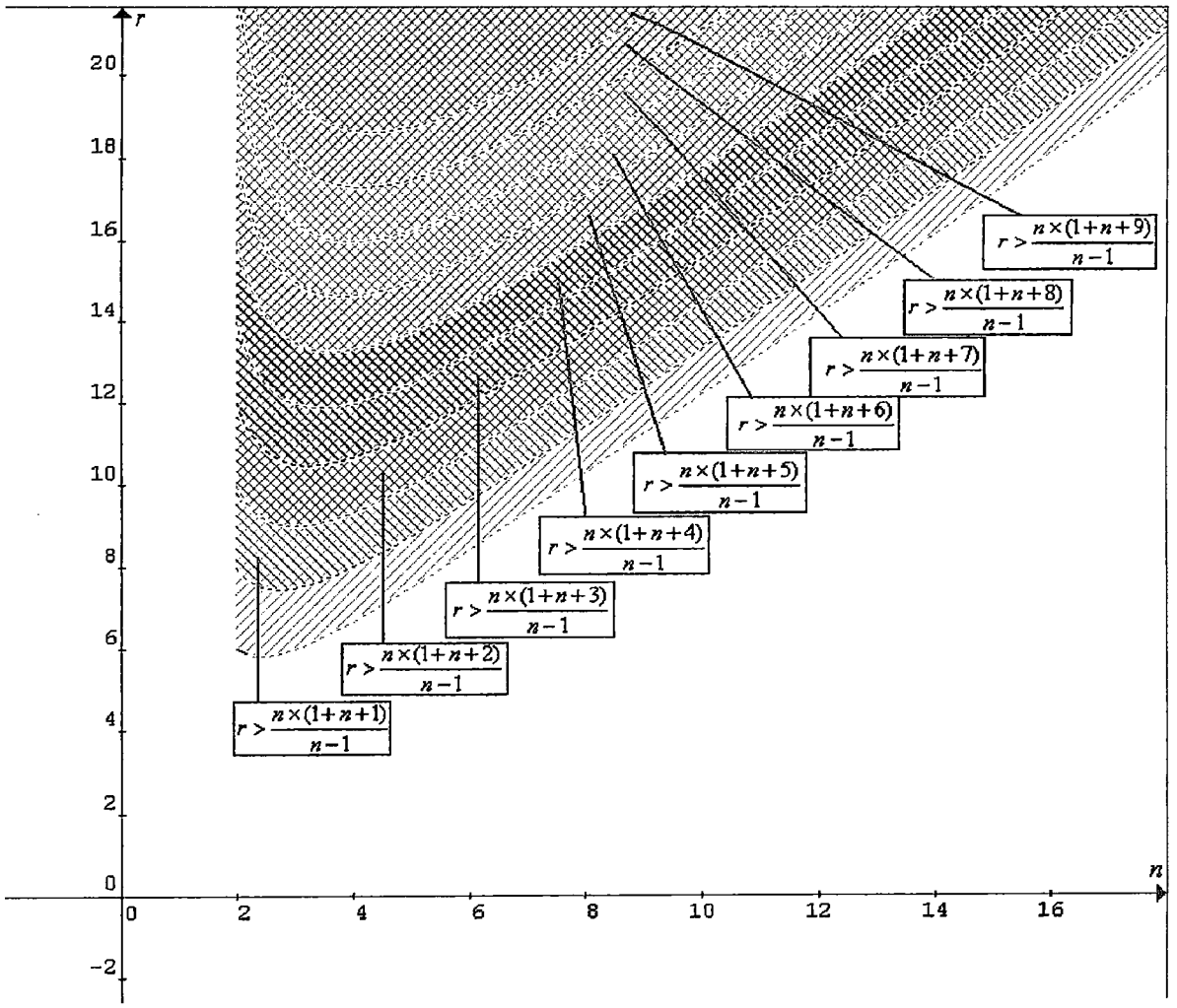
La grafica se presenta a continuación:



La parte doblemente achurada corresponde a la solución de la inecuación ya que n>=2.

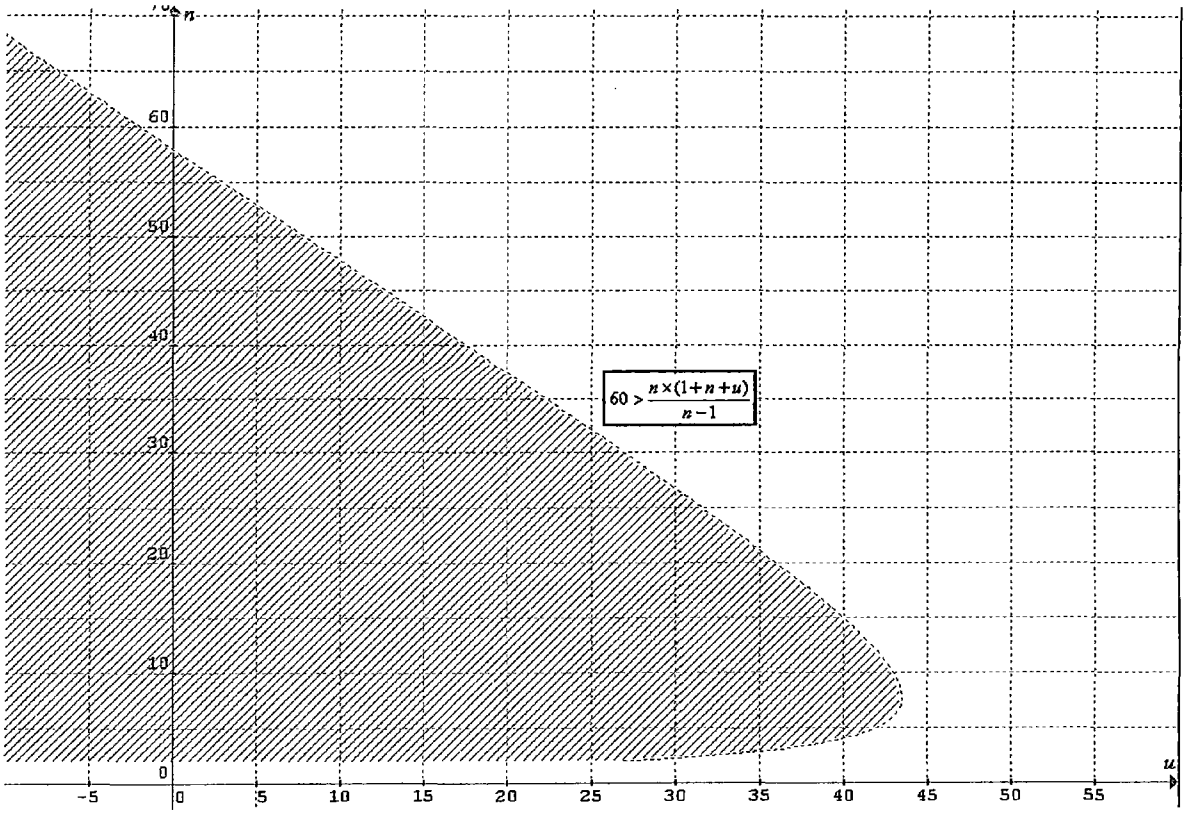
Esta grafica indica que cuando la consulta no devuelve resultados existen valores para los cuales la inecuación no se cumple, por ejemplo: teniendo u=0, y n = 4, r =4.

Ahora se analizará gráficamente el comportamiento de la curva cuando se incrementa la variable "u":



Analizando el comportamiento se ve que al incrementarse el número de registros que devuelve la consulta, crece el número de registros **mínimo** que debería comparar la consulta, para que el procesarla en cluster sea más óptimo que procesarla en un solo nodo.

Pero la grafica no permite conocer si existe un límite superior respecto al número de nodos, para analizar esto se va a usar la misma Inecuación (I) pero graficando "n" en función de "u", para lo cual s va a reemplazar la variable "r" por un valor arbitrario 60,



La grafica de esta ecuación permite afirmar lo siguiente:

1.- El lado superior de la grafica que cruza con el eje "n" indica el numero **máximo** de nodos que se podría usar para procesar la consulta y seguir siendo mas rápida, según el numero de registros (eje "u") que devuelve la consulta.

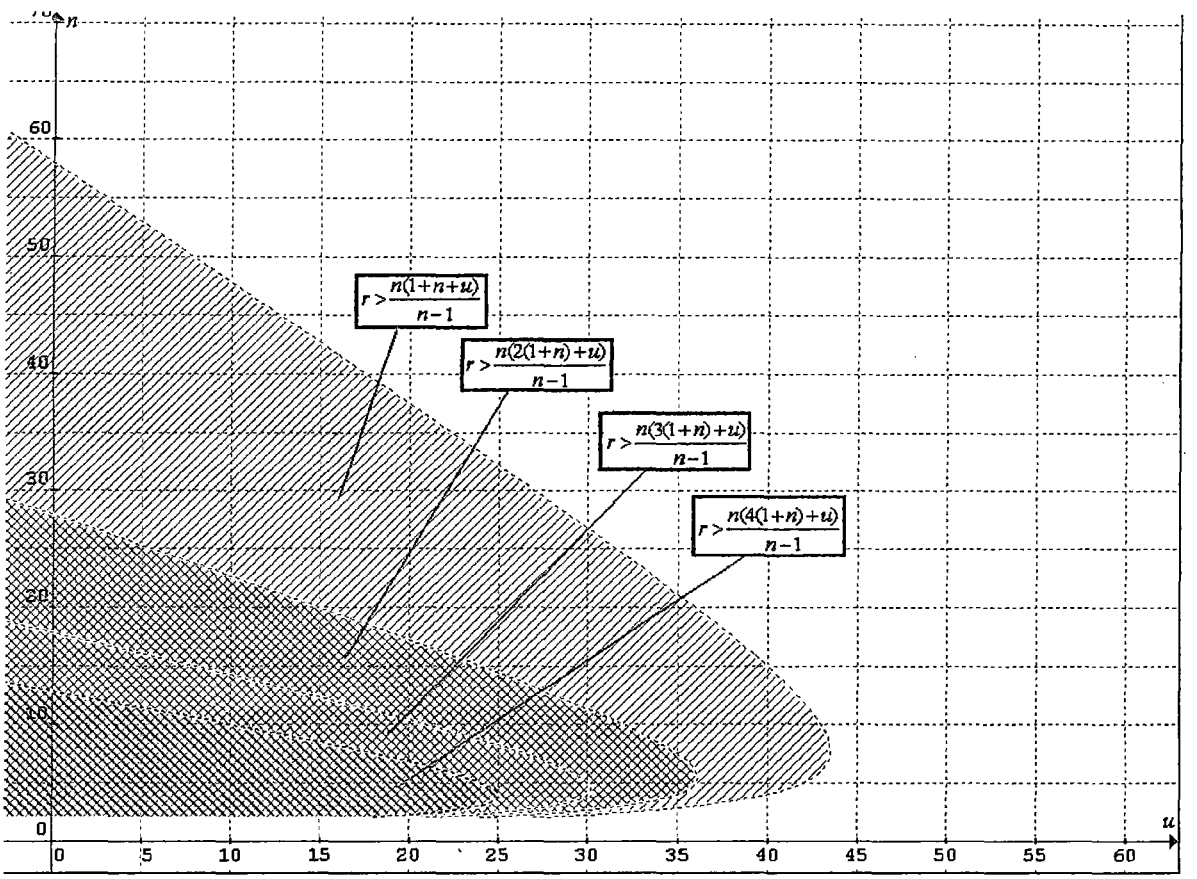
2.- El lado inferior de la grafica indica el número **mínimo** de nodos que se debe usar para procesar la consulta y seguir siendo más rápida.

3.- El lado derecho de la grafica marca el máximo número de registros que puede devolver la consulta y seguir siendo más rápida.

4.- Se puede apreciar que según la cantidad de registros que procesa la consulta (para este caso usamos 60) y el numero de registros que devuelve

la consulta ("u"), existe un mínimo y un máximo de nodos que se pueden usar para procesar la consulta y seguir siendo mas rápida que procesar la consulta en un solo nodo, pero además existe un limite en el numero de registros que puede devolver la consulta.

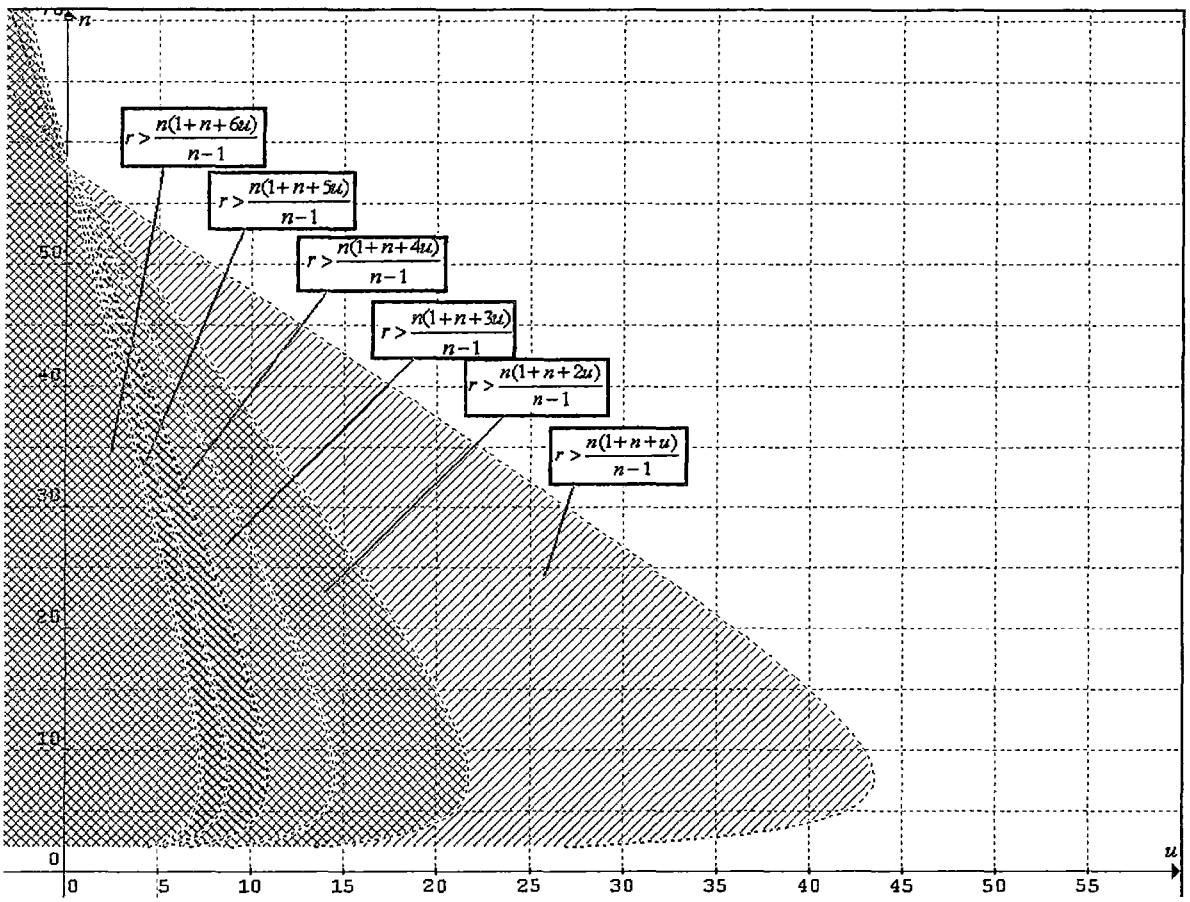
5.- El limite superior de la Curva es controlada por el numero de subdivisiones que se realiza a la consulta, y cuanto mas demore afecta directamente al numero máximo de nodos que se pueden utilizar, esto se puede deducir de la simple observación de la siguiente grafica, donde se ha multiplicado por 2, 3, 4 el tiempo que demora en subdividir las consultas.



En la práctica no hay razón para que el tiempo de división demore más de lo estimado ya que la subdivisión es un proceso sencillo que se realiza en un bucle, de 1 hasta el número de nodos.

6.- El limite derecho de la curva es controlado por el numero de registros que devuelve la consulta, cuando mas grande, mas demora en unir los resultados. Esto se deduce de la simple observación de la siguiente grafica, donde se aprecia como cambia la curva cuando "u" se multiplica por 2, 3, 4, 5, 6.

r = 60.



Es decir cuando mas se demore en reunir los resultados disminuye la cantidad de registros que puede devolver la consulta y permanecer en el intervalo de validez.

Aunque el tiempo de reunir los resultados se puede ver afectado por cosas como, el trafico de red, o la velocidad del procesador, los limites hallados son en realidad lo suficientemente holgados, ya que se esta graficando para un r = 60, solo para poder apreciar la forma de la curva. Si se considera que la base de datos con la que se va a probar tiene relaciones que generan un intervalo para "r" de $6x10^6 < r < 6x10^9$, y que
el numero de nodos es 10 y que por la naturaleza de las consultas que se usa devuelve pocas filas, entonces siempre se estará dentro del intervalo de validez.

CONCLUSION:

Como se puede apreciar de lo anteriormente visto, se acepta la hipótesis dado que se pudo demostrar que el tiempo de procesamiento de consultas complejas con división vertical es menor al tiempo que procesarlo con división horizontal, ya que a mayor número de nodos, menor será el tamaño de las particiones, por lo tanto menor el tiempo de ejecución de las subconsultas.

GRAFICA DEL TIEMPO DE RESPUESTA DE LA FUNCION

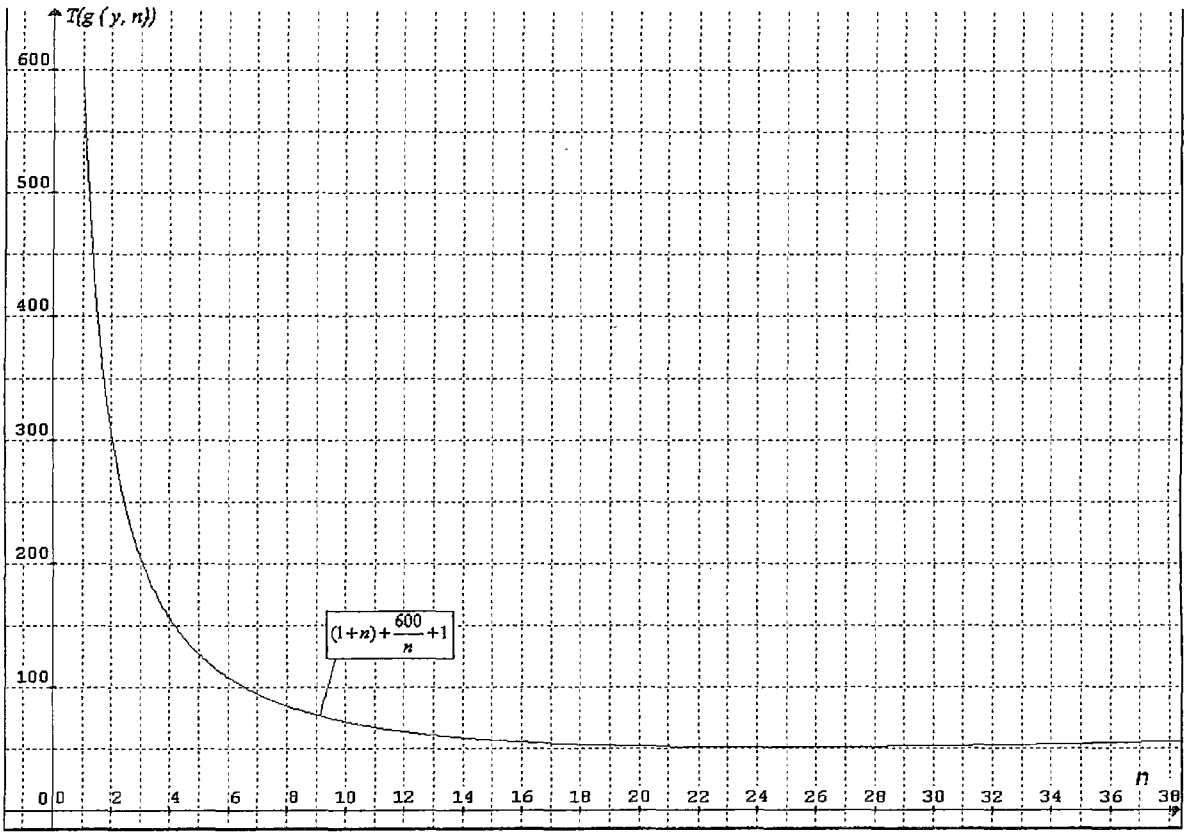
Para consolidar la conclusión anterior vamos a mostrar la grafica del tiempo de respuesta de la función de ejecución de consultas sub-divididas
Se sabe por los resultados anteriores que:

$$T(g(y, n)) = (1+n) + \frac{r}{n} + u$$

Luego se podrá graficar T(g (y, n)) en función de n (numero de nodos), asumiendo que r=600 y u = 1 solo para propósitos ilustrativos.
Luego se tiene:

$$T(g(y,n)) = (1+n)+\frac{600}{n}+1$$

Grafica de la curva



Como se observa a mas nodos menor será el tiempo de Ejecución de la función, pero como se podrá observar existe un punto en que el numero de nodos ya no mejora el tiempo, es mas se vuelve contraproducente, esto es debido a los limites que anteriormente se analizo, que son generados por el costo de dividir la consulta y unir los resultados.

CAPITULO V

ARQUITECTURA DEL SOFTWARE

Este capitulo presenta el "Documento de Arquitectura de Software", que viene a ser el artefacto producto de la fase de Inicio de la Metodología RUP; que servirá como base para el posterior desarrollo del prototipo, siguiendo las fases señaladas por esta metodología.

Este artefacto proporciona una descripción de la arquitectura del sistema, haciendo uso de diversas visiones arquitectónicas para representar diversos aspectos del sistema. Se basa en el modelo de Vistas 4+1 de Kruchten.

5.1. VISTA DE CASOS DE USO

5.1.1. Descripción del Software

Se requiere desarrollar un Software que se sitúe entre un cluster de base de datos y las aplicaciones clientes que hacen uso de esos datos, Cada servidor contara con un SGBD y una base de datos replicada, el Software se situara en un servidor que será llamado "principal" que contendrá la base de datos original. A los servidores los llamaremos "nodos" y al conjunto lo denominaremos "cluster".

El sistema debe agregar o eliminar nodos a petición del administrador del sistema. Si se agregan nodos estos deben contar con una replica de la base de datos principal antes de comenzar a utilizarlos.

Las consultas de los usuarios serán recibidas siempre por el Middleware el cual deberá identificar si se trata de una petición (SELECT) o una actualización (INSERT, UPDATE, DELETE) para proceder según corresponda, en el caso de una petición el Middleware dividirá la consulta en varias sub-consultas (División Vertical de las Consultas) para poder procesarla en varios nodos a la vez. Los resultados una vez devueltos por los nodos serán enviados al cliente.

Respecto a las consultas de actualización están serán enviadas siempre a todos los nodos para mantener sincronizadas las replicas con el servidor principal.

Los Usuarios no pueden conectarse directamente a ningún nodo. Solo podrán acceder a la Base de Datos a través del Software.
El sistema deberá en lo posible de liberar de carga de trabajo al nodo central ya que este se encarga de la gestión de todos los nodos, además de interactuar con los clientes.

5.1.1.1. Procesos del Negocio

Los siguientes procesos son de la arquitectura del Software:

- Procesar Actualización (P1)
- Procesar petición (P2)
- Agregar Nodo (P3)
- Quitar Nodo (P4)

5.1.1.1.1. Procesar Actualización (P1)

Este proceso se encarga del manejo de las consultas de tipo Actualización (INSERT, DELETE, UPDATE). El algoritmo de este proceso es el siguiente:

1. El Software recibe la consulta (INSERT, DELETE, UPDATE) del cliente..
2. El Software extrae los metadatos de una consulta.
3. El Software propaga la actualización entre todos los nodos del cluster.
4. Cada nodo del cluster ejecuta la actualización.
5. El Software retorna mensaje de éxito al cliente.

5.1.1.1.2. Procesar Petición (P2)

Este proceso se encarga del manejo de las consultas de tipo Petición (SELECT). El algoritmo de este proceso es el siguiente:

1. El Software recibe la consulta (SELECT) del cliente
2. El Software extrae los metadatos de una consulta.
3. El Software subdivide la consulta
4. El Software asigna las sub-consultas a los nodos para su procesamiento paralelo.
5. El cluster procesa las subconsultas.
6. El Software reúne los resultados parciales devueltos por los nodos
7. El Software retorna el resultado al cliente

5.1.1.1.3. Agregar Nodo (P3)

Este proceso se encarga de la solicitud para agregar un nuevo nodo al cluster. El algoritmo de este proceso es el siguiente:

1. El administrador solicita al Software la agregación de un nuevo nodo y le proporciona los datos necesarios.
2. El Software verifica si se puede conectar al nuevo nodo.
3. Si la conexión es fallida. El Software informa al administrador sobre el error.
4. Si la conexión es satisfactoria, el Software agrega el nodo a su lista de nodos.
5. El Software informa al administrador sobre el éxito.

5.1.1.1.4. Quitar Nodo (P4)

Este proceso se encarga de la solicitud para quitar un nodo del cluster. El algoritmo de este proceso es el siguiente:

1. El administrador solicita al Software la eliminación de un nodo del cluster y le proporciona los datos necesarios.
2. El Software retira este nodo de su lista de nodos.
3. El Software verifica si este nodo esta ocupado.
4. Si el nodo esta ocupado espera hasta que finalice su trabajo.
5. El Software libera el nodo.
6. El Software informa al administrador del éxito de la operación.

5.1.1.2. Modelo de Dominio

En este modelo se incluye lo más significativo del vocabulario del dominio.

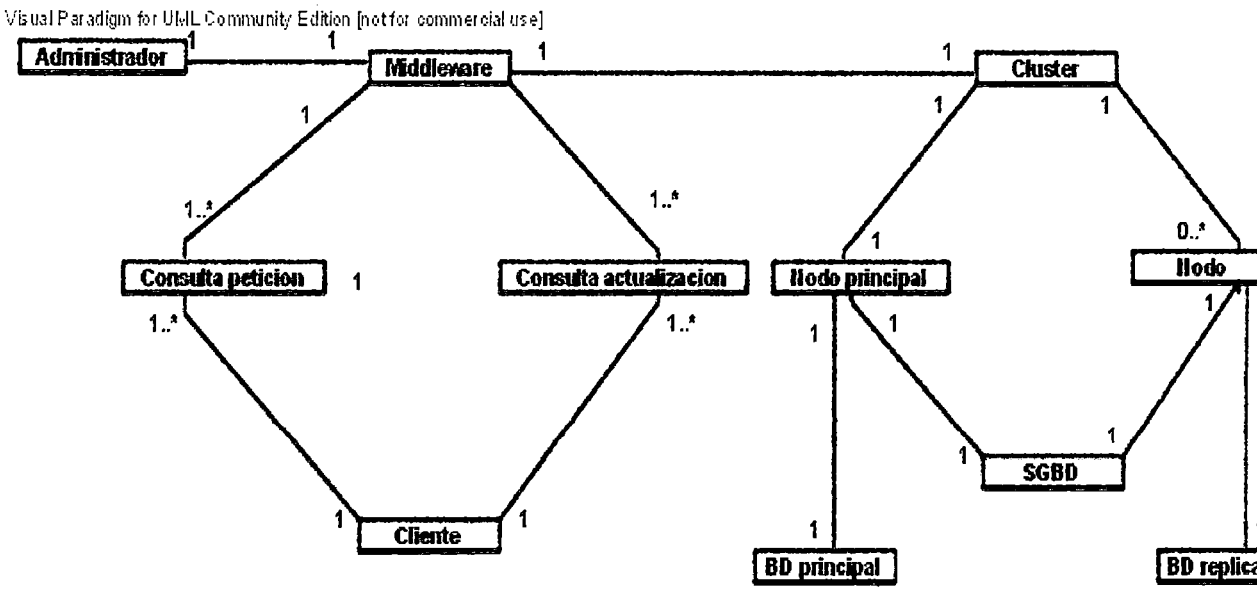
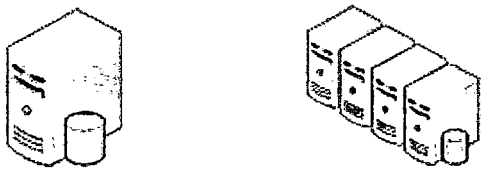


Figura 5-1: Modelo del Dominio

5.1.1.3. Actores

Los siguientes actores son los que interactúan con el Software:

<<Aplicación>> Cliente Administrador



Nodo **Cluster**

Figura 5-2: Actores de la Arquitectura del Middleware

- **Cliente**

Es toda aplicación que hace uso de los servicios del Software.

- **Administrador**

Es la persona encargada de agregar y quitar nodos del cluster.

- **Nodo**

Es una computadora con una replica de la base de datos original y una instancia de un sistema de gestión de base de datos.

- **Cluster**

Conjunto de nodos.

5.1.2. Casos de Uso

Se describe los casos de uso del Software.

5.1.2.1. Diagrama de Casos de Uso

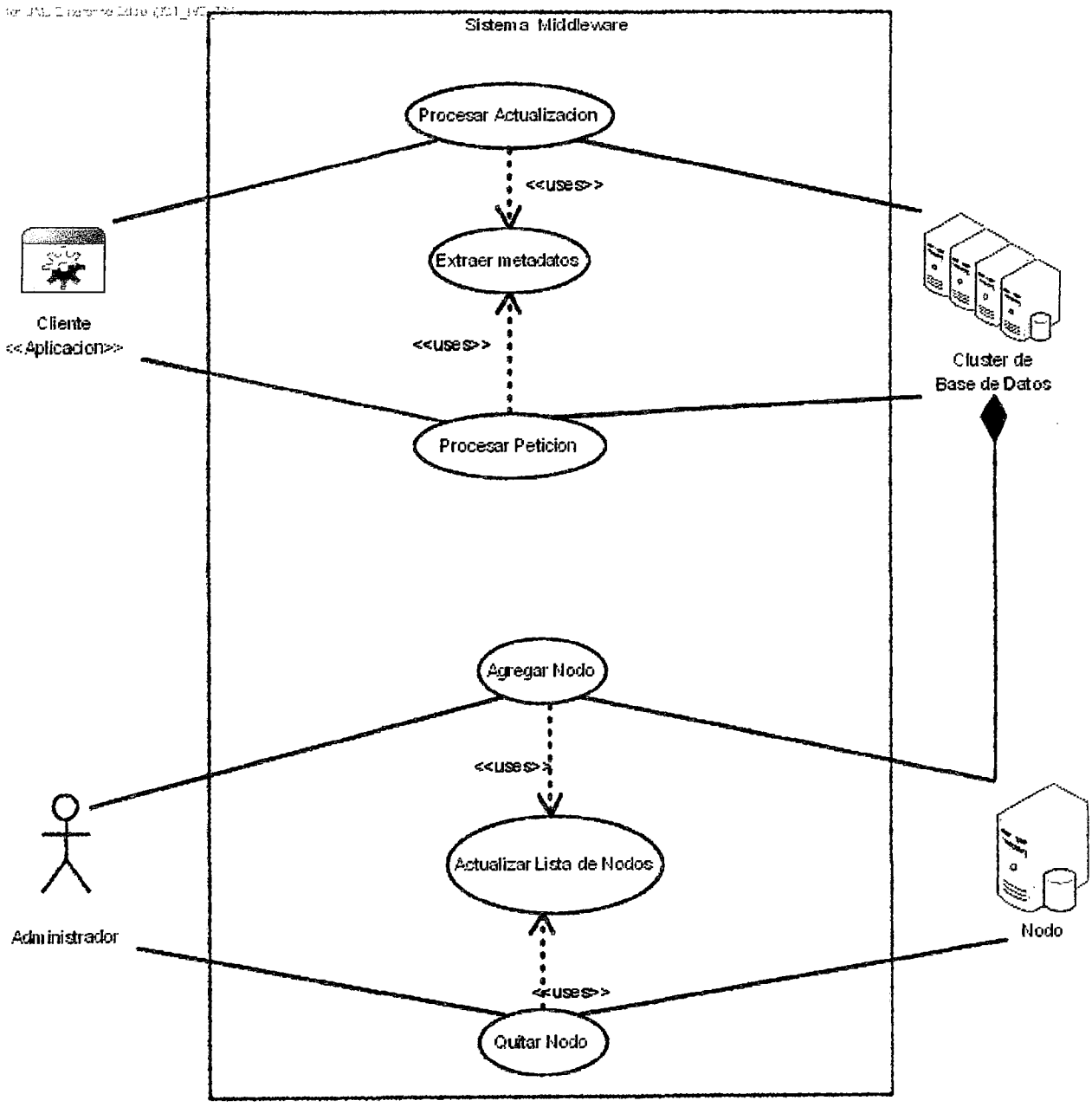


Figura 5-3: Casos de Uso - Middleware

5.1.2.2. Especificación de los Casos de Usos

Se describe los casos de usos en forma expandida.

5.1.2.2.1. Procesar Actualización (CU 1)

Nombre	**Procesar Actualización (CU 1)**
Actores	Cliente, Cluster de Base de Datos
Actividades	Procesar consulta de actualización, Ejecutar consulta en todos los nodos.
Visión General	Este caso de uso comienza cuando el cliente envía una consulta, y esta es recepcionada por el Software. . El Software extrae los metadatos de la consulta y propaga la actualización entre todos los nodos del cluster. Cada nodo del cluster ejecuta la actualización. El Software retorna mensaje de éxito al cliente.
Pre condición:	{ El nodo cuenta con una base de datos replicada de la base de datos principal }
Post condición:	{ Todos los nodos se encuentran sincronizados con la base de datos principal }
Curso Típico de Eventos	1. El cliente envía una consulta de actualización (INSERT, DELETE UPDATE) al Software. 2. El Software extrae los metadatos de la consulta(CU 5). 3. El Software propaga la actualización entre todos los nodos del cluster. 4. Cada nodo del cluster ejecuta la actualización. 5. El Software retorna mensaje de éxito al cliente.
Extensiones	3a. Mensaje de error: 1. Error al propagar la actualización. 4a. Mensaje de error 1. Error al ejecutar la actualización en algún nodo o en todos.

Diagrama de Actividad - Procesar Actualización

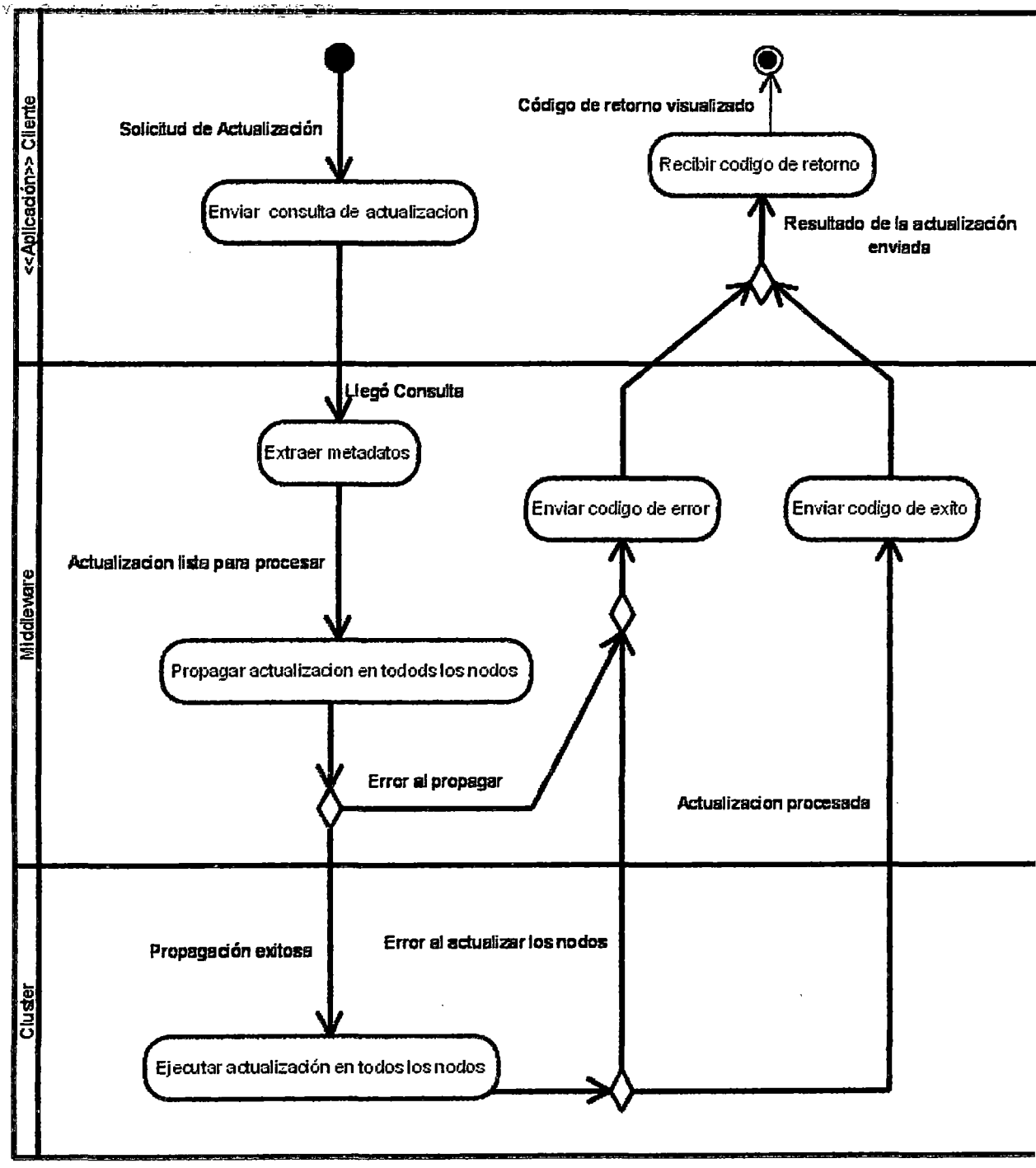


Figura 5-4: Diagrama de Actividad del Proceso Procesar Actualización

5.1.2.2.2. Procesar Petición (CU 2)

Nombre	**Procesar Petición (CU 2)**
Actores	Cliente, Cluster de Base de Datos
Actividades	Procesar consulta de petición, subdividir la consulta, ejecutar las sub-consultas en los nodos y retornar resultados al Software.
Visión General	Este caso de uso comienza cuando el cliente envía una consulta de petición (SELECT) y esta es recepcionada por el Software. . El Software extrae los metadatos de la consulta y subdivide verticalmente la consulta, luego ejecuta las subconsultas en el cluster de BD y retorna resultados.
Pre condición:	{ El nodo cuenta con una base de datos replicada de la base de datos principal }
Post condición:	{ Todos los nodos se encuentran sincronizados con la base de datos principal }
Curso Típico de Eventos	1. El cliente envía una consulta de petición (SELECT). 2. . El Software extrae los metadatos de la consulta). 3. El Software subdivide la consulta verticalmente. 4. El Software asigna las subconsultas a los nodos para su procesamiento concurrente. 5. El cluster procesa las subconsultas. 6. El Software reúne los resultados parciales retornados por los nodos. 7. El Software retorna el resultado al cliente
Extensiones	3a. Error al subdividir: 1. El Software envía un mensaje de error. 4a. Error al asignar las subconsultas a los nodos del cluster: 1. El Software envía un mensaje de error. 5a. Error al procesar las subconsultas 1. El Software envía muestra un mensaje de error 6a. Error al reunir los resultados:

	1. El Software envía un mensaje de error.

Diagrama de Actividad - Procesar Petición

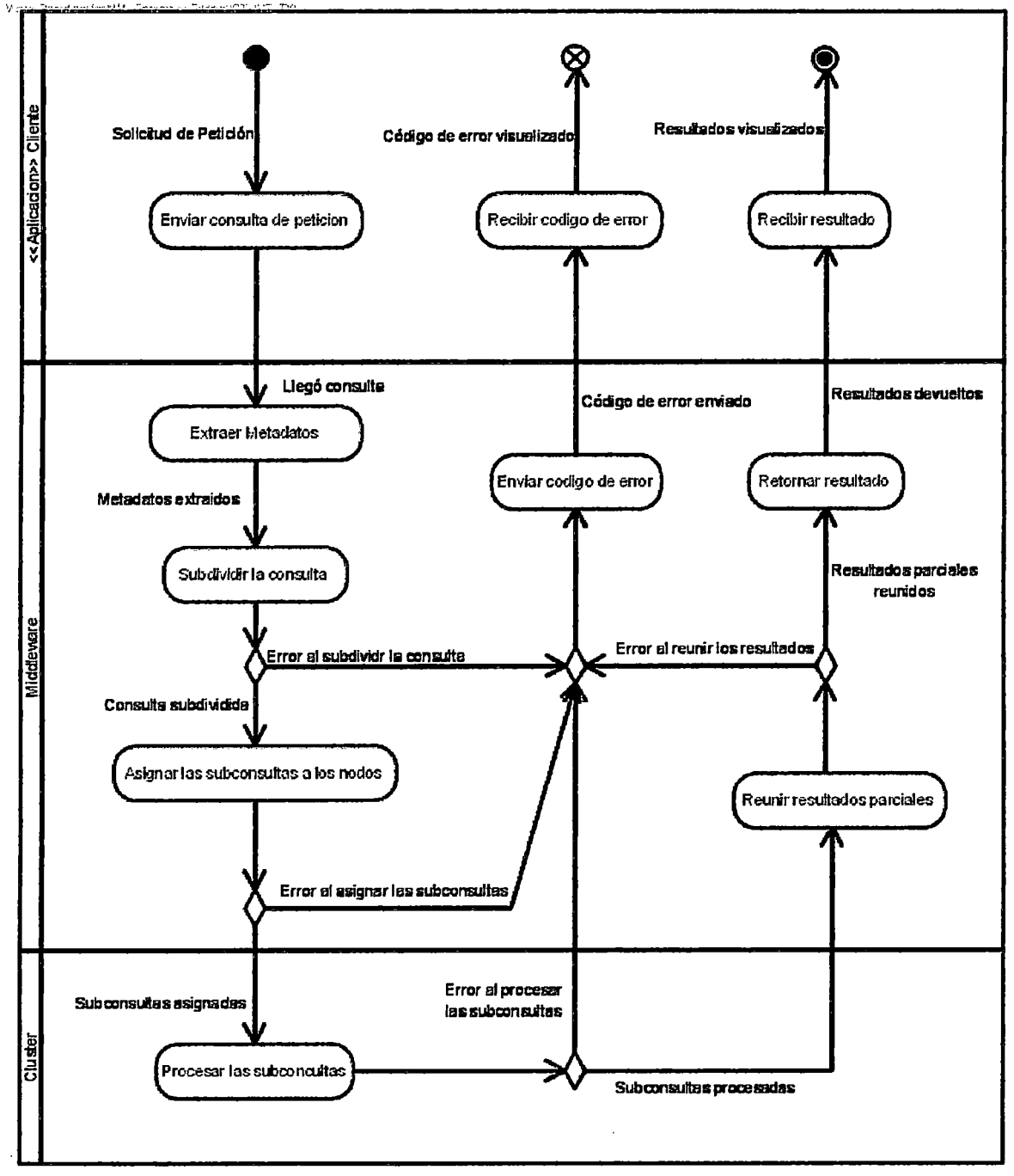


Figura 5-5: Diagrama de Actividad del Proceso Procesar Petición

5.1.2.2.3. Agregar Nodo (CU 3)

Nombre	**Agregar Nodo (CU 3)**
Actores	Administrador, Software
Actividades	Agregar un nodo al cluster y actualizar el cluster
Visión General	Este caso de uso comienza cuando el administrador adiciona un nodo. El Software actualiza el cluster.
Pre condición:	{ El nodo cuenta con una replica de la base de datos principal y un SGBD }
Post condición:	{ El nodo se agrego a la lista de nodos del Software }
Curso Típico de Eventos	1. El administrador adiciona un nodo al cluster. 2. El Software se conecta al nodo. 3. El Software actualiza el cluster (CU 6). 4. El Software retorna al administrador una notificación de éxito.
Extensiones	2a. El Software no se pudo conectar al nodo: 1. El Software envía un mensaje de error al administrador.

Diagrama de Actividad – Agregar Nodo

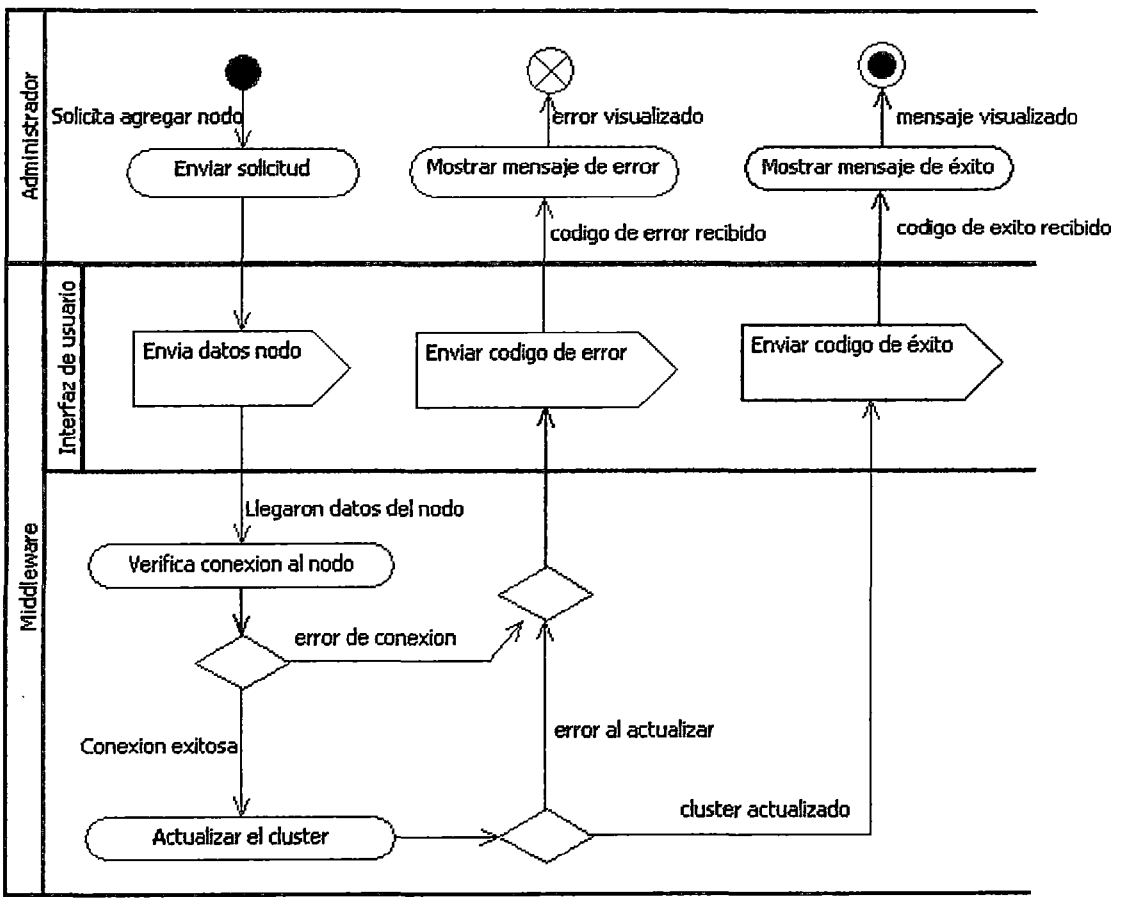


Figura 5-6: Diagrama de Actividad del Proceso Agregar Nodo

5.1.2.2.4. Quitar Nodo (CU 4)

Nombre	**Quitar Nodo (CU 4)**
Actores	Administrador, Software
Actividades	Quitar un nodo del cluster, actualizar el cluster
Visión General	Este caso de uso comienza cuando el administrador quita un nodo del cluster. El Software quita el nodo de su lista de nodos. El Software actualiza el cluster.
Pre condición	{ El nodo se encuentra en la lista de nodos del Software }
Post condición:	{ El nodo fue eliminado de la lista de nodos del Software }
Curso Típico de Eventos	1. El administrador quita un nodo del cluster. 2. El Software quita el nodo de su lista de nodos. 3. El Software espera a que el nodo termine su carga de trabajo. 4. El Software se desconecta del nodo. 5. El Software retorna al administrador una notificación de éxito.
Extensiones	N.A.

Diagrama de Actividad – Quitar Nodo

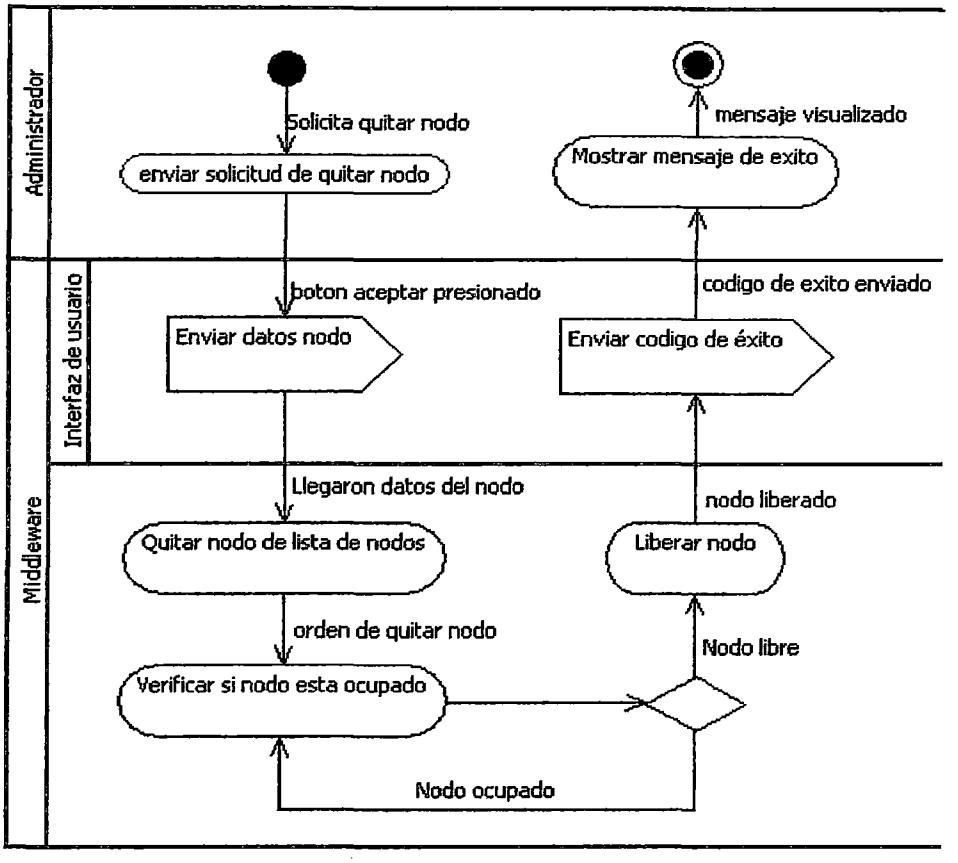


Figura 5-7: Diagrama de Actividad del Proceso Quitar Nodo

5.1.2.3. Casos de Uso de solo Inclusión

5.1.2.3.1. Extraer Metadatos (CU 5)

Nombre	**Extraer metadatos (CU 5)**
Actor	
Actividades	Extraer los metadatos de una consulta
Visión General	Este caso de uso comienza cuando el Software extrae los metadatos de una consulta de Actualización o Petición.

Pre condicion	{ La consulta fue decepcionada por el Software }
Post condicion	{ Se extrayeron los metadatos de la consulta }
Curso Típico de Eventos	1. El Software analiza una consulta de Actualización o Petición. 2. El Software extrae los metadatos de una consulta
Extensiones	N.A.

5.1.2.3.2. Actualizar la Lista de Nodos (CU 6)

Nombre	**Actualizar Lista de Nodos (CU 6)**
Actor	
Actividades	Actualizar lista de nodos
Visión General	Este caso de uso comienza cuando el Software actualiza su lista de nodos según sea la acción de agregación o eliminación de un nodo del cluster.
Pre condicion	{ Se agregó o eliminó un nodo del Software }
Post condicion	{ La lista de nodos fue actualizada }
Curso Típico de Eventos	1. El Software actualiza su lista de nodos. 2. El Software informa sobre la actualización de su lista de nodos.
Extensiones	N.A.

5.2. VISTA DE RESTRICCIONES

Aquí se presentan las restricciones normativas, de estándares y de tecnológicas, a las cuales esta sujeto el proceso de desarrollo del Software.

5.2.1. NORMATIVAS

En nuestro país existen las siguientes leyes y normas que podrían afectar al desarrollo del Software. Se tomaron las medidas necesarias para no incurrir en infracción a alguna de ellas.

5.2.1.1. Leyes

a) Delitos informáticos

Resolución Suprema Nro 026-2002-MTC
Contempla los siguientes delitos informáticos entre otros:

- Violaciones de Propiedad Intelectual.
- Uso ilegal de marcas.
- Vulneración de la privacidad de la información.

b) Piratería y derechos de autor:

Ley sobre el derecho de autor (Decreto Legislativo 822, Diario Oficial El Peruano 24 de Abril de 1996).
Código Penal (Ley Nro 28289, "Ley de Lucha Contra la Piratería", Diario Oficial el Peruano, 20 Julio del 2004).
Decisión Andina 351 ("Régimen Común sobre Derecho de Autor y Derechos Conexos", Cap VIII "De los Programas de Ordenador y Base de Datos", Comunidad Andina).

5.2.1.2. Licenciamiento

El software aquí desarrollado será entregado a la comunidad bajo licencia publica GNU, (GPL), la cual se encuentra detallada en el Anexo A.

5.2.2. ESTANDARES

El desarrollo del Software utilizará los estándares siguientes

5.2.2.1. UML

UML fue adoptado en Noviembre de1997 por OMG[9] (Object Management Group) como una de sus especificaciones y desde entonces se ha convertido en un estándar de facto para visualizar, especificar y documentar los modelos que se crean durante el desarrollo de un software. UML ha ejercido un gran impacto en la comunidad software, tanto a nivel de desarrollo como de investigación.

Desde la primera versión UML ha evolucionado a través de diferentes versiones. En Octubre del 2004 se adopto el estándar actual UML 2.0. uno de cuyos objetivos principales es proporcionar una sólida base para MDA[10]. Se incorporo soporte para el desarrollo basado en componentes, y también incorporo mejoras relacionadas con los diagramas de secuencias y con el modelado de comportamiento en general [UML06].

Para el modelado del Software se utilizará el estándar UML 2.0.

---

[9] El **Object Management Group** u **OMG** (de sus siglas en inglés Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización NO lucrativa que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para tecnologías orientadas a objetos. El grupo está formado por compañías y organizaciones de software como lo son: Hewlett-Packard (HP), IBM, Sun Microsystems, Apple Computer.

[10] es un estándar propuesto por el OMG [Object Management Group] que promete acelerar el desarrollo de aplicaciones, simplificar la integración entre distintas tecnologías y reducir el coste de la migración de las aplicaciones a nuevas plataformas.

5.2.2.2. SQL

Las organizaciones que se involucraron en la estandarización de SQL, y por lo tanto en el desarrollo de SQL:1999 son ANSI[11] e ISO[12]. Más específicamente la comunidad internacional de trabajos mediante ISO/IEC JTC1 (Joint Technical Comité 1), un comité formado por la organización internacional de estandarización junto con la comisión internacional electrotécnica. La responsabilidad de las JTC1 es desarrollar y mantener la información relativa a la tecnología. Dentro de JTC1, el subcomité SC32, cuya función era el intercambio y gestión de datos se formo para la estandarización de normas relativas a varias bases de datos y metadatos. SC32, actualmente viene desarrollando trabajos técnicos como WG3(Lenguajes de bases de datos) es responsable de las normas SQL, mientas WG4 esta desarrollando el SQL/MM (SQL multimedia, un departamento de normas que especifiquen las bibliotecas de tipos usando facilidades de SQL orientado a objetos).

Las primeras versiones de la norma son conocidas como SQL-86(o SQL-87, porque la versión no fue publicada hasta 1987), SQL-89 y SQL-92. La versión actual es SQL-1999 [AAS05].

En el desarrollo del Software se utiliza el estándar SQL-1999.

[11] El **Instituto Nacional Estadounidense de Estándares** (ANSI, por sus siglas en inglés: American National Standards Institute) es una organización sin ánimo de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.

[12] La **Organización Internacional para la Estandarización** (**ISO**) es una organización internacional no gubernamental, compuesta por representantes de los organismos de normalización (ONs) nacionales, que produce normas internacionales industriales y comerciales. Dichas normas se conocen como **normas ISO** y su finalidad es la coordinación de las normas nacionales, en consonancia con el Acta Final de la Organización Mundial del Comercio, con el propósito de facilitar el comercio, facilitar el intercambio de información y contribuir a la transferencia de tecnologías.

5.2.3. TECNOLOGIA

El desarrollo del Software deberá estar realizado en la plataforma Microsoft Windows, pudiendo extenderse a otras plataformas mediante el uso de la tecnología adecuada.

Además de utilizar un conjunto completo de herramientas de desarrollo

5.2.4. SOPORTE

El soporte del programa se rige al especificado en el documento de licencia publica GNU (GPL) bajo el cual se entrega este proyecto a la comunidad, el cual puede ser consultado en el Anexo A.

5.3. VISTA QoS

En esta sección describimos los requerimientos no-funcionales del Software:

5.3.1. Usabilidad

El principal criterio para hacer el Software un sistema usable es tomar en cuenta la dificultad que presentan los casos de uso de alta frecuencia. Como se ve en los casos de uso Agregar y Quitar Nodo, la dificultad depende del conocimiento que el administrador debe tener en cada paso, las decisiones que éste debe realizar en cada paso, y la mecánica de cada paso; por ejemplo al agregar un nodo al cluster, el administrador deberá proporcionar todos los parámetros solicitados por el Software; en caso fallará, el Software retorna un mensaje de error en el cual el administrador volverá a ingresar los datos solicitados por la interfaz del Software. El administrador realiza los pasos indicados hasta que el Software retorne un mensaje de éxito.

5.3.2. Confiabilidad

El Software no debe fallar en los procesos de: procesar consultas de petición, actualización, agregar y quitar nodo. Ya que son críticos para el buen funcionamiento del Software.

5.3.3. Performance

Siendo el objetivo principal del Software ofrecer alto rendimiento, este debe ser el requerimiento con mayor prioridad.
Por lo tanto se debe optimizar el funcionamiento de todo el conjunto de procesos que son parte del Software.

5.3.4. Seguridad

Las medidas mínimas que el Software debe considerar son los siguientes

- El acceso al Sistema esta restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al sistema las personas que estén registradas.
- Las contraseñas de seguridad deben tener de 8 a 20 caracteres de longitud. e incluir por lo menos dos números y dos caracteres alfanuméricos
- El Software debe ubicarse tras un Firewall.
- La red dedicada del cluster no debe tener puntos de acceso ni de salida que no sean a través del Software.
- Las aplicaciones que hagan uso de los servicios del Software deben quedar debidamente registradas en un registro de transacciones del sistema.

5.3.5. Escalabilidad

El Software debe permitir y ofrecer todos los métodos y mecanismos necesarios para hacer crecer el sistema, el incremento de nodos debe ser transparente y no afectar lo menos posible al funcionamiento del Software.

5.4. VISTA LOGICA

Esta vista presenta 3 niveles de arquitectura para el Software. Cada nivel corresponde a un refinamiento del nivel anterior. El ultimo nivel es el q presenta mayores detalles; en el se presentan los módulos participantes de la arquitectura junto a un diagrama.

Este capitulo se organiza de la siguiente forma.

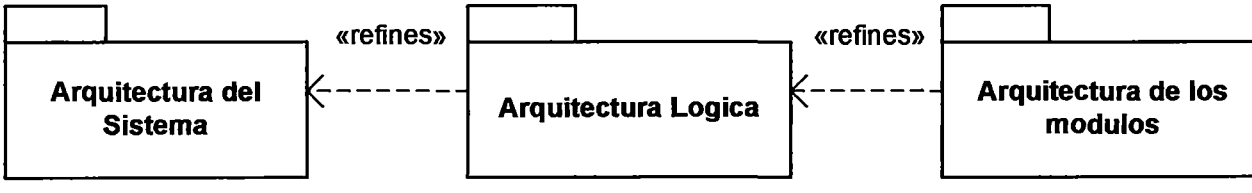


Figura 5-8: Niveles de la Arquitectura del Middleware

5.4.1. Arquitectura del Sistema

Esta arquitectura esta organizada utilizando del patrón de arquitectura en capas; el mismo se conforma de 4 capas. El siguiente diagrama presenta la arquitectura del sistema.

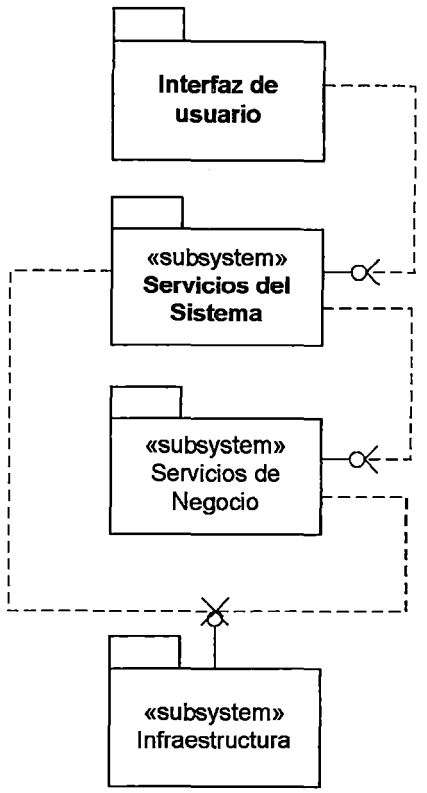


Figura 5-9: Arquitectura del Sistema del Middleware

La interfaz de usuario, tiene como objetivo el manejo de la lógica del usuario, en este caso específico el administrador del sistema, que se encargara de la Gestión de Nodos.

Los servicios del sistema, representan los servicios básicos que debe proveer el sistema; estos servicios son directamente utilizados por los módulos de la capa superior. Los servicios en esta capa son de función específica y particulares para cada subsistema del Software.

Los servicios de negocio, son servicios de manejo de información del negocio; son servicios aun más básicos que los de la capa superior, esto permite que sean reutilizados por los servicios de sistema.

La capa de la infraestructura, contiene todos los módulos necesarios para utilizar los servicios de la plataforma. Como por ejemplo los mecanismos de comunicación o las capas de acceso a datos.

5.4.2. Arquitectura Lógica

La arquitectura lógica presenta un refinamiento de la arquitectura del sistema. La dimensión de los requerimientos, principalmente la vista de casos de uso va a verse realizada por los módulos aquí presentados. Se analizará los módulos presentes en cada capa de la arquitectura del sistema, presentando finalmente la arquitectura lógica.

5.4.2.1. Interfaz de Usuario

La vista de casos de uso muestra el front-end del sistema. Para nuestro Sistema solo los casos de uso "Agregar Nodo" y "Quitar Nodo" presentan una interfaz de usuario los demás casos de uso no presentan interfaces de usuario por lo tanto sus interfaces de comunicación no se encuentran en esta capa.

Esta capa consiste en un modulo por cada caso de uso identificado, cada modulo contiene la lógica que lleva adelante el caso de uso y los formularios respectivos.

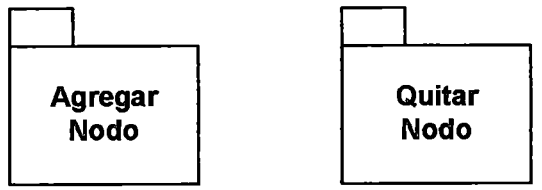


Figura 5-10: Front- end del Middleware

5.4.2.2. Servicios del Sistema

Cada Modulo de la capa superior utiliza los servicios de esta capa, aquí se cuenta con una interfaz de comunicación por cada caso de uso; estas ofrecen los servicios que los módulos de la capa superior requieren, aquí también se incluyen las interfaces de comunicación con la que interactuaran los sistemas externos al Software, exactamente para el procesamiento de consultas.

El siguiente diagrama muestra los módulos identificados.

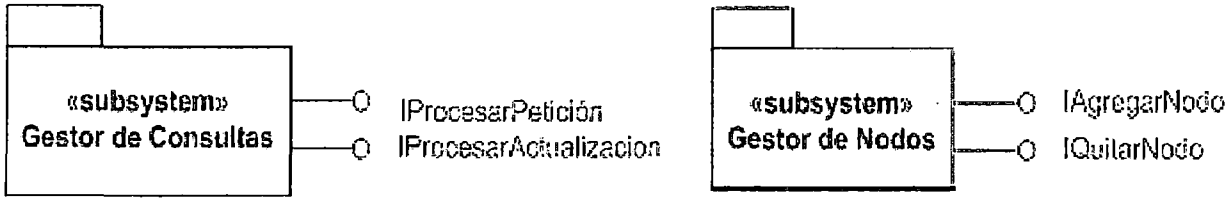


Figura 5-11: Servicios del Sistema

5.4.2.3. Servicios del Negocio

Los Servicios provistos por estos módulos son los necesarios para poder llevar adelante las operaciones de los módulos de la capa superior.

Los módulos identificados son Analizador de Consultas, Divisor de Consultas, Coordinador de Ejecución de Consultas y Ejecutor de Consultas, cada modulo ofrece una única interfaz con los servicios requeridos.

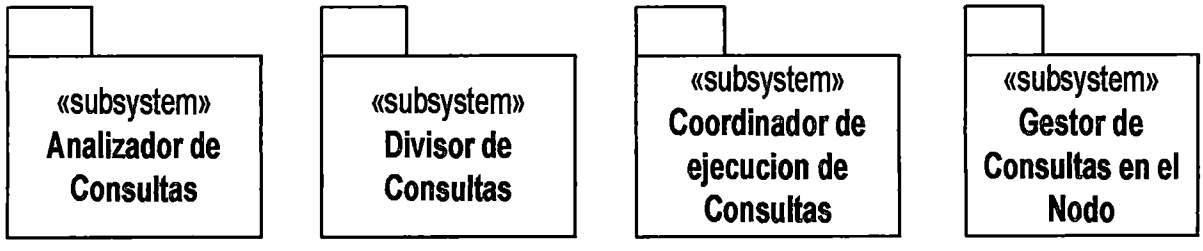


Figura 5-12: Servicios del Negocio

5.4.2.4. Infraestructura

Los módulos de esta capa son de 2 tipos, adaptadores a sistemas existentes o servicios generales útiles para cualquier aplicación. Estos módulos están disponibles para todos los módulos en las otras capas.
Aquí podemos encontrar los siguientes sistemas: Sistema de Comunicaciones, que encapsula todo lo referente para las comunicaciones a través de la red. Y por ultimo el modulo de Acceso a Datos, que encapsula la conexión al SGBD.

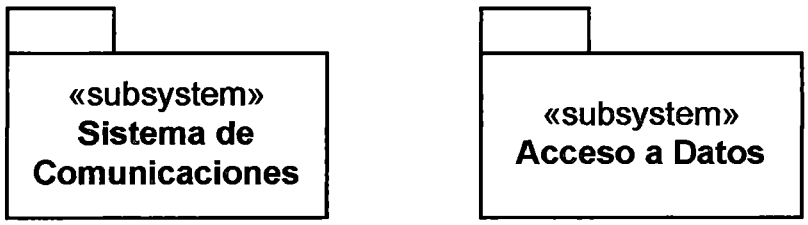


Figura 5-13: Infraestructura

5.4.3. Arquitectura de los Módulos

La arquitectura de los módulos presenta un refinamiento de la arquitectura lógica. Esta incluye, para cada modulo, el punto de vista que mejor define su diseño. Para cada tipo de modulo de cada capa se utilizará un diagrama diferente.

5.4.3.1. Interfaz de Usuario

Los módulos en esta capa encapsulan la lógica del caso de uso (Ver los diagramas de Actividades de los Casos de Uso: Agregar y Quitar Nodo).

5.4.3.2. Servicios de Sistema

Los módulos de esta capa encapsulan la lógica del sistema. Los servicios ofrecidos son realizados utilizando servicios de negocio y servicios de infraestructura; esta realización se hace mediante el envió de mensajes a los módulos ubicados en estas capas. Por esta razón un diagrama de secuencia es el más adecuado para describir su diseño interno.
Para cada modulo se presenta los servicios brindados por cada una de sus interfaces y luego los diagramas de secuencia para las operaciones mas complejas.

5.4.3.2.1. Gestor de Consultas

Las interfaces provistas por este modulo se describen a continuación. La realización de las mismas se hace mediante la solicitud de los servicios respectivos a los módulos en la capa de servicios de negocio.

Figura 5-14: Servicios del Sistema – Procesar Actualización

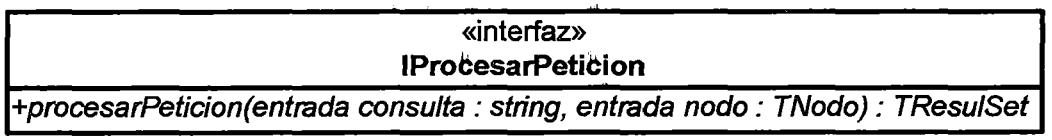


Figura 5-15: Servicios del Sistema - Procesar Petición

5.4.3.2.2. Gestor de Nodos

Las interfaces provistas por ese modulo se describen a continuación.

Figura 5-16: Servicios del Sistema – Agregar Nodo

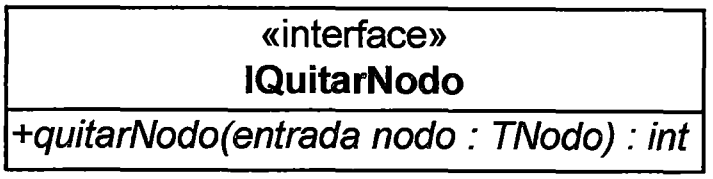


Figura 5-17: Servicios del Sistema – Quitar Nodo

5.4.3.3. Servicios de Negocio

Estos servicios manejan la información del Software; encapsulan la forma en que los datos están almacenados, la distribución de estos datos y el acceso a ellos. Por esta razón, una interfaz identificando sus servicios y el modelo de información es el adecuado para describir su diseño interno.

5.4.3.3.1. Analizador de Consultas

La interfaz soportada por este modulo es el siguiente:

Figura 5-18: Servicios de Negocio- Analizar Consulta

5.4.3.3.2. Divisor de Consultas

La interfaz soportada por este modulo es el siguiente:

Figura 5-19: Servicios de Negocio – Dividir Consulta

5.4.3.3.3. Coordinador de Ejecución de Consultas

La interfaz soportada por este modulo es el siguiente:

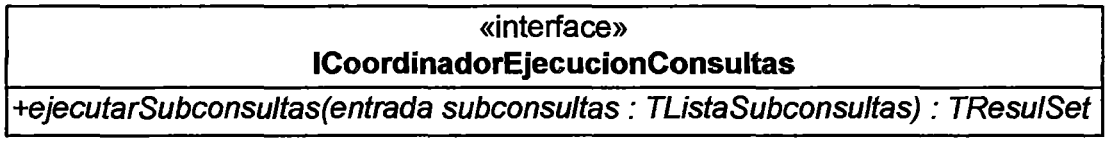


Figura 5-20: Servicios de Negocio – Ejecutar Subconsultas

5.4.3.3.4. Gestor de Consultas en el Nodo

La interfaz soportada por este modulo es el siguiente:

Figura 5-21: Servicios de Negocio – Gestor Consulta en el Nodo

5.4.3.4. Infraestructura

El refinamiento para los módulos en esta capa consta de la interfaz soportada por los mismos.

Se presenta a continuación estas interfaces:

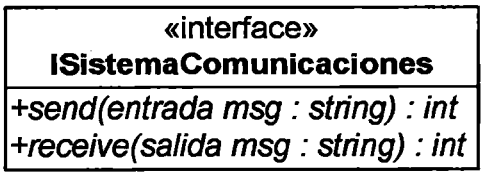


Figura 5-22: Infraestructura – Sistema de Comunicaciones

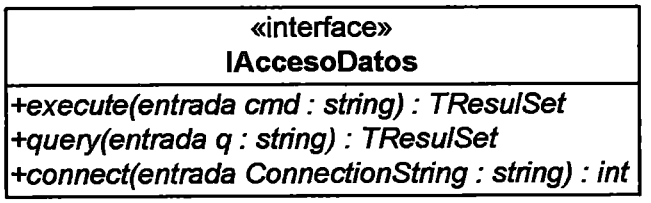


Figura 5-23: Infraestructura – Acceso a Datos

5.5. VISTA DE PROCESOS

La vista de procesos describe los módulos activos del Software; estos son módulos que estarán en ejecución en forma simultánea. Esta vista describe, además, el soporte multiusuario de la aplicación.

5.5.1. Procesos Distribuidos

5.5.1.1. Distribución de las Capas

De los módulos presentes en las diferentes capas que componen la aplicación, se encontraran distribuidos el "Ejecutor de Consultas" y el "Replicador de Base de Datos". Estos se distribuyen para mejorar el rendimiento, minimizar el tráfico por la red y acelerar el procesamiento de consultas.

5.5.1.2. Servicios de Infraestructura

Los servicios de Infraestructura son de dos tipos, aplicaciones llegadas que deben ser utilizadas y el motor de base de datos. Este último corre independientemente, atendiendo incluso solicitudes de otras aplicaciones.

5.5.2. Arquitectura de Procesos

En base a las decisiones descritas en la sección anterior se muestra a continuación la arquitectura de procesos del Software.

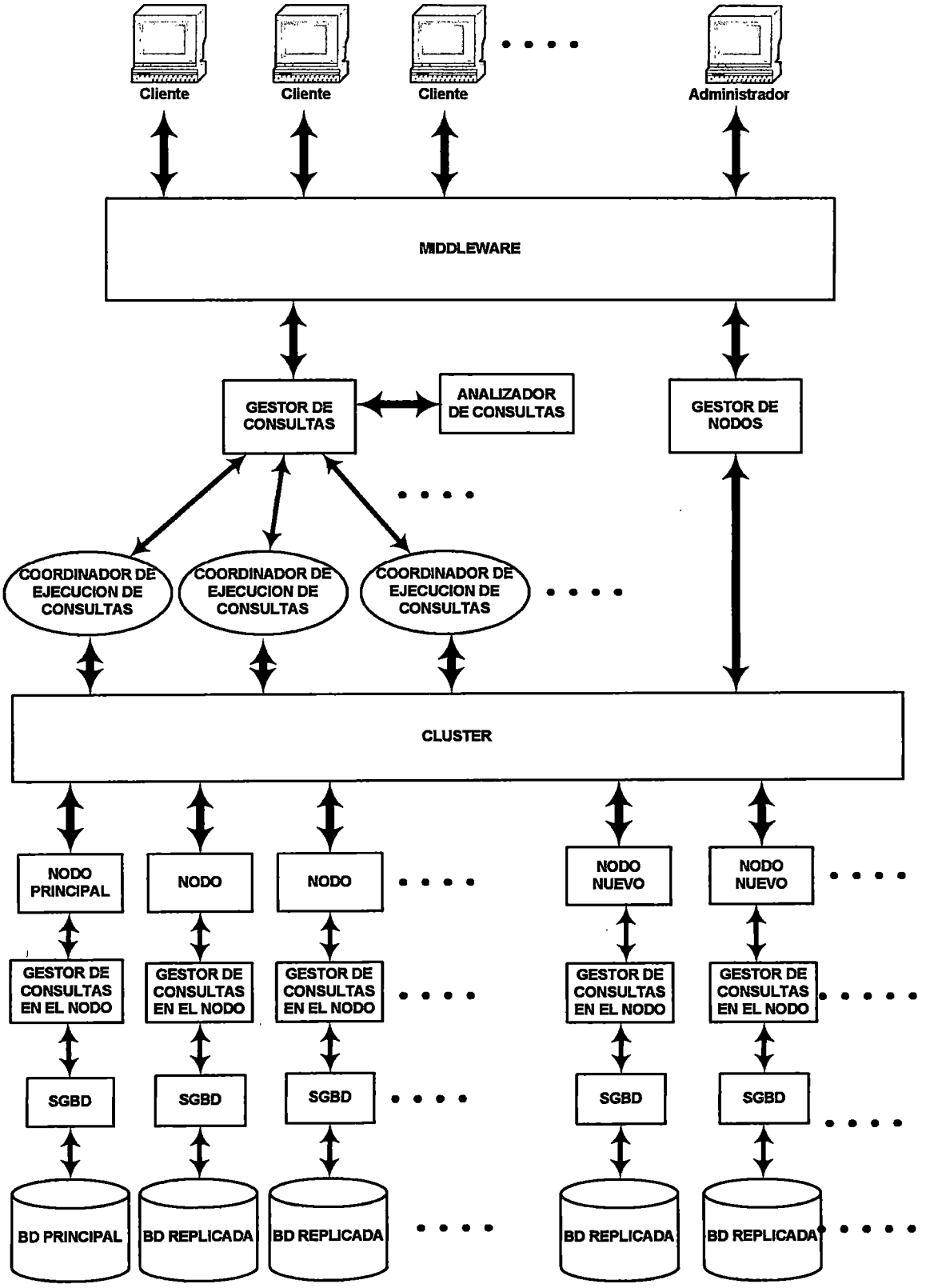


Figura 5-24: Arquitectura de Procesos del Middleware

Los procesos de AplicaciónCliente y SistemaAdministracion dependen directamente del Software. Ya que estos actuarán de acuerdo a los procesos del Software. El proceso encargado de recibir las consultas será el proceso Gestor de Consultas para que este administre las consultas y verifique los procesos que esta debería realizar, este envía la consulta al proceso AnalizadorConsultas mediante una llamada síncrona, luego enviará los resultados al proceso DivisorConsultas también mediante una llamada síncrona, luego creará un proceso CoordinadorEjecucionConsultas en un nuevo hilo de ejecución al cual le enviará los resultados del Divisor de Consultas, el proceso CoordinadorEjecucionConsultas interactuará con varios procesos GestorConsultasNodo para completar su trabajo de procesamiento.

El proceso GestorNodos recibirá las peticiones de agregar o quitar nodos, en cada caso el proceso GestorNodos realizará operaciones según corresponda.

5.6. VISTA DE IMPLEMENTACION

La vista de implementación muestra los componentes de deployment construidos para el Software.
Bajo entorno Windows la unidad de deployment son los ejecutables y las librerías dinámicas. Las librerías dinámicas son una colección de funcionalidades construida, versionada e instalada como una unidad de implementación, dentro de ellos se pueden almacenar múltiples objetos y clases.
Las librerías dinámicas pueden ser ejecutadas por si solas mediante el uso de la aplicación rundll32.exe que viene en todas las versiones de Windows
Además podemos añadir que si se desarrolla en java para ambiente multiplataforma la unidad será el package de extensión jar. Los package se ejecutan mediante el uso de la aplicación java.exe

5.6.1. Estructura de la Aplicación

Cada capa es decir cada subsistema en el primer refinamiento de la arquitectura lógica será desarrollado en una librería independiente del resto. Dentro de la librería de cada capa se encuentran los módulos que residen en ella desde el punto de vista lógico.
Las librerías para las capas de Servicios de Sistema, Servicios de Negocio e Infraestructura serán los siguientes respectivamente: Sistema.dll, negocio.dll, infraestructura.dll.
También se incluirá una librería adicional que cuenta con la biblioteca de clases utilizada para compartir información entre las capas: libreria.dll
También se incluirá una librería para los servicios de negocio que se ejecutan en los nodos: negocionodo.dll
Por ultimo se debe contar con una librería adicional conteniendo las interfaces de usuario, para los casos de uso Agregar Nodo y Quitar Nodo, que llamaremos, interfaces.dll

5.6.2. Arquitectura de Implementación

Las dependencias entre estos paquetes se muestran a continuación:

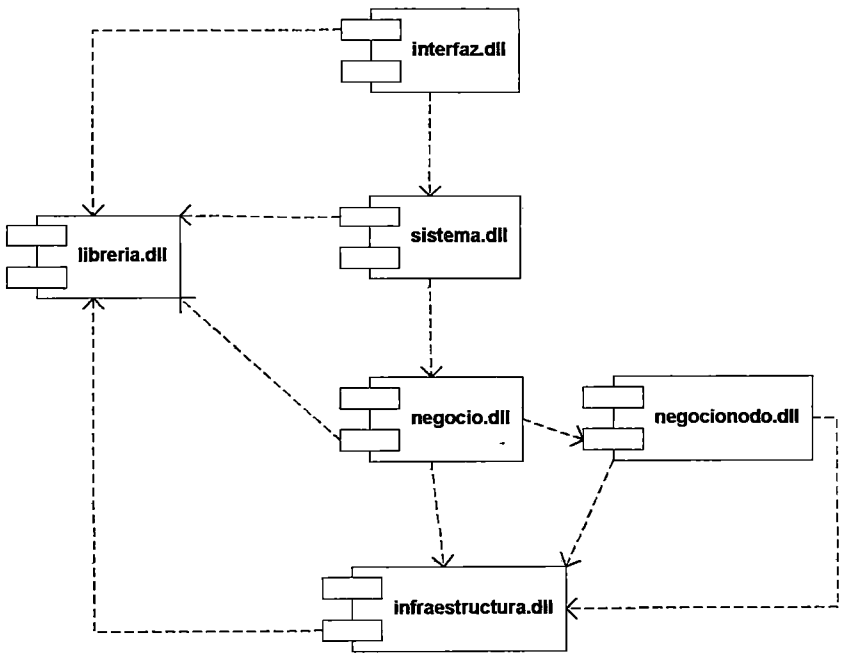


Figura 5-25: Arquitectura de Implementación

5.6.2.1. Infraestructura

Además, la capa de infraestructura hará uso de paquetes externos a la aplicación para la comunicación y para el acceso a datos aun no definidos en este documento de arquitectura:

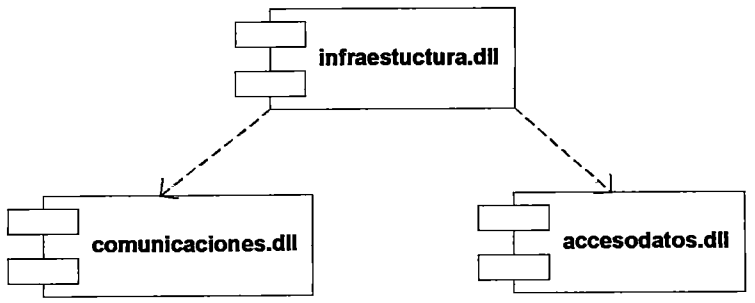


Figura 5-26: Arquitectura de Implementación - Infraestructura

5.7. VISTA FISICA O DE DEPLOYMENT

La vista física presenta la infraestructura necesaria para instalar el Software. Se presenta aquí la arquitectura técnica de la aplicación indicando los nodos presentes en la infraestructura, tecnológica esperada, y la localización de dichos componentes en dichos nodos.

5.7.1. Arquitectura Técnica

Considerando la distribución de la aplicación desde el punto de vista de los procesos es posible identificar 4 tipos de nodos:

- Cliente: que representa a las estaciones de trabajo de los usuarios que ejecutan aplicaciones de acceso a datos y que usaran los servicios del Software.
- Administrador: que representa a la estación de trabajo que sirve para administrar el Software, aunque esta podría estar ubicado en el servidor principal.
- Servidor Principal: es el servidor donde estará instalado el Software y todos sus componentes.
- Nodo: representa a los servidores adicionales que se pueden agregar o eliminar del cluster según se vea por conveniente. En el se ubicará una parte de la capa de negocio del Software.

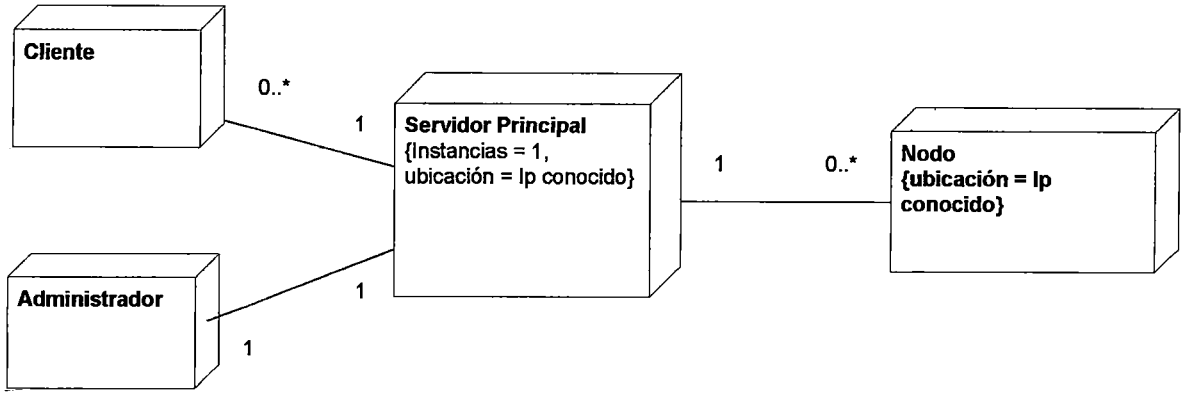


Figura 5-27: Arquitectura técnica

5.7.2. Tecnología Requerida

El servidor principal y cada nodo del cluster deben contar con un SGBD, que se encargue del manejo de la base de datos, además que servirá como repositorio para los datos del programa.

Todos los computadores deben estar interconectados con una línea de transmisión de alta velocidad y utilizar el protocolo TCP-IP

5.7.3. Deployment

A continuación se muestra la distribución de las librerías en los nodos:

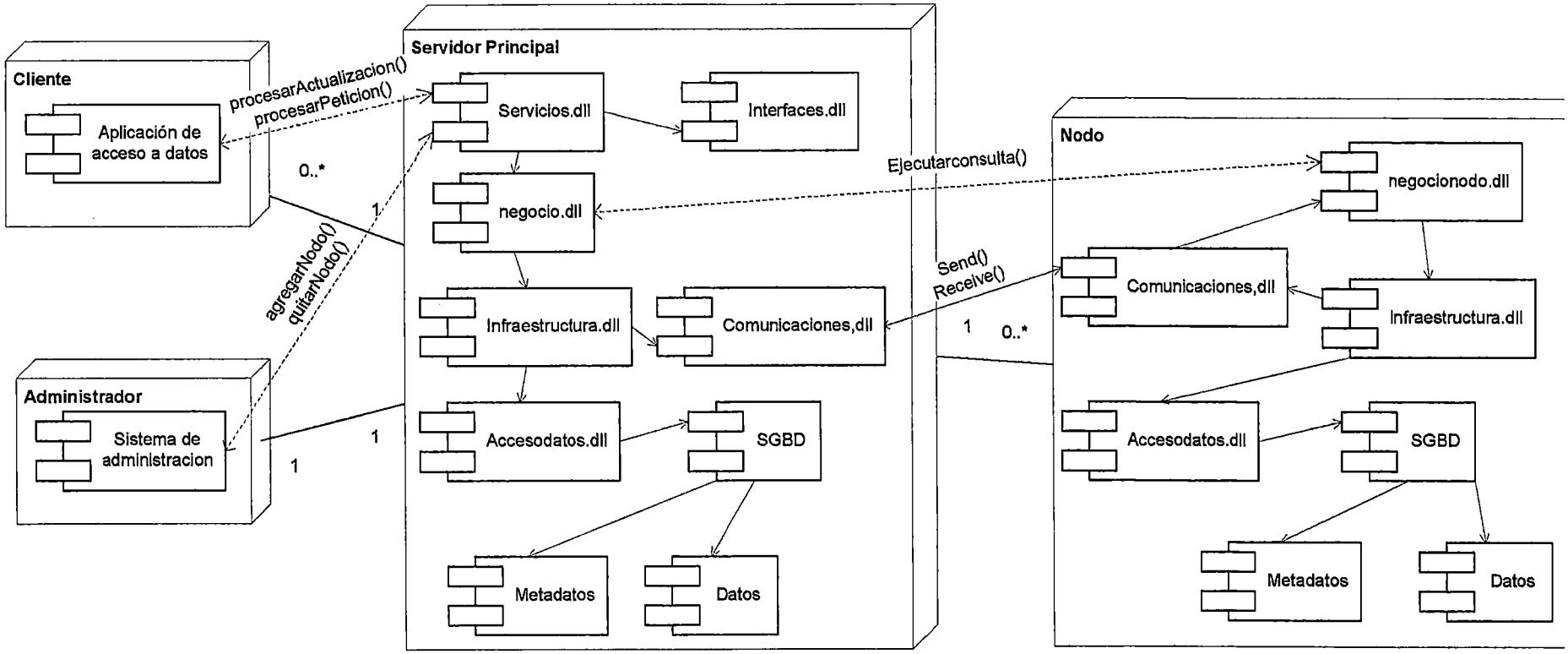


Figura 5-28: Distribución de las Librerías en los Nodos

CAPITULO VI

DESARROLLO DEL PROTOTIPO

Este capítulo desarrolla una iteración dentro de la fase de Elaboración del desarrollo del software, siguiendo la metodología RUP, la cual esta basada en la "Arquitectura del Software" desarrollado en el capitulo 5, el cual es producto de la fase de Inicio de la metodología.
Esta iteración desarrolla: el modelo de análisis, el modelo de diseño y el modelo de implementación.

Los artefactos producto del modelo de Análisis son: Diagramas de Casos de Uso de Análisis, Diccionario de Clases, Diagramas de Secuencia, Diagramas de Comunicación, Casos de Usos Expandidos.

Los artefactos producto del modelo de Diseño son: Diagrama de clases de Diseño, Diagrama de Componentes Black Box, Diagrama de Componentes White Box.

El artefacto producto del modelo de Implementación es el diagrama de Despliegue.

6.1. MODELO DE ANALISIS

El resultado del análisis será un modelo conceptual, formando un paquete que se denomina Modelo del Análisis.

El proceso de análisis se realiza mediante los siguientes pasos:

6.1.1. Diagramas de Casos de Uso Actualizado

6.1.1.1. Diagrama de Casos de Uso: Procesar Consulta

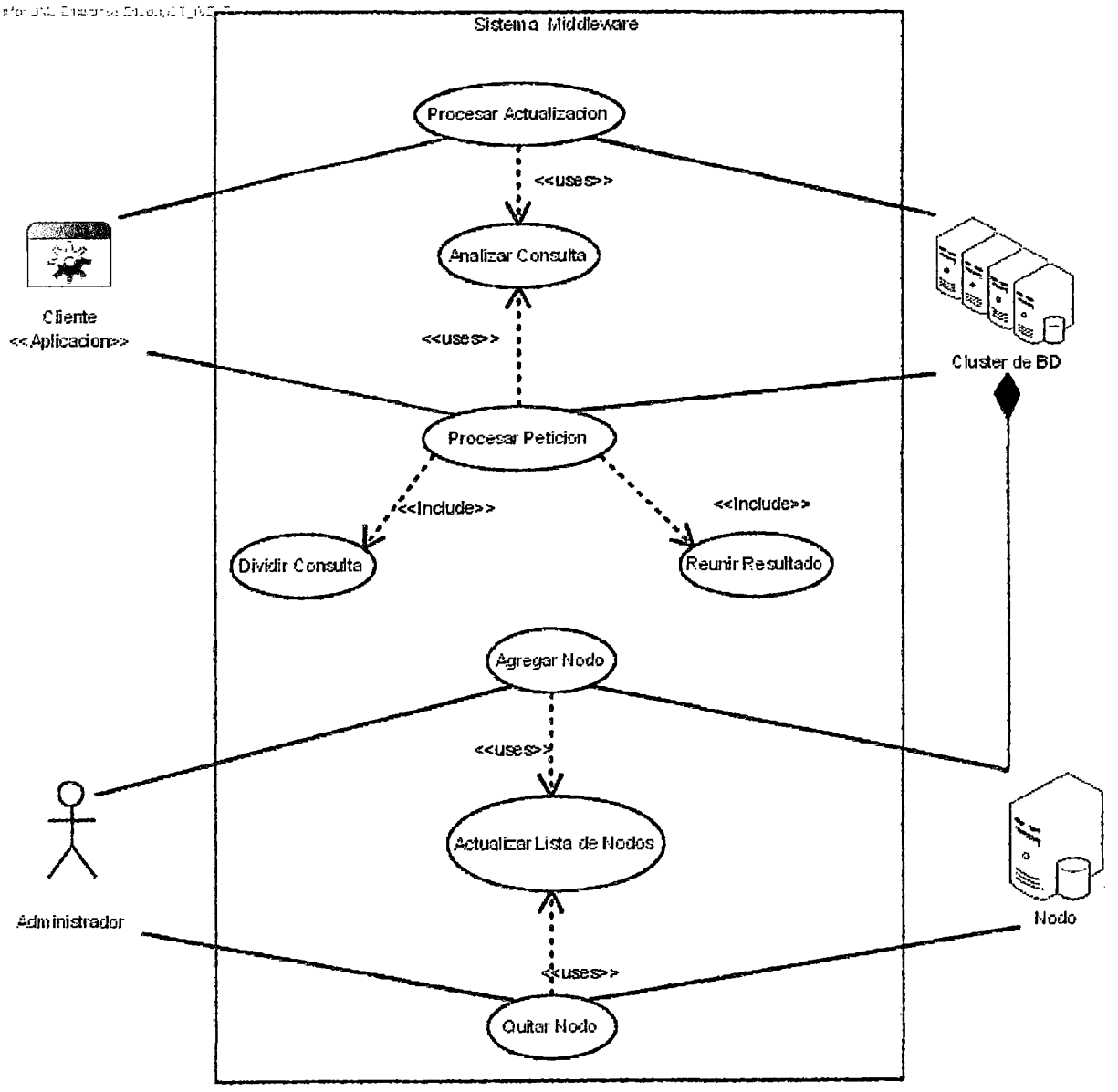


Figura 6-1: Diagrama de Casos de Uso Actualizado Procesar Consulta

6.1.1.2. Diagrama de Casos de Uso: Procesar Consulta en Nodo

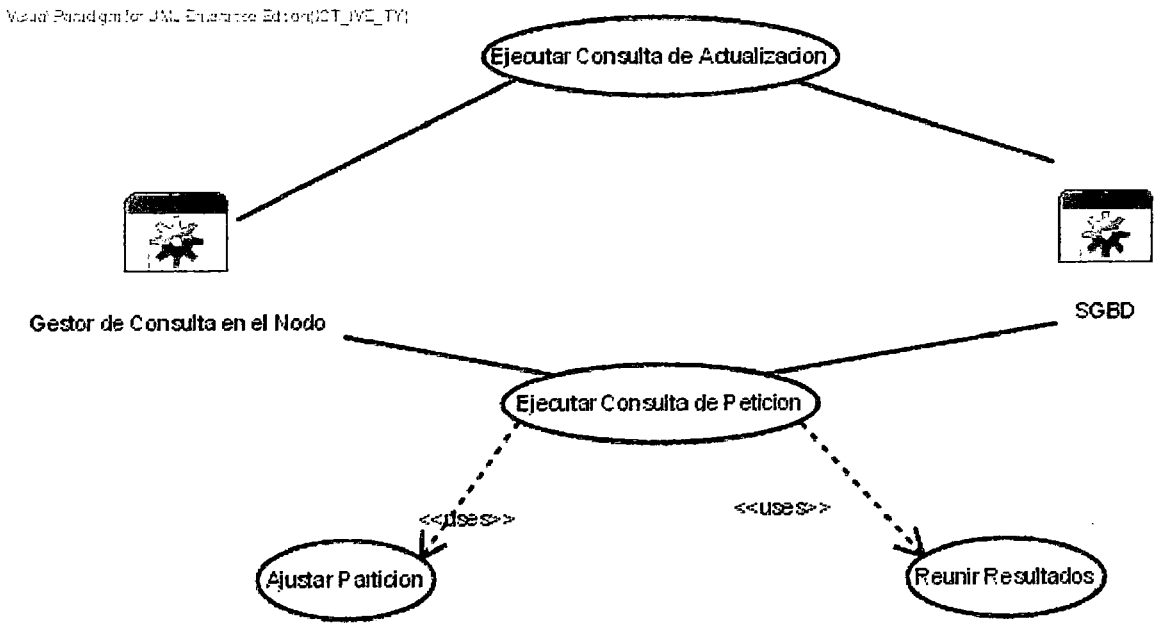


Figura 6-2: Diagrama de Casos de Uso Actualizado Procesar Consulta en el Nodo

6.1.2. Diccionario de Clases

- Gestor de consultas

Es el encargado de recibir las consultas de actualización y petición de los clientes

- Analizador de consultas

Es el encargado de extraer los metadatos de las consultas

- Divisor de consultas

Es el encargado de subdividir las consultas de acuerdo al número de nodos existentes en el cluster

- Coordinador de ejecución de consultas

Es el encargado de supervisar el procesamiento de una consulta, se crea uno por cada consulta

- Gestor de consultas en el nodo

Es el encargado de recibir las consultas de actualización o petición en el nodo

- Coordinador de ejecución de consultas en el nodo

Es el encargado de supervisar el procesamiento de una consulta en el nodo, se crea uno por cada consulta.

- Ejecutor de consulta en el nodo

Es el encargado de interactuar con el gestor de base de datos para el procesamiento de las consultas.

- Gestor de nodos

Es el encargado de recibir las peticiones de agregar o quitar nodo del cluster.

- Ajustador de partición

Es el encargado de proporcionar al coordinador de ejecución en el nodo el tamaño de partición virtual mas adecuado para procesar las sub consultas.

- Recolector de resultados

Esta clase sirve para almacenar y reunir los resultados parciales temporalmente y entregar el resultado final a quien lo solicite.

6.1.3. Análisis de Casos de Uso

Al análisis de Casos de Uso también se le denomina Refinamiento de Casos de Uso. Refinamos los casos de uso mostrando como colaboran los objetos de las clases requeridas para cumplir con la funcionalidad del caso de uso.

Análisis de Clases según Casos de Uso

6.1.3.1. Caso de Uso: Procesar Actualización

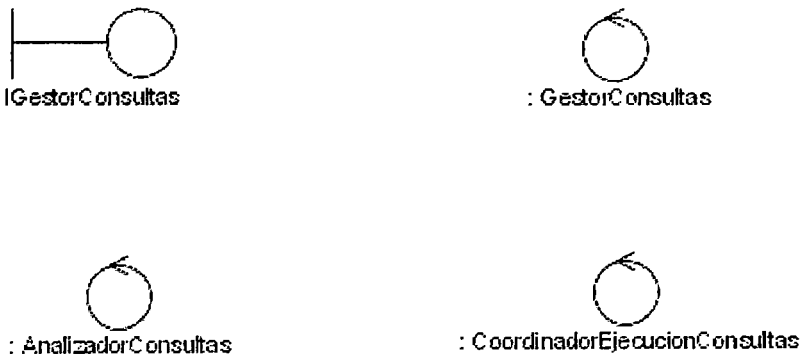


Figura 6-29: Clases de Análisis Procesar Actualización

6.1.3.2. Caso de Uso: Procesar Petición

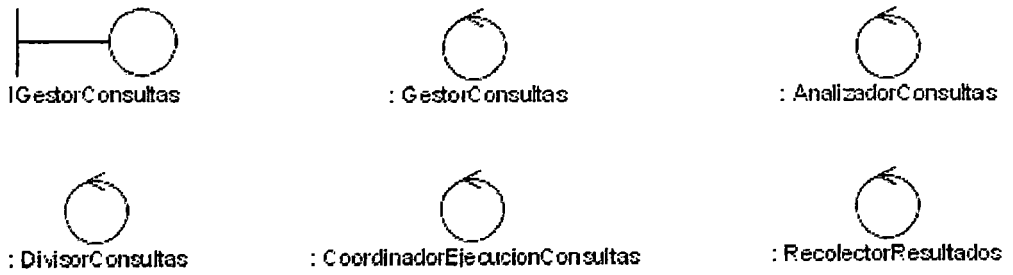


Figura 6-4: Clases de Análisis Procesar Petición

6.1.3.3. Caso de Uso: Agregar Nodo

Figura 6-5: Clases de Análisis Agregar Nodo

6.1.3.4. Caso de Uso: Quitar Nodo

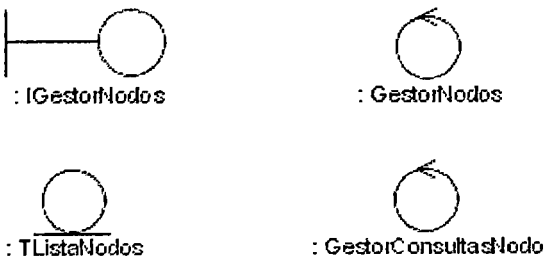


Figura 6-6: Clases de Análisis Quitar Nodo

6.1.3.5. Actualización en el Nodo

Figura 6-7: Clases de Análisis Actualización en el Nodo

6.1.3.6. Petición en el Nodo

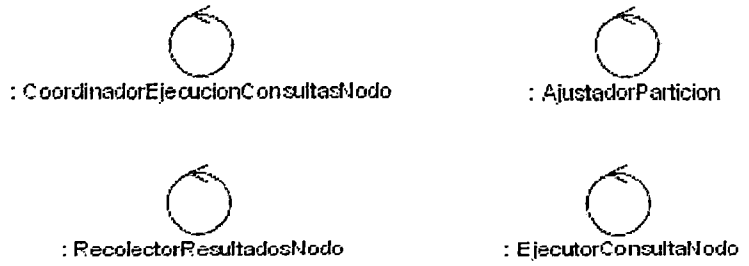


Figura 6-8: Clases de Análisis Petición en el Nodo

6.1.4. Diagrama de Secuencia

6.1.4.1. Gestor de Consultas: Procesar Actualización

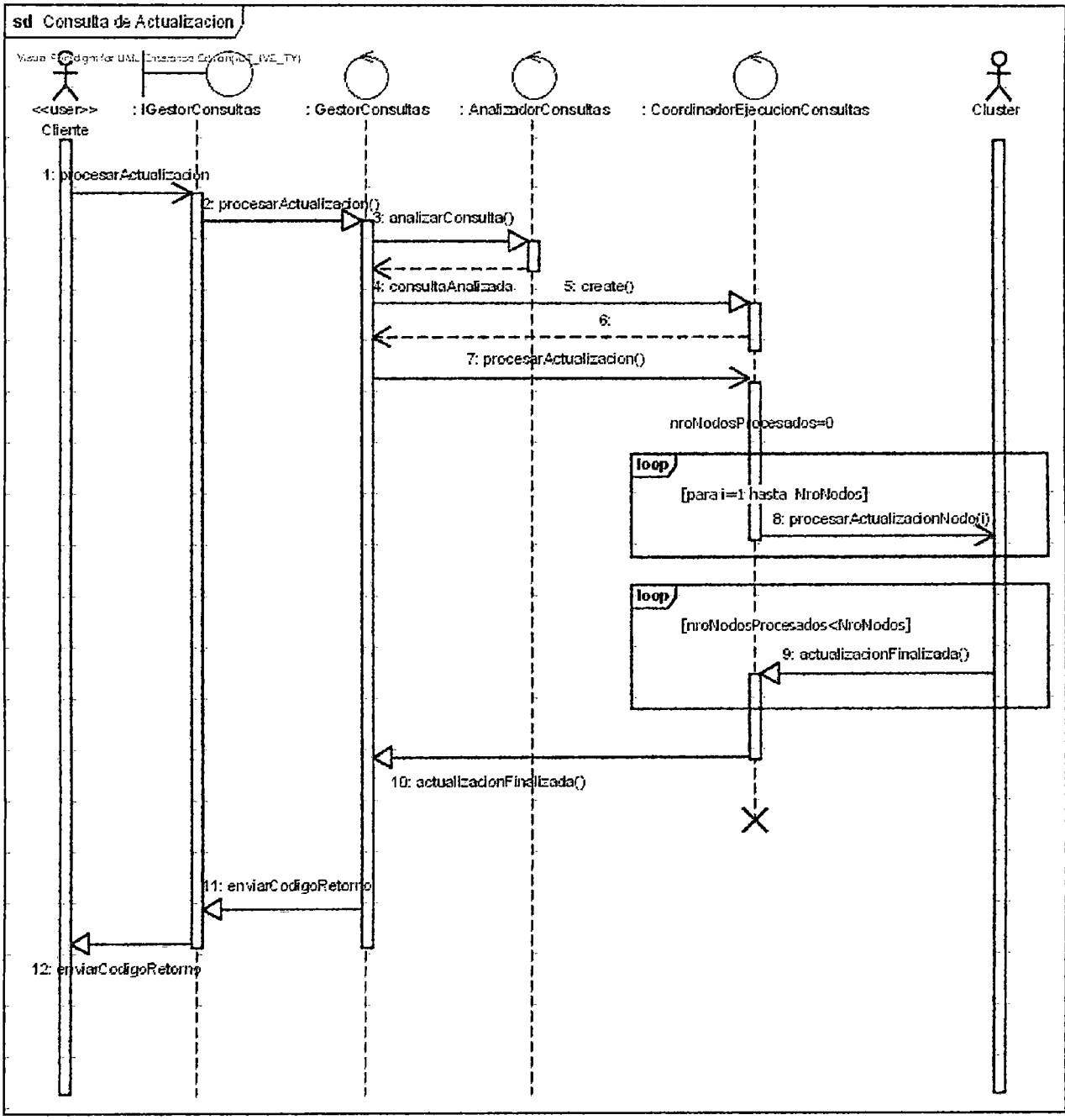


Figura 6-9: Diagrama de Secuencia Procesar Actualización

6.1.4.2. Gestor de Consultas: Ejecutar Actualización en el Nodo

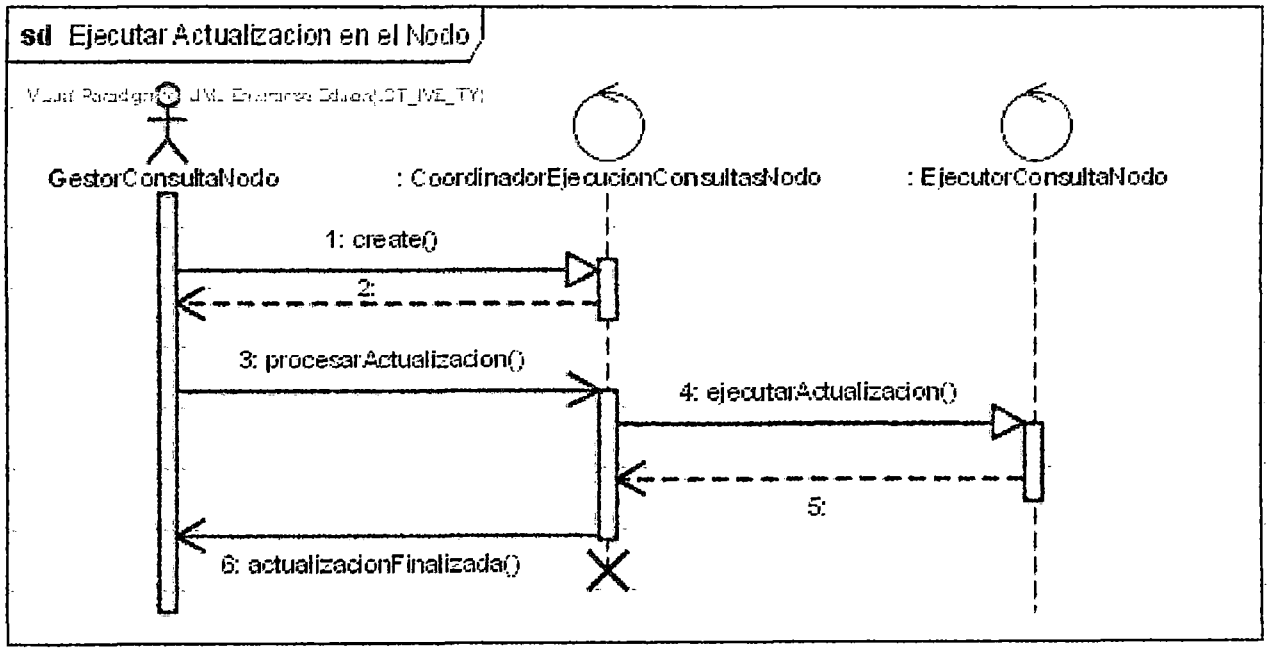


Figura 6-10: Diagrama de Secuencia Ejecutar Actualización en el Nodo

6.1.4.3. Gestor de Consultas: Procesar Petición

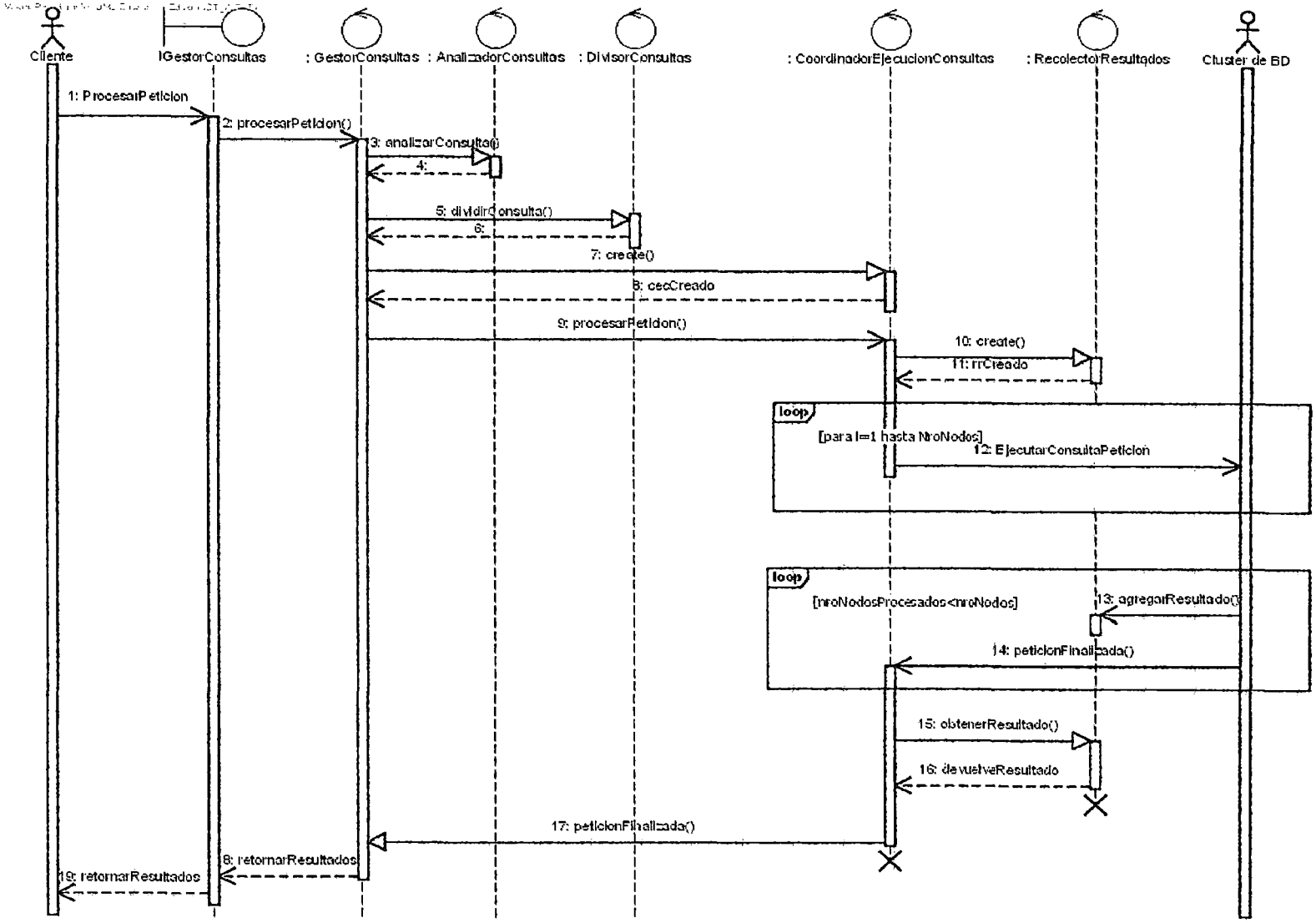


Figura 6-11: Diagrama de Secuencia Procesar Petición

6.1.4.4. Gestor de Consultas: Ejecutar Petición en el Nodo

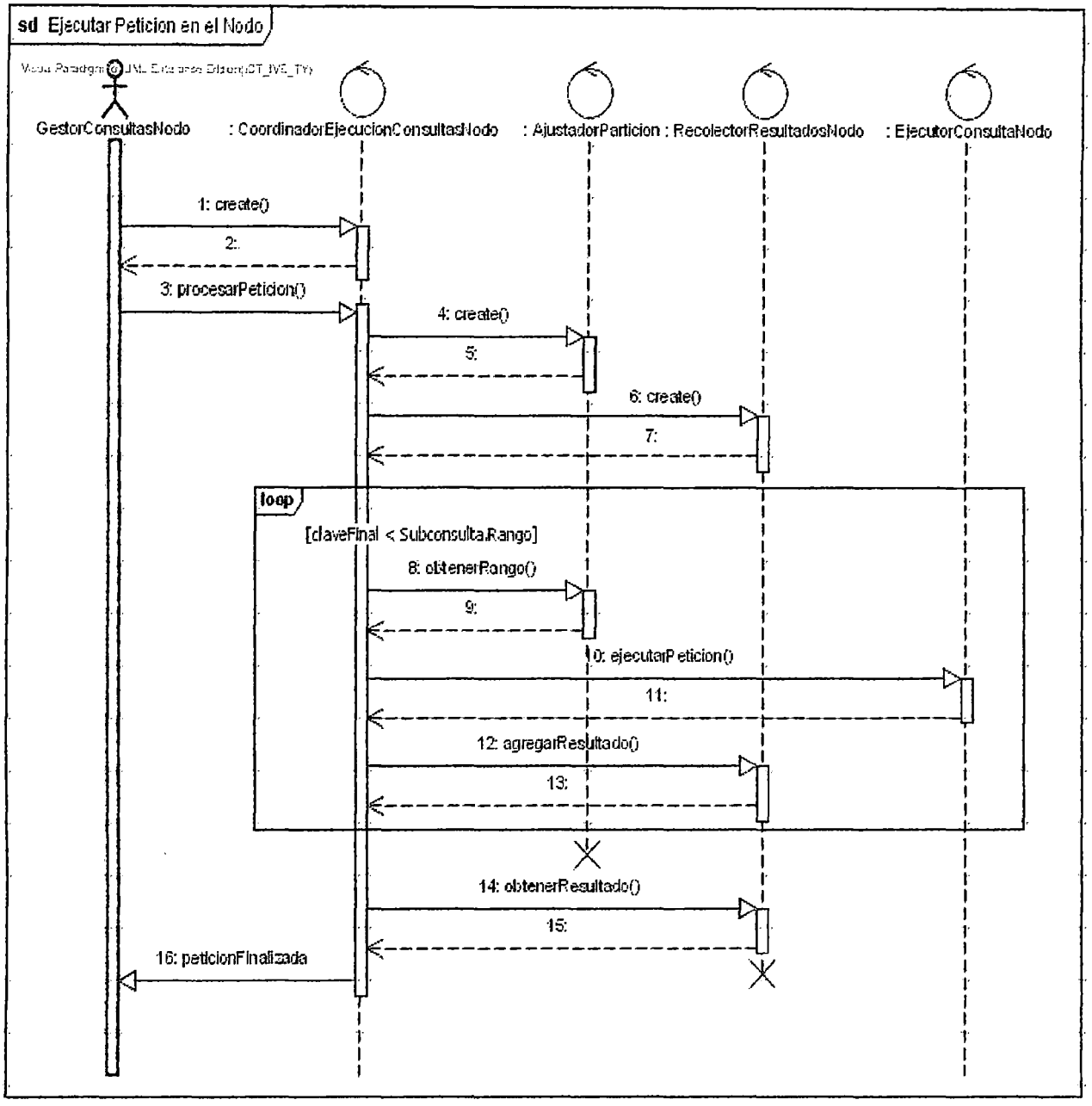


Figura 6-12: Diagrama de Secuencia Ejecutar Petición en el Nodo

6.1.4.5. Gestor de Nodos: AgregarNodo

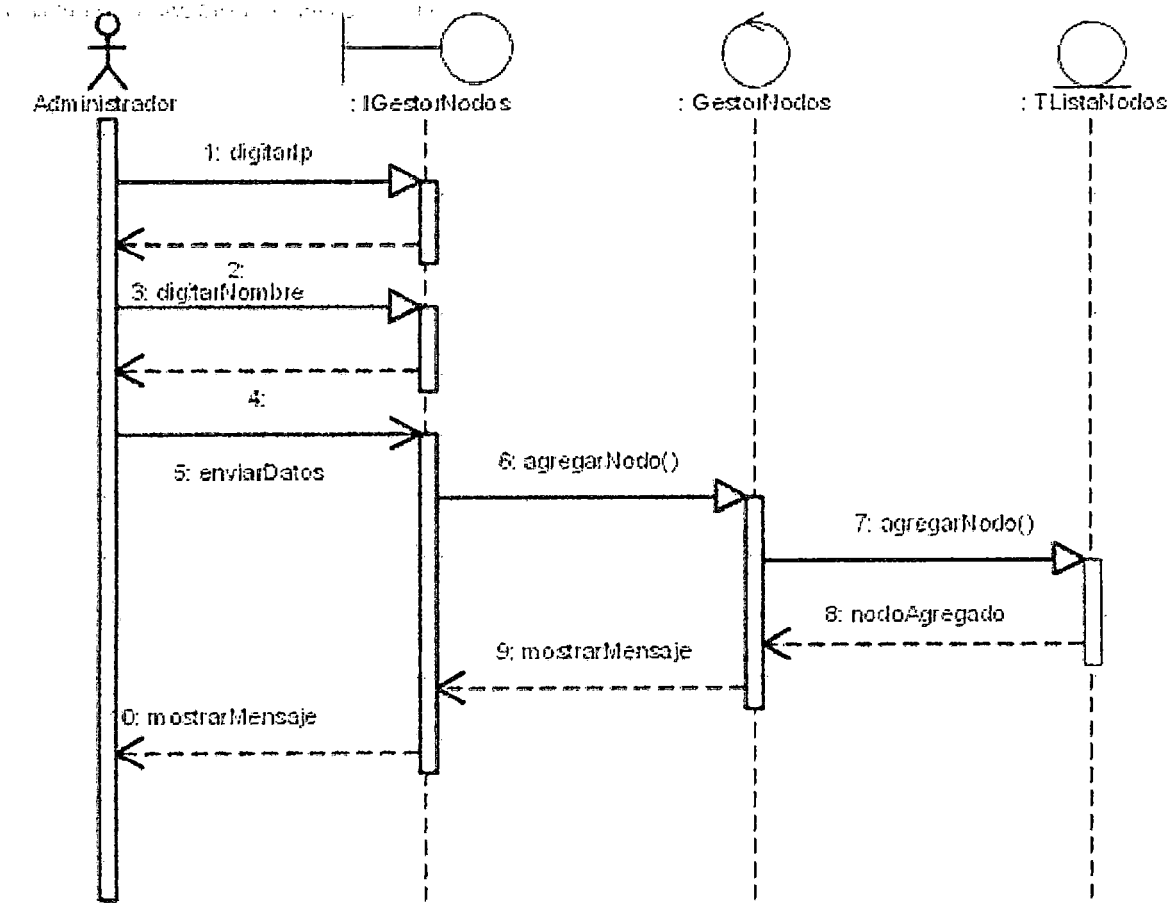


Figura 6-13 : Diagrama de Secuencia Agregar Nodo

6.1.4.6. Gestor de Nodos: QuitarNodo

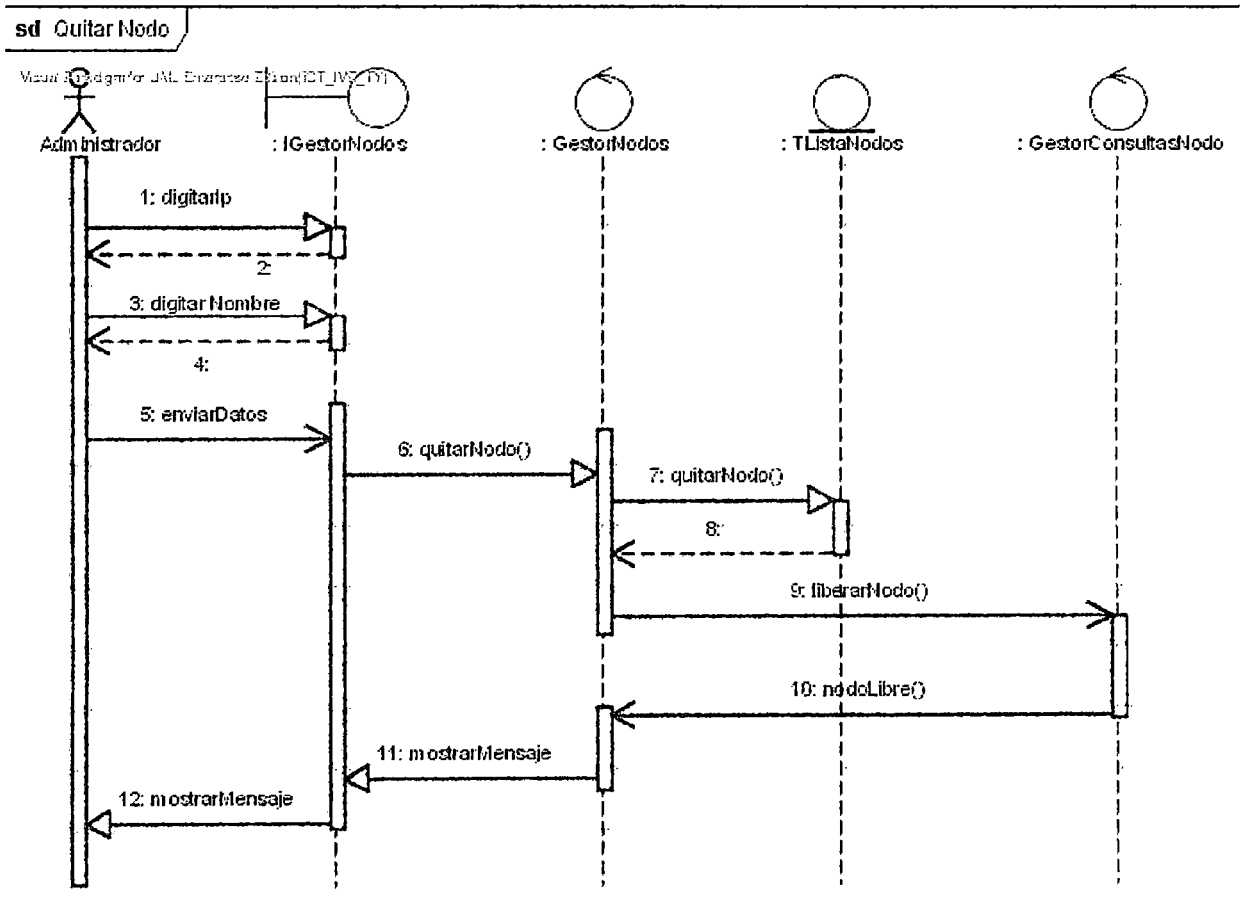


Figura 6-14: Diagrama de Secuencia Quitar Nodo

6.1.5. Diagramas de Comunicación

6.1.5.1. Gestor de Consultas: Procesar Actualización

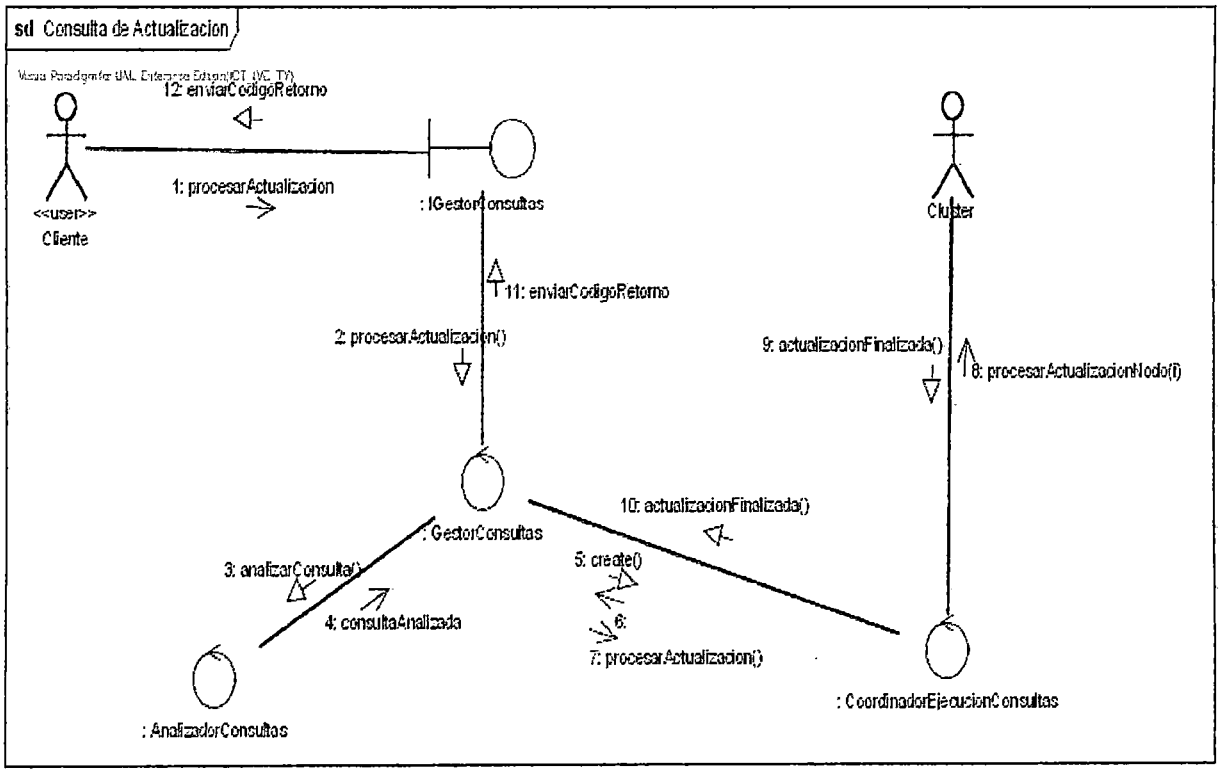


Figura 6-15: Diagrama de Comunicación Procesar Actualización

6.1.5.2. Gestor de Consultas: Ejecutar Actualización en el Nodo

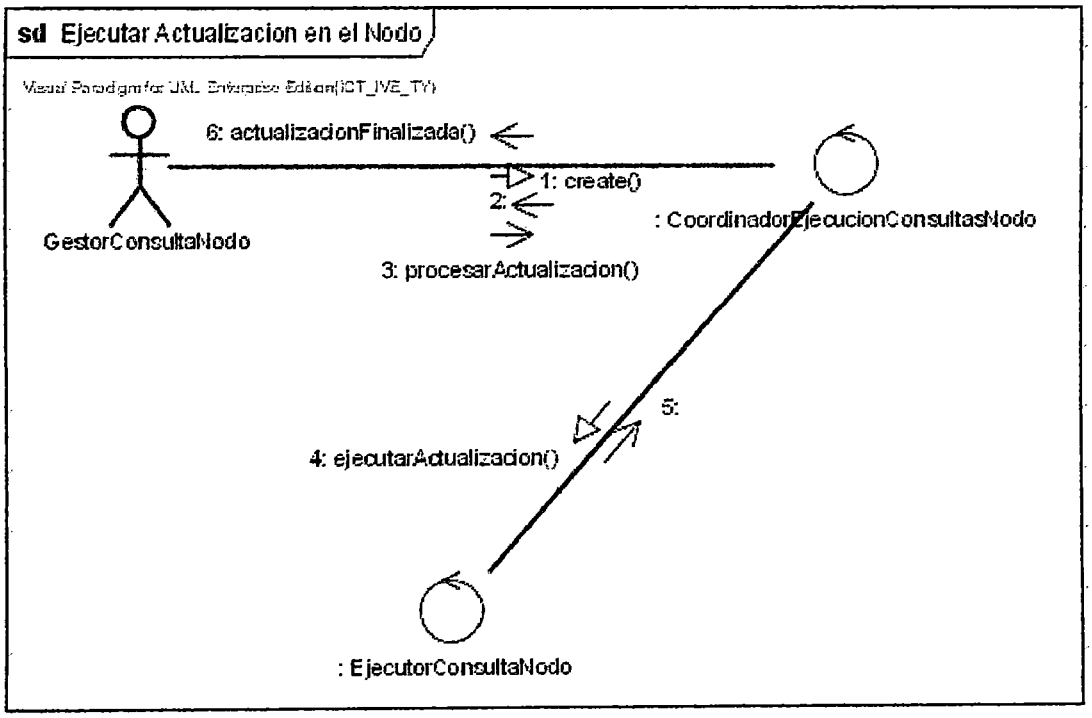


Figura 6-16: Diagrama de Comunicación Ejecutar Actualización en el Nodo

6.1.5.3. Gestor de Consultas: Procesar Petición

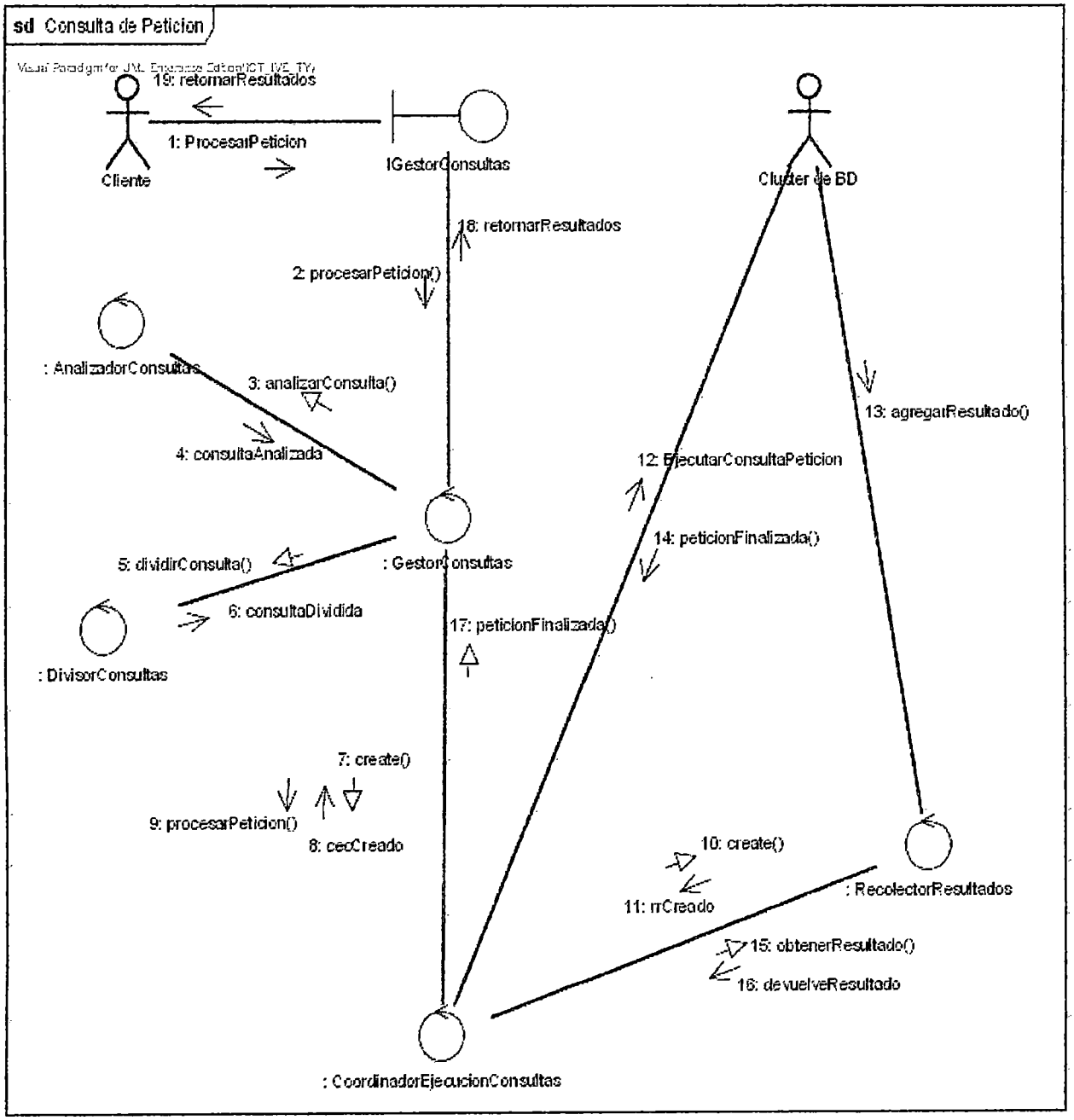


Figura 6-17: Diagrama de Comunicación Procesar Petición

6.1.5.4. Gestor de Consultas: Ejecutar Petición en el Nodo

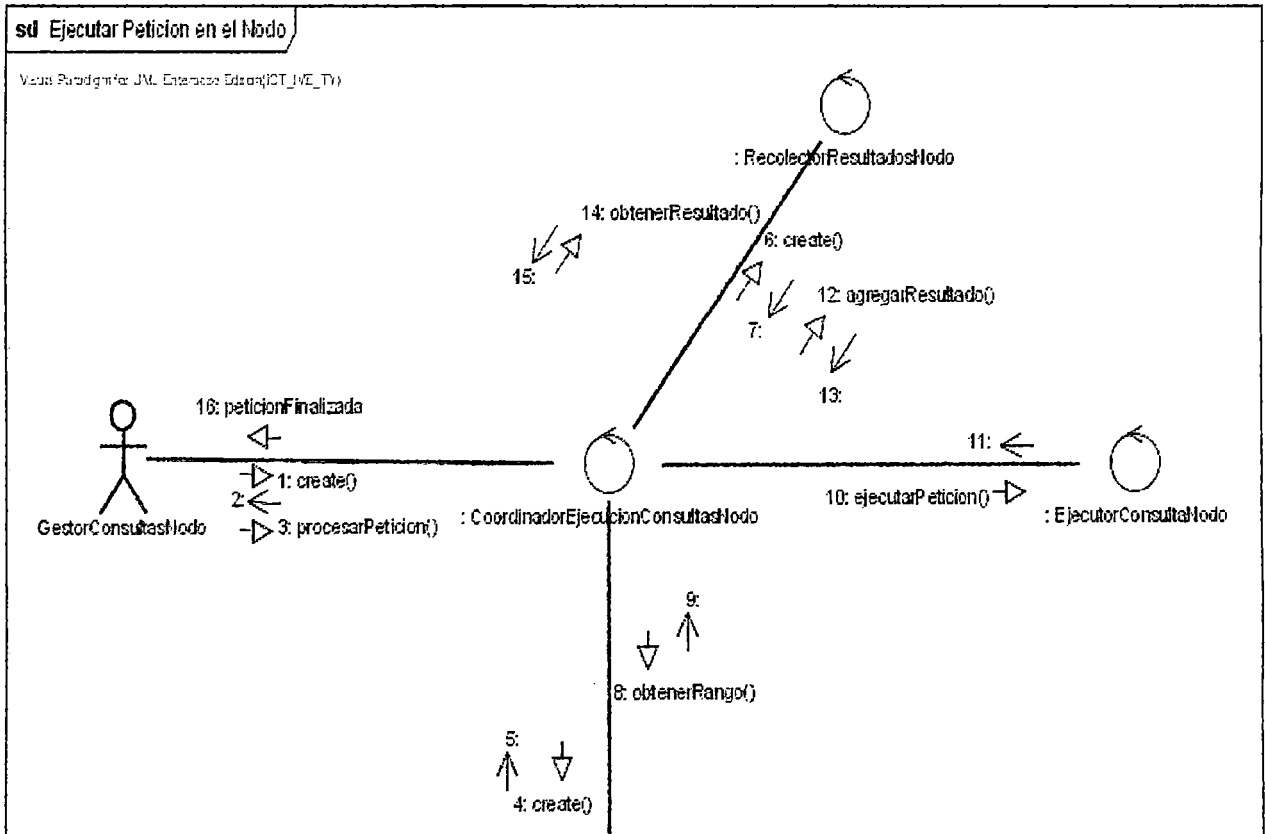


Figura 6-18: Diagrama de Comunicación Ejecutar Petición en el Nodo

6.1.5.5. Gestor de Nodos: AgregarNodo

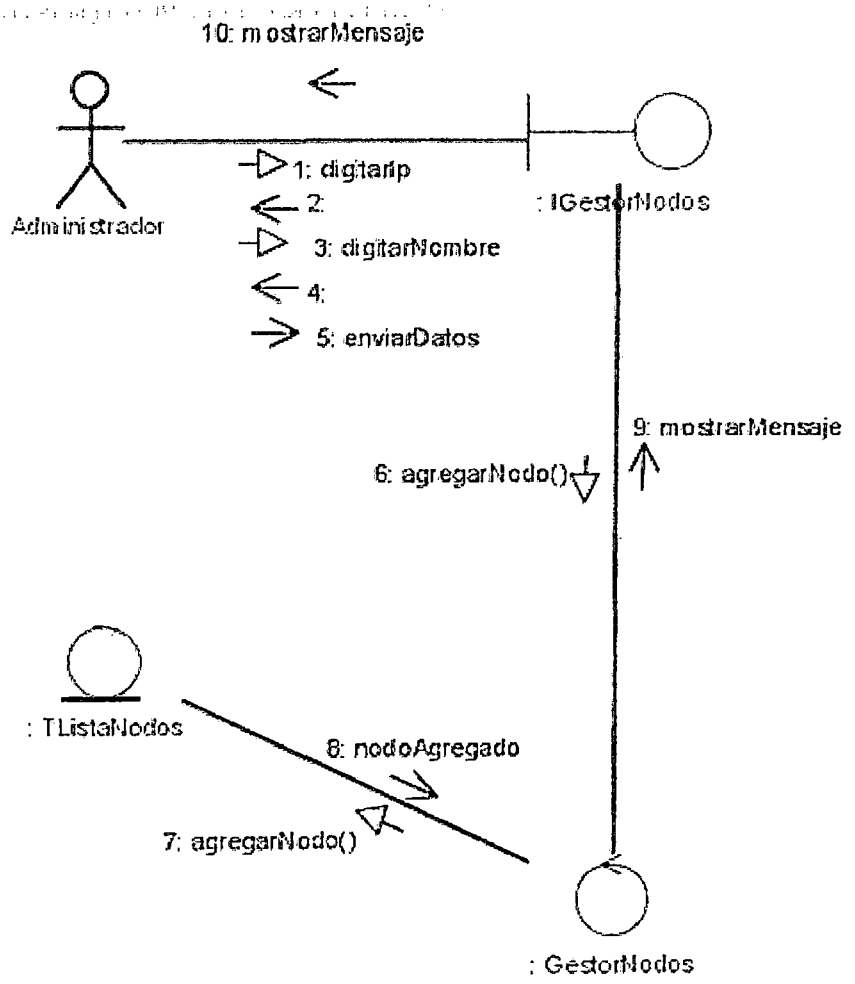


Figura 6-19: Diagrama de Comunicación Agregar Nodo

6.1.5.6. Gestor de Nodos: QuitarNodo

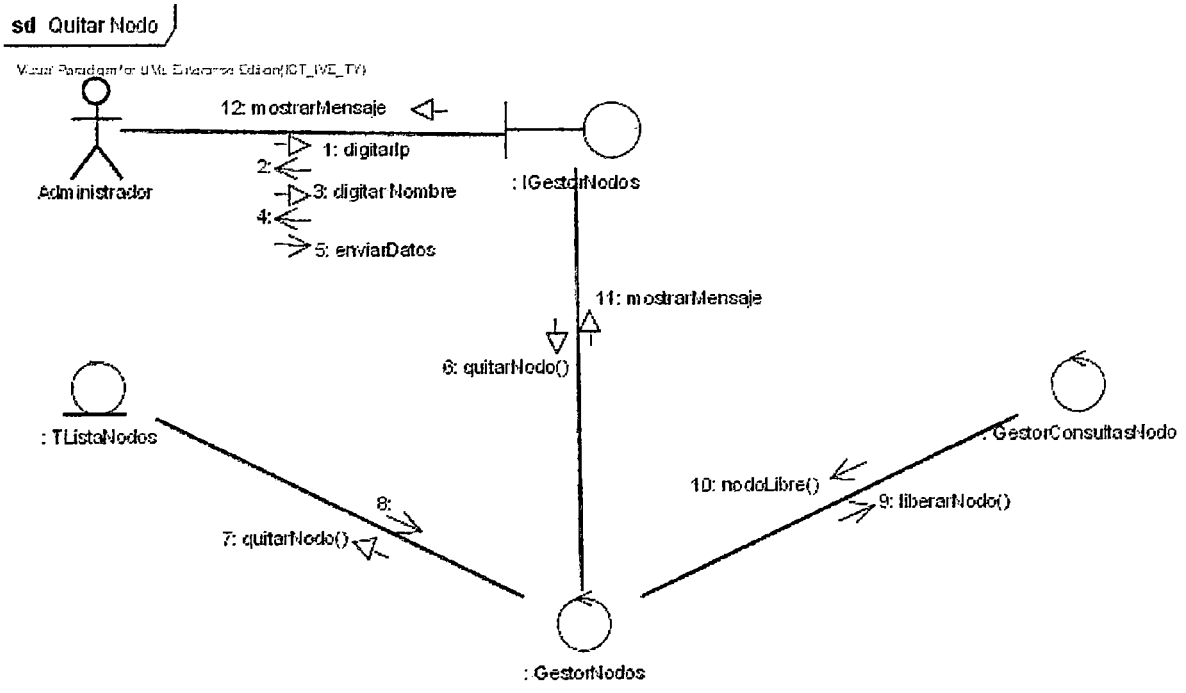


Figura 6-20: Diagrama de Comunicación Quitar Nodo

6.1.6. Casos de Uso Expandidos

Los casos de uso más importantes y los que más influencian al resto, son descritos a un nivel más detallado, en el formato expandido:

6.1.6.1. Caso de Uso: Procesar Actualización

Caso de Uso: Procesar Actualización					
Propósito: Este caso de uso comienza cuando el cliente envía una consulta, y esta es recepcionada por el Software. El Software analiza la consulta para extraer sus metadatos y propaga la actualización entre todos los nodos del cluster. Cada nodo del cluster ejecuta la actualización. El Software retorna mensaje de éxito al cliente.					
Pre condición: { El nodo cuenta con una base de datos replicada de la base de datos principal }					
Post condición: { Todos los nodos se encuentran sincronizados con la base de datos principal }					
n	**Flujo principal de eventos actor**	**Flujo principal de eventos sistema**		**Extensiones**	**Excepciones**
1	Envía una consulta de actualización (INSERT).				
2		Analiza la consulta para extraer sus metadatos.			
3		Propaga la actualización entre todos los nodos del cluster.	a		Mensaje de error: Error al propagar la actualización.
4		Ejecuta la actualización en cada nodo.	a		Mensaje de error: Error al ejecutar la actualización en algún nodo o en todos.
5		Retorna mensaje de éxito al cliente.			

6.1.6.2. Caso de Uso: Procesar Petición

Caso de Uso: Procesar Petición					
Propósito: Este caso de uso comienza cuando el cliente envía una consulta de petición (SELECT) y esta es recepcionada por el Software. Software analiza la consulta y subdivide verticalmente la consulta, luego ejecuta las subconsultas en el cluster de BD y retorna resultados.					
Pre condición: { El nodo cuenta con una base de datos replicada de la base de datos principal }					
Post condición: { Todos los nodos se encuentran sincronizados con la base de datos principal }					
n	**Flujo principal de eventos actor**	**Flujo principal de eventos sistema**		**Extensiones**	**Excepciones**
1	envía una consulta de petición (SELECT).				
2		Analiza la consulta para extraer sus metadatos.			
3		subdivide la consulta verticalmente	a		Error al subdividir: El Software envía un mensaje de error
4		Asigna las subconsultas a los nodos para su procesamiento concurrente.	a		Error al asignar las subconsultas a los nodos del cluster: El Software envía un mensaje de error
5		El cluster procesa las subconsultas	a		Error al procesar las subconsultas El Software envía muestra un mensaje de error

6		Reúne los resultados parciales retornados por los nodos	a		Error al reunir los resultados: El Software envía un mensaje de error.
7		Retorna el resultado al cliente			

6.1.6.3. Caso de Uso: Agregar Nodo

Caso de Uso: Agregar Nodo					
Propósito: Este caso de uso comienza cuando el administrador adiciona un nodo y actualiza el cluster.					
Pre condición: { El nodo cuenta con una replica de la base de datos principal y un SGBD }					
Post condición: { El nodo se agrego a la lista de nodos del Software }					
n	**Flujo principal de eventos actor**	**Flujo principal de eventos sistema**		**Extensiones**	**Excepciones**
1	Adiciona un nodo al cluster.				
2		Se conecta al nodo	a		El Software no se pudo conectar al nodo El Software envía un mensaje de error al administrador.
4		Actualiza el cluster			
5		Retorna al administrador una notificación de éxito.			

6.1.6.4. Caso de Uso: Quitar Nodo

Caso de Uso: Quitar Nodo					
Propósito: Este caso de uso comienza cuando el administrador quita un nodo del cluster. El Software quita el nodo de su lista de nodos. El Software actualiza el cluster.					
Pre condición: { El nodo se encuentra en la lista de nodos del Software }					
Post condición: { El nodo fue eliminado de la lista de nodos del Software }					
n	**Flujo principal de eventos actor**	**Flujo principal de eventos sistema**		**Extensiones**	**Excepciones**
1	Quita un nodo del cluster				
2		Quita el nodo de su lista de nodos			
3		Espera a que el nodo termine su carga de trabajo			
4		Se desconecta del nodo			
5		Retorna al administrador una notificación de éxito			

6.2. MODELO DE DISEÑO

El modelo del diseño describe la manera en que se concretaran físicamente los casos de usos.

Aquí se muestra los diagramas de clases y de componentes.

6.2.1. Clases del Modelo del Diseño

Clase Interfaz del Sistema: Gestor de Consultas

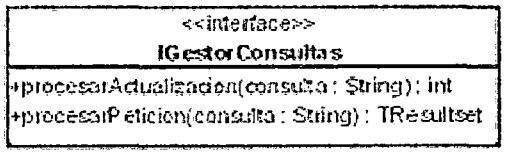


Figura 6-21: Clase Interfaz del Sistema: Gestor de Consultas

Clase Interfaz del Sistema: Gestor de Consultas

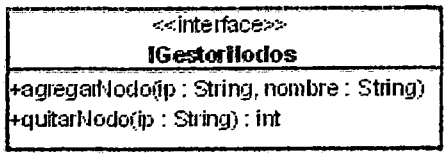


Figura 6-22: Clase Interfaz del Sistema: Gestor de Nodos

Clase Gestor Consultas

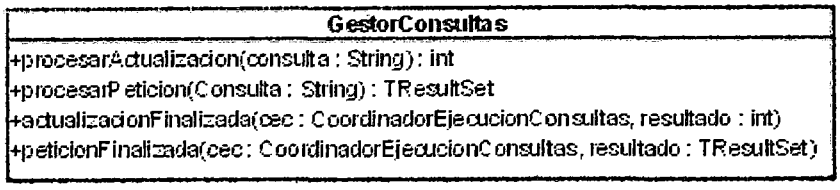


Figura 6-23: Clase Gestor Consultas

Clase Analizador de Consultas

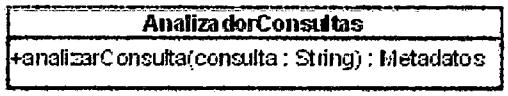


Figura 6-24: Clase Analizador de Consultas

Clase Divisor de Consultas

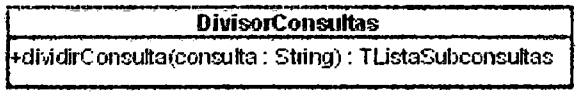


Figura 6-25: Clase Divisor de Consultas

Clase Recolector de Resultados

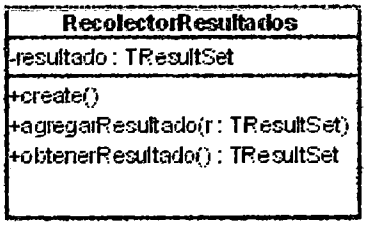


Figura 6-26: Clase Recolector de Resultados

Clase Coordinador de Ejecución de Consultas

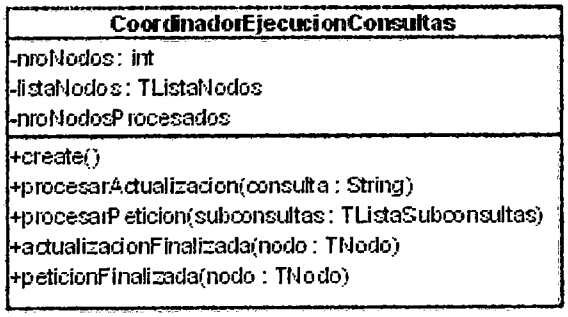


Figura 6-27: Clase Coordinador de Ejecución de Consultas

Clase Gestor de Consultas Nodo

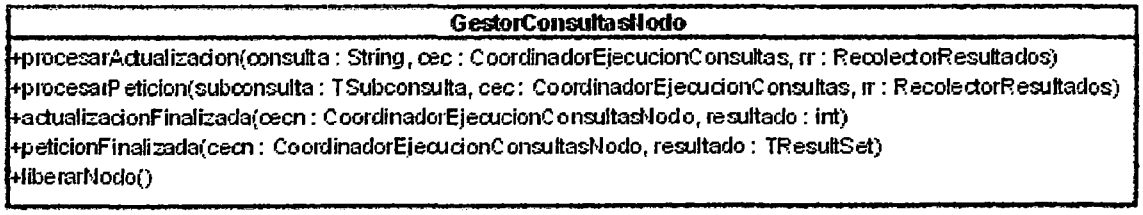


Figura 6-28: Clase Gestor de Consultas Nodo

Clase Coordinador de Ejecución de Consultas en el Nodo

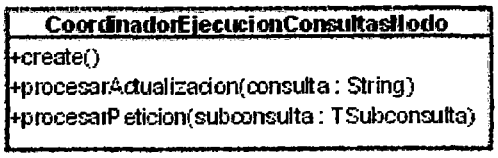


Figura 6-29: Clase Coordinador de Ejecución de Consultas en el Nodo

Clase Ajustador de Partición

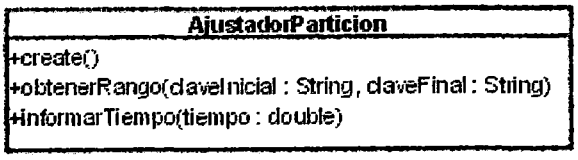


Figura 6-30: Clase Ajustador de Partición

Clase Ejecutor de Consulta en el Nodo

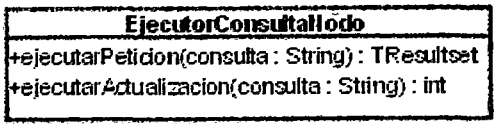


Figura 6-31: Clase Ejecutor de Consulta en el Nodo

Clase Recolector de Resultados en el Nodo

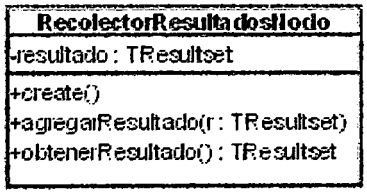


Figura 6-32: Clase Recolector de Resultados en el Nodo

Clase Gestor de Nodos

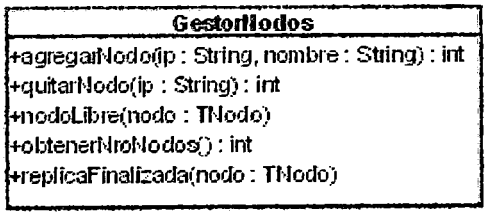


Figura 6-33: Clase Gestor de Nodos

Clase TLista de Nodos

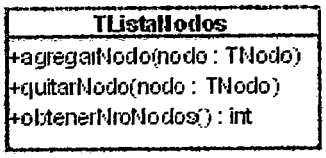


Figura 6-34: Clase TLista de Nodos

Clase TNodo

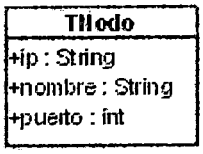


Figura 6-35: Clase TNodo

6.2.2. Diagrama de Clases

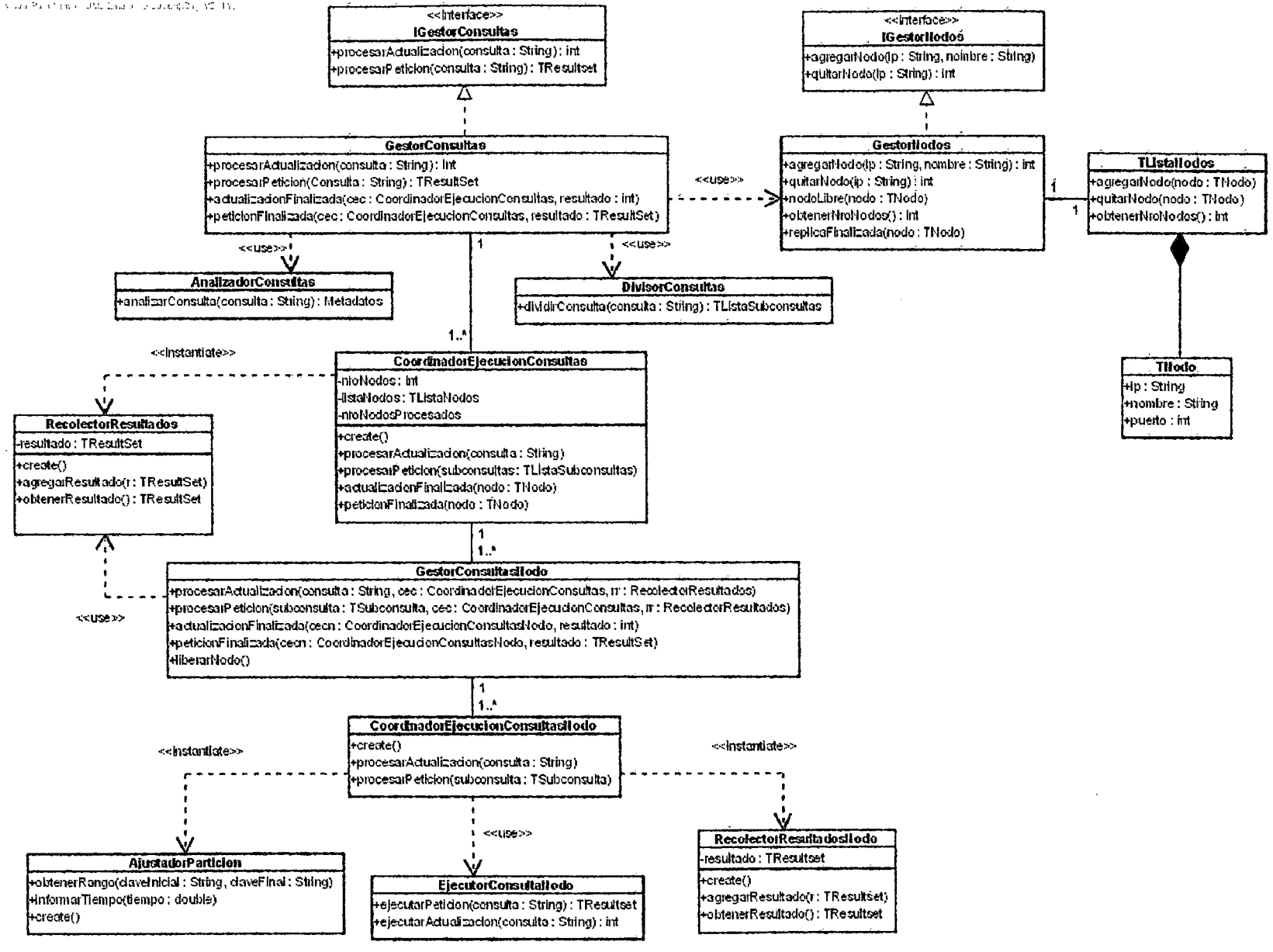


Figura 6-36: Diagrama de Clases

6.2.3. Diagrama de Componentes del Software Black Box

El diagrama de componentes muestra la vista física del sistema, el cual modela la estructura de implementación de la aplicación, su organización en componentes y su despliegue en nodos de ejecución.
El diagrama de componente Black Box muestra las interfaces que el componente provee, las interfaces que requiere y algún otro detalle para explicar su comportamiento.

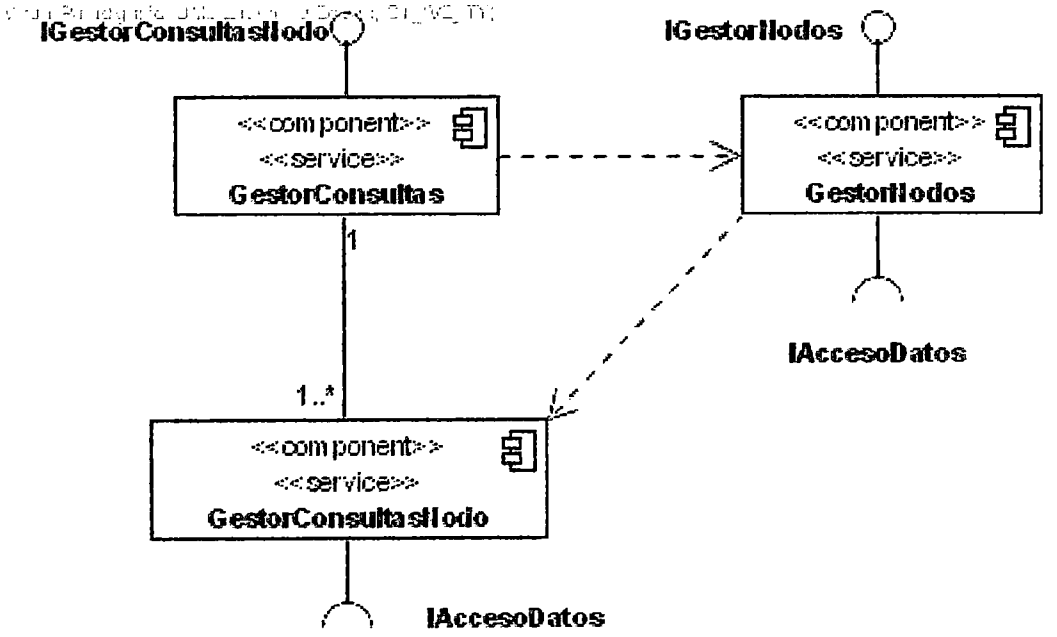


Figura 6-37: Diagrama de Componente del Middleware Black Box

6.2.4. Diagrama de Componentes del Software WhiteBox

El diagrama de componente WhiteBox provee detalles acerca de la implementación , muestra exactamente como un componente realiza las interfaces que provee.

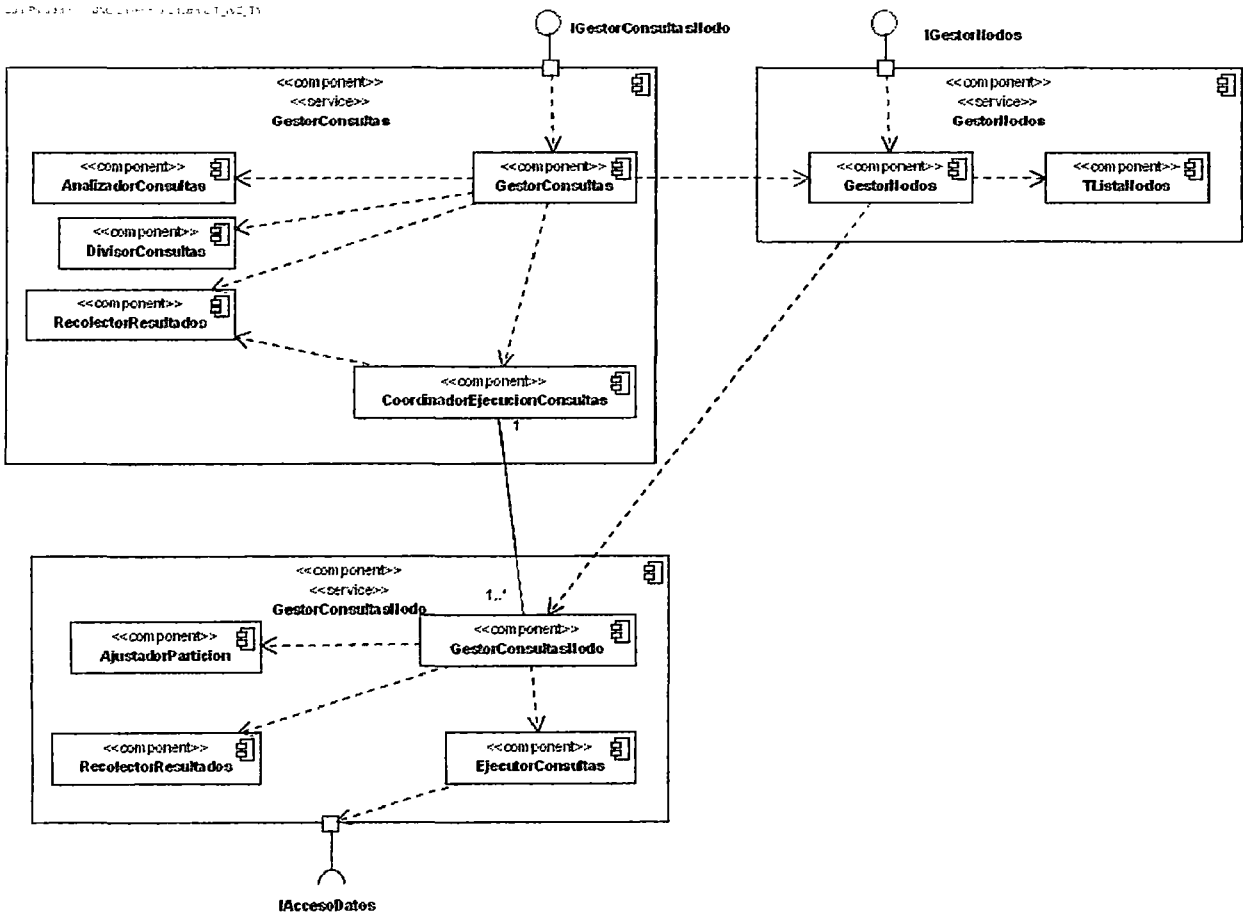


Figura 6-38: Diagrama de Componente del Middleware White Box

6.3. MODELO DE IMPLEMENTACION

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, esto es, código fuente, archivos de configuración, componentes binarios, ejecutables y otros.

6.3.1. Diagrama de Despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución.

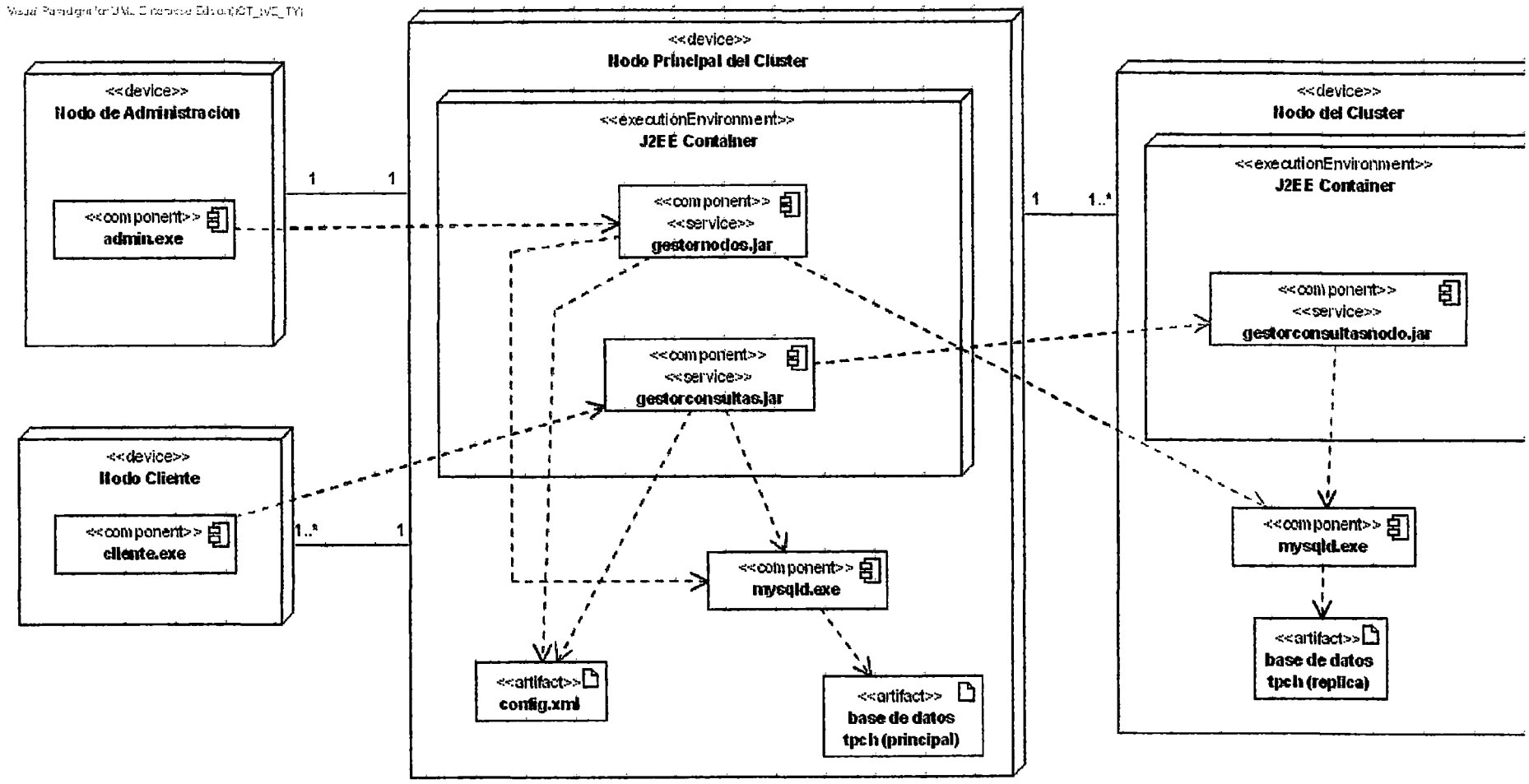


Figura 6-39: Diagrama de Despliegue

CAPITULO VII

DEMOSTRACION EXPERIMENTAL

Finalmente se realiza las pruebas experimentales del prototipo desarrollado en los capítulos previos, pruebas que verifican los resultados obtenidos en la demostración teórica (capitulo 4).

7.1. DEMOSTRACION EXPERIMENTAL DE LA HIPOTESIS

Para esta prueba y posteriores se utilizo:

- Como Software de prueba se usara el prototipo desarrollado en el capitulo anterior, el cual implementa el algoritmo de división vertical de consultas.
- Para enviar consultas y visualizar los resultados y otra información se utilizara una aplicación desarrollada en java "Interfaz de administración del Software" que se conecta al Software mediante JDBC.
- Como base de datos la proporcionada por el TPC-H que se encuentra definida en el capitulo 2.6 con un factor de escala 1 (1 GB aprox).
- Como SGBD se usara MySQL Server 5.1
- Cada nodo contara con una instancia del SGBD y una replica de la Base de Datos.

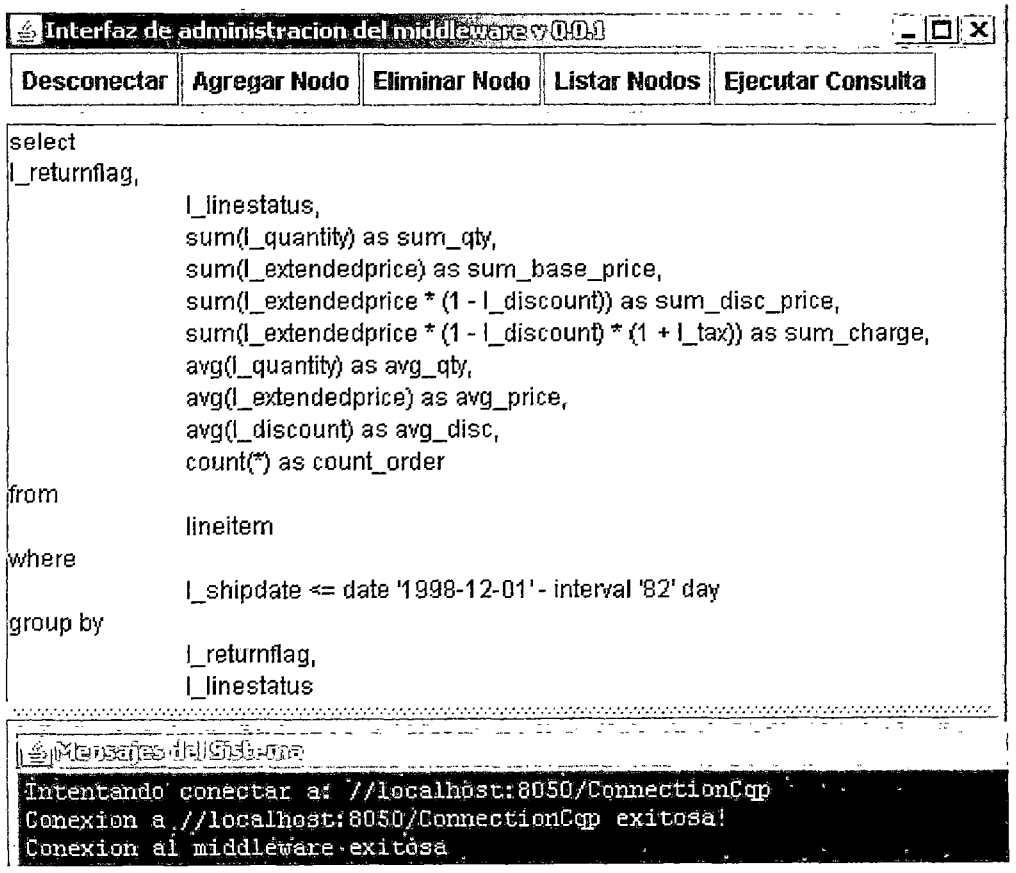
- El campo o atributo a particionar será "hidecolum" (A') que es un campo autoincremental, indexado y con valores únicos.
- La consulta de prueba será la siguiente: Q_0

```
select
        l_returnflag,
        l_linestatus,
        sum(l_quantity) as sum_qty,
        sum(l_extendedprice) as sum_base_price,
        sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
        sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
        avg(l_quantity) as avg_qty,
        avg(l_extendedprice) as avg_price,
        avg(l_discount) as avg_disc,
        count(*) as count_order
from
        lineitem
where
        l_shipdate <= date '1998-12-01' - interval '82' day
group by
        l_returnflag,
        l_linestatus
```

La siguiente pantalla muestra la interfaz desarrollada para interactuar con el Software desarrollado, que tiene las funciones básicas, "Conectar" que permite conectarse o desconectarse del Software, "Agregar Nodo" que sirve para agregar un nodo al cluster, "Eliminar Nodo" que sirve para eliminar un nodo del cluster, "Listar Nodos" que sirve para visualizar la lista de nodos

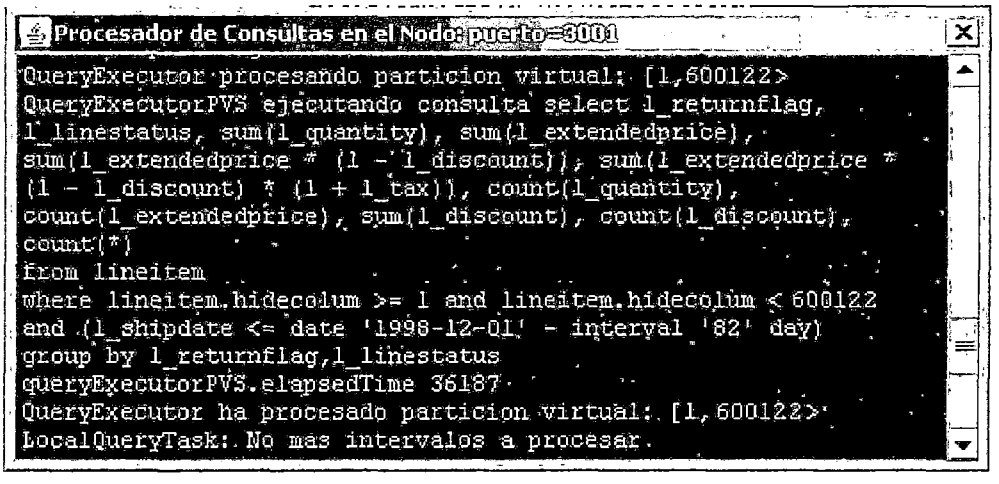
disponibles, y finalmente "Ejecutar Consulta", que permite ejecutar la consulta que esta escrita en la ventana.

Se escribe la consulta de Prueba y se procede a ejecutar.

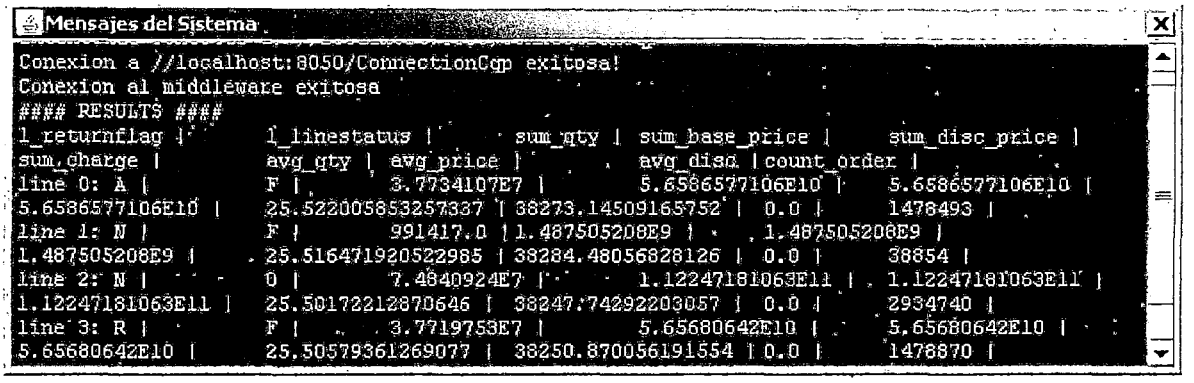


Esta ventana permite visualizar los mensajes del Nodo Principal, en el cual muestra que ha recibido la consulta, ejecuta la consulta con División vertical, muestra que divide la carga entre los 10 nodos, e inicializa los hilos para comenzar a procesar.

Esta ventana muestra los mensajes que se visualizan en uno de los nodos, para este caso nos conectamos al nodo 1, en el cual se visualiza que recibió la consulta mas la condición que procese solo el intervalo deseado, así cada nodo recibirá su correspondiente query procesando solo el intervalo que le fue indicado por el nodo principal.



- Una vez concluido el procesamiento por parte de todos los nodos, el nodo principal reúne los resultados y nos muestra el resultado final del query.



7.2. EXPERIMENTACION

Ahora se procede a realizar la toma de muestras para poder graficar el comportamiento de la curva del tiempo, según la variación del numero de nodos, esto se realiza utilizando el prototipo desarrollado y la siguiente consulta de prueba

Consulta de prueba: SELECT count(*) FROM lineitem

El muestreo se hizo comenzando por usar un solo nodo, se efectúa 5 ejecuciones de la consulta se descarto el tiempo mas alto y el tiempo mas bajo, luego se anotaron los otros 3 tiempos restantes y se saco la media, luego se realizara los mismos pasos con 2, 3, ... y hasta los 10 nodos, los resultados obtenidos se pueden apreciar en la siguiente tabla:

Nro. Nodos	t1	t2	t3	Media
1	3312	3360	3391	**3354.33**
2	2407	2422	2500	**2443.00**
3	1750	1656	1672	**1692.67**
4	1312	1312	1313	**1312.33**
5	1109	1125	1110	**1114.67**
6	1078	1078	1079	**1078.33**
7	1016	969	1000	**995.00**
8	953	954	906	**937.67**
9	844	897	860	**867.00**
10	828	859	813	**833.33**

Tabla 7-1: Tiempos de Ejecución de la Consulta

Tomando como base los valores de la Media de los tiempos obtenidos del muestreo, se procede a graficar la siguiente curva.

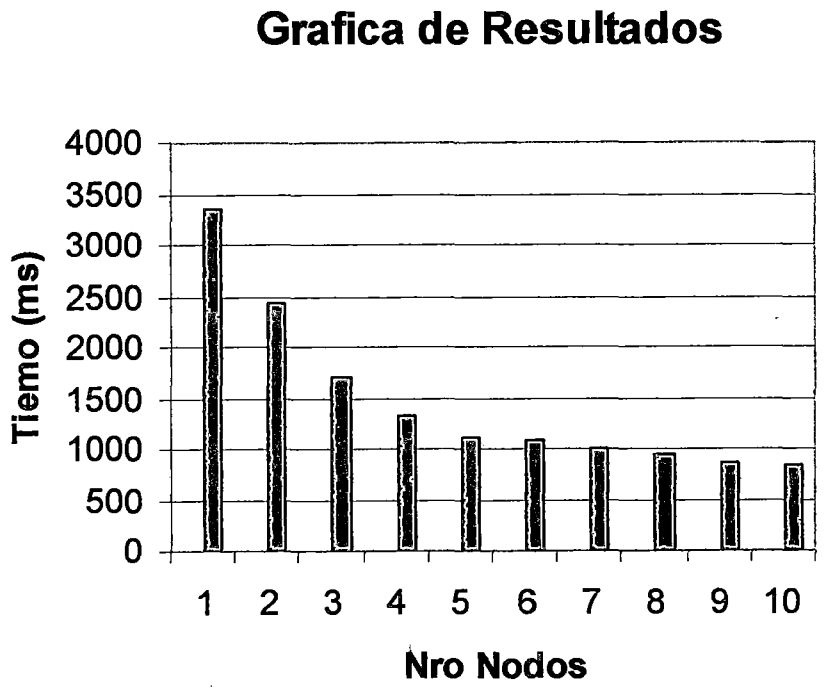


Figura 7-1: Grafica de Resultados

Como se aprecia el comportamiento de esta curva es similar al obtenido teóricamente en la Conclusión del Capitulo 4. Lo cual confirma la validez de la hipótesis planteada.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1.- Se diseño un algoritmo de división vertical de consultas basándose en toda la información referente a división de consultas, que fue recopilada y revisada en la presente investigación. Cabe señalar que no significa que este algoritmo sea el mejor, así como tampoco significa que no se puedan desarrollar otros.

2.- Se definió formalmente el algoritmo de división vertical de consultas propuesto, y se procedió a realizar el análisis teórico de la hipótesis, basándose ello en las graficas del comportamiento del algoritmo frente a la variación de sus variables independientes. Concluyendo que el tiempo de procesamiento del algoritmo es inversamente proporcional al número de nodos del cluster, es decir, que a mayor número de nodos en el cluster menor será el tiempo de procesamiento de las consultas, pero esto no puede disminuir indefinidamente; observándose que existen límites al máximo número de nodos, a partir del cual el rendimiento empieza a disminuir. Se observa de igual manera que cuanto menos registros tienen las tablas involucradas en la consulta el rendimiento global del algoritmo disminuye, lo cual significa que el uso de la división vertical es mas adecuado para consultas que involucran gran cantidad de data a procesar.

3.- Por otro lado se ha desarrollado la Arquitectura de un Software que subdivide las consultas dirigidas a un SGBD, para ser procesadas paralelamente en un cluster de computadoras; el cual puede ser usado como punto de partida para futuros trabajos de este tipo, orientados al procesamiento paralelo de consultas SQL; cabe señalar en este punto que la Arquitectura del Software no limita ni restringe al uso de tal o cual algoritmo de división de consultas, mas bien permite tener una visión global del software.

4.- Tomando como base la Arquitectura ya elaborada, se diseño un prototipo de Software que implemente el algoritmo de división vertical de consultas propuesto, para ello se aplicó la Metodología RUP, y se elaboró los artefactos propuestos por dicha metodología, algunos de los cuales son: Diagramas de Casos de Uso, Diagramas de Secuencia, Diagramas de Colaboración, Diagramas de Despliegue, Diagrama de Interacción de Objetos.

5.- Basado en el Diseño del software se implementó con éxito un prototipo, el cual fue escrito en el lenguaje de programación Java y utilizando como Sistema de Gestión de Base de Datos a MySQL 5.0. Se hace énfasis que esto no significa que sea el único SGBD con el que el Software pueda interactuar. Este software es para todos los efectos otorgado a la Comunidad como Software Libre, bajo la Licencia Pública General GNU.

6.- Se efectuaron las pruebas experimentales con el prototipo del Software y utilizando un cluster de Computadoras de 10 nodos. De la medición de los tiempos de ejecución de una misma consulta procesándose con diferente

cantidad de nodos, se hace evidente que el rendimiento del Algoritmo es directamente proporcional al número de nodos del cluster, es decir a mayor numero de nodos mas rápido es el procesamiento de la consulta, Aquí cabe señalar que por las limitaciones físicas, en este caso el numero de nodos del cluster, y debido al gran numero de registros que evalúa esta consulta (1 000 000), no se pudo establecer si existía algún limite en el numero de nodos a partir del cual el rendimiento podría empezar a disminuir. En la Demostración Teórica se pudo observar que existían límites al crecimiento de nodos, y según la formula hallada, para esta consulta se requerirían de un número bastante elevado de nodos, muy por encima de los 10 que se usaron en esta prueba.

7.- Al final comparando los resultados obtenidos en la Prueba Experimental y de lo resultados de la Demostración Teórica, se logra demostrar que "El tiempo de respuesta de una consulta con complejidad alta usando división vertical es menor que el tiempo de respuesta de la misma consulta usando división horizontal", que era la hipótesis planteada en esta investigación y se desarrolló el Software para procesamiento paralelo de consultas, el cual era el objetivo general de la Investigación.

RECOMENDACIONES

A la conclusión de este trabajo de tesis se recomienda:

1.- Continuar con las investigaciones sobre los temas tratados en este trabajo de tesis, especialmente en lo que respecta al procesamiento paralelo y buscar nuevas formas de aprovechar los cluster de computadores.

2.- Diseñar e implementar nuevos y mejores algoritmos de división de consultas, persiguiendo el objetivo de disminuir los tiempos de respuesta de la ejecución de consultas de alta complejidad.

3. Desarrollar otras arquitecturas de software para el manejo de clusters de computadoras.

4.- Realizar estudios matemáticos y estadísticos con el algoritmo y el prototipo para encontrar con mayor aproximación el punto de inflexión de la curva de rendimiento del algoritmo. Para ello se deben incluir más variables, como por ejemplo: la latencia de la red, la velocidad del procesador, la cantidad de Memoria del Computador, el costo de lectura escritura en la memoria.

REFERENCIAS BIBLIOGRAFIAS

[AAS05] Área de Administración de Servidores, "Estándares ANSI SQL", Universidad Nacional Autónoma de México. [citado 17 de septiembre del 2008] disponible en: <http://sunsite.unam.mx/servidores/docs/bd/ANSI%20SQL.pdf>

[ARI02] Ariannejad. "Trends in Software Systems". Computer Engineering Research Group. [citado 25 de Junio del 2008] disponible en: <http://www.eecg.toronto.edu/~jacobsen/courses/mwci/scribes/scribe1/scribe1.htm>

[BCK98] Bass Len, Clements Paul y Kazman Rick; "Software Architecture in Practice". Reading, Addison-Wesley. Publicado 2003 Segunda Edición (560 Pags) [citado 17 de septiembre del 2007] disponible en: <http://books.google.com.pe/books?id=mdilu8Kk1WMC&dq=Software+Architecture+in+Practice>

[BCLMO06] Becerra Mariana, Canumán José, Laguía Daniel, Marín Mauricio, Osiris Sofía; Universidad de Magallanes, Chile; "Procesamiento paralelo de consultas SQL generadas desde la Web" Agosto del 2004 [citado 20 de Septiembre del 2008] disponible en: <www.dcc.uchile.cl/~mmarin/papers/wsdp01.ps.gz>

[BEL06] Bell Gordon, "Arquitecturas Avanzadas", The Berkeley NOW Project, [citado 10 de Enero del 2008] disponible en: <http://now.cs.berkeley.edu/>

[BUD08] Burleson Donald K. "Oracle9i RAC Tips", [citado 01 de Febrero del 2008] disponible en: <http://www.remote-dba.cc/teas_aegis_rac17.htm>

[BUR06] Burbano Proaño, Diego Javier; " Análisis Comparativo de Bases de Datos de Código Abierto VS Código Cerrado

(Determinación de Índices de Comparación)". [citado 15 de Noviembre del 2007] disponible en: <http://www.mysql-hispano.org/articulos/num43/analisis-comparativo.pdf>

[CAR06] Carrillo Ledesma, Antonio; "Aplicaciones del Cómputo en Paralelo a la Modelación de Sistemas Terrestres", Tesis. [citado 15 de Enero del 2008] disponible en: <http://mmc.igeofcu.unam.mx/acl/Archivos/Tesis/PresentacionAplicacionesDelComputoEnParaleloALaModelacionDeSistemasTerrestres.pdf>

[CHMP05] Cañizares Ángel Ramón, Hortolano Julián, Moreno Sergio, Moreno Felipe, Pinilla Miguel Ángel, Olmo Ángel Rafael. "Bases de Datos Paralelas y Bases de Datos Grids". Grupo Rendimiento, Ingeniería Superior Informática. [citado 10 de Enero del 2008] disponible en: <http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/RENDIMIENTO.pdf>

[DCA06] De Careaga Alfonso, "La tecnología ORACLE RAC" . [citado 10 de Noviembre del 2007] disponible en: <http://www.protecmedia.com/es/magazine/Magazine_Sep06/4noticia.htm>

[DCCAI01] Departamento de Control de Calidad y Auditoria Informática, Sistemas en Arquitectura Cliente/Servidor. [citado 10 de Enero del 2008] disponible en: <http://sistemas.dgsca.unam.mx/publica/pdf/clienteservidor.PDF>

[GNU99] GNU (GNU is Not GNU), Free Software Foundation. [citado 10 de Enero del 2009] disponible en: <http://www.gnu.org/licenses/licenses.es.html>

[GW04] Gray Jim, Microsoft Research y Waymire Richard, SQL Server Development, "Megaservidores SQL Server: escalabilidad, disponibilidad y capacidad de administración". [citado 05 de Diciembre del 2007] disponible en: <http://www.microsoft.com/latam/technet/articulos/200312/art01/default.asp>

[HER06] Hernández Suárez, José de Jesús; "Arquitectura de Software Importancia de su ciclo de vida"; [citado 10 de Enero del 2008] disponible en: <http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm>

[KRU95] Kruchten P., Rational Software Corp. IEEE Software 12 (6); Architectural Blueprints—The "4+1" View Model of Software Architecture. [citado 10 de Agosto del 2008] disponible en: <http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>

[MySQL09] MySQL; "MySQL Cluster 7.0: Architecture and New Features". . [citado 15 de Agosto del 2008] disponible en: <http://www.mysql.com/why-mysql/white-papers/mysql_wp_cluster7_architecture.php>

[PER02] Pereyra López, Ángel; "Análisis de Estructuras y Replicación de Base de Datos", Escuela técnica Superior de Informática, Universidad de Málaga. . [citado 10 de Octubre del 2008] disponible en: <http://www.informatica.uma.es/ETSIIpub/Pfc/PDF/2002/APL2.pdf>

[PMC06] Pérez María, Mendoza Luis E., Carvajal Yorka, "Orientaciones para la Selección de Tecnologías de Integración de Sistemas de Software", Laboratorio de Investigación en Sistemas de Información (LISI). Departamento de Procesos y Sistemas, Universidad Simón Bolívar. . [citado 10 de Agosto del 2008] disponible en: <http://www.actea.es/C19/Tecnologia/Document%20Library/Integracion%20Sistemas%20Software.pdf>

[RK04] Reynoso, Carlos y Kicillof, Nicolás; "Estilos y Patrones en la Estrategia de la Arquitectura de Microsoft", . [citado 12 de Agosto del 2008] disponible en: http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/lenguaje.asp

[SQS06] Santos Espino, José Miguel - Alexis Quesada Arencibia - Francisco Santana – ULPGC, "Concurrencia. Sincronización y comunicación de procesos", . [citado 10 de Enero del 2009] disponible en: <http://sopa.dis.ulpgc.es/so/material_didactico_teoria/tema6_6transporpagina.pdf>

[TPC-H] TPC BENCHMARK™ H, Transaction Processing Performance Council (TPC) USA, . [citado 05 de Diciembre del 2007] disponible en: <http://www.tpc.org/tpch/>

[UML06] Unified Modeling Language, "UML 2.0, The Current Official Version", Object Management Group. . [citado 10 de Marzo del 2009] disponible en: <http://www.uml.org/#UML2.0>

ANEXOS

ANEXOS A :
LICENCIA PUBLICA GNU (GPL) [GNU99]

NOTA IMPORTANTE:

Esta es una traducción no oficial al español, la versión original en ingles puede ser consultada en el siguiente enlace: http://www.gnu.org/licenses/licenses.es.html.

Preámbulo

Las licencias que cubren la mayor parte del software están diseñadas para quitarle a usted la libertad de compartirlo y modificarlo. Por el contrario, la Licencia Pública General de GNU pretende garantizarle la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software del la Free Software Foundation y a cualquier otro programa si sus autores se comprometen a utilizarla. (Existe otro software de la Free Software Foundation que está cubierto por la Licencia Pública General de GNU para Bibliotecas). Si quiere, también puede aplicarla a sus propios programas.

Cuando hablamos de software libre, estamos refiriéndonos a libertad, no a precio. Nuestras Licencias Públicas Generales están diseñadas para asegurarnos de que tenga la libertad de distribuir copias de software libre (y cobrar por ese servicio si quiere), de que reciba el código fuente o que pueda conseguirlo si lo quiere en nuevos programas libres, y de que sepa que puede hacer todas estas cosas. Para proteger sus derechos necesitamos algunas restricciones que prohíban a cualquiera negarle a usted estos derechos o pedirle que renuncie a ellos. Estas restricciones se traducen en ciertas obligaciones que le afectan si distribuye copias del software, o si lo modifica.

Por ejemplo, si distribuye copias de uno de estos programas, sea gratuitamente, o a cambio de una contraprestación, debe dar a los receptores todos los derechos que tiene. Debe asegurarse de que ellos también reciben, o pueden conseguir, el código fuente. Y debe mostrarles estas condiciones de forma que conozcan sus derechos.

Protegemos sus derechos con la combinación de dos medidas:

Ponemos el software bajo copyright y le ofrecemos esta licencia, que le da permiso legal para copiar, distribuir y/o modificar el software.

También, para la protección de cada autor y la nuestra propia, queremos asegurarnos de que todo el mundo comprende que no se proporciona ninguna garantía para este software libre. Si el software se modifica por cualquiera y éste a su vez lo distribuye, queremos que sus receptores sepan que lo que tienen no es el original, de forma que cualquier problema introducido por otros no afecte a la reputación de los autores originales.

Por último, cualquier programa libre está constantemente amenazado por patentes sobre el software. Queremos evitar el peligro de que los redistribuidores de un programa libre obtengan patentes por su cuenta, convirtiendo de facto el programa en propietario. Para evitar esto, hemos dejado claro que cualquier patente debe ser pedida para el uso libre de cualquiera, o no ser pedida.

Los términos exactos y las condiciones para la copia, distribución y modificación se exponen a continuación.

Términos y condiciones para la copia, distribución y modificación. Esta Licencia se aplica a cualquier programa u otro tipo de trabajo que contenga una nota colocada por el tenedor del copyright diciendo que puede ser distribuido bajo los términos de esta Licencia Pública General. En adelante, «Programa» se referirá a cualquier programa o trabajo que cumpla esa condición y «trabajo basado en el Programa» se referirá bien al Programa o a cualquier trabajo derivado de él según la ley de copyright. Esto es, un trabajo que contenga el programa o una porción de él, bien en forma literal o con modificaciones y/o traducido en otro lenguaje. Por lo tanto, la traducción está incluida sin limitaciones en el término «modificación». Cada concesionario (licenciatario) será denominado «usted».

Cualquier otra actividad que no sea la copia, distribución o modificación no está cubierta por esta Licencia, está fuera de su ámbito. El acto de ejecutar el Programa no está restringido, y los resultados del Programa están cubiertos únicamente si sus contenidos constituyen un trabajo basado en el Programa, independientemente de haberlo producido mediante la ejecución del programa. El que esto se cumpla, depende de lo que haga el programa.

Usted puede copiar y distribuir copias literales del código fuente del Programa, según lo has recibido, en cualquier medio, supuesto que de forma adecuada y bien visible publique en cada copia un anuncio de copyright adecuado y un repudio de

garantía, mantenga intactos todos los anuncios que se refieran a esta Licencia y a la ausencia de garantía, y proporcione a cualquier otro receptor del programa una copia de esta Licencia junto con el Programa.

Puede cobrar un precio por el acto físico de transferir una copia, y puede, según su libre albedrío, ofrecer garantía a cambio de unos honorarios.

Puede modificar su copia o copias del Programa o de cualquier porción de él, formando de esta manera un trabajo basado en el Programa, y copiar y distribuir esa modificación o trabajo bajo los términos del apartado 1, antedicho, supuesto que además cumpla las siguientes condiciones:

Debe hacer que los ficheros modificados lleven anuncios prominentes indicando que los ha cambiado y la fecha de cualquier cambio.

Debe hacer que cualquier trabajo que distribuya o publique y que en todo o en parte contenga o sea derivado del Programa o de cualquier parte de él sea licenciada como un todo, sin carga alguna, a todas las terceras partes y bajo los términos de esta Licencia.

Si el programa modificado lee normalmente órdenes interactivamente cuando es ejecutado, debe hacer que, cuando comience su ejecución para ese uso interactivo de la forma más habitual, muestre o escriba un mensaje que incluya un anuncio de copyright y un anuncio de que no se ofrece ninguna garantía (o por el contrario que sí se ofrece garantía) y que los usuarios pueden redistribuir el programa bajo estas condiciones, e indicando al usuario cómo ver una copia de esta licencia. (Excepción: si el propio programa es interactivo pero normalmente no muestra ese anuncio, no se requiere que su trabajo basado en el Programa muestre ningún anuncio).

Estos requisitos se aplican al trabajo modificado como un todo. Si partes identificables de ese trabajo no son derivadas del Programa, y pueden, razonablemente, ser consideradas trabajos independientes y separados por ellos mismos, entonces esta Licencia y sus términos no se aplican a esas partes cuando sean distribuidas como trabajos separados. Pero cuando distribuya esas mismas secciones como partes de un todo que es un trabajo basado en el Programa, la distribución del todo debe ser según los términos de esta licencia, cuyos permisos para otros licenciatarios se extienden al todo completo, y por lo tanto a todas y cada una de sus partes, con independencia de quién la escribió.

Por lo tanto, no es la intención de este apartado reclamar derechos o desafiar sus derechos sobre trabajos escritos totalmente por usted mismo. El intento es ejercer el derecho a controlar la distribución de trabajos derivados o colectivos basados en el Programa.

Además, el simple hecho de reunir un trabajo no basado en el Programa con el Programa (o con un trabajo basado en el Programa) en un volumen de almacenamiento o en un medio de distribución no hace que dicho trabajo entre dentro del ámbito cubierto por esta Licencia.

Puede copiar y distribuir el Programa (o un trabajo basado en él, según se especifica en el apartado 2, como código objeto o en formato ejecutable según los términos de los apartados 1 y 2, supuesto que además cumpla una de las siguientes condiciones:

Acompañarlo con el código fuente completo correspondiente, en formato electrónico, que debe ser distribuido según se especifica en los apartados 1 y 2 de esta Licencia en un medio habitualmente utilizado para el intercambio de programas, o acompañarlo con una oferta por escrito, válida durante al menos tres años, de proporcionar a cualquier tercera parte una copia completa en formato electrónico del código fuente correspondiente, a un coste no mayor que el de realizar físicamente la distribución del fuente, que será distribuido bajo las condiciones descritas en los apartados 1 y 2 anteriores, en un medio habitualmente utilizado para el intercambio de programas, o acompañarlo con la información que recibiste ofreciendo distribuir el código fuente correspondiente. (Esta opción se permite sólo para distribución no comercial y sólo si usted recibió el programa como código objeto o en formato ejecutable con tal oferta, de acuerdo con el apartado b anterior).

Por código fuente de un trabajo se entiende la forma preferida del trabajo cuando se le hacen modificaciones. Para un trabajo ejecutable, se entiende por código fuente completo todo el código fuente para todos los módulos que contiene, más cualquier fichero asociado de definición de interfaces, más los guiones utilizados para controlar la compilación e instalación del ejecutable. Como excepción especial el código fuente distribuido no necesita incluir nada que sea distribuido normalmente (bien como fuente, bien en forma binaria) con los componentes principales (compilador, kernel y similares) del sistema operativo en el cual funciona el ejecutable, a no ser que el propio componente acompañe al ejecutable.

Si la distribución del ejecutable o del código objeto se hace mediante la oferta acceso para copiarlo de un cierto lugar, entonces se considera la oferta de acceso para copiar el código fuente del mismo lugar como distribución del código fuente, incluso aunque terceras partes no estén forzadas a copiar el fuente junto con el código objeto.

No puede copiar, modificar, sublicenciar o distribuir el Programa excepto como prevé expresamente esta Licencia. Cualquier intento de copiar, modificar sublicenciar o distribuir el Programa de otra forma es inválida, y hará que cesen automáticamente los derechos que te proporciona esta Licencia. En cualquier caso, las partes que hayan recibido copias o derechos de usted bajo esta Licencia no cesarán en sus derechos mientras esas partes continúen cumpliéndola.

No está obligado a aceptar esta licencia, ya que no la ha firmado. Sin embargo, no hay hada más que le proporcione permiso para modificar o distribuir el Programa o sus trabajos derivados. Estas acciones están prohibidas por la ley si no acepta esta Licencia. Por lo tanto, si modifica o distribuye el Programa (o cualquier trabajo basado en el Programa), está indicando que acepta esta Licencia para poder hacerlo, y todos sus términos y condiciones para copiar, distribuir o modificar el Programa o trabajos basados en él.

Cada vez que redistribuya el Programa (o cualquier trabajo basado en el Programa), el receptor recibe automáticamente una licencia del licenciatario original para copiar, distribuir o modificar el Programa, de forma sujeta a estos términos y condiciones. No puede imponer al receptor ninguna restricción más sobre el ejercicio de los derechos aquí garantizados. No es usted responsable de hacer cumplir esta licencia por terceras partes.

Si como consecuencia de una resolución judicial o de una alegación de infracción de patente o por cualquier otra razón (no limitada a asuntos relacionados con patentes) se le imponen condiciones (ya sea por mandato judicial, por acuerdo o por cualquier otra causa) que contradigan las condiciones de esta Licencia, ello no le exime de cumplir las condiciones de esta Licencia. Si no puede realizar distribuciones de forma que se satisfagan simultáneamente sus obligaciones bajo esta licencia y cualquier otra obligación pertinente entonces, como consecuencia, no puede distribuir el Programa de ninguna forma. Por ejemplo, si una patente no permite la redistribución libre de derechos de autor del Programa por parte de todos aquellos que reciban copias directa o indirectamente a través de usted, entonces la

única forma en que podría satisfacer tanto esa condición como esta Licencia sería evitar completamente la distribución del Programa.

Si cualquier porción de este apartado se considera inválida o imposible de cumplir bajo cualquier circunstancia particular ha de cumplirse el resto y la sección por entero ha de cumplirse en cualquier otra circunstancia.

No es el propósito de este apartado inducirle a infringir ninguna reivindicación de patente ni de ningún otro derecho de propiedad o impugnar la validez de ninguna de dichas reivindicaciones. Este apartado tiene el único propósito de proteger la integridad del sistema de distribución de software libre, que se realiza mediante prácticas de licencia pública. Mucha gente ha hecho contribuciones generosas a la gran variedad de software distribuido mediante ese sistema con la confianza de que el sistema se aplicará consistentemente. Será el autor/donante quien decida si quiere distribuir software mediante cualquier otro sistema y una licencia no puede imponer esa elección.

Este apartado pretende dejar completamente claro lo que se cree que es una consecuencia del resto de esta Licencia.

Si la distribución y/o uso de el Programa está restringida en ciertos países, bien por patentes o por interfaces bajo copyright, el tenedor del copyright que coloca este Programa bajo esta Licencia puede añadir una limitación explícita de distribución geográfica excluyendo esos países, de forma que la distribución se permita sólo en o entre los países no excluidos de esta manera. En ese caso, esta Licencia incorporará la limitación como si estuviese escrita en el cuerpo de esta Licencia.

La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia Pública General de tiempo en tiempo. Dichas nuevas versiones serán similares en espíritu a la presente versión, pero pueden ser diferentes en detalles para considerar nuevos problemas o situaciones.

Cada versión recibe un número de versión que la distingue de otras. Si el Programa especifica un número de versión de esta Licencia que se refiere a ella y a «cualquier versión posterior», tienes la opción de seguir los términos y condiciones, bien de esa versión, bien de cualquier versión posterior publicada por la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, puedes escoger cualquier versión publicada por la Free Software Foundation.

Si quiere incorporar partes del Programa en otros programas libres cuyas condiciones de distribución son diferentes, escribe al autor para pedirle permiso. Si el software tiene copyright de la Free Software Foundation, escribe a la Free Software Foundation: algunas veces hacemos excepciones en estos casos. Nuestra decisión estará guiada por el doble objetivo de de preservar la libertad de todos los derivados de nuestro software libre y promover el que se comparta y reutilice el software en general.

AUSENCIA DE GARANTÍA

Como el programa se licencia libre de cargas, no se ofrece ninguna garantía sobre el programa, en todas la extensión permitida por la legislación aplicable. Excepto cuando se indique de otra forma por escrito, los tenedores del copyright y/u otras partes proporcionan el programa «tal cual», sin garantía de ninguna clase, bien expresa o implícita, con inclusión, pero sin limitación a las garantías mercantiles implícitas o a la conveniencia para un propósito particular. Cualquier riesgo referente a la calidad y prestaciones del programa es asumido por usted. Si se probase que el Programa es defectuoso, asume el coste de cualquier servicio, reparación o corrección.

En ningún caso, salvo que lo requiera la legislación aplicable o haya sido acordado por escrito, ningún tenedor del copyright ni ninguna otra parte que modifique y/o redistribuya el Programa según se permite en esta Licencia será responsable ante usted por daños, incluyendo cualquier daño general, especial, incidental o resultante producido por el uso o la imposibilidad de uso del Programa (con inclusión, pero sin limitación a la pérdida de datos o a la generación incorrecta de datos o a pérdidas sufridas por usted o por terceras partes o a un fallo del Programa al funcionar en combinación con cualquier otro programa), incluso si dicho tenedor u otra parte ha sido advertido de la posibilidad de dichos daños.

FIN DE TÉRMINOS Y CONDICIONES